Tor specifications

These specifications describe how Tor works. They t
sufficient detail to allow the reader to implement a
without ever having to read the Tor source code.

They were once a separate set of text files, but in lat
We're in the process of updating these documents t

This is a living document: we are always changing ar
them easier and more accurate, and to improve the
maintained as a set of documents in a gitlab reposit
see their history.

Additionally, the proposals directory holds our desig
documents that have now been merged into the ma
that are still under discussion. Of particular interres
They are the ones that have been implemented, but
documents.

Getting started

There's a lot of material here, and it's not always as
broken it into a few major sections.

For a table of contents, click on the menu icon to th
should probably start by reading the core tor protoc
our protocol works. After that, you should be able t
interest you most. The introduction of each top-leve
introduction.

A short introduction to T

Basic functionality

Tor is a distributed overlay network designed to anc applications such as web browsing, secure shell, anc built of a number of servers, called **relays** (also calle older documentation).

To connect to the network, a client needs to downlo the relays on the network. These directory documer of semi-trusted **directory authority** servers, and ar a client does not yet have a directory, it finds a cach locations, distributed along with its source code.)

> For more information on the directory subsysten specification.

After the client knows the relays on the network, it c one of these relays. A channel is an encrypted reliab between a client and a relay or a relay and a relay, u **cells**. (Under the hood, a channel is just a TLS conne encoding for cells.)

To anonymize its traffic, a client chooses a **path**—a and opens a channel to the first relay on the path (if open to that relay). The client then uses that channe structure called a **circuit**. A circuit is built over a seq relay in the circuit knows its precessor and successo Many circuits can be multiplexed over a single chan

---

> For more information on how paths are selected, first hop on a path, also called a **guard node**, has for more on those, see the guard specification.

Once a circuit exists, the client can use it to exchang relay on the circuit. These relay cells are wrapped in of building the circuit, the client negotiates a separa relay on the circuit. Each relay removes (or adds) a s relay cell before passing it on.

A client uses these relay cells to exchange **relay me**
"relay messages" in turn are used to actually deliver
simplest use case, the client sends a `BEGIN` messag
(called the **exit node**) to create a new session, or **st**
a new TCP connection to a target host. The exit nod
to say that the TCP connection has succeeded. Then
`DATA` messages to represent the contents of the an

> Note that as of 2023, the specifications do not pe
> cells and relay messages. This is because, until re
> relationship between the two: every relay cell hel
> proposal 340 is implemented, we will revise the s
> on this point.

Other kinds of relay messages can be used for more

Using a system called **conflux** a client can build mul
and associate those circuits within a **conflux set**. O
be sent over *either* circuit in the set, depending on c

> For more on conflux, which has been integrated i
> not yet (as of 2023) into this document, see prop

Advanced topics: Onion services and resp

In addition to *initiating* anonymous communications
communications without revealing their identity or l
anonymity, and the mechanism Tor uses to achieve
"hidden services" or "rendezvous services" in some

> For the details on onion services, see the Tor Ren

Advanced topics: Censorship resistence

In some places, Tor is censored. Typically, censors d
addresses of the known Tor relays, and by blocking

To resist this censorship, some Tor relays, called **bri**

directory: their addresses are distributed by other n
published relays from bridges, we sometimes call th

Additionally, Tor clients and bridges can use extensi
transports, that obfuscate their traffic to make it ha

Notation and conventio

These conventions apply, at least in theory, to all of
stated otherwise.

> Remember, our specification documents were or
> files, written separately and edited over the cours
>
> While we are trying (as of 2023) to edit them into
> that these conventions are not now followed unif

MUST, SHOULD, and so on

The key words "MUST", "MUST NOT", "REQUIRED", "
"SHOULD NOT", "RECOMMENDED", "MAY", and "OP
interpreted as described in RFC 2119.

Data lengths

Unless otherwise stated, all lengths are given as a n

> All bytes are 8 bits long. We sometimes call them
> are interchangeable.

When referring to longer lengths, we use SI binary p
and so on) to refer unambiguously to increments of

> If you encounter a reference to "kilobytes", "meg
> infer whether the author intended a decimal (10C
> interpretation. In these cases, it is better to revise

Integer encoding

Unless otherwise stated, all multi-byte integers are e
order.

> For example, 4660 (0x1234), when encoded as a t
> followed by the byte 0x34. ([12 34])
>
> When encoded as a four-byte integer, it is the byt
> 0x12, and the byte 0x34. ([00 00 12 34]).

Binary-as-text encodings

When we refer to "base64", "base32", or "base16", w
RFC 4648, with the following notes:

- In base32, we never insert linefeeds in base32,
  characters.
- In base64, we *sometimes* omit trailing `=` paddi
  linefeeds unless explicitly noted.
- We do not insert any other whitespace, except

Base 16 and base 32 are case-insensitive. Unless oth
should accept any cases, and should produce a sing

We sometimes refer to base16 as "hex" or "hexadec

> Note that as of 2023, in some places, the specs ar
>
> - which base64 strings are multiline
> - which base32 strings and base16 strings sh
>
> This is something we should correct.

Notation

Operations on byte strings

- `A | B` represents the concatenation of two bi

Binary literals

When we write a series of one-byte hexadecimal lite
a multi-byte binary string.

> For example, `[6f 6e 69 6f 6e 20 72 6f 75 74`
> representing the unterminated ASCII string, `onic`

Tor Protocol Specificatio

Note: This document aims to specify Tor as currentl
a little time to become fully up to date. Future versic
protocols, and compatibility is not guaranteed. We r
notes for other obsolete versions of Tor as they bec

This specification is not a design document; most de
more information on why Tor acts as it does, see tor

Preliminaries

Notation and encoding

```
KP -- a public key for an asymmetric ciphe
KS -- a private key for an asymmetric ciph
K  -- a key for a symmetric cipher.
N  -- a "nonce", a random value, usually d
      from other inputs using hashing.
```

H(m) -- a cryptographic hash of m.

Security parameters

Tor uses a stream cipher, a public-key cipher, the Di
function.

KEY_LEN -- the length of the stream cipher's key, in b

```
KP_ENC_LEN -- the length of a public-key e
KP_PAD_LEN -- the number of bytes added in
  encryption, in bytes. (The largest numbe
  in a single public-key operation is ther

DH_LEN -- the number of bytes used to repr
  Diffie-Hellman group.
DH_SEC_LEN -- the number of bytes used in
(x).

HASH_LEN -- the length of the hash functio
```

Message lengths

Some message lengths are fixed in the Tor protocol
message lengths depend on the version of the Tor li
protocol is denoted in this table with `v`.

Name	Length in bytes

Name	Length in bytes	
`PAYLOAD_LEN`	509	T le
`CIRCID_LEN(v)` , `v` < 4	2	T
`CIRCID_LEN(v)` , `v` ≥ 4	4	
`CELL_LEN(v)` , `v` < 4	512	T
`CELL_LEN(v)` , `v` ≥ 4	514	

Note that for all `v` , `CELL_LEN(v) = 1 + CIRCID_LEN`

Ciphers

These are the ciphers we use *unless otherwise specifi*
for new use.

For a stream cipher, unless otherwise specified, we
an IV of all 0 bytes. (We also require AES256.)

For a public-key cipher, unless otherwise specified, \
fixed exponent of 65537. We use OAEP-MGF1 paddi
We leave the optional "Label" parameter unset. (For
ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-

We also use the Curve25519 group and the Ed2551

For Diffie-Hellman, unless otherwise specified, we u
modulus (p), we use the 1024-bit safe prime from rf
representation is:

```
"FFFFFFFFFFFFFFFFC90FDAA22168C234C4C662
"8A67CC74020BBEA63B139B22514A08798E3404
"302B0A6DF25F14374FE1356D6D51C245E485B5
"A637ED6B0BFF5CB6F406B7EDEE386BFB5A899F
"49286651ECE65381FFFFFFFFFFFFFFFF"
```

As an optimization, implementations SHOULD choo
Implementations that do this MUST never use any D
implementations reuse their DH keys?? -RD] [Probal
away with changing DH keys once per second, but tl
me to be comfortable that this is safe. -NM]

For a hash function, unless otherwise specified, we u

KEY_LEN=16. DH_LEN=128; DH_SEC_LEN=40. KP_EN
HASH_LEN=20.

We also use SHA256 and SHA3-256 in some places.

When we refer to "the hash of a public key", unless o
SHA-1 hash of the DER encoding of an ASN.1 RSA pu

All "random" values MUST be generated with a cryp
number generator seeded from a strong entropy so

A bad hybrid encryption algorith

Some specifications will refer to the "legacy hybrid e
a public key KP. It is computed as follows:

```
1. If the length of M is no more than K
   pad and encrypt M with KP.
2. Otherwise, generate a KEY_LEN byte r
   Let M1 = the first KP_ENC_LEN-KP_PAD
   and let M2 = the rest of M.
   Pad and encrypt K|M1 with KP.  Encry
   using the key K.  Concatenate these
```

Note that this "hybrid encryption" approach does no
removing bytes to the end of M. It also allows attack
by the OAEP -- see Goldberg's PET2006 paper for de
new protocols! Also note that as used in Tor's proto

Relay keys and identitie

Every Tor relay has multiple public/private keypairs,
We explain them here.

Each key here has an English name (like "Ed25519 ic
identifier (like `KP_relayid_ed`).

In an identifier, a `KP_` prefix denotes a public key, a
corresponding secret key.

> For historical reasons or reasons of space, you wi
> English names for the same key, or shortened ve
> for a key, however, should always be unique and

For security reasons, **all keys MUST be distinct**: the
used for separate roles within the Tor protocol suite
example, a relay's identity key `KP_relayid_ed` MUS
term signing key `KP_relaysign_ed` .

Identity keys

An **identity key** is a long-lived key that uniquely ide
same set of identity keys are considered to be the s
identity key is considered to have become a differer

An identity keypair's lifetime is the same as the lifeti

Two identity keys are currently defined:

- `KP_relayid_ed` , `KS_relayid_ed` : An "ed2551
  "master identity key".

  This is an Ed25519 key. This key never expires.
  signing the `KP_relaysign_ed` key, which is use
  objects.

- `KP_relayid_rsa` , `KS_relayid_rsa` : A *legacy* "F

  This is an RSA key. It never expires. It is always
  above) its exponent must be 65537. It is used t
  certificates.

Note that because the legacy RSA identity key is so s
secure against an attacker. It exists for legacy purpo
a failure to prove an expected RSA identity is sufficie
authenticate, but a successful proof of an RSA ident
relay's identity. Parties SHOULD NOT use the RSA id

We write `KP_relayid` to refer to a key which is eithe
`KP_relayid_ed`.

Online signing keys

Since Tor's design tries to support keeping the high-
offline, we need a corresponding key that can be us

- `KP_relaysign_ed`, `KS_relaysign_ed`: A mediu
  key is signed by the identity key `KP_relayid_e`
  one should be generated periodically. It signs r
  directory objects, and certificates for other key

When this key is generated, it needs to be signed wi
a certificate of type `IDENTITY_V_SIGNING`. The `KP_r`
anything else.

Circuit extension keys

Each relay has one or more **circuit extension keys**
creating or extending a circuit, a client uses this key
key exchange with the target relay. If the recipient d
the handshake will fail.

Circuit extension keys have moderate lifetimes, on t
published as part of the directory protocol, and rela
while after publishing any new key. (The exact durat
network parameters.)

There are two current kinds of circuit extension keys

- `KP_ntor`, `KS_ntor`: A curve25519 key used for
  extension handshakes.

- `KP_onion_tap`, `KS_onion_tap`: A 1024 bit RSA
  extension handshake.

Channel authentication

There are other keys that relays use to authenticate
handshakes.

These keys are authenticated with other, longer live
often as they like, and SHOULD rotate them frequer

- `KP_legacy_conn_tls`, `KS_legacy_conn_tls`: A
  used to negotiate TLS connections. Tor implem
  often as they like, and SHOULD rotate this key
- `KP_link_ed`, `KS_link_ed`. A short-term Ed255
  authenticate the link handshake: see "Negotiat
  is signed by the "signing" key, and should be re

Channels

A channel is a direct encrypted connection between
and a relay.

Channels are implemented as TLS sessions over TCI

Clients and relays may both open new channels; on
channel.

> Historical note: in some older documentation, ch
> "connections". This proved to be confusing, and v

As part of establishing a channel, the responding re
ownership of one or more **relay identities**, using a
facilities and a series of Tor messages. The initiator
of their own relay identities, if they have any: public
when they initiate a channel, whereas clients and br

Parties should usually reuse an existing channel ratl
the same relay. There are exceptions here; we discu

To open a channel, a client or relay must know the I
(This is sometimes called the "OR address" or "OR p
participant will also know one or more expected ide
reject the channel if the target relay cannot cryptogr
identities.

> (When initiating a connection, if a reasonably live
> expected identity key is taken from that consensu
> otherwise, the expected identity key is the one gi
> fallback list. Finally, when creating a connection b
> cell, the expected identity key is the one given in

Opening a channel is multi-step process:

1. The initiator opens a new TLS session with cert
   relay checks and enforces those properties.
2. Both parties exchange messages over this TLS
   identity or identities.
3. Both parties verify that the identities that they
   expected. (If any expected key is missing or no

the connection.)

Once this is done, the channel is Open, and regular

Channel lifetime

Channels are not permanent. Either side MAY close running on it and an amount of time (KeepalivePeri since the last time any traffic was transmitted over i connection with no circuits open, if it is likely that a connection.

Negotiating and initializ

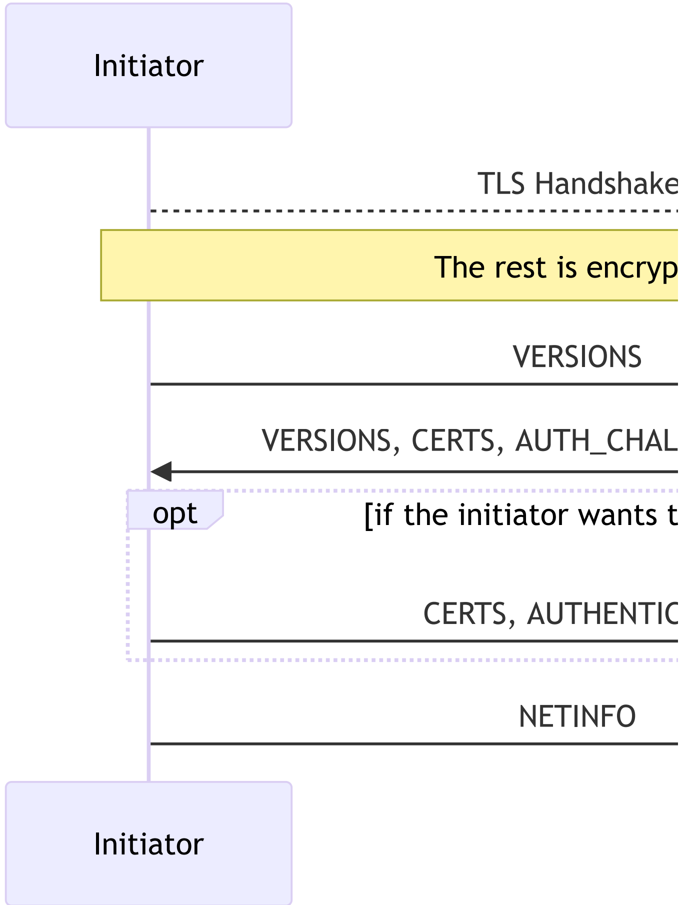
Here we describe the primary TLS behavior used by
channel. There are older versions of these handshal
section.

In brief:

- The initiator starts the handshake by opening
- Both parties send a VERSIONS to negotiate the
- The responder sends a CERTS cell to give the ir learn the responder's identity, an AUTH_CHALI include as part of its answer if it chooses to au establish clock skew and IP addresses.
- The initiator checks whether the CERTS cell is c authenticate.
- If the initiator does not wants to authenticate,
- If the initiator wants to authenticate, it sends a NETINFO cell.

When this handshake is in use, the first cell must be
AUTHORIZE, and no other cell type is allowed to inte
for VPADDING cells.

(The AUTHORIZE cell type is reserved for future use
not specified here.)



The TLS handshake

The initiator must send a ciphersuite list containing
those listed in the obsolete v1 handshake.

> This is trivially achieved by using any modern TLS
> implementations will not need to worry about it.
>
> This requirement distinguishes the current proto
> protocol" or "v3" handshake) from the obsolete v

TLS security considerations

(Standard TLS security guarantees apply; this is not

Implementations SHOULD NOT allow TLS session re
attacks (e.g. the "Triple Handshake" attack from Feb
forward secrecy guarantees.

Implementations SHOULD NOT allow TLS compress
to apply a CRIME-style attack to current Tor directly,

Negotiating versions with VERSI

There are multiple instances of the Tor channel prot

Once the TLS handshake is complete, both parties s
which one they will use.

The payload in a VERSIONS cell is a series of big-end
MUST select as the link protocol version the highest
VERSIONS cell they sent and in the versions cell they
version in common, they cannot communicate and
party MUST close the connection if the versions cell
payload contains an odd number of bytes).

Any VERSIONS cells sent after the first VERSIONS cel
correctly, later VERSIONS cells MUST have a CIRCID_
with the first VERSIONS cell.)

> (The obsolete v1 channel protocol does note VER
> MUST NOT list version 1 in their VERSIONS cells. T
> can only be used after renegotiation; implementa
> their VERSIONS cells unless they have renegotiate

The currently specified Link protocols are:

Version	Descr
1	(Obsolete) The "certs up front" hand
2	(Obsolete) Uses the renegotiation-b variable-length cells.
3	Uses the current ("in-protocol") han advertised.
4	Increases circuit ID width to 4 bytes
5	Adds support for link padding and r

CERTS cells

The CERTS cell describes the keys that a Tor instanc
length cell. Its payload format is:

Field	Description
N	Number of certs ir
N times:	
- CertType	
- CLEN	
- Certificate	

Any extra octets at the end of a CERTS cell MUST be

Relevant certType values are:

certType	Descı
1	Link key certificate certified by RSA
2	RSA1024 Identity certificate, self-si
3	RSA1024 AUTHENTICATE cell link c key.
4	Ed25519 signing key, signed with i
5	TLS link certificate, signed with ed2
6	Ed25519 AUTHENTICATE cell key, s
7	Ed25519 identity, signed with RSA

The certificate format for certificate types 1-3 is DER
format is as documented in a later section

Note that type 7 uses a different format from types

A CERTS cell may have no more than one certificate

To authenticate the responder as having a given Ed2
the initiator MUST check the following.

- The CERTS cell contains exactly one CertType 2
- The CERTS cell contains exactly one CertType 4
- The CERTS cell contains exactly one CertType 5
- The CERTS cell contains exactly one CertType 7
- All X.509 certificates above have validAfter and certificates are expired.
- All certificates are correctly signed.
- The certified key in the Signing->Link certificate certificate that was used to authenticate the TL

- The identity key listed in the ID->Signing cert w
- The Signing->Link cert was signed with the Sigı
- The RSA->Ed25519 cross-certificate certifies th the RSA identity listed in the "ID" certificate.
- The certified key in the ID certificate is a 1024-l
- The RSA ID certificate is correctly self-signed.

To authenticate the responder as having a given RS/ check the following:

- The CERTS cell contains exactly one CertType 1
- The CERTS cell contains exactly one CertType 2
- Both certificates have validAfter and validUntil
- The certified key in the Link certificate matches negotiate the TLS connection.
- The certified key in the ID certificate is a 1024-l
- The certified key in the ID certificate was used
- The link certificate is correctly signed with the l
- The ID certificate is correctly self-signed.

In both cases above, checking these conditions is su initiator is talking to the Tor node with the expected certificate(s).

To authenticate the initiator as having a given Ed25! responder MUST check the following:

- The CERTS cell contains exactly one CertType 2
- The CERTS cell contains exactly one CertType 4
- The CERTS cell contains exactly one CertType 6
- The CERTS cell contains exactly one CertType 7
- All X.509 certificates above have validAfter and certificates are expired.
- All certificates are correctly signed.
- The identity key listed in the ID->Signing cert w
- The Signing->AUTH cert was signed with the Si cert.
- The RSA->Ed25519 cross-certificate certifies th the RSA identity listed in the "ID" certificate.
- The certified key in the ID certificate is a 1024-l
- The RSA ID certificate is correctly self-signed.

To authenticate the initiator as having an RSA identi check the following:

- The CERTS cell contains exactly one CertType 3

- The CERTS cell contains exactly one CertType 2
- Both certificates have validAfter and validUntil
- The certified key in the AUTH certificate is a 10
- The certified key in the ID certificate is a 1024-l
- The certified key in the ID certificate was used
- The auth certificate is correctly signed with the
- The ID certificate is correctly self-signed.

Checking these conditions is NOT sufficient to authe claims; to do so, AUTH_CHALLENGE and AUTHENTIC exchanged.

AUTH_CHALLENGE cells

An AUTH_CHALLENGE cell is a variable-length cell wi

Field	S
Challenge	32 octets
N_Methods	2 octets
Methods	2 * N_Me

It is sent from the responder to the initiator. Initiato the end of the cell. Responders MUST generate ever strong RNG or PRNG.

The Challenge field is a randomly generated string t as part of authenticating. The methods are the auth responder will accept. Only two authentication meth SHA256-TLSSecret and Ed25519-SHA256-RFC570 be

AUTHENTICATE cells

If an initiator wants to authenticate, it responds to tl CERTS cell and an AUTHENTICATE cell. The CERTS ce that instead of sending a CertType 1 (and possibly C certificates, the initiator sends a CertType 3 (and po: RSA/Ed25519 AUTHENTICATE key.

This difference is because we allow any link key type described here will only work for specific key types a and Ed25519-SHA256-RFC570 below.

An AUTHENTICATE cell contains the following:

Field	
AuthType	2 oc
AuthLen	2 oc
Authentication	Autl

Responders MUST ignore extra bytes at the end of a
AuthTypes are 1 and 3, described in the next two se

Initiators MUST NOT send an AUTHENTICATE cell be
presented in the responder's CERTS cell, and auther

Link authentication type 1: RSA-SHA256-T

If AuthType is 1 (meaning "RSA-SHA256-TLSSecret"),
AUTHENTICATE cell contains the following:

- TYPE: The characters "AUTH0001" [8 octets]
- CID: A SHA256 hash of the initiator's RSA1024 i
- SID: A SHA256 hash of the responder's RSA102
- SLOG: A SHA256 hash of all bytes sent from th the negotiation up to and including the AUTH_ cell, the CERTS cell, the AUTH_CHALLENGE cell,
- CLOG: A SHA256 hash of all bytes sent from th the negotiation so far; that is, the VERSIONS ce cells. [32 octets]
- SCERT: A SHA256 hash of the responder's TLS
- TLSSECRETS: A SHA256 HMAC, using the TLS m following: - client_random, as sent in the TLS C the TLS Server Hello - the NUL terminated ASC certification" [32 octets]
- RAND: A 24 byte value, randomly chosen by th gmt_unix_time field, older versions of Tor sent bytes of this field; new implementations shoul
- SIG: A signature of a SHA256 hash of all the pr "Authenticate" key as presented. (As always in Ciphers) [variable length]

To check the AUTHENTICATE cell, a responder check
TLSSECRETS contain their unique correct values as c

signature. The server MUST ignore any extra bytes i
field.

Responders MUST NOT accept this AuthType if the i
Ed25519 identity.

(There is no AuthType 2: It was reserved but never i

Link authentication type 3: Ed25519-SHA2

If AuthType is 3, meaning "Ed25519-SHA256-RFC570
AuthType cell is as below:

Modified values and new fields below are marked w

- TYPE: The characters "AUTH0003" [8 octets]
- CID: A SHA256 hash of the initiator's RSA1024 i
- SID: A SHA256 hash of the responder's RSA102
- CID_ED: The initiator's Ed25519 identity key [3
- SID_ED: The responder's Ed25519 identity key,
- SLOG: A SHA256 hash of all bytes sent from th the negotiation up to and including the AUTH_ cell, the CERTS cell, the AUTH_CHALLENGE cell,
- CLOG: A SHA256 hash of all bytes sent from th the negotiation so far; that is, the VERSIONS ce cells. [32 octets]
- SCERT: A SHA256 hash of the responder's TLS
- TLSSECRETS: The output of an RFC5705 Export as its inputs:
    - The label string "EXPORTER FOR TOR TLS
    - The context value equal to the initiator's
    - The length 32. [32 octets]
- RAND: A 24 byte value, randomly chosen by th
- SIG: A signature of all previous fields using the (as in the cert with CertType 6). [variable lengt

To check the AUTHENTICATE cell, a responder check
TLSSECRETS contain their unique correct values as c
signature. The server MUST ignore any extra bytes i
field.

NETINFO cells

If version 2 or higher is negotiated, each party send
payload is:

Field	Descripti
TIME	Timestamp
OTHERADDR:	Other OR's address
- ATYPE	Address type
- ALEN	Address length
- AVAL	Address value in N
NMYADDR	Number of this OR'
NMYADDR times:	
- ATYPE	Address type
- ALEN	Address length
- AVAL	Address value in N

Recognized address types (ATYPE) are:

ATYPE	Descri
0x04	IPv4
0x06	IPv6

ALEN MUST be 4 when ATYPE is 0x04 (IPv4) and 16
value is wrong for the given ATYPE value, then the p

The timestamp is a big-endian unsigned integer nu
Implementations MUST ignore unexpected bytes at
send "0" as their timestamp, to avoid fingerprinting.

Implementations MAY use the timestamp value to h
Initiators MAY use "other OR's address" to help lear
may be originating from, if they do not know it; and
the current connection as canonical. Implementatio
unconditionally, especially when they come from n
can lie about the time or IP addresses it sees.

Initiators SHOULD use "this OR's address" to make
another OR at its canonical address. (See Canonical

Obsolete channel hands

These handshake variants are no longer in use. Cha
Relays MAY detect and reject them.

> If you are experienced with TLS, you will find som
> strange or obfuscated. Several historical factors l
>
> First, before the development of pluggable transp
> by mimicking the behavior of a web client negotia
> wanted a secure option that was not in common
> option.
>
> Second, prior to the introduction of TLS 1.3, many
> (such as the number and nature of certificates se
> clear, and were easy to distinguish.
>
> Third, prior to the introduction of TLS 1.3, there w
> mechanism that a client could use to declare how
> handshake to proceed. Thus, we wound up using
> ciphersuites to send a signal about which variatic

Version 1, or "certificates up front"

With this obsolete handshake, the responding relay
(`KP_relayid_rsa`), and the initiator also proves own

(If the initiator does not have an RSA identity to prov
afterwards.)

To select this handshake, the initiator starts a TLS ha
other than these:

```
TLS_DHE_RSA_WITH_AES_256_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
```

> Note that because of this list, it is impossible to u
> TLS 1.3.

As part of the TLS handshake, the initiator sends a t

X.509 certificate for its short-term connection public
KP_relayid_rsa , and a second self-signed X.509 ce
The responder sends a similar certificate chain.

Once the TLS handshake is done, both parties valida
they are valid, then the connection is Open, and bot

Version 2, or "renegotiation"

In "renegotiation" (a.k.a. "the v2 handshake"), the co
ciphersuite not in the list above. The initiator sends
sends a single connection certificate in return.

(If the responder sends a certificate chain, the initiat
the v1 handshake.)

Once this initial TLS handshake is complete, the initi
the renegotiation, each party sends a two-certificate
handshake above.

When this handshake is used, both parties immedia
negotiating a link protocol version (which will be 2),
their addresses and timestamps. At that point, the c
cell types are allowed.

Indicating support for the in-pro

When the in-protocol handshake was new, we place
certificate that the responder would send to indicat

Specifically, if at least one of these properties was tr
initiator could be sure that the responder supportec

- The certificate is self-signed
- Some component other than "commonName"
  the certificate.
- The commonName of the subject or issuer of t
  than ".net".
- The certificate's public key modulus is longer t

Otherwise, the initiator would assume that only the

Fixed ciphersuite list

For a long time, clients would advertise a certain "fi
whether they actually supported those ciphers.

That list is:

```
TLS1_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
TLS1_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS1_DHE_RSA_WITH_AES_256_SHA
TLS1_DHE_DSS_WITH_AES_256_SHA
TLS1_ECDH_RSA_WITH_AES_256_CBC_SHA
TLS1_ECDH_ECDSA_WITH_AES_256_CBC_SHA
TLS1_RSA_WITH_AES_256_SHA
TLS1_ECDHE_ECDSA_WITH_RC4_128_SHA
TLS1_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
TLS1_ECDHE_RSA_WITH_RC4_128_SHA
TLS1_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS1_DHE_RSA_WITH_AES_128_SHA
TLS1_DHE_DSS_WITH_AES_128_SHA
TLS1_ECDH_RSA_WITH_RC4_128_SHA
TLS1_ECDH_RSA_WITH_AES_128_CBC_SHA
TLS1_ECDH_ECDSA_WITH_RC4_128_SHA
TLS1_ECDH_ECDSA_WITH_AES_128_CBC_SHA
SSL3_RSA_RC4_128_MD5
SSL3_RSA_RC4_128_SHA
TLS1_RSA_WITH_AES_128_SHA
TLS1_ECDHE_ECDSA_WITH_DES_192_CBC3_SHA
TLS1_ECDHE_RSA_WITH_DES_192_CBC3_SHA
SSL3_EDH_RSA_DES_192_CBC3_SHA
SSL3_EDH_DSS_DES_192_CBC3_SHA
TLS1_ECDH_RSA_WITH_DES_192_CBC3_SHA
TLS1_ECDH_ECDSA_WITH_DES_192_CBC3_SHA
SSL3_RSA_FIPS_WITH_3DES_EDE_CBC_SHA
SSL3_RSA_DES_192_CBC3_SHA
[*] The "extended renegotiation is supp
    not counted when checking the list
```

When encountering this list, a responder would not
mandatory-to-implement TLS_DHE_RSA_WITH_AES_
TLS_DHE_RSA_WITH_AES_128_CBC_SHA, and SSL_D

Clients no longer report ciphers that they do not su

Cells (messages on chan

The basic unit of communication on a Tor channel is

Once a TLS connection is established, the two partie
sent serially, one after another.

Cells may be sent embedded in TLS records of any s
the framing of TLS records MUST NOT leak informat
cells.

Most cells are of fixed length, with the actual length
protocol on the channel. Below we designate the ne

As an exception, `VERSIONS` cells are always sent wit
been negotiated.

A fixed-length cell has this format:

Field	Size in bytes
`CircID`	`CIRCID_LEN(v)`
`Command`	1
`Payload`	`PAYLOAD_LEN`

The value of `CIRCID_LEN` depends on the negotiate

Some cells have variable length; the length of these

A variable-length cell has this format:

Field	Size in bytes	
`CircID`	`CIRCID_LEN(v)`	
`Command`	1	
`Length`	2	
`Payload`	`Length`	

Fixed-length and variable-length cells are distinguisl
Command field:

- Command 7 (`VERSIONS`) is variable-length.
- Every other command less than 128 denotes a
- Every command greater than or equal to 128 c

Historical note:

On version 1 connections, all cells were fixed-leng

On version 2 connections, only the `VERSIONS` cor others were fixed-length.

These link protocols are obsolete, and implemen them.

Interpreting the fields: CircID

The `CircID` field determines which circuit, if any, th not associated with any circuit, its `CircID` is set to (

Note that a CircID is a channel-local identifier.

A single multi-hop circuit will have a different Circ transmit its data.

Interpreting the fields: Comman

The `Command` field of a fixed-length cell holds one o

Value	C	P	Identifier
0	N		`PADDING`
1	Y		`CREATE`
2	Y		`CREATED`
3	Y		`RELAY`
4	Y		`DESTROY`
5	Y		`CREATE_FAST`
6	Y		`CREATED_FAST`
8	N		`NETINFO`
9	Y		`RELAY_EARLY`
10	Y		`CREATE2`

Value	C	P	Identifier
11	Y		`CREATED2`
12	Y	5	`PADDING_NEGOTIATE`

The variable-length `Command` values are:

Value	C	P	Identifier
7	N		`VERSIONS`
128	N		`VPADDING`
129	N		`CERTS`
130	N		`AUTH_CHALLENGE`
131	N		`AUTHENTICATE`
132	N	n/a	`AUTHORIZE`

In the tables above, **C**=Y indicates that a command r
indicates that a command must have a zero CircId. \
version to support a command. Commands with no
least in link protocols 3 and above.

No other command values are allowed. Implementa
command values. Upon receiving an unrecognized c
silently drop the cell and MAY terminate the channe

> Extensibility note:
>
> When we add new cell command types, we defin
> indicate support for that command.
>
> Therefore, implementations can now safely assu
> implementations will never send them an unreco
>
> Historically, before the link protocol was not vers
> drop cells with unrecognized commands, under t
> was sent by a more up-to-date version of Tor.

Interpreting the fields: Payload

The interpretation of Payload depends on the cell's
command descriptions above for more information

Padding fixed-length cell payloa

Often, the amount of information to be sent in a fixe
PAYLOAD_LEN bytes. When this happens, the sender
payload with zero-valued bytes.

Recipients MUST ignore padding bytes.

> RELAY and RELAY_EARLY cel payloads contain enc
> from the point of the view of the channel layer.
>
> The *plaintext* of these cells' contents may be padc
> mechanism and does not interact with channel p

Variable-length cells never have extra space, so ther
Unless otherwise specified, variable-length cells hav

Circuit management

This section describes how circuits are created, and constructed.

CREATE and CREATED ce

Users set up circuits incrementally, one hop at a tim CREATE/CREATE2 cell to the first node, with the first that node responds with a CREATED/CREATED2 cell handshake. To extend a circuit past the first hop, th cell (see EXTEND and EXTENDED cells which instruct CREATE/CREATE2 cell to extend the circuit.

There are two kinds of CREATE and CREATED cells: T and the newer "CREATE2/CREATED2" format. The ne the older one is not.

A CREATE2 cell contains:

Field	Description
`HTYPE`	Client Handshake Type
`HLEN`	Client Handshake Data
`HDATA`	Client Handshake Data

A CREATED2 cell contains:

Field	Description
`HLEN`	Server Handshake Data
`HDATA`	Server Handshake Data

Recognized HTYPEs (handshake types) are:

Value	Descri
0x0000	TAP -- the original Tor handshake; se
0x0001	reserved
0x0002	ntor -- the ntor+curve25519+sha256 handshake
0x0003	ntor-v3 -- ntor extended with extra c

The format of a CREATE cell is one of the following:

Field	Description	
`HDATA`	Client Handshake Data	

or

Field	Description
HTAG	Client Handshake Type Tag
HDATA	Client Handshake Data

The first format is equivalent to a CREATE2 cell with
TAP_C_HANDSHAKE_LEN . The second format is a way
into the old CREATE cell format for migration. See "E
Recognized HTAG values are:

Value
'ntorNTORntorNTOR'

The format of a CREATED cell is:

Field	Description
HDATA	Server Handshake Data

(It's equivalent to a CREATED2 cell with length of TA

As usual with DH, x and y MUST be generated ran

In general, clients SHOULD use CREATE whenever th
CREATE2 otherwise. Clients SHOULD NOT send the s
one with the handshake type tag) to a server directl

Servers always reply to a successful CREATE with a C
with a CREATED2. On failure, a server sends a DESTI

[CREATE2 is handled by Tor 0.2.4.7-alpha and later.]

Choosing circuit IDs in create cel

The CircID for a CREATE/CREATE2 cell is a nonzero in
OR) that sends the CREATE/CREATED2 cell. Dependi
are certain rules for choosing the value of CircID wh
implementations MAY decide to refuse in case of a v
CircIDs are 2 bytes long; in protocol 4 or higher, Circ

In link protocol version 3 or lower, the nodes choose
values based on the ORs' public identity keys, in ord
node has a lower key, it chooses a CircID with an MS
with an MSB of 1. (Public keys are compared numer
public key MAY choose any CircID it wishes, since cli

CREATE/CREATE2 cells.

In link protocol version 4 or higher, whichever node
MSB to 1, and whichever node didn't initiate the cor

The CircID value 0 is specifically reserved for cells th
0 MUST not be used for circuits. No other CircID valu
is reserved.

Existing Tor implementations choose their CircID va
available unused values. To avoid distinguishability,
same. Implementations MAY give up and stop attem
channel, if a certain number of randomly chosen Cir
stops after 64).

EXTEND and EXTENDED cells

To extend an existing circuit, the client sends an EXT
the last node in the circuit.

An EXTEND2 cell's relay payload contains:

Field	Description
`NSPEC`	Number of link spec
`NSPEC` times:	
- `LSTYPE`	Link specifier type
- `LSLEN`	Link specifier length
- `LSPEC`	Link specifier
`HTYPE`	Client Handshake Ty
`HLEN`	Client Handshake D
`HDATA`	Client Handshake D

Link specifiers describe the next node in the circuit a
specifiers are:

Value	Descrip
[00]	TLS-over-TCP, IPv4 address. A four-by ORPort.
[01]	TLS-over-TCP, IPv6 address. A sixteen- ORPort.

Value	Descri
[02]	Legacy identity. A 20-byte SHA1 identi listed.
[03]	Ed25519 identity. A 32-byte Ed25519 i may be listed.

Nodes MUST ignore unrecognized specifiers, and M specifiers other than 'legacy identity' and 'Ed25519 i specifier lists that include multiple instances of eith

For purposes of indistinguishability, implementatior if using them, in this order: [00], [02], [03], [01].

The relay payload for an EXTEND relay cell consists

Field	
Address	4 bytes
Port	2 bytes
Onion skin	`TAP_C_H`
Identity fingerprint	`HASH_LE`

The "legacy identity" and "identity fingerprint" fields ASN1 encoding of the next onion router's identity (s Ciphers") The "Ed25519 identity" field is the Ed2551 Including this key information allows the extending to the correct target OR, and prevents certain man-i

Extending ORs MUST check *all* provided identity key and MUST NOT extend the circuit if the target OR di identity key. If only one identity key is provided, but (from directory information), then the OR SHOULD a

If an extending OR has a channel with a given Ed255 request for that Ed25519 ID and a different RSA ider another connection: it should just fail and DESTROY

The client MAY include multiple IPv4 or IPv6 link spe implementations only consider the first of each type

After checking relay identities, extending ORs gener contents of the EXTEND/EXTEND2 cell. See Creating

The payload of an EXTENDED cell is the same as the

The payload of an EXTENDED2 cell is the same as th

[Support for EXTEND2/EXTENDED2 was added in To

Clients SHOULD use the EXTEND format whenever s
use it whenever the EXTEND cell will be handled by
old to support EXTEND2. In other cases, clients SHO

When generating an EXTEND2 cell, clients SHOULD i
whenever the target has one, and whenever the tar
version "3". (See LinkAuth).

When encoding a non-TAP handshake in an EXTEND
with 'client handshake type tag'.

The "TAP" handshake

This handshake uses Diffie-Hellman in Z_p and RSA t
the client knows are shared only with a particular se
shared with whomever sent the original handshake
fast and not very good. (See Goldberg's "On the Sec
Protocol".)

Define `TAP_C_HANDSHAKE_LEN` as `DH_LEN+KEY_LEN+K`
`TAP_S_HANDSHAKE_LEN` as `DH_LEN+HASH_LEN`.

The payload for a CREATE cell is an 'onion skin', whi
handshake data (also known as `g^x`). This value is e
encryption" algorithm (see "Preliminaries » A bad hy
server's onion key, giving a client handshake:

Field	
KP-encrypted:	
- Padding	`KP_PAD_LEN` byt
- Symmetric key	`KEY_LEN` bytes
- First part of `g^x`	`KP_ENC_LEN-KP_`
Symmetrically encrypted	
- Second part of `g^x`	`DH_LEN-(KP_ENC`

The payload for a CREATED cell, or the relay payloa

Field	
DH data (`g^y`)	`DH_LEN` byte

Field	
Derivative key data (KH)	HASH_LEN by

Once the handshake between the OP and an OR is c
g^xy with ordinary DH. Before computing g^xy , bo
received g^x or g^y value is not degenerate; that i
and strictly less than p-1 where p is the DH modu
complete a handshake with degenerate keys. Imple
"weak" g^x values.

(Discarding degenerate keys is critical for security; if
attacker can substitute the OR's CREATED cell's g^y
g^xy and impersonating the OR. Discarding other k
the private key.)

Once both parties have g^xy , they derive their shar
data' value via the KDF-TOR function.

The "ntor" handshake

This handshake uses a set of DH handshakes to con
client knows are shared only with a particular serve
with whomever sent the original handshake (or with
"curve25519" group and representation as specified
speed records" by D. J. Bernstein.

[The ntor handshake was added in Tor 0.2.4.8-alpha

In this section, define:

```
H(x,t) as HMAC_SHA256 with message x and key
H_LENGTH  = 32.
ID_LENGTH = 20.
G_LENGTH  = 32
PROTOID   = "ntor-curve25519-sha256-1"
t_mac     = PROTOID | ":mac"
t_key     = PROTOID | ":key_extract"
t_verify  = PROTOID | ":verify"
G         = The preferred base point for curv
KEYGEN()  = The curve25519 key generation alg
            a private/public keypair.
m_expand  = PROTOID | ":key_expand"
KEYID(A)  = A
EXP(a, b) = The ECDH algorithm for establish
```

To perform the handshake, the client needs to know and an ntor onion key (a curve25519 public key) for `B`. The client generates a temporary keypair:

```
x,X = KEYGEN()
```

and generates a client-side handshake with content

Field	Value
`NODEID`	Server identity diges
`KEYID`	KEYID(B)
`CLIENT_KP`	X

The server generates a keypair of `y,Y = KEYGEN()`, compute:

```
secret_input = EXP(X,y) | EXP(X,b) | ID | B |
KEY_SEED = H(secret_input, t_key)
verify = H(secret_input, t_verify)
auth_input = verify | ID | B | Y | X | PROTOI
```

The server's handshake reply is:

Field	Value
`SERVER_KP`	`Y`
`AUTH`	`H(auth_input, t_ma`

The client then checks `Y` is in G* [see NOTE below],

```
secret_input = EXP(Y,x) | EXP(B,x) | ID | B |
KEY_SEED = H(secret_input, t_key)
verify = H(secret_input, t_verify)
auth_input = verify | ID | B | Y | X | PROTOI
```

The client verifies that `AUTH == H(auth_input, t_m`

Both parties check that none of the `EXP()` operatio [NOTE: This is an adequate replacement for checkin group is curve25519.]

Both parties now have a shared value for `KEY_SEED` needed for the Tor relay protocol, using the KDF des `m_expand`.

The "ntor-v3" handshake

This handshake extends the ntor handshake to inclu transmitted as part of the handshake. Both the clier data; in both cases, the extra data is encrypted, but secrecy.

To advertise support for this handshake, servers ad version. To select it, clients use the 'ntor-v3' HTYPE v

In this handshake, we define:

```
PROTOID = "ntor3-curve25519-sha3_256-1"
t_msgkdf = PROTOID | ":kdf_phase1"
t_msgmac = PROTOID | ":msg_mac"
t_key_seed = PROTOID | ":key_seed"
t_verify = PROTOID | ":verify"
t_final = PROTOID | ":kdf_final"
t_auth = PROTOID | ":auth_final"

`ENCAP(s)` -- an encapsulation function.  We
as `htonll(len(s)) | s`.  (Note that `len(ENC

`PARTITION(s, n1, n2, n3, ...)` -- a functior
bytestring `s` into chunks of length `n1`, `r
on. Extra data is put into a final chunk.  If
enough, the function fails.

H(s, t) = SHA3_256(ENCAP(t) | s)
MAC(k, msg, t) = SHA3_256(ENCAP(t) | ENCAP(k)
KDF(s, t) = SHAKE_256(ENCAP(t) | s)
ENC(k, m) = AES_256_CTR(k, m)

EXP(pk,sk), KEYGEN: defined as in curve25519

DIGEST_LEN = MAC_LEN = MAC_KEY_LEN = ENC_KEY_

ID_LEN = 32  (representing an ed25519 identit

For any tag "t_foo":
   H_foo(s) = H(s, t_foo)
   MAC_foo(k, msg) = MAC(k, msg, t_foo)
   KDF_foo(s) = KDF(s, t_foo)
```

Other notation is as in the ntor description above.

The client begins by knowing:

```
B, ID -- The curve25519 onion key and Ed2551
         wants to use.
CM -- A message it wants to send as part of t
VER -- An optional shared verification string
```

The client computes:

```
x,X = KEYGEN()
Bx = EXP(B,x)
secret_input_phase1 = Bx | ID | X | B | PROT(
phase1_keys = KDF_msgkdf(secret_input_phase1)
(ENC_K1, MAC_K1) = PARTITION(phase1_keys, EN(
encrypted_msg = ENC(ENC_K1, CM)
msg_mac = MAC_msgmac(MAC_K1, ID | B | X | enc
```

The client then sends, as its CREATE handshake:

Field	Value
NODEID	ID
KEYID	B
CLIENT_PK	X
MSG	encrypted_msg
MAC	msg_mac

The client remembers `x`, `X`, `B`, `ID`, `Bx`, and `msg_m`

When the server receives this handshake, it checks w
looks up the `(b,B)` keypair corresponding to `KEYID`
`NODEID` is wrong, the handshake fails.

Now the relay uses `X=CLIENT_PK` to compute:

```
Xb = EXP(X,b)
secret_input_phase1 = Xb | ID | X | B | PROT(
phase1_keys = KDF_msgkdf(secret_input_phase1)
(ENC_K1, MAC_K1) = PARTITION(phase1_keys, EN(

expected_mac = MAC_msgmac(MAC_K1, ID | B | X
```

If `expected_mac` is not `MAC`, the handshake fails. Ot

```
CM = DEC(MSG, ENC_K1)
```

The relay then checks whether `CM` is well-formed, a
reply that it wants to send as part of the handshake
keypair:

```
y,Y = KEYGEN()
```

and computes the rest of the handshake:

```
Xy = EXP(X,y)
secret_input = Xy | Xb | ID | B | X | Y | PR(
ntor_key_seed = H_key_seed(secret_input)
verify = H_verify(secret_input)

RAW_KEYSTREAM = KDF_final(ntor_key_seed)
(ENC_KEY, KEYSTREAM) = PARTITION(RAW_KEYSTRE/

encrypted_msg = ENC(ENC_KEY, SM)

auth_input = verify | ID | B | Y | X | MAC |
    PROTOID | "Server"
AUTH = H_auth(auth_input)
```

The relay then sends as its CREATED handshake:

Field	Value	
`Y`	`Y`	`PUB_KEY_LE`
`AUTH`	`AUTH`	`DIGEST_LEN`
`MSG`	`encrypted_msg`	`len(SM)` by

Upon receiving this handshake, the client computes

```
Yx = EXP(Y, x)
secret_input = Yx | Bx | ID | B | X | Y | PR(
ntor_key_seed = H_key_seed(secret_input)
verify = H_verify(secret_input)

auth_input = verify | ID | B | Y | X | MAC |
    PROTOID | "Server"
AUTH_expected = H_auth(auth_input)
```

If `AUTH_expected` is equal to `AUTH`, then the handsh
then calculate:

```
RAW_KEYSTREAM = KDF_final(ntor_key_seed)
(ENC_KEY, KEYSTREAM) = PARTITION(RAW_KEYSTRE/

SM = DEC(ENC_KEY, MSG)
```

SM is the message from the relay, and the client use
secrets for the newly created circuit.

Now both parties share the same KEYSTREAM, and
keys.

CREATE_FAST/CREATED_FAST cell

When initializing the first hop of a circuit, the OP has
and negotiated a secret key using TLS. Because of th
OP to perform the public key operations to create a
a CREATE_FAST cell instead of a CREATE cell for the f
CREATED_FAST cell, and the circuit is created.

A CREATE_FAST cell contains:

Field	
Key material (X)	HAS

A CREATED_FAST cell contains:

Field	
Key material (Y)	HASH_LEN
Derivative key data	HASH_LEN

The values of X and Y must be generated randoml

Once both parties have X and Y , they derive their s
data' value via the KDF-TOR function.

The CREATE_FAST handshake is currently deprecate
migration is controlled by the "usecreatefast" netwo
dir-spec.txt.

[Tor 0.3.1.1-alpha and later disable CREATE_FAST by

Additional data in CREATE/CREAT

Some handshakes (currently ntor-v3 defined above)
additional data as part of the handshake. When use
this additional data must have the following format:

Field	
N_EXTENSIONS	one
N_EXTENSIONS times:	
- EXT_FIELD_TYPE	one
- EXT_FIELD_LEN	one

Field	
- EXT_FIELD	EXT

(EXT_FIELD_LEN may be zero, in which case EXT_FI

All parties MUST reject messages that are not well-f

We do not specify specific TYPE semantics here; we specifications.

Parties MUST ignore extensions with EXT_FIELD_TY

Unless otherwise specified in the documentation fo

- Each extension type SHOULD be sent only onc
- Parties MUST ignore any occurrences all occur type after the first such occurrence.
- Extensions SHOULD be sent in numerically asc

(The above extension sorting and multiplicity rules a overridden in the description of individual extensio

Currently supported extensions are:

- 1 -- CC_FIELD_REQUEST [Client to server]

  Contains an empty payload. Signifies that the c congestion control described in proposal 324.

- 2 -- CC_FIELD_RESPONSE [Server to client]

  Indicates that the relay will use the congestion by the client. One byte in length:

  ```
  sendme_inc        [1 byte]
  ```

- 3 -- Subprotocol Request [Client to Server]

  (RESERVED) Tells the endpoint what protocol v 346).

Setting circuit keys

KDF-TOR

This key derivation function is used by the TAP and (
current hidden service protocol. It shouldn't be usec

If the TAP handshake is used to extend a circuit, bot
K0=g^xy, represented as a big-endian unsigned inte

If CREATE_FAST is used, both parties base their key r

From the base key material K0, they compute KEY_L
key data as

K = H(K0 | [00]) | H(K0 | [01]) | H(K0 | [02]) | ...

The first HASH_LEN bytes of K form KH; the next HA
the next HASH_LEN 41-60 form the backward digest
and the final KEY_LEN form Kb. Excess bytes from K

KH is used in the handshake response to demonstra
shared key. Df is used to seed the integrity-checking
from the OP to the OR, and Db seeds the integrity-cl
from the OR to the OP. Kf is used to encrypt the stre
OR, and Kb is used to encrypt the stream of data go

KDF-RFC5869

For newer KDF needs, Tor uses the key derivation fu
instantiated with SHA256. (This is due to a construct
key material is:

K = K_1 | K_2 | K_3 | ...

```
    Where H(x,t) is HMAC_SHA256 with value
      and K_1     = H(m_expand | INT8(1) ,
      and K_(i+1) = H(K_i | m_expand | INT
      and m_expand is an arbitrarily chose
      and INT8(i) is a octet with the valu
```

In RFC5869's vocabulary, this is HKDF-SHA256 with i

IKM == secret_input.

When used in the ntor handshake, the first HASH_LE
the next HASH_LEN form the backward digest Db; th
KEY_LEN form Kb, and the final DIGEST_LEN bytes a
of KH in the hidden service protocol. Excess bytes fr

Creating circuits

When creating a circuit through the network, the cir
following steps:

1. Choose an onion router as an end node (R_N):

   - N MAY be 1 for non-anonymous director
     rendezvous connections.
   - N SHOULD be 3 or more for anonymous
     streams (see "Relay Cells"), others are int
     the Rendezvous Spec).

2. Choose a chain of (N-1) onion routers (R_1...R_
   no router appears in the path twice.

3. If not already connected to the first router in tl
   that router.

4. Choose a circID not already in use on the conn
   chain; send a CREATE/CREATE2 cell along the c
   onion router.

5. Wait until a CREATED/CREATED2 cell is receive
   the forward key Kf_1 and the backward key Kb

6. For each subsequent onion router R (R_2 throu

To extend the circuit by a single onion router R_M, t

1. Create an onion skin, encrypted to R_M's publi

2. Send the onion skin in a relay EXTEND/EXTEND
   and EXTENDED cells" and "Routing relay cells")

3. When a relay EXTENDED/EXTENDED2 cell is rec
   shared keys. The circuit is now extended.

When an onion router receives an EXTEND relay cell
onion router, with the enclosed onion skin as its pay

When an onion router receives an EXTEND2 relay ce
onion router, with the enclosed HLEN, HTYPE, and H
onion router chooses some circID not yet used on tl
routers. (But see section "Choosing circuit IDs in cre

As special cases, if the EXTEND/EXTEND2 cell include

fingerprint of all zeroes, or asks to extend back to th
circuit will fail and be torn down.

Ed25519 identity keys are not required in EXTEND2
accepted. If the extending relay knows the ed25519
also check that key. (See EXTEND and EXTENDED cel

If an EXTEND2 cell contains the ed25519 key of the r
circuit will fail and be torn down.

When an onion router receives a CREATE/CREATE2 c
given connection with the given circID, it drops the c
CREATE/CREATE2 cell, it completes the specified har
CREATED/CREATED2 cell.

Upon receiving a CREATED/CREATED2 cell, an onion
EXTENDED/EXTENDED2 relay cell, and sends that ce
EXTENDED/EXTENDED2 relay cell, the OP can retriev

(As an optimization, OR implementations may delay
traffic allows time to do so without harming networ

Canonical connections

It is possible for an attacker to launch a man-in-the-
telling OR Alice to extend to OR Bob at some addres
attacker cannot read the encrypted traffic, but the a
all bytes sent between Alice and Bob (assuming Alic

To prevent this, when an OR gets an extend request
connection if the ID matches, and ANY of the followi

- The IP matches the requested IP.
- The OR knows that the IP of the connection it's
  listed in the NETINFO cell.

ORs SHOULD NOT check the IPs that are listed in th
makes it easier to covertly impersonate a relay, afte

Tearing down circuits

Circuits are torn down when an unrecoverable erro
streams on a circuit are closed and the circuit's inte

ORs SHOULD also tear down circuits which attempt

- streams with RELAY_BEGIN, or
- rendezvous points with ESTABLISH_RENDEZVC
  Tor be used as a single hop proxy makes exit a
  attractive target for compromise.

ORs MAY use multiple methods to check if they are

```
  * If an OR sees a circuit created with CRE
    the first hop of a circuit.
  * If an OR is the responder, and the init
    * did not authenticate the link, or
    * authenticated with a key that is not
    then the OR is probably the first hop of
    a circuit via a bridge relay).

Circuits may be torn down either complete
```

To tear down a circuit completely, an OR or OP senc
nodes on that circuit, using the appropriate directio

Upon receiving an outgoing DESTROY cell, an OR fre
corresponding circuit. If it's not the end of the circui
circuit to the next OR in the circuit. If the node is the
any associated edge connections (see Calculating th

After a DESTROY cell has been processed, an OR ign
corresponding circuit.

To tear down part of a circuit, the OP may send a RE
OR (Stream ID zero). That OR sends a DESTROY cell
replies to the OP with a RELAY_TRUNCATED cell.

[Note: If an OR receives a TRUNCATE cell and it has
circuit for the next node it will drop them without se
conformant behavior, but it probably won't get fixed
clients SHOULD NOT send a TRUNCATE cell to a noc
if a) they have sent relay cells through that node, an
cells have been sent on yet.]

```
When an unrecoverable error occurs along o
must report it as follows:
  * If possible, send a DESTROY cell to OR
  * If possible, send *either* a DESTROY o
    a RELAY_TRUNCATED cell towards the cl
```

Current versions of Tor do not reuse truncated REL/
receiving a RELAY_TRUNCATED, will send forward a
down the circuit. Because of this, we recommend th
towards the client, not RELAY_TRUNCATED.

```
NOTE:
  In tor versions before 0.4.5.13, 0.4.6.1
  handle an inbound DESTROY by sending the
  message.  Beginning with those versions,
  DESTROY cells in either direction, in or
  intermediary ORs to stop queuing data or
  behavior created queuing pressure on the
```

The payload of a DESTROY and RELAY_TRUNCATED
the reason that the circuit was closed. RELAY_TRUN(
_towards the client, should contain the actual reaso
Reasons in DESTROY cell SHOULD NOT be propagat
potential side channel risk: An OR receiving a DESTR
DESTROYED reason for its next cell. An OP should al
own DESTROY cells.

The error codes are:

```
     0 -- NONE            (No reason given.)
     1 -- PROTOCOL        (Tor protocol viola
     2 -- INTERNAL        (Internal error.)
     3 -- REQUESTED       (A client sent a TR
     4 -- HIBERNATING     (Not currently oper
 bandwidth.)
     5 -- RESOURCELIMIT   (Out of memory, soc
     6 -- CONNECTFAILED   (Unable to reach re
     7 -- OR_IDENTITY     (Connected to relay
                           as expected.)
     8 -- CHANNEL_CLOSED  (The OR connection
                           died.)
     9 -- FINISHED        (The circuit has ex
    10 -- TIMEOUT         (Circuit construct
    11 -- DESTROYED       (The circuit was de
    12 -- NOSUCHSERVICE   (Request for unknow
```

Routing relay cells

Circuit ID Checks

When a node wants to send a RELAY or RELAY_EARL
determines whether the corresponding circuit along
the node drops the cell.

When a node receives a RELAY or RELAY_EARLY cell,
determines whether it has a corresponding circuit a
drops the cell.

Forward Direction

The forward direction is the direction that CREATE/C

Routing from the Origin

When a relay cell is sent from an OP, the OP encrypt
as follows:

```
OP sends relay cell:
   For I=N...1, where N is the destination nc
      Encrypt with Kf_I.
   Transmit the encrypted cell to node 1.
```

Relaying Forward at Onion Routers

When a forward relay cell is received by an OR, it de
cipher, as follows:

```
'Forward' relay cell:
   Use Kf as key; decrypt.
```

The OR then decides whether it recognizes the relay
described in Relay cells. If the OR recognizes the cell
relay cell. Otherwise, it passes the decrypted relay c
continues. If the OR at the end of the circuit encoun

error has occurred: the OR sends a DESTROY cell to

For more information, see Application connections a

Backward Direction

The backward direction is the opposite direction fro

Relaying Backward at Onion Routers

When a backward relay cell is received by an OR, it e
cipher, as follows:

```
'Backward' relay cell:
   Use Kb as key; encrypt.
```

Routing to the Origin

When a relay cell arrives at an OP, the OP decrypts t
follows:

```
OP receives relay cell from node 1:
   For I=1...N, where N is the final node on
       Decrypt with Kb_I.
       If the payload is recognized (see [1])
           The sending node is I.
           Stop and process the payload.
```

[1]: "Relay cells"

Handling relay_early cell

A RELAY_EARLY cell is designed to limit the length ar
receives a RELAY_EARLY cell, and the next node in th
protocol or later, the OR relays the cell as a RELAY_E
relay it as a RELAY cell.

If a node ever receives more than 8 RELAY_EARLY ce
SHOULD close the circuit. If it receives any inbound
circuit immediately.

When speaking v2 of the link protocol or later, client
cells inside RELAY_EARLY cells. Clients SHOULD send
targeted at the first hop of any circuit as RELAY_EAR
conceal the circuit length.

[Starting with Tor 0.2.3.11-alpha, relays should rejec
received in a RELAY_EARLY cell.]

Application connections and stream management

This section describes how clients use RELAY cells to
how use this communication channel to send and re

Relay cells

Within a circuit, the OP and the end node use the cc
end-to-end commands and TCP connections ("Strea
commands can be initiated by either edge; streams

End nodes that accept streams may be:

- exit relays (RELAY_BEGIN, anonymous),
- directory servers (RELAY_BEGIN_DIR, anonymo
- onion services (RELAY_BEGIN, anonymous via a

The payload of each unencrypted RELAY cell consist

Field	
Relay command	1 byte
'Recognized'	2 bytes
StreamID	2 bytes
Digest	4 bytes
Length	2 bytes
Data	Length bytes
Padding	PAYLOAD_LE

The relay commands are:

Command	Identifier
1	RELAY_BEGIN
2	RELAY_DATA
3	RELAY_END
4	RELAY_CONNECTED
5	RELAY_SENDME
6	RELAY_EXTEND
7	RELAY_EXTENDED
8	RELAY_TRUNCATE
9	RELAY_TRUNCATED
10	RELAY_DROP

Command	Identifier
11	RELAY_RESOLVE
12	RELAY_RESOLVED
13	RELAY_BEGIN_DIR
14	RELAY_EXTEND2
15	RELAY_EXTENDED2
16..18	Reserved for UDP; Not yet in use, see prop339.
19..22	Reserved for Conflux, see prop329.
32..40	Used for hidden services; see the rendezvous spec.
41..42	Used for circuit padding; see "Circuit-level padding" in the padding spec.
43	XON (See Sec 4 of prop324)
44	XOFF (See Sec 4 of prop324)

Commands labelled as "forward" must only be sent
Commands labelled as "backward" must only be ser
the originator. Commands marked as either can be
nodes.

The 'recognized' field is used as a simple indication t
optimization to avoid calculating expensive digests f
unencrypted 'recognized' MUST be set to zero.

When receiving and decrypting cells the 'recognized
endpoint that the cell is destined for. For cells that v
will usually be nonzero, but will accidentally be zero

When handling a relay cell, if the 'recognized' in fielc
the 'digest' field is computed as the first four bytes c
that have been destined for this hop of the circuit or
circuit, seeded from Df or Db respectively (obtained
this RELAY cell's entire payload (taken with the diges
digests *do* include the padding bytes at the end of th
the digest is correct, the cell is considered "recogniz
(see Routing relay cells).

(The digest does not include any bytes from relay ce
of the circuit. That is, it does not include forwarded
but the digest does not match, the running digest at
and the cell should be forwarded on.)

All RELAY cells pertaining to the same tunneled stre
StreamIDs are chosen arbitrarily by the OP. No strea
Rather, RELAY cells that affect the entire circuit rath
StreamID of zero -- they are marked in the table abo
cells are marked as "sometimes control" because th
depending on their purpose -- see Flow control.)

The 'Length' field of a relay cell contains the number
contain real payload data. The remainder of the une
padding bytes. Implementations handle padding by
do padding bytes for other cell types; see Cell Packe

The 'Padding' field is used to make relay cell content
attacks (see proposal 289 for rationale). Implementa
zero-valued bytes, followed by as many random byt
bytes for padding, then they should all be filled with

Implementations MUST NOT rely on the contents of

If the RELAY cell is recognized but the relay comman
dropped and ignored. Its contents still count with re
windows, though.

Calculating the 'Digest' field

The 'Digest' field itself serves the purpose to check i
is, all onion layers have been removed. Having a sin
sufficient, as outlined above.

When ENCRYPTING a RELAY cell, an implementation

```
# Encode the cell in binary (recognized and c
tmp = cmd + [0, 0] + stream_id + [0, 0, 0, 0]

# Update the digest with the encoded data
digest_state = hash_update(digest_state, tmp)
digest = hash_calculate(digest_state)

# The encoded data is the same as above with
# zero anymore
encoded = cmd + [0, 0] + stream_id + digest[0
          padding

# Now we can encrypt the cell by adding the c
```

When DECRYPTING a RELAY cell, an implementation

```
decrypted = decrypt(cell)

# Replace the digest field in decrypted by ze
tmp = decrypted[0..5] + [0, 0, 0, 0] + decryp

# Update the digest field with the decrypted
# set to zero
digest_state = hash_update(digest_state, tmp)
digest = hash_calculate(digest_state)

if digest[0..4] == decrypted[5..9]
  # The cell has been fully decrypted ...
```

The caveat itself is that only the binary data with the
taken into account when calculating the running dig
digest field set to its actual value) are not taken into

Opening streams and tr

Opening a new stream: The begi handshake

To open a new anonymized TCP connection, the OP
that may be able to connect to the destination addr
yet used on that circuit, and constructs a RELAY_BE
address and port of the destination host. The paylo

```
        ADDRPORT [nul-terminated string]
        FLAGS    [4 bytes, optional]

    ADDRPORT is made of ADDRESS | ':' | PORT |
```

where ADDRESS can be a DNS hostname, or an IPv4
IPv6 address surrounded by square brackets; and w
between 1 and 65535, inclusive.

The ADDRPORT string SHOULD be sent in lower cas
Implementations MUST accept strings in any case.

The FLAGS value has one or more of the following b
32-bit value, and "bit 32" is the MSB. (Remember th
the MSB of a 4-byte value is the MSB of the first byt
LSB of its last byte.)

If FLAGS is absent, its value is 0. Whenever 0 would
from the message payload.

```
      bit   meaning
       1 -- IPv6 okay.  We support learning ab
            connecting to IPv6 addresses.
       2 -- IPv4 not okay.  We don't want to l
            or connect to them.
       3 -- IPv6 preferred.  If there are both
            we want to connect to the IPv6 one
            to the IPv4 address.)
       4..32 -- Reserved. Current clients MUST
            MUST ignore them.
```

Upon receiving this cell, the exit node resolves the a
new TCP connection to the target port. If the addres
can't be established, the exit node replies with a RE

Otherwise, the exit node replies with a RELAY_CONN
of the following formats:

```
        The IPv4 address to which the connect
        A number of seconds (TTL) for which th
octets]

    or

        Four zero-valued octets [4 octets]
        An address type (6)     [1 octet]
        The IPv6 address to which the connect
        A number of seconds (TTL) for which th
octets]
```

[Tor exit nodes before 0.1.2.0 set the TTL field to a fi
to the last value seen from a DNS server, and expire
interval. This prevents certain attacks.]

Transmitting data

Once a connection has been established, the OP an
RELAY_DATA cells, and upon receiving such cells, ec
corresponding TCP stream.

If the exit node does not support optimistic data (i.e
alpha), then the OP MUST wait for a RELAY_CONNEC
the exit node supports optimistic data (i.e. its versio
then the OP MAY send RELAY_DATA cells immediate
(and before receiving either a RELAY_CONNECTED o

RELAY_DATA cells sent to unrecognized streams are
optimistic data, then RELAY_DATA cells it receives o
RELAY_BEGIN but have not yet been replied to with
are queued. If the stream creation succeeds with a R
processed immediately afterwards; if the stream cre
contents of the queue are deleted.

Relay RELAY_DROP cells are long-range dummies; u
OP must drop it.

Opening a directory stream

If a Tor relay is a directory server, it should respond
received a BEGIN cell requesting a connection to its
ignore exit policy, since the stream is local to the To

Directory servers may be:

- authoritative directories (RELAY_BEGIN_DIR, us
- bridge authoritative directories (RELAY_BEGIN_
- directory mirrors (RELAY_BEGIN_DIR, usually n
- onion service directories (RELAY_BEGIN_DIR, a

If the Tor relay is not running a directory service, it s
REASON_NOTDIRECTORY RELAY_END cell.

Clients MUST generate an all-zero payload for RELAY
ignore the payload.

In response to a RELAY_BEGIN_DIR cell, relays respo
cell on success, or a RELAY_END cell on failure. They
all-zero payload, and clients MUST ignore the paylo

[RELAY_BEGIN_DIR was not supported before Tor 0.
send it to routers running earlier versions of Tor.]

Closing streams

When an anonymized TCP connection is closed, or a
stream, it sends a 'RELAY_END' cell along the circuit
connection immediately. If an edge node receives a
closes the TCP connection completely, and sends no
stream.

The payload of a RELAY_END cell begins with a singl
stream is closing. For some reasons, it contains add
reason.) The values are:

```
 1 -- REASON_MISC           (catch-all
 2 -- REASON_RESOLVEFAILED  (couldn't l
 3 -- REASON_CONNECTREFUSED (remote hos
 4 -- REASON_EXITPOLICY     (OR refuses
 5 -- REASON_DESTROY        (Circuit is
 6 -- REASON_DONE           (Anonymizec
 7 -- REASON_TIMEOUT        (Connectior
                             while conr
 8 -- REASON_NOROUTE        (Routing er
                             contact de
 9 -- REASON_HIBERNATING    (OR is temp
10 -- REASON_INTERNAL       (Internal e
11 -- REASON_RESOURCELIMIT  (OR has no
12 -- REASON_CONNRESET      (Connectior
13 -- REASON_TORPROTOCOL    (Sent when
                             Tor protoc
14 -- REASON_NOTDIRECTORY   (Client ser
                             non-direct

[*] Older versions of Tor also send this r
    reset.
```

OPs and ORs MUST accept reasons not on the abov
provide more fine-grained reasons.

For most reasons, the format of RELAY_END is:

Reason [1 byte]

For REASON_EXITPOLICY, the format of RELAY_END

```
Reason                      [1 byte]
IPv4 or IPv6 address        [4 bytes or
TTL                         [4 bytes]
```

(If the TTL is absent, it should be treated as if it were
is the wrong length, the RELAY_END message shoul

Tors SHOULD NOT send any reason except REASON
originated.

Implementations SHOULD accept empty RELAY_END
specified REASON_MISC.

Upon receiving a RELAY_END cell, the recipient may
arrive on that stream, and can treat such cells as a p

After sending a RELAY_END cell, the sender needs to
that cell. In the meantime, the sender SHOULD rem
(CONNECTED, SENDME, DATA) that it would have ac
kill the circuit if it receives more than permitted.

--- [The rest of this section describes unimplemente

Because TCP connections can be half-open, we follo
ACK/ACK protocol to close streams.

An exit (or onion service) connection can have a TCP
'OPEN', 'DONE_PACKAGING', and 'DONE_DELIVERIN
transitions, we treat 'CLOSED' as a fourth state, alth
in fact, tracked by the onion router.

A stream begins in the 'OPEN' state. Upon receiving
connection, the edge node sends a 'RELAY_FIN' cell a
to 'DONE_PACKAGING'. Upon receiving a 'RELAY_FIN
the corresponding TCP connection (e.g., by calling sh
state to 'DONE_DELIVERING'.

When a stream in already in 'DONE_DELIVERING' rec
'RELAY_FIN' along the circuit, and changes its state t
'DONE_PACKAGING' receives a 'RELAY_FIN' cell, it se
'CLOSED'.

If an edge node encounters an error on any stream,
possible) and closes the stream immediately.

Remote hostname looku

To find the address associated with a hostname, the
containing the hostname to be resolved with a NUL
lookup, the OP sends a RELAY_RESOLVE cell contain
replies with a RELAY_RESOLVED cell containing any r
the form:

```
        Type   (1 octet)
        Length (1 octet)
        Value  (variable-width)
        TTL    (4 octets)
    "Length" is the length of the Value field.
    "Type" is one of:

       0x00 -- Hostname
       0x04 -- IPv4 address
       0x06 -- IPv6 address
       0xF0 -- Error, transient
       0xF1 -- Error, nontransient

     If any answer has a type of 'Error', then
     given.

     The 'Value' field encodes the answer:
         IP addresses are given in network orc
         Hostnames are given in standard DNS c
           and not NUL-terminated.
         The content of Errors is currently ig
           set it to the string "Error resolv
           terminating NUL. Implementations MU

     For backward compatibility, if there are
     must be given as the first answer.

     The RELAY_RESOLVE cell must use a nonzerc
     corresponding RELAY_RESOLVED cell must us
     is actually created by the OR when resolv
```

Flow control

Link throttling

Each client or relay should do appropriate bandwidt

Communicants rely on TCP's default flow control to

The mainline Tor implementation uses token bucke
the rate limiting.

Since 0.2.0.x, Tor has let the user specify an additior
traffic, so people can deploy a Tor relay with strict ra
as a client. To avoid partitioning concerns we combi
OR connection, and keep track of the last time we re
relayed) cell. If it's been less than N seconds (curren
connection high priority, else we give the whole con
priority to reads and writes for connections that are
proposal 111 for details.

Link padding

Link padding can be created by sending PADDING o
connection; relay cells of type "DROP" can be used f
of PADDING, VPADDING, or DROP cells are filled wit
format.

If the link protocol is version 5 or higher, link level pa
spec.txt. On these connections, clients may negotiat
CELL_PADDING_NEGOTIATE command whose forma

```
Version          [1 byte]
Command          [1 byte]
ito_low_ms       [2 bytes]
ito_high_ms      [2 bytes]
```

Currently, only version 0 of this cell is defined. In it, t
padding) or 2 (start padding). For the start padding c
specifying a low and a high range bounds for randor
specified as unsigned integer values in milliseconds.
lower than the current consensus parameter value f

ito_high_ms field should not be lower than ito_low_r
range value, they clamp it so that it is in-range.)

For the stop padding command, the timeout fields s
distinguishability) and ignored by the recipient.

For more details on padding behavior, see padding-

Circuit-level flow control

To control a circuit's bandwidth usage, each OR kee
of how many RELAY_DATA cells it is allowed to origir

These two windows are respectively named: the pac
transmission) and the deliver window (delivered for

Because of our leaky-pipe topology, every relay on t
the OP has a pair of windows for every relay on the
to relayed cells, however, and a relay that is never u
its window or cause the client to decrement a windc

Each 'window' value is initially set based on the cons
directory (see dir-spec.txt), or to 1000 data cells if nc
direction, cells that are not RELAY_DATA cells do not

An OR or OP (depending on the stream direction) se
that it is willing to receive more cells when its delive
increment (100). For example, if the window started
RELAY_SENDME when it reaches 900.

When an OR or OP receives a RELAY_SENDME, it inc
value of 100 (circuit window increment) and proceec
RELAY_DATA cells.

If a package window reaches 0, the OR or OP stops r
streams on the corresponding circuit, and sends no
receiving a RELAY_SENDME cell.

If a deliver window goes below 0, the circuit should
Starting with tor-0.4.1.1-alpha, authenticated SENDN
below). This means that both the OR and OP need t
cell that precedes (triggers) a RELAY_SENDME. This c
gets to a multiple of the circuit window increment (1

When the RELAY_SENDME version 1 arrives, it will co
one remembered. This represents a proof that the e
cells. On failure to match, the circuit should be torn

To ensure unpredictability, random bytes should be
cell within one increment window. In other word, ev
bytes should be introduced in at least one cell.

SENDME Cell Format

A circuit-level RELAY_SENDME cell always has its Stre

An OR or OP must obey these two consensus param
to emit and accept.

```
'sendme_emit_min_version': Minimum vers
'sendme_accept_min_version': Minimum ve
```

If a RELAY_SENDME version is received that is below
circuit should be closed.

The RELAY_SENDME payload contains the following:

```
VERSION     [1 byte]
DATA_LEN    [2 bytes]
DATA        [DATA_LEN bytes]
```

The VERSION tells us what is expected in the DATA s
to handle it. The recognized values are:

0x00: The rest of the payload should be ignored.

0x01: Authenticated SENDME. The DATA section MU

DIGEST [20 bytes]

```
If the DATA_LEN value is less than 2
dropped and the circuit closed. If t
then the first 20 bytes should be re

The DIGEST is the rolling digest val
immediately preceded (triggered) thi
matched on the other side from the p
must remember.

(Note that if the digest in use has
bytes—as is the case for the hop of
circuit created by the hs_ntor hands
to 20 bytes here.)
```

If the VERSION is unrecognized or below the minimu
consensus), the circuit should be torn down.

Stream-level flow control

Edge nodes use RELAY_SENDME cells to implement
connections across circuits. Similarly to circuit-level
window of cells (500) per stream, and increment the
receiving a RELAY_SENDME cell. Edge nodes initiate
the window is <= 450, and b) there are less than ten
at that edge.

Stream-level RELAY_SENDME cells are distinguished
are still empty; the body still SHOULD be ignored.

Subprotocol versioning

This section specifies the Tor subprotocol versionin types with their current version numbers. Any new v this section.

The dir-spec.txt details how those versions are enco descriptor and the "recommended-relay-protocols", "recommended-client-protocols" and "required-clie vote/consensus format.

Here are the rules a relay and client should follow w the consensus:

- When a relay lacks a protocol listed in recomm its operator that the relay is obsolete.
- When a relay lacks a protocol listed in require operator as above. If the consensus is newer t released or scheduled for release, it must not
- When a client lacks a protocol listed in recomm warn the user that the client is obsolete.
- When a client lacks a protocol listed in require user as above. If the consensus is newer than released, it must not connect to the network. T shutdown" mechanism for zombie clients.
- If a client or relay has a cached consensus telli and it does not implement that protocol, it SH consensus.

Software release dates SHOULD be automatically u to prevent forgetting to move them forward. Softwa adjusted by maintainers if necessary.

Starting in version 0.2.9.4-alpha, the initial required Recommend and Require are:

```
Cons=1-2 Desc=1-2 DirCache=1 HSDir=1 HSIntro=
LinkAuth=1 Microdesc=1-2 Relay=2
```

For relays we will Require:

```
Cons=1 Desc=1 DirCache=1 HSDir=1 HSIntro=3 HS
LinkAuth=1 Microdesc=1 Relay=1-2
```

For relays, we will additionally Recommend all protc

"Link"

The "link" protocols are those used by clients and re connections and to handle cells on OR connections. correspond 1:1 to those versions.

Two Tor instances can make a connection to each o protocol in common.

The current "link" versions are: "1" through "5". See cells for more information. All current Tor versions s alpha and on support "1-4"; versions from 0.3.1.1-al we will drop "1" and "2".

"LinkAuth"

LinkAuth protocols correspond to varieties of Authe protocols.

Current versions are:

- "1" is the RSA link authentication described in SHA256-TLSSecret.
- "2" is unused, and reserved by proposal 244.
- "3" is the ed25519 link authentication describe Ed25519-SHA256-RFC5705.

"Relay"

The "relay" protocols are those used to handle CREA handle the various RELAY cell types received after a cells used to manage introduction and rendezvous p and "HSRend" protocols respectively.)

Current versions are as follows.

- "1" -- supports the TAP key exchange, with all f
CREATE and CREATED and CREATE_FAST and C
EXTENDED.

- "2" -- supports the ntor key exchange, and all f
support for CREATE2 and CREATED2 and EXTE

  Relay=2 has limited IPv6 support:

  - Clients might not include IPv6 ORPorts in
  - Relays (and bridges) might not initiate IPv
EXTEND2 cells containing IPv6 ORPorts, e
IPv6 ORPort.

  However, relays support accepting inbound co
they might extend circuits via authenticated IP

- "3" -- relays support extending over IPv6 conne
cell containing an IPv6 ORPort.

  Bridges might not extend over IPv6, because th

  A successful IPv6 extend requires:

  - Relay subprotocol version 3 (or later) on t
  - an IPv6 ORPort on the extending relay,
  - an IPv6 ORPort for the accepting relay in
  - an IPv6 ORPort on the accepting relay. (B
have different views of the network, thes
path is selected. Extending relays should
before attempting the extend.)

  When relays receive an EXTEND2 cell containir
and there is no existing authenticated connect
extending relay may choose between IPv4 and
might not try the other address, if the first con

  As is the case with other subprotocol versions,
requires support for this protocol version, rega

  In particular:

  - relays without an IPv6 ORPort, and
  - tor instances that are not relays, have the
their configuration:
  - advertise support for "Relay=3" in their d

directory authority), and
- react to consensuses recommending or r

This subprotocol version is described in propo 0.4.5.1-alpha.

- "4" -- support the ntorv3 (version 3) key exchar This adds a new CREATE2 cell type. See propos for more details.

- "5" -- [RESERVED] support the ntorv3 subproto allowing a client to request what features to be

"HSIntro"

The "HSIntro" protocol handles introduction points.

- "3" -- supports authentication as of proposal 1
- "4" -- support ed25519 authentication keys wh part of proposal 224 in Tor 0.3.0.4-alpha.
- "5" -- support ESTABLISH_INTRO cell DoS parar version 3 only in Tor 0.4.2.1-alpha.

"HSRend"

The "HSRend" protocol handles rendezvous points.

- "1" -- supports all features in Tor 0.0.6.
- "2" -- supports RENDEZVOUS2 cells of arbitrary of cookie in Tor 0.2.9.1-alpha.

"HSDir"

The "HSDir" protocols are the set of hidden service to, understood by, and downloaded from a tor relay fetch them.

- "1" -- supports all features in Tor 0.2.0.10-alph

- "2" -- support ed25519 blinded keys request w as part of proposal 224 in Tor 0.3.0.4-alpha.

"DirCache"

The "DirCache" protocols are the set of documents a directory cache via BEGIN_DIR, and the set of URLs a URLs for hidden service objects.)

- "1" -- supports all features in Tor 0.2.4.19.
- "2" -- adds support for consensus diffs in Tor 0

"Desc"

Describes features present or absent in descriptors.

Most features in descriptors don't require a "Desc" someday be required. For example, someday clients identities.

- "1" -- supports all features in Tor 0.2.4.19.
- "2" -- cross-signing with onion-keys, signing wit

"Microdesc"

Describes features present or absent in microdescri

Most features in descriptors don't require a "MicroD someday be required. These correspond more or le

- "1" -- consensus methods 9 through 20.
- "2" -- consensus method 21 (adds ed25519 key

"Cons"

Describes features present or absent in consensus

Most features in consensus documents don't requir
need to someday be required.

These correspond more or less with consensus met

- "1" -- consensus methods 9 through 20.
- "2" -- consensus method 21 (adds ed25519 key

"Padding"

Describes the padding capabilities of the relay.

- "1" -- [DEFUNCT] Relay supports circuit-level pa
  used as it was also enabled in relays that don't
  Advertised by Tor versions from tor-0.4.0.1-alp
  tor-0.4.1.4-rc.
- "2" -- Relay supports the HS circuit setup paddi
  Advertised by Tor versions from tor-0.4.1.5 and

"FlowCtrl"

Describes the flow control protocol at the circuit and
advertised, tor supports the unauthenticated flow c

- "1" -- supports authenticated circuit level SEND
  alpha.
- "2" -- supports congestion control by the Exits
  and algorithm. See proposal 324 for more deta

"Conflux"

Describes the communications mechanisms used to
split traffic across multiple paths.

> TODO: This is not yet described here. For details s

"Datagram"

Describes the UDP protocol capabilities of a relay.

- "1" -- [RESERVED] supports UDP by an Exit as ir
CONNECTED_UDP and DATAGRAM. See propo
advertised, reserved)

Certificates in Tor

This document describes a certificate formats that T
certificates, and discusses how that format is labele

This format is not the only certificate format that To
authorities use for their signing keys, see "Creating I

Additionally, Tor uses TLS, which depends on X.509

> The certificates in this document were first introc
> first supported by Tor in Tor version 0.2.7.2-alpha

Signing

All signatures here, unless otherwise specified, are c

In order to future-proof the format, before signing a
prefixed with a personalization string, which will be

Document formats

X.509 certificates

Describing this format is out of scope for the Tor sp

Ed25519 Certificates

When generating a signing key, we also generate a c
for this certificate is:

Field	Size	
VERSION	1	The
CERT_TYPE	1	Purp
EXPIRATION_DATE	4	Whe
CERT_KEY_TYPE	1	Type

Field	Size	
`CERTIFIED_KEY`	32	Cert
`N_EXTENSIONS`	1	Nun
`N_EXTENSIONS` times:		
- `ExtLen`	2	Len
- `ExtType`	1	Typ
- `ExtFlags`	1	Con
- `ExtData`	`ExtLen`	Enc
SIGNATURE	64	Sign

The `VERSION` field holds the value `[01]`.

The `CERT_TYPE` field holds a value depending on th types".)

The `CERTIFIED_KEY` field is an Ed25519 public key if of some other key type depending on the value of C key types".)

The `EXPIRATION_DATE` is a date, given in **hours** sinc certificate isn't valid.

(A four-byte date here will work fine until 10136 A

The `ExtFlags` field holds flags. Only one flag is curr

- **1**: `AFFECTS_VALIDATION`. If this flag is present, the certificate is valid; implementations MUST unless they recognize the `ExtType` and accept

The interpretation of `ExtBody` depends on the `Ext` extensions" below.

It is an error for an extension to be truncated; such

Before processing any certificate, parties SHOULD k signed by, and then check the signature.

The signature is created by signing all the fields in th `SIGNATURE`.

Recognized extensions

Signed-with-ed25519-key extension [type 04]

In several places, it's desirable to bundle the signing
so with this extension.

With this extension:

- `ExtLen` is 32.
- `ExtData is a 32-byte Ed25519 public key.

When this extension is present, it MUST match the k

RSA→Ed25519 cross-certificate

In one place, we have a binary certificate that signs
bit RSA key. Its format is:

Field	Size	
`ED25519_KEY`	32	The
`EXPIRATION_DATE`	4	Wh
`SIGLEN`	1	Len
`SIGNATURE`	`SIGLEN`	RSA

Just as with the Ed25519 certificates above, the `EXP`
hours since the epoch.

As elsewhere, the RSA signature is generated using
algorithm OIDs omitted.

The signature is computed on the SHA256 hash of
the string `"Tor TLS RSA/Ed25519 cross-certifica`
and `FIELDS` is all other fields in the certificate (othe

Certificate types (CERT_TYPE fiel

This table shows values of the `CERT_TYPE` field in Ed
field used in a `CERTS` cell during channel negotiatio

You might ned to scroll this table to view it all.

We'll try to fix this once we have a better grip on

Type	Mnemonic	Format
[01]	TLS_LINK_X509	X.509
[02]	RSA_ID_X509	X.509
[03]	LINK_AUTH_X509	X.509
[04]	IDENTITY_V_SIGNING	Ed
[05]	SIGNING_V_TLS_CERT	Ed
[06]	SIGNING_V_LINK_AUTH	Ed
[07]	RSA_ID_V_IDENTITY	Rsa
[08]	BLINDED_ID_V_SIGNING	Ed
[09]	HS_IP_V_SIGNING	Ed
[0A]	NTOR_CC_IDENTITY	Ed
[0B]	HS_IP_CC_SIGNING	Ed

Note 1: The certificate types `[09] HS_IP_V_SIGNING` implemented incorrectly, and now cannot be chang keys, as implemented, are given in the table. They w of this order.

List of extension types

- [04] - signed-with-ed25519-key

List of signature prefixes

We describe various documents as being signed wit

"Tor router descriptor signature v1" (see dir-spec.txt

List of certified key types (CERT_

- [01] : ed25519 key
- [02] : SHA256 hash of an RSA key. (Not currer
- [03] : SHA256 hash of an X.509 certificate. (Us

> (NOTE: Up till 0.4.5.1-alpha, all versions of Tor hav
> types of certified key. Implementations SHOULD
> the actual key type from the CERT_TYPE field.

Tor directory protocol, v

This directory protocol is used by Tor version 0.2.0.x
for information on the protocol used up to the 0.1.0
information on the protocol used by the 0.1.1.x and

This document merges and supersedes the followin

- 101 Voting on the Tor Directory System
- 103 Splitting identity key from regularly used s
- 104 Long and Short Router Descriptors

XXX timeline XXX fill in XXXXs

The key words "MUST", "MUST NOT", "REQUIRED", "
"SHOULD NOT", "RECOMMENDED", "MAY", and "OP
interpreted as described in RFC 2119.

History

The earliest versions of Onion Routing shipped with
When the set of routers changed, users needed to f

The Version 1 Directory protocol

Early versions of Tor (0.0.2) introduced "Directory au
"directory" documents containing a list of signed "se
summary of the status of each router. Thus, clients
the state of the network automatically, and be certa
attested by a trusted directory authority.

Later versions (0.0.8) added directory caches, which
authorities and serve them to clients. Non-caches fe
fetching from the authorities, thus distributing banc

Also added during the version 1 directory protocol v
documents that listed only the up/down status of th
a complete list of all the descriptors. Clients and cac
more frequently than they would fetch full directori

The Version 2 Directory Protocol

During the Tor 0.1.1.x series, Tor revised its handlin
address two major problems:

```
* Directories had grown quite large (ov
  downloads consisted mainly of server
  already had.

* Every directory authority was a trust
  directory authority lied, it could ma
  an arbitrarily distorted view of the
  trusted the most recent signed docume
  adding more authorities would make th
  more.
```

To address these, we extended the directory protoc
signed "network status" documents. Each network s
network: a hash of its identity key, a hash of its mos
what the authority believed about its status. Clients
network status documents in turn, and believe state
attested to by more than half of the authorities.

Instead of downloading all server descriptors at onc
descriptors that they did not have. Descriptors were
prevent malicious caches from giving different versi
clients.

Routers began working harder to upload new descr
substantially changed.

Goals of the version 3 protocol

Version 3 of the Tor directory protocol tries to solve

```
* A great deal of bandwidth used to tra
  used by two fields that are not actua
  (namely read-history and write-histor
  moving them into a separate document
  fetch or use.

* It was possible under certain pervers
  to download an unusual set of networl
  partitioning themselves from clients
  typical set of documents.  Even under
  clients were sensitive to the ages of
  they downloaded.  Therefore, instead
  correlate multiple network status doc
  authorities collectively vote on a s
  document.

* The most sensitive data in the entire
  of the directory authorities) needed
  that the authorities can sign networl
  Now, the authorities' identity keys a
  to certify medium-term signing keys t
```

Outline

There is a small set (say, around 5-10) of semi-truste
of authorities is shipped with the Tor software. User
encouraged not to do so, in order to avoid partition

Every authority has a very-secret, long-term "Author
encrypted and/or offline, and is used to sign "key ce
certificate contains a medium-term (3-12 months) "a
the authority to sign other directory information. (N
distinct from the router identity key that the authori
router.)

Routers periodically upload signed "routers descript
describing their keys, capabilities, and other informa
"extra-info documents" containing information that
Directory authorities serve server descriptors index
the descriptor.

Routers may act as directory caches to reduce load
announce this in their descriptors.

Periodically, each directory authority generates a vie
status for known routers. They send a signed summ
other authorities. The authorities compute the resu
status" document containing the result of the vote.

Directory caches download, cache, and re-serve con

Clients, directory caches, and directory authorities a
out when their list of routers is out-of-date. (Directo
If it is, they download any missing server descriptors
descriptors from caches; caches and authorities dov
are downloaded by the hash of the descriptor, not b
prevents directory servers from attacking clients by
uses.

All directory information is uploaded and download

What's different from version 2?

Clients used to download multiple network status d
"status votes" above. They would compute the resul

Authorities used to sign documents using the same
as routers. This forced them to keep these extremel
unencrypted.

All of the information in extra-info documents used

Document meta-format

Server descriptors, directories, and running-routers
lightweight extensible information format.

The highest level object is a Document, which consis
begins with a KeywordLine, followed by zero or mor
a Keyword, optionally followed by whitespace and m
ends with a newline. A Keyword is a sequence of on
z0-9-], but may not start with -. An Object is a block
Enhanced-Mail (PEM) style format: that is, lines of en
inserting an ascii linefeed ("LF", also called newline,
§3.1). When line wrapping, implementations MUST w
decoding, implementations MUST ignore and discar

More formally:

```
NL = The ascii LF character (hex value 0x0a).
Document ::= (Item | NL)+
Item ::= KeywordLine Object?
KeywordLine ::= Keyword (WS Argument)*NL
Keyword = KeywordStart KeywordChar*
KeywordStart ::= 'A' ... 'Z' | 'a' ... 'z' |
KeywordChar ::= KeywordStart | '-'
Argument := ArgumentChar+
ArgumentChar ::= any graphical printing ASCI
WS = (SP | TAB)+
Object ::= BeginLine Base64-encoded-data EndL
BeginLine ::= "-----BEGIN " Keyword (" " Keyw
EndLine ::= "-----END " Keyword (" " Keyword)
```

A Keyword may not be `-----BEGIN`.

The BeginLine and EndLine of an Object must use th

When interpreting a Document, software MUST ignc
keyword it doesn't recognize; future implementatio
to understand any KeywordLine not currently descr

Other implementations that want to extend Tor's di

own items. The keywords for extension items SHOU
"X-", to guarantee that they will not conflict with key

In our document descriptions below, we tag Items w
tags are:

```
"At start, exactly once": These items MUS
  the document type, and MUST appear exac
  first item in their documents.

"Exactly once": These items MUST occur ex
  instance of the document type.

"At end, exactly once": These items MUST
  the document type, and MUST appear exac
  last item in their documents.

"At most once": These items MAY occur zer
  instance of the document type, but MUST

"Any number": These items MAY occur zero,
  instance of the document type.

"Once or more": These items MUST occur at
  of the document type, and MAY occur mor
```

For forward compatibility, each item MUST allow ext
unless otherwise noted. So if an item's description b

"thing" int int int NL

then implementations SHOULD accept this string as

"thing 5 9 11 13 16 12" NL

but not this string:

"thing 5" NL

and not this string:

```
    "thing 5 10 thing" NL
.
```

Whenever an item DOES NOT allow extra argument
arguments".

Signing documents

Every signable document below is signed in a simila
a final "Signature Item", a digest algorithm, and a sig

The Initial Item must be the first item in the docume

The Signature Item has the following format:

```
<signature item keyword> [arguments] NL SIGNA
```

The "SIGNATURE" Object contains a signature (using
padded digest of the entire document, taken from t
through the newline after the Signature Item's keyw

The signature does not include the algorithmIdentif

Unless specified otherwise, the digest algorithm is S

All documents are invalid unless signed with the cor

The "Digest" of a document, unless stated otherwise
signature scheme.

Voting timeline

Every consensus document has a "valid-after" (VA) t
"valid-until" (VU) time. VA MUST precede FU, which M
chosen so that every consensus will be "fresh" until
and "valid" for a while after. At least 3 consensuses :

The timeline for a given consensus is as follows:

VA-DistSeconds-VoteSeconds: The authorities excha
their vote to all other authorities.

VA-DistSeconds-VoteSeconds/2: The authorities try t
have.

Authorities SHOULD also reject any votes that other
time. (0.4.4.1-alpha was the first version to reject vo

Note: Refusing late uploaded votes minimizes the cl
when authorities are under bandwidth pressure. If a
vote, and finally uploads to a fraction of authorities
consensus different from the others. By refusing up
increase the likelihood that most authorities will use

Rejecting late uploaded votes does not fix the probl
able to download a specific vote, but others fail to d
consensus split. However, this change does remove
splits.

VA-DistSeconds: The authorities calculate the conse
the earliest point at which anybody can possibly get

VA-DistSeconds/2: The authorities try to download a

VA: All authorities have a multiply signed consensus

```
VA ... FU: Caches download the consensus.
     no way of telling what VA and FU are
     the consensus, they assume that the p
     equal to the previous one's FU, and t
     that.)

FU: The consensus is no longer the freshes

FU ... (the current consensus's VU): Clien
     (See note above: clients guess that t
     two intervals after the current VA.)
```

VU: The consensus is no longer valid; clients should
consensus if they have not done so already.

VU + 24 hours: Clients will no longer use the consen

VoteSeconds and DistSeconds MUST each be at leas
each be at least 5 minutes.

Router operation and fo

This section describes how relays must behave whe
directory authorities, and the formats that they use

Uploading server descri info documents

ORs SHOULD generate a new server descriptor and any of the following events have occurred:

```
        - A period of time (18 hrs by default)
          time a descriptor was generated.

        - A descriptor field other than bandwid

        - Its uptime is less than 24h and bandw
 2
          from the last time a descriptor was g
          interval of time (3 hours by default)

        - Its uptime has been reset (by restart

        - It receives a networkstatus consensus

        - It receives a networkstatus consensus
          with the StaleDesc flag.

        [XXX this list is incomplete; see route
         in routerlist.c for others]
```

ORs SHOULD NOT publish a new server descriptor c above events have occurred and not much time has

Tor versions older than 0.3.5.1-alpha ignore uptime changes.

After generating a descriptor, ORs upload them to e posting them (in order) to the URL

http://hostname:port/tor/

Server descriptors may not exceed 20,000 bytes in l exceed 50,000 bytes in length. If they do, the author

Server descriptor forma

Server descriptors consist of the following items.

In lines that take multiple arguments, extra argume
Many of the nonterminals below are defined in sect

Note that many versions of Tor will generate an extr
descriptors. Implementations MUST tolerate one or
single descriptor or a list of concatenated descripto
NOT generate such blank lines.

"router" nickname address ORPort SOCKSPort DirPo

[At start, exactly once.]

Indicates the beginning of a server descriptor. "nick
nickname as specified in section 2.1.3. "address" mu
format. The last three numbers indicate the TCP po
functionality. ORPort is a port at which this OR acce
protocol; SOCKSPort is deprecated and should alwa
which this OR accepts directory-related HTTP conne
the value 0 is given instead of a port number. (At lea
SHOULD be set; authorities MAY reject any descript

```
     "identity-ed25519" NL "-----BEGIN ED2551
            "-----END ED25519 CERT-----" NL
```

[Exactly once, in second position in document.] [No

The certificate is a base64-encoded Ed25519 certific
terminating =s removed. When this element is prese
second element in the router descriptor.

The certificate has CERT_TYPE of [04]. It must includ
extension (see cert-spec.txt, section 2.2.1), so that w

[Before Tor 0.4.5.1-alpha, this field was optional.]

"master-key-ed25519" SP MasterKey NL

[Exactly once]

Contains the base-64 encoded ed25519 master key
MUST match the identity key in the identity-ed2551

[Before Tor 0.4.5.1-alpha, this field was optional.]

"bandwidth" bandwidth-avg bandwidth-burst bandw

[Exactly once]

Estimated bandwidth for this router, in bytes per se
volume per second that the OR is willing to sustain o
bandwidth is the volume that the OR is willing to sus
"observed" value is an estimate of the capacity this r
remembers the max bandwidth sustained output ov
5 days, and another sustained input. The "observed
numbers.

Tor versions released before 2018 only kept bandwi
versions are no longer supported or recommended

"platform" string NL

[At most once]

A human-readable string describing the system on w
include the operating system, and SHOULD include
implementing the Tor protocol.

"published" YYYY-MM-DD HH:MM:SS NL

[Exactly once]

The time, in UTC, when this descriptor (and its corre
was generated.

"fingerprint" fingerprint NL

[At most once]

A fingerprint (a HASH_LEN-byte of asn1 encoded pu
space after every 4 characters) for this router's iden
invalid (and MUST be rejected) if the fingerprint line

```
        [We didn't start parsing this line unt
         be marked with "opt" until earlier ve

  "hibernating" bool NL

        [At most once]
```

If the value is 1, then the Tor relay was hibernating w
and shouldn't be used to build circuits.

```
    [We didn't start parsing this line unt
     marked with "opt" until earlier vers

"uptime" number NL

    [At most once]

    The number of seconds that this OR pro

"onion-key" NL a public key in PEM format
```

[Exactly once] [No extra arguments]

This key is used to encrypt CREATE cells for this OR. least 1 week after any new key is published in a sub bits.

The key encoding is the encoding of the key as a PK encoded in base64, and wrapped in "-----BEGIN RSA PUBLIC KEY-----".

"onion-key-crosscert" NL a RSA signature in PEM for

[Exactly once] [No extra arguments]

This element contains an RSA signature, generated

```
A SHA1 hash of the RSA identity key
  i.e. RSA key from "signing-key" (
The Ed25519 identity key,
  i.e. Ed25519 key from "master-key
```

If there is no Ed25519 identity key, or if in some futu key, the corresponding field must be zero-filled.

Parties verifying this signature MUST allow addition above.

This signature proves that the party creating the de key corresponding to the onion-key.

[Before Tor 0.4.5.1-alpha, this field was optional wh

"ntor-onion-key" base-64-encoded-key

[Exactly once]

A curve25519 public key used for the ntor circuit ext encoding of the OR's curve25519 public key, encode be omitted from the base64 encoding. The key MUS after any new key is published in a subsequent desc

[Before Tor 0.4.5.1-alpha, this field was optional.]

```
"ntor-onion-key-crosscert" SP Bit NL
       "-----BEGIN ED25519 CERT-----" NL
       "-----END ED25519 CERT-----" NL
```

[Exactly once] [No extra arguments]

A signature created with the ntor-onion-key, using tl
cert-spec.txt, with type [0a]. The signed key here is t

Bit must be "0" or "1". It indicates the sign of the ed2
ntor onion key. If Bit is "0", then implementations M
of the resulting ed25519 public key is positive. Othe
x-coordinate MUST be negative.

To compute the ed25519 public key corresponding t
explanation on key formats, see appendix C.

This signature proves that the party creating the des
key corresponding to the ntor-onion-key.

[Before Tor 0.4.5.1-alpha, this field was optional whe

"signing-key" NL a public key in PEM format

[Exactly once] [No extra arguments]

The OR's long-term RSA identity key. It MUST be 102

The encoding is as for "onion-key" above.

"accept" exitpattern NL "reject" exitpattern NL

[Any number]

These lines describe an "exit policy": the rules that a
to allow a new stream to a given address. The 'exitp
There MUST be at least one such entry. The rules ar
matches, the address will be accepted. For clarity, th
or reject *:*.

"ipv6-policy" SP ("accept" / "reject") SP PortList NL

[At most once.]

An exit-policy summary as specified in sections 3.4.1

rules for connecting to IPv6 addresses. A missing "ip
policy reject 1-65535".

"overload-general" SP version SP YYYY-MM-DD HH:M

[At most once.]

Indicates that a relay has reached an "overloaded st
following load metrics:

```
- Any OOM invocation due to memory p
- Any ntor onionskins are dropped
- TCP port exhaustion
```

The timestamp is when at least one metrics was det
and thus, as an example, "2020-01-10 13:00:00" is an
a binary state, if the line is present, we consider tha
somewhere between the provided timestamp and t
document which is when the document was genera

The overload-general line should remain in place fo
limits are reached again in this period, the timestam
restarts.

The 'version' field is set to '1' for now.

```
  (Introduced in tor-0.4.6.1-alpha, but m
   descriptor in tor-0.4.6.2-alpha)

"router-sig-ed25519" SP Signature NL

   [Exactly once.]
```

It MUST be the next-to-last element in the descripto
RSA signature. It MUST contain an Ed25519 signatur
document. This digest is taken from the first charac
after the "router-sig-ed25519" string. Before compu
descriptor signature v1" is prefixed to the documen

The signature is encoded in Base64, with terminatin

The signing key in the identity-ed25519 certificate M
document.

[Before Tor 0.4.5.1-alpha, this field was optional wh

"router-signature" NL Signature NL

[At end, exactly once] [No extra arguments]

The "SIGNATURE" object contains a signature of the
server descriptor, taken from the beginning of the "
the "router-signature" line. The server descriptor is
performed with the router's identity key.

"contact" info NL

[At most once]

Describes a way to contact the relay's administrator
address and a PGP key fingerprint.

"bridge-distribution-request" SP Method NL

[At most once, bridges only.]

The "Method" describes how a Bridge address is dis
methods are: "none", "any", "https", "email", "moat"
distributing your bridge address. If set to "any", Brid
your bridge address. Choosing any of the other met
your bridge via a specific method:

```
           - "https" specifies distribution v
              https://bridges.torproject.org;
           - "email" specifies distribution v
             bridges@torproject.org;
           - "moat" specifies distribution via
             Browser; and

         Potential future "Method" specifiers
             Method = (KeywordChar | "_") +
```

All bridges SHOULD include this line. Non-bridges M

BridgeDB SHOULD treat unrecognized Method valu

(Default: "any")

[This line was introduced in 0.3.2.3-alpha, with a mi
0.2.9.14, 0.3.0.13, 0.3.1.9, and later.]

"family" names NL

[At most once]

'Names' is a space-separated list of relay nicknames
another in their "family" entries, then OPs should tr
purpose of path selection.

For example, if node A's descriptor contains "family
"family A", then node A and node B should never be

```
"read-history" YYYY-MM-DD HH:MM:SS (NSEC
    [At most once]
"write-history" YYYY-MM-DD HH:MM:SS (NSEC
    [At most once]
```

(These fields once appeared in router descriptors, b
descriptors since 0.2.0.x.)

"eventdns" bool NL

[At most once]

Declare whether this version of Tor is using the new
Tor with this field set to false SHOULD NOT be used

```
    [This option is obsolete.  All Tor cu
     to have the evdns backend.]

"caches-extra-info" NL
```

[At most once.] [No extra arguments]

Present only if this router is a directory cache that p

[Versions before 0.2.0.1-alpha don't recognize this]

"extra-info-digest" SP sha1-digest [SP sha256-digest

[At most once]

"sha1-digest" is a hex-encoded SHA1 digest (using u
extra-info document, as signed in the router's extra-
signature). (If this field is absent, the router is not up
document.)

"sha256-digest" is a base64-encoded SHA256 digest
the "sha1-digest", this digest is calculated over the e
signature. This difference is due to a long-lived bug
would be difficult to roll out an incremental fix for, r
algorithms specified should not include the signatur
digest.

[Versions before 0.2.7.2-alpha did not include a SHA
alpha don't recognize this field at all.]

"hidden-service-dir" NL

[At most once.]

Present only if this router stores and serves hidden supports the descriptor versions declared in the HS entry, this router supports version 2 descriptors.

```
    "protocols" SP "Link" SP LINK-VERSION-LIST
           CIRCUIT-VERSION-LIST NL

        [At most once.]
```

An obsolete list of protocol versions, superseded by parsed, and has not been emitted since Tor 0.2.9.4- generate nor parse this line.

"allow-single-hop-exits" NL

[At most once.] [No extra arguments]

Present only if the router allows single-hop circuits relays do not support this: this is included for specia perspective access and such. This is obsolete in tor

"or-address" SP ADDRESS ":" PORT NL

[Any number]

ADDRESS = IP6ADDR | IP4ADDR IPV6ADDR = an ipv brackets. IPV4ADDR = an ipv4 address, represented between 1 and 65535 inclusive.

An alternative for the address and ORPort of the "ro capabilities:

```
          * or-address can be either an IPv4 o
          * or-address allows for multiple ORP
```

A descriptor SHOULD NOT include an or-address lin address:port pair from its "router" line.

The ordering of or-address lines and their PORT ent limited number of address/port pairs. As of Tor 0.2. advertised and used.

"tunnelled-dir-server" NL

[At most once.] [No extra arguments]

```
        Present if the router accepts "tunnele
        BEGIN_DIR cell over the router's OR po
           (Added in 0.2.8.1-alpha. Before th
           tunneled directory requests only if
           or if they were bridges.)

    "proto" SP Entries NL

        [Exactly once.]
```

Entries = Entries = Entry Entries = Entry SP Entries

Entry = Keyword "=" Values

Values = Values = Value Values = Value "," Values

Value = Int Value = Int "-" Int

Int = NON_ZERO_DIGIT Int = Int DIGIT

Each 'Entry' in the "proto" line indicates that the Tor of the protocol in question. Entries should be sortec numerically ascending within each entry. (This impli overlapping ranges.) Ranges should be represented be no larger than 63.

This field was first added in Tor 0.2.9.x.

[Before Tor 0.4.5.1-alpha, this field was optional.]

Extra-info document for

Extra-info documents consist of the following items:

```
"extra-info" Nickname Fingerprint NL
    [At start, exactly once.]
```

Identifies what router this is an extra-info descriptor
(using upper-case letters), with no spaces.

```
"identity-ed25519"
    [As in router descriptors]

"published" YYYY-MM-DD HH:MM:SS NL

   [Exactly once.]
```

The time, in UTC, when this document (and its corre
was generated. It MUST match the published time ir

```
"read-history" YYYY-MM-DD HH:MM:SS (NSEC
    [At most once.]
"write-history" YYYY-MM-DD HH:MM:SS (NSE(
    [At most once.]
```

Declare how much bandwidth the OR has used rece
NSEC seconds. The YYYY-MM-DD HH:MM:SS field de
interval. The numbers are the number of bytes usec
from oldest to newest.

These fields include both IPv4 and IPv6 traffic.

```
"ipv6-read-history" YYYY-MM-DD HH:MM:SS (
    [At most once]
"ipv6-write-history" YYYY-MM-DD HH:MM:SS
    [At most once]
```

Declare how much bandwidth the OR has used rece
history" and "write-history" for full details.

```
"geoip-db-digest" Digest NL
    [At most once.]
```

SHA1 digest of the IPv4 GeoIP database file that is u
country codes.

```
"geoip6-db-digest" Digest NL
    [At most once.]
```

SHA1 digest of the IPv6 GeoIP database file that is u
country codes.

("geoip-start-time" YYYY-MM-DD HH:MM:SS NL) ("ge
CC=NUM,CC=NUM,... NL)

Only generated by bridge routers (see blocking.pdf),
configured with a geoip database. Non-bridges SHO
Contains a list of mappings from two-letter country
that have connected to that bridge from that countr
the nearest multiple of 8 in order to hamper traffic a
it has at least one address. The time in "geoip-start-
collecting geoip statistics.

"geoip-start-time" and "geoip-client-origins" have be
"bridge-ips" in 0.2.2.4-alpha. The reason is that the r
stats" as determined by subtracting "geoip-start-tim
variable length, whereas the measurement interval
exactly 24 hours long. In order to clearly distinguish
the old ones, the new keywords have been introduc

```
"bridge-stats-end" YYYY-MM-DD HH:MM:SS (N
    [At most once.]
```

YYYY-MM-DD HH:MM:SS defines the end of the inclu
NSEC seconds (86400 seconds by default).

A "bridge-stats-end" line, as well as any other "bridg
has been running as a bridge for at least 24 hours.

```
"bridge-ips" CC=NUM,CC=NUM,... NL
    [At most once.]
```

List of mappings from two-letter country codes to th
that have connected from that country to the bridge
rounded up to the nearest multiple of 8.

```
"bridge-ip-versions" FAM=NUM,FAM=NUM,...
    [At most once.]
```

List of unique IP addresses that have connected to t

```
"bridge-ip-transports" PT=NUM,PT=NUM,...
    [At most once.]
```

List of mappings from pluggable transport names tc
that have connected using that pluggable transport.
counted using the reserved pluggable transport nar
received a connection from a transport proxy but w
pluggable transport, we use the reserved pluggable

(" `<OR>` " and " `<??>` " are reserved because normal p
match the following regular expression: " `[a-zA-Z_]`

The pluggable transport name list is sorted into lexi

If no clients have connected to the bridge yet, we or
stats file.

```
"dirreq-stats-end" YYYY-MM-DD HH:MM:SS (N
    [At most once.]
```

YYYY-MM-DD HH:MM:SS defines the end of the inclu
NSEC seconds (86400 seconds by default).

A "dirreq-stats-end" line, as well as any other "dirrec
has opened its Dir port and after 24 hours of measu

```
"dirreq-v2-ips" CC=NUM,CC=NUM,... NL
    [At most once.]
"dirreq-v3-ips" CC=NUM,CC=NUM,... NL
    [At most once.]
```

List of mappings from two-letter country codes to th
that have connected from that country to request a
the nearest multiple of 8. Only those IP addresses a
answer with a 200 OK status code. (Note here and b
0.2.5.2-alpha, no longer cache or serve v2 networkst

```
"dirreq-v2-reqs" CC=NUM,CC=NUM,... NL
    [At most once.]
"dirreq-v3-reqs" CC=NUM,CC=NUM,... NL
    [At most once.]
```

List of mappings from two-letter country codes to th
network statuses from that country, rounded up to
requests are counted that the directory can answer

```
"dirreq-v2-share" NUM% NL
    [At most once.]
"dirreq-v3-share" NUM% NL
    [At most once.]
```

The share of v2/v3 network status requests that the
clients based on its advertised bandwidth comparec
capacity. Shares are formatted in percent with two c
as means over the whole 24-hour interval.

```
"dirreq-v2-resp" status=NUM,... NL
    [At most once.]
"dirreq-v3-resp" status=NUM,... NL
    [At most once.]
```

List of mappings from response statuses to the num
statuses that were answered with that response sta
multiple of 4. Only response statuses with at least 1
response statuses can be added at any time. The cu
follows:

```
        "ok": a network status request is ans
           corresponds to the sum of all requ
           "dirreq-v2-reqs" or "dirreq-v3-rec
           rounding up.
        "not-enough-sigs: a version 3 network
           sufficient number of requested aut
        "unavailable": a requested network st
        "not-found": a requested network stat
        "not-modified": a network status has
           If-Modified-Since time that is inc
        "busy": the directory is busy.

"dirreq-v2-direct-dl" key=NUM,... NL
    [At most once.]
"dirreq-v3-direct-dl" key=NUM,... NL
    [At most once.]
"dirreq-v2-tunneled-dl" key=NUM,... NL
    [At most once.]
"dirreq-v3-tunneled-dl" key=NUM,... NL
    [At most once.]
```

List of statistics about possible failures in the downl
statuses. Requests are either "direct" HTTP-encodec
port, or "tunneled" requests using a BEGIN_DIR cell
possible statistics can change, and statistics can be l
list of statistics is as follows:

Successful downloads and failures:

```
        "complete": a client has finished the
        "timeout": a download did not finish
           starting to send the response.
        "running": a download is still runnir
           measurement period for less than 1
           send the response.

        Download times:

        "min", "max": smallest and largest me
        "d[1-4,6-9]": 1st to 4th and 6th to 9
           bandwidth in B/s. For a given dec
           had a smaller bandwidth than di, a
           had a larger bandwidth than di.
        "q[1,3]": 1st and 3rd quartile of mea
           fourth of all downloads had a smal
           fourth of all downloads had a larg
           remaining half of all downloads ha
           q3.
        "md": median of measured bandwidth ir
           had a smaller bandwidth than md, t
           bandwidth than md.

    "dirreq-read-history" YYYY-MM-DD HH:MM:SS
        [At most once]
    "dirreq-write-history" YYYY-MM-DD HH:MM:S
        [At most once]
```

Declare how much bandwidth the OR has spent on a
is divided into intervals of NSEC seconds. The YYYY-I
end of the most recent interval. The numbers are th
recent intervals, ordered from oldest to newest.

```
    "entry-stats-end" YYYY-MM-DD HH:MM:SS (NS
        [At most once.]
```

YYYY-MM-DD HH:MM:SS defines the end of the inclu
NSEC seconds (86400 seconds by default).

An "entry-stats-end" line, as well as any other "entry
has been running for at least 24 hours.

```
    "entry-ips" CC=NUM,CC=NUM,... NL
        [At most once.]
```

List of mappings from two-letter country codes to th
that have connected from that country to the relay a
rounded up to the nearest multiple of 8.

```
"cell-stats-end" YYYY-MM-DD HH:MM:SS (NSE
    [At most once.]
```

YYYY-MM-DD HH:MM:SS defines the end of the inclu
NSEC seconds (86400 seconds by default).

A "cell-stats-end" line, as well as any other "cell-*" lir
been running for at least 24 hours.

```
"cell-processed-cells" NUM,...,NUM NL
    [At most once.]
```

Mean number of processed cells per circuit, subdivi
number of cells they have processed in descending
circuits.

```
"cell-queued-cells" NUM,...,NUM NL
    [At most once.]
```

Mean number of cells contained in queues by circui
by 1) determining the mean number of cells in a sin
termination and 2) calculating the mean for all circu
"cell-processed-cells". Numbers have a precision of

Note that this statistic can be inaccurate for circuits
end of the measurement interval.

```
"cell-time-in-queue" NUM,...,NUM NL
    [At most once.]
```

Mean time cells spend in circuit queues in millisecor
determining the mean time cells spend in the queue
the mean for all circuits in a given decile as determi

Note that this statistic can be inaccurate for circuits
end of the measurement interval.

```
"cell-circuits-per-decile" NUM NL
    [At most once.]
```

Mean number of circuits that are included in any of
integer.

```
"conn-bi-direct" YYYY-MM-DD HH:MM:SS (NSE
    [At most once]
```

Number of connections, split into 10-second interva bi-directionally as observed in the NSEC seconds (us MM-DD HH:MM:SS. Every 10 seconds, we determine read and wrote less than a threshold of 20 KiB (BEL(we wrote (READ), wrote at least 10 times more than more than the threshold, but not 10 times more in e classifying a connection, read and write counters ar interval.

This measurement includes both IPv4 and IPv6 conn

```
    "ipv6-conn-bi-direct" YYYY-MM-DD HH:MM:SS
NL
        [At most once]
```

Number of IPv6 connections that are used uni-direc bi-direct" for more details.

```
    "exit-stats-end" YYYY-MM-DD HH:MM:SS (NSE
        [At most once.]
```

YYYY-MM-DD HH:MM:SS defines the end of the inclu NSEC seconds (86400 seconds by default).

An "exit-stats-end" line, as well as any other "exit-*" been running for at least 24 hours and only if the re single port and IP address is sufficient).

```
    "exit-kibibytes-written" port=N,port=N,..
        [At most once.]
    "exit-kibibytes-read" port=N,port=N,... N
        [At most once.]
```

List of mappings from ports to the number of kibiby read from exit connections to that port, rounded up limit the number of listed ports and subsume any re

```
    "exit-streams-opened" port=N,port=N,... N
        [At most once.]
```

List of mappings from ports to the number of opene up to the nearest multiple of 4. Relays may limit the any remaining opened streams under port "other".

```
"hidserv-stats-end" YYYY-MM-DD HH:MM:SS (
    [At most once.]
"hidserv-v3-stats-end" YYYY-MM-DD HH:MM:S
    [At most once.]
```

YYYY-MM-DD HH:MM:SS defines the end of the inclu
NSEC seconds (86400 seconds by default).

A "hidserv-stats-end" line, as well as any other "hids
relay has been running for at least 24 hours.

(Introduced in tor-0.4.6.1-alpha)

```
"hidserv-rend-relayed-cells" SP NUM SP ke
    [At most once.]
"hidserv-rend-v3-relayed-cells" SP NUM SP
    [At most once.]
```

Approximate number of RELAY cells seen in either d
and successfully processing a RENDEZVOUS1 cell.

The original measurement value is obfuscated in se
the nearest multiple of 'bin_size' which is reported i
a (possibly negative) noise value is added to the resu
sampling from a Laplace distribution with mu = 0 ar
and 'epsilon' being reported in the key=val part, too
obfuscation steps is truncated to the next smaller ir
that the overall reported value can be negative.

(Introduced in tor-0.4.6.1-alpha)

```
"hidserv-dir-onions-seen" SP NUM SP key=v
    [At most once.]
"hidserv-dir-v3-onions-seen" SP NUM SP ke
    [At most once.]
```

Approximate number of unique hidden-service ider
and accepted by this hidden-service directory.

The original measurement value is obfuscated in the
reported in "hidserv-rend-relayed-cells", but possibl
reported in the key=val part of this line. Note that th
negative.

(Introduced in tor-0.4.6.1-alpha)

```
"transport" transportname address:port [a
    [Any number.]
```

Signals that the router supports the 'transportname 'address' and TCP port 'port'. A single descriptor MU line with the same 'transportname'.

Pluggable transports are only relevant to bridges, bu bridge relays as well.

```
"padding-counts" YYYY-MM-DD HH:MM:SS (NSE
    [At most once.]
```

YYYY-MM-DD HH:MM:SS defines the end of the inclu NSEC seconds (86400 seconds by default). Counts a

The keyword list is currently as follows:

```
        bin-size
          - The current rounding value for c
            default)
        write-drop
          - The number of RELAY_DROP cells t
        write-pad
          - The number of CELL_PADDING cells
        write-total
          - The total number of cells this r
        read-drop
          - The number of RELAY_DROP cells t
        read-pad
          - The number of CELL_PADDING cells
        read-total
          - The total number of cells this r
        enabled-read-pad
          - The number of CELL_PADDING cells
            connections that support padding
        enabled-read-total
          - The total number of cells this r
            that support padding
        enabled-write-pad
          - The total number of cells this r
            that support padding
        enabled-write-total
          - The total number of cells sent b
            that support padding
        max-chanpad-timers
          - The maximum number of timers tha
            padding in the previous NSEC int

  "overload-ratelimits" SP version SP YYYY-
                  SP rate-limit SP burst-
                  SP read-overload-count
      [At most once.]

      Indicates that a bandwidth limit was
```

The "rate-limit" and "burst-limit" are the raw values
BandwidthBurst found in the torrc configuration file

The "{read|write}-overload-count" are the counts of
of burst/rate were exhausted and thus the maximum
occurrences. To make the counter more meaningfu
saturating the counter when a relay is overloaded, v

The 'version' field is set to '1' for now.

(Introduced in tor-0.4.6.1-alpha)

```
  "overload-fd-exhausted" SP version YYYY-M
      [At most once.]
```

Indicates that a file descriptor exhaustion was expe

The timestamp indicates that the maximum was rea
"published" timestamp of the document.

This overload field should remain in place for 72 ho
are reached again in this period, the timestamp is u
restarts.

The 'version' field is set to '1' for the initial impleme
only when a socket open fails.

(Introduced in tor-0.4.6.1-alpha)

```
"router-sig-ed25519"
    [As in router descriptors]

"router-signature" NL Signature NL
    [At end, exactly once.]
    [No extra arguments]
```

A document signature as documented in section 1.3
the final item "router-signature", signed with the ro

Nonterminals in server (

```
nickname ::= between 1 and 19 alphanumeric
   case-insensitive.
hexdigest ::= a '$', followed by 40 hexade
   ([A-Fa-f0-9]). [Represents a relay by t
   key.]
```

exitpattern ::= addrspec ":" portspec portspec ::= "*"
integer between 1 and 65535, inclusive.

```
[Some implementations incorrectly gener
 Implementations SHOULD accept this, an
 Connections to port 0 are never permit
```

addrspec ::= "*" | ip4spec | ip6spec ipv4spec ::= ip4
ip4mask ip4 ::= an IPv4 address in dotted-quad form
dotted-quad format num_ip4_bits ::= an integer betw
num_ip6_bits ip6 ::= an IPv6 address, surrounded by
integer between 0 and 128

bool ::= "0" | "1"

Directory authority oper formats

Every authority has two keys used in this protocol: a key. (Authorities also have a router identity key usec earlier versions of the directory protocol.) The ident sign new key certificates using new signing keys; it is used to sign key certificates and status documents.

Creating key certificates

Key certificates consist of the following items:

"dir-key-certificate-version" version NL

[At start, exactly once.]

Determines the version of the key certificate. MUST
this document. Implementations MUST reject forma

```
"dir-address" IPPort NL
    [At most once]

    An IP:Port for this authority's direc

"fingerprint" fingerprint NL

    [Exactly once.]
```

Hexadecimal encoding without spaces based on the

"dir-identity-key" NL a public key in PEM format

[Exactly once.] [No extra arguments]

The long-term authority identity key for this authori
bits long; it MUST NOT be shorter than 1024 bits.

"dir-key-published" YYYY-MM-DD HH:MM:SS NL

[Exactly once.]

The time (in UTC) when this document and correspo

Implementations SHOULD reject certificates that are
though they MAY tolerate some clock skew.

"dir-key-expires" YYYY-MM-DD HH:MM:SS NL

[Exactly once.]

A time (in UTC) after which this key is no longer valic

Implementations SHOULD reject expired certificates
clock skew.

"dir-signing-key" NL a key in PEM format

[Exactly once.] [No extra arguments]

The directory server's public signing key. This key M
longer.

"dir-key-crosscert" NL CrossSignature NL

[Exactly once.] [No extra arguments]

CrossSignature is a signature, made using the certifi
PKCS1-padded hash of the certificate's identity key.
broken versions of the parser, we wrap the base64-
SIGNATURE---- and -----END ID SIGNATURE----- tags.
portion to be omitted, however.

Implementations MUST verify that the signature is a
identity key using the signing key.

"dir-key-certification" NL Signature NL

[At end, exactly once.] [No extra arguments]

A document signature as documented in section 1.3
certificate-version" and the final item "dir-key-certifi
identity key.

Authorities MUST generate a new signing key and co
key expires.

Accepting server descrip info document uploads

When a router posts a signed descriptor to a directo whether it is well-formed and correctly self-signed. I the nickname in question is not already assigned to Finally, the authority MAY check that the router is nc another reason.

An authority also keeps a record of all the Ed25519/ seen before. It rejects any descriptor that has a kno already seen accompanied by a different RSA/Ed ide

At a future date, authorities will begin rejecting all d previously accompanied by an Ed25519 key, if the d key.

At a future date, authorities will begin rejecting all d key.

If the descriptor passes these tests, and the authori for a router with this public key, it accepts the descr

If the authority *does* have a descriptor with the sam descriptor is remembered if its publication time is m descriptor for that router, and either:

```
- There are non-cosmetic differences be
  new one.
- Enough time has passed between the de
  (Currently, 2 hours.)
```

Differences between server descriptors are "non-co force an upload as described in section 2.1 above.

Note that the "cosmetic difference" test only applies descriptors that the authority downloads from othe

When a router posts a signed extra-info document t again checks it for well-formedness and correct sign the extra-info-digest in some router descriptor that accepts it and stores it and serves it as requested. If

Computing microdescrip

Microdescriptors are a stripped-down version of ser
directory authorities which may additionally contain
Microdescriptors contain only the most relevant par
Microdescriptors are expected to be relatively static
week. Microdescriptors do not contain any informat
which servers to fetch information about, or which s

Microdescriptors are a straight transform from the s
method. Microdescriptors have no header or footer
the hash of its concatenated elements without a sig
Microdescriptors do not contain any version inform
determined by the consensus method.

Starting with consensus method 8, microdescriptors
from or based on the server descriptor. Order matt
authorities must be able to transform a given server
into the exact same microdescriptor.

"onion-key" NL a public key in PEM format

[Exactly once, at start] [No extra arguments]

The "onion-key" element as specified in section 2.1.

When generating microdescriptors for consensus m
must be absent. For consensus method 29 or earlie

"ntor-onion-key" SP base-64-encoded-key NL

[Exactly once]

The "ntor-onion-key" element as specified in sectior

(Only included when generating microdescriptors fo

[Before Tor 0.4.5.1-alpha, this field was optional.]

"a" SP address ":" port NL

[Any number]

Additional advertised addresses for the OR.

Present currently only if the OR advertises at least o
address is included and all others are omitted. Any
be ignored.

Address and port are as for "or-address" as specifie

(Only included when generating microdescriptors fc

"family" names NL

[At most once]

The "family" element as specified in section 2.1.1.

When generating microdescriptors for consensus m
canonicalization algorithm is applied to improve cor

```
For all entries of the form $hexic
remove the =name or ~name portion.

Remove all entries of the form $he
40 hexadecimal characters long.

If an entry is a valid nickname, p

If an entry is a valid $hexid, put

If there are any entries, add a s
the relay in question, so that it
family.

Sort all entries in lexical order.

Remove duplicate entries.
```

(Note that if an entry is not of the form "nickname",
"$hexid~nickname", then it will be unchanged: this i
compatible.)

"p" SP ("accept" / "reject") SP PortList NL

[Exactly once.]

The exit-policy summary as specified in sections 3.4

[With microdescriptors, clients don't learn exact exit
whether a relay accepts their request, try the BEGIN
exit-policy if they guessed wrong, in which case they

[In consensus methods before 5, this line was omitt

"p6" SP ("accept" / "reject") SP PortList NL

[At most once]

The IPv6 exit policy summary as specified in section
equivalent to "p6 reject 1-65535".

(Only included when generating microdescriptors fc

"id" SP "rsa1024" SP base64-encoded-identity-digest

[At most once]

The node identity digest (as described in tor-spec.tx
=s. This line is included to prevent collisions betwee

Implementations SHOULD ignore these lines: they a
prevent collisions.

(Only included when generating microdescriptors fc

"id" SP "ed25519" SP base64-encoded-ed25519-ider

[At most once]

The node's master Ed25519 identity key, base64 enc

All implementations MUST ignore this key for any m
entry in the consensus includes the 'NoEdConsensu

(Only included when generating microdescriptors fc

"id" SP keytype ... NL

[At most once per distinct keytype.]

Implementations MUST ignore "id" lines with unreco
"rsa1024" or "ed25519"

"pr" SP Entries NL

[Exactly once.]

The "proto" element as specified in section 2.1.1.

[Before Tor 0.4.5.1-alpha, this field was optional.]

(Note that with microdescriptors, clients do not lear
they only learn a hash of the RSA identity key. This is
identity key when doing a TLS handshake, and all th
in their CREATE cells.)

Exchanging votes

Authorities divide time into Intervals. Authority adm
same interval length, and SHOULD pick intervals tha
time (e.g., 5 minutes, 15 minutes, 30 minutes, 60 mi
SHOULD be chosen to divide evenly into a 24-hour c

Authorities SHOULD act according to interval and de
a latest consensus, they SHOULD default to a 30-mi
and a 5 minute DistDelay.

Authorities MUST take pains to ensure that their clo
seconds. (Running NTP is usually sufficient.)

The first voting period of each day begins at 00:00 (
day would be truncated by one-half or more, it is m

An authority SHOULD publish its vote immediately a
(minus VoteSeconds+DistSeconds). It does this by m

```
http://<hostname>/tor/status-vote/next/author
```

and sending it in an HTTP POST request to each oth

```
http://<hostname>/tor/post/vote
```

If, at the start of the voting period, minus DistSecon
current statement from another authority, the first
statement.

Once an authority has a vote from another authority

```
http://<hostname>/tor/status-vote/next/<fp>.z
```

where `<fp>` is the fingerprint of the other authority

```
http://<hostname>/tor/status-vote/next/d/<d>.
```

where `<d>` is the digest of the vote document.

Also, once an authority receives a vote from anothe
descriptors and fetches them from that authority. T
authority to hear about relays that didn't publish th
while it's too late for the authority to include relays i
in its next vote. See section 3.6 below for details.

Vote and consensus stat formats

Votes and consensuses are more strictly formatted specification, since different authorities must be abl consensus given the same set of votes.

The procedure for deciding when to generate vote a described in section 1.4 on the voting timeline.

Status documents contain a preamble, an authority and one or more footer signature, in that order.

Unlike other formats described above, a SP in these character (hex 20).

Some items appear only in votes, and some items a specified, items occur in both.

The preamble contains the following items. They SH

"network-status-version" SP version NL

[At start, exactly once.]

A document format version. For this specification, tl

"vote-status" SP type NL

[Exactly once.]

The status MUST be "vote" or "consensus", dependi

"consensus-methods" SP IntegerList NL

[At most once for votes; does not occur in consensu

A space-separated list of supported methods for ge section 3.8.1 for details. Absence of the line means t

"consensus-method" SP Integer NL

[At most once for consensuses; does not occur in vo

See section 3.8.1 for details.

(Only included when the vote is generated with cons "published" SP YYYY-MM-DD SP HH:MM:SS NL

[Exactly once for votes; does not occur in consensus

The publication time for this status document (if a v

"valid-after" SP YYYY-MM-DD SP HH:MM:SS NL

[Exactly once.]

The start of the Interval for this vote. Before this tim
from this vote is not officially in use.

(Note that because of propagation delays, clients an
documents that are up to `DistSeconds` earlier than
them.)

See section 1.4 for voting timeline information.

"fresh-until" SP YYYY-MM-DD SP HH:MM:SS NL

[Exactly once.]

The time at which the next consensus should be pro
point in downloading another consensus, since ther
for voting timeline information.

"valid-until" SP YYYY-MM-DD SP HH:MM:SS NL

[Exactly once.]

The end of the Interval for this vote. After this time,
recent consensus. See section 1.4 for voting timelin

In practice, clients continue to use the consensus fo
valid, if no more recent consensus can be download

"voting-delay" SP VoteSeconds SP DistSeconds NL

[Exactly once.]

VoteSeconds is the number of seconds that we will a
authorities; DistSeconds is the number of seconds w
all authorities. See section 1.4 for voting timeline inf

"client-versions" SP VersionList NL

[At most once.]

A comma-separated list of recommended Tor versic

order. The versions are given as defined by version-
about client versions.

"server-versions" SP VersionList NL

[At most once.]

A comma-separated list of recommended Tor versic
order. The versions are given as defined by version-
about server versions.

"package" SP PackageName SP Version SP URL SP D

[Any number of times.]

For this element:

```
PACKAGENAME = NONSPACE
VERSION = NONSPACE
URL = NONSPACE
DIGESTS = DIGEST | DIGESTS SP DIGEST
DIGEST = DIGESTTYPE "=" DIGESTVAL
NONSPACE = one or more non-space prir
DIGESTVAL = DIGESTTYPE = one or more
      other than "=".
```

Indicates that a package called "package" of version
its digest as computed with DIGESTTYPE is equal to
lines are sorted lexically by "PACKAGENAME VERSIO
appear in ascending order. A consensus must not co
VERSION" more than once. If a vote contains the sar
than once, all but the last is ignored.

Included in consensuses only for methods 19-33. Ea
method 34 removed it.

"known-flags" SP FlagList NL

[Exactly once.]

A space-separated list of all of the flags that this doc
"known" either because the authority knows about t
or because enough votes were counted for the cons
have been formed about their status.

"flag-thresholds" SP Thresholds NL

[At most once for votes; does not occur in consensu

```
A space-separated list of the intern
that the directory authority had at
a vote.

The metaformat is:
   Thresholds = Threshold | Threshol
   Threshold = ThresholdKey '=' Thre
   ThresholdKey = (KeywordChar | "_'
   ThresholdVal = [0-9]+("."[0-9]+)?

Commonly used Thresholds at this po

"stable-uptime" -- Uptime (in second
                   to be marked as s

"stable-mtbf" -- MTBF (in seconds) r
                 marked as stable.

"enough-mtbf" -- Whether we have mea
                 at stable-mtbf inst

"fast-speed" -- Bandwidth (in bytes
                a relay to be marked

"guard-wfu" -- WFU (in seconds) requ
               marked as guard.

"guard-tk" -- Weighted Time Known (
              relay to be marked as

"guard-bw-inc-exits" -- If exits car
                        must have a

"guard-bw-exc-exits" -- If exits car
                        must have a

"ignoring-advertised-bws" -- 1 if we
                        that we'll
                        claims of re
bandwidth.
```

"recommended-client-protocols" SP Entries NL "reco
NL "required-client-protocols" SP Entries NL "requir

[At most once for each.]

The "proto" element as specified in section 2.1.1.

To vote on these entries, a protocol/version combin
majority of the voters.

These lines should be voted on. A majority of votes
supported. A supermajority of authorities (2/3) are r

The required protocols should not be torrc-configur
in the Tor code.

The tor-spec.txt section 9 details how a relay and a c
encounter these lines in the consensus.

"params" SP [Parameters] NL

[At most once]

Parameter ::= Keyword '=' Int32 Int32 ::= A decimal i
2147483647. Parameters ::= Parameter | Parameter

The parameters list, if present, contains a space-sep
pairs, sorted in lexical order by their keyword (as AS
its own meaning.

(Only included when the vote is generated with cons

See param-spec.txt for a list of parameters and thei

"shared-rand-previous-value" SP NumReveals SP Va

[At most once]

NumReveals ::= An integer greater or equal to 0. Val

The shared_random_value that was generated durir
randomness protocol run. For example, if this docur
November, this field carries the shared random valu
of the 3rd of November.

See section [SRCALC] of srv-spec.txt for instructions
see section [CONS] for why we include old shared ra

Value is the actual shared random value encoded in
long. NumReveals is the number of commits used to

"shared-rand-current-value" SP NumReveals SP Valu

[At most once]

NumReveals ::= An integer greater or equal to 0. Val

The shared_random_value that was generated durir
protocol run. For example, if this document was cre
field carries the shared random value generated du
November

See section [SRCALC] of srv-spec.txt for instructions
the active commits.

Value is the actual shared random value encoded in
long. NumReveals is the number of commits used to

"bandwidth-file-headers" SP KeyValues NL

[At most once for votes; does not occur in consensu

KeyValues ::= "" | KeyValue | KeyValues SP KeyValue
::= ArgumentCharValue+ ArgumentCharValue ::= any
and SP.

The headers from the bandwidth file used to genera
headers are described in bandwidth-file-spec.txt.

If an authority is not configured with a V3Bandwidth
its vote.

If an authority is configured with a V3BandwidthsFil
appear in its vote, but without any headers.

First-appeared: Tor 0.3.5.1-alpha.

"bandwidth-file-digest" 1*(SP algorithm "=" digest) N

[At most once for votes; does not occur in consensu

A digest of the bandwidth file used to generate this
hash algorithm producing "digest", which can be "sh
is the base64 encoding of the hash of the bandwidth

If an authority is not configured with a V3Bandwidth
its vote.

If an authority is configured with a V3BandwidthsFil
appear in its vote, with the digest(s) of the unparsea

First-appeared: Tor 0.4.0.4-alpha

The authority section of a vote contains the followin
authority's current key certificate:

```
"dir-source" SP nickname SP identity SP a
   orport NL

    [Exactly once, at start]
```

Describes this authority. The nickname is a convenie identity is an uppercase hex fingerprint of the autho key. The address is the server's hostname. The IP is dirport is its current directory port. The orport is the authority listens for OR connections.

"contact" SP string NL

[Exactly once]

An arbitrary string describing how to contact the dir Administrators should include at least an email add

"legacy-dir-key" SP FINGERPRINT NL

[At most once]

Lists a fingerprint for an obsolete *identity* key still us clients working. This option is used to keep key arou authorities need to migrate many identity keys at or happen because of a security vulnerability that affe Debian OpenSSL RNG bug of May 2008.)

"shared-rand-participate" NL

[At most once]

Denotes that the directory authority supports and c protocol.

"shared-rand-commit" SP Version SP AlgName SP Id

[Any number of times]

Version ::= An integer greater or equal to 0. AlgNam Identity ::= 40* HEXDIG Commit ::= Base64-encoded

Denotes a directory authority commit for the share the commitment value and potentially also the reve [COMMITREVEAL] and [VALIDATEVALUES] of srv-spe these values.

Version is the current shared randomness protocol algorithm that is used (e.g. "sha3-256") and Identity fingerprint. Commit is the encoded commitment val it's set, it contains the reveal value in base64.

If a vote contains multiple commits from the same a consider the first commit listed.

"shared-rand-previous-value" SP NumReveals SP Va

[At most once]

See shared-rand-previous-value description above.

"shared-rand-current-value" SP NumReveals SP Valu

[At most once]

See shared-rand-current-value description above.

The authority section of a consensus contains group given, with one group for each authority that contrib sorted by authority identity digest:

```
"dir-source" SP nickname SP identity SP a
   orport NL

    [Exactly once, at start]

    As in the authority section of a vote

"contact" SP string NL

    [Exactly once.]

    As in the authority section of a vote

"vote-digest" SP digest NL

    [Exactly once.]
```

A digest of the vote from the authority that contribu is, not including the signature). (Hex, upper-case.)

For each "legacy-dir-key" in the vote, there is an add that legacy key's fingerprint, the authority's nicknam other fields as in the main "dir-source" line for that a not have corresponding "contact" or "vote-digest" e

Each router status entry contains the following item ascending order by identity digest.

```
"r" SP nickname SP identity SP digest SP
    SP DirPort NL

    [At start, exactly once.]
```

"Nickname" is the OR's nickname. "Identity" is a has
base64, with trailing equals sign(s) removed. "Digest
descriptor as signed (that is, not including the signat
section 1.3.), encoded in base64.

"Publication" was once the publication time of the r
form YYYY-MM-DD HH:MM:SS, in UTC. Now it is only
fixed value in consensus documents. Implementatic
vote documents.

"IP" is its current IP address; ORPort is its current OI
port, or "0" for "none".

"a" SP address ":" port NL

[Any number]

The first advertised IPv6 address for the OR, if it is r

Present only if the OR advertises at least one IPv6 a
the first advertised address is reachable. Any other
ignored.

Address and port are as for "or-address" as specifie

(Only included when the vote or consensus is gener
later.)

"s" SP Flags NL

[Exactly once.]

A series of space-separated status flags, in lexical or
documented flags are:

```
          "Authority" if the router is a dire
          "BadExit" if the router is believed
             (because its ISP censors it, bec
             proxy, or for some similar reaso
          "Exit" if the router is more useful
             general-purpose exit circuits th
             path building algorithm uses thi
          "Fast" if the router is suitable fo
          "Guard" if the router is suitable f
          "HSDir" if the router is considered
          "MiddleOnly" if the router is consi
             usage other than as a middle rel
             to handle this option, since whe
             will automatically vote against
             usable in other positions. (Sinc
          "NoEdConsensus" if any Ed25519 key
             microdescriptor does not reflect
          "Stable" if the router is suitable
          "StaleDesc" if the router should up
             the old one is too old.
          "Running" if the router is currentl
             ORPorts. (Authorities ignore IPv
             check IPv6 reachability.) Relays
             from the consensus, and current
             assume that every listed relay h
          "Valid" if the router has been 'val
             0.2.9.4-alpha would not use rout
             default. Currently, relays witho
             from the consensus, and current
             assume that every listed relay h
          "V2Dir" if the router implements th
             higher.

    "v" SP version NL

        [At most once.]
```

The version of the Tor protocol that this relay is run
the rest of the string is a Tor version number, and th
supported by the given version of Tor." Otherwise, i
string, Tor has upgraded to a more sophisticated pr
protocol is "a version of the Tor protocol more recen

Directory authorities SHOULD omit version strings t
would cause "v" lines to be over 128 characters long

"pr" SP Entries NL

[At most once.]

The "proto" family element as specified in section 2.

During voting, authorities copy these lines immediat

descriptor does not contain a "proto" entry, the auth
the approach described below in section D. They are
same rules as currently used for "v" lines, if a suffici

"w" SP "Bandwidth=" INT [SP "Measured=" INT] [SP "

[At most once.]

An estimate of the bandwidth of this relay, in an arb
second). Used to weight router selection. See sectio
Bandwidth is determined in a consensus.

Additionally, the Measured= keyword is present in v
measurement authorities to indicate a measured ba
measuring stream capacities. It does not occur in co

'Bandwidth=' and 'Measured=' values must be betw

The "Unmeasured=1" value is included in consensus
when the 'Bandwidth=' value is not based on a thres
this relay.

Other weighting keywords may be added later. Clien
recognize.

"p" SP ("accept" / "reject") SP PortList NL

[At most once.]

PortList = PortOrRange PortList = PortList "," PortOr

A list of those ports that this router supports (if 'acc
for exit to "most addresses".

"m" SP methods 1*(SP algorithm "=" digest) NL

[Any number, only in votes.]

Microdescriptor hashes for all consensus methods t
use the same microdescriptor format. "methods" is
consensus methods that the authority believes will p
name of the hash algorithm producing "digest", whi
depending on the consensus "methods" supporting
encoding of the hash of the router's microdescripto

```
"id" SP "ed25519" SP ed25519-identity NL
"id" SP "ed25519" SP "none" NL
   [vote only, at most once]

"stats" SP [KeyValues] NL

   [At most once. Vote only]
```

KeyValue ::= Keyword '=' Number Number ::= [0-9]+(
KeyValues SP KeyValue

Line containing various statistics that an authority h
is represented as a key + value. Reported keys are:

```
"wfu"  - Weighted Fractional Uptime
"tk"   - Weighted Time Known
"mtbf" - Mean Time Between Failure

(As of tor-0.4.6.1-alpha)
```

The footer section is delineated in all votes and cons
method 9 and above with the following:

"directory-footer" NL [No extra arguments]

It contains two subsections, a bandwidths-weights li
consensus method 9, footers only contained directo
footer' line or bandwidth-weights.)

The bandwidths-weights line appears At Most Once
votes.

"bandwidth-weights" [SP Weights] NL

Weight ::= Keyword '=' Int32 Int32 ::= A decimal integ
2147483647. Weights ::= Weight | Weights SP Weigh

List of optional weights to apply to router bandwidth
sorted in lexical order (as ASCII byte strings) and val
"bwweightscale" param. Definition of our known en

```
Wgg - Weight for Guard-flagged nodes
Wgm - Weight for non-flagged nodes
Wgd - Weight for Guard+Exit-flagged

Wmg - Weight for Guard-flagged nodes
Wmm - Weight for non-flagged nodes
Wme - Weight for Exit-flagged nodes
Wmd - Weight for Guard+Exit flagged

Weg - Weight for Guard flagged nodes
Wem - Weight for non-flagged nodes
Wee - Weight for Exit-flagged nodes
Wed - Weight for Guard+Exit-flagged

Wgb - Weight for BEGIN_DIR-supportir
Wmb - Weight for BEGIN_DIR-supportir
Web - Weight for BEGIN_DIR-supportir
Wdb - Weight for BEGIN_DIR-supportir

Wbg - Weight for Guard flagged nodes
Wbm - Weight for non-flagged nodes
Wbe - Weight for Exit-flagged nodes
Wbd - Weight for Guard+Exit-flagged

These values are calculated as specif
```

The signature contains the following item, which ap
Least Once for a consensus.

```
"directory-signature" [SP Algorithm] SP
    NL Signature
```

This is a signature of the status document, with the
and the signature item "directory-signature", using t
the hash through the *space* after directory-signature
authorities sign the same thing.) "identity" is the he
identity key of the signing authority, and "signing-ke
the current authority signing key of the signing auth

The Algorithm is one of "sha1" or "sha256" if it is pre
directory-signature entries with an unrecognized Alg
Algorithm is given. The algorithm describes how to
before signing it.

"ns"-flavored consensus documents must contain o
microdescriptor documents may contain other sign
signature from each authority should be "counted"
signed the consensus.

(Tor clients before 0.2.3.x did not understand the 'al

Assigning flags in a vote

(This section describes how directory authorities ch
routers. Later directory authorities MAY do things d
working well. Clients MUST NOT depend on the exa

In the below definitions, a router is considered "acti
hibernating.

When we speak of a router's bandwidth in this secti
bandwidth, or its advertised bandwidth. If a sufficie
MinMeasuredBWsForAuthToIgnoreAdvertised, 500
bandwidth values, then the authority bases flags on
nodes with non-measured bandwidths as if their ba
uses measured bandwidths for nodes that have the
other nodes.

When computing thresholds based on percentiles o
nodes that are active, that have not been omitted a
bandwidth is at least 4 KB. Nodes that don't meet th
threshold calculations (including calculation of stabi
thresholds) and also do not have their Exit status ch

"Valid" -- a router is 'Valid' if it is running a version o
directory authority has not blacklisted it as suspicio

```
"Named" --
"Unnamed" -- Directory authorities no long
   They were once used to determine whethe
   canonically linked to its public key.
```

"Running" -- A router is 'Running' if the authority ma
within the last 45 minutes on all its published ORPo

```
  * the IPv4 ORPort in the "r" line, and
  * the IPv6 ORPort considered for the "a'
    * the router advertises at least one I
    * AuthDirHasIPv6Connectivity 1 is set
```

A minority of voting authorities that set AuthDirHas
unreachable IPv6 ORPorts from the full consensus.
IPv6 ORPorts in the microdesc consensus, so that a
ORPorts from all consensus flavors. Consensus met
microdescriptors.

"Stable" -- A router is 'Stable' if it is active, and eithe

median for known active routers or its Weighted MT
Routers are never called Stable if they are running a
stupidly. (0.1.1.10-alpha through 0.1.1.16-rc are stup

To calculate weighted MTBF, compute the weighted
when the router was observed to be up, weighting i
the amount of time that has passed since the interv
that measurements over approximately one month
MTBF much.

[XXXX what happens when we have less than 4 days

"Exit" -- A router is called an 'Exit' iff it allows exits to
each of ports 80 and 443. (Up until Tor version 0.3.2
at least two of the ports 80, 443, and 6667.)

"Fast" -- A router is 'Fast' if it is active, and its bandw
known active routers or at least 100KB/s.

"Guard" -- A router is a possible Guard if all of the fo

```
        - It is Fast,
        - It is Stable,
        - Its Weighted Fractional Uptime is at
          active routers,
        - It is "familiar",
        - Its bandwidth is at least AuthDirGua
          default), OR its bandwidth is among
        - It qualifies for the V2Dir flag as d
          constraint was added in 0.3.3.x, bec
          started avoiding guards that didn't
```

To calculate weighted fractional uptime, compute th
in any given day, weighting so that downtime and up

A node is 'familiar' if 1/8 of all active nodes have app
been around for a few weeks.

"Authority" -- A router is called an 'Authority' if the a
status document believes it is an authority.

"V2Dir" -- A router supports the v2 directory protoco
port OR a tunnelled-dir-server line in its router desc
the directory protocol that supports the functionalit
supported version of Tor supports the functionality
might set "DirCache 0" or set really low rate limiting
directory mirror, i.e. they will omit the tunnelled-dir-

"HSDir" -- A router is a v2 hidden service directory if

descriptors, has the Stable and Fast flag, and the au
least 96 hours (or the current value of MinUptimeHi

"MiddleOnly" -- An authority should vote for this flag
for use except as a middle relay. When voting for th
against "Exit", "Guard", "HsDir", and "V2Dir". When v
votes on the "BadExit" flag, the authority should vot
added in 0.4.7.2-alpha.)

"NoEdConsensus" -- authorities should not vote on t
consensus for consensus method 22 or later.

"StaleDesc" -- authorities should vote to assign this f
descriptor is over 18 hours in the past. (This flag wa:

"Sybil" -- authorities SHOULD NOT accept more than
happens, the authority *should* vote for the excess re
Valid flags and instead should assign the "Sybil" flag
AuthDirMaxServersPerAddr) relays to choose from,
authorities to non-authorities, then prefer Running
bandwidth to low-bandwidth relays. In this compari:
unless it is not present for a router, in which case ad

Thus, the network-status vote includes all non-black
descriptors.

The bandwidth in a "w" line should be taken as the b
capacity that the authority has. For now, this should
bandwidth and bandwidth rate limit from the server
second, and capped at some arbitrary value (curren

The Measured= keyword on a "w" line vote is curren
previous published consensus bandwidth by the rat
stream capacity to the network average. If 3 or more
keyword for a router, the authorities produce a cons
keyword equal to the median of the Measured= vote

As a special case, if the "w" line in a vote is about a r
not include a Measured= keyword. The goal is to lea
Unmeasured, so they can reserve their attention for
for votes about authorities may include the bandwid
different keyword, e.g. MeasuredButAuthority=, so i
for posterity.

The ports listed in a "p" line should be taken as thos
policy permits 'most' addresses, ignoring any accept
rejects for private netblocks. "Most" addresses are p

addresses (two /8 networks) were blocked. The list i
3.8.2.

Serving bandwidth list fi

If an authority has used a bandwidth list file to gene
make it available at

```
http://<hostname>/tor/status-vote/next/bandwi
```

at the start of each voting period.

It MUST NOT attempt to send its bandwidth list file i
and it SHOULD NOT make bandwidth list files from

If an authority makes this file available, it MUST be t
vote document available at

```
http://<hostname>/tor/status-vote/next/author
```

To avoid inconsistent reads, authorities SHOULD on
voting period. Further processing and serving SHOU

The bandwidth list format is described in bandwidth

The standard URLs for bandwidth list files first-appe

Downloading informatio directory authorities

Downloading missing certificate authorities

XXX when to download certificates.

Downloading server descriptors authorities

Periodically (currently, every 10 seconds), directory a any specific descriptors that they do not have and th download. Authorities identify them by hash in vote than the descriptor we currently have).

[XXXX need a way to fetch descriptors ahead of the v now.]

If so, the directory authority launches requests to th such that each authority is only asked for descriptor more than one authority lists the descriptor, we cho

If one of these downloads fails, we do not try to dow authority that failed to serve it again unless we rece (consensus or vote) from that authority that lists the

```
    Directory authorities must potentially cac
    router. Authorities must not discard any d
    consensus.  If there is enough space to st
    authorities SHOULD try to hold those which
 the
    most.  (Currently, this is judged based on
    descriptor seemed newest.)
 [XXXX define recent]
```

Authorities SHOULD NOT download descriptors for reject for reasons listed in section 3.2.

Downloading extra-info docume directory authorities

Periodically, an authority checks whether it is missir words, if it has any server descriptors with an extra- any of the extra-info documents currently held. If sc documents are missing. We follow the same splittin,

Computing a consensus votes

Given a set of votes, authorities compute the conter

The consensus status, along with as many signature section 3.10 below), should be available at

```
http://<hostname>/tor/status-vote/next/consen
```

The contents of the consensus document are as foll

The "valid-after", "valid-until", and "fresh-until" time respective values from all the votes.

The times in the "voting-delay" line are taken as the DistSeconds times in the votes.

Known-flags is the union of all flags known by any v

Entries are given on the "params" line for every keyv authorities (total authorities, not just those particip least three authorities voted for that parameter. Th all votes on that keyword.

(In consensus methods 7 to 11 inclusive, entries we keyword on which *any* authority voted, the value giv on that keyword.)

```
"client-versions" and "server-versions" a
 order; A version is recommended in the c
 by more than half of the voting authorit
 client-versions or server-versions lines
```

With consensus methods 19 through 33, a package PACKAGENAME/VERSION pair if at least three autho votes. (Call these lines the "input" lines for PACKAGE every "package" line that is listed verbatim by more line for the PACKAGENAME/VERSION pair, and no ot

The authority item groups (dir-source, contact, finge votes of the voting authorities. These groups are so identity keys, in ascending order. If the consensus m must be included for every vote with legacy-key entr the voter's ordinary nickname with the string "-lega from the original vote's dir-source line.

```
        A router status entry:
           * is included in the result if some r
             identity is included by more than h
             authorities, not just those whose v
             (Consensus method earlier than 21)

           * is included according to the rules
             3.8.0.2 below. (Consensus method 22

           * For any given RSA identity digest,
             one router status entry.

           * For any given Ed25519 identity, we
             status entry.

           * A router entry has a flag set if th
             of the authorities who care about t

           * Two router entries are "the same" i
             (descriptor digest, published time,
             We choose the tuple for a given rou
             for that router in the most votes.
             the more recently published, then t
             descriptor digest.

          [
           * The Named flag appears if it is inc
             _any_ authority, and if all authori
             nickname. However, if consensus-met
             any authority calls this identity/n
             this routerstatus does not get the

           * If consensus-method 2 or later is i
             set for a routerstatus if any autho
different
             identities to be Named with that ni
             lists that nickname/ID pair as Unna

             (With consensus-method 1, Unnamed i

             [But note that authorities no longe
             and the above two bulletpoints are
          ]

           * The version is given as whichever v
             voters, with ties decided in favor

           * If consensus-method 4 or later is i
             do not have the Running flag are no

           * If consensus-method 5 or later is i
             is generated using a low-median of
             the votes that included "w" lines f

           * If consensus-method 5 or later is i
             is taken from the votes that have t
```

```
  for the descriptor we are listing.
  same.  If they are not, we pick the
  one, breaking ties in favor of the
  vote.)  The port list is encoded as

* If consensus-method 6 or later is 
  authorities provide a Measured= key
  a router, the authorities produce a
  Bandwidth= keyword equal to the med

* If consensus-method 7 or later is 
  included in the output.

* If the consensus method is under 11
  possible exits when computing bandw
  method 11 or later is in use, any r
  the BadExit flag doesn't count when

* If consensus method 12 or later is 
  parameters that more than half of t
  authorities voted for are included 

[ As of 0.2.6.1-alpha, authorities no
  any consensus methods lower than 13

* If consensus method 13 or later is 
  omit any router for which no microd

* If consensus method 14 or later is 
  microdescriptors may include an "a"
  an IPv6 OR port.

* If consensus method 15 or later is 
  include "p6" lines including IPv6 e

* If consensus method 16 or later is 
  are included in microdescriptors

* If consensus method 17 or later is 
  maximum on the Bandwidth= values th
  line for any router that doesn't ha
  bandwidth values in votes. They als
  flag to such 'w' lines.

* If consensus method 18 or later is 
  "id" lines in microdescriptors. Th

* If consensus method 19 or later is 
  "package" lines in consensuses.

* If consensus method 20 or later is 
  GuardFraction information in microd

* If consensus method 21 or later is 
  an "id" line for ed25519 identities

[ As of 0.2.8.2-alpha, authorities no
```

```
  consensus method 21, because it co

* If consensus method 22 or later is
  produce a majority consensus about
  3.8.0.1 below), the consensus must
  the "s" line for every relay whose
  consensus.

* If consensus method 23 or later is
  shared randomness protocol data on

* If consensus-method 24 or later is
  do not have the Valid flag are not

[ As of 0.3.4.1-alpha, authorities no
  any consensus methods lower than 25

* If consensus-method 25 or later is
  on recommended-protocols and requi
  consensus.  We also include protoco
  entries.

* If consensus-method 26 or later is
  bandwidth weights to 1 in our calcu
  division-by-zero errors on unusual

* If consensus method 27 or later is
  may include an "a" line for each ro

[ As of 0.4.3.1-alpha, authorities no
  any consensus methods lower than 28

* If consensus method 28 or later is
  include "a" lines.

* If consensus method 29 or later is
  lines are canonicalized to improve

* If consensus method 30 or later is
  ntor-onion-key does not include the

* If consensus method 31 or later is
  "bwweightscale" and "maxunmeasured
  computing votes.

* If consensus method 32 or later is
  "MiddleOnly" flag specially when co
  voters agree to include "MiddleOnl
  automatically remove "Exit", "Guar
  the BadExit flag is included in th
  add it to the routerstatus.

* If consensus method 33 or later is
  flavor is "microdesc", then the "P
  line is set to "2038-01-01 00:00:0

* If consensus method 34 or later is
```

```
        does not include any "package" line
```

The signatures at the end of a consensus document
identity digest.

All ties in computing medians are broken in favor of

Deciding which Ids to include

This sorting algorithm is used for consensus-metho

```
First, consider each listing by tuple of <E
  may be "None" if the voter included "id e
  the authority knows what ed25519 identit
  key doesn't have one.

For each such <Ed, RSA> tuple that is liste
  total authorities (not just total votes),
  possible for any other <id-Ed, id-RSA'> t
  than half of the authorities list a singl
  consider that Ed key to be "consensus"; s
  NoEdConsensus flag.

Log any other id-RSA values corresponding t
  other id-Ed values corresponding to an id

For each <id-RSA> that is not yet included,
  half of the total authorities, and we do
  some <id-Ed>, include it, but do not cons
```

Deciding which descriptors to include

Deciding which descriptors to include.

A tuple belongs to an `<id-RSA, id-Ed>` identity if it
parts, or if it is an old tuple (one with no Ed opinion)
belongs to an `<id-RSA>` identity if its RSA identity m

A tuple matches another tuple if all the fields that ar
same.

For every included identity, consider the tuples belo
into sets of matching tuples. Include the tuple that r
in favor of the most recently published, and then in
digest.

Forward compatibility

Future versions of Tor will need to include new infor
but it is important that all authorities (or at least hal
consensus.

To achieve this, authorities list in their votes their su
consensuses from votes. Later methods will be assig
specified methods:

```
     "1" -- The first implemented version.
     "2" -- Added support for the Unnamed fla
     "3" -- Added legacy ID key support to a
     "4" -- No longer list routers that are r
     "5" -- adds support for "w" and "p" line
     "6" -- Prefers measured bandwidth values
     "7" -- Provides keyword=integer pairs of
     "8" -- Provides microdescriptor summarie
     "9" -- Provides weights for selecting fl
     "10" -- Fixes edge case bugs in router f
     "11" -- Don't consider BadExits when cal
     "12" -- Params are only included if enou
     "13" -- Omit router entries with missing
     "14" -- Adds support for "a" lines in ns
microdescriptors.
     "15" -- Adds support for "p6" lines.
     "16" -- Adds ntor keys to microdescripto
     "17" -- Adds "Unmeasured=1" flags to "w"
     "18" -- Adds 'id' to microdescriptors.
     "19" -- Adds "package" lines to consensu
     "20" -- Adds GuardFraction information t
     "21" -- Adds Ed25519 keys to microdescri
     "22" -- Instantiates Ed25519 voting algo
     "23" -- Adds shared randomness protocol
     "24" -- No longer lists routers that are
     "25" -- Vote on recommended-protocols an
     "26" -- Initialize bandwidth weights to
     "27" -- Adds support for "a" lines in mi
     "28" -- Removes "a" lines from microdesc
     "29" -- Canonicalizes families in microd
     "30" -- Removes padding from ntor-onion-
     "31" -- Uses correct parsing for bwweigh
             when computing weights
     "32" -- Adds special handling for Middle
     "33" -- Sets "publication" field in micr
             to a meaningless value.
     "34" -- Removes "package" lines from con
```

Before generating a consensus, an authority must d
use. To do this, it looks for the highest version numb
authorities voting. If it supports this method, then it
newest consensus method that it supports (which w

signed consensus).

All authorities MUST support method 25; authoritie
methods as well. Authorities SHOULD NOT support
before 25. Clients MAY assume that they will never s
for any method before method 25.

(The consensuses generated by new methods must
only understand the old methods, and must not cau
compromise their anonymity. This is a means for m
consensus; not for making backward-incompatible c

The following methods have incorrect implementati
advertise support for them:

"21" -- Did not correctly enable support for ed25519

Encoding port lists

Whether the summary shows the list of accepted po
depends on which list is shorter (has a shorter strin
choose the list of accepted ports. As an exception to
represented as "accept 1-65535" instead of "reject "
given as "reject 1-65535".

Summary items are compressed, that is instead of "
item of "80-100", similarly instead of "20,21" a sumn

Port lists are sorted in ascending order.

The maximum allowed length of a policy summary (
1000 characters. If a summary exceeds that length v
list as much of the port list as is possible within thes

Computing Bandwidth Weights

Let weight_scale = 10000, or the value of the "bwwe
consensus method 31 there was a bug in parsing bv
consensus parameters after it alphabetically, it wou
similar bug existed for "maxunmeasuredbw".)

Starting with consensus method 26, G, M, E, and D a

consensus methods initialize them all to 0. With this
small or new are much more likely to produce band
extra bandwidth has a negligible impact on the ban
network.

Let G be the total bandwidth for Guard-flagged nod
non-flagged nodes. Let E be the total bandwidth for
bandwidth for Guard+Exit-flagged nodes. Let T = G+

Let Wgd be the weight for choosing a Guard+Exit fo
weight for choosing a Guard+Exit for the middle po
choosing a Guard+Exit for the exit position.

Let Wme be the weight for choosing an Exit for the
weight for choosing a Guard for the middle position

Let Wgg be the weight for choosing a Guard for the
for choosing an Exit for the exit position.

Balanced network conditions then arise from soluti
equations:

Wgg*G* + *Wgd*D == M + Wmd*D* + *Wme*E + Wmg*G (guar*
*Wee*E + Wed*D (guard bw = exit bw) Wed*D + Wmd*D* + *W*
weight_scale) Wmg*G* + *Wgg*G == G (aka: Wgg = weigh
Wee = weight_scale-Wme)

We are short 2 constraints with the above set. The r
examining different cases of network load. The follo
consensus method 10 and above. There are anothe
constraints used for these same cases in consensus
in Tor 0.2.2.10-alpha to 0.2.2.16-alpha.

Case 1: E >= T/3 && G >= T/3 (Neither Exit nor Guar

In this case, the additional two constraints are: Wm

```
    This leads to the solution:
        Wgd = weight_scale/3
        Wed = weight_scale/3
        Wmd = weight_scale/3
        Wee = (weight_scale*(E+G+M))/(3*E)
        Wme = weight_scale - Wee
        Wmg = (weight_scale*(2*G-E-M))/(3*G)
        Wgg = weight_scale - Wmg

  Case 2: E < T/3 && G < T/3 (Both are scarc
```

Let R denote the more scarce class (Rare) between scarce class.

Subcase a: R+D < S

In this subcase, we simply devote all of D bandwidth

```
Wgg = Wee = weight_scale
Wmg = Wme = Wmd = 0;
if E < G:
  Wed = weight_scale
  Wgd = 0
else:
  Wed = 0
  Wgd = weight_scale
```

```
Subcase b: R+D >= S
```

In this case, if M <= T/3, we have enough bandwidth condition.

Add constraints Wgg = weight_scale, Wmd == Wgd t position while still allowing exits to be used as midd

Wee = (weight_scale*(E - G + M))/E Wed = (weight_sc (weight_scale*(G-M))/E Wmg = 0 Wgg = weight_scale (weight_scale - Wed)/2

If this system ends up with any values out of range (use the constraints Wgg == weight_scale and Wee = positions are scarce:

```
            Wgg = weight_scale
            Wee = weight_scale
            Wed = (weight_scale*(D - 2*E + G + M
            Wmd = (weight_Scale*(D - 2*M + G + E
            Wme = 0
            Wmg = 0
            Wgd = weight_scale - Wed - Wmd

         If M > T/3, then the Wmd weight above w
         in this case:
            Wmd = 0
            Wgd = weight_scale - Wed

     Case 3: One of E < T/3 or G < T/3

       Let S be the scarce class (of E or G).

       Subcase a: (S+D) < T/3:
         if G=S:
           Wgg = Wgd = weight_scale;
           Wmd = Wed = Wmg = 0;
           // Minor subcase, if E is more scarce
           // keep its bandwidth in place.
           if (E < M) Wme = 0;
           else Wme = (weight_scale*(E-M))/(2*E)
           Wee = weight_scale-Wme;
         if E=S:
           Wee = Wed = weight_scale;
           Wmd = Wgd = Wme = 0;
           // Minor subcase, if G is more scarce
           // keep its bandwidth in place.
           if (G < M) Wmg = 0;
           else Wmg = (weight_scale*(G-M))/(2*G)
           Wgg = weight_scale-Wmg;

       Subcase b: (S+D) >= T/3
         if G=S:
           Add constraints Wgg = weight_scale, W
           in the guard position, while still al
           used as middle nodes:
             Wgg = weight_scale
             Wgd = (weight_scale*(D - 2*G + E +
             Wmg = 0
             Wee = (weight_scale*(E+M))/(2*E)
             Wme = weight_scale - Wee
             Wmd = (weight_scale - Wgd)/2
             Wed = (weight_scale - Wgd)/2
         if E=S:
           Add constraints Wee == weight_scale,
           in the exit position:
             Wee = weight_scale;
             Wed = (weight_scale*(D - 2*E + G +
             Wme = 0;
             Wgg = (weight_scale*(G+M))/(2*G);
             Wmg = weight_scale - Wgg;
             Wmd = (weight_scale - Wed)/2;
```

```
        Wgd = (weight_scale - Wed)/2;
```

To ensure consensus, all calculations are performed
precision determined by the bwweightscale consens
1, Max: INT32_MAX). (See note above about parsing
consensus method 31.)

For future balancing improvements, Tor clients sup
directory requests and middle weighting. These wei
with the exception of the following groups of assign

Directory requests use middle weights:

Wbd=Wmd, Wbg=Wmg, Wbe=Wme, Wbm=Wmm

Handle bridges and strange exit policies:

Wgm=Wgg, Wem=Wee, Weg=Wed

Computing consensus flavors

Consensus flavors are variants of the consensus tha
use instead of the unflavored consensus. The purpo
or replace information in the unflavored consensus
information they would not use anyway.

Directory authorities can produce and serve an arbi
consensus. A downside of creating too many new fl
distinguishable based on which flavor they downloa
when adding a field instead wouldn't be too onerou

Examples for consensus flavors include:

```
      - Publishing hashes of microdescriptors
        full descriptors (see section 3.9.2).
      - Including different digests of descr
        perhaps-soon-to-be-totally-broken SHA
```

Consensus flavors are derived from the unflavored
complete. This is to avoid consensus synchronizatio

Every consensus flavor has a name consisting of a s
alphanumeric characters and dashes. For compatibi
consensus type is called "ns".

The supported consensus flavors are defined as par
method.

All consensus flavors have in common that their firs
where version is 3 or higher, and the flavor is a strin
characters and dashes:

"network-status-version" SP version [SP flavor] NL

ns consensus

The ns consensus flavor is equivalent to the unflavo
omitted from the "network-status-version" line, it sh
implementations may explicitly state that the flavor
but should accept consensuses where the flavor is c

Microdescriptor consensus

The microdescriptor consensus is a consensus flavo
hashes instead of descriptor hashes and that omits
contained in microdescriptors. The microdescriptor
elements that are small and frequently changing. Cl
microdescriptor consensus to decide which servers
servers to fetch information from.

The microdescriptor consensus is based on the unfl
as follows:

"network-status-version" SP version SP "microdesc"

[At start, exactly once.]

The flavor name of a microdescriptor consensus is "

Changes to router status entries are as follows:

```
    "r" SP nickname SP identity SP publicatio
        SP DirPort NL

        [At start, exactly once.]

        Similar to "r" lines in section 3.4.1
element.

    "a" SP address ":" port NL

        [Any number]

        Identical to the "r" lines in section
```

(Only included when the vote is generated with cons
consensus is generated with consensus-method 27

"p" ... NL

[At most once]

Not currently generated.

Exit policy summaries are contained in microdescrip
microdescriptor consensus.

"m" SP digest NL

[Exactly once.*]

"digest" is the base64 of the SHA256 hash of the rou
omitted. For a given router descriptor digest and co
a single microdescriptor digest in the "m" lines of all
different microdescriptor digests for the same descr
at least one of the authorities is broken. If this happ
contain whichever microdescriptor digest is most co
break ties in the favor of the lexically earliest.

[*Before consensus method 13, this field was somet

Additionally, a microdescriptor consensus SHOULD
its signatures.

Exchanging detached sig

Once an authority has computed and signed a cons
its detached signature to each other authority in an

```
http://<hostname>/tor/post/consensus-signatur
```

[XXX Note why we support push-and-then-pull.]

All of the detached signatures it knows for consensu

```
http://<hostname>/tor/status-vote/next/consen
```

Assuming full connectivity, every authority should c
including any flavors in each period. Therefore, it isr
consensus or any flavors of it computed by each aut
push/fetch each others' signatures. A "detached sig
follows:

"consensus-digest" SP Digest NL

[At start, at most once.]

The digest of the consensus being signed.

"valid-after" SP YYYY-MM-DD SP HH:MM:SS NL "fres
NL "valid-until" SP YYYY-MM-DD SP HH:MM:SS NL

[As in the consensus]

"additional-digest" SP flavor SP algname SP digest N

[Any number.]

For each supported consensus flavor, every directo
"additional-digest" lines. "flavor" is the name of the
name of the hash algorithm that is used to generate
encoded digest.

The hash algorithm for the microdescriptor consens
algname "sha256".

```
    "additional-signature" SP flavor SP algna
        signing-key-digest NL signature.

       [Any number.]
```

For each supported consensus flavor and defined d

authority adds an "additional-signature" line. "flavor
"algname" is the name of the algorithm that was us
keys, and to compute the signature. "identity" is the
identity key of the signing authority, and "signing-ke
the current authority signing key of the signing auth

The "sha256" signature format is defined as the RSA
SHA256 digest of the item to be signed. When check
be treated as valid if the signature material begins v
data can get added later. [To be honest, I didn't fully
and only copied it from the proposals. Review caref

"directory-signature"

[As in the consensus; the signature object is the sam

Publishing the signed co

The voting period ends at the valid-after time. If the
majority of authorities, these documents are made

```
http://<hostname>/tor/status-vote/current/con
```

and

```
http://<hostname>/tor/status-vote/current/con
```

```
    [XXX current/consensus-signatures is not
     is not used in the voting protocol.]

    [XXX possible future features include supp
     consensuses.]

    The other vote documents are analogously m

 http://<hostname>/tor/status-vote/current/aut
 http://<hostname>/tor/status-vote/current/<fp
 http://<hostname>/tor/status-vote/current/d/<
 http://<hostname>/tor/status-vote/current/ban
```

once the voting period ends, regardless of the numb

The authorities serve another consensus of each fla

```
 /tor/status-vote/(current|next)/consensus-F.z
 /tor/status-vote/(current|next)/consensus-F/<
```

The standard URLs for bandwidth list files first-appe

Directory cache operati

All directory caches implement this section, except a

Downloading consensus status c directory authorities

All directory caches try to keep a recent network-sta clients. A cache ALWAYS downloads a network-statu are true:

- The cache has no consensus document.
- The cache's consensus document is no longer

Otherwise, the cache downloads a new consensus c in the first half-interval after its current consensus s at random to avoid swarming the authorities at the is inferred from the difference between the valid-aft the consensus.)

```
[For example, if a cache has a consensus t
 and is fresh until 2:00, that cache will
 a random time between 2:00 and 2:30.]
```

Directory caches also fetch consensus flavors from t correctness of consensus flavors, but do not check a consensus document beyond its digest and length. (from the same locations as the directory authorities

Downloading server descriptors authorities

Periodically (currently, every 10 seconds), directory c specific descriptors that they do not have and that t download. Caches identify these descriptors by hasl consensus documents.

If so, the directory cache launches requests to the a

If one of these downloads fails, we do not try to dov

authority that failed to serve it again unless we rece
that lists the same descriptor.

Directory caches must potentially cache multiple de
must not discard any descriptor listed by any recent
to store additional descriptors, caches SHOULD try t
to download the most. (Currently, this is judged bas
descriptor seemed newest.)

[XXXX define recent]

Downloading microdescriptors f authorities

Directory mirrors should fetch, cache, and serve eac
authorities.

The microdescriptors with base64 hashes `<D1>` , `<D`

```
http://<hostname>/tor/micro/d/<D1>-<D2>-<D3>[
```

`<Dn>` are base64 encoded with trailing =s omitted f
microdescriptor consensus format. -s are used inste
+ character is used in base64 encoding.

Directory mirrors should check to make sure that th
serve match the right hashes (either the hashes fror
the consensus, respectively).

(NOTE: Due to squid proxy url limitations at most 92
retrieved in a single request.)

Downloading extra-info docume authorities

Any cache that chooses to cache extra-info docume

Periodically, the Tor instance checks whether it is m
other words, if it has any server descriptors with an
match any of the extra-info documents currently he
info documents are missing. Caches download from

splitting and back-off rules as in section 4.2.

Consensus diffs

Instead of downloading an entire consensus, clients
containing an ed-style diff from a previous consensu
make these diffs as they learn about new consensus
record of older consensuses.

(Support for consensus diffs was added in 0.3.1.1-al
DirCache protocol version "2" or later.)

Consensus diff format

Consensus diffs are formatted as follows:

The first line is "network-status-diff-version 1" NL

The second line is

"hash" SP FromDigest SP ToDigest NL

where FromDigest is the hex-encoded SHA3-256 dig
consensus that the diff should be applied to, and To
digest of the *entire* consensus resulting from applyi
on that part of a consensus is signed.)

The third and subsequent lines encode the diff from
subset of the ed diff format, as specified in appendi

Serving and requesting diffs

When downloading the current consensus, a client r
form

X-Or-Diff-From-Consensus: HASH1, HASH2, ...

where the HASH values are hex-encoded SHA3-256
more consensuses that the client knows about.

If a cache knows a consensus diff from one of those
consensus of the requested flavor, it may send that
consensus.

Caches also serve diffs from the URIs:

```
/tor/status-vote/current/consensus/diff/<HASH
/tor/status-vote/current/consensus-<FLAVOR>/d
```

where FLAVOR is the consensus flavor, defaulting to
of recognized authority identity fingerprints as in ap

Retrying failed downloads

See section 5.5 below; it applies to caches as well as

Client operation

Every Tor that is not a directory server (that is, those
implements this section.

Downloading network-status do

Each client maintains a list of directory authorities. I
use the same list.

```
[Newer versions of Tor (0.2.8.1-alpha and l
 Each client also maintains a list of defau
 (fallbacks). Each released version of Tor
 depending on the mirrors that satisfy the
 release time.]
```

Clients try to have a live consensus network-status d
status document is "live" if the time in its valid-after
valid-until field has not passed.

When a client has no consensus network-status doc
randomly chosen fallback directory mirror or autho
authorities, trying them earlier and more frequently
downloads from caches randomly chosen from amo
servers. (This information comes from the network-

After receiving any response client MUST discard an
did not request.

On failure, the client waits briefly, then tries that net
another cache. The client does not build circuits unt
consensus document, and it has descriptors for a si
that it believes are running (this is configurable usin
parameters).

```
[Newer versions of Tor (0.2.6.2-alpha and l
 If the consensus contains Exits (the typic
 exit and internal circuits. When bootstrap
 to handle an application requesting an ex
 World Wide Web.
```

If the consensus does not contain Exits, Tor will only
earlier statuses will have included "internal" as indic
completes, Tor will be ready to handle an applicatio

hidden services at ".onion" addresses.

If a future consensus contains Exits, exit circuits ma

(Note: clients can and should pick caches based on t
have: once they have first fetched network-status in
should not need to go to the authority directly agair
at random, based on its consensus weight in the cu

To avoid swarming the caches whenever a consensu
consensuses at a randomly chosen time after the ca
consensus, but before their consensus will expire. (
random from the interval between the time 3/4 into
is no longer fresh, and 7/8 of the time remaining aft
invalid.)

```
[For example, if a client has a consensus
 and is fresh until 2:00, and expires at 4
 a new consensus at a random time between
 of the one-hour interval is 45 minutes, a
 minutes is 65 minutes.]
```

Clients may choose to download the microdescripto
network status consensus. In that case they should
the normal consensus. They should not download n

When a client does not have a live consensus, it will
consensus it has if that consensus is "reasonably liv
one that expired less than 24 hours ago.

Downloading server descriptors

Clients try to have the best descriptor for each route

- It is listed in the consensus network-status doc

Periodically (currently every 10 seconds) clients chec
"downloadable" descriptors. A descriptor is downloa

```
- It is the "best" descriptor for some
- The descriptor was published at least
  (This prevents clients from trying to
  mirrors have probably not yet retriev
- The client does not currently have it
- The client is not currently trying to
- The client would not discard it immed
- The client thinks it is running and v
```

If at least 16 known routers have downloadable des
10 minutes) has passed since the last time the client
launches requests for all downloadable descriptors.

When downloading multiple server descriptors, the
that:

```
  - At least 3 different mirrors are used,
    in more than one request for under 4 d
  - No more than 128 descriptors are reque
  - Otherwise, as few mirrors as possible
After choosing mirrors, the client divides
randomly.
```

After receiving any response the client MUST discard
request.

When a descriptor download fails, the client notes it
descriptor downloadable again until a certain amou
seconds for the first failure, 60 seconds for the seco
minutes for the fourth, and 1 day thereafter.) Period
reset the failure count.

Clients retain the most recent descriptor they have
as it is listed in the consensus. If it is not listed, they
(currently, ROUTER_MAX_AGE=48 hours) and no bet
downloaded for the same relay. Caches retain descr
OLD_ROUTER_DESC_MAX_AGE=5 days old.

Clients which chose to download the microdescripto
consensus must download the referenced microdes
Clients fetch and cache microdescriptors preemptiv
like they currently fetch descriptors. After bootstrap
microdescriptors that have changed.

When a client gets a new microdescriptor consensus
microdescriptors it needs to learn, and launches a r

Clients maintain a cache of microdescriptors along v

referenced by a consensus, and which identity key it
microdescriptor until it hasn't been mentioned in an
clients might cache them for longer or shorter times

Downloading extra-info docume

Any client that uses extra-info documents should im

Note that generally, clients don't need extra-info do

Periodically, the Tor instance checks whether it is m
other words, if it has any server descriptors with an
match any of the extra-info documents currently he
info documents are missing. Clients try to download
splitting and back-off rules as in section 5.2.

Retrying failed downloads

This section applies to caches as well as to clients.

When a client fails to download a resource (a consen
microdescriptor, etc) it waits for a certain amount of
To determine the amount of time to wait, clients use
algorithm. (Specifically, they use a variation of the "d
https://aws.amazon.com/blogs/architecture/expone

The specific formula used to compute the 'i+1'th del

```
Delay_{i+1} = MIN(cap, random_betweer
  where upper_bound = MAX(lower_bounc
        lower_bound = MAX(1, base_del
```

The value of 'cap' is set to INT_MAX; the value of 'ba
downloaded, whether the client is fully bootstrappe
where it is downloading from. Current base_delay va

```
Consensus objects, as a non-bridge cache:
      0 (TestingServerConsensusDownloadIn

Consensus objects, as a client or bridge 
      0 (TestingClientConsensusDownloadIn

Consensus objects, as a client or bridge 
when connecting to an authority because no
known:
      0 (ClientBootstrapConsensusAuthority

Consensus objects, as a client or bridge 
when "fallback" caches are known but conne
anyway:
      6 (ClientBootstrapConsensusAuthority

Consensus objects, as a client or bridge 
when downloading from a "fallback" cache.
      0 (ClientBootstrapConsensusFallbackD

Bridge descriptors, as a bridge-using clie
is usable:
      10800 (TestingBridgeDownloadInitialD

Bridge descriptors, otherwise:
      0 (TestingBridgeBootstrapDownloadIn

Other objects, as cache or authority:
      0 (TestingServerDownloadInitialDela

Other objects, as client:
      0 (TestingClientDownloadInitialDela
```

Standards compliance

All clients and servers MUST support HTTP 1.0. Clier
versions of HTTP as well.

HTTP headers

Servers SHOULD set Content-Encoding to the algori
document(s) being served. Recognized algorithms a

```
    - "identity"     -- RFC2616 section 3.5
    - "deflate"      -- RFC2616 section 3.5
    - "gzip"         -- RFC2616 section 3.5
    - "x-zstd"       -- The zstandard compre
    - "x-tor-lzma"   -- The lzma compressior
                        value no higher thar
```

Clients SHOULD use Accept-Encoding on most direc
above compression algorithms they support. If they
0.3.1.1-alpha), then the server should serve only "de
documents, based on the presence or absence of th

Note that for anonymous directory requests (that is
circuits, like those for onion service lookups) implen
any Accept-Encoding values other than deflate. To d
fingerprinting opportunity.

When receiving multiple documents, clients MUST a
documents and concatenated compressed docume

Servers MAY set the Content-Length: header. When
of compressed bytes that they are sending.

Servers MAY include an X-Your-Address-Is: header, v
address of the client connecting to them (as a dotte
tunneled over a BEGIN_DIR stream, servers SHOULD
carrying the BEGIN_DIR stream reached them.

Servers SHOULD disable caching of multiple networ
descriptors. Servers MAY enable caching of single de
the list of all server descriptors, a v1 directory, or a v
mention times.

HTTP status codes

Tor delivers the following status codes. Some were (
code SHOULD NOT rely on specific status codes yet.

```
200 -- the operation completed successfully
    -- the user requested statuses or serve
       requested were found (0.2.0.4-alpha

304 -- the client specified an if-modified-
       requested resources have changed sin

400 -- the request is malformed, or
    -- the URL is for a malformed variation
        or
    -- the client tried to post to a non-au
    -- the authority rejected a malformed p

404 -- the requested document was not found
    -- the user requested statuses or serve
       requested were found (0.2.0.5-alpha

503 -- we are declining the request in orde
    -- user requested some items that we or
       but we do not have any available.
```

Consensus-negotiation t

```
Period begins: this is the Published time.
  Everybody sends votes
Reconciliation: everybody tries to fetch m
  consensus may exist at this point.
End of voting period:
  everyone swaps signatures.
Now it's okay for caches to download
  Now it's okay for clients to download.

Valid-after/valid-until switchover
```

General-use HTTP URLs

"Fingerprints" in these URLs are base16-encoded SH

The most recent v3 consensus should be available a

```
http://<hostname>/tor/status-vote/current/con
```

Similarly, the v3 microdescriptor consensus should

```
http://<hostname>/tor/status-vote/current/con
```

Starting with Tor version 0.2.1.1-alpha is also availa

```
http://<hostname>/tor/status-vote/current/con
```

(NOTE: Due to squid proxy url limitations at most 96 single request.)

Where F1, F2, etc. are authority identity fingerprints return a consensus if more than half of the request document, otherwise a 404 error will be sent back. length of any multiple of two, using only the leftmos uses 3 bytes (6 hex characters) of the fingerprint.

Clients SHOULD sort the fingerprints in ascending o

Clients SHOULD use this format when requesting co authority servers and from caches running a version URL format.

A concatenated set of all the current key certificates

```
http://<hostname>/tor/keys/all.z
```

The key certificate for this server should be availabl

```
http://<hostname>/tor/keys/authority.z
```

The key certificate for an authority whose authority available at:

```
http://<hostname>/tor/keys/fp/<F>.z
```

The key certificate whose signing key fingerprint is

```
http://<hostname>/tor/keys/sk/<F>.z
```

The key certificate whose identity key fingerprint is
fingerprint is `<S>` should be available at:

```
http://<hostname>/tor/keys/fp-sk/<F>-<S>.z
```

(As usual, clients may request multiple certificates u

```
http://<hostname>/tor/keys/fp-sk/<F1>-<S1>+<F
```

[The above fp-sk format was not supported before T

The most recent descriptor for a server whose ident
should be available at:

```
http://<hostname>/tor/server/fp/<F>.z
```

The most recent descriptors for servers with identity
should be available at:

```
http://<hostname>/tor/server/fp/<F1>+<F2>+<F3
```

(NOTE: Due to squid proxy url limitations at most 96
single request.

Implementations SHOULD NOT download descripto
allows a corrupted server (in collusion with a cache)
client, and thereby partition that client from the rest

The server descriptor with (descriptor) digest `<D>` (i

```
http://<hostname>/tor/server/d/<D>.z
```

The most recent descriptors with digests `<D1>`, `<D2`

```
http://<hostname>/tor/server/d/<D1>+<D2>+<D3>
```

The most recent descriptor for this server should be

```
http://<hostname>/tor/server/authority.z
```

This is used for authorities, and also if a server is co
implementations (starting at 0.1.1.x) use this resour
DirPort is reachable. It is also useful for debugging p

A concatenated set of the most recent descriptors fo
available at:

```
http://<hostname>/tor/server/all.z
```

Extra-info documents are available at the URLS:

```
http://<hostname>/tor/extra/d/...
http://<hostname>/tor/extra/fp/...
http://<hostname>/tor/extra/all[.z]
http://<hostname>/tor/extra/authority[.
```

(These work like the `/tor/server/` URLs: they supp
their digest, by the fingerprint of their servers, or all
we serve the extra-info that corresponds to the desc
fingerprint. Only directory authorities of version 0.2
support the first three classes of URLs. Caches may
them if they have advertised "caches-extra-info".)

For debugging, directories SHOULD expose non-con
above, but without the final ".z". If the client uses Ac
override the presence or absence of the ".z" (see sec

Clients SHOULD use upper case letters (A-F) when b
MUST accept both upper and lower case fingerprint

Converting a curve2551 ed25519 public key

Given an X25519 key, that is, an affine point (u,v) on

bv^2 = u(u^2 + au +1)

where

```
a = 486662
b = 1
```

and comprised of the compressed form (i.e. consisti retrieve the y-coordinate of the affine point (x,y) on defined by

-x^2 + y^2 = 1 + d x^2 y^2

where

d = - 121665/121666

by computing

y = (u-1)/(u+1).

and then we can apply the usual curve25519 twistec algorithm to find *an* x-coordinate of an affine twiste with. Signing keys for ed25519 are compressed curv a y-coordinate and the sign of the x-coordinate), anc points in Montgomery form (i.e. a u-coordinate).

However, note that compressed point in Montgome sign of the corresponding twisted Edwards x-coordi of the x-coordinate to do this operation; otherwise, that might have correspond to the ed25519 public k

To get the sign, the easiest way is to take the corres ed25519 public key generation algorithm, and see w

[Recomputing the sign bit from the private key ever inefficient to me… —isis]

Note that in addition to its coordinates, an expande byte random value, "prefix", used to compute interr security, this prefix value should be derived determi

The Tor implementation derives it as `SHA512(priva`
the nul-terminated string:

"Derive high part of ed25519 key from curve25519 k

On the client side, where there is no access to the cu
the curve25519 public key's Montgomery u-coordin
v-coordinate by computing the right-hand side of th

bv^2 = u(u^2 + au +1)

where

```
a = 486662
b = 1
```

Then, knowing the intended sign of the Edwards x-c
x-coordinate by computing:

x = (u/v) * sqrt(-a - 2)

Inferring missing proto l

The directory authorities no longer allow versions o
there is no version of Tor in the consensus before 0
versions of Tor earlier than 0.2.4.19, so that we can
Tor versions include:

Cons=1-2 Desc=1-2 DirCache=1 HSDir=1 HSIntro=3 I
Microdesc=1-2 Relay=1-2

For Desc, Microdesc and Cons, Tor versions before (
support version 1.

Limited ed diff format

We support the following format for consensus diffs
but clients MUST NOT accept other ed commands.

We support the following ed commands, each on a l

```
- "<n1>d"           Delete line n1
- "<n1>,<n2>d"      Delete lines n1 throug
- "<n1>,$d"         Delete line n1 through
- "<n1>c"           Replace line n1 with t
- "<n1>,<n2>c"      Replace lines n1 throu
                    following block.
- "<n1>a"           Append the following b
```

Note that line numbers always apply to the file after
been applied. Note also that line numbers are 1-ind

The commands MUST apply to the file from back to
referred to by their position in the original file.

If there are any directory signatures on the original
be a " `<n1>,$d` " form to remove all of the directory s
that the client will successfully apply the diff even if
the signatures.

The replace and append command take blocks. The
diff after the line with the command. A line with just
not part of the lines to add). Note that it is impossib

Tor Shared Random Subs Specification

This document specifies how the commit-and-revea works. This text used to be proposal 250-commit-re

Introduction

Motivation

For the next generation hidden services project, we fresh random value every day in such a way that it c influenced by an attacker.

Currently we need this random value to make the H which should resolve a wide class of hidden service harder for people to gauge the popularity and activi Furthermore this random value can be used by othe randomness like Tor-related protocols (e.g. OnioNS) canaries).

Previous work

Proposal 225 specifies a commit-and-reveal protoco and have the results be fed to the directory authorit operators feel unsafe running a third-party script th connections from the Internet. Hence, this proposal reveal idea in the Tor voting process which should n maintain.

Overview

This proposal alters the Tor consensus protocol suc
every midnight by the directory authorities during tl
distributed random generator scheme is based on t

The proposal also specifies how the final shared rar
documents so that clients who need it can get it.

Introduction to our commit-and-

Every day, before voting for the consensus at 00:00l
random value and keeps it for the whole day. The au
random value and calls the output its "commitment
called the "reveal" value.

The idea is that given a reveal value you can cryptog
to a given commitment value (by hashing it). Howev
should not be able to derive the underlying reveal v
is specified in section [COMMITREVEAL].

Ten thousand feet view of the pı

Our commit-and-reveal protocol aims to produce a
shared_random_value here and elsewhere) every da
random value is embedded in the consensus docum

Our protocol has two phases and uses the hourly vc
lasts 12 hours, which means that 12 voting rounds h
protocol works as follows:

Commit phase:

```
          Starting at 00:00UTC and for a perioc
          hour include their commitment in the
          received commitments from other autho

        Reveal phase:

          At 12:00UTC, the reveal phase starts
          protocol at 00:00UTC. In this stage,
value
          they committed to in the previous pha
          values from other authorities, when a
          vote.

        Shared Randomness Calculation:

          At 00:00UTC, the shared random value
          revealed values and added to the cons

     This concludes the commit-and-reveal proto
```

How we use the consensus

The produced shared random values need to be rea
reason we include them in the consensus documen

Every hour the consensus documents need to incluc
day, as well as the shared random value of the prev
these values might be needed at a given time for a T
according to section [TIME-OVERLAP] of proposal 22
values need to be included in votes as well.

Hence, consensuses need to include:

```
        (a) The shared random value of the curr
        (b) The shared random value of the prev
```

For this, a new SR consensus method will be neede
this new protocol.

Inserting Shared Random Values in the co

After voting happens, we need to be careful on how
(SRV) to put in the consensus, to avoid breaking the
having different views of the commit-and-reveal pro
some rounds of the protocol).

For this reason, authorities look at the received vote
employ the following logic:

```
    - First of all, they make sure that the ag
      above the SR consensus method.

    - Authorities include an SRV in the conse
      been voted by at least the majority of a

    - For the consensus at 00:00UTC, authorit
consensus
      if and only if the SRV has been voted by
      authorities (where AuthDirNumAgreements
      parameter).
```

Authorities include in the consensus the most popu
constraints. Otherwise, no SRV should be included.

The above logic is used to make it harder to break tl
causes.

We use the AuthDirNumAgreements consensus par
of dirauths supports the SR protocol during SRV cre
dirauths drop offline in the middle of the run the SR
go to extra lengths to ensure this because changing
terrible reachability consequences for hidden servic

Persistent State of the Protocol

A directory authority needs to keep a persistent stat
run. This allows an authority to join the protocol sea

During the commitment phase, it is populated with
Then during the reveal phase, the reveal values are

As discussed previously, the shared random values
period must also be present in the state at all times

Protocol Illustration

An illustration for better understanding the protoco

https://people.torproject.org/~asn/hs_notes/shared

It reads left-to-right.

The illustration displays what the authorities (A_1, A
-> c_1 -> r_1' denotes that authority A_1 committed
the reveal value r_1.

The illustration depicts only a few rounds of the who
three rounds of the commit phase, then it jumps to
continues with the first two rounds of the reveal pha
round of the protocol run. It finally shows the first r
protocol run (00:00UTC) where the final Shared Ran
fictional example, the SRV was computed with 3 aut
"a56fg39h".

We advice you to revisit this after you have read the

Protocol

In this section we give a detailed specification of the
participants' logic and the messages they send. The
in the next section ([SPEC]).

Now we go through the phases of the protocol:

Commitment Phase

The commit phase lasts from 00:00UTC to 12:00UTC

During this phase, an authority commits a value in it
state as well.

Authorities also save any received authoritative com
permanent state. We call a commit by Alice "authori
vote.

Voting During Commitment Phase

During the commit phase, each authority includes in

```
     - The commitment value for this protocol
     - Any authoritative commitments received
     - The two previous shared random values p
 any).
```

The commit phase lasts for 12 hours, so authorities
their values. An authority MUST NOT commit a seco
of the commit phase.

If an authority publishes a second commitment valu
first commitment should be taken in account by oth
commitments MUST be ignored.

Persistent State During Commitment Pha

During the commitment phase, authorities save in t
commits they have received from each authority. O
considered trusted and active at a given time.

Reveal Phase

The reveal phase lasts from 12:00UTC to 00:00UTC.

Now that the commitments have been agreed on, it
random values.

Voting During Reveal Phase

During the reveal phase, each authority includes in i

```
    - Its reveal value that was previously co
    - All the commitments and reveals receive
    - The two previous shared random values p
any).
```

The set of commitments have been decided during
remain the same. If an authority tries to change its c
or introduce a new commitment, the new commitm

Persistent State During Reveal Phase

During the reveal phase, authorities keep the autho
phase in their persistent state. They also save any re
authoritative commits and are valid (as specified in [

An authority that just received a reveal value from a
the next voting round before including that reveal va

Shared Random Value Calculatic

Finally, at 00:00UTC every day, authorities compute
value must be added to the consensus so clients car

Authorities calculate the shared random value using
specified in subsection [SRCALC].

Authorities at 00:00UTC start including this new sha
replacing the one from two protocol runs ago. Autho
shared random value in the consensus as well.

Apart from that, authorities at 00:00UTC proceed vo first round of the commitment phase (section [COM

Shared Randomness Calculation

An authority that wants to derive the shared randor
appropriate reveal values for that time period and c

HASHED_REVEALS = H(ID_a | R_a | ID_b | R_b | ..)

```
SRV = SHA3-256("shared-random" | INT_8(
                 HASHED_REVEALS | PREVIOU
```

where the ID_a value is the identity key fingerprint c
corresponding reveal value of that authority for the

Also, REVEAL_NUM is the number of revealed values
protocol version number and PREVIOUS_SRV is the p
previous shared random value is known, then PREV
bytes.

To maintain consistent ordering in HASHED_REVEAL
based on the R_a value in ascending order.

Bootstrapping Procedure

As described in [CONS], two shared random values a
periods to work properly as specified in proposal 22
randomness of this system till it has bootstrapped c
random values are included in a consensus. This sh
consensuses have been produced, which takes 48 h

Rebooting Directory Authorities

The shared randomness protocol must be able to su
or join in the middle of the protocol execution.

An authority that commits in the Commitment Phas
its reveal value on disk so that it continues participa
or during the Reveal Phase. The reveal value MUST l
sending it on wrong protocol runs.

An authority that misses the Commitment Phase ca
to participate in the protocol for that run. Same goe
Reveal phase. Authorities who do not participate in
commits and reveals of others in their vote.

Finally, authorities MUST implement their persisten
never commit two different values in the same prot
in the middle (assuming that their persistent state fi
structure the persistent state is found at [STATEFOR

Specification

Voting

This section describes how commitments, reveals ar
describe how to encode both the authority's own co
commitments/reveals received from the other auth
share the same line, but reveals are optional.

Participating authorities need to include the line:

"shared-rand-participate"

in their votes to announce that they take part in the

Computing commitments and reveals

A directory authority that wants to participate in thi
of commitment/reveal values for every protocol run
fresh pair of such values right before the first comm
00:00UTC).

The value REVEAL is computed as follows:

REVEAL = base64-encode(TIMESTAMP || H(RN))

```
      where RN is the SHA3 hashed value of a
 the
      random value to avoid exposing raw byte
 (see
      [RANDOM-REFS]).

      TIMESTAMP is an 8-bytes network-endian
      set TIMESTAMP to the valid-after time c
 plan
      to publish their commit into (so usuall
 start
      up in a later commit round).

   The value COMMIT is computed as follows:

      COMMIT = base64-encode( TIMESTAMP || H(
```

Validating commitments and reveals

Given a COMMIT message and a REVEAL message it
indeed correspond. To do so, the client extracts the
message, hashes it, and compares it with the H(H(RI
say that the COMMIT and REVEAL messages corresp
successful.

Participants MUST also check that corresponding CC
same timestamp value.

Authorities should ignore reveal values during the R
commit values published during the Commitment P

Encoding commit/reveal values in votes

An authority puts in its vote the commitments and r
the other authorities. To do so, it includes the follow

"shared-rand-commit" SP VERSION SP ALGNAME SP

where VERSION is the version of the protocol the co
the authority's SHA1 identity fingerprint and COMM
[COMMITREVEAL]. Authorities during the reveal pha
encoded reveal value REVEAL. There MUST be only c
considered invalid. Finally, the ALGNAME is the hash
compute COMMIT and REVEAL which is "sha3-256" f

Shared Random Value

Authorities include a shared random value (SRV) in t
encoding for the previous and current value respect

```
"shared-rand-previous-value" SP NUM_REVE
"shared-rand-current-value" SP NUM_REVEA
```

where VALUE is the actual shared random value enc
in section [SRCALC]. NUM_REVEALS is the number o
SRV.

To maintain consistent ordering, the shared random
be listed before the values of the current period.

Encoding Shared Random Values

Authorities insert the two active shared random valı
same encoding format as in [SRVOTE].

Persistent state format

As a way to keep ground truth state in this protocol,
state of the protocol. The next sub-section suggest ε
same as the current state file format.

It contains a preamble, a commitment and reveal se
values.

The preamble (or header) contains the following iteı
given here:

"Version" SP version NL

[At start, exactly once.]

A document format version. For this specification, v

"ValidUntil" SP YYYY-MM-DD SP HH:MM:SS NL

[Exactly once]

```
        After this time, this state is expire
        trusted. The validity time period is
        protocol run (the upcoming noon).
```

The following details the commitment and reveal se
in the vote. This makes it easier for implementation

"Commit" SP version SP algname SP identity SP com

[Exactly once per authority]

The values are the same as detailed in section [COM

This line is also used by an authority to store its owı

Finally is the shared random value section.

"SharedRandPreviousValue" SP num_reveals SP valu

[At most once]

```
    This is the previous shared random va
    period. The fields are the same as ir

"SharedRandCurrentValue" SP num_reveals

    [At most once]

    This is the latest shared random valu
    section [SRVOTE].
```

Security Analysis

Security of commit-and-reveal a

The security of commit-and-reveal protocols is well
Basically, the protocol is insecure to the extent that
authorities gets to choose among 2^b outcomes for
an attacker who is not a dirauth should not be able

We believe that this system offers sufficient security
situation. More secure solutions require much more
protocols so this seems like an acceptable solution f

Here are some examples of possible future directio

- Schemes based on threshold signatures (e.g. s
- Unicorn scheme by Lenstra et al. [UNICORN]
- Schemes based on Verifiable Delay Functions [

For more alternative approaches on collaborative ra
the discussion at [RNGMESSAGING].

Predicting the shared random va phase

The reveal phase lasts 12 hours, and most authoriti
first round of the reveal phase. This means that an a
random value about 12 hours before it's generated.

This does not pose a problem for the HSDir hash rin
restriction on HSDir nodes, so 12 hours predictabilit

Any other protocols using the shared random value
this property.

Partition attacks

This design is not immune to certain partition attack
gain to an attacker as they are very easy to detect a

would need to compromise a directory authority at
byzantine general problem, it's very hard (even imp
against all such attacks. Nevertheless, this section d
and how to detect them.

Partition attacks during commit phase

A malicious directory authority could send only its c
results in that authority having an extra commit valu
that the others don't have. Since the consensus nee
SRV value. However, the attacker, using this attack, c
authority from the consensus decision at 24:00 whe

An attacker could also partition the authorities by se
values to different authorities during the commit ph

All of the above is fairly easy to detect. Commitment
authority should NEVER be different between autho
ongoing or very bad bug (highly unlikely).

Partition attacks during reveal phase

Let's consider Alice, a malicious directory authority.
round, and reveal its value to half of the authorities.
into two sets: the ones who think that the shared ra
reveal, and the rest who don't know about it. This w
shared random value.

A similar attack is possible. For example, two rounds
Alice could advertise her reveal value to only half of
reveal phase round, half of the dirauths will include
the others will not. In the end of the reveal phase, h
different shared randomness value than the others.

We claim that this attack is not particularly fruitful: A
random values to choose from which is a fundamen
protocols as well (since the last person can always a
sabotage the consensus, but there are other ways th
voting system.

Furthermore, we claim that such an attack is very no
requires the authority to sabotage two consensuses
Furthermore, the authority needs to send different v

detectable. Like the commit phase attack, the detec
commitment values in a vote coming from an autho
authority.

Discussion

Why the added complexity from

The complexity difference between this proposal an
doesn't specify how the shared random value gets t
lots of effort specifying how the two shared random
accessible to clients.

Why do you do a commit-and-re
rounds?

The reader might be wondering why we span the pr
(24 hours), when only 3 rounds would be sufficient t

We decided to do it this way, because we piggyback
happens every hour.

We could instead only do the shared randomness p
Or to do it multiple times a day.

However, we decided that since the shared random
anyway, carrying the commitments/reveals as well v
way we give more chances for a failing dirauth to re

Why can't we recover if the 00:0

If the 00:00UTC consensus fails, there will be no sha
In theory, we could recover by calculating the share
instead. However, the engineering issues with addir
For example, it's not easy for an authority who just k
consensus failed to be created.

Acknowledgements

Thanks to everyone who has contributed to this des

Thanks go to arma, ioerror, kernelcorn, nickm, s7r, S
else!

References:


```
[RANDOM-REFS]:
   http://projectbullrun.org/dual-ec/ext-ranc
   https://lists.torproject.org/pipermail/tor

[RNGMESSAGING]:
   https://moderncrypto.org/mail-archive/mess

[HOPPER]:
   https://lists.torproject.org/pipermail/tor

[UNICORN]:
   https://eprint.iacr.org/2015/366.pdf

[VDFS]:
   https://eprint.iacr.org/2018/601.pdf
```

Tor Path Specification

Roger Dingledir
Nick Mathewsor

Note: This is an attempt to specify Tor as currently ir
will implement improved algorithms.

This document tries to cover how Tor chooses to bu
circuits. Other implementations MAY take other app
aware of the anonymity and load-balancing implicat

THIS SPEC ISN'T DONE YET.

The key words "MUST", "MUST NOT", "REQUIRED", "
"SHOULD NOT", "RECOMMENDED", "MAY", and "OP
interpreted as described in RFC 2119.

General operation

Tor begins building circuits as soon as it has enough
circuits are built preemptively because we expect to
some are built because of immediate need (for user
handle, for testing the network or our reachability, a

```
[Newer versions of Tor (0.2.6.2-alpha and l
 If the consensus contains Exits (the typic
 exit and internal circuits. When bootstrap
 to handle an application requesting an ex
 World Wide Web.
```

If the consensus does not contain Exits, Tor will only
earlier statuses will have included "internal" as indic
completes, Tor will be ready to handle an applicatio
hidden services at ".onion" addresses.

If a future consensus contains Exits, exit circuits may

When a client application creates a new stream (by
launching a resolve request), we attach it to an appr
wait if an appropriate circuit is in-progress. We laun
circuit can handle the request. We rotate circuits ov
attacks.

To build a circuit, we choose all the nodes we want t
Sometimes, when we want a circuit that ends at a gi
unused circuit, we "cannibalize" the existing circuit a

These processes are described in more detail below

This document describes Tor's automatic path selec
overridden by a controller (with the EXTENDCIRCUIT
Paths constructed through these means may violate

Terminology

A "path" is an ordered sequence of nodes, not yet b

A "clean" circuit is one that has not yet been used fo

A "fast" or "stable" or "valid" node is one that has th
respectively, based on our current directory informa

consisting only of "fast" or "stable" nodes.

In an "exit" circuit, the final node is chosen based or in any case it avoids nodes with exit policy of "reject hand, is one where the final node is chosen just like policy).

A "request" is a client-side stream or DNS resolve th

A "pending" circuit is one that we have started to bu

A circuit or path "supports" a request if it is okay to request, according to the rules given below. A circui some aspect of the request is unknown (usually its t probably supports the request according to the rule

A relay's bandwidth

Old versions of Tor did not report bandwidths in ne had to learn them from the routers' advertised relay

For versions of Tor prior to 0.2.1.17-rc, everywhere "bandwidth", we mean its clipped advertised bandw of the 'rate' and 'observed' arguments to the "band descriptor. If a router's advertised bandwidth is gre MAX_BELIEVABLE_BANDWIDTH (currently 10 MB/s),

For more recent versions of Tor, we take the bandw and fall back to the clipped advertised bandwidth or bandwidths listed.

Building circuits

Here we describe a number of rules for building circ
so, how we choose the paths for them, when we giv
what we do when circuit construction fails.

When we build

We don't build circuits until we h info

There's a class of possible attacks where our directo about the relays that they would like us to use. To p multi-hop circuits (including preemptive circuits, [or onion-service circuits] or self-testing testing circuits) directory information to be reasonably confident th

Here, "enough" directory information is defined as:

```
* Having a consensus that's been valid
  last REASONABLY_LIVE_TIME interval (2

* Having enough descriptors that we cou
  fraction F of all bandwidth-weighted
  ExitNodes/EntryNodes/etc into account

  (F is set by the PathsNeededToBuildC
  defaulting to the 'min_paths_for_circ
  parameter, with a final default value

* Having enough descriptors that we cou
  fraction F of all bandwidth-weighted
  ExitNodes/EntryNodes/etc into account

  (F is as above.)

* Having a descriptor for every one of
  NUM_USABLE_PRIMARY_GUARDS guards amon
  guard-spec.txt)
```

We define the "fraction of bandwidth-weighted path fractions.

```
* The fraction of descriptors that we h
  flag, weighted by their bandwidth for
* The fraction of descriptors that we h
  weighted by their bandwidth for the m
* The fraction of descriptors that we h
  flag, weighted by their bandwidth for
```

If the consensus has zero weighted bandwidth for a Exit), Tor instead uses the fraction of relays for whic by bandwidth at all).

If the consensus lists zero exit-flagged relays, Tor ins
relays.

Clients build circuits preemptive

When running as a client, Tor tries to maintain at lea
so that new streams can be handled quickly. To incr
tries to predict what circuits will be useful by choosi
the ports we have used in the recent past (by defaul
Tor tries to maintain one clean fast exit circuit that a
least two fast clean stable internal circuits in case w
service request (at least three if we *run* a hidden ser

After that, Tor will adapt the circuits that it preempti
sees from the user: it tries to have two fast clean ex
within the past hour (each circuit can be adequate f
need two separate circuits for each port), and it tries
available if we've seen resolves or hidden service ac
12 or more clean circuits open, it doesn't open more

Only stable circuits can "cover" a port that is listed in
Similarly, hidden service requests to ports listed in L
internal circuits.

Note that if there are no requests from the user for
build no preemptive circuits.

The Tor client SHOULD NOT store its list of predicted

Clients build circuits on demand

Additionally, when a client request exists that no cir
we create a new circuit to support the request. For e
that will handle the most pending requests (choosin
circuit to end there, and repeat until every unattach
pending or built circuit. For internal circuits, we pick
repeating as needed.

Clients consider a circuit to become "dirty" as soon a
other request is performed over the circuit. If a circu

MaxCircuitDirtiness seconds, new circuits may not b

In some cases we can reuse an already established
circuits"

for details.

Relays build circuits for testing r bandwidth

Tor relays test reachability of their ORPort once the
startup and whenever their IP address changes). Th
circuit with themselves as the last hop. As soon as a
relay decides it's reachable and is willing to publish

We launch multiple testing circuits (one at a time), u
NUM_PARALLEL_TESTING_CIRC (4) such circuits ope
sending a certain number of relay drop cells down e
CELL_NETWORK_SIZE total cells divided across the f
CIRCWINDOW_START (1000) cells total. This exercise
bandwidth, and helps to jumpstart the observed ba

Tor relays also test reachability of their DirPort once
they use an ordinary exit circuit for this purpose.

Hidden-service circuits

See section 4 below.

Rate limiting of failed circuits

If we fail to build a circuit N times in a X second peri
works), we stop building circuits until the X seconds

When to tear down circuits

Clients should tear down circuits (in general) only w

them. Additionally, clients should tear-down stream
following conditions:

```
- The circuit has never had a stream att
  long in the past (based on CircuitsAva
  cbtlearntimeout, depending on timeout

- The circuit is dirty (has had a stream
  dirty for at least MaxCircuitDirtiness
```

Path selection and const

We choose the path for each new circuit before we l directory information. (Clients and relays use the lat directory authorities use their own opinions.)

We choose the exit node first, followed by the other other words, for a 3-hop circuit, we first pick hop 3,

Universal constraints

All paths we generate obey the following constraint

- We do not choose the same router twice for th
- We do not choose any router in the same fami routers are in the same family if each one lists descriptor.)
- We do not choose more than one router in a g /16 for IPv4 and /32 for IPv6. (C Tor overrides t Arti overrides this with `ipv[46]_subnet_famil`
- The first node must be a Guard (see discussior specification).
- XXXX Choosing the length

Special-purpose constraints

Additionally, we may be building circuits with one or request puts certain constraints on paths.

Most circuits need to be "Fast". For these, we only cl non-"fast" circuits, nodes without the `Fast` flag are

- TODO document which circuits (do not) need t

Similarly, some circuits need to be "Stable". For thes Stable flag.

- All service-side introduction circuits and all ren
- All connection requests for connections that w time require Stable circuits. Currently, Tor deci target port, and comparing it to a list of "long-l

1863, 5050, 5190, 5222, 5223, 6667, 6697, 830(

Weighting node selection

For all circuits, we weight node selection according t

We also weight the bandwidth of Exit and Guard fla
of total bandwidth that they make up and dependin
selected for.

These weights are published in the consensus, and
"Computing Bandwidth Weights" in the directory sp

```
Wgg - Weight for Guard-flagged nodes ir
Wgm - Weight for non-flagged nodes in t
Wgd - Weight for Guard+Exit-flagged noc

Wmg - Weight for Guard-flagged nodes ir
Wmm - Weight for non-flagged nodes in t
Wme - Weight for Exit-flagged nodes in
Wmd - Weight for Guard+Exit flagged noc

Weg - Weight for Guard flagged nodes ir
Wem - Weight for non-flagged nodes in t
Wee - Weight for Exit-flagged nodes in
Wed - Weight for Guard+Exit-flagged noc

Wgb - Weight for BEGIN_DIR-supporting (
Wmb - Weight for BEGIN_DIR-supporting r
Web - Weight for BEGIN_DIR-supporting E
Wdb - Weight for BEGIN_DIR-supporting (

Wbg - Weight for Guard+Exit-flagged noc
Wbm - Weight for Guard+Exit-flagged noc
Wbe - Weight for Guard+Exit-flagged noc
Wbd - Weight for Guard+Exit-flagged noc
```

If any of those weights is malformed or not present
the regular path selection algorithm setting the wei

Choosing an exit

If we know what IP address we want to connect to, v
router will support it by simulating its declared exit

(DNS resolve requests are only sent to relays whose

:".)

Because we often connect to addresses of the form know the target IP address when we select an exit n an exit node that "might support" connections to a g address. An exit node "might support" such a conne connections to that port precedes all clauses (if any) port.

Unless requested to do so by the user, we never cho by more than half of the authorities who advertise t

User configuration

Users can alter the default behavior for path selecti

```
    - If "ExitNodes" is provided, then every r
      the ExitNodes list.  (If a request is su
 list,
      and StrictExitNodes is false, then Tor t
      ExitNodes were not provided.)

    - "EntryNodes" and "StrictEntryNodes" beha

    - If a user tries to connect to or resolve
      <target>.<servername>.exit, the request
      <target>, and the request is only suppor
      or fingerprint is <servername>.

    - When set, "HSLayer2Nodes" and "HSLayer3N
      restrictions to allow nodes in the same
      in the path. They also allow the guard r
      and HSDIR, and as the hop before those p
```

Cannibalizing circuits

If we need a circuit and have a clean one already est
the clean circuit for our new purpose. Specifically,

For hidden service interactions, we can "cannibalize
available, so we don't need to build those circuits fr

We can also cannibalize clean circuits when the clier
via the ".exit" notation or because the destination is
exit node.

Learning when to give u circuit construction

Since version 0.2.2.8-alpha, Tor clients attempt to le on network conditions.

Distribution choice

Based on studies of build times, we found that the c appears to be a Frechet distribution (and a multi-mo one guard or bridge is used). However, estimators a distribution are difficult to work with and slow to co interested in the accuracy of the tail, clients approxi distribution with a single Pareto curve.

How much data to record

From our observations, the minimum number of cir appears to be on the order of 100. However, to keep store 1000 most recent circuit build times in a circul

These build times only include the times required to times required to build the first three hops of circuit of fewer than three hops are not recorded, and hop

The Tor client should build test circuits at a rate of o until 'cbtmincircs' (100 circuits) are built, with a max 10) circuits open at once. This allows a fresh Tor to h within 30 minutes after install or network change (se Conditions below.)

Timeouts are stored on disk in a histogram of 10ms calculate the Xm value above. The timeouts recorde after being read from disk, to preserve a proper exp

Thus, some build time resolution is lost during resta different persistence mechanism than this histogram binning is still needed for parameter estimation.

Parameter estimation

Once 'cbtmincircs' build times are recorded, Tor clie
parameters and recompute the timeout every circui
when to pause and reset timeout due to too many c

Tor clients calculate the parameters for a Pareto dis
maximum likelihood estimator. For derivation, see:
/wiki/Pareto_distribution#Estimation_of_parameters

Because build times are not a true Pareto distributio
max likelihood estimator, the mode of the distributi

Instead of using the mode of discrete build times di
parameter using the weighted average of the midpo
frequently occurring 10ms histogram bins. Ties are
in favor of bins corresponding to shorter build times

(The use of 10 modes was found to minimize error f
10ms bins for quantiles 60-80, compared to many o

To avoid ln(1.0+epsilon) precision issues, use log law
as the sum of logs followed by subtraction, rather th

```
alpha = n/(Sum_n{ln(MAX(Xm, x_i))} - n\*ln(Xm
```

In this, n is the total number of build times that have
build time, and Xm is the modes of x_i as above.

All times below Xm are counted as having the Xm va
estimators, Xm is supposed to be the lowest value. I
averaging to estimate Xm, there can be values below
estimator then treats that everything smaller than X
that if clients did not do this, alpha could underflow
an exponential curve, not a Pareto probability distril

The timeout itself is calculated by using the Pareto Q
give us the value on the CDF such that 80% of the m
timeout value (parameter 'cbtquantile').

The Pareto Quantile Function (inverse CDF) is:

```
F(q) = Xm/((1.0-q)^(1.0/alpha))
```

Thus, clients obtain the circuit build timeout for 3-ho

```
timeout_ms = F(0.8)    # 'cbtquantile' == 0.8
```

With this, we expect that the Tor client will accept th
paths on the network.

Clients obtain the circuit close time to completely ab

```
close_ms = F(0.99)     # 'cbtclosequantile' =
```

To avoid waiting an unreasonably long period of tim
that are down, Tor clients cap timeout_ms at the ma
and cap close_ms at twice this max, but at least 60 s

```
      timeout_ms = MIN(timeout_ms, max_observe
      close_ms = MAX(MIN(close_ms, 2*max_obser
 'cbtinitialtimeout')
```

Calculating timeouts thresholds lengths

The timeout_ms and close_ms estimates above are
only 3-hop circuits are recorded in the list of build ti

To calculate the appropriate timeouts and close tim
client multiples the timeout_ms and close_ms value
the number of communication hops needed to buil

```
 timeout_ms\[hops=n\] = timeout_ms * Actions(N

 close_ms\[hops=n\] = close_ms * Actions(N) /
```

where `Actions(N) = N * (N + 1) / 2.`

To calculate timeouts for operations other than circ
Actions(N) for every round-trip communication requ

How to record timeouts

Pareto estimators begin to lose their accuracy if the
actually calculate two timeouts: a usage timeout, an

Circuits that pass the usage timeout are marked as
allowed to continue to build until the close timeout
'cbtclosequantile' (default 99) on the Pareto curve, o

The actual completion times for these measuremen

Implementations should completely abandon a circ
build time exceeds the close threshold. Such closed
typically means one of the relays in the path is offlir

Detecting Changing Network Co

Tor clients attempt to detect both network connecti
timeout characteristics.

To detect changing network conditions, clients keep
timeout status of the past 'cbtrecentcount' circuits (
completed at least one hop. If more than 90% of the
all buildtimes history, resets the timeout to 'cbtinitia
begins recomputing the timeout.

If the timeout was already at least `cbtinitialtimeo`

The records here (of how many circuits succeeded c
'cbrrecentcount') are not stored as persistent state.
state.

Consensus parameters governin

Clients that implement circuit build timeout learning
parameters that govern behavior, in order to allow u
behaviors due to client circuit construction. If these
consensus, the listed default values should be used

```
cbtdisabled
  Default: 0
  Min: 0
  Max: 1
  Effect: If 1, all CircuitBuildTime le
          disabled and history should b
          emergency situations only.

cbtnummodes
  Default: 10
  Min: 1
  Max: 20
  Effect: This value governs how many m
  average calculation of Pareto paramet
  average of multiple modes improves ac
  for quantile cutoffs from 60-80% (see

cbtrecentcount
  Default: 20
  Min: 3
  Max: 1000
  Effect: This is the number of circuit
          timeout) to keep track of for

cbtmaxtimeouts
  Default: 18
  Min: 3
  Max: 10000
  Effect: When this many timeouts happe
          circuit attempts, the client
          history and begin learning a

          Note that if this parameter's
          of 'cbtrecentcount', then the
          discarded because of this fea

cbtmincircs
  Default: 100
  Min: 1
  Max: 10000
  Effect: This is the minimum number of
          computing a timeout.

          Note that if this parameter's
          number of time observations t
          circular buffer), circuit bu
          effectively disabled, and the
          indefinitely.

cbtquantile
  Default: 80
  Min: 10
  Max: 99
  Effect: This is the position on the q
          timeout value. It is a percen
```

```
        cbtclosequantile
          Default: 99
          Min: Value of cbtquantile parameter
          Max: 99
          Effect: This is the position on the c
                  timeout value to use to actua
                  percent (0-99).

        cbttestfreq
          Default: 10
          Min: 1
          Max: 2147483647 (INT32_MAX)
          Effect: Describes how often in second
                  gather timeout values. Only a
'cbtmincircs'
                  have been recorded.

        cbtmintimeout
          Default: 10
          Min: 10
          Max: 2147483647 (INT32_MAX)
          Effect: This is the minimum allowed t

        cbtinitialtimeout
          Default: 60000
          Min: Value of cbtmintimeout
          Max: 2147483647 (INT32_MAX)
          Effect: This is the timeout value to
                  to compute a timeout, in mill
                  enough data to compute a time
                  then we use this interval bot
                  abandon timeout.

        cbtlearntimeout
          Default: 180
          Min: 10
          Max: 60000
          Effect: This is how long idle circuit
                  learning a new timeout value.

        cbtmaxopencircs
          Default: 10
          Min: 0
          Max: 14
          Effect: This is the maximum number of
                  at the same time during the c
phase.
```

Handling failure

If an attempt to extend a circuit fails (either because
subsequent extend failed) then the circuit is torn do
really?) Requests that might have been supported b
unsupported, and a new circuit needs to be constru

If a stream "begin" attempt fails with an EXITPOLICY
exit policy is not correctly advertised, so we treat th
until we retrieve a fresh descriptor for it.

Excessive amounts of either type of failure can indic
discussion of path bias detection for how excessive

Attaching streams to cir

When a circuit that might support a request is built,
stream to the circuit and sends a BEGIN, BEGIN_DIR
If the request completes unsuccessfully, Tor conside
relay cell. [XXX yes, and?]

After a request has remained unattached for SocksT
abandons the attempt and signals an error to the cl
the SOCKS connection).

XXX Timeouts and when Tor auto-retries.

- What stream-end-reasons are appropriate for

If no reply to BEGIN/RESOLVE, then the stream will t

Hidden-service related c

XXX Tracking expected hidden service use (client-sid

Guard nodes

We use Guard nodes (also called "helper nodes" in t
certain profiling attacks. For an overview of our Gua
grown rather complex -- see guard-spec.txt.

How consensus bandwidth weig guard selection

When weighting a list of routers for choosing an ent
parameters (from the "bandwidth-weights" line) app

```
Wgg - Weight for Guard-flagged nodes ir
Wgm - Weight for non-flagged nodes in t
Wgd - Weight for Guard+Exit-flagged noc
Wgb - Weight for BEGIN_DIR-supporting (
Wmb - Weight for BEGIN_DIR-supporting r
Web - Weight for BEGIN_DIR-supporting E
Wdb - Weight for BEGIN_DIR-supporting (
```

Please see "bandwidth-weights" in §3.4.1 of dir-spec
these parameters.

If a router has been marked as both an entry guard
more, with our preference for doing so (roughly) lin
non-guard bandwidth and bandwidth weight (calcul
into account). From proposal 236:

> Let Wpf denote the weight from the 'band
> client would apply to N for position p if it l
> flag, Wpn the weight if it did not have the
> measured bandwidth of N in the consensu
> N for position p proportionally to Wpf*B or*
> choose N proportionally to F*Wpf*B + (1-F)И

where F is the weight as calculated using the above

Server descriptor purpo

There are currently three "purposes" supported for
controller, and bridge. Most descriptors are of type
the consensus, and the ones fetched and used in nc

Controller-purpose descriptors are those delivered
they will be kept around (and expire like normal des
controller in its CIRCUITEXTEND commands. Otherw
chooses paths.

Bridge-purpose descriptors are for routers that are
paper/blocking.pdf for more design explanation, or
Currently bridge descriptors are used in place of no
have UseBridges enabled.

Detecting route manipu nodes (Path Bias)

The Path Bias defense is designed to defend against malicious Guard nodes deliberately fail or choke cir nodes to maximize their network utilization in favor

In the extreme, the attack allows an adversary that deanonymize c/n of the network connections, break original threat model. It also allows targeted attacks specific users, bridges, or Guard nodes.

There are two points where path selection can be m during usage. Circuit construction can be manipulat circuit extend steps, which causes the Tor client to t construction with a new path. Circuit usage can be r retry features of Tor (for example by withholding str client until the stream timeout has expired), at whicl transparently retry the stream on a new path.

The defense as deployed therefore makes two inde successful path use: one during circuit construction,

The intended behavior is for clients to ultimately dis for excessive circuit failure of either type (for the pa "Parameterization" below); however known issues w the defense to being informational only at this stage enforcement").

Measuring path construction su

Clients maintain two counts for each of their guards circuit was extended to at least two hops through th of circuits that successfully complete through that g is used to determine a circuit success rate for that G

Circuit build timeouts are counted as construction f before the 95% "right-censored" timeout interval, nc

If a circuit closes prematurely after construction but the client, this is counted as a failure.

Measuring path usage success ra

Clients maintain two usage counts for each of their
usage attempts, and a count of the number of succe

A usage attempt means any attempt to attach a stre

Usage success status is temporarily recorded by sta
success counts are not incremented until circuit clos
used if we receive a properly recognized RELAY cell
the current circuit purpose.

If subsequent stream attachments fail or time out, t
circuit is cleared, causing it once again to be regarde

Upon close by the client, all circuits that are still mar
using a RELAY_BEGIN cell constructed with a destina
a.b.c is a 24 bit random nonce. If we get a RELAY_CC
our nonce, the circuit is counted as successfully use

If any unrecognized RELAY cells arrive after the prob
counted as a usage failure.

If the stream failure reason codes DESTROY, TORPR
response to any stream attempt, such circuits are n
failures.

Prematurely closed circuits are not probed, and are

Scaling success counts

To provide a moving average of recent Guard activit
verify correctness, we periodically "scale" the succes
scale factor between 0 and 1.0.

Scaling is performed when either usage or construc
parametrized value.

To avoid error due to scaling during circuit construc
are subtracted from the usage counts before scaling

Parametrization

The following consensus parameters tune various a

```
pb_mincircs
  Default: 150
  Min: 5
  Effect: This is the minimum number of
          at least 2 hops before we beg

pb_noticepct
  Default: 70
  Min: 0
  Max: 100
  Effect: If the circuit success rate fa
          we emit a notice log message.

pb_warnpct
  Default: 50
  Min: 0
  Max: 100
  Effect: If the circuit success rate fa
          we emit a warn log message.

pb_extremepct
  Default: 30
  Min: 0
  Max: 100
  Effect: If the circuit success rate fa
          we emit a more alarmist warnin
          pb_dropguard is set to 1, we a
          guard.

pb_dropguards
  Default: 0
  Min: 0
  Max: 1
  Effect: If the circuit success rate fa
          when pb_dropguard is set to 1,
          guard.

pb_scalecircs
  Default: 300
  Min: 10
  Effect: After this many circuits have
          Tor performs the scaling descr
      ["Scaling success counts"](#scalin

pb_multfactor and pb_scalefactor
  Default: 1/2
  Min: 0.0
  Max: 1.0
  Effect: The double-precision result ob
          pb_multfactor/pb_scalefactor
          counts to scale them.

pb_minuse
  Default: 20
  Min: 3
  Effect: This is the minimum number of
```

```
            use before we begin evaluating

  pb_noticeusepct
    Default: 80
    Min: 3
    Effect: If the circuit usage success r
            we emit a notice log message.

  pb_extremeusepct
    Default: 60
    Min: 3
    Effect: If the circuit usage success r
            we emit a warning log message.
            guard if pb_dropguards is set.

  pb_scaleuse
    Default: 100
    Min: 10
    Effect: After we have attempted to use
            Tor performs the scaling descr
            ["Scaling success counts"](#sc
```

Known barriers to enforcement

Due to intermittent CPU overload at relays, the norr completion is highly variable. The Guard-dropping v deployed until the ntor circuit handshake is enablec induced failure is better understood.

Tor Guard Specification

Introduction and motivation

Tor uses entry guards to prevent an attacker who co
from observing a fraction of every user's traffic. If us
uniformly at random from the list of servers every ti
adversary who had (k/N) of the network would dear
after a given user had built C circuits, the attacker w
probability 1-(1-F)^C. With large C, the attacker woul
with probability 1.

To prevent this from happening, Tor clients choose
3). These guard nodes are the only nodes that the cl
not compromised, the user's paths are not compror

This specification outlines Tor's guard housekeeping
following goals:

```
  - Heuristics and algorithms for determin
    are chosen should be kept as simple and
    possible.

  - Clients in censored regions or who are
    firewall who connect to the Tor network
    any significant disadvantage in terms c
    usability.

  - Tor should make a best attempt at disco
    appropriate behavior, with as little us
    configuration as possible.

  - Tor clients should discover usable guar
    delay.

  - Tor clients should resist (to the exter
    that try to force them onto compromised

  - Should maintain the load-balancing offe
    algorithm
```

State instances

In the algorithm below, we describe a set of persiste
These variables should be treated as an object, of w

In particular, we specify the use of three particular i

A. UseBridges

```
    If UseBridges is set, then we replace t
    [Sec:GUARDS] below with the list of con
    bridges.  We maintain a separate persis
    {SAMPLED_GUARDS} and {CONFIRMED_GUARDS}
    values for the UseBridges case.

    In this case, we impose no upper limit

  B. EntryNodes / ExcludeNodes / Reachable
      FascistFirewall / ClientUseIPv4=0

    If one of the above options is set, and
    then we compare the fraction of usable
    to the total number of guards in the co

    If this fraction is less than {MEANINGF
    we use a separate instance of the state

    (While Tor is running, we do not change
    the separate instance of the state and
    unless the fraction of usable guards is
    lower than, {MEANINGFUL_RESTRICTION_FRA
    from flapping back and forth between in
    hit {MEANINGFUL_RESTRICTION_FRAC} exact

    If this fraction is less than {EXTREME_
    separate instance of the state, and war

    [TODO: should we have a different insta
    restricted options?]

 C. Default

    If neither of the above variant-state
    we use a default instance.
```

Circuit Creation, Entry G (1000 foot view)

A circuit in Tor is a path through the network conne high-level, a three-hop exit circuit will look like this:

Client <-> Entry Guard <-> Middle Node <-> Exit Nod

Entry guards are the only nodes which a client will c nodes by which traffic exits the Tor network in orde destination.

Path selection

For any multi-hop circuit, at least one entry guard ar exit node is required if traffic will exit the Tor netwo relay listed in a consensus could be used for any of specification defines how entry guards specifically s opposed to middle or exit nodes.

Managing entry guards

At a high level, a relay listed in a consensus will mov process from initial selection to eventual usage as a

```
relays listed in consensus
          |
        sampled
        |     |
   confirmed   filtered
        |     |      |
        primary      usable_filtered
```

Relays listed in the latest consensus can be samplec "Guard" flag. Sampling is random but weighted by a bandwidth-weights (Wgg if guard only, Wgd if guard

Once a path is built and a circuit established using tl Until this point, guards are first sampled and then fi our current configuration (see SAMPLED and FILTER usable_filtered if the guard is not primary but can b

It is always preferable to use a primary guard when
reduce guard churn; only on failure to connect to ex
be used.

Middle and exit node selection

Middle nodes are selected at random from relays lis
by bandwidth and bandwidth-weights. Exit nodes ar
relays with a sufficiently permissive exit policy.

Circuit Building

Once a path is chosen, Tor will use this path to builc

If the circuit is built successfully, Tor will either use i
circuit with a more preferred guard if there's a good
one.

If the circuit fails in a way that makes us conclude th
is marked as unreachable, the circuit is closed, and

The algorithm

The guards listed in the current

By {set:GUARDS} we mean the set of all guards in th
for all circuits and directory requests. (They must ha
Guard.)

Rationale

We require all guards to have the flags that we pote
all guards are usable for all circuits.

The Sampled Guard Set.

We maintain a set, {set:SAMPLED_GUARDS}, that pe
subset of the nodes ordered by a sample idx that w
consensus at some point. For each such guard, we r

```
    - {pvar:ADDED_ON_DATE}: The date on wh
      sampled_guards.

      We set this value to a point in the p
      RAND(now, {GUARD_LIFETIME}/10). See
      Appendix [RANDOM] below.

    - {pvar:ADDED_BY_VERSION}: The version
      sampled_guards.

    - {pvar:IS_LISTED}: Whether it was list
      the _most recent_ consensus we have s

    - {pvar:FIRST_UNLISTED_AT}: If IS_LISTE
      of the earliest consensus in which th
      have not seen it listed in any later
      We randomize this to a point in the p
        RAND(added_at_time, {REMOVE_UNLISTE
```

For each guard in {SAMPLED_GUARDS}, we also reco

```
- {tvar:last_tried_connect}: A 'last tr
  time.  Default 'never'.

- {tvar:is_reachable}: an "is reachable
  possible values { <state:yes>, <state
  Default '<maybe>.'

        [Note: "yes" is not strictly r
         making it distinct from "mayb
         logic clearer.  A guard is "m
         worth trying. A guard is "yes
         it and succeeded.]

        [Note 2: This variable is, in
        of different context-sensitive
        on the _purpose_ for which we
        When we are selecing a guard f
        circuit, we look at the regula
        But when we are selecting the
        directory circuit, we also loo
   of {is_reachable} that tracks whet
   downloads of the types we are mak
   recently.]

- {tvar:failing_since}: The first time
  connect to this guard. Defaults to "n
  "never" when we successfully connect

- {tvar:is_pending} A "pending" flag.
  are trying to build an exploratory ci
  guard, and we don't know whether it w

- {tvar:pending_since}: A timestamp.  S
  {tvar:is_pending} to true; cleared wh
  {tvar:is_pending} to false. NOTE
```

We require that {SAMPLED_GUARDS} contain at leas from the consensus (if possible), but not more than number of guards in the consensus, and not more t if the maximum would be smaller than {MIN_FILTER {MIN_FILTERED_SAMPLE}.)

To add a new guard to {SAMPLED_GUARDS}, pick ar {SAMPLED_GUARDS}), according to the path selectio

We remove an entry from {SAMPLED_GUARDS} if:

```
  * We have a live consensus, and {IS_LIS
    {FIRST_UNLISTED_AT} is over {REMOVE_U
    days in the past.

OR

  * We have a live consensus, and {ADDED_
    {GUARD_LIFETIME} ago, *and* {CONFIRME
    "never", or over {GUARD_CONFIRMED_MIN
```

Note that {SAMPLED_GUARDS} does not depend on
we can't actually connect to any of these guards.

Rationale

The {SAMPLED_GUARDS} set is meant to limit the to
connect to in a given period. The upper limit on its s
many guards.

The first expiration mechanism is there so that our
accumulate so many dead guards that we cannot ac

The second expiration mechanism makes us rotate

Ordering the {SAMPLED_GUARDS} set in the order in
picking guards from that set according to this order
closer to offer the expected usage of the guard nod

The ordering also improves on another objective of
adversary pushing clients over compromised guards
clients to exhaust all their initial {SAMPLED_GUARDS
newly deployed adversary node.

The Usable Sample

We maintain another set, {set:FILTERED_GUARDS}, t
from:

```
- {SAMPLED_GUARDS}
- our current configuration,
- the path bias information.
```

A guard is a member of {set:FILTERED_GUARDS} if a

```
- It is a member of {SAMPLED_GUARDS},
  true.
- It is not disabled because of path b
- It is not disabled because of Reacha
  the ClientUseIPv4 setting, the Clien
  the FascistFirewall setting, or some
  option that prevents using some addr
- It is not disabled because of Excluc
- It is a bridge if UseBridges is true
  bridge if UseBridges is false.
- Is included in EntryNodes if EntryNo
  UseBridges is not. (But see 2.B abov
```

We have an additional subset, {set:USABLE_FILTERE the subset of {FILTERED_GUARDS} where {is_reacha

We try to maintain a requirement that {USABLE_FILT {MIN_FILTERED_SAMPLE} elements:

```
Whenever we are going to sample from {US
and it contains fewer than {MIN_FILTEREI
add new elements to {SAMPLED_GUARDS} unt
is true:

  * {USABLE_FILTERED_GUARDS} is large en
OR
  * {SAMPLED_GUARDS} is at its maximum s

** Rationale **
```

These filters are applied *after* sampling: if we applie sample would reflect the set of filtering restrictions

The confirmed-guard list.

[formerly USED_GUARDS]

We maintain a persistent ordered list, {list:CONFIRM we have used before, in our preference order of usi {SAMPLED_GUARDS}. For each guard in this list, we

- {pvar:IDENTITY} Its fingerprint.

```
- {pvar:CONFIRMED_ON_DATE} When we adde
  {CONFIRMED_GUARDS}.

  Randomized to a point in the past as
```

We append new members to {CONFIRMED_GUARDS
through a guard as "for user traffic."

Whenever we remove a member from {SAMPLED_G
{CONFIRMED_GUARDS}.

```
[Note: You can also regard the {CONFIRM
total ordering defined over a subset of
```

Definition: we call Guard A "higher priority" than and
both reachable, we would rather use A. We define p

```
  * Every guard in {CONFIRMED_GUARDS} has
    than every guard not in {CONFIRMED_GUA

  * Among guards in {CONFIRMED_GUARDS}, th
    on the {CONFIRMED_GUARDS} list has a h

  * Among guards that do not appear in {CO
    {is_pending}==true guards have higher

  * Among those, the guard with earlier {l
    has higher priority.

  * Finally, among guards that do not appe
    {CONFIRMED_GUARDS} with {is_pending==f
    priority.

** Rationale **
```

We add elements to this ordering when we have act
circuit. We could mark them at some other time (su
them, or when we actually connect to them), but thi
to a guard before we actually use it for sensitive tra

The Primary guards

We keep a run-time non-persistent ordered list of {I
of {FILTERED_GUARDS}. It contains {N_PRIMARY_GU

To compute primary guards, take the ordered inters
{FILTERED_GUARDS}, and take the first {N_PRIMARY
fewer than {N_PRIMARY_GUARDS} elements, appen
PRIMARY_GUARDS chosen from ({FILTERED_GUARD
in "sample order" (that is, by {ADDED_ON_DATE}).

Once an element has been added to {PRIMARY_GUA

replaced by some element from {CONFIRMED_GUAI
becomes confirmed and not every primary guard is
guards list is regenerated, first from the confirmed g
non-confirmed primary guards.

Note that {PRIMARY_GUARDS} do not have to be in
might be unreachable.

Rationale

These guards are treated differently from other gua
use it right away. For other guards {FILTERED_GUAR
we might first double-check whether perhaps one o
all.

Retrying guards.

(We run this process as frequently as needed. It can
time.)

If a primary sampled guard's {is_reachable} status is
update its {is_reachable} status to `<maybe>` based c
{failing_since} time, and the {PRIMARY_GUARDS_RET

If a non-primary sampled guard's {is_reachable} sta
to update its {is_reachable} status to `<maybe>` base
{failing_since} time, and the {GUARDS_RETRY_SCHEI

Rationale

An observation that a guard has been 'unreachable'
since we can't infer that it's unreachable now from t
minutes ago.

Circuit status

Sometimes the guard selection algorithm will returr
happy to use; but in other cases, the guard selectior
shouldn't use without gathering additional informat

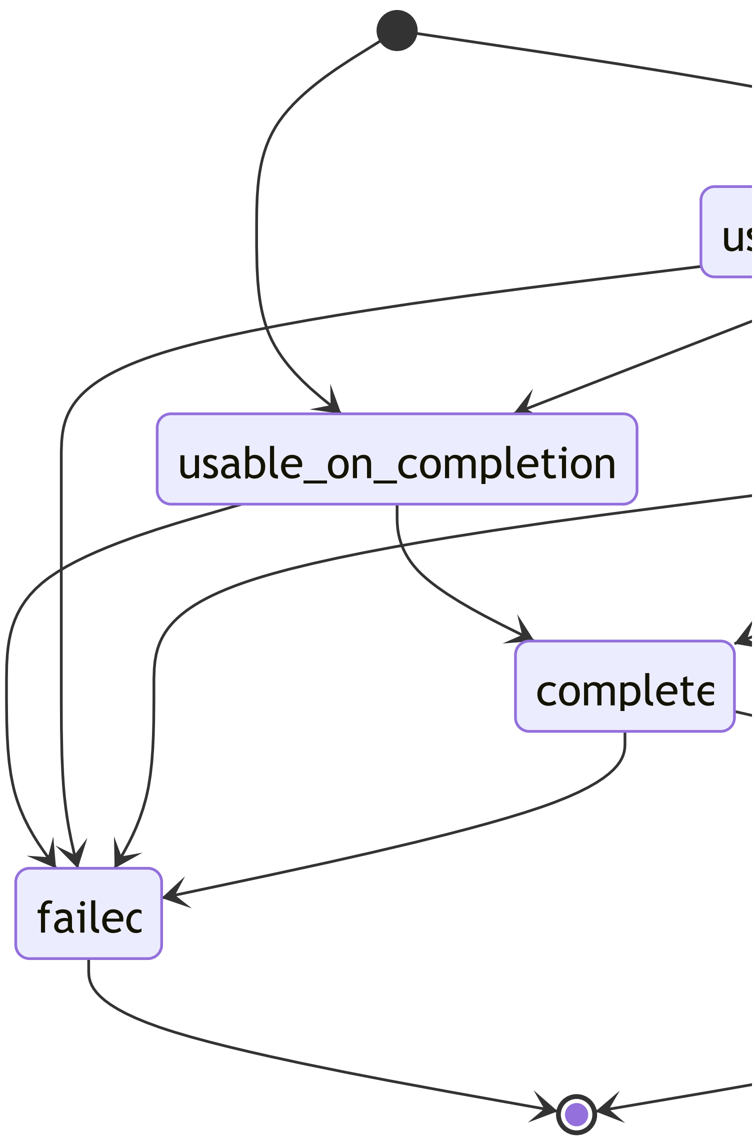
From the point of view of guard selection, every circ
of these states:

- `<state:usable_on_completion>`
- `<state:usable_if_no_better_guard>`
- `<state:waiting_for_better_guard>`
- `<state:complete>`

You may only attach streams to `<complete>` circuits
RENDEZVOUS messages, ESTABLISH_INTRO messag
`<complete>` circuits.)

The per-circuit state machine is:

- New circuits are `<usable_on_completion>` or
- A `<usable_on_completion>` circuit may becom
- A `<usable_if_no_better_guard>` circuit may b
may become `<waiting_for_better_guard>` ; o
- A `<waiting_for_better_guard>` circuit will be
or will fail.
- A `<complete>` circuit remains `<complete>` unt



Each of these transitions is described in sections bel

Selecting guards for circuits. [Se

Now that we have described the various lists of gua
chosen for each circuit.

We keep, as global transient state:

- {tvar:last_time_on_internet} -- the last time at v
  connected to a guard. At startup we set this to

As an input to the algorithm, we take a list of *restrict*
relays or families that we need to avoid.

Here is the algorithm. It is given as a series of sub-al
preference from best case to worst case. When we v

pick a guard:

- In the base case, if any entry in PRIMARY_GUAI `<maybe>` or `<yes>`, consider only such guards guards".

  Start by considering the the first {NUM_USABL {NUM_USABLE_PRIMARY_DIRECTORY_GUARDS any guards that do not follow our path selectic list. If that temporary list contains at least one uniformly at random.

  If the temporary list contains no guards, returr primary guard that does obey the path restrict

  When selecting a guard according to this appr `<usable_on_completion>`.

  [Note: We do not use {is_pending} on primary build multiple circuits through them before we and since we will not use any non-primary gua guards are all down. (XX is this good?)]

- Otherwise, if the ordered intersection of {CON {USABLE_FILTERED_GUARDS} is nonempty, ret that has {is_pending} set to false. Set its value {pending_since} to the current time. The circui `<usable_if_no_better_guard>`. (If all entries h one.)

- Otherwise, if there is no such entry, select a m {USABLE_FILTERED_GUARDS} in sample order. set its {pending_since} to the current time. The `<usable_if_no_better_guard>`.

- Otherwise, in the worst case, if USABLE_FILTER exhausted all the sampled guards. In this case `<maybe>` reachable so that we can keep on try

Whenever we select a guard for a new circuit attem time for the guard to 'now.'

In some cases (for example, when we need a certair to avoid using a certain exit as a guard), we need to single circuit. When this happens, we remember the choosing the guard for that circuit, since we will nee [UPDATE_WAITING].).

Rationale

We're getting to the core of the algorithm here. Our

1. If it's possible to use a primary guard, we do.
2. We probably use the first primary guard.

So we only try non-primary guards if we're pretty su
down, and we only try a given primary guard if the e

When we *do* try non-primary guards, however, we o
give it a chance to succeed or fail. If ever such a circ
we're pretty sure that it's the best guard we're gettir

[XXX timeout.]

When a circuit fails.

When a circuit fails in a way that makes us conclude
the following steps:

```
        * Set the guard's {is_reachable} status
          {is_pending} set to true, we make it
          {pending_since}.

        * Close the circuit, of course.  (This
          consideration by the algorithm in [UF

        * Update the list of waiting circuits.
          below.)
```

[Note: the existing Tor logic will cause us to create n
these steps; and also see [ON_CONSENSUS].]

[Note 2: In the case of a one-hop circuit made for a
request to fail *after* the circuit is built: for example, i
we are told "404". In this case, we mark the appropr
`{is_reachable}` instance for that guard to `<no>` .]

> C tor implements the above "note 2" by treating r
> if they had an extra type of restriction, rather tha
> `{is_reachable}` . (For more on restrictions, see "
> requires the C tor impementation to special-case
> treated the same way as an `{is_reachable}` var

Rationale

See [SELECTING] above for rationale.

When a circuit succeeds

When a circuit succeeds in a way that makes us con
take these steps:

```
      * We set its {is_reachable} status to <
      * We set its {failing_since} to "never'
      * If the guard was {is_pending}, we cle
        and set {pending_since} to false.
      * If the guard was not a member of {CON
        it to the end of {CONFIRMED_GUARDS}.

      * If this circuit was <usable_on_comple
        now <complete>. You may attach stream
        and use it for hidden services.

      * If this circuit was <usable_if_no_bet
        <waiting_for_better_guard>.  You may
        Then check whether the {last_time_on_
        {INTERNET_LIKELY_DOWN_INTERVAL} secon

           * If it is, then mark all {PRIMARY
             reachable.

           * If it is not, update the list of
             [UPDATE_WAITING] below)
```

[Note: the existing Tor logic will cause us to create m
these steps; and see [ON_CONSENSUS].]

Rationale

See [SELECTING] above for rationale.

Updating the list of waiting circu

We run this procedure whenever it's possible that a
might be ready to be called `<complete>`.

```
* If any circuit C1 is <waiting_for_better
    * All primary guards have reachable st
    * There is no circuit C2 that "blocks'
  Then, upgrade C1 to <complete>.

Definition: In the algorithm above, C2 "bl
    * C2 obeys all the restrictions that (
    * C2 has higher priority than C1, AND
    * Either C2 is <complete>, or C2 is <w
      or C2 has been <usable_if_no_better_
      {NONPRIMARY_GUARD_CONNECT_TIMEOUT} s

We run this procedure periodically:

* If any circuit stays in <waiting_for_bet
  for more than {NONPRIMARY_GUARD_IDLE_TIM
  time it out.

   **Rationale**
```

If we open a connection to a guard, we might want t that it's the best we can do), or we might want to wa circuit which we like better will finish.

When we mark a circuit `<complete>`, we don't close immediately: we might decide to use them after all i before {NONPRIMARY_GUARD_IDLE_TIMEOUT} secc

Without a list of waiting circuits

As an alternative to the section [SECTION:UPDATE_V a new way to maintain guard status independently (This formulation gives a result equivalent or similar the necessary communications between the guard a

As before, when all primary guards are Unreachable We select the first such guard (in preference order) Pending. Whenever we give out such a guard, if the call that guard "Pending" with its {is_pending} flag, u fails. We remember when the guard became Pendir

After completing a circuit, the implementation must guard's usability status may be "usable", "unusable" according to these rules:

1. Primary guards are always usable.

2. Non-primary guards are usable *for a given circu*

preference list is either unsuitable for that circ
restrictions), or marked as Unreachable, or has
`{NONPRIMARY_GUARD_CONNECT_TIMEOUT}` .

Non-primary guards are not usable *for a given*
preference list is suitable for the circuit *and* Re

Non-primary guards are unusable if they have
`{NONPRIMARY_GUARD_IDLE_TIMEOUT}` seconds.

3. If a circuit's guard is not usable or unusable im
instead, it is kept (but not used) until the guard

Whenever we get a new consens

We update {GUARDS}.

For every guard in {SAMPLED_GUARDS}, we update
{FIRST_UNLISTED_AT}.

[**] We remove entries from {SAMPLED_GUARDS} if
sampled-guards expiration rules. If they were in {CC
them from {CONFIRMED_GUARDS}.

We recompute {FILTERED_GUARDS}, and everything
{USABLE_FILTERED_GUARDS}, and {PRIMARY_GUAR

(Whenever one of the configuration options that aff
the process above, starting at the [**] line.)

```
4.11. Deciding whether to generate a new circ
  [Section:NEW_CIRCUIT_NEEDED]
```

We generate a new circuit when we don't have enou
to handle a given stream, or an expected stream.

For the purpose of this rule, we say that `<waiting_f`
neither built nor in-progress; that `<complete>` circu
are in-progress.

```
4.12. When we are missing descriptors.
   [Section:MISSING_DESCRIPTORS]
```

We need either a router descriptor or a microdescri
a guard. If we do not have such a descriptor for a gu

one-hop directory fetches, but not for longer circuits

(Also, when we are missing descriptors for our first ·
primary guards, we don't build circuits at all until we

Appendices

Acknowledgements

This research was supported in part by NSF grants C
CNS-1526306, CNS-1619454, and CNS-1640548.

Parameters with suggested valu

(All suggested values chosen arbitrarily)

{param:MAX_SAMPLE_THRESHOLD} -- 20%

{param:MAX_SAMPLE_SIZE} -- 60

{param:GUARD_LIFETIME} -- 120 days

```
{param:REMOVE_UNLISTED_GUARDS_AFTER} -- 20
  [previously ENTRY_GUARD_REMOVE_AFTER]

{param:MIN_FILTERED_SAMPLE} -- 20

{param:N_PRIMARY_GUARDS} -- 3

{param:PRIMARY_GUARDS_RETRY_SCHED}

   We recommend the following schedule, wh
   used in Arti:

   -- Use the "decorrelated-jitter" algori
      section 5.5 where `base_delay` is 30
      is 6 hours.

   This legacy schedule is the one used in

   -- every 10 minutes for the first six h
   -- every 90 minutes for the next 90 hou
   -- every 4 hours for the next 3 days,
   -- every 9 hours thereafter.

{param:GUARDS_RETRY_SCHED} --

   We recommend the following schedule, wh
   used in Arti:

   -- Use the "decorrelated-jitter" algori
      section 5.5 where `base_delay` is 10
      is 36 hours.

   This legacy schedule is the one used in

   -- every hour for the first six hours,
   -- every 4 hours for the 90 hours,
   -- every 18 hours for the next 3 days,
   -- every 36 hours thereafter.

{param:INTERNET_LIKELY_DOWN_INTERVAL} -- 1

{param:NONPRIMARY_GUARD_CONNECT_TIMEOUT} -

{param:NONPRIMARY_GUARD_IDLE_TIMEOUT} -- 1

{param:MEANINGFUL_RESTRICTION_FRAC} -- .2

{param:EXTREME_RESTRICTION_FRAC} -- .01

{param:GUARD_CONFIRMED_MIN_LIFETIME} -- 60

{param:NUM_USABLE_PRIMARY_GUARDS} -- 1

{param:NUM_USABLE_PRIMARY_DIRECTORY_GUARDS
```

Random values

Frequently, we want to randomize the expiration tin
for an observer to match it to its start time. We do tl
little, so that we only need to remember a fixed exp

By RAND(now, INTERVAL) we mean a time between
chosen uniformly at random.

Persistent state format

The persistent state format doesn't need to be part
implementations can do it differently. Nonetheless,

The "state" file contains one Guard entry for each sa
guard state (see section 2). The value of this Guard e
entries, where K contains any nonspace character e:
characters.

Implementations must retain any unrecognized K=V
they regenerate the state file.

The order of K=V entries is not allowed to matter.

Recognized fields (values of K) are:

```
        "in" -- the name of the guard state i
        sampled guard is in.  If a sampled gu
        states instances, it appears twice, w
        field each time. Required.

        "rsa_id" -- the RSA id digest for thi
        hex. Required.

        "bridge_addr" -- If the guard is a br
        port (this can be the ORPort or a plu
Optional.

        "nickname" -- the guard's nickname, i

        "sampled_on" -- the date when the gua

        "sampled_by" -- the Tor version that 
        Optional.

        "unlisted_since" -- the date since wh
        unlisted. Optional.

        "listed" -- 0 if the guard is not lis

        "confirmed_on" -- date when the guard
        confirmed. Optional.

        "confirmed_idx" -- position of the gu
        list. Optional.

        "pb_use_attempts", "pb_use_successes"
        "pb_circ_successes", "pb_successful_c
        "pb_collapsed_circuits", "pb_unusable
        "pb_timeouts" -- state for the circui
        given in decimal fractions. Optional.
```

All dates here are given as a (spaceless) ISO8601 cor
2016-11-29T19:39:31).

Tor Vanguards Specificat

Introduction and motivation

A guard discovery attack allows attackers to determ
hidden service protocol provides an attack vector fo
anyone can force an HS to construct a 3-hop circuit
until one of the adversary's middle relays eventually
attacks are also possible to perform against clients,
repeated connections to multiple unique onion serv

The adversary must use a protocol side channel to c
this position (see Proposal #344), and then learns th

When a successful guard discovery attack is followe
guard relay, the onion service or onion client can be
analytics data purchase can be (and has been) used
interacting with the Guard relay directly (see again F

This specification assumes that Tor protocol side ch
undetectable, for simplicity in reasoning about expe
100% accurate side channels exist in silent form, in t

As work on addressing Tor's protocol side channels
application-layer activity that can be monitored and
opposed to silent and unobservable side channel ac
Application-layer side channels are also expected to
protocol side channels, due to the possibility of false
application activity elsewhere on the Tor network. D
original assumption of 100% accuracy, for simplicity

Overview

In this specification, we specify two forms of a multi
lived services, called Full Vanguards, and one for on
called Vanguards-Lite.

Both approaches use a mesh topology, where circui
preceding layer to any relay in a subsequent layer.

The core difference between these two mechanisms

additional layers of fixed vanguard relays, which res
latency. In contrast, Vanguards-Lite has only one ad
and preserves the original path lengths in use by on
comes with a performance cost, where as Vanguard
of the two approaches also differ.

Vanguards-Lite MUST be the default for all onion se
Vanguards SHOULD be available as an optional con

Neither system applies to Exit activity.

Terminology

Tor's original guards are called First Layer Guards.

The first layer of vanguards is at the second hop, an

The second layer of vanguards is at the third hop, a

Visualizing Full Vanguards

Full Vanguards pins these two middle positions into
a layer can be used in that position in a circuit, as fo

```
                   -> vanguard_2A
                                 -> vang
      -> guard_1A  -> vanguard_2B -> vang
   HS                            -> vang
      -> guard_1B  -> vanguard_2C -> vang
                                 -> vang
                   -> vanguard_2D -> vang
```

Additionally, to avoid trivial discovery of the third la
insert an extra middle relay after the third layer gua
circuits, and service-side rendezvous circuits. This m
(C) and Service (S) side look like this:

```
Client hsdir:  C - G - L2 - L3 - M - HSD
Client intro:  C - G - L2 - L3 - M - I
Client rend:   C - G - L2 - L3 - R
Service hsdir: S - G - L2 - L3 - HSDIR
Service intro: S - G - L2 - L3 - I
Service rend:  S - G - L2 - L3 - M - R
```

Visualizing Vanguards-Lite

Vanguards-Lite uses only one layer of vanguards:

```
                    -> vanguard_2A

     -> guard_1A  -> vanguard_2B
  HS
     -> guard_1B  -> vanguard_2C

                    -> vanguard_2D
```

This yields shorter path lengths, of the following for

```
Client hsdir:  C -> G -> L2 -> M -> HSD
Client intro:  C -> G -> L2 -> M -> Intr
Client rend:   C -> G -> L2 -> Rend
Service hsdir: C -> G -> L2 -> M -> HSD
Service intro: C -> G -> L2 -> M -> Intr
Service rend:  C -> G -> L2 -> M -> Renc
```

Alternatives

An alternative to vanguards for client activity is to re that a Tor client is allowed to connect to, in a certair explored in Onion Not Found.

We have opted not to deploy this defense, for three

1. It does not generalize to the service-side of on
2. Setting appropriate rate limits on the number a page for Tor Browser is difficult. Sites like Fa for various content elements on a single page.
3. It is even more difficult to limit the number of applications, such as cryptocurrency wallets, m deployed on top of onion services that connec Ricochet).

Full Vanguards

Full Vanguards is intended for use by long-lived onio
operation for longer than one month.

Full Vanguards achieves this longer expected durati
fixed relays, of different rotation periods.

The rotation period of the first vanguard layer (layer
requires an extremely long and persistent Sybil atta

The rotation period of the second vanguard layer (la
enough to force the adversary to perform a Sybil att
attempting to coerce these relays.

Threat model, Assumptions, and

Consider an adversary with the following powers:

```
    - Can launch a Sybil guard discovery att
      rendezvous circuit. The slower the rot
      position, the longer the attack takes.
      percentage of the network is controlle
      attack runs.

    - Can compromise additional relays on th
      takes time and potentially even coerc
      of discovery.
```

We also make the following assumptions about the

1. A Sybil attack is observable by both people mo
numbers of new relays, as well as vigilant hidd
large amounts of traffic sent towards the hidd
circuits.

2. A Sybil attack requires either a protocol side ch
side channel in order to determine successful
adversary is attempting to discover. When Tor'
dealt with, this will be both observable and cor
operators.

3. The adversary is strongly disincentivized from
may prove useless, as active compromise atte
adversary than a Sybil attack in terms of being

adversary is unlikely to attempt to compromis
in use for only a short period of time.

Given this threat model, our security parameters we
of guards should take a very long period of time to a
attack and hence require a relay compromise attack

On the other hand, the outermost layer of guards (t
enough to *require* a Sybil attack. If the adversary we
coerce these relays after they are discovered, their r
that the adversary has a very high probability of the

Design

When a hidden service picks its guard relays, it also
NUM_LAYER2_GUARDS-sized set of middle relays fo
NUM_LAYER3_GUARDS-sized set of middle relays fo

When a hidden service needs to establish a circuit to
rendezvous point, it uses relays from `second_guard`
and relays from `third_guard_set` as third hop of th

A hidden service rotates relays from the 'second_gu
between MIN_SECOND_GUARD_LIFETIME hours and
hours, chosen for each layer 2 relay. Implementatio
configuration option to pin specific relays, similar to
Guards.

A hidden service rotates relays from the 'third_guar
MIN_THIRD_GUARD_LIFETIME and MAX_THIRD_GUA
the max(X,X) distribution, chosen for each relay. Thi
that there is some probability of a very short rotatio
compromise/coercion, but biased towards the longe
lengthy Sybil attack. For this reason, users SHOULD

Each relay's rotation time is tracked independently,
of the primary and second-level guards.

The selected vanguards and their rotation timestam

Parameterization

We set NUM_LAYER2_GUARDS to 4 relays and NUM_

We set MIN_SECOND_GUARD_LIFETIME to 30 days, a
to 60 days inclusive, for an average rotation rate of 4
This range was chosen to average out to half of the
strong motivation for it otherwise, other than to be
the Guard rotation, and longer periods MAY be prov

From the Sybil rotation table in statistical analysis, w
be seen that this means that the Sybil attack on laye
18*45 days (2.2 years) for the 1% adversary, 180 day
for the 10% adversary, with a 45 day average rotatio

If this range is set equal to the Guard rotation perio
Sybil success requires 18*90 days (4.4 years) for the
the 5% adversary, and 2*90 days (6 months) for the

We set MIN_THIRD_GUARD_LIFETIME to 1 hour, and
hours inclusive, for an average rotation rate of 31.5
(Again, this wide range and bias is used to discourag
performing coercive attacks, as opposed to mountin
substantially is not recommended).

From the Sybil rotation table in statistical analysis, w
be seen that this means that the Sybil attack on laye
9*31.5 hours (15.75 days) for the 1% adversary, ~4 d
days for the 10% adversary.

See the statistical analysis for more analysis on thes

Vanguards-Lite

Vanguards-Lite is meant for short-lived onion servic
onion client activity.

It is designed for clients and services that expect to
month.

Design

Because it is for short-lived activity, its rotation time
in mind, using the max(X,X) skewed distribution.

We let NUM_LAYER2_GUARDS=4. We also introduce
`l2-number` that controls the number of layer2 guarc
guards).

No third layer of guards is used.

We don't write guards on disk. This means that the g
restarts.

Rotation Period

The Layer2 lifetime uses the max(x,x) distribution wi
maximum of 22 days. This makes the average lifetin
Significant extensions of this lifetime are not recom
in favor of coercive attacks.

From the Sybil Rotation Table, with NUM_LAYER2_G
means that the Sybil attack on Layer2 will complete
days) for the 1% adversary, 4*14 days (two months)
(one month) for the 10% adversary.

Path Construction

Both vanguards systems use a mesh topology: this r
each layer independently, allowing paths from any r
layer.

Selecting Relays

Vanguards relays are selected from relays with the S

Tor replaces a vanguard whenever it is no longer list
with the goal that we will always have the right num

For implementation reasons, we also replace a vang
because the path selection logic wants middle node
building preemptive vanguard-using circuits.

The design doesn't have to be this way: we might ins
in our list as long as possible, and continue to use th
This tradeoff is similar to the one in Bug #17773, ab
Guards if they lose the Guard flag -- and Tor's curren
too.

Path Restriction Changes

Path restrictions, as well as the ordering of their app
problematic, resulting in information leaks with this
disable many of them for onion service circuits.

In particular, we allow the following:

1. Nodes from the same /16 and same family for
2. Guard nodes can be chosen for RP/IP/HSDIR
3. Guard nodes can be chosen for hop before RP

The first change prevents the situation where paths
the same subnet and/or node family, or if a layer co
family is the same as the RP/IP/HSDIR. It also preven
guard based on the family or subnet of the IP, HSDII
permissive behavior are possible: For example, each
consist solely of the same family or /16, but this pro

circuits).

The second change prevents an adversary from forc
by enumerating all guard-flagged nodes as the RP. T
services support conflux.

The third change prevents an adversary from learni
which nodes were not chosen for the hop before it.
services support conflux.

Vanguard Rotation Stati

Sybil rotation counts for a given

The probability of Sybil success for Guard discovery choosing 1 or more malicious middle nodes for a se time.

```
P(At least 1 bad middle) = 1 - P(All Good M
                         = 1 - P(One Good m
                         = 1 - (1 - c/n)^(r
```

c/n is the adversary compromise percentage

In the case of Vanguards, num_middles is the numb given time period. This is a function of the number c well as the number of rotations (r).

```
P(At least one bad middle) = 1 - (1 - c/n)^
```

Here's detailed tables in terms of the number of rot success rate for certain number of guards.

```
  1.0% Network Compromise:
   Sybil Success   One   Two  Three  Four  F
Twelve  Sixteen
    10%             11     6     4     3     3
1
    15%             17     9     6     5     4
2
    25%             29    15    10     8     6
2
    50%             69    35    23    18    14
5
    60%             92    46    31    23    19
6
    75%            138    69    46    35    28
9
    85%            189    95    63    48    38
12
    90%            230   115    77    58    46
15
    95%            299   150   100    75    60
19
    99%            459   230   153   115    92
29

  5.0% Network Compromise:
   Sybil Success   One   Two  Three  Four  F
Twelve  Sixteen
    10%              3     2     1     1     1
1
    15%              4     2     2     1     1
1
    25%              6     3     2     2     2
1
    50%             14     7     5     4     3
1
    60%             18     9     6     5     4
2
    75%             28    14    10     7     6
2
    85%             37    19    13    10     8
3
    90%             45    23    15    12     9
3
    95%             59    30    20    15    12
4
    99%             90    45    30    23    18
6

  10.0% Network Compromise:
   Sybil Success   One   Two  Three  Four  F
Twelve  Sixteen
    10%              2     1     1     1     1
1
    15%              2     1     1     1     1
1
    25%              3     2     1     1     1
```

```
1
    50%              7     4     3     2     2
1
    60%              9     5     3     3     2
1
    75%             14     7     5     4     3
1
    85%             19    10     7     5     4
2
    90%             22    11     8     6     5
2
    95%             29    15    10     8     6
2
    99%             44    22    15    11     9
3
```

The rotation counts in these tables were generated

Skewed Rotation Distribution

In order to skew the distribution of the third layer gu
max(X,X) for the distribution, where X is a random va
uniform distribution.

Here's a table of expectation (arithmetic means) for
0..N-1). The table was generated with the following p

```
def ProbMinXX(N, i): return (2.0*(N-i)-1)/(
def ProbMaxXX(N, i): return (2.0*i+1)/(N*N)

def ExpFn(N, ProbFunc):
  exp = 0.0
  for i in range(N): exp += i*ProbFunc(N, i
  return exp
```

The current choice for second-layer Vanguards-Lite
current choice for third-layer Full Vanguards is note

```
Range  Min(X,X)   Max(X,X)
22      6.84        14.16**
23      7.17        14.83
24      7.51        15.49
25      7.84        16.16
26      8.17        16.83
27      8.51        17.49
28      8.84        18.16
29      9.17        18.83
30      9.51        19.49
31      9.84        20.16
32      10.17       20.83
33      10.51       21.49
34      10.84       22.16
35      11.17       22.83
36      11.50       23.50
37      11.84       24.16
38      12.17       24.83
39      12.50       25.50
40      12.84       26.16
40      12.84       26.16
41      13.17       26.83
42      13.50       27.50
43      13.84       28.16
44      14.17       28.83
45      14.50       29.50
46      14.84       30.16
47      15.17       30.83
48      15.50       31.50***
```

The Cumulative Density Function (CDF) tells us the p
be in use after a given number of time units have pa

Because the Sybil attack on the third node is expect
second node's rotation period with uniform probab
probability that a second-level Guard node will still b
compute the probability distribution of the rotation
a uniformly random point in time. Let's call this P(R=

For P(R=r), the probability of the rotation duration d
a rotation duration, and the fraction of total time th
can be written as:

```
P(R=r) = ProbMaxXX(X=r)*r / \sum_{i=1}^N Pr
```

or in Python:

```
def ProbR(N, r, ProbFunc=ProbMaxXX):
   return ProbFunc(N, r)*r/ExpFn(N, ProbFur
```

For the full CDF, we simply sum up the fractional pro

durations. For rotation durations less than t days, w
that period to the density function. For durations d
fraction of that rotation period's selection probabilit
the density. In other words:

```
def FullCDF(N, t, ProbFunc=ProbR):
  density = 0.0
  for d in range(N):
    if t >= d: density += ProbFunc(N, d)
    # The +1's below compensate for 0-inde
    else: density += ProbFunc(N, d)*(float(
  return density
```

Computing this yields the following distribution for

```
 t          P(SECOND_ROTATION <= t)
 1               0.03247
 2               0.06494
 3               0.09738
 4               0.12977
 5               0.16207
10               0.32111
15               0.47298
20               0.61353
25               0.73856
30               0.84391
35               0.92539
40               0.97882
45               1.00000
```

This CDF tells us that for the second-level Guard rot
3.3% of the time, their third-level Sybil attack will pr
node that has only 1 day remaining before it rotates
day or less remaining, and 9.7% of the time, 3 days

Note that this distribution is still a day-resolution ap

Tor Padding Specificatio

Mike Perry, George Kadianakis

Note: This is an attempt to specify Tor as currently i
will implement improved algorithms.

This document tries to cover how Tor chooses to us
traffic patterns from external and internal observer
other approaches, but implementors should be awa
balancing implications of their choices.

Overview

Tor supports two classes of cover traffic: connectior
padding.

Connection-level padding uses the CELL_PADDING c
as circuit-level padding uses the RELAY_COMMAND_
CELL_PADDING is single-hop only and can be differe
relays ("internal" observers), but not by entities mor
("external" observers).

RELAY_COMMAND_DROP is multi-hop, and is not vis
because the relay command field is covered by circu
'recognized' field allows RELAY_COMMAND_DROP p
node in a circuit (as per Section 6.1 of tor-spec.txt).

Tor uses both connection level and circuit level padc
described in section 2. Circuit level padding is descri

The circuit-level padding system is completely ortho
padding. The connection-level padding system regai
data traffic, and hence the connection-level padding
overhead while the circuit-level padding system is a

Connection-level paddin

Background

Tor clients and relays make use of CELL_PADDING t
level metadata retention by ISPs and surveillance in

Such metadata retention is implemented by Interne
jFlow, Netstream, or IPFIX records. These records ar
form and then exported (often over plaintext) to a "
verbatim, or reduces their granularity further[1].

Netflow records and the associated data collection a
configurable, and have many modes of operation, e
high throughput. However, at ISP scale, per-flow rec
since they are the default, and also provide very hig
activity, second only to full packet and/or header ca

Per-flow records record the endpoint connection 5-t
bytes sent and received by that 5-tuple during a par
additional fields as well, but it is primarily timing an
us.

When configured to provide per-flow data, routers e
periodically for all active connections passing throu
the "active flow timeout" and the "inactive flow time

The "active flow timeout" causes the router to emit
active TCP session that continuously sends data. Th
routers is 30 minutes, meaning that a new record is
every 30 minutes, no matter what. This value can be
minutes on major routers.

The "inactive flow timeout" is used by routers to cre
inactive for some number of seconds. It allows rout
number of idle connections in memory, and instead
there is activity. This value ranges from 10 seconds t
appears as though no routers support a value lowe

For reference, here are default values and ranges (i
common routers, along with citations to their manu

Some routers speak other collection protocols than

use different timeouts for these protocols. Where th
noted.

```
                                Inactive Timeout
     Cisco IOS[3]                 15s (10-600s)
     Cisco Catalyst[4]            5min
     Juniper (jFlow)[5]           15s (10-600s)
     Juniper (Netflow)[6,7]       60s (10-600s)
     H3C (Netstream)[8]           60s (60-600s)
     Fortinet[9]                  15s
     MicroTik[10]                 15s
     nProbe[14]                   30s
     Alcatel-Lucent[2]            15s (10-600s)
```

The combination of the active and inactive netflow r
low-cost padding defense that causes what would o
at the router even before they are exported to the c
connection transmits data before the "inactive flow
continue to count the total bytes on that flow before
"active flow timeout".

This means that for a minimal amount of padding th
timeout" from expiring, it is possible to reduce the r
data to the total amount of bytes send and received
reduction in resolution for HTTP, IRC, XMPP, SSH, ar
traffic, especially when all user traffic in that time pe
connection (as it is with Tor).

Though flow measurement in principle can be bidire
directions between a pair of IPs) or unidirectional (c
another), we assume for safety that all measuremer
be sent by both parties in order to prevent record s

Implementation

Tor clients currently maintain one TLS connection to
application traffic, and make up to 3 additional conn
directory information.

We pad only the client's connection to the Guard no
treat Bridge node connections to the Tor network as
but otherwise not pad between normal relays.

Both clients and Guards will maintain a timer for all
connections. Every time a padding packet sent by ar

a timeout value from the max(X,X) distribution desc
is from 1.5 seconds to 9.5 seconds time range, subj
specified in Section 2.6.

(The timing is randomized to avoid making it obviou

If another cell is sent for any reason before this tim
random value.

If the connection remains inactive until the timer ex
be sent on that connection (which will also start a n

In this way, the connection will only be padded in a
idle in that direction, and will always transmit a pac
inactive timeout.

(In practice, an implementation may not be able to
on a given channel. For example, even though the c
call to `send(2)` , the kernel may still be buffering th
implementations should use a reasonable proxy fo
example, when the cell is queued. If this strategy is
observe the innermost (closest to the wire) queue t
queue is already nonempty, padding should not be
become empty.)

Padding Cell Timeout Distributio

To limit the amount of padding sent, instead of sam
uniformly, we instead sample it from max(X,X), whe

If X is a random variable uniform from 0..R-1 (where
variable Y = max(X,X) has Prob(Y == i) = (2.0*i + 1)/(R*R

Then, when both sides apply timeouts sampled fro
padding packet rate is now a third random variable:

The distribution of Z is slightly bell-shaped, but mos
out that Exp[Z] ~= Exp[X]. Here's a table of average

```
R        Exp[X]    Exp[Z]    Exp[min(X,X)
2000      999.5    1066        666.2
3000     1499.5    1599.5      999.5
5000     2499.5    2666       1666.2
6000     2999.5    3199.5     1999.5
7000     3499.5    3732.8     2332.8
8000     3999.5    4266.2     2666.2
10000    4999.5    5328       3332.8
15000    7499.5    7995       4999.5
20000    9900.5    10661      6666.2
```

Maximum overhead bounds

With the default parameters and the above distribu to send one padding cell every 5.5 seconds. This ave duplex (~52 bytes/sec in each direction), assuming a TLS+TCP+IP headers. For a client connection that re ~50 minute lifespan (governed by the circuit availab connection timeout), this is about 154.5KB of overhe

With 2.5M completely idle clients connected simulta amounts to 130MB/second in each direction networ amount of Tor directory traffic[11]. Of course, our 2 connected simultaneously, nor entirely idle, so we e lower than this.

Reducing or Disabling Padding vi

To allow mobile clients to either disable or reduce th CELL_PADDING_NEGOTIATE cell (tor-spec.txt section relays. This cell is used to instruct relays to cease se

If the client has opted to use reduced padding, it con from the range [9000,14000] milliseconds (subject to per Section 2.6), still using the Y=max(X,X) distributio unidirectional, the expected frequency of padding c distribution above as opposed to Z. For a range of 5 send a padding packet every 9000+3332.8 = 12332.8 timeout from ~50min down to ~25min, which cause closed shortly there after when it is idle, thus reduci

These two changes cause the padding overhead to g

connection down to 69KB per one-time-use Tor con
maximum overhead goes from 103 bytes/sec down

If a client opts to completely disable padding, it send
instruct the relay not to pad, and then does not sen

Currently, clients negotiate padding only when a cha
sending their NETINFO cell. Recipients SHOULD, how
messages at any time.

If a client which previously negotiated reduced, or d
enable default padding (ie padding according to the
send CELL_PADDING_NEGOTIATE START with zero ir
fields. (It therefore SHOULD NOT copy the values fro
the CELL_PADDING_NEGOTIATE cell.) This avoids the
padding negotiations if the consensus parameters s
clamping of the timing parameters will cause the re
consensus parameters.

Clients and bridges MUST reject padding negotiation
channel if they receive one.

Consensus Parameters Governir

Connection-level padding is controlled by the follow

```
    * nf_ito_low
      - The low end of the range to send padd
      - Default: 1500

    * nf_ito_high
      - The high end of the range to send pad
      - Default: 9500
      - If nf_ito_low == nf_ito_high == 0, pa

    * nf_ito_low_reduced
      - For reduced padding clients: the low
        when inactive, in ms.
      - Default: 9000

    * nf_ito_high_reduced
      - For reduced padding clients: the high
padding,
        in ms.
      - Default: 14000

    * nf_conntimeout_clients
      - The number of seconds to keep never-u
        available for clients to use. Note th
        randomized uniformly from this value
      - The number of seconds to keep idle (n
        channels are open and available. (We
        time duration of padding, which is th
      - This value is also used to determine
        used, we should attempt to keep build
        port. (See path-spec.txt section 2.1.
        originally added to work around imple
        serves as a reasonable default regard
      - For all use cases, reduced padding cl
        value.
      - Implementations MAY mark circuits hel
        quantity (half the consensus value) a
        to prevent their use from becoming a
      - Default: 1800

    * nf_pad_before_usage
      - If set to 1, OR connections are padde
        for any application traffic. If 0, OR
        until application data begins.
      - Default: 1

    * nf_pad_relays
      - If set to 1, we also pad inactive rel
      - Default: 0

    * nf_conntimeout_relays
      - The number of seconds that idle relay
        open.
      - Default: 3600
```

Circuit-level padding

The circuit padding system in Tor is an extension of
machine design[15]. At a high level, this design plac
machines at the client, and one or more padding st
circuit.

State transition and histogram generation has been
programmable, and probability distribution support
representations like APE[16]. Additionally, packet co
application conditions have been added.

At present, Tor uses this system to deploy two pairs
obscure differences between the setup phase of clie
the first 10 cells.

This specification covers only the resulting behavior
does not cover the state machine implementation d
using the circuit padding system to develop future p
developer documentation[17].

Circuit Padding Negotiation

Circuit padding machines are advertised as "Paddin
spec.txt Section 9). The onion service circuit padding
"Padding=2".

Because circuit padding machines only become acti
and because more than one padding machine may
lifetime, there is also a padding negotiation cell and
relay commands 41 and 42, with relay headers as p

The fields of the relay cell Data payload of a negotia

```
const CIRCPAD_COMMAND_STOP = 1;
const CIRCPAD_COMMAND_START = 2;

const CIRCPAD_RESPONSE_OK = 1;
const CIRCPAD_RESPONSE_ERR = 2;

const CIRCPAD_MACHINE_CIRC_SETUP = 1;

struct circpad_negotiate {
  u8 version IN [0];
  u8 command IN [CIRCPAD_COMMAND_START,

  u8 machine_type IN [CIRCPAD_MACHINE_CI

  u8 unused; // Formerly echo_request

  u32 machine_ctr;
};
```

When a client wants to start a circuit padding machi
destination hop advertises the appropriate subprot
sends a circpad_negotiate cell to that hop with comı
and machine_type=CIRCPAD_MACHINE_CIRC_SETUF
destination hop is the second hop in the circuit). The
machine instance this is on the circuit. It is used to c

When a relay receives a circpad_negotiate cell, it che
machine, and sends a circpad_negotiated cell, which
relay cell with command number 42 (see tor-spec.tx

```
struct circpad_negotiated {
  u8 version IN [0];
  u8 command IN [CIRCPAD_COMMAND_START,
  u8 response IN [CIRCPAD_RESPONSE_OK, C

  u8 machine_type IN [CIRCPAD_MACHINE_CI

  u32 machine_ctr;
};
```

If the machine is supported, the response field will c
is not, it will contain CIRCPAD_RESPONSE_ERR.

Either side may send a CIRCPAD_COMMAND_STOP t
(clients MUST only send circpad_negotiate, and relay
for this purpose).

If the machine_ctr does not match the current mach
command is ignored.

Circuit Padding Machine Messag

Clients MAY send padding cells towards the relay be
response, to allow for outbound cover traffic before

Clients MAY send another circpad_negotiate cell bef
response, to allow for rapid machine changes.

Relays MUST NOT send padding cells or circpad_neg
machine is active. Any padding-related cells that arr
relay sources are protocol violations, and clients MA
to avoid side channel risk.

Obfuscating client-side onion se

The circuit padding currently deployed in Tor attem
circuit setup. Service-side setup is not covered, beca
significantly more overhead, and/or require interact

The approach taken aims to make client-side introd
the cell direction sequence and cell count of 3 hop g
traffic, for the first 10 cells only. The lifespan of intro
match the lifespan of general circuits.

Note that inter-arrival timing is not obfuscated by th

Common general circuit construction seq

Most general Tor circuits used to surf the web or do
with the following 6-cell relay cell sequence (cells su
the others are incoming):

[EXTEND2] -> EXTENDED2 -> [EXTEND2] -> EXTENDE

When this is done, the client has established a 3-ho
the other end. Usually after this comes a series of D
establishes an SSL connection or fetches directory i

[DATA] -> [DATA] -> DATA -> DATA...(inbound cells c

The above stream of 10 relay cells defines the granc
come out of Tor browser during our testing, and it's
and rendezvous circuits blend in.

Please note that in this section we only investigate r
cells like CREATE/CREATED or AUTHENTICATE/etc. th
handshake. The rationale is that connection-level ce
and are not an effective fingerprint for a network/gu

Client-side onion service introduction circ

Two circuit padding machines work to hide client-sid
at the origin, and one machine at the second hop of
padding towards the other. The padding from the o
second hop and does not get forwarded to the actu

From Section 3.3.1 above, most general circuits have
sequence (outgoing cells marked in [brackets]):

```
[EXTEND2] -> EXTENDED2 -> [EXTEND2] -> EXTE
  -> [DATA] -> [DATA] -> DATA -> DATA...(ir

Whereas normal introduction circuits usuall

[EXTEND2] -> EXTENDED2 -> [EXTEND2] -> EXTE
  -> [INTRO1] -> INTRODUCE_ACK
```

This means that up to the sixth cell (first line of each
intro circuits have identical cell sequences. After tha
sequence of

-> [DATA] -> [DATA] -> DATA -> DATA...(inbound data

We achieve this by starting padding INTRODUCE1 ha
negotiation cells, in the common case of the second

-> [INTRO1] -> [PADDING_NEGOTIATE] -> PADDING_

Then, the middle node will send between INTRO_MA
INTRO_MACHINE_MAXIMUM_PADDING (10) cells, to
continue)" portion of the trace (aka the rest of an HT

We also set a special flag which keeps the circuit ope
performed. With this feature the circuit will stay aliv
web circuits before they expire (usually 10 minutes)

Client-side rendezvous circuit hiding

Following a similar argument as for intro circuits, we

circuits to blend in with the initial cell sequence of g
this:

```
[EXTEND2] -> EXTENDED2 -> [EXTEND2] -> EXTE
   -> [DATA] -> [DATA] -> DATA -> DATA...(

Whereas normal rendezvous circuits usually

[EXTEND2] -> EXTENDED2 -> [EXTEND2] -> EXTE
   -> REND2 -> [BEGIN]
```

This means that up to the sixth cell (the first line), bo
identical cell sequences.

After that we want to mimic a [DATA] -> [DATA] -> D

With padding negotiation right after the REND_ESTA

```
[EXTEND2] -> EXTENDED2 -> [EXTEND2] -> EXTE
   -> [PADDING_NEGOTIATE] -> [DROP] -> PADD

After which normal application DATA cells c
```

Hence this way we make rendezvous circuits look lik
the circuit setup.

After that our machine gets deactivated, and we let
the traffic flow. Since rendezvous circuits usually im
to surf the web), we can expect that they will look al

Circuit setup machine overhead

For the intro circuit case, we see that the origin-side
[PADDING_NEGOTIATE] cell, whereas the origin-side
PADDING_NEGOTIATED cell and between 7 to 10 DF
overhead of this machine is 11 padding cells per intr

For the rend circuit case, this machine is quite light.
total of 4 padding cells.

Circuit padding consensus parar

The circuit padding system has a handful of consens
circuit padding entirely, or rate limit the total overhe

```
* circpad_padding_disabled
  - If set to 1, no circuit padding machine
    current padding machines will cease pac
  - Default: 0

* circpad_padding_reduced
  - If set to 1, only circuit padding mach
    overhead" will be used. (Currently no s
    as "reduced overhead").
  - Default: 0

* circpad_global_allowed_cells
  - This is the number of padding cells tha
    the 'circpad_global_max_padding_percent
  - Default: 0

* circpad_global_max_padding_percent
  - This is the maximum ratio of padding ce
    as a percent. If the global ratio of pa
    across all circuits exceeds this percer
    until the ratio becomes lower. 0 means
  - Default: 0

* circpad_max_circ_queued_cells
  - This is the maximum number of cells tha
    before padding stops being sent on that
  - Default: CIRCWINDOW_START_MAX (1000)
```

Acknowledgments

This research was supported in part by NSF grants C
CNS-1526306, CNS-1619454, and CNS-1640548.

1. https://en.wikipedia.org/wiki/NetFlow
2. http://infodoc.alcatel-lucent.com/html/0_a
/7750_SR_OS_Router_Configuration_Guide/Cflowd
3. http://www.cisco.com/en/US/docs/ios/12_3t/
/nfl_a1gt_ps5207_TSD_Products_Command_Referen
4. http://www.cisco.com/c/en/us/support/docs/
switches/70974-netflow-catalyst6500.html#opco
5. https://www.juniper.net/techpubs/software/
vol1/html/ip-jflow-stats-config4.html#560916
6. http://www.jnpr.net/techpubs/en_US/junos15
/configuration-statement/flow-active-timeout-
7. http://www.jnpr.net/techpubs/en_US/junos15
/configuration-statement/flow-active-timeout-
8. http://www.h3c.com/portal/Technical_Suppor
/Technical_Documents/Switches/H3C_S9500_Serie
/H3C_S9500_CM-Release1648%5Bv1.24%5D-System_V
/624854_1285_0.htm#_Toc217704193
9. http://docs-legacy.fortinet.com/fgt/handbo
/FortiOS%205.2%20CLI/config_system.23.046.htm
10. http://wiki.mikrotik.com/wiki/Manual:IP/T
11. https://metrics.torproject.org/dirbytes.h
12. http://freehaven.net/anonbib/cache/murdoc
13. https://spec.torproject.org/proposals/188
14. http://www.ntop.org/wp-content/uploads/20
15. http://arxiv.org/pdf/1512.00524
16. https://www.cs.kau.se/pulls/hot/thebasket
17. https://github.com/torproject/tor/tree/ma
/CircuitPaddingDevelopment.md
18. https://www.usenix.org/node/190967
https://blog.torproject.org/technical-sum

Denial-of-service preven in Tor

This document covers the strategy, motivation, and mitigation systems designed into Tor.

The older `dos-spec` document is now the Memory

An in-depth description of the proof of work mecha proposal 327, is now in the Proof of Work for onion

Overview

As a public and anonymous network, Tor is open to
attempts. It's necessary to constantly develop a vari
types of attacks.

These mitigations are expected to improve network
important for limiting the avenues an attacker coulc
anonymity. For example, the ability to kill targeted T
traffic analysis. See the "Sniper Attack" paper by Jan
Scheuermann.

The attack and defense environment changes over t
attempt to describe the current state of things, but t

The defenses here are organized by the type of resc
physical resources (Memory, CPU, Bandwidth) or pr
Introductions).

In practice there are always overlaps between these
onion service, for example, puts some strain on eve

Physical resources

Memory

Memory exhaustion is both one of the most serious
subject of the most fully developed defense mechar
use and free the most disposable objects first when

CPU

The available CPU time on a router can be exhauste
not capable of processing network input at line rate
problematic in the single-threaded C implementatio
circuit extension handshakes are deferred to a threa
is still a precious resource.

We currently don't directly monitor and respond to
limits for protocol resources, like circuits extensions

are associated with this CPU load.

Bandwidth

Relay operators can place hard limits on total bandv `RelayBandwidth` options. These options can help re on their network, however they aren't designed as c mechanisms.

Beyond just shaving off harmful bandwidth peaks it disrupted too much, and especially not disrupted in this goal we rely on flow control and fair dequeuein;

Protocol resources

Channels

All channels to some extent are a limited resource, I floods of incoming TLS connections.

Excessive incoming TLS connections consume mem operating system resources. Excessive incoming cor denial of service attack.

The C Tor implementation establishes limits on botł connections per IP address and the rate of new con family of configuration options and their correspon(

Circuits

Excessive circuit creation can impact the entire path reject these attacks any time they can be identified. possible, before they have fully built the circuit.

Because of Tor's anonymity, most affected nodes ex from every direction. The guard position, however, I that are creating too many circuits.

The C Tor implementation limits the acceptable rate address using the `DoSCircuit` configuration option parameters.

Onion service introductions

Flooding an onion service with introduction attempt
addition to the CPU, memory, and bandwidth load e
and the service, all involved relays experience a circ

We have two types of onion service DoS mitigations
as needed by individual onion servce operators.

Mitigation by rate limiting

Introduction attempts can be rate-limited by each ir
the service.

This defense is configured by an operator using the
configuration options. Services use the introduction
settings to each introduction point.

Mitigation using proof of work

A short non-interactive computational puzzle can be
attempt. Requests provided by the client will be ent
puzzle solution's effort score. Requests are processe
which can be adjusted to a value within the server's

Based on the queue behavior, servers will continuou
suggestion. Queue backlogs cause the effort to rise,
effort to decay. If the queue is never overfull the eff
to include a proof-of-work solution at all.

We may support multiple cryptographic algorithms
currently we support one type. It's called `v1` in our
algorithm developed for this purpose. See the docu
service introduction.

This defense is configured by an operator using the
options. Additionally, it requires both the client and
the `pow` module (and `--enable-gpl` mode) availab
setting, proof of work *is* available through common
Debian.

Memory exhaustion

Memory exhaustion is a broad issue with many und
requires clients, onion services, relays, and authoriti
information in buffers and caches. But an attacker c
exhaust the memory of the a targeted Tor process,
that process.

With this in mind, any Tor implementation (especiall
service) must take steps to prevent memory-based

Detecting low memory

The easiest way to notice you're out of memory wou
when you try to allocate more. Unfortunately, some
give you an "out of memory" error when you're low
overcommit and promise you memory that they car
on, they might kill processes that actually try to use
given out.

So in practice, the mainline Tor implementation use
imposed "MaxMemInQueues" value as an upper bo
to allocate to certain kinds of queued usages. This v
derived from a fraction of the total amount of syste

As of Tor 0.4.7.x, the MaxMemInQueues mechanism
allocation:

- Cells queued on circuits.
- Per-connection read or write buffers.
- On-the-fly compression or decompression stat
- Half-open stream records.
- Cached onion service descriptors (hsdir only).
- Cached DNS resolves (relay only).
- GEOIP-based usage activity statistics.

Note that directory caches aren't counted, since tho
via mmap.

Responding to low memory

If our allocations exceed MaxMemInQueues, then w our memory allocation.

Freeing from caches: For each of our onion service de
our GEOIP statistics cache, we check whether they a
total allocation. If they do, we free memory from the
remaining is no more than 10% of our total allocatic

When freeing entries from a cache, we aim to free (a

Freeing from buffers: After freeing data from caches,
above 90% of MaxMemInQueues. If they are, we try
we are below 90% of MaxMemInQueues.

When deciding to what circuits to free, we sort them
in their queues, and free the ones with the oldest da
single cell has been queued for 5 minutes would be
have been queued for 5 seconds.) "Data queued on
could drop if the circuit were destroyed: not only the
also any bytes queued in buffers associated with str
to the circuit.

We free non-tunneled directory connections accord
age of their oldest queued data.

Upon freeing a circuit, a "DESTROY cell" must be ser

Reporting low memory

We define a "low threshold" equal to 3/4 of MaxMer
usage is above the low threshold, we record ourselv
pressure".

(This is not currently reported.)

Tor's extensions to the S

Overview

The SOCKS protocol provides a generic interface for
to a SOCKS server via TCP, and requests a TCP conn
The SOCKS server establishes the connection, and r
After the connection has been established, the clien
usual.

Tor supports SOCKS4 as defined in [1], SOCKS4A as
in [3] and [4].

The stickiest issue for Tor in supporting clients, in pr
occur at the OR side: if clients do their own DNS lool
addresses the client wants to reach. SOCKS4 suppor
SOCKS4A is a kludge on top of SOCKS4 to allow add
supports IPv4, IPv6, and hostnames.

Extent of support

Tor supports the SOCKS4, SOCKS4A, and SOCKS5 st

BOTH:

- The BIND command is not supported.

SOCKS4,4A:

- SOCKS4 usernames are used to implement str

```
    SOCKS5:
    - The (SOCKS5) "UDP ASSOCIATE" command is n
    - SOCKS5 BIND command is not supported.
    - IPv6 is not supported in CONNECT commands
    - SOCKS5 GSSAPI subnegotiation is not suppo
    - The "NO AUTHENTICATION REQUIRED" (SOCKS5)
      supported; and as of Tor 0.2.3.2-alpha, t
      authentication method [02] is supported t
      implement stream isolation. As an extens
 clients,
      we allow clients to pass "USERNAME/PASSWO
      even if no authentication was selected. F
      username/password fields of this message
      violates RFC1929 [4], but ensures interop
      SOCKS5 client implementations.
    - Custom reply error code. The "REP" fields
      unassigned values which are used to descr
      ExtendedErrors in the tor.1 man page for
      back if this SocksPort flag is set.
```

(For more information on stream isolation, see Isola

Name lookup

As an extension to SOCKS4A and SOCKS5, Tor imple
"RESOLVE" [F0]. When Tor receives a "RESOLVE" SO
lookup of the hostname provided as the target addr
either an error (if the address couldn't be resolved)
success, the address is stored in the portion of the S
IP address.

(We support RESOLVE in SOCKS4 too, even though i

For SOCKS5 only, we support reverse resolution witl
"RESOLVE_PTR" [F1]. In response to a "RESOLVE_PTF
address as its target, Tor attempts to find the canon
and returns it in the "server bound address" portion
supported before Tor 0.1.2.2-alpha.)

Other command extensions

Tor 0.1.2.4-alpha added a new command value: "CO
open an encrypted direct TCP connection to the dire
by address:port (the port specified should be the OF

tunnel and a "BEGIN_DIR" relay cell to accomplish th

The F2 command value was removed in Tor 0.2.0.10
flag in edge_connection_t.

HTTP-resistance

Tor checks the first byte of each SOCKS request to s
request (that is, it starts with a "G", "H", or "P"). If so,
the user that his/her browser is misconfigured. This
mistakenly try to use Tor as an HTTP proxy instead o

Optimistic data

Tor allows SOCKS clients to send connection data be
When using an exit node that supports "optimistic d
server without waiting to see whether the connectio
can save a single round-trip time when starting conn
client speaks first (like HTTP). Clients that do this mu
connection has succeeded or failed *after* they have s

Extended error codes

We define a set of additional extension error codes
implementation in response to failed onion service

(In the C Tor implementation, these error codes can
flag. In Arti, these error codes are enabled wheneve

- X'F0' Onion Service Descriptor Can Not be Fou

```
      The requested onion service descriptor
      and thus not reachable by the client.

    * X'F1' Onion Service Descriptor Is Inval

      The requested onion service descriptor
      validation failed.

    * X'F2' Onion Service Introduction Failed

      Client failed to introduce to the serv
      found but the service is not anymore at
      service has likely changed its descript

    * X'F3' Onion Service Rendezvous Failed

      Client failed to rendezvous with the se
client
      is unable to finalize the connection.

    * X'F4' Onion Service Missing Client Auth

      Tor was able to download the requested
      unable to decrypt its content because t
authorization
      information for it.

    * X'F5' Onion Service Wrong Client Author

      Tor was able to download the requested
      unable to decrypt its content using the
information
      it has. This means the client access we

    * X'F6' Onion Service Invalid Address

      The given .onion address is invalid. In
      error is returned: address checksum doe
      key is invalid or the encoding is inval

    * X'F7' Onion Service Introduction Timed

      Similar to X'F2' code but in this case,
      have failed due to a time out.
```

(Note that not all of the above error codes are curre
2023.)


```
References:
 [1] http://en.wikipedia.org/wiki/SOCKS#SOCKS
 [2] http://en.wikipedia.org/wiki/SOCKS#SOCKS
 [3] SOCKS5: RFC 1928 https://www.ietf.org/rf
 [4] RFC 1929: https://www.ietf.org/rfc/rfc19
```

Special Hostnames in To

Overview

Most of the time, Tor treats user-specified hostnam
connects to <www.torproject.org>, Tor picks an exit
to "www.torproject.org". Some hostnames, however
behavior and circuit-building rules.

These hostnames can be passed to Tor as the addre
request. If the application is connected to Tor using
TransPort, or NATDPort), these hostnames can be su
using the MapAddress configuration option or the M

.exit

```
SYNTAX:  [hostname].[name-or-digest].exit
         [name-or-digest].exit
```

Hostname is a valid hostname; [name-or-digest] is e
the hex-encoded digest of that node's public key.

When Tor sees an address in this format, it uses the
If no "hostname" component is given, Tor defaults t
exit node.

It is valid to try to resolve hostnames, and in fact up
mapaddress of the form "www.google.com.foo.exit=
subsequent lookups.

The .exit notation is disabled by default as of Tor 0.2
application-level attacks.

```
EXAMPLES:
   www.example.com.exampletornode.exit

      Connect to www.example.com from the r

   exampletornode.exit

      Connect to the published IP address c
      "exampletornode" as the exit.
```

.onion

```
SYNTAX:  [digest].onion
         [ignored].[digest].onion
```

Version 2 addresses (deprecated since 0.4.6.1-alpha
SHA1 hash of the identity key for a hidden service, e

Version 3 addresses, the digest is defined as:

```
     onion_address = base32(PUBKEY | CHECKSUM
     CHECKSUM = H(".onion checksum" | PUBKEY

     where:
       - PUBKEY is the 32 bytes ed25519 maste
       - VERSION is a one byte version field
       - ".onion checksum" is a constant stri
       - H is SHA3-256
       - CHECKSUM is truncated to two bytes b
onion_address
```

When Tor sees an address in this format, it tries to l
onion service. See rend-spec-v3.txt for full details.

The "ignored" portion of the address is intended for
Tor 0.2.4.10-alpha and later.

.noconnect

SYNTAX: [string].noconnect

When Tor sees an address in this format, it immedia
attaching it to any circuit. This is useful for controlle
application is indeed using the same instance of Tor

This feature was added in Tor 0.1.2.4-alpha, and tak
that it provided another avenue for detecting Tor us

Tor Rendezvous Specific

This document specifies how the hidden service ver
to be proposal 224-rend-spec-ng.txt.

This document describes a proposed design and sp
version 0.2.5.x or later. It's a replacement for the cu
clarity and for improved design.

Hidden services: overvie preliminaries

Hidden services aim to provide responder anonymit communication on the Tor network. Unlike regular T connection initiator receives anonymity but the resp attempt to provide bidirectional anonymity.

Participants:

Operator -- A person running a hidden service

```
Host, "Server" -- The Tor software run
   a hidden service.

User -- A person contacting a hidden se

Client -- The Tor software running on t

Hidden Service Directory (HSDir) -- A T
  statements from hidden service hosts
  contact with them.

Introduction Point -- A Tor node that a
  for hidden services and anonymously r
  hidden service.

Rendezvous Point -- A Tor node to which
  connect and which relays traffic betw
```

Improvements over previous ver

Here is a list of improvements of this proposal over

a) Better crypto (replaced SHA1/DH/RSA1024 with S Improved directory protocol leaking less to director protocol with smaller surface for targeted attacks. d against impersonation. e) More extensible introduct keys for onion services g) Advanced client authoriza

Notation and vocabulary

Unless specified otherwise, all multi-octet integers a

We write sequences of bytes in two ways:

```
     1. A sequence of two-digit hexadecimal v
        as in [AB AD 1D EA].

     2. A string of characters enclosed in qu
        characters in these strings are encod
        representations; strings are NOT nul-
        explicitly described as NUL terminate

 We use the words "byte" and "octet" interc

 We use the vertical bar | to denote concat
```

We use INT_N(val) to denote the network (big-endiar "val" in N bytes. For example, INT_4(1337) is [00 00 0 % (2 ^ (N * 8)). For example, INT_4(42) is 42 % 42949

Cryptographic building blocks

This specification uses the following cryptographic b

```
   * A pseudorandom number generator backe
     The output of the PRNG should always
     the network to avoid leaking raw PRNG
     (see [PRNG-REFS]).

   * A stream cipher STREAM(iv, k) where
     S_IV_LEN bytes and k is a key of leng

   * A public key signature system SIGN_KE
     SIGN_SIGN(seckey,msg)->sig; and SIGN_
     { "OK", "BAD" }; where secret keys ar
     bytes, public keys are of length SIGN
     signatures are of length SIGN_SIG_LEN

     This signature system must also suppo
     as discussed in appendix [KEYBLIND] a
     SIGN_BLIND_SECKEY(seckey, blind)->sec
     SIGN_BLIND_PUBKEY(pubkey, blind)->pub

   * A public key agreement system "PK", p
     PK_KEYGEN()->seckey, pubkey; PK_VALID
     and PK_HANDSHAKE(seckey, pubkey)->out
     of length PK_SECKEY_LEN bytes, public
     PK_PUBKEY_LEN bytes, and the handshak
     length PK_OUTPUT_LEN bytes.

   * A cryptographic hash function H(d), w
     collision resistant. It produces hash
     bytes.

   * A cryptographic message authenticatio
     produces outputs of length MAC_LEN by

   * A key derivation function KDF(message

As a first pass, I suggest:

   * Instantiate STREAM with AES256-CTR.

   * Instantiate SIGN with Ed25519 and the
     [KEYBLIND].

   * Instantiate PK with Curve25519.

   * Instantiate H with SHA3-256.

   * Instantiate KDF with SHAKE-256.

   * Instantiate MAC(key=k, message=m) wit
     where k_len is htonll(len(k)).
```

When we need a particular MAC key length below, w
bits).

For legacy purposes, we specify compatibility with o

point and rendezvous point protocols. These used R
as discussed in rend-spec.txt.

As in [proposal 220], all signatures are generated nc
those strings prefixed with a distinguishing value.

Protocol building blocks

In sections below, we need to transmit the locations
in the link identification format used by EXTEND2 ce

```
NSPEC      (Number of link specifier
NSPEC times:
  LSTYPE (Link specifier type)
  LSLEN  (Link specifier length)
  LSPEC  (Link specifier)
```

Link specifier types are as described in tor-spec.txt.
include at minimum specifiers of type [00] (TLS-over
and [03] (ed25519 identity key). Sets of link specifier
be rejected.

As of 0.4.1.1-alpha, Tor includes both IPv4 and IPv6
protocol link specifier lists. All available addresses S
regardless of the address that Tor actually used to c

We also incorporate Tor's circuit extension handsha
CREATED2 cells described in tor-spec.txt. In these ha
public key for a server sends a message and receive
the exchange is done, the two parties have a sharec
and the client knows that nobody else shares that k
secret key corresponding to the server's public key.

Assigned relay cell types

These relay cell types are reserved for use in the hic

32 -- RELAY_COMMAND_ESTABLISH_INTRO

```
        Sent from hidden service host to
        establishes introduction point. D
        [REG_INTRO_POINT].

   33 -- RELAY_COMMAND_ESTABLISH_RENDEZVOU

        Sent from client to rendezvous po
        point. Discussed in [EST_REND_POI

   34 -- RELAY_COMMAND_INTRODUCE1

        Sent from client to introduction
        introduction. Discussed in [SEND_

   35 -- RELAY_COMMAND_INTRODUCE2

        Sent from introduction point to h
        introduction. Same format as INTR
        [FMT_INTRO1] and [PROCESS_INTRO2]

   36 -- RELAY_COMMAND_RENDEZVOUS1

        Sent from hidden service host to
        attempts to join host's circuit t
        client's circuit. Discussed in [J

   37 -- RELAY_COMMAND_RENDEZVOUS2

        Sent from rendezvous point to cli
        reports join of host's circuit to
        client's circuit. Discussed in [J

   38 -- RELAY_COMMAND_INTRO_ESTABLISHED

        Sent from introduction point to h
        reports status of attempt to esta
        point. Discussed in [INTRO_ESTABL

   39 -- RELAY_COMMAND_RENDEZVOUS_ESTABLIS

        Sent from rendezvous point to cli
        receipt of ESTABLISH_RENDEZVOUS c
        [EST_REND_POINT]

   40 -- RELAY_COMMAND_INTRODUCE_ACK

        Sent from introduction point to c
        receipt of INTRODUCE1 cell and re
        Discussed in [INTRO_ACK]
```

Acknowledgments

This design includes ideas from many people, incluc

```
Christopher Baines,
Daniel J. Bernstein,
Matthew Finkel,
Ian Goldberg,
George Kadianakis,
Aniket Kate,
Tanja Lange,
Robert Ransom,
Roger Dingledine,
Aaron Johnson,
Tim Wilson-Brown ("teor"),
special (John Brooks),
s7r
```

It's based on Tor's original hidden service design by
and Paul Syverson, and on improvements to that de
including

```
Tobias Kamm,
Thomas Lauterbach,
Karsten Loesing,
Alessandro Preite Martinez,
Robert Ransom,
Ferdinand Rieger,
Christoph Weingarten,
Christian Wilms,
```

We wouldn't be able to do any of this work without
including

```
  Alex Biryukov,
  Lasse Øverlier,
  Ivan Pustogarov,
  Paul Syverson,
  Ralf-Philipp Weinmann,

See [ATTACK-REFS] for their papers.

Several of these ideas have come from conv

   Christian Grothoff,
   Brian Warner,
   Zooko Wilcox-O'Hearn,
```

And if this document makes any sense at all, it's tha

```
Matthew Finkel,
George Kadianakis,
Peter Palfrader,
Tim Wilson-Brown ("teor"),
```

[XXX Acknowledge the huge bunch of people workir huge bunch of people working on 8244.]

Please forgive me if I've missed you; please forgive r ideas here too.

Protocol overview

In this section, we outline the hidden service protoc
the name of simplicity; those are given more fully be
more detail.

View from 10,000 feet

A hidden service host prepares to offer a hidden ser
serve as its introduction points. It builds circuits to t
introduction requests to it using those circuits.

Once introduction points have been picked, the hos
"hidden service descriptors" (or just "descriptors" fo
HSDir nodes. These documents list the hidden servi
describe how to make contact with the hidden servi

When a client wants to connect to a hidden service,
to be its "rendezvous point" and builds a circuit to th
does not have an up-to-date descriptor for the servi
and requests such a descriptor.

The client then builds an anonymous circuit to one c
points listed in its descriptor, and gives the introduc
pass to the hidden service. This introduction reques
and the first part of a cryptographic handshake.

Upon receiving the introduction request, the hidder
circuit to the rendezvous point and completes the cr
rendezvous point connects the two circuits, and the
two parties a shared key and proves to the client tha
service.

Once the two circuits are joined, the client can send
RELAY_BEGIN cells open streams to an external proc
server; RELAY_DATA cells are used to communicate

In more detail: naming hidden se

A hidden service's name is its long term master iden
hostname by encoding the entire key in Base 32, inc

checksum, and then appending the string ".onion" a
domain name.

(This is a change from older versions of the hidden s
80-bit truncated SHA1 hash of a 1024 bit RSA key.)

The names in this format are distinct from earlier na
name might look like:

```
        unlikelynamefora.onion
        yyhws9optuwiwsns.onion

    And a new name following this specificatio

        l5satjgud6gucryazcyvyvhuxhr74u6ygigiu

    Please see section [ONIONADDRESS] for the
```

In more detail: Access control

Access control for a hidden service is imposed at mu
above. Furthermore, there is also the option to impo
access control using pre-shared secrets exchanged
service and its clients.

The first stage of access control happens when dow
in order to download a descriptor, clients must know
to sign it. (See the next section for more info on key

To learn the introduction points, clients must decryp
descriptor. To do so, clients must know the *unblinde*
makes the descriptor unusable by entities without t
know the onion address).

Also, if optional client authorization is enabled, hidd
superencrypted using each authorized user's identit
unauthorized entities cannot decrypt it.

In order to make the introduction point send a rend
client needs to use the per-introduction-point authe
service descriptor.

The final level of access control happens at the serv
or not respond to the client's request depending on
protocol is extensible at this point: at a minimum, th

demonstrate knowledge of the contents of the encr
descriptor. If optional client authorization is enablec
the client to prove knowledge of a pre-shared privat

In more detail: Distributing hidd

Periodically, hidden service descriptors become stor
single directory or small set of directories from becc
a hidden service.

For each period, the Tor directory authorities agree
random value. (See section 2.3 for a description of h
voting practice; generating the value is described in
[SHAREDRANDOM-REFS].) That value, combined wit
identity keys, determines each HSDir's position in th
that period.

Each hidden service's descriptors are placed into th
that was used to sign them. Note that hidden servic
services' public keys directly. Instead, we use a key-b
a new key-of-the-day for each hidden service. Any cl
public identity key can derive these blinded signing
impossible to derive the blinded signing key lacking

This is achieved using two nonces:

```
* A "credential", derived from the public
  N_hs_cred.

* A "subcredential", derived from the cre
  and information which various with the
  N_hs_subcred.
```

The body of each descriptor is also encrypted with a
key.

To avoid a "thundering herd" problem where every
descriptor at the start of each period, each descript
period that depends on its blinded signing key. The
until the new keys come online.

In more detail: Scaling to multip

This design is compatible with our current approach
Specifically, hidden service operators can use onion
between multiple nodes on the HSDir layer. Further
to load balance their hidden services on the introdu
further discussions on this topic and alternative des

```
1.6. In more detail: Backward compatibility w
      protocols
```

This design is incompatible with the clients, server, a
versions of the hidden service protocol as describec
it is designed to enable the use of older Tor nodes a
points.

In more detail: Keeping crypto k

In this design, a hidden service's secret identity key
generate blinded signing keys, which are used to sig

In order to operate a hidden service, the operator ca
blinded signing keys and descriptor signing keys (an
and [HS-DESC-ENC] below), and their corresponding
export those to the hidden service hosts.

As a result, in the scenario where the Hidden Servic
can only impersonate it for a limited period of time
were generated in advance).

It's important to not send the private part of the blir
since an attacker can derive from it the secret maste
signing key should only be used to create credential

(NOTE: although the protocol allows them, offline ke
alpha.)

In more detail: Encryption Keys A

To avoid replays of an introduction request by an in
host must never accept the same request twice. Ear
design used an authenticated timestamp here, but i
can create a problematic fingerprint. (See proposal :

In more detail: A menagerie of k

[In the text below, an "encryption keypair" is roughly
with" and a "signing keypair" is roughly "a keypair yo

Public/private keypairs defined in this document:

```
Master (hidden service) identity key --
  used as the identity for a hidden ser
  term and not used on its own to sign
  to generate blinded signing keys as d
  and [SUBCRED]. The public key is enco
  address according to [NAMING].
  KP_hs_id, KS_hs_id.


Blinded signing key -- A keypair derive
  used to sign descriptor signing keys.
  each service. Clients who know a 'cre
  service's public identity key and an
  the public blinded identity key for a
  as an index in the DHT-like structure
  (see [SUBCRED]).
  KP_hs_blind_id, KS_hs_blind_id.



Descriptor signing key -- A key used to
  descriptors.  This is signed by blind
  blinded signing keys and master ident
  of this key must be stored online by
  public part of this key is included i
  of HS descriptors (see [DESC-OUTER]).
  KP_hs_desc_sign, KS_hs_desc_sign.


Introduction point authentication key -
  keypair used to identify a hidden ser
  introduction point. The service makes
  introduction point; these are used to
  hidden service host makes when establ
  point, so that clients who know the p
  can get their introduction requests s
  service. No keypair is ever used with
  point. (previously called a "service
  KP_hs_ipt_sid, KS_hs_ipt_sid
  ("hidden service introduction point s


Introduction point encryption key -- A
  keypair used when establishing connec
  point. Plays a role analogous to Tor
  makes a fresh keypair for each introd
  KP_hss_ntor, KS_hss_ntor.
```

```
      Ephemeral descriptor encryption key --
        keypair made by the service, and usec
        of hidden service descriptors when cl
        use.
        KP_hss_desc_enc, KS_hss_desc_enc


   Nonces defined in this document:

      N_hs_desc_enc -- a nonce used to derive
        encryption layer of hidden service de
        sometimes also called a "descriptor c

   Public/private keypairs defined elsewhere:

      Onion key -- Short-term encryption keyp

      (Node) identity key (KP_relayid).

   Symmetric key-like things defined elsewher

      KH from circuit handshake -- An unpred
      part of the Tor circuit extension hands
      to a particular circuit.
```

In even more detail: Client authorization

When client authorization is enabled, each authoriz
more asymmetric keypairs which are shared with th
those keys is not able to use the hidden service. Thr
that these pre-shared keys are exchanged between
secure out-of-band fashion.

Specifically, each authorized client possesses:

```
   - An x25519 keypair used to compute decryp
to
     decrypt the hidden service descriptor. S
     the client's counterpart to KP_hss_desc_
     KP_hsc_desc_enc, KS_hsd_desc_enc.

   - An ed25519 keypair which allows the clie
     prove to the hidden service that the cl
     signatures are inserted into the INTRODU
     introduction to the hidden service canno
AUTH].
     KP_hsc_intro_auth, KS_hsc_intro_auth.
```

The right way to exchange these keys is to have the
corresponding public keys to the hidden service out

way of doing this exchange would be to have the hid
and pass the corresponding private keys to its client
for more details on how these keys should be mana

[TODO: Also specify stealth client authorization.]

(NOTE: client authorization is implemented as of 0.3

Generating and publishi descriptors [HSDIR]

Hidden service descriptors follow the same metafor
They are published anonymously to Tor servers with
version and tor version >= 0.3.0.8 (because a bug wa

Deriving blinded keys ar

In each time period (see [TIME-PERIODS] for a defini
host uses a different blinded private key to sign its c
a different blinded public key as the index for fetchi

For a candidate for a key derivation method, see Ap

Additionally, clients and hosts derive a subcredentia
subcredential is needed to decrypt hidden service d
authenticate with the hidden service host in the intr
credential, it changes each period. Knowing the sub
the blinded private key, does not enable the hidden
credential--therefore, it is safe to put the subcreden
leaving the hidden service's private key offline.

The subcredential for a period is derived as:

N_hs_subcred = H("subcredential" | N_hs_cred | blin

In the above formula, credential corresponds to:

N_hs_cred = H("credential" | public-identity-key)

where public-identity-key is the public identity mast

```
 2.2. Locating, uploading, and downloading hic
        [HASHRING]
```

To avoid attacks where a hidden service's descriptor
store them at different directories over time, and us
those directories from being predictable far in adva

Which Tor servers hosts a hidden service depends c

```
          * the current time period,
          * the daily subcredential,
          * the hidden service directories' pu
          * a shared random value that changes
            shared_random_value.
          * a set of network-wide networkstatu
            (Consensus parameters are integer
            and published in the consensus doc
            dir-spec.txt, section 3.3.)

    Below we explain in more detail.
```

Dividing time into periods

To prevent a single set of hidden service directory fr
looking to permanently censor a hidden service, hid
to different locations that change over time.

The length of a "time period" is controlled by the co
and is a number of minutes between 30 and 14400
length is 1440 (one day).

Time periods start at the Unix epoch (Jan 1, 1970), a
number of minutes since the epoch and dividing by
our time periods to start at a regular offset from the
a "rotation time offset" of 12 voting periods from th
before dividing by the time period (effectively makir
12:00UTC when the voting period is 1 hour.)

Example: If the current time is 2016-04-13 11:15:01
epoch 1460546101, and the number of minutes sinc
subtract the "rotation time offset" of 12*60 minutes
get 24341715. If the current time period length is 14
see that we are currently in time period number 169

Specifically, time period #16903 began 16903*1440*6(
at 2016-04-12 12:00 UTC, and ended at 16904*1440*6
at 2016-04-13 12:00 UTC.

When to publish a hidden servic

Hidden services periodically publish their descriptor
responsible HSDirs is determined as specified in [W

Specifically, every time a hidden service publishes it:
a random time between 60 minutes and 120 minute
triggers, the hidden service needs to publish its des
HSDirs for that time period. [TODO: Control republis
parameter?]

Overlapping descriptors

Hidden services need to upload multiple descriptors
clients with older or newer consensuses than them.

descriptors to the HSDirs *before* the beginning of ea
are readily available for clients to fetch them. Furthe
uploading their old descriptor even after the end of
reachable by clients that still have consensuses fron

Hence, services maintain two active descriptors at e
don't have a notion of overlapping descriptors, and
descriptor for the current time period and shared ra
to ensure that descriptors will be available for all cli
[FETCHUPLOADDESC] for how this is achieved.

[TODO: What to do when we run multiple hidden se

Where to publish a hidden servic

This section specifies how the HSDir hash ring is for
time value is needed (e.g. to get the current time pe
and services use the valid-after time from their lates

The following consensus parameters control where

```
        hsdir_n_replicas = an integer in rang
        hsdir_spread_fetch = an integer in ra
 3.
        hsdir_spread_store = an integer in ra
 4.
           (Until 0.3.2.8-rc, the default was
```

To determine where a given hidden service descript
after the blinded public key for that period is derive
calculates:

```
        for replicanum in 1...hsdir_n_replica
            hs_service_index(replicanum) = H(
                                    blinded_
                                    INT_8(re
                                    INT_8(pe
                                    INT_8(pe
```

where blinded_public_key is specified in section [KE
of the time period in minutes, and period_num is ca
"valid-after" as specified in section [TIME-PERIODS].

Then, for each node listed in the current consensus
directory index for that node as:

```
hs_relay_index(node) = H("node-id
                        shared_rand
                        INT_8(perioc
                        INT_8(perioc
```

where shared_random_value is the shared value ge
[PUB-SHAREDRANDOM], and node_identity is the ed

Finally, for replicanum in 1...hsdir_n_replicas, the hid
to the first hsdir_spread_store nodes whose indices
hs_service_index(replicanum). If any of those nodes
lower-numbered replica of the service, any nodes al
skipped over) when choosing a replica's hsdir_sprea

When choosing an HSDir to download from, clients
first hsdir_spread_fetch nodes after the indices. (No
better tolerate disappearing HSDirs, hsdir_spread_f
hsdir_spread_store.) Again, nodes from lower-numb
choosing the spread for a replica.

Using time periods and SRVs to f descriptors

Hidden services and clients need to make correct us
random values (SRVs) to successfully fetch and uplo
problems with skewed clocks, both clients and servi
consensus as a way to take decisions with regards t
By using the consensus times as the ground truth h
desynchronization of clients and services due to sys
decisions are taken in this section, assume that they
times.

As [PUB-SHAREDRANDOM] specifies, consensuses c
(the current one and the previous one). Hidden serv
these shared random values with descriptor time pe
fetching/uploading descriptors. This section attemp

Let's start with an illustration of the system:

```
+------------------------------------------
|
| 00:00      12:00       00:00       12
| SRV#1      TP#1        SRV#2       TP
|
|  $==========|-----------$===========|
|
|
+------------------------------------------
```

```
                                Legend:
#1]
moment]
```

Client behavior for fetching descriptors

And here is how clients use TPs and SRVs to fetch de

Clients always aim to synchronize their TP with SRV, SRV#N: To achieve this wrt time periods, clients alwa fetching descriptors. Now wrt SRVs, if a client is in th period and a new SRV (i.e. the segments drawn with client is in a time segment between a new SRV and a drawn with "="), it uses the previous SRV.

Example:

```
+------------------------------------------
|
| 00:00      12:00       00:00       12:00
| SRV#1      TP#1        SRV#2       TP#2
|
|  $==========|-----------$===========|-----
|             ^           ^
|             C1          C2
+------------------------------------------
```

If a client (C1) is at 13:00 right after TP#1, then it will descriptors. Also, if a client (C2) is at 01:00 right afte SRV#1.

Service behavior for uploading descriptor

As discussed above, services maintain two active de the "first" and "second" service descriptors. Services

they receive a consensus with a valid_after time pas
rotate their descriptors by discarding their first desc
to the first, and rebuilding their second descriptor w

Services like clients also employ a different logic for
their position in the graph above. Here is the logic:

First descriptor upload logic

Here is the service logic for uploading its first descri

When a service is in the time segment between a ne
segments drawn with "-"), it uses the previous time
uploading its first descriptor: that's meant to cover f
is still in the previous time period.

Example: Consider in the above illustration that the
will upload its first descriptor using TP#0 and SRV#0
consensus it will be able to access it based on the cl

Now if a service is in the time segment between a n
segments drawn with "=") it uses the current time p
descriptor: that's meant to cover clients with an up-
period as the service.

Example:

```
+-----------------------------------------------
|
| 00:00      12:00       00:00       12:00
| SRV#1      TP#1        SRV#2       TP#2
|
|  $==========|-----------$===========|------
|                          ^
|                          S
+-----------------------------------------------
```

Consider that the service is at 01:00 right after SRV#
using TP#1 and SRV#1.

Second descriptor upload logic

Here is the service logic for uploading its second des

When a service is in the time segment between a ne
segments drawn with "-"), it uses the current time p
its second descriptor: that's meant to cover for clien

on the same TP as the service.

Example: Consider in the above illustration that the
will upload its second descriptor using TP#1 and SR

Now if a service is in the time segment between a n
segments drawn with "=") it uses the next time peri
descriptor: that's meant to cover clients with a newe
next time period).

Example:

```
+-----------------------------------------------
|
| 00:00      12:00      00:00      12:00
| SRV#1      TP#1       SRV#2      TP#2
|
|  $==========|-----------$===========|-----
|                          ^
|                          S
+-----------------------------------------------
```

Consider that the service is at 01:00 right after SRV#
using TP#2 and SRV#2.

Directory behavior for handling descripto

Upon receiving a hidden service descriptor publish r
following:

```
      * The outer wrapper of the descriptor ca
        [DESC-OUTER]
      * The version-number of the descriptor i
      * If the directory has already cached a
service,
        the revision-counter of the uploaded d
the
        revision-counter of the cached one
      * The descriptor signature is valid
```

If any of these basic validity checks fails, the directo

NOTE: Even if the descriptor passes the checks abov
be invalid: directories cannot validate the encrypted
not have access to the public key of the service (req
encryption), or the necessary client credentials (for c

Expiring hidden service descript

Hidden services set their descriptor's "descriptor-life
Hidden services ensure that their descriptor will ren
republishing their descriptors periodically as specifi

Hidden services MUST also keep their introduction
including those intro points are valid (even if that's a

URLs for anonymous uploading a

Hidden service descriptors conforming to this specif
POST request to the URL `/tor/hs/<version>/publi`
directory's root, and downloaded with an HTTP GET
`/<version>/<z>` where `<z>` is a base64 encoding o
key and `<version>` is the protocol version which is

These requests must be made anonymously, on circ

Client-side validation of onion ac

When a Tor client receives a prop224 onion address
the onion address before attempting to connect or
fails, the client MUST refuse to connect.

As part of the address validation, Tor clients should
key does not have a torsion component. If Tor accep
components, attackers could create multiple equiva
ed25519 key, which would map to the same service.
could lead to phishing attacks and surprising behavi
that blocks onion addresses, but could be bypassed
with a torsion component).

The right way for clients to detect such fraudulent a
malevolently and never naturally) is to extract the ec
address and multiply it by the ed25519 group order
ed25519 identity element. For more details, please s

Publishing shared rando

Our design for limiting the predictability of HSDir up
random value (SRV) that isn't predictable in advance
The authorities must run a protocol to generate suc
period. Here we describe how they publish these va
generate them can change independently of the res
information see [SHAREDRANDOM-REFS].

According to proposal 250, we add two new lines in

```
"shared-rand-previous-value" SP NUM_REVE
"shared-rand-current-value" SP NUM_REVEA
```

Client behavior in the absence o

If the previous or current shared random value cann
clients and services need to generate their own ranc
HSDirs.

To do so, Tor clients and services use:

SRV = H("shared-random-disaster" | INT_8(period_le

where period_length is the length of a time period ir
period_num is calculated as specified in [TIME-PERIC
value that could not be found originally.

Hidden services and changing sh

It's theoretically possible that the consensus shared
disappear in the middle of a time period because of
or misbehaving.

To avoid client reachability issues in this rare event,
shared random values to find the new responsible H
there.

XXX How long should they upload descriptors there

Hidden service descripto wrapper [DESC-OUTER]

The format for a hidden service descriptor is as follo spec.txt.

"hs-descriptor" SP version-number NL

[At start, exactly once.]

```
    The version-number is a 32 bit unsigne
    of the descriptor. Current version is

  "descriptor-lifetime" SP LifetimeMinutes

    [Exactly once]

    The lifetime of a descriptor in minute
    hidden service descriptor at least Lif
    uploaded.

    The LifetimeMinutes field can take val
    hours).


 "descriptor-signing-key-cert" NL certific

    [Exactly once.]

    The 'certificate' field contains a cer
    proposal 220, wrapped with "-----BEGIN
    certificate cross-certifies the short-
    the blinded public key.  The certifica
    blinded public key must be present as


  "revision-counter" SP Integer NL

    [Exactly once.]

    The revision number of the descriptor.
    second descriptor for a key that it al
    it should retain and serve the descrip
    revision-counter.

    (Checking for monotonically increasing
    prevents an attacker from replacing a
    a given key with a copy of an older ve

    Implementations MUST be able to parse
    counters.
```

```
      "superencrypted" NL encrypted-string

        [Exactly once.]

        An encrypted blob, whose format is dis
The
        blob is base64 encoded and enclosed ir
        ----END MESSAGE---- wrappers. (The res
with
        a newline character.)


      "signature" SP signature NL

        [exactly once, at end.]

        A signature of all previous fields, us
        descriptor-signing-key-cert line, pref
        service descriptor sig v3". We use a s
        the hidden service host does not need
        online.
```

HSDirs accept hidden service descriptors of up to 50
should also be introduced to control this value).

Hidden service descripto format

Hidden service descriptors are protected by two lay decrypt both layers to connect to the hidden service

The first layer of encryption provides confidentiality public key of the hidden service (e.g. HSDirs), while t useful when client authorization is enabled and pro possess valid client credentials.

First layer of encryption

The first layer of HS descriptor encryption is designe against entities who don't know the public identity k

First layer encryption logic

The encryption keys and format for the first layer of specified in [HS-DESC-ENCRYPTION-KEYS] with custo

```
SECRET_DATA = blinded-public-key
STRING_CONSTANT = "hsdir-superencrypted-
```

The encryption scheme in [HS-DESC-ENCRYPTION-K is derived from the public identity key (see [SUBCRE know the public identity key can decrypt the first de

The ciphertext is placed on the "superencrypted" fie

Before encryption the plaintext is padded with NUL bytes.

First layer plaintext format

After clients decrypt the first layer of encryption, the the second layer ciphertext which is contained in th

If client auth is enabled, the hidden service generate (`N_hs_desc_enc` , 32 random bytes) and encrypts it u

x25519 key. Authorized clients can use the descripto
the second (inner) layer of encryption. Our encryptio
service to also generate an ephemeral x25519 keypa

If client auth is disabled, fake data is placed in each
whether client authorization is enabled.

Here are all the supported fields:

"desc-auth-type" SP type NL

[Exactly once]

```
        This field contains the type of author
        descriptor. The only recognized type is
        encryption scheme described in this sec

        If client authorization is disabled, th

      "desc-auth-ephemeral-key" SP KP_hs_desc_

        [Exactly once]

        This field contains `KP_hss_desc_enc`,
        key generated by the hidden service and
        is used by the encryption scheme below.

        If client authorization is disabled, th
        x25519 pubkey that will remain unused.

      "auth-client" SP client-id SP iv SP encr

        [At least once]

        When client authorization is enabled, t
        "auth-client" line for each of its auth
        authorization is disabled, the fields h
        data of the right size (that's 8 bytes
'iv'
        and 16 bytes for 'encrypted-cookie' all

        When client authorization is enabled, e
        contains the descriptor cookie `N_hs_de
        individual client. We assume that each
        a pre-shared x25519 keypair (`KP_hsc_de
        decrypt the descriptor cookie.

        We now describe the descriptor cookie e
        the hidden service computes:

            SECRET_SEED = x25519(KS_hs_desc_eph
            KEYS = KDF(N_hs_subcred | SECRET_SE
            CLIENT-ID = fist 8 bytes of KEYS
            COOKIE-KEY = last 32 bytes of KEYS

        Here is a description of the fields in

        - The "client-id" field is CLIENT-ID fr

        - The "iv" field is 16 random bytes enc

        - The "encrypted-cookie" field contains
ciphertext
          as follows and is encoded in base64:
             encrypted-cookie = STREAM(iv, COOK

        See section [FIRST-LAYER-CLIENT-BEHAVIO
        how to decrypt the descriptor cookie.
```

```
    "encrypted" NL encrypted-string

     [Exactly once]

      An encrypted blob containing the seconc
is
      discussed in [HS-DESC-SECOND-LAYER] bel
      and enclosed in -----BEGIN MESSAGE----
wrappers.

    Compatibility note: The C Tor implementat
    newline when generating this first-layer-
    implementations MUST accept this section
    newline.  Other implementations MAY gener
    newline themselves, to avoid being distir
```

Client behavior

```
    The goal of clients at this stage is to c
    described in [HS-DESC-SECOND-LAYER].

    If client authorization is enabled, autho
the
    descriptor cookie to proceed with decrypt
    follows:

    An authorized client parsing the first la
    extracts the ephemeral key from "desc-aut
    CLIENT-ID and COOKIE-KEY as described in
    x25519 private key. The client then uses
    "auth-client" field which contains the c
    cookie. The client then uses COOKIE-KEY a
    descriptor_cookie, which is used to decry
descriptor
    encryption as described in [HS-DESC-SECON
```

Hiding client authorization data

```
    Hidden services should avoid leaking whet
    enabled or how many authorized clients th

    Hence even when client authorization is c
    fake "desc-auth-type", "desc-auth-ephemer
to
    the descriptor, as described in [HS-DESC-

    The hidden service also avoids leaking th
by
    adding fake "auth-client" entries to its
    descriptors always contain a number of au
    multiple of 16 by adding fake "auth-clier
    [XXX consider randomization of the value

    Clients MUST accept descriptors with any
    long as the total descriptor size is with
    controlled with a consensus parameter).
```

Second layer of encryption

The second layer of descriptor encryption is designe against unauthorized clients. If client authorization i descriptor_cookie, and contains needed informatior like the list of its introduction points.

If client authorization is disabled, then the second la any additional security, but is still used.

Second layer encryption keys

The encryption keys and format for the second laye specified in [HS-DESC-ENCRYPTION-KEYS] with custo

```
     SECRET_DATA = blinded-public-key | descr
     STRING_CONSTANT = "hsdir-encrypted-data'

   If client authorization is disabled the 'c
blank.

   The ciphertext is placed on the "encryptec
```

Second layer plaintext format

After decrypting the second layer ciphertext, clients
etc. The plaintext has the following format:

```
"create2-formats" SP formats NL

\[Exactly once\]

      A space-separated list of integers deno
      (handshake types) that the server recog
      ntor as described in tor-spec.txt. See
      of recognized handshake types.


     "intro-auth-required" SP types NL

      [At most once]

      A space-separated list of introduction-
      section [INTRO-AUTH] for more info. A c
      least one of these authentication types
      host. Recognized types are: 'ed25519'.


     "single-onion-service"

      [At most once]

      If present, this line indicates that th
      Service (see prop260 for more details a
      field has been introduced in 0.3.0 mean
      this.
```

```
      "pow-params" SP type SP seed-b64 SP sugg
                   SP expiration-time NL

       [At most once per "type"]

       If present, this line provides paramete
       client puzzle. A client that supports a
       corresponding solution in its introduct
       in the service's processing queue.

       Only version 1 is currently defined.
       Other versions may have a different for
       Introduced in tor-0.4.8.1-alpha.

         type: The type of PoW system used. We
"v1".

         seed-b64: A random seed that should b
                   hash function. Should be 32
                   without trailing padding.

         suggested-effort: An unsigned integer
                   clients should aim for when
                   zero to mean that PoW is av
                   suggested for a first conne

         expiration-time: A timestamp in "YYYY
time
                          with no space) after
and
                          is no longer valid a
```

Followed by zero or more introduction points as foll
[NUM_INTRO_POINT] below for accepted values):

```
"introduction-point" SP link-specifie

  [Exactly once per introduction poir
    point section]

  The link-specifiers is a base64 enc
  block in the format described in [E

  As of 0.4.1.1-alpha, services inclu
  specifiers in descriptors. All ava
  included in the descriptor, regardl
  onion service actually used to conr
  point.

  The client SHOULD NOT reject any LS
  recognize; instead, it should use t
  request to the introduction point.

  The client SHOULD perform the basic
  specifiers in the descriptor, descr
  section 5.1.2. These checks SHOULD
  detailed information about the clie
  or consensus. (See 3.3 for service

  When connecting to the introductior
  this list of link specifiers verbat
  here.

  The client MAY reject the list of l
  inconsistent with relay informatior
  NOT modify it.


"onion-key" SP "ntor" SP key NL

  [Exactly once per introduction poir

  The key is a base64 encoded curve25
onion
  key of the introduction point Tor r
  when a client extends to it.


"onion-key" SP KeyType SP key.. NL

  [Any number of times]

  Implementations should accept other
  syntax (where "KeyType" is some str
  unrecognized key types should be ig
```

```
"auth-key" NL certificate NL

  [Exactly once per introduction poir

  The certificate is a proposal 220 c
  "-----BEGIN ED25519 CERT-----".  It
  point authentication key (`KP_hs_ip
  the descriptor signing key (`KP_hs_
  certificate type must be [09], and
  is mandatory.

  NOTE: This certificate was original
  constructed the other way around: t
  are meant to be reversed.  However,
  backwards, and other implementation
  in order to conform.  (Since this s
  descriptor, which is _already_ sign
  the verification aspect of this cer
  its current form.)


"enc-key" SP "ntor" SP key NL

  [Exactly once per introduction poir

  The key is a base64 encoded curve25
  the introduction request to service


"enc-key" SP KeyType SP key.. NL

  [Any number of times]

  Implementations should accept other
  syntax (where "KeyType" is some str
  unrecognized key types should be ig
```

```
"enc-key-cert" NL certificate NL

  [Exactly once per introduction poir

  Cross-certification of the encrypti
  signing key.

  For "ntor" keys, certificate is a p
  wrapped in "-----BEGIN ED25519 CERT
  key is the the ed25519 equivalent c
  encryption key (`KP_hss_ntor`), wit
  derived using the process in propos
  signing key is the descriptor signi
  The certificate type must be [0B],
  extension is mandatory.

  NOTE: As with "auth-key", this cert
  constructed the other way around.
  with C tor, implementations need to
  serves even less point than "auth-k
  encryption key `KP_hss_ntor` is alr
  the `enc-key` entry.

"legacy-key" NL key NL

  [None or at most once per introduct
  [This field is obsolete and should
   is included for historical reasons

  The key is an ASN.1 encoded RSA pub
  legacy introduction point as descri

  This field is only present if the i
  legacy protocol (v2) that is <= 0.2
  "HSIntro 3".

"legacy-key-cert" NL certificate NL

  [None or at most once per introduct
  [This field is obsolete and should
   is included for historical reasons

  MUST be present if "legacy-key" is

  The certificate is a proposal 220 F
  in "-----BEGIN CROSSCERT-----" armo
  public key found in "legacy-key" us
```

To remain compatible with future revisions to the d
unrecognized lines in the descriptor. Other encrypti
are allowed; clients should ignore ones they do not

Clients who manage to extract the introduction poir
with the introduction protocol as specified in [INTRO

Compatibility note: At least some versions of OnionI
when generating this inner plaintext section; other i
section even if it is missing its final newline.

Deriving hidden service descript

In this section we present the generic encryption fo
We use the same encryption format in both encrypt
customization parameters SECRET_DATA and STRIN
layers.

The SECRET_DATA parameter specifies the secret da
key generation, while STRING_CONSTANT is merely
of the KDF.

Here is the key generation logic:

```
       SALT = 16 bytes from H(random), change
              descriptor even if the content
changed.
              (So that we don't leak whether
changed)

       secret_input = SECRET_DATA | N_hs_subc

       keys = KDF(secret_input | salt | STRIN
+ MAC_KEY_LEN)

       SECRET_KEY = first S_KEY_LEN bytes of
       SECRET_IV  = next S_IV_LEN bytes of ke
       MAC_KEY    = last MAC_KEY_LEN bytes of

   The encrypted data has the format:

       SALT       hashed random bytes from ab
       ENCRYPTED  The ciphertext
       MAC        D_MAC of both above fields

   The final encryption format is ENCRYPTED =
XOR Plaintext .

   Where D_MAC = H(mac_key_len | MAC_KEY | sa
   and
    mac_key_len = htonll(len(MAC_KEY))
   and
    salt_len = htonll(len(SALT)).
```

Number of introduction points

This section defines how many introduction points a
at minimum, by default and the maximum:

Minimum: 0 - Default: 3 - Maximum: 20

A value of 0 would means that the service is still aliv
any client at the moment. Note that the descriptor s
introduction points are added.

The reason for a maximum value of 20 is to give enc
OnionBalance to be able to load balance up to 120 s
order for the descriptor size to not overwhelmed hi
defined values that could be gigantic.

The introduction protoc

The introduction protocol proceeds in three steps.

First, a hidden service host builds an anonymous cir
circuit as an introduction point.

Single Onion Services attempt to build a non-anonyı
anonymous 3-hop circuit if:

```
* the intro point is on an address that
  a direct connection, or
* the initial attempt to connect to the
  circuit fails, and they are retrying t

   [After 'First' and before 'Second', t
   introduction points and associated ke
   them as described in section [HSDIR]
```

Second, a client builds an anonymous circuit to the i
introduction request.

Third, the introduction point relays the introduction
to the hidden service host, and acknowledges the in

Registering an introduction poin

Extensible ESTABLISH_INTRO protocol

When a hidden service is establishing a new introdu
ESTABLISH_INTRO cell with the following contents:

```
AUTH_KEY_TYPE    [1 byte]
AUTH_KEY_LEN     [2 bytes]
AUTH_KEY         [AUTH_KEY_LEN bytes]
N_EXTENSIONS     [1 byte]
N_EXTENSIONS times:
   EXT_FIELD_TYPE [1 byte]
   EXT_FIELD_LEN  [1 byte]
   EXT_FIELD      [EXT_FIELD_LEN bytes]
HANDSHAKE_AUTH   [MAC_LEN bytes]
SIG_LEN          [2 bytes]
SIG              [SIG_LEN bytes]
```

The AUTH_KEY_TYPE field indicates the type of the ir and the type of the MAC to use in HANDSHAKE_AUT

```
[00, 01] -- Reserved for legacy introc
            [LEGACY_EST_INTRO below]
[02] -- Ed25519; SHA3-256.
```

The AUTH_KEY_LEN field determines the length of tl
field contains the public introduction point authenti

The EXT_FIELD_TYPE, EXT_FIELD_LEN, EXT_FIELD ent
introduction protocol. Extensions with unrecognizec
ignored. (`EXT_FIELD_LEN` may be zero, in which cas

```
Unless otherwise specified in the document
   * Each extension type SHOULD be sent or
   * Parties MUST ignore any occurrences a
     with a given type after the first suc
   * Extensions SHOULD be sent in numerica
(The above extension sorting and multiplic
they may be overridden in the descriptions
```

The following extensions are currently defined:

`EXT_FIELD_TYPE`	
`[01]`	D

The HANDSHAKE_AUTH field contains the MAC of al
key the shared per-circuit material ("KH") generated
see tor-spec.txt section 5.2, "Setting circuit keys". It
cells.

SIG_LEN is the length of the signature.

SIG is a signature, using AUTH_KEY, of all contents o
SIG_LEN and SIG. These contents are prefixed with t

Upon receiving an ESTABLISH_INTRO cell, a Tor nod
signature, and checks the signature. The node must
destroy the circuit in these cases:

```
* If the key type is unrecognized
* If the key is ill-formatted
* If the signature is incorrect
* If the HANDSHAKE_AUTH value is inco

* If the circuit is already a rendezv
* If the circuit is already an introd
  [TODO: some scalability designs fa
* If the key is already in use by ano
```

Otherwise, the node must associate the key with the
cells.

Denial-of-Service defense extension (DOS_PARAM

The `DOS_PARAMS` extension in ESTABLISH_INTRO is u
parameters to the introduction point in order for it t
circuit.

This is for the rate limiting DoS mitigation specificall

The `EXT_FIELD_TYPE` value for the `DOS_PARAMS` exte

The content is defined as follows:

Field	Size	
`N_PARAMS`	1	Nur
`N_PARAMS` times:		
- PARAM_TYPE	1	Ide
- PARAM_VALUE	8	Inte

Recognized values for `PARAM_TYPE` in this extension

`PARAM_TYPE`	Name
[01]	`DOS_INTRODUCE2_RATE_PE`
[02]	`DOS_INTRODUCE2_BURST_P`

Together, these parameters configure a token buck
INTRODUCE2 messages the introduction point may

The `DOS_INTRODUCE2_RATE_PER_SEC` paramete
of messages; The `DOS_INTRODUCE2_BURST_PER_S
allowable burst of messages (that is, the size of the

Technically speaking, the `BURST` parameter is mis

> "per second": only a *rate* has an associated time.

If either of these parameters is set to 0, the defense
point should ignore the other parameter.

If the burst is lower than the rate, the introduction p

> Using this extension extends the payload of the E
> bringing it from 134 bytes to 155 bytes.

When this extension is not *sent*, introduction points
from the consensus parameters HiddenServiceEnab
HiddenServiceEnableIntroDoSRatePerSec, and Hidd

This extension can only be used with relays support

Introduced in tor-0.4.2.1-alpha.

Registering an introduction poin

> This section is obsolete and refers to a workarou
> versions. It is included for historical reasons.

Tor nodes should also support an older version of tl
documented in rend-spec.txt. New hidden service h
establishing introduction points at older Tor nodes t
in [EST_INTRO].

In this older protocol, an ESTABLISH_INTRO cell cont

```
        KEY_LEN        [2 bytes]
        KEY            [KEY_LEN bytes]
        HANDSHAKE_AUTH [20 bytes]
        SIG            [variable, up to end c

   The KEY_LEN variable determines the lengtl
```

The KEY field is the ASN1-encoded legacy RSA public
hidden service descriptor.

The HANDSHAKE_AUTH field contains the SHA1 dige

The SIG field contains an RSA signature, using PKCS1

Older versions of Tor always use a 1024-bit RSA key
keys.

Acknowledging establishment of introdu

After setting up an introduction circuit, the introduc
the hidden service host with an INTRO_ESTABLISHE

The INTRO_ESTABLISHED cell has the following cont

```
N_EXTENSIONS [1 byte]
N_EXTENSIONS times:
  EXT_FIELD_TYPE [1 byte]
  EXT_FIELD_LEN  [1 byte]
  EXT_FIELD      [EXT_FIELD_LEN bytes]
```

Older versions of Tor send back an empty INTRO_ES
must accept an empty INTRO_ESTABLISHED cell fro
paragraph is obsolete and refers to a workaround f
is included for historical reasons.]

The same rules for multiplicity, ordering, and handli
extension fields here as described [EST_INTRO] abo

Sending an INTRODUCE1 cell to t

In order to participate in the introduction protocol, a

```
* An introduction point for a service.
* The introduction authentication key fo
* The introduction encryption key for th
```

The client sends an INTRODUCE1 cell to the introdu
for the service, an identifier for the encryption key t
opaque blob to be relayed to the hidden service hos

In reply, the introduction point sends an INTRODUC
informing it that its request has been delivered, or t

```
   [TODO: specify what tor should do when rec
it?
         Kill circuit? This goes for all pos
```

Extensible INTRODUCE1 cell format

When a client is connecting to an introduction point
form:

```
LEGACY_KEY_ID   [20 bytes]
AUTH_KEY_TYPE   [1 byte]
AUTH_KEY_LEN    [2 bytes]
AUTH_KEY        [AUTH_KEY_LEN bytes]
N_EXTENSIONS    [1 byte]
N_EXTENSIONS times:
  EXT_FIELD_TYPE [1 byte]
  EXT_FIELD_LEN  [1 byte]
  EXT_FIELD      [EXT_FIELD_LEN bytes]
ENCRYPTED        [Up to end of relay pay
```

The `ENCRYPTED` field is described in the [PROCESS_I

AUTH_KEY_TYPE is defined as in [EST_INTRO]. Curre
for this cell is an Ed25519 public key [02].

The LEGACY_KEY_ID field is used to distinguish betw
INTRODUCE1 cells. In new style INTRODUCE1 cells, L
receiving an INTRODUCE1 cell, the introduction poir
LEGACY_KEY_ID is non-zero, the INTRODUCE1 cell sh
INTRODUCE1 cell by the intro point.

Upon receiving a INTRODUCE1 cell, the introduction
matches the introduction point authentication key f
the introduction point sends an INTRODUCE2 cell w
service, and sends an INTRODUCE_ACK response to

(Note that the introduction point does not "clean up
retransmits. Specifically, it does not change the orde
sent by the client.)

The same rules for multiplicity, ordering, and handli
extension fields here as described [EST_INTRO] abo

Proof-of-work extension to INTRODUCE1

This extension can be used to optionally attach a pr
request. The proof must be calculated using unique
specific service. An acceptable proof will raise the pr
according to the proof's verified computational effo

This is for the proof-of-work DoS mitigation, describ onion service introduction specification.

If used, it needs to be encoded within the N_EXTENS
cell defined in the previous section. The content is d

EXT_FIELD_TYPE:

[02] -- `PROOF_OF_WORK`

```
The EXT_FIELD content format is:

    POW_VERSION    [1 byte]
    POW_NONCE      [16 bytes]
    POW_EFFORT     [4 bytes]
    POW_SEED       [4 bytes]
    POW_SOLUTION   [16 bytes]

where:

POW_VERSION is 1 for the protocol specified h
POW_NONCE is the nonce value chosen by the cl
POW_EFFORT is the effort value chosen by the
           as a 32-bit integer in network byt
POW_SEED identifies which seed was in use, by
POW_SOLUTION is a matching proof computed by
```

Only version 1 is currently defined. Other versions n
correctly functioning client only submits solutions w
advertised by the server and have not yet expired. A
or expired seed is suspicious and SHOULD result in

This will increase the INTRODUCE1 payload size by 4
length is 2 extra bytes, the N_EXTENSIONS field is al
and the EXT_FIELD is 41 bytes. According to ticket #3
have more than 200 bytes available.

Introduced in tor-0.4.8.1-alpha.

INTRODUCE_ACK cell format.

An INTRODUCE_ACK cell has the following fields:

```
  STATUS       [2 bytes]
  N_EXTENSIONS [1 bytes]
  N_EXTENSIONS times:
    EXT_FIELD_TYPE [1 byte]
    EXT_FIELD_LEN  [1 byte]
    EXT_FIELD      [EXT_FIELD_LEN bytes]

Recognized status values are:

  [00 00] -- Success: cell relayed to hidd
  [00 01] -- Failure: service ID not recog
  [00 02] -- Bad message format
  [00 03] -- Can't relay cell to service
```

The same rules for multiplicity, ordering, and handli
extension fields here as described [EST_INTRO] abo

Processing an INTRODUCE2 cell a

Upon receiving an INTRODUCE2 cell, the hidden ser
AUTH_KEY or LEGACY_KEY_ID field matches the keys

The service host then checks whether it has receive
rendezvous cookie before. If it has, it silently drops i
replay cache for as long as it accepts cells with the s
encryption format below should be non-malleable.)

If the cell is not a replay, it decrypts the ENCRYPTED
the client, and authenticates the whole contents of t
since they left the client. There may be multiple way
depending on the chosen type of the encryption key
handshake protocol are described in [INTRO-HANDS
section [NTOR-WITH-EXTRA-DATA].

The decrypted plaintext must have the form:

```
RENDEZVOUS_COOKIE
N_EXTENSIONS
N_EXTENSIONS times:
    EXT_FIELD_TYPE
    EXT_FIELD_LEN
    EXT_FIELD
ONION_KEY_TYPE
ONION_KEY_LEN
ONION_KEY
NSPEC      (Number of link specifiers)
NSPEC times:
    LSTYPE (Link specifier type)
    LSLEN  (Link specifier length)
    LSPEC  (Link specifier)
PAD        (optional padding)
```

Upon processing this plaintext, the hidden service n
authentication is present in the extension fields, anc
the node described in the LSPEC fields, using the ON
As mentioned in [BUILDING-BLOCKS], the "TLS-over-
specifiers must be present.

As of 0.4.1.1-alpha, clients include both IPv4 and IPv
All available addresses SHOULD be included in the c
client actually used to extend to the rendezvous poi

The hidden service should handle invalid or unrecog
clients do in section 2.5.2.2. In particular, services SH
on link specifiers, and SHOULD NOT reject unrecogn
information leaks. The list of link specifiers received
sent verbatim when extending to the rendezvous pc

The service MAY reject the list of link specifiers if it is
from the directory, but SHOULD NOT modify it.

The ONION_KEY_TYPE field is:

[01] NTOR: ONION_KEY is 32 bytes long.

The ONION_KEY field describes the onion key that n
rendezvous point. It must be of a type listed as supp
descriptor.

The PAD field should be filled with zeros; its size sho
INTRODUCE2 message occupies a fixed maximum s
encrypted data. (This maximum size is 490, since we
implementations will implement proposal 340 and t
can be contained in a single relay message.) Note al
pad the INTRODUCE2 message up to 246 bytes.

Upon receiving a well-formed INTRODUCE2 cell, the

```
* The information needed to connect to t
  rendezvous point.
* The second half of a handshake to auth
  shared key with the hidden service cl
* A set of shared keys to use for end-to
```

The same rules for multiplicity, ordering, and handli
extension fields here as described [EST_INTRO] abo

INTRODUCE1 Extensions

The following sections details the currently support
`INTRODUCE1` cell.

Congestion Control

This is used to request that the rendezvous circuit w
congestion control.

EXT_FIELD_TYPE:

```
\[01\] -- Congestion Control Request.
```

This field is has zero payload length. Its presence sig
congestion control. The client MUST NOT set this fie
list "2" in the `FlowCtrl` line in the descriptor. The cl
the consensus parameter 'cc_alg' is 0.

The service MUST ignore any unknown fields.

Proof-of-Work (PoW)

This is used to send the work done by the client to t

EXT_FIELD_TYPE:

```
\[02\] -- Proof Of Work.
```

EXT_FIELD content format is:

```
POW_VERSION    [1 byte]
POW_NONCE      [16 bytes]
POW_EFFORT     [4 bytes]
POW_SEED       [4 bytes]
POW_SOLUTION   [16 bytes]
```

where:

```
POW_VERSION is 1 for the protocol specified
POW_NONCE is the nonce 'N' from the section
POW_EFFORT is the 32-bit integer effort valu
POW_SEED is the first 4 bytes of the seed us
```

When a service receives an INTRODUCE1 with the ex
configuration on whether proof-of-work is enabled
extension SHOULD be ignored. If enabled, even if th
the service follows the procedure detailed in prop32

If the service requires the PROOF_OF_WORK extensi
without any embedded proof-of-work, the service S
effort introduction for the purposes of the priority q
prop327).

(TODO: We should have a proof-of-work.md to fold
proposal.)

Subprotocol Request

[RESERVED]

EXT_FIELD_TYPE:

```
\[03\] -- Subprotocol Request
```

Introduction handshake encryption requi

When decoding the encrypted information in an INT
must be able to:

```
* Decrypt additional information include
  to include the rendezvous token and th
  extend to the rendezvous point.

* Establish a set of shared keys for use

* Authenticate that the cell has not bee
  generated it.
```

Note that the old TAP-derived protocol of the previc
first two requirements, but not the third.

```
3.3.2. Example encryption handshake: ntor wit
       [NTOR-WITH-EXTRA-DATA]

   [TODO: relocate this]
```

This is a variant of the ntor handshake (see tor-spec
and see "Anonymity and one-way authentication in
Stebila, and Ustaoglu).

It behaves the same as the ntor handshake, except
forward secure keys, it also provides a means for er
the server (in this case, to the hidden service host) a

Notation here is as in section 5.1.4 of tor-spec.txt, w

The PROTOID for this variant is "tor-hs-ntor-curve25
following tweak values:

```
t_hsenc    = PROTOID | ":hs_key_extract
t_hsverify = PROTOID | ":hs_verify"
t_hsmac    = PROTOID | ":hs_mac"
m_hsexpand = PROTOID | ":hs_key_expand'
```

To make an INTRODUCE1 cell, the client must know
hidden service on this introduction circuit. The clien

x,X = KEYGEN()

and computes:

```
            intro_secret_hs_input = EXP(B,x)
            info = m_hsexpand | N_hs_subcred
            hs_keys = KDF(intro_secret_hs_ir
S_KEY_LEN+MAC_LEN)
            ENC_KEY = hs_keys[0:S_KEY_LEN]
            MAC_KEY = hs_keys[S_KEY_LEN:S_KE

   and sends, as the ENCRYPTED part of the IN

         CLIENT_PK                [PK_PUBKEY
         ENCRYPTED_DATA           [Padded to
         MAC                      [MAC_LEN b
```

Substituting those fields into the INTRODUCE1 cell b
[FMT_INTRO1] above, we have

```
          LEGACY_KEY_ID                [20 b
          AUTH_KEY_TYPE                [1 by
          AUTH_KEY_LEN                 [2 by
          AUTH_KEY                     [AUTH
          N_EXTENSIONS                 [1 by
          N_EXTENSIONS times:
             EXT_FIELD_TYPE            [1 by
             EXT_FIELD_LEN             [1 by
             EXT_FIELD                 [EXT_
          ENCRYPTED:
             CLIENT_PK                 [PK_P
             ENCRYPTED_DATA            [Padd
             MAC                       [MAC_
```

(This format is as documented in [FMT_INTRO1] abo
to build the ENCRYPTED portion.)

Here, the encryption key plays the role of B in the re
AUTH_KEY field plays the role of the node ID. The CL
ENCRYPTED_DATA field is the message plaintext, en
ENC_KEY. The MAC field is a MAC of all of the cell fro
ENCRYPTED_DATA, using the MAC_KEY value as its k

To process this format, the hidden service checks PK
then computes ENC_KEY and MAC_KEY as the client
EXP(CLIENT_PK,b) in the calculation of intro_secret_h
whether the MAC is correct. If it is invalid, it drops th
plaintext by decrypting ENCRYPTED_DATA.

The hidden service host now completes the service
as described in tor-spec.txt section 5.1.4, with the m
be explicit, the hidden service host generates a keyp
introduction point encryption key 'b' to compute:

```
      intro_secret_hs_input = EXP(X,b) | AUTH
      info = m_hsexpand | N_hs_subcred
      hs_keys = KDF(intro_secret_hs_input | t
S_KEY_LEN+MAC_LEN)
      HS_DEC_KEY = hs_keys[0:S_KEY_LEN]
      HS_MAC_KEY = hs_keys[S_KEY_LEN:S_KEY_LE

      (The above are used to check the MAC ar
      encrypted data.)

      rend_secret_hs_input = EXP(X,y) | EXP(X
PROTOID
      NTOR_KEY_SEED = MAC(rend_secret_hs_inpu
      verify = MAC(rend_secret_hs_input, t_hs
      auth_input = verify | AUTH_KEY | B | Y
      AUTH_INPUT_MAC = MAC(auth_input, t_hsma

      (The above are used to finish the ntor

   The server's handshake reply is:

       SERVER_PK   Y
       AUTH        AUTH_INPUT_MAC
```

These fields will be sent to the client in a RENDEZVO element (see [JOIN_REND]).

The hidden service host now also knows the keys ge will use to encrypt and authenticate data end-to-end These keys are as computed in tor-spec.txt section 5 AES-128 and SHA1 for this hop, we use AES-256 and

Authentication during the introd

Hidden services may restrict access only to authoriz the credential mechanism, where only users who kn may connect at all.

There is one defined authentication type: `ed25519` .

Ed25519-based authentication `ed25519`

(NOTE: This section is not implemented by Tor. It is l its design substantially before deploying any implem want to bind these extensions to a single onion serv want to look for ways to limit the number of keys a u

To authenticate with an Ed25519 private key, the us
the encrypted part of the INTRODUCE1 cell with an
contents:

```
Nonce     [16 bytes]
Pubkey    [32 bytes]
Signature [64 bytes]
```

Nonce is a random value. Pubkey is the public key t
[TODO: should this be an identifier for the public ke
using Ed25519, of:

```
"hidserv-userauth-ed25519"
Nonce       (same as above)
Pubkey      (same as above)
AUTH_KEY    (As in the INTRODUCE1 cel
```

The hidden service host checks this by seeing wheth
signature from the provided public key. If it would, t
correct. If it is, then the correct user has authenticat

Replay prevention on the whole cell is sufficient to p

Users SHOULD NOT use the same public key with m

The rendezvous protoco

Before connecting to a hidden service, the client firs
chosen Tor node (known as the rendezvous point), a
ESTABLISH_RENDEZVOUS cell. The hidden service la
sends a RENDEZVOUS cell. Once this has occurred, t
RENDEZVOUS cell to the client, and joins the two cir

Single Onion Services attempt to build a non-anony
anonymous 3-hop circuit if:

```
* the rend point is on an address that 
  a direct connection, or
* the initial attempt to connect to the
  circuit fails, and they are retrying 
```

Establishing a rendezvous point

The client sends the rendezvous point a RELAY_COM
containing a 20-byte value.

RENDEZVOUS_COOKIE [20 bytes]

Rendezvous points MUST ignore any extra bytes in a
(Older versions of Tor did not.)

The rendezvous cookie is an arbitrary 20-byte value,
client SHOULD choose a new rendezvous cookie for
rendezvous cookie is already in use on an existing c
reject it and destroy the circuit.

Upon receiving an ESTABLISH_RENDEZVOUS cell, th
cookie with the circuit on which it was sent. It replie
RENDEZVOUS_ESTABLISHED cell to indicate success
in a RENDEZVOUS_ESTABLISHED cell.

The client MUST NOT use the circuit which sent the
rendezvous with the given location-hidden service.

The client should establish a rendezvous point BEFC
service.

Joining to a rendezvous point

To complete a rendezvous, the hidden service host
point and sends a RENDEZVOUS1 cell containing:

```
        RENDEZVOUS_COOKIE           [20 bytes]
        HANDSHAKE_INFO              [variable;
                                     used.]
```

where RENDEZVOUS_COOKIE is the cookie suggeste
introduction (see [PROCESS_INTRO2]) and HANDSH
EXTRA-DATA].

If the cookie matches the rendezvous cookie set on
rendezvous point, the rendezvous point connects th
RENDEZVOUS2 cell to the client containing the HAN
RENDEZVOUS1 cell.

Upon receiving the RENDEZVOUS2 cell, the client ve
completes a handshake. To do so, the client parses
and reverses the final operations of section [NTOR-

```
      rend_secret_hs_input = EXP(Y,x) | EXP(
 PROTOID
      NTOR_KEY_SEED = MAC(rend_secret_hs_inpu
      verify = MAC(rend_secret_hs_input, t_hs
      auth_input = verify | AUTH_KEY | B | Y
      AUTH_INPUT_MAC = MAC(auth_input, t_hsma
```

Finally the client verifies that the received AUTH fiel
computed AUTH_INPUT_MAC.

Now both parties use the handshake output to deri
as specified in the section below:

Key expansion

The hidden service and its client need to derive cryp
part of the handshake output. To do so, they use th

K = KDF(NTOR_KEY_SEED | m_hsexpand, HASH_LEN

The first HASH_LEN bytes of K form the forward dig
the backward digest Db; the next S_KEY_LEN bytes f
form Kb. Excess bytes from K are discarded.

Subsequently, the rendezvous point passes relay ce
circuits to the other. When Alice's OP sends RELAY c
with Df, and encrypts them with the Kf, then with all
of the circuit; and when Alice's OP receives RELAY ce
with the keys for the ORs in Alice's side of the circuit
checks integrity with Db. Bob's OP does the same, w

[TODO: Should we encrypt HANDSHAKE_INFO as we
necessary, but it could be wise. Similarly, we should

Using legacy hosts as rendezvou

[This section is obsolete and refers to a workarounc
It is included for historical reasons.]

The behavior of ESTABLISH_RENDEZVOUS is unchar
protocol, except that relays should now ignore unex

Old versions of Tor required that RENDEZVOUS cell
All shorter rendezvous payloads should be padded t
make them difficult to distinguish from older protoc

Relays older than 0.2.9.1 should not be used for ren
onion services because they enforce too-strict lengt
the "HSRend" protocol from proposal#264 should b
points.

Encrypting data betwee

A successfully completed handshake, as embedded
cells, gives the client and hidden service host a shar
they use for sending end-to-end traffic encryption a
Tor relay encryption protocol, applying encryption w
encryption, and decrypting with these keys before o
with Kf and decrypts with Kb; the service host does

Encoding onion address [ONIONADDRESS]

The onion address of a hidden service includes its ic basic checksum. All this information is then base32

```
      onion_address = base32(PUBKEY | CHECKSUM
      CHECKSUM = H(".onion checksum" | PUBKEY

      where:
        - PUBKEY is the 32 bytes ed25519 maste
        - VERSION is a one byte version field
        - ".onion checksum" is a constant str
        - CHECKSUM is truncated to two bytes b
 onion_address

   Here are a few example addresses:

        pg6mmjiyjmcrsslvykfwnntlaru7p5svn6y2ym
        sp3k262uwy4r2k3ycr5awluarykdpag6a7y33j
        xa4r2iadxm55fbnqgwwi5mymqdcofiu3w6rpbt
```

For historical notes and rationales about this enc

Managing streams

Sending BEGIN messages

In order to open a new stream to an onion service, t
an established rendezvous circuit.

When sending a BEGIN message to an onion service
as the target address, and not set any flags on the b

> For example, to open a connection to `<some_add`
> would send a BEGIN message with the address:p
> value of 0. The 0-values `FLAGS` would not be enc
> for encoding BEGIN messages.

Receiving BEGIN messages

When a service receives a BEGIN message, it should
fields in the begin message, including its address and

If a service chooses to reject a BEGIN message, it sh
entirely to prevent port scanning, resource exhausti
If it does not, it should send back an `END` message
any further information.

References

How can we improve the HSDir unpredictability des
See these references for discussion.

```
[SHAREDRANDOM-REFS]:
        https://gitweb.torproject.org/torspec
reveal-consensus.txt
        https://trac.torproject.org/projects/
```

Scaling hidden services is hard. There are on-going
help with:

```
[SCALING-REFS]:
        https://lists.torproject.org/piperma
/005556.html
```

How can hidden service addresses become memora
authenticating and decentralized nature? See these
more are possible.

```
[HUMANE-HSADDRESSES-REFS]:
        https://gitweb.torproject.org/torspec
/xxx-onion-nyms.txt
        http://archives.seul.org/or/dev/Dec-2
```

Hidden Services are pretty slow. Both because of th
because the final circuit has 6 hops. How can we ma
faster? See these references for some suggestions.

```
[PERFORMANCE-REFS]:
        "Improving Efficiency and Simplicity
        establishment and hidden services" by
        P. Syverson

        [TODO: Need more here! Do we have any
```

Other references:

```
[KEYBLIND-REFS]:
        https://trac.torproject.org/projects/
        https://lists.torproject.org/piperma
/004026.html

[KEYBLIND-PROOF]:
        https://lists.torproject.org/piperma
/005943.html

[ATTACK-REFS]:
        "Trawling for Tor Hidden Services: De
        Deanonymization" by Alex Biryukov, Iv
        Ralf-Philipp Weinmann

        "Locating Hidden Servers" by Lasse Øv
        Syverson

[ED25519-REFS]:
        "High-speed high-security signatures'
        J. Bernstein, Niels Duif, Tanja Lange
        Bo-Yin Yang. http://cr.yp.to/papers.h

[ED25519-B-REF]:
        https://tools.ietf.org/html/draft-jos
5:

[PRNG-REFS]:
        http://projectbullrun.org/dual-ec/ext
        https://lists.torproject.org/piperma
/009954.html

[SRV-TP-REFS]:
        https://lists.torproject.org/piperma

[VANITY-REFS]:
        https://github.com/Yawning/horse2551

[ONIONADDRESS-REFS]:
        https://lists.torproject.org/piperma
/011816.html

[TORSION-REFS]:
        https://lists.torproject.org/piperma
        https://getmonero.org/2017/05/17/disc
cryptonote-based-currencies.html
```

Appendix A: Signature s blinding

Key derivation overview

As described in [IMD:DIST] and [SUBCRED] above, w works (roughly) as follows:

There is a master keypair (sk, pk).

```
Given the keypair and a nonce n, ther
that gives a new blinded keypair (sk_
be used for signing.

Given only the public key and the nor
that gives pk_n.

Without knowing pk, it is not possibl
knowing sk, it is not possible to der

It's possible to check that a signatu
knowing only pk_n.

Someone who sees a large number of bl
signatures made using those public ke
signatures and which blinded keys wer
master keypair.

You can't forge signatures.

[TODO: Insert a more rigorous definit
```

Tor's key derivation scheme

We propose the following scheme for key blinding, b

(This is an ECC group, so remember that scalar mult and it's defined in terms of iterated point addition. S ED25519-REFS] for a fairly clear writeup.)

Let B be the ed25519 basepoint as found in section

```
      B =
(15112221349535400772501151409588531511454012
2,

463168356949264781694283940034751631413079938
)
```

Assume B has prime order l, so lB=0. Let a master k
the private key and A is the public key (A=aB).

To derive the key for a nonce N and an optional sec
this:

```
           h = H(BLIND_STRING | A | s | B | N
           BLIND_STRING = "Derive temporary s
           N = "key-blind" | INT_8(period-num
           B = "(1511[...]2202, 4631[...]5960

   then clamp the blinding factor 'h' accordin

           h[0] &= 248;
           h[31] &= 63;
           h[31] |= 64;

   and do the key derivation as follows:

       private key for the period:

           a' = h a mod l
           RH' = SHA-512(RH_BLIND_STRING | RH
           RH_BLIND_STRING = "Derive temporar

       public key for the period:

           A' = h A = (ha)B
```

Generating a signature of M: given a deterministic r
take R=rB, S=r+hash(R,A',M)ah mod l. Send signatur

Verifying the signature: Check whether SB = R+hash

```
   (If the signature is valid,
        SB = (r + hash(R,A',M)ah)B
           = rB + (hash(R,A',M)ah)B
           = R + hash(R,A',M)A' )

   This boils down to regular Ed25519 with key
```

See [KEYBLIND-REFS] for an extensive discussion on
alternatives. Also, see [KEYBLIND-PROOF] for a secu

Appendix B: Selecting nc

Picking introduction points Picking rendezvous poin

(TODO: This needs a writeup)

Appendix C: Recommend searching for vanity .oni

EDITORIAL NOTE: The author thinks that it's silly to b that, when base-32 encoded, spells out the name of dangerous to me. If you train your users to connect

llamanymityx4fi3l6x2gyzmtmgxjyqyorj9qsb5r543izc

I worry that you're making it easier for somebody to

llamanymityb4sqi0ta0tsw6uovyhwlezkcrmczeuzdvfa

Nevertheless, people are probably going to try to do use.

To search for a public key with some criterion X:

Generate a random (sk,pk) pair.

While pk does not satisfy X:

```
        Add the number 8 to sk
        Add the point 8*B to pk

    Return sk, pk.
```

We add 8 and 8*B, rather than 1 and B, so that sk is key, with the lowest 3 bits equal to 0.

This algorithm is safe [source: djb, personal commu understood correctly!] so long as only the final (sk,p are discarded.

To parallelize this algorithm, start with an independ independent thread, and let each search proceed in

See [VANITY-REFS] for a reference implementation o

Appendix E: Reserved nu

We reserve these certificate type values for Ed2551

```
      [08] short-term descriptor signing key,
           public key. (Section 2.4)
      [09] intro point authentication key, cr
           signing key. (Section 2.5)
      [0B] ed25519 key derived from the curve
key,
           cross-certifying the descriptor s

      Note: The value "0A" is skipped because
            cross-certifying ntor identity ke
```

Appendix F: Hidden serv format [HIDSERVDIR-FOI

This appendix section specifies the contents of the I

- "hostname" [FILE]

This file contains the onion address of the onion ser

- "private_key_ed25519" [FILE]

This file contains the private master ed25519 key of keys]

```
- "./authorized_clients/"
  "./authorized_clients/alice.auth"
  "./authorized_clients/bob.auth"
  "./authorized_clients/charlie.auth"
```

If client authorization is enabled, this directory MUS authorized client. Each such file contains the public are transmitted to the service operator by the client

See section [CLIENT-AUTH-MGMT] for more details a

(NOTE: client authorization is implemented as of 0.3

Appendix G: Managing a data [CLIENT-AUTH-MGM

Hidden services and clients can configure their auth torrc, or using the control port. This section present client authorization. Please see appendix [HIDSERVD about relevant hidden service files.

(NOTE: client authorization is implemented as of 0.3

G.1. Configuring client authorization using torrc

G.1.1. Hidden Service side configuration

```
     A hidden service that wants to enable cl
     populate the "authorized_clients/" direc
     directory with the ".auth" files of its

     When Tor starts up with a configured oni
     <HiddenServiceDir>/authorized_clients/ d
if
     any recognized and parseable such files
     authorization becomes activated for that

  G.1.2. Service-side bookkeeping

     This section contains more details on ho
keeping
     track of their client ".auth" files.

     For the "descriptor" authentication type
     the x25519 public key of that client. He

        <auth-type>:<key-type>:<base32-encode

     Here is an an example:


descriptor:x25519:OM7TGIVRYMY6PFX6GAC6ATRTA5U

     Tor SHOULD ignore lines it does not reco
     Tor SHOULD ignore files that don't use t

  G.1.3. Client side configuration

     A client who wants to register client au
     services needs to add the following line
     directory which hosts ".auth_private" fi
     credentials for onion services:

         ClientOnionAuthDir <DIR>

     The <DIR> contains a file with the suffi
     service the client is authorized with. T
     ".auth_private" files to find which onio
     authorization from this client.

     For the "descriptor" auth-type, a ".auth
     private x25519 key:

        <onion-address>:descriptor:x25519:<ba

     The keypair used for client authorizatio
tool
     for which the public key needs to be tra
     in a secure out-of-band way. The third p
     headers to the private key file to ensur
     give out their private key.

  G.2. Configuring client authorization using
```

```
  G.2.1. Service side

     A hidden service also has the option to
     using the control port. The idea is that
use
     controller utilities that manage their a
     the filesystem to register client keys.

     Specifically, we require a new control p
ADD_ONION_CLIENT_AUTH
     which is able to register x25519/ed2551
     authorized client.
      [XXX figure out control port command fo

     Hidden services who use the control port
     to perform their own key management.

  G.2.2. Client side

     There should also be a control port inte
     authorization data for hidden services w
     torrc. It should allow both generation o
     keys, and also to import client authoriz
     service

     This way, Tor Browser can present "Gener
"Import
     client auth keys" dialogs to users when
service
     that is protected by client authorizatio

     Specifically, we require two new control
                  IMPORT_ONION_CLIENT_AUTH_D
                  GENERATE_ONION_CLIENT_AUTH
     which import and generate client authori

     [XXX how does key management work here?]
     [XXX what happens when people use both t
the
          filesystem interface?]
```

Appendix F: Two methoc revision counters

Implementations MAY generate revision counters ir are monotonically increasing over the lifetime of ea fingerprinting, implementors SHOULD choose a stra implementations. Here we describe two, and additic implementors should NOT use.

F.1. Increment-on-generation

This is the simplest strategy, and the one used by Tc alpha.

Whenever using a new blinded key, the service reco used with that key. When generating a descriptor, th negative number higher than any number it has alre

In other words, the revision counters under this syst as 0, 1, 2, 3, and so on.

F.2. Encrypted time in period

This scheme is what we recommend for situations v to coordinate their revision counters, without an act

Let T be the number of seconds that have elapsed s plus 1. (T must be at least 1.)

Let S be a per-time-period secret that all the service `= KS_hs_blind_id`; when `KS_hs_blind_id` is not av `= KS_hs_desc_sign`.)

Let K be an AES-256 key, generated as

```
K = H("rev-counter-generation" | S)
```

Use `K`, and AES in counter mode with IV=0, to gene Consider these bytes as a sequence of T 16-bit little-

Let the sum of these words, plus T, be the revision c

> (We include T in the sum so that every increment
> output.)

Cryptowiki attributes roughly this scheme to G. Beb

> G. Bebek. Anti-tamper database research: Inferer
> Report EECS 433 Final Report, Case Western Rese

Although we believe it is suitable for use in this appl
preserving encryption algorithm (and all order-pres
Please think twice before using it for anything else.

(This scheme can be optimized pretty easily by cach
etc for some well chosen `X`.)

For a slow reference implementation that can gener
`/ope_ref.py` in the Tor source repository.

> Note:
>
> Some onion service operators have historically re
> scheme to provide a kind of ersatz load-balancing
> each with the the same `KP_hs_id`. The onion ser
> points, and race each other to publish their HSDe
>
> It's probably better to use Onionbalance or a sim
> Nonetheless, onion services implemenentations I
> particular OPE scheme exactly in order to provide
> use case.

F.X. Some revision-counter strat

Though it might be tempting, implementations SHO
current time within the period directly as their revisi
view of the current time, which can be used to link t
on the same host.

Similarly, implementations SHOULD NOT let the rev
resetting it -- doing so links the service across chang

Appendix G: Text vector

G.1. Test vectors for hs-ntor / NTOR-WITH-EXTRA-DA

```
Here is a set of test values for the hs-r
[NTOR-WITH-EXTRA-DATA] in this document.
instrumenting Tor's code to dump the valu
handshake, and then by running that code

We assume an onion service with:

  KP_hs_ipd_sid = 34E171E4358E501BFF21ED9
                  FEF697C779D040BBAF49AC0
  KP_hss_ntor   = 8E5127A40E83AABF6493E41
                  604B85A3961CD7E38D24723
  KS_hss_ntor   = A0ED5DBF94EEB2EDB3B514E
                  022051CC5F103391F1970A3
  N_hs_subcred  = 0085D26A9DEBA252263BF02
                  17CA11BAD8A218238AD6487

The client wants to make in INTRODUCE rec
the following header (everything before t
of its INTRODUCE1 cell:

  H = 000000000000000000000000000000000
      21ED907E96AC6BFEF697C779D040BBAF49A

It generates the following plaintext body
is the "decrypted plaintext body" from [F

  P = 6BD364C12638DD5C3BE23D76ACA05B04E6C
      C4EDDA24E21220CC3EADAE403EF6B7D11C8
      0113890214F823C4F8CC085C792E0AEE028
      7B7077A0118A900FF4456C382F0041300A0
      000000000000000000000000000000000
      000000000000000000000000000000000

  (Note! This should in fact be padded to
  test vectors were generated, the target
  Tor was needlessly short.)

The client now begins the hs-ntor handsha
a curve25519 keypair:

  x = 60B4D6BF5234DCF87A4E9D7487BDF3F4
      A69B6729835E825CA29089CFDDA1E341
  X = BF04348B46D09AED726F1D66C618FDEA
      1DE58E8CB8B89738D7356A0C59111D5D

Then it calculates:

  ENC_KEY = 9B8917BA3D05F3130DACCE5300C3D
            F6D012912F1C733036F822D0ED238
  MAC_KEY = FC4058DA59D4DF61E7B40985D122F
            FD59336BC21C30CAF5E7F0D4A2C38

With these, it encrypts the plaintext boc
an encrypted value C.  It computes MAC(MA
getting a MAC value M.  It then assembles
body as H | X | C | M:
```

```
0000000000000000000000000000000000000
21ED907E96AC6BFEF697C779D040BBAF49ACC3
726F1D66C618FDEA1DE58E8CB8B89738D7356A
4DCC179D3D9E437B452AF5702CED2CCFEC085B
563C362FDFFB802DAB8CD9EBC7A5EE17DA62E3
3F391B7566F42ADC97C46BA7588278273A44CE
7E0F173DBC0BDDCD4ACB4C4600980A7DDD9EAE
35717012B78B4930569F895CB349A07538E423
DEE97DA623F1AEC0A47F150002150455845C38
D1A51B7554D8B3692D85AC587FB9E69DF990EF
```

Later the service receives that body in an INTRODU
the hs-ntor handshake, and recovers the client's pla
handshake, the service chooses a curve25519 keyp

```
y = 68CB5188CA0CD7924250404FAB54EE13
    92D3D2B9C049A2E446513875952F8F55
Y = 8FBE0DB4D4A9C7FF46701E3E0EE7FD05
    CD28BE4F302460ADDEEC9E93354EE700
```

```
From this and the client's input, it comp
```

```
AUTH_INPUT_MAC = 4A92E8437B8424D5E5EC2
                 25A0327ACF6DAF902079F
NTOR_KEY_SEED  = 4D0C72FE8AFF35559D95E
                 83402B28CDFD48C8A530A
```

The service sends back Y | AUTH_INPUT_MAC in its
the client finishes the handshake, validates AUTH_IN
NTOR_KEY_SEED.

Now that both parties have the same NTOR_KEY_SE
material they will use for their circuit.

Proof of Work for onion introduction

The overall denial-of-service prevention strategies ir
service prevention mechanisms in Tor document. Tl
mitigation, the proof-of-work client puzzle for onion

This was originally proposal 327, A First Take at PoW
by George Kadianakis, Mike Perry, David Goulet, anc

Motivation

See the denial-of-service overview for the big-pictur
mitigation for attacks on one specific resource: onio

Attackers can generate low-effort floods of introduc
and all involved relays to perform a disproportionat
of-service opportunity. This proof-of-work scheme i
unattractive to attackers, reducing the network-wid

Previous to this work, our attempts at limiting the in
attacks on onion services has been focused on horiz
optimizing the CPU usage of Tor and applying rate li
the goalpost forward, a core problem with onion se
circuits is a costly procedure both for the service an

For more information on the limitations of rate-limit
see `draft-nygren-tls-client-puzzles-02`.

If we ever hope to have truly reachable global onion
for attackers to overload the service with introducti
this by allowing onion services to specify an optiona
its clients need to participate in if they want to get s

With the right parameters, this proof-of-work schem
amplification attacks by attackers while letting legiti

Related work

For a similar concept, see the three internet drafts t
defending against TLS-based DDoS attacks using cli

- `draft-nygren-tls-client-puzzles-02`
- `draft-nir-tls-puzzles-00`
- `draft-ietf-ipsecme-ddos-protection-10`

Threat model

Attacker profiles

This mitigation is written to thwart specific attackers
to defend against all and every DoS attack on the In
models we can defend against.

Let's start with some adversary profiles:

- "The script-kiddie"

  The script-kiddie has a single computer and pu
  has a VPS and a pwned server. We are talking a
  10 GHz of CPU and 10 GB of RAM. We conside
  zero $.

- "The small botnet"

  The small botnet is a bunch of computers line
  attack. Assuming 500 medium-range compute
  with total access to 10 THz of CPU and 10 TB o
  for this attacker to be about $400.

- "The large botnet"

  The large botnet is a serious operation with m
  organized to do this attack. Assuming 100k me
  talking about an attacker with total access to 2
  The upfront cost for this attacker is about $36

We hope that this proposal can help us defend agai
small botnets. To defend against a large botnet we
(see the discussion on future designs).

User profiles

We have attackers and we have users. Here are a fe

- "The standard web user"

  This is a standard laptop/desktop user who is
  know how these defences work and they don't
  the site doesn't load, they are gonna close thei
  a 2GHz computer with 4GB of RAM.

- "The motivated user"

  This is a user that really wants to reach their d
  journey; they just want to get there. They know
  make their computer do expensive multi-minu

they want to be.

- "The mobile user"

  This is a motivated user on a mobile phone. Ev
  article, they don't have much leeway on stress
  computation.

We hope that this proposal will allow the motivated
want to connect to, and also give more chances to tl
destination.

The DoS Catch-22

This proposal is not perfect and it does not cover all
covering some use cases and giving reachability to t
severely demotivate the attackers from continuing t
DoS threat all together. Furthermore, by increasing
class of DoS attackers will disappear from the map,

Common protocol

We have made an effort to split the design of the pr
algorithm-specific piece that can be upgraded, and a
queueing and effort adjustment.

Currently there is only one versioned subprotocol d

- Version 1, Equi-X and Blake2b

Overview

```
 +----------------------------------+
                                                  |
 |
    +-------+ INTRO1  +-----------+ INTRO2 +--
 |
    |Client |-------->|Intro Point|------->|
 |
    +-------+         +-----------+        |Ve
 |
                                           +--
 |
                                                  |
 |
                                                  |
 |
                                                  |
 |
                                                  |
 |
 +---------+--------------------+---+
```

The proof-of-work scheme specified in this documen
phase of the onion service protocol.

The system described in this proposal is not meant t
entirely disabled for services that do not experience

When the subsystem is enabled, suggested effort is
computational puzzle can be bypassed entirely whe
cases, the proof-of-work subsystem can be dorman

parameters for clients to voluntarily provide effort i
priority queue.

The protocol involves the following major steps:

1. Service encodes PoW parameters in descripto
   plaintext format.
2. Client fetches descriptor and begins solving. C
   algorithm.
3. Client finishes solving and sends results using
   INTRODUCE1.
4. Service verifies the proof and queues an intro
   currently uses the `v1` verify algorithm only.
5. Requests are continuously drained from the q
   multiple constraints on speed. See below for n

Replay protection

The service MUST NOT accept introduction requests
For this reason a replay protection mechanism mus

The simplest way is to use a hash table to check wh
used before for the active duration of a seed. Depe
this might be a viable solution with reasonable mem

If there is a worry that we might get too many intro
we can use a Bloom filter or similar as our replay ca
means that we will potentially flag some connection
this false positive probability increasing as the num
parameter tuning this probability should be negligi
retried by the client.

The introduction queue

When proof-of-work is enabled for a service, that se
requests to a priority queue system rather than han

Adding introductions to the introduction

When PoW is enabled and an introduction request i

queues each request in a data structure sorted by e
all MUST be assigned an effort of zero. Requests wit
rejected and not enqueued.

Services MUST check whether the queue is overfull
processing requests. Floods of low-effort and zero-e
efficiently discarded when the queue is growing fast

The C implementation chooses a maximum number
configured dequeue rate limit multiplied by the circu
threshold are expected not to be reachable by the t
is exceeded, the queue experiences a mass trim eve
items are discarded.

Handling queued introductions

When deciding which introduction request to consid
highest available effort. When efforts are equivalent
chosen.

The service should handle introductions only by pul
call this part of introduction handling the "bottom h
happens in this stage.

For more on how we expect such a system to work i
discussion section.

Effort control

Overall strategy for effort determination

Denial-of-service is a dynamic problem where the at
change, and hence we want our proof-of-work syste
static difficulty setting. Instead of forcing clients to g
the service operator, we ask clients to "bid" using th
higher priority the higher effort they put into their p
increase their bid when retrying, and services regula
based on the recent queue status.

Motivated users can spend a high amount of effort i
should guarantee access to the service given reasor

An effective effort control algorithm will improve rea
that reduce overall service load to tolerable values w
tolerable overall delay.

The service starts with a default suggested-effort va
defenses dormant until we notice signs of queue ov

The entire process of determining effort can be thou
feedback loops. Clients perform their own effort adj
base effort suggested by the service. That suggestio
adjustments atop a base effort calculated using a su

Each feedback loop has an opportunity to cover diff
adjustments at every single circuit creation request,
extra load that frequent updates would place on HS

In the combined client/service system these client-s
the most effective quick response to an emerging D
the effort using timeouts, later clients benefit from t
queued effort and publishing a larger suggested eff

Effort increases and decreases both have a cost. Inc
more expensive to contact, and decreasing effort m
backlogged behind older requests. The steady state
these side-effects, but ultimately it's expected that t
some degree.

Service-side effort control

Services keep an internal suggested effort target wh
timer in response to measurements made on queue
These internal effort changes can optionally trigger
the difference is great enough to warrant republicat

This evaluation and update period is referred to as
effort control loop takes inspiration from TCP conge
multiplicative decrease approach, but unlike a typica
and doesn't update immediately in response to ever

TODO: `HS_UPDATE_PERIOD` is hardcoded to 300 (5 m
configurable in some way. Is it more appropriate to
consensus parameter?

Per-period service state

During each update period, the service maintains sc

1. `TOTAL_EFFORT` , a sum of all effort values for re successfully validated and enqueued.
2. `REND_HANDLED` , a count of rendezvous request Requests that made it to dequeueing but were included.
3. `HAD_QUEUE` , a flag which is set if at any time in queue filled with more than a minimum amou expect to process in approximately 1/4 seconc
4. `MAX_TRIMMED_EFFORT` , the largest observed sin during the period. Requests are discarded eith culling events that discard the bottom half of t

Service AIMD conditions

At the end of each period, the service may decide to make no changes, based on these accumulated stat

1. If `MAX_TRIMMED_EFFORT` > our previous interna INCREASE. Requests that follow our latest advi
2. If the `HAD_QUEUE` flag was set and the queue st effort >= our previous internal `suggested_eff` reached the point of dropping requests, this si suggestion isn't high enough and requests will
3. If neither condition 1 or 2 are taking place and expect to process in approximately 1/4 seconc
4. If none of these conditions match, the `sugges`

When we INCREASE, the internal `suggested_effort` value + 1, or (`TOTAL_EFFORT` / `REND_HANDLED`), which

When we DECREASE, the internal `suggested_effort`

Over time, this will continue to decrease our effort s processing its request queue. If the queue stays em to zero and clients should no longer submit a proof- connection attempt.

It's worth noting that the `suggested_effort` is not a accepted by the service, and it's only meant to serve the number of unsuccessful requests that get to the queue, services do accept valid solutions with effort values from `pow-params` .

Updating descriptor with new suggested effort

The service descriptors may be updated for multiple
rotation common to all v3 onion services, scheduled
for `v1` parameters, and updates to the effort sugge
value updates on a regular timer, we avoid propaga
and the HSDir hosts unless there is a significant cha

If the PoW params otherwise match but the seed ha
services SHOULD NOT upload a new descriptor.

Client-side effort control

Clients are responsible for making their own effort a
connection trouble, to allow the system a chance to
published new effort values. This is an important to
relying on excessively frequent updates through the

TODO: This is the weak link in user experience for o
implementation does not detect and retry onion ser
would like. Currently our best strategy to improve re

Failure ambiguity

The first challenge in reacting to failure, in our case,
understand when a failure has occurred.

This proposal introduces a bunch of new ways wher
the onion service. Furthermore, there is currently no
to inform the client that the introduction failed. The
from the introduction point to the client) and hence
inform the client that the rendezvous is never gonna

From the client's perspective there's no way to attrib
rather than the introduction point, so error account
introduction-point. Prior mechanisms will discard an
too many retries.

Clients handling timeouts

Alice can fail to reach the onion service if her introdu
priority queue when enqueueing new requests, or if
priority queue in time and the connection times out

This section presents a heuristic method for the clie
scenarios.

If the rendezvous request times out, the client SHOU
service to make sure that it's using the right suggest
PoW seed. If the fetched descriptor includes a new s
retry the request with these parameters.

TODO: This is not actually implemented yet, but we
at most try to fetch new descriptors? Determined by
will also allow clients to retry effectively in cases wh
reconfigured to enable PoW defenses.

Every time the client retries the connection, it will co
point. These counts of previous retries are combine
`suggested_effort` when calculating the actual effo
to a service that advertises PoW support, even when
`suggested_effort` is zero.

On each retry, the client modifies its solver effort:

1. If the effort is below `CLIENT_POW_EFFORT_DOUB`
2. Otherwise, multiply the effort by `CLIENT_POW_`
3. Constrain the effort to no less than `CLIENT_MI`
   this limit is specific to retries only. Clients may
   connection attempt.
4. Apply the maximum effort limit described belo

Client-imposed effort limits

There isn't a practical upper limit on effort defined b
choose a maximum effort limit to enforce. It may be
improve responsiveness, but the main reason for th
for weak cancellation support in our implementatio

Effort values used for both initial connections and re
greater than `CLIENT_MAX_POW_EFFORT` (= 10000).

TODO: This hardcoded limit should be replaced by t
solver with robust cancellation. This is issue 40787 i

Onion service proof-of-w Equi-X and Blake2b

Implementations

For our `v1` proof-of-work function we use the Equi- by tevador. The concept and the C implementation use case by tevador, based on a survey of existing w requirements.

- Original Equi-X source repository
- Development log

Equi-X is an asymmetric PoW function based on Equ underlying layer. It features lightning fast verificatio the asymmetry between CPU and GPU. Furthermore case and hence cryptocurrency miners are not incer it.

At this point there is no formal specification for Equ We have two actively maintained implementations c to automated cross-compatibility and fuzz testing:

- A fork of tevador's implementation is maintain

  This is the `src/ext/equix` subdirectory. Curre security, portability, and testability which have implementation is released under the LGPL lic required `--enable-gpl` option this code will b

- As part of Arti, a new Rust re-implementation v original.

  This is the `equix` crate. This implementation c verification performance than the original but

Algorithm overview

The overall scheme consists of several layers that pr functionality:

1. At the lowest layers, Blake2b and siphash are u
that are well suited to common 64-bit CPUs.
2. A custom hash function family, HashX, random
seed value. These functions are tuned to utiliz
on a modern 64-bit CPU. This layer provides th
hardware reimplementation would need to inc
unit to keep up.
3. The Equi-X layer itself builds on HashX and adc
designed to be strongly asymmetric and to rec
4. The PoW protocol itself builds on this Equi-X fu
of the challenge input and particular constrain
solution. This layer provides a linearly adjustab
5. At this point, all further layers are part of the c
individual PoW handshakes, the client and serv
adjusts the effort of future handshakes.

Equi-X itself provides two functions that will be usec

- `equix_solve` (`challenge`) which solves a puzz
number of solutions per invocation depending
- `equix_verify` (`challenge` , `solution`) which v
Verification still depends on executing the Has
when searching for a solution.

For the purposes of this proposal, all cryptographic
and consume byte strings, even if internally they op
bit words. This is conventionally little endian order f
typical use of big endian. HashX itself is configured v
single 64-bit word of undefined byte order, of which
Equi-X in its solution output. We treat Equi-X solutio
packed little endian 16-bit representation.

Linear effort adjustment

The underlying Equi-X puzzle has an approximately
effort comes from the construction of the overlying
to test a variable number of Equi-X solutions in orde
this layer's effort constraint.

It's common for proof-of-work systems to define an
a particular number of leading zero bits or equivale
system, it's quite useful if we have a linear scale inst
hashed version of the Equi-X solution as a uniformly

Conceptually we could define a function:

```
unsigned effort(uint8_t *token)
```

which takes as its argument a hashed solution, inter
quotient of dividing a bitstring of 1s by it.

So for example:

```
effort(00000001100010101101) = 11111111111111
                                    / 0000000110
```

or the same in decimal:

```
effort(6317) = 1048575 / 6317 = 165.
```

In practice we can avoid even having to perform this multiply instead to see if a request's claimed effort i resulting 32-bit hash prefix. This assumes we send t each PoW solution. We do want to force clients to pi a solution, otherwise a client could opportunistically lucky hash prefix comes up. Thus the effort is comm and it forms part of the concatenated Equi-X challen

Parameter descriptor

This whole protocol starts with the service encoding within the 'encrypted' (inner) part of the v3 descript describes it canonically. The parameters offered are

- `type` , always `v1` for the algorithm described l
- `seed-b64` , a periodically updated 32-byte ranc
- `suggested-effort` , the latest output from the
- `expiration-time` , a timestamp when we plan

Seed expiration and rotation allows used nonces to At every seed rotation, a new expiration time is chos recommended range:

- At the earliest, 105 minutes in the future
- At the latest, 2 hours in the future (15 minutes

The service SHOULD refresh its seed when expiratic

keep its previous seed in memory and accept PoWs
clients that have an old seed. The service SHOULD a
seeds that have a common 4 bytes prefix; see the u
introduction extension.

Client computes a solution

If a client receives a descriptor with `pow-params`, it s
prepared to receive PoW solutions as part of the int

The client parses the descriptor and extracts the Po
`expiration-time` has not expired. If it has, the desc
SHOULD fetch a fresh descriptor if the descriptor is

Inputs to the solver:

1. Effort `E`, the client-side effort choice made ba
   `effort` and the client's connection attempt his
2. Constant personalization string `P`, equal to th
   `"Tor hs intro v1\0"`.
3. Identity string `ID`, a 32-byte value unique to th
   blinded public ID key `KP_hs_blind_id`.
4. Seed `C`, a 32-byte random value decoded from
5. Initial nonce `N`, a 16-byte value generated usin

The solver itself is iterative; the following steps are r

1. Construct the *challenge string* by concatenating

2. Calculate a candidate proof `S` by passing this

   ```
   S = equix_solve(P || ID || C || N || htor
   ```

3. Calculate a 32-bit check value by interpreting a
   concatenated challenge and solution as an inte

   ```
   R = ntohl(blake2b_32(P || ID || C || N |
   ```

4. Check if 32-bit multiplication of `R * E` would c

   If `R * E` overflows (the result would be greate
   retry with another nonce value. The client inte
   integer, increments it by 1, and goes back to st

   If there is no overflow (the result is less than o

solution. The client can submit final nonce `N`,
and proof `S`.

Note that the Blake2b hash includes the output leng
so a `blake2b_32` is not equivalent to the prefix of a
Blake2b specifically, and interpret it in network byte

At the end of the above procedure, the client shoulc
nonce `N` that satisfies both the Equi-X proof conditi
time taken, on average, is linearly proportional with

The algorithm as described is suitable for single-thr
client may choose multiple nonces and attempt sev
CPU cores. The specific choice of nonce is entirely u
choices like this do not impact the network protocol

Client sends its proof in an INTR

Now that the client has an answer to the puzzle it's
cell. To do so the client adds an extension to the en
cell by using the EXTENSIONS field. The encrypted p
gets read by the onion service and is ignored by the

This extension includes the chosen nonce and effor
proof. Clients provide only the first 4 bytes of the se
multiple recent seeds offered by the service.

This format is defined canonically as the proof-of-w

Service verifies PoW and handle:

When a service receives an INTRODUCE1 with the `P`
check its configuration on whether proof-of-work is
enabled, the extension SHOULD BE ignored. If enab
currently zero, the service follows the procedure de

If the service requires the `PROOF_OF_WORK` extension
without any embedded proof-of-work, the service S
effort introduction for the purposes of the priority c

To verify the client's proof-of-work the service MUST

1. Find a valid seed `C` that starts with `POW_SEED`.
2. Fail if `N = POW_NONCE` is present in the replay p
3. Construct the *challenge string* as above by conc `htonl(E)`. In this case, `E` and `N` are values pr
4. Calculate `R = ntohl(blake2b_32(P || ID ||` above
5. Fail if the the effort test overflows (`R * E > UI`
6. Fail if Equi-X reports that the proof `S` is malfor (`equix_verify(P || ID || C || N || htonl(`
7. If both the Blake2b and Equi-X tests pass, the r `E`.

It's a minor performance optimization for services t invoking `equix_verify`. Blake2b verification is chea ordering slightly raises the minimum effort requirec

If any of these steps fail the service MUST ignore thi protocol.

In this document we call the above steps the "top ha steps of the "top half" have passed, then the circuit

Analysis and discussion

Warning: Take all the PoW performance numbers or
Most of this is based on very early analysis that has
state of implementation.

For current performance numbers on a specific piec
`bench` from the `equix/bench` crate within Arti. This
implementations side-by-side.

Attacker strategies

To design a protocol and choose its parameters, we
level attacker strategies to see what we are fighting

Overwhelm PoW verification ("top half")

A basic attack here is the adversary spamming with
does not have computing capacity to even verify the
overwhelm the procedure in the `v1` verification alg

That's why we need the PoW algorithm to have a ch
attack is not possible: we explore this PoW paramet
verification.

Overwhelm rendezvous capacity ("bottom

Given the way the introduction queue works, a very
to totally overwhelm the queue processing by sendi
than the onion service can handle at any given tick.
process of handling queued introductions.

To do so, the attacker would have to send at least 2(
100ms, where high-effort is a PoW which is above th
user".

An easier attack for the adversary, is the same strat
all above the comfortable level of "the standard use
users and only allow motivated users to pass.

Hybrid overwhelm strategy

If both the top- and bottom- halves are processed b
possibility for a "hybrid" attack. Given the performa
ms/req.) and the top half (5.5 ms/req.), the attacker
submitting 91 high-effort requests and 1520 invalid
completely saturate the main loop because:

```
0.31*(1520+91) ~ 0.5 sec.
5.5*91         ~ 0.5 sec.
```

This attack only has half the bandwidth requiremen
compute requirement of a bottom-half attack..

Alternatively, the attacker can adjust the ratio betw
depending on their bandwidth and compute capabi

Gaming the effort control logic

Another way to beat this system is for the attacker t
Essentially, there are two attacks that we are trying

- Attacker sets descriptor suggested-effort to a v
  impossible for most clients to produce a PoW t
- Attacker sets descriptor suggested-effort to a v
  aim for a small value while the attacker comfo
  using medium effort PoW (see this post by tev

Precomputed PoW attack

The attacker may precompute many valid PoW non
before the current seed expires, overwhelming the
single computer. The current scheme gives the atta
since each seed lasts 2 hours and the service cache

An attacker with this attack might be aiming to DoS
time, or to cause an effort control attack.

Parameter tuning

There are various parameters in this PoW system th

We first start by tuning the time it takes to verify a P
fundamental to the performance of onion services a
own. We will do this tuning in a way that's agnostic t

We previously considered the concept of a nonzero
still references such a concept, even though the cur
uses a starting effort of zero. (We now expect early i
be driven primarily by client retry behavior.)

At the end of this section we will estimate the resou
overwhelm the onion service, the resources that the
requests, and the resources that legitimate clients n

PoW verification

Verifying a PoW token is the first thing that a service
INTRODUCE2 cell. Our current implementation is de
algorithm specification.

Verification time is a critical performance parameter
`cargo bench` now, and the verification speeds we a
microseconds. The specific numbers below are date
preserved as an illustration of the design space we a

To defend against a top-half attack it's important th
in-between receiving an introduction request over t
effort-prioritized queue.

All time spent verifying PoW adds overhead to the a
handling an introduction cell. Hence we should be c

During our performance measurements on tor we l
0.26 msecs in average, without doing any sort of Po
compute the following table, that describes the num
(aka times we can perform the "top half" process) fo
time:

PoW Verification Time	Total "top half" ti
0 msec	0.26 msec
1 msec	1.26 msec
2 msec	2.26 msec
3 msec	3.26 msec
4 msec	4.26 msec

PoW Verification Time	Total "top half" ti
5 msec	5.26 msec
6 msec	6.26 msec
7 msec	7.26 msec
8 msec	8.26 msec
9 msec	9.26 msec
10 msec	10.26 msec

Here is how you can read the table above:

- For a PoW function with a 1ms verification tim dummy introduction cells per second to succe
- For a PoW function with a 2ms verification tim dummy introduction cells per second to succe
- For a PoW function with a 10ms verification tin dummy introduction cells per second to succe

Whether an attacker can succeed at that depends o the network's capacity.

Our purpose here is to have the smallest PoW verifi allows us to achieve all our other goals.

Note that the table above is simply the result of a na into account all the auxiliary overheads that happen the mainloop, the bottom-half processes, or pretty r half" processing.

During our measurements the time to handle INTRO action time: There might be events that require a lo pretty infrequent (like uploading a new HS descripto smooth out. Hence extrapolating the total cells queu half" time seems like good enough to get some initia "Cells queued per second" from the table above, are above because of all the auxiliary overheads.

PoW difficulty analysis

The difficulty setting of our PoW basically dictates h success in our PoW system. An attacker who can get a successful bottom-half attack against our system.

In classic PoW systems, "success" is defined as gettin

However, since our system is dynamic, we define "su computation.

The original analysis here concluded that we still ne be used for bootstrapping the system. The client an but for UX purposes and for analysis purposes it wa to minimize retries by clients.

In current use it was found that an effort of 1 makes normally have a concept of minimum effort. Consid now to simply be the expected runtime of one singl

Analysis based on adversary power

In this section we will try to do an analysis of PoW di related or PoW-related benchmark numbers.

We created the table (see `[REF_TABLE]`) below whic client with a single machine should expect to burn b

The x-axis is how many successes we want the attac more successes we allow the adversary, the more th queue. The y-axis is how many machines the advers just 5 to 1000.

```
      =========================================
      |    Expected Time (in seconds) Per Su
===============================================
|
|   Attacker Succeses        1       5
|       per second
|
|           5                5       1
|           50               50      10
|           100              100     20
| Attacker  200              200     40
|  Boxes    300              300     60
|           400              400     80
|           500              500     100
|           1000             1000    200
|
===============================================
```

Here is how you can read the table above:

- If an adversary has a botnet with 1000 boxes, a per second, then a legitimate client with a sing 1000 seconds getting a single success.

- If an adversary has a botnet with 1000 boxes, a
successes per second, then a legitimate client
to spend 200 seconds getting a single success.
- If an adversary has a botnet with 500 boxes, an
per second, then a legitimate client with a sing
100 seconds getting a single success.
- If an adversary has access to 50 boxes, and we
second, then a legitimate client with a single b
seconds getting a single success.
- If an adversary has access to 5 boxes, and we
second, then a legitimate client with a single b
seconds getting a single success.

With the above table we can create some profiles fo

Analysis based on Tor's performance

To go deeper here, we can use the performance me
specific intuition on the starting difficulty. In particul
handling an introduction cell takes 5.55 msecs in av
compute the following table, that describes the num
handle per second for different values of PoW verifi

PoW Verification Time	Total time to han introduction ce
0 msec	5.55 msec
1 msec	6.55 msec
2 msec	7.55 msec
3 msec	8.55 msec
4 msec	9.55 mesc
5 msec	10.55 msec
6 msec	11.55 msec
7 msec	12.55 msec
8 msec	13.55 msec
9 msec	14.55 msec
10 msec	15.55 msec

Here is how you can read the table above:

- For a PoW function with a 1ms verification tim
effort introduction cells per second to succeed

- For a PoW function with a 10ms verification tin
  effort introduction cells per second to succeed

We can use this table to specify a starting difficulty t
to succeed in an bottom-half attack attack.

Note that in practice verification times are much lov
not match the current implementation's reality.

User experience

This proposal has user facing UX consequences.

When the client first attempts a pow, it can note hov
take, and then use this to determine an estimation c
estimation could be communicated via the control p
the browser could display how long the PoW is expe
device is a mobile platform, and this time estimatior
user try from a desktop machine.

Future work

Incremental improvements to this propos

There are various improvements that can be done in
trying to keep this `v1` version simple, we need to ke
build more features into it. In particular:

- End-to-end introduction ACKs

  This proposal suffers from various UX issues b
  mechanism for an onion service to inform the
  If we had end-to-end introduction ACKs many
  effort estimation would be alleviated. The prol
  require modifications on the introduction poin
  a lengthy process.

- Multithreading scheduler

  Our scheduler is pretty limited by the fact that
  we improve our multithreading support we co

introduction requests per second.

Future designs

This is just the beginning in DoS defences for Tor an
and schemes that we can investigate. Here is a brief

- "More advanced PoW schemes" -- We could us schemes like MTP-argon2 or Itsuku to make it successful PoWs. Unfortunately these scheme: they won't fit in INTRODUCE1 cells. See #3122:
- "Third-party anonymous credentials" -- We car third-party token issuance server on the clearr CAPTCHA and then use those tokens to get acc for more details.
- "PoW + Anonymous Credentials" -- We can ma we present a hard puzzle to the user when cor they solve it we then give the user a bunch of a the future. This can all happen between the cli a third party.

All of the above approaches are much more complic we want to start easy before we get into more serio implementation requires complexity within the Equi the overall tor network can be relatively simple.

Environment

This algorithm shares a broad concept, proof of wor hungry and wasteful software. We love the environr how proof of work schemes typically waste huge an hash iterations.

Nevertheless, there are some massive differences ir what we are doing here: we are performing fairly sn used as part of a scheme to disincentivize attacks er stop computing these proof-of-work functions entir

We think we aren't making a bad situation worse: D already happening and attackers are already burnin in the denial-of-service overview, attacks on onion s

downstream resource consumption of nearly every
increased CPU load from the circuit floods they proc
clients to carry out PoW computations doesn't affec
attacker right now can very quickly use the same res
clients do in a whole day.

We hope to make things better: The hope is that sys
actors go away and hence the PoW system will not b
there will be a waste of energy, but if we manage to
means, the network as a whole will less wasteful. Als

Acknowledgements

Thanks a lot to tevador for the various improvemen
understand and tweak the RandomX scheme.

Thanks to Solar Designer for the help in understand
various approaches we could take, and teaching us

Scheduler implementation for C

This section describes how we will implement this p
tor).

The following should be read as if tor is an onion se
inbound data.

The Main Loop

Tor uses libevent for its mainloop. For network I/O c
to inform tor if it can read on a certain socket, or a c

From there, this event will empty the connection inp
processing a cell at a time. The mainloop is single th
sequentially.

Processing an INTRODUCE2 cell at the onion service
order):

1. Unpack cell from inbuf to local buffer.
2. Decrypt cell (AES operations).

3. Parse cell header and process it depending on
4. INTRODUCE2 cell handling which means buildi
   - Path selection
   - Launch circuit to first hop.
5. Return to mainloop event which essentially me

Tor will read at most 32 cells out of the inbuf per ma

Requirements for PoW

With this proposal, in order to prioritize cells by the
cells can *not* be processed sequentially as described

Thus, we need a way to queue a certain number of c
some cell(s) from the top of the queue (that is, the c
effort).

We thus require a new cell processing flow that is *n*
The elements are:

- Validate PoW and place cells in a priority queu
  introduction queue).
- Defer "bottom half" INTRO2 cell processing for
  priority queue.

Proposed scheduler

The intuitive way to address the queueing requirem
and naive approach:

1. Mainloop: Empty inbuf INTRODUCE2 cells into
2. Process all cells in pqueue
3. Goto (1)

However, we are worried that handling all those cell
opens possibilities of attack by an adversary since th
up to date while we process all those cells. This mea
dealing with introductions that don't deserve it.

We thus propose to split the INTRODUCE2 handling
and "bottom half" process.

Top half and bottom half

The top half process is responsible for queuing intro
follows:

1. Unpack cell from inbuf to local buffer.
2. Decrypt cell (AES operations).
3. Parse INTRODUCE2 cell header and validate Po
4. Return to mainloop event which essentially me

The top-half basically does all operations from the r

An then, the bottom-half process is responsible for
rendezvous. To achieve this we introduce a new ma
queue *after* the top-half event has completed. This r
sequentially:

1. Pop INTRODUCE2 cell from priority queue.
2. Parse and process INTRODUCE2 cell.
3. End event and yield back to mainloop.

Scheduling the bottom half process

The question now becomes: when should the "botto
mainloop?

We propose that this event is scheduled in when the
cell into the priority queue. Then, as long as it has a
itself for immediate execution meaning at the next r
again.

The idea is to try to empty the queue as fast as it ca
time to an introduction request but always leave a c
between cell processing by yielding back to the main
always have the most up-to-date version of the prio
introductions: this way we are prioritizing clients tha
completing their PoW.

If the size of the queue drops to 0, it stops schedulin
loop. The network I/O event will re-schedule it in tim

Notice that the proposed solution will make the serv
request at every main loop event. However, when w
might learn that it's preferable to bump the number
where N <= 32.

Performance measurements

This section will detail the performance measureme
handling an INTRODUCE2 cell and then a discussion
add (for PoW validation) before it badly degrades ou

Tor measurements

In this section we will derive measurement numbers
parts of handling an introduction cell.

These measurements have been done on tor.git at c
`80031db32abebaf4d0a91c01db258fcdbd54a471`.

We've measured several set of actions of the INTRO
Intel(R) Xeon(R) CPU E5-2650 v4. Our service was acc
introduction requests for a period of 60 seconds.

1. Full Mainloop Event

   We start by measuring the full time it takes for
   containing INTRODUCE2 cells. The mainloop ev
   invocation on average during our measuremen

   ```
   Total measurements: 3279

     Min: 0.30 msec - 1st Q.: 5.47 msec -
     Mean: 13.43 msec - 3rd Q.: 16.20 mse
   ```

2. INTRODUCE2 cell processing (bottom-half)

   We also measured how much time the "bottor
   That's the heavy part of processing an introdu
   main loop section above:

   ```
   Total measurements: 7931

     Min: 0.28 msec - 1st Q.: 5.06 msec -
     Mean: 5.29 msec - 3rd Q.: 5.57 msec
   ```

3. Connection data read (top half)

   Now that we have the above pieces, we can us
   part of the procedure. That's when bytes are ta

buffer and parsed into an INTRODUCE2 cell wh

There is an average of 2.42 INTRODUCE2 cells
that by the full mainloop event mean time to g
subtract the "bottom half" mean time to get ho

```
=> 13.43 / (7931 / 3279) = 5.55
=> 5.55 - 5.29 = 0.26

Mean: 0.26 msec
```

To summarize, during our measurements the avera
mainloop event processed is ~2.42 cells (7931 cells f

This means that, taking the mean of mainloop event
(13.43/2.42) to completely process an INTRODUCE2
that the "top half" of INTRODUCE2 cell processing ta
the "bottom half" takes around 5.33 msec.

The heavyness of the "bottom half" is to be expecte
work takes place: in particular the rendezvous path

References


```
    [REF_EQUIX]: https://github.com/tevador/e
                 https://github.com/tevador/e
    [REF_TABLE]: The table is based on the sc
editing for readability:
                 https://gist.github.com/asn-
d6/99a936b0467b0cef88a677baaf0bbd04
    [REF_BOTNET]: https://media.kasperskycont
/sites/43/2009/07/01121538/ynam_botnets_0907_
    [REF_CREDS]: https://lists.torproject.org
March/014198.html
    [REF_TARGET]: https://en.bitcoin.it/wiki/
    [REF_TEVADOR_2]: https://lists.torproject
June/014358.html
    [REF_TEVADOR_SIM]: https://github.com/mik
/blob/master/tor-pow/effort_sim.py#L57
```

BridgeDB specification

This document specifies how BridgeDB processes bı
new bridges, maintains persistent assignments of b
which bridges to give out upon user requests.

Some of the decisions here may be suboptimal: this
behavior as of August 2013, not to specify ideal beh

Importing bridge network status descriptors

BridgeDB learns about bridges by parsing bridge ne
and extra info documents as specified in Tor's direc
bridge network status file first and at least one brid
extra info file afterwards.

BridgeDB scans its files on sighup.

BridgeDB does not validate signatures on descriptor
needs to make sure that these documents have con
validation for us.

Parsing bridge network statuses

Bridge network status documents contain the inforn
the bridge authority and which flags the bridge auth
bridge network statuses to contain at least the follo
given order (format fully specified in Tor's directory

```
"r" SP nickname SP identity SP digest S
    SP DirPort NL
"a" SP address ":" port NL (no more tha
"s" SP Flags NL
```

BridgeDB parses the identity and the publication tin
address(es) and ORPort(s) from the "a" line(s), and t
specifically checking the assignment of the "Running
memorizes all bridges that have the Running flag as
be given out to bridge users. BridgeDB memorizes a
sets of bridges given out should contain at least a gi

flags.

Parsing bridge descriptors

BridgeDB learns about a bridge's most recent IP add
descriptors. In theory, both IP address and OR port
"r" line of the bridge network status, so there is no r
descriptors. But the functionality described in this s
need data from the bridge descriptor in the future.

Bridge descriptor files may contain one or more brid
descriptor to contain at least the following lines in t

```
"@purpose" SP purpose NL
"router" SP nickname SP IP SP ORPort SP
"published" SP timestamp
["opt" SP] "fingerprint" SP fingerprint
"router-signature" NL Signature NL
```

BridgeDB parses the purpose, IP, ORPort, nickname
BridgeDB skips bridge descriptors if the fingerprint i
status parsed earlier or if the bridge does not have t
bridge descriptors which have a different purpose t
configured to only accept descriptors with another p
based on purpose at all. BridgeDB memorizes the IP
remaining bridges. If there is more than one bridge
BridgeDB memorizes the IP address and OR port of
descriptor. If BridgeDB does not find a bridge descri
bridge network status parsed before, it does not ad
be given out to bridge users.

Parsing extra-info documents

BridgeDB learns if a bridge supports a pluggable tra
documents. Extra-info documents contain the name
the bridge's fingerprint, the type of pluggable transp
and port number on which each transport listens, re

Extra-info documents may contain zero or more ent
info entry to contain the following lines in the statec

```
"extra-info" SP name SP fingerprint NL
"transport" SP transport SP IP ":" PORT
```

BridgeDB parses the fingerprint, transport type, IP a
are specified on these lines. BridgeDB skips the nam
BridgeDB skips the entry. BridgeDB memorizes the
number, and any arguments that are be provided a
corresponding bridge based on the fingerprint. Argu
of the form k=v,k=v. Bridges that do not have an ass
invalid.

Assigning bridges to distributors

A "distributor" is a mechanism by which bridges are
current distributors are "email", "https", and "unallo

BridgeDB assigns bridges to distributors based on a
secret and makes these assignments persistent. Per
database to map node ID to distributor. Each bridge
(including the "unallocated" distributor). BridgeDB m
non-empty subset of the distributors specified in thi
configured to use different probabilities for assignin
BridgeDB does not change existing assignments of b
probabilities for assigning bridges to distributors ch
entirely.

Giving out bridges upon request

Upon receiving a client request, a BridgeDB distribu
assigned to it. BridgeDB only gives out bridges that
parsed bridge network status and that have the Run
may be configured to give out a different number of
the distributor. BridgeDB may define an arbitrary nu
specify the criteria by which a bridge is selected. Spe
the IP address version, OR port number, transport t
which the bridge should not be blocked.

Selecting bridges to be given out

```
   BridgeDB may be configured to support one
   gives out bridges based on the requestor's
   is how the HTTPS distributor works.
   The goal is to avoid handing out all the b
   IP space and time.

> Someone else should look at proposals/ideas
> to see if this section is missing relevant

   BridgeDB fixes the set of bridges to be re
   period.
   BridgeDB considers all IP addresses coming
   as the same IP address and returns the sam
   this non-unique address will be referred t
   BridgeDB divides the IP address space equa

> Note, changed term from "areas" to "disjoin

   disjoint clusters (typically 4) and return
   coming from addresses that are placed into

> I found that BridgeDB is not strict in retu
> given area.  If a ring is empty, it conside
> expected behavior?  -KL
>
> This does not appear to be the case, anymor
> BridgeDB simply returns an empty set of bri
>
> I also found that BridgeDB does not make th
> persistent in the database.  So, if we chan
> will assign bridges to other rings.  I assu

   BridgeDB maintains a list of proxy IP addr
   set of bridges to requests coming from the
   The bridges returned to proxy IP addresses
   set as those for the general IP address sp
```

BridgeDB can be configured to include bridge finger
addresses and OR ports. BridgeDB can be configure
user must solve prior to returning the requested bri

The current algorithm is as follows. An IP-based dist
into a set of "rings" based on an HMAC of their ID. S
parts of IP space; some are "category" rings for cate
client makes a request from an IP, the distributor fir
categories it knows. If so, the distributor returns an
distributor maps the IP into an "area" (that is, a /24),
area to one of the area rings.

When the IP-based distributor determines from whi
it identifies which rules it will use to choose appropr
searches its cache of rings for one that already adhe

request. If one exists, then BridgeDB maps the curr
IP's area (/24) to a point on the ring based on HMAC
If a ring does not already exist which satisfies this re
filled with bridges that fulfill the requirements. This
described.

"Mapping X to Y based on an HMAC" above means c

```
- We keep all of the elements of Y in s
  from all 160-bit strings to positions
- We take an HMAC of X using some fixed
  160-bit value.  We then map that valu
```

When giving out bridges based on a position in a rin
requirements and port requirements. For example,
out at least L bridges with port 443, and at least M b
bridges total." To do this, BridgeDB combines to the

```
- The first L bridges in the ring after
  port 443, and
- The first M bridges in the ring after
  flag stable and that it has not alrea
- The first N-L-M bridges in the ring a
  has not already decided to give out.

After BridgeDB selects appropriate bridge
then prioritises the ordering of them in
are fulfilled as possible within the firs
truncated to N bridges, if possible. N is
piecewise function of the number of bridg

          /
         |  1,   if len(ring) < 20
         |
     N = |  2,   if 20 <= len(ring) <= 10(
         |
         |  3,   if 100 <= len(ring)
          \

The bridges in this sublist, containing r
bridges returned to the requestor.
```

Selecting bridges to be given out addresses

```
   BridgeDB can be configured to support one
   giving out bridges based on the requestor'
   this is how the email distributor works.
   The goal is to bootstrap based on one or m
   sybil prevention algorithms.

> Someone else should look at proposals/ideas
> to see if this section is missing relevant

   BridgeDB rejects email addresses containin
   ones that RFC2822 allows.
   BridgeDB may be configured to reject email
   characters it might not process correctly.

> I don't think we do this, is it worthwhile?

   BridgeDB rejects email addresses coming fr
   configured set of permitted domains.
   BridgeDB normalizes email addresses by rem
   removing parts after the first "+" charact
   BridgeDB can be configured to discard requ
   value "pass" in their X-DKIM-Authenticatio
   have this header.  The X-DKIM-Authenticati
   the incoming mail stack that needs to chec

   BridgeDB does not return a new set of brid
   until a given time period (typically a few

> Why don't we fix the bridges we give out fo
> like we do for IP addresses?  This way we c
> addresses.  -KL
>
> The 3-hour value is probably much too short
> time values, then people get new bridges wh
> opposed to then we decide to reset the brid
> problem exists for the IP distributor). -NM
>
> I'm afraid I don't fully understand what yo
> elaborate?  -KL
>
> Assuming an average churn rate, if we use s
> requestor will receive new bridges based on
> eventually work their way around the ring;
bridges
> available to them from this distributor. If
> then each time the period expires there wil
> thus reducing the likelihood of all bridges
> the time and effort required to enumerate a
> understanding, not from Nick) -MF
>
> Also, we presently need the cache to preven
> sent multiple requests with different crite
> additional bridges otherwise. -MF

   BridgeDB can be configured to include brid
   along with bridge IP addresses and OR port
```

```
BridgeDB can be configured to sign all rep
BridgeDB periodically discards old email-a
BridgeDB rejects too frequent email reques
normalized address.
```

To map previously unseen email addresses to a set
follows:

```
  - It normalizes the email address as abo
    removing all of the localpart after th
    in lowercase.  (Example: "John.Doe+bri
    "johndoe@example.com".)
  - It maps an HMAC of the normalized addr
    of bridges.
  - It hands out bridges starting at that
    port/flag requirements, as specified a

 See section 4 for the details of how brid
 and returned to the requestor.
```

Selecting unallocated bridges to buckets

> Kaner should have a look at this section. -NM

```
BridgeDB can be configured to reserve a su
them out via one of the distributors.
BridgeDB assigns reserved bridges to one o
sizes and write these file buckets to disk
BridgeDB ensures that a file bucket always
number of running bridges.
If the requested number of bridges in a fi
file bucket is not required anymore, the u
returned to the reserved set of bridges.
If a bridge stops running, BridgeDB replac
from the reserved set of bridges.

> I'm not sure if there's a design bug in fil
> we add a bridge X to file bucket A, and X g
> another bridge Y to file bucket A.  OK, but
> cannot put it back in file bucket A, becaus
> add it to a different file bucket?  Doesn't
> will be contained in most file buckets over
>
> This should be handled the same as if the f
> If X returns, then it should be added to th
```

Displaying Bridge Information

After bridges are selected using one of the methods
output in one of two formats. Bridges are formattec

```
<address:port> NL
```

Pluggable transports are formatted as:

```
<transportname> SP <address:port> [SP arglist
```

where arglist is an optional space-separated list of k

Previously, each line was prepended with the "bridg

```
"bridge" SP <address:port> NL
```

```
"bridge" SP <transportname> SP <address:port>
```

> We don't do this anymore because Vidalia and Tc
> commit message for b70347a9c5fd769c6d5d0c0

Writing bridge assignments for s

BridgeDB can be configured to write bridge assignm
The start of a bridge assignment is marked by the fc

"bridge-pool-assignment" SP YYYY-MM-DD HH:MM:S

YYYY-MM-DD HH:MM:SS is the time, in UTC, when B
bridges and assigning them to distributors.

For every running bridge there is a line with the follc

fingerprint SP distributor (SP key "=" value)* NL

The distributor is one out of "email", "https", or "una

Both "email" and "https" distributors support addin
"transport". Respectively, the port number, flag nam
These are used to indicate that a bridge matches ce
requests.

The "https" distributor also allows the key "ring" witl which IP address area the bridge is returned.

The "unallocated" distributor allows the key "bucket
indicate which file bucket a bridge is assigned to.

Extended ORPort for plu transports

George Kadianakis, Nick Mathew

Overview

This document describes the "Extended ORPort" pr ordinary ORPort protocol for use by bridges that su provides a way for server-side PTs and bridges to ex beginning the actual OR connection.

See `tor-spec.txt` for information on the regular O information on pluggable transports.

This protocol was originally proposed in proposal 1 in proposal 217.

Establishing a connection and a

When a client (that is to say, a server-side pluggable ORPort, the server sends:

```
    AuthTypes
    EndAuthTypes

  Where,

  + AuthTypes are the authentication schemes
    for this session. They are multiple conc
    take values from 1 to 255.
  + EndAuthTypes is the special value 0.
```

The client reads the list of supported authentication back:

AuthType [1 octet]

Where,

```
+ AuthType is the authentication scheme tha
  for this session. A valid authentication 
  255. A value of 0 means that the client c
  authentication types offered by the serve
```

If the client sent an AuthType of value 0, or an AuthT
the server MUST close the connection.

Authentication type: SAFE_COOKIE

We define one authentication type: SAFE_COOKIE. It
the client proving to the bridge that it can access a g
purpose of authentication is to defend against cross

If the Extended ORPort is enabled, Tor should regen
store it in $DataDirectory/extended_orport_auth_co

The location of the cookie can be overridden by usir
ExtORPortCookieAuthFile, which is defined as:

```
ExtORPortCookieAuthFile <path>
```

where `<path>` is a filesystem path.

Cookie-file format

The format of the cookie-file is:

```
     StaticHeader
     Cookie

   Where,
   + StaticHeader is the following string:
     "! Extended ORPort Auth Cookie !\x0a"
   + Cookie is the shared-secret. During the S
     cookie is called CookieString.
```

Extended ORPort clients MUST make sure that the S
file, before proceeding with the authentication proto

SAFE_COOKIE Protocol specification

A client that performs the SAFE_COOKIE handshake

```
ClientNonce
```

Where,

- ClientNonce is 32 octets of random data.

Then, the server replies with:

```
     ServerHash
     ServerNonce

  Where,
  + ServerHash is computed as:
      HMAC-SHA256(CookieString,
        "ExtORPort authentication server-to-c
ServerNonce)
  + ServerNonce is 32 random octets.
```

Upon receiving that data, the client computes Serve
ServerHash provided by the server.

If the server-provided ServerHash is invalid, the clie

Otherwise the client replies with:

```
ClientHash                                  |

  Where,
  + ClientHash is computed as:
      HMAC-SHA256(CookieString,
        "ExtORPort authentication client-to-s
ServerNonce)
```

Upon receiving that data, the server computes Clien
ClientHash provided by the client.

Finally, the server replies with:

Status [1 octet]

```
  Where,
  + Status is 1 if the authentication was suc
    authentication failed, Status is 0.
```

The extended ORPort protocol

Once a connection is established and authenticated
protocol described here.

Protocol

The extended server port protocol is as follows:

```
    COMMAND [2 bytes, big-endian]
    BODYLEN [2 bytes, big-endian]
    BODY [BODYLEN bytes]

    Commands sent from the transport proxy t

    [0x0000] DONE: There is no more informat
      bytes sent by the transport will be th
      (body ignored)

    [0x0001] USERADDR: an address:port strin
      client's address.

    [0x0002] TRANSPORT: a string of the name
      transport currently in effect on the c

    Replies sent from tor to the proxy are:

    [0x1000] OKAY: Send the user's traffic.

    [0x1001] DENY: Tor would prefer not to g
      this address for a while. (body ignore

    [0x1002] CONTROL: (Not used)

Parties MUST ignore command codes that they
```

If the server receives a recognized command that do connection to the client.

Command descriptions

USERADDR

```
An ASCII string holding the TCP/IP address
pluggable transport proxy. A Tor bridge SHO
collect statistics about its clients.  Reco
  1.2.3.4:5678
  [1:2::3:4]:5678
```

(Current Tor versions may accept other formats, but send them.)

The string MUST not be NUL-terminated.

TRANSPORT

An ASCII string holding the name of the pluggable tr
pluggable transport proxy. A Tor bridge that suppor
that information to collect statistics about the popul
transports.

The string MUST not be NUL-terminated.

Pluggable transport names are C-identifiers and Tor

Security Considerations

Extended ORPort or TransportControlPort do *not* pr
authentication or integrity. Sensitive data, like crypto
transferred through them.

An attacker with superuser access is able to sniff ne
TransportControlPort identifiers and any data passe

Tor SHOULD issue a warning if the bridge operator t
non-localhost address.

Pluggable transport proxies SHOULD issue a warnin
a non-localhost Extended ORPort.

Pluggable Transport Spe (Version 1)

Abstract

Pluggable Transports (PTs) are a generic mechanism deployment of censorship circumvention, based arc processes that transform traffic to defeat censors.

This document specifies the sub-process startup, sh communication mechanisms required to utilize PTs.

Introduction

This specification describes a way to decouple protc
application's client/server code, in a manner that pr
obfuscation/circumvention tools and promotes reus
Project's efforts in that area.

This is accomplished by utilizing helper sub-process
forward/reverse proxy servers that handle the cens
defined and standardized configuration and manag

Any application code that implements the interfaces
able to use all spec compliant Pluggable Transports.

Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "
"SHOULD NOT", "RECOMMENDED", "MAY", and "OP
interpreted as described in [RFC2119].

Architecture Overview

```
+------------+                    +-----
| Client App +-- Local Loopback --+ PT C
+------------+                    +-----

            Public Internet (Obfuscated/

+------------+                    +-----
| Server App +-- Local Loopback --+ PT S
+------------+                    +-----
```

On the client's host, the PT Client software exposes
application, and obfuscates or otherwise transforms
server's host.

On the server's host, the PT Server software expose
connections from PT Clients, and handles reversing
applied to traffic, before forwarding it to the actual s
lightweight protocol exists to facilitate communicati
otherwise be lost such as the source IP address and

All PT instances are configured by the respective pa
environment variables (3.2) that are set at launch tir
back to the parent via writing output in a standardiz

Each invocation of a PT MUST be either a client OR a

All PT client forward proxies MUST support either SO
prefer SOCKS 5 over SOCKS 4.

Specification

Pluggable Transport proxies follow the following wo

```
1) Parent process sets the required env
   and launches the PT proxy as a sub-pr

2) The PT Proxy determines the versions
   supported by the parent"TOR_PT_MANAGE

   2.1) If there are no compatible vers
        writes a "VERSION-ERROR" message
        terminates.

   2.2) If there is a compatible versio
        a "VERSION" message (3.3.1) to s

3) The PT Proxy parses the rest of the e

   3.1) If the environment values are ma
        invalid, the PT proxy writes a '
        (3.3.1) to stdout and terminates

   3.2) Determining if it is a client s
        a server side reverse proxy can
        the "TOR_PT_CLIENT_TRANSPORTS" a
        environment variables.

4) (Client only) If there is an upstrean
   "TOR_PT_PROXY" (3.2.2), the PT proxy
   provided.

   4.1) If the upstream proxy is unusabl
        a "PROXY-ERROR" message (3.3.2)
        terminates.

   4.2) If there is a supported and well
        the PT proxy writes a "PROXY DON
        stdout.

5) The PT Proxy initializes the transpoi
   status via stdout (3.3.2, 3.3.3)

6) The PT Proxy forwards and transforms

7) Upon being signaled to terminate by t
   the PT Proxy gracefully shuts down.
```

Pluggable Transport Nar

Pluggable Transport names serve as unique identifi
name.

PT names MUST be valid C identifiers. PT names MU
and the remaining characters MUST be ASCII letters
limit is imposted.

PT names MUST satisfy the regular expression " `[a-`

Pluggable Transport Cor Environment Variables

All Pluggable Transport proxy instances are configui time via a set of well defined environment variables

The "TOR_PT_" prefix is used for namespacing reasc relations to Tor, except for the origins of this specifi

Common Environment Variables

When launching either a client or server Pluggable T common environment variables MUST be set.

"TOR_PT_MANAGED_TRANSPORT_VER"

Specifies the versions of the Pluggable Transport sp supports, delimited by commas. All PTs MUST accep compatible version is present.

Valid versions MUST consist entirely of non-whitesp characters.

The version of the Pluggable Transport specification

Example:

TOR_PT_MANAGED_TRANSPORT_VER=1,1a,2b,this_i:

"TOR_PT_STATE_LOCATION"

Specifies an absolute path to a directory where the persisted across invocations. The directory is not re launched, however PT implementations SHOULD be

PTs MUST only store files in the path provided, and elsewhere on the system.

Example:

TOR_PT_STATE_LOCATION=/var/lib/tor/pt_state/

"TOR_PT_EXIT_ON_STDIN_CLOSE"

Specifies that the parent process will close the PT pr
indicate that the PT proxy should gracefully exit.

PTs MUST NOT treat a closed stdin as a signal to ter
variable is set to "1".

PTs SHOULD treat stdin being closed as a signal to g
environment variable is set to "1".

Example:

TOR_PT_EXIT_ON_STDIN_CLOSE=1

"TOR_PT_OUTBOUND_BIND_ADDRESS_V4"

Specifies an IPv4 IP address that the PT proxy SHOU
outgoing IPv4 IP packets. This feature allows people
specify explicitly which interface they prefer the PT p

If this value is unset or empty, the PT proxy MUST us
outgoing connections.

This setting MUST be ignored for connections to loo

Example:

TOR_PT_OUTBOUND_BIND_ADDRESS_V4=203.0.113

"TOR_PT_OUTBOUND_BIND_ADDRESS_V6"

Specifies an IPv6 IP address that the PT proxy SHOU
outgoing IPv6 IP packets. This feature allows people
specify explicitly which interface they prefer the PT p

If this value is unset or empty, the PT proxy MUST us
outgoing connections.

This setting MUST be ignored for connections to the

IPv6 addresses MUST always be wrapped in square

Example::

TOR_PT_OUTBOUND_BIND_ADDRESS_V6=[2001:db8

Pluggable Transport Client Envir

Client-side Pluggable Transport forward proxies are
environment variables.

"TOR_PT_CLIENT_TRANSPORTS"

Specifies the PT protocols the client proxy should in
PT names.

PTs SHOULD ignore PT names that it does not recog

Parent processes MUST set this environment variab
proxy instance.

Example:

TOR_PT_CLIENT_TRANSPORTS=obfs2,obfs3,obfs4

"TOR_PT_PROXY"

Specifies an upstream proxy that the PT MUST use v
connections. It is a URI [RFC3986] of the format:

```
<proxy_type>://[<user_name>[:<password>][@]<i
```

The "TOR_PT_PROXY" environment variable is OPTIC
no need to connect via an upstream proxy.

Examples:

```
        TOR_PT_PROXY=socks5://tor:test123
        TOR_PT_PROXY=socks4a://198.51.100.
        TOR_PT_PROXY=http://198.51.100.3:
```

Pluggable Transport Server Envi

Server-side Pluggable Transport reverse proxies are
environment variables.

"TOR_PT_SERVER_TRANSPORTS"

Specifies the PT protocols the server proxy should in
PT names.

PTs SHOULD ignore PT names that it does not recog

Parent processes MUST set this environment variab
reverse proxy instance.

Example:

TOR_PT_SERVER_TRANSPORTS=obfs3,scramblesuit

"TOR_PT_SERVER_TRANSPORT_OPTIONS"

Specifies per-PT protocol configuration directives, as `<key>:<value>` pairs, where `<key>` is a PT name an options that are to be passed to the transport.

Colons, semicolons, and backslashes MUST be escap

If there are no arguments that need to be passed to "TOR_PT_SERVER_TRANSPORT_OPTIONS" MAY be om

Example:

TOR_PT_SERVER_TRANSPORT_OPTIONS=scramblesu automata:depth=3

```
        Will pass to 'scramblesuit' the para
        'automata' the arguments 'rule=110'

    "TOR_PT_SERVER_BINDADDR"
```

A comma separated list of `<key>-<value>` pairs, wh `<value>` is the `<address>:<port>` on which it shou connections.

The keys holding transport names MUST be in the sa "TOR_PT_SERVER_TRANSPORTS".

The `<address>` MAY be a locally scoped address as externally.

The `<address>:<port>` combination MUST be an IP MUST NOT be a host name.

Applications MUST NOT set more than one `<addres`

If there is no specific `<address>:<port>` combinatio "TOR_PT_SERVER_BINDADDR" MAY be omitted.

Example:

TOR_PT_SERVER_BINDADDR=obfs3-198.51.100.1:19

"TOR_PT_ORPORT"

Specifies the destination that the PT reverse proxy s
transforming it as appropriate, as an `<address>:<p`

Connections to the destination specified via "TOR_P
application payload. If the parent process requires t
connections (or other metadata), it should set "TOR_
instead.

Example:

TOR_PT_ORPORT=127.0.0.1:9001

"TOR_PT_EXTENDED_SERVER_PORT"

Specifies the destination that the PT reverse proxy s
Extended ORPort protocol [EXTORPORT] as an `<add`

The Extended ORPort protocol allows the PT revers
connection metadata such as the PT name and clier
process.

If the parent process does not support the ExtORPo
"TOR_PT_EXTENDED_SERVER_PORT" to an empty str

Example:

TOR_PT_EXTENDED_SERVER_PORT=127.0.0.1:4200

"TOR_PT_AUTH_COOKIE_FILE"

Specifies an absolute filesystem path to the Extende
required to communicate with the Extended ORPort
"TOR_PT_EXTENDED_SERVER_PORT".

If the parent process is not using the ExtORPort pro
"TOR_PT_AUTH_COOKIE_FILE" MUST be omitted.

Example:

TOR_PT_AUTH_COOKIE_FILE=/var/lib/tor/extended_

Pluggable Transport To I Communication

All Pluggable Transport Proxies communicate to the terminated lines to stdout. The line metaformat is:

```
<Line> ::= <Keyword> <OptArgs> <NL>
<Keyword> ::= <KeywordChar> | <Keyword> <Keyw
<KeywordChar> ::= <any US-ASCII alphanumeric,
<OptArgs> ::= <Args>*
<Args> ::= <SP> <ArgChar> | <Args> <ArgChar>
<ArgChar> ::= <any US-ASCII character but NUL
<SP> ::= <US-ASCII whitespace symbol (32)>
<NL> ::= <US-ASCII newline (line feed) charac
```

The parent process MUST ignore lines received fron

Common Messages

When a PT proxy first starts up, it must determine w Transports Specification to use to configure itself.

It does this via the "TOR_PT_MANAGED_TRANSPORT which contains all of the versions supported by the

Upon determining the version to use, or lack thereo two messages.

```
VERSION-ERROR <ErrorMessage>
```

The "VERSION-ERROR" message is used to signal tha Transport Specification version present in the "TOR_

The `<ErrorMessage>` SHOULD be set to "no-version set to a useful error message instead.

PT proxies MUST terminate after outputting a "VERS

Example:

```
VERSION-ERROR no-version
```

```
VERSION <ProtocolVersion>
```

The "VERSION" message is used to signal the Plugga

(as in "TOR_PT_MANAGED_TRANSPORT_VER") that tl
transports and communicate with the parent proce:

The version for the environment values and reply m
"1".

PT proxies MUST either report an error and termina
before moving on to client/server proxy initializatior

Example:

VERSION 1

After version negotiation has been completed the P
the required environment variables are provided, ar
supplied are well formed.

At any point, if there is an error encountered relatec
environment variables, it MAY respond with an erro

```
ENV-ERROR <ErrorMessage>
```

The "ENV-ERROR" message is used to signal the PT p
configuration environment variables (3.2).

The `<ErrorMessage>` SHOULD consist of a useful er
diagnose and correct the root cause of the failure.

PT proxies MUST terminate after outputting a "ENV-

Example:

ENV-ERROR No TOR_PT_AUTH_COOKIE_FILE when T

Pluggable Transport Client Mess

After negotiating the Pluggable Transport Specificat
first validate "TOR_PT_PROXY" (3.2.2) if it is set, befo

Assuming that an upstream proxy is provided, PT cli
message indicating that the proxy is valid, supporte
message.

PROXY DONE

The "PROXY DONE" message is used to signal the PT proxy specified by "TOR_PT_PROXY".

```
PROXY-ERROR <ErrorMessage>
```

The "PROXY-ERROR" message is used to signal that t
malformed/unsupported or otherwise unusable.

PT proxies MUST terminate immediately after outpu

Example:

PROXY-ERROR SOCKS 4 upstream proxies unsuppor

After the upstream proxy (if any) is configured, PT cl
transports in "TOR_PT_CLIENT_TRANSPORTS" and ir

For each transport initialized, the PT proxy reports t
via messages to stdout.

```
CMETHOD <transport> <'socks4','socks5'> <addr
```

The "CMETHOD" message is used to signal that a re
launched, the protocol which the parent should use
the IP address and port that the PT transport's forw

Example:

```
CMETHOD trebuchet socks5 127.0.0.1:19999
```

```
CMETHOD-ERROR <transport> <ErrorMessage>
```

The "CMETHOD-ERROR" message is used to signal tl
unable to be launched.

Example:

CMETHOD-ERROR trebuchet no rocks available

Once all PT transports have been initialized (or have
final message indicating that it has finished initializi

CMETHODS DONE

The "CMETHODS DONE" message signals that the P
the transports that it is capable of handling.

Upon sending the "CMETHODS DONE" message, the

Notes:

```
- Unknown transports in "TOR_PT_CLIENT_TR
  entirely, and MUST NOT result in a "CME
  Thus it is entirely possible for a give
  immediately output "CMETHODS DONE".

- Parent processes MUST handle "CMETHOD",
  messages in any order, regardless of or
  "TOR_PT_CLIENT_TRANSPORTS".
```

Pluggable Transport Server Mess

PT server reverse proxies iterate over the requestec "TOR_PT_CLIENT_TRANSPORTS" and initialize the list

For each transport initialized, the PT proxy reports t via messages to stdout.

```
SMETHOD <transport> <address:port> [options]
```

The "SMETHOD" message is used to signal that a re launched, the protocol which will be used to handle address and port that clients should use to reach th

If there is a specific address:port provided for a give "TOR_PT_SERVER_BINDADDR", the transport MUST b address.

The OPTIONAL 'options' field is used to pass additio the parent process.

The currently recognized 'options' are:

```
ARGS:[<Key>=<Value>,]+[<Key>=<Value>]
```

```
          The "ARGS" option is used to pass
          formatted information that clients
          the reverse proxy.

          Equal signs and commas MUST be esc

          Tor: The ARGS are included in the
          Bridge's extra-info document.

        Examples:

          SMETHOD trebuchet 198.51.100.1:1999
          SMETHOD rot_by_N 198.51.100.1:2323 A

      SMETHOD-ERROR <transport> <ErrorMessage>
```

The "SMETHOD-ERROR" message is used to signal th proxy was unable to be launched.

Example:

SMETHOD-ERROR trebuchet no cows available

Once all PT transports have been initialized (or have final message indicating that it has finished initializi

SMETHODS DONE

The "SMETHODS DONE" message signals that the PT the transports that it is capable of handling.

Upon sending the "SMETHODS DONE" message, the

Pluggable Transport Log Messag

This message is for a client or server PT to be able to stdout or stderr any log messages.

A log message can be any kind of messages (human the parent process can gather information about wh is not intended for the parent process to parse and used for plain logging.

For example, the tor daemon logs those messages a onto the control port using the PT_LOG (see control pick them up for debugging.

The format of the message:

```
LOG SEVERITY=Severity MESSAGE=Message
```

The SEVERITY value indicate at which logging level th
values for `<Severity>` are: error, warning, notice, ir

The MESSAGE value is a human readable string forn
contains the log message which can be a String or C
spec.txt).

Example:

```
LOG SEVERITY=debug MESSAGE="Connected to brid
```

Pluggable Transport Status Mess

This message is for a client or server PT to be able to
stdout or stderr any status messages.

The format of the message:

```
STATUS TRANSPORT=Transport <K_1>=<V_1> [<K_2>
```

The TRANSPORT value indicates a hint on what the I
protocol used for instance. As an example, obfs4prc
Transport value can be anything the PT itself define
section 2 in control-spec.txt).

The `<K_n>=<V_n>` values are specific to the PT and t
messages that reflects the status that the PT wants
CString.

Examples (fictional):

```
STATUS TRANSPORT=obfs4 ADDRESS=198.51.100.123
STATUS TRANSPORT=obfs4 ADDRESS=198.51.100.222
FINGERPRINT=<Fingerprint> ERRSTR="Connection
STATUS TRANSPORT=trebuchet ADDRESS=198.51.100
```

Pluggable Transport Shu

The recommended way for Pluggable Transport usi
Transports to handle graceful shutdown is as follow

```
- (Parent) Set "TOR_PT_EXIT_ON_STDIN_CL
  launching the PT proxy, to indicate th
  for graceful shutdown notification.

- (Parent) When the time comes to termin

  1. Close the PT proxy's stdin.
  2. Wait for a "reasonable" amount of t
  3. Attempt to use OS specific mechanis
     PT shutdown (eg: 'SIGTERM')
  4. Use OS specific mechanisms to force
     (eg: 'SIGKILL', 'ProccessTerminate(

- PT proxies SHOULD monitor stdin, and e
  it is closed, if the parent supports t

- PT proxies SHOULD handle OS specific m
  terminate (eg: Install a signal handle
  causes cleanup and a graceful shutdown

- PT proxies SHOULD attempt to detect wh
  terminated (eg: via detecting that its
  changed on U*IX systems), and graceful
```

Pluggable Transport Clie Connection Arguments

Certain PT transport protocols require that the clien when making outgoing connections. On the server s optional argument as part of the "SMETHOD" messa

On the client side, arguments are passed via the aut SOCKS protocol.

First the " `<Key>=<Value>` " formatted arguments MU backslash, equal sign, and semicolon characters are

Second, all of the escaped are concatenated togeth

Example:

shared-secret=rahasia;secrets-file=/tmp/blob

Lastly the arguments are transmitted when making authentication mechanism specific to the SOCKS pr

```
    - In the case of SOCKS 4, the concatenat
      transmitted in the "USERID" field of th

    - In the case of SOCKS 5, the parent pro
      "Username/Password" authentication [RF
      the arguments encoded in the "UNAME" an
```

If the encoded argument list is less than 255 bytes i to "1" and the "PASSWD" field must contain a single

Anonymity Consideratio

When designing and implementing a Pluggable Trar
preserve the privacy of clients and to avoid leaking p

Examples of client related considerations are:

- Not logging client IP addresses to disk.
- Not leaking DNS addresses except when neces

```
- Ensuring that "TOR_PT_PROXY"'s "fail c
  implemented correctly.
```

Additionally, certain obfuscation mechanisms rely o
address/port being confidential, so clients also neec
information confidential when applicable.

References


```
[RFC2119]     Bradner, S., "Key words for
              Requirement Levels", BCP 14,

[RFC1928]     Leech, M., Ganis, M., Lee, Y
              Koblas, D., Jones, L., "SOCK
              RFC 1928, March 1996.

[EXTORPORT]   Kadianakis, G., Mathewson, N
              TransportControlPort", Tor P

[RFC3986]     Berners-Lee, T., Fielding, R
              Resource Identifier (URI): G
              January 2005.

[RFC1929]     Leech, M., "Username/Passwor
              SOCKS V5", RFC 1929, March 1
```

Acknowledgments

This specification draws heavily from prior versions
Mathewson, and George Kadianakis.

Appendix A: Example Cli Transport Session

Environment variables:

TOR_PT_MANAGED_TRANSPORT_VER=1 TOR_PT_STA
TOR_PT_EXIT_ON_STDIN_CLOSE=1 TOR_PT_PROXY=s
TOR_PT_CLIENT_TRANSPORTS=obfs3,obfs4

Messages the PT Proxy writes to stdin:

VERSION 1 PROXY DONE CMETHOD obfs3 socks5 12
socks5 127.0.0.1:37347 CMETHODS DONE

Appendix B: Example Ser Transport Session

Environment variables:

TOR_PT_MANAGED_TRANSPORT_VER=1 TOR_PT_ST/
TOR_PT_EXIT_ON_STDIN_CLOSE=1 TOR_PT_SERVER_
TOR_PT_SERVER_BINDADDR=obfs3-198.51.100.1:19

Messages the PT Proxy writes to stdin:

VERSION 1 SMETHOD obfs3 198.51.100.1:1984 SME
ARGS:cert=HszPy3vWfjsESCEOo9ZBkRv6zQ/1mGHz
bz3ddFw,iat-mode=0 SMETHODS DONE

TC: A Tor control protoco

Scope

This document describes an implementation-specifi
programs (such as frontend user-interfaces) to com
process. It is not part of the Tor onion routing proto

This protocol replaces version 0 of TC, which is now
described in "control-spec-v0.txt". Implementors are
directly, but instead to use a library that can easily b
(Version 0 is used by Tor versions 0.1.0.x; the protoc
Tor versions in the 0.1.1.x series and later.)

```
The key words "MUST", "MUST NOT", "REQU
NOT", "SHOULD", "SHOULD NOT", "RECOMMEN
"OPTIONAL" in this document are to be 
RFC 2119.
```

Protocol outline

TC is a bidirectional message-based protocol. It assu communication between a controlling process (the ' process (or "server"). The stream may be implemen domain socket, or so on, but it must provide reliable stream should not be accessible by untrusted partie

In TC, the client and server send typed messages to stream. The client sends "commands" and the serve

By default, all messages from the server are in resp Some client requests, however, will cause the server indefinitely far into the future. Such "asynchronous'

Servers respond to messages in the order messages

Forward-compatibility

This is an evolving protocol; new client and server b versions. To allow new backward-compatible behav new commands and allow existing commands to tal To allow new backward-compatible server behavior, servers speaking a future version of this protocol m clients should/must "tolerate" unexpected elements that we do this:

- Adding a new field to a message:

```
  For example, we might say "This message h
  fields; clients MUST tolerate more fields
  client MUST NOT crash or otherwise fail t
  other subsequent messages when there are
  that it SHOULD function at least as well
  provided as it does when it only gets the
  most obvious way to do this is by ignorin
  next-most-obvious way is to report additi
  user, perhaps as part of an expert UI.

* Adding a new possible value to a list of

  For example, we might say "This field wil
  CONNECTED.  Clients MUST tolerate unexpec
  that a client MUST NOT crash or otherwise
  or other subsequent messages when there a
  that it SHOULD try to handle the rest of
  can.  The most obvious way to do this is
  list of alternatives has an additional "u
  and mapping any unrecognized values to th
  next-most-obvious way is to create a sepa
  element for each unrecognized value.

  Clients SHOULD NOT "tolerate" unrecognize
  pretending that the message containing th
  a stream closed for an unrecognized reaso
  and should be reported as such.

  (If some list of alternatives is given, a
  statement that clients must tolerate unex
  must tolerate unexpected values. The only
  were an explicit statement that no future
```

Message format

Description format

The message formats listed below use ABNF as desc
is loosely based on SMTP (see RFC 2821).

We use the following nonterminals from RFC 2822: a

We define the following general-use nonterminals:

QuotedString = DQUOTE *qcontent DQUOTE

There are explicitly no limits on line length. All 8-bit
explicitly disallowed. In QuotedStrings, backslashes
characters need not be escaped.

Wherever CRLF is specified to be accepted from the
Tor, however, MUST NOT generate LF instead of CRL
CRLF.

Notes on an escaping bug

CString = DQUOTE *qcontent DQUOTE

Note that although these nonterminals have the sar
differently. In a QuotedString, a backslash followed
character. But in a CString, the escapes "\n", "\t", "\r'
represent newline, tab, carriage return, and the 256

The use of CString in this document reflects a bug ir
QuotedString instead. In the future, they may migra
do, the QuotedString implementation will never plac
digit, to ensure that old controllers don't get confuse

For future-proofing, controller implementors MAY u
compatible with buggy Tor implementations and wit
spec as intended:

```
Read \n \t \r and \0 ... \377 as C escape
Treat a backslash followed by any other c
```

Currently, many of the QuotedString instances belo

We intend to fix this in future versions of Tor, and d
(See bugtracker ticket #14555 for a bit more inform

Note that this bug exists only in strings generated b
should parse input QuotedStrings from the controll

Commands from controller to To

```
    Command = Keyword OptArguments CRLF / "+'
CmdData
    Keyword = 1*ALPHA
    OptArguments = [ SP *(SP / VCHAR) ]
```

A command is either a single line containing a Keyw
command whose initial keyword begins with +, and
"." on a line of its own. (We use a special character t
that Tor can correctly parse multi-line commands th
commands and their arguments are described belo

Replies from Tor to the controlle

```
    Reply = SyncReply / AsyncReply
    SyncReply = *(MidReplyLine / DataReplyLin
    AsyncReply = *(MidReplyLine / DataReplyL

    MidReplyLine = StatusCode "-" ReplyLine
    DataReplyLine = StatusCode "+" ReplyLine
    EndReplyLine = StatusCode SP ReplyLine
    ReplyLine = [ReplyText] CRLF
    ReplyText = XXXX
    StatusCode = 3DIGIT
```

Unless specified otherwise, multiple lines in a single
guaranteed to share the same status code. Specific
section 3, and described more fully in section 4.

[Compatibility note: versions of Tor before 0.2.0.3-al
AsyncReplies of the form "*(MidReplyLine / DataRep
controllers that need to work with these versions of
line AsyncReplies with the final line (usually "650 OK

General-use tokens

; CRLF means, "the ASCII Carriage Return character (
ASCII Linefeed character (decimal value 10)." CRLF =

; How a controller tells Tor about a particular OR. Th
$Fingerprint -- The router whose identity key hashes
preferred way to refer to an OR. ; $Fingerprint~Nick
hashes to the ; given fingerprint, but only if the rout
$Fingerprint=Nickname -- The router whose identity
but only if the router is Named and has the given ; r
router with the given nickname, or, if no such ; route
matches the one given. ; This is not a safe way to ref
could under some circumstances change over time.
above follow:

ServerSpec = LongName / Nickname LongName = Fi

; For tors older than 0.3.1.3-alpha, LongName may h
lieu of a tilde ("~"). The presence of an equal sign ; d
"Named" flag:

LongName = Fingerprint [("=" / "~") Nickname]

Fingerprint = "$" 40*HEXDIG NicknameChar = "a"-"z" /*
NicknameChar

; What follows is an outdated way to refer to ORs. ; F
ServerID with LongName in events and ; GETINFO re
enabled starting in Tor version ; 0.1.2.2-alpha and it
later. ServerID = Nickname / Fingerprint

; Unique identifiers for streams or circuits. Currently
change StreamID = 1*16 IDChar CircuitID = 1*16 IDCha
IDChar IDChar = ALPHA / DIGIT

Address = ip4-address / ip6-address / hostname (XX

; A "CmdData" section is a sequence of octets conclu
CRLF "." CRLF. The terminating sequence may not ap
Leading periods on lines in the data are escaped wit
RFC 2821 section 4.5.2. CmdData = *DataLine "." CRLF*
NonDotItem *LineItem CRLF LineItem = NonCR / 1*CR N
1*CR NonCRLF

; ISOTime, ISOTime2, and ISOTime2Frac are time for
example ISOTime: "2012-01-11 12:15:33" ; example

example ISOTime2Frac: "2012-01-11T12:15:33.51" Is
2*DIGIT IsoTimePart = 2*DIGIT ":" 2*DIGIT ":" 2*DIGIT ISOT
ISOTime2 = IsoDatePart "T" IsoTimePart ISOTime2Fr

; Numbers LeadingDigit = "1" - "9" UInt = LeadingDig

Commands

All commands are case-insensitive, but most keywor

SETCONF

Change the value of one or more configuration vari

```
"SETCONF" 1*(SP keyword ["=" value]) CRLF
value = String / QuotedString
```

Tor behaves as though it had just read each of the k
file. Keywords with no corresponding values have th
NULL (use RESETCONF if you want to set it back to it
if there is an error in any of the configuration setting

Tor responds with a "250 OK" reply on success. If so
found, Tor replies with a "552 Unrecognized option"
with a "513 syntax error in configuration values" rep
impossible configuration setting" reply on a semant

Some configuration options (e.g. "Bridge") take mult
keys (e.g. for hidden services and for entry guard lis
where order matters (see GETCONF below). In these
SETCONF command is taken to reset all of the other
ORListenAddress values are configured, and a SETC
single ORListenAddress value, the new command's v

Sometimes it is not possible to change configuration
SETCONF commands, because the value of one of th
the value of another which has not yet been set. Suc
setting multiple configuration options with a single S
ORPort=443 ORListenAddress=9001).

RESETCONF

Remove all settings for a given configuration option
any), and then assign the String provided. Typically t
an option back to its default. The syntax is:

"RESETCONF" 1*(SP keyword ["=" String]) CRLF

Otherwise it behaves like SETCONF above.

GETCONF

Request the value of zero or more configuration var

"GETCONF" *(SP keyword) CRLF

If all of the listed keywords exist in the Tor configura lines of the form:

250 keyword=value

If any option is set to a 'default' value semantically c may reply with a reply line of the form:

250 keyword

Value may be a raw value or a quoted string. Tor wil when the value could be misinterpreted through no supports no such misinterpretable values for config

If some of the listed keywords can't be found, Tor re configuration keyword" message.

If an option appears multiple times in the configura returned in order.

If no keywords were provided, Tor responds with "2

Some options are context-sensitive, and depend on keywords. These cannot be fetched directly. Curren clients should use the "HiddenServiceOptions" virtu HiddenServiceDir, HiddenServicePort, HiddenServic HiddenserviceAuthorizeClient option settings.

SETEVENTS

Request the server to inform the client about intere

"SETEVENTS" [SP "EXTENDED"] *(SP EventCode) CRL

EventCode = 1*(ALPHA / "_") (see section 4.1.x for ev

Any events *not* listed in the SETEVENTS line are turn
an empty body turns off all event reporting.

The server responds with a "250 OK" reply on succe
reply if one of the event codes isn't recognized. (On
isn't changed.)

If the flag string "EXTENDED" is provided, Tor may p
for this connection; see 4.1 for more information. N
will be provided in extended format, or none. NOTE
Tor 0.1.1.9-alpha; it is always-on in Tor 0.2.2.1-alpha

Each event is described in more detail in Section 4.1

AUTHENTICATE

Sent from the client to the server. The syntax is:

"AUTHENTICATE" [SP 1*HEXDIG / QuotedString] CR

This command is used to authenticate to the server
following:

```
    * (For the HASHEDPASSWORD authenticatior
      The original password represented as a

    * (For the COOKIE is authentication meth
      The contents of the cookie file, forma

    * (For the SAFECOOKIE authentication met
      The HMAC based on the AUTHCHALLENGE me
```

The server responds with "250 OK" on success or "5
authentication cookie is incorrect. Tor closes the cor

The authentication token can be specified as either
unquoted hexadecimal encoding of that same string

For information on how the implementation securel
on disk, see section 5.1.

Before the client has authenticated, no command ot
AUTHCHALLENGE, AUTHENTICATE, or QUIT is valid.
command, or sends a malformed command, or sen
command, or sends PROTOCOLINFO or AUTHCHALI
error reply and closes the connection.

To prevent some cross-protocol attacks, the AUTHE
even if all authentication methods in Tor are disable
just send "AUTHENTICATE" CRLF.

(Versions of Tor before 0.1.2.16 and 0.2.0.4-alpha di
authentication failure.)

SAVECONF

Sent from the client to the server. The syntax is:

"SAVECONF" [SP "FORCE"] CRLF

Instructs the server to write out its config options in
successful, or "551 Unable to write configuration to
other error occurs.

If the %include option is used on torrc, SAVECONF w
If the flag string "FORCE" is provided, the configurati
%include is used. Using %include on defaults-torrc c
(Introduced in 0.3.1.1-alpha.)

See also the "getinfo config-text" command, if the co
itself.

See also the "getinfo config-can-saveconf" comman
required. (Also introduced in 0.3.1.1-alpha.)

SIGNAL

Sent from the client to the server. The syntax is:

"SIGNAL" SP Signal CRLF

```
     Signal = "RELOAD" / "SHUTDOWN" / "DUMP"
              "HUP" / "INT" / "USR1" / "USR2'
              "CLEARDNSCACHE" / "HEARTBEAT" ,

  The meaning of the signals are:

      RELOAD    -- Reload: reload config iten
      SHUTDOWN  -- Controlled shutdown: if se
                   If it's an OR, close liste
                   ShutdownWaitLength seconds
      DUMP      -- Dump stats: log informatio
                   circuits.
      DEBUG     -- Debug: switch all open log
      HALT      -- Immediate shutdown: clean
      CLEARDNSCACHE -- Forget the client-side
      NEWNYM    -- Switch to clean circuits,
                   don't share any circuits w
                   the client-side DNS cache.
                   response to this signal.)
      HEARTBEAT -- Make Tor dump an unschedul
      DORMANT   -- Tell Tor to become "dorman
                   try to avoid CPU and netwo
                   user-initiated network rec
                   on relays or hidden servic
      ACTIVE    -- Tell Tor to stop being "do
                   a user-initiated network r
```

The server responds with "250 OK" if the signal is re if it was asked to close immediately), or "552 Unreco unrecognized.

Note that not all of these signals have POSIX signal e below. You may also use these POSIX names for the

```
      RELOAD: HUP
      SHUTDOWN: INT
      HALT: TERM
      DUMP: USR1
      DEBUG: USR2

  [SIGNAL DORMANT and SIGNAL ACTIVE were adde
```

MAPADDRESS

Sent from the client to the server. The syntax is:

"MAPADDRESS" 1*(Address "=" Address SP) CRLF

The first address in each pair is an "original" addres address. The client sends this message to the server requests for connections to the original address sho the specified replacement address. If the addresses able to fulfill the request, the server replies with a 2!

```
250-OldAddress1=NewAddress1
250 OldAddress2=NewAddress2
```

containing the source and destination addresses. If replies with "512 syntax error in command argumen request, it replies with "451 resource exhausted".

The client may decline to provide a body for the orig special null address ("0.0.0.0" for IPv4, "::0" for IPv6, the server should choose the original address itself, The server should ensure that it returns an element in actual use. If there is already an address mapped may reuse that mapping.

If the original address is already mapped to a differe removed. If the original address and the destinatior removes any mapping in place for the original addre

Example:

```
C: MAPADDRESS 1.2.3.4=torproject.org
S: 250 1.2.3.4=torproject.org

C: GETINFO address-mappings/control
S: 250-address-mappings/control=1.2.3.4 t
S: 250 OK

C: MAPADDRESS 1.2.3.4=1.2.3.4
S: 250 1.2.3.4=1.2.3.4

C: GETINFO address-mappings/control
S: 250-address-mappings/control=
S: 250 OK
```

{Note: This feature is designed to be used to help To SOCKS4 or hostname-less SOCKS5. There are three

```
    1. Somehow make them use SOCKS4a or SOCK
    2. Use tor-resolve (or another interface
       feature) to resolve the hostname remo
       with special addresses like x.onion c
    3. Use MAPADDRESS to map an IP address t
       arrange to fool the application into
       has resolved to that IP.

This functionality is designed to help impl
```

Mappings set by the controller last until the Tor pro
controller wants the mapping to last only a certain t
the address when that time has elapsed.

MapAddress replies MAY contain mixed status code

Example:

```
  C: MAPADDRESS xxx=@@@ 0.0.0.0=bogus1.goog
  S: 512-syntax error: invalid address '@@(
  S: 250 127.199.80.246=bogus1.google.com
```

GETINFO

Sent from the client to the server. The syntax is as fo

"GETINFO" 1*(SP keyword) CRLF

Unlike GETCONF, this message is used for data that
configuration file, and that may be longer than a sin
sent for each requested value, followed by a final 25
single line, the format is:

```
    250-keyword=value
If a value must be split over multiple line

    250+keyword=
    value
    .
The server sends a 551 or 552 error on fail

Recognized keys and their values include:

  "version" -- The version of the server's
    name of the software, such as "Tor 0.0.
    if absent, is assumed to be "Tor".

  "config-file" -- The location of Tor's co

  "config-defaults-file" -- The location of
    defaults file ("torrc.defaults").  This
    torrc, and is typically used to replace
    configuration values. [First implemente

  "config-text" -- The contents that Tor wo
    a SAVECONF command, so the controller c
    disk itself. [First implemented in 0.2.

  "exit-policy/default" -- The default exit
    *append* to the ExitPolicy config optio

  "exit-policy/reject-private/default" -- T
    that Tor will *prepend* to the ExitPol
    ExitPolicyRejectPrivate is 1.

  "exit-policy/reject-private/relay" -- The
    lines that Tor will *prepend* to the Ex
    on the current values of ExitPolicyReje
    ExitPolicyRejectLocalInterfaces. These
    addresses configured in the torrc and p
    interfaces. Will send 552 error if the
    onion router. Will send 551 on internal

  "exit-policy/ipv4"
  "exit-policy/ipv6"
  "exit-policy/full" -- This OR's exit pol
    all-entries flavors. Handles errors in
    reject-private/relay" does.

  "desc/id/<OR identity>" or "desc/name/<OR
    server descriptor for a given OR.  (Not
    do not download server descriptors by d
    microdescriptors instead.  If microdesc
    need to use "md" instead.)

  "md/all" -- all known microdescriptors fo
    Each microdescriptor is terminated by a
    [First implemented in 0.3.5.1-alpha]
```

```
"md/id/<OR identity>" or "md/name/<OR nic
  microdescriptor for a given OR. Empty 
  that OR (because we haven't downloaded 
  consensus). [First implemented in 0.2.3

"desc/download-enabled" -- "1" if we try
  "0" otherwise. [First implemented in 0.

"md/download-enabled" -- "1" if we try to
  "0" otherwise. [First implemented in 0.

"dormant" -- A nonnegative integer: zero 
  building circuits, and nonzero if Tor h
  or some similar reason.  [First impleme

"desc-annotations/id/<OR identity>" -- ou
  (source, timestamp of arrival, purpose,
  descriptor. [First implemented in 0.2.0

"extra-info/digest/<digest>"  -- the extr
  hex) is <digest>.  Only available if we
  documents.

"ns/id/<OR identity>" or "ns/name/<OR nic
  status info (v3 directory style) for a 
  info is as given in dir-spec.txt, and r
  consensus opinion about the
  router in question. Like directory clie
  tolerate unrecognized flags and lines. 
  descriptor digest are those believed to
  not necessarily those for a descriptor 
  [First implemented in 0.1.2.3-alpha.]
  [In 0.2.0.9-alpha this switched from v2

"ns/all" -- Router status info (v3 direct
  that the consensus has an opinion about
  [First implemented in 0.1.2.3-alpha.]
  [In 0.2.0.9-alpha this switched from v2

"ns/purpose/<purpose>" -- Router status i
  for all ORs of this purpose. Mostly des
  queries.
  [First implemented in 0.2.0.13-alpha.]
  [In 0.2.0.9-alpha this switched from v2
  [In versions before 0.4.1.1-alpha we se
   bridges when /ns/purpose/bridge is acc
  [In 0.4.1.1-alpha we set the Running fl
   bridge networkstatus file is written t

"desc/all-recent" -- the latest server de
  Tor knows about.  (See md note about "d

"network-status" -- [Deprecated in 0.3.1.
  in 0.4.5.1-alpha.]

"address-mappings/all"
"address-mappings/config"
```

```
"address-mappings/cache"
"address-mappings/control" -- a \r\n-sepa
  mappings, each in the form of "from-add
  The 'config' key returns those address
  configuration; the 'cache' key returns
  client-side DNS cache; the 'control' ke
  via the control interface; the 'all' ta
  set through any mechanism.
  Expiry is formatted as with ADDRMAP eve
  always a time in UTC or the string "NEV
  First introduced in 0.2.0.3-alpha.

"addr-mappings/*" -- as for address-mappi
  expiry portion of the value.  Use of th
  since 0.2.0.3-alpha; use address-mappin

"address" -- the best guess at our extern
  have no guess, return a 551 error. (Add

"address/v4"
"address/v6"
  the best guess at our respective extern
  If we have no guess, return a 551 error

"fingerprint" -- the contents of the fing
  writes as a relay, or a 551 if we're no
  (Added in 0.1.2.3-alpha)

"circuit-status"
  A series of lines as for a circuit stat
  the form described in section 4.1.1, om
  "650 CIRC ".  Note that clients must be
  arguments as described in section 4.1.

"stream-status"
  A series of lines as for a stream statu
     StreamID SP StreamStatus SP CircuitI

"orconn-status"
  A series of lines as for an OR connecti
  0.1.2.2-alpha with feature VERBOSE_NAME
  0.2.2.1-alpha and later by default, eac
     LongName SP ORStatus CRLF

 In Tor versions 0.1.2.2-alpha through 0.
 VERBOSE_NAMES turned off and before vers
 is of the form:
     ServerID SP ORStatus CRLF

"entry-guards"
  A series of lines listing the currently
  In Tor 0.1.2.2-alpha with feature VERBO
  0.2.2.1-alpha and later by default, eac
     LongName SP Status [SP ISOTime] CRLF

 In Tor versions 0.1.2.2-alpha through 0.
 VERBOSE_NAMES turned off and before vers
```

```
 is of the form:
     ServerID2 SP Status [SP ISOTime] CRL
     ServerID2 = Nickname / 40*HEXDIG

  The definition of Status is the same fo
     Status = "up" / "never-connected" /
              "unusable" / "unlisted"

  [From 0.1.1.4-alpha to 0.1.1.10-alpha,
   "helper-nodes". Tor still supports cal
    is deprecated and should not be used.

  [Older versions of Tor (before 0.1.2.x-
   of unlisted/unusable. Between 0.1.2.x-
   'down' was never generated.]

  [XXXX ServerID2 differs from ServerID 
   with a $.  This is an implementation e
   the $ back in if we can do so without

"traffic/read" -- Total bytes read (downl

"traffic/written" -- Total bytes written

"uptime" -- Uptime of the Tor daemon (in
   0.3.5.1-alpha.

"accounting/enabled"
"accounting/hibernating"
"accounting/bytes"
"accounting/bytes-left"
"accounting/interval-start"
"accounting/interval-wake"
"accounting/interval-end"
  Information about accounting status.  I
  "enabled" is 1; otherwise it is 0.  The
  if we are accepting no data; "soft" if
  connections, and "awake" if we're not h
  and "bytes-left" fields contain (read-b
  start and the rest of the interval resp
  and 'interval-end' fields are the borde
  'interval-wake' field is the time withi
  where we plan[ned] to start being activ

"config/names"
  A series of lines listing the available
  of the form:
     OptionName SP OptionType [ SP Docume
     OptionName = Keyword
     OptionType = "Integer" / "TimeInterv
       "DataSize" / "Float" / "Boolean" /
       "Dependent" / "Virtual" / "String"
     Documentation = Text
  Note: The incorrect spelling "Dependant
key
  was introduced in Tor 0.1.1.4-alpha unt
  0.3.0.2-alpha.  It is recommended that
```

```
    "config/defaults"
      A series of lines listing default value
      option. Options which don't have a val
      in the list.  Introduced in Tor 0.2.4.1
         OptionName SP OptionValue CRLF
         OptionName = Keyword
         OptionValue = Text

    "info/names"
      A series of lines listing the available
      one of these forms:
         OptionName SP Documentation CRLF
         OptionPrefix SP Documentation CRLF
         OptionPrefix = OptionName "/*"
      The OptionPrefix form indicates a numbe
      prefix. So if "config/*" is listed, oth
      "config/" will work, but "config/*" its

    "events/names"
      A space-separated list of all the event
      Tor's SETEVENTS.

    "features/names"
      A space-separated list of all the featu
      of Tor's USEFEATURE.

    "signal/names"
      A space-separated list of all the value
      command.

    "ip-to-country/ipv4-available"
    "ip-to-country/ipv6-available"
      "1" if the relevant geoip or geoip6 dat
      This field was added in Tor 0.3.2.1-alp

    "ip-to-country/*"
      Maps IP addresses to 2-letter country c
      "GETINFO ip-to-country/18.0.0.1" should

    "process/pid" -- Process id belonging to
    "process/uid" -- User id running the tor
     unimplemented on Windows, returning -1).
    "process/user" -- Username under which th
     providing an empty string if none exists
     Windows, returning an empty string).
    "process/descriptor-limit" -- Upper bound
-1
     if unknown

    "dir/status-vote/current/consensus" [adde
    "dir/status-vote/current/consensus-microd
alpha]
    "dir/status/authority"
    "dir/status/fp/<F>"
    "dir/status/fp/<F1>+<F2>+<F3>"
    "dir/status/all"
```

```
"dir/server/fp/<F>"
"dir/server/fp/<F1>+<F2>+<F3>"
"dir/server/d/<D>"
"dir/server/d/<D1>+<D2>+<D3>"
"dir/server/authority"
"dir/server/all"
  A series of lines listing directory cor
  specification for the URLs listed in Se
  that Tor MUST NOT provide private infor
  routers not marked as general-purpose.
  information for which this Tor is not a
  an empty string.

  Note that, as of Tor 0.2.3.3-alpha, Tor
  descriptors anymore, but microdescripto
  unavailable" reply to all "GETINFO dir/
  correct.  If you have an old program wh
  descriptors to work, try setting UseMic
  FetchUselessDescriptors 1 in your clien

"status/circuit-established"
"status/enough-dir-info"
"status/good-server-descriptor"
"status/accepted-server-descriptor"
"status/..."
  These provide the current internal Tor
  states. See Section 4.1.10 for explanat
  status events are available as getinfo'
  you want more exposed.)
"status/reachability-succeeded/or"
  0 or 1, depending on whether we've foun
"status/reachability-succeeded/dir"
  0 or 1, depending on whether we've foun
  1 if there is no DirPort, and therefore
  check.
"status/reachability-succeeded"
  "OR=" ("0"/"1") SP "DIR=" ("0"/"1")
  Combines status/reachability-succeeded/
  unrecognized elements in this entry.
"status/bootstrap-phase"
  Returns the most recent bootstrap phase
  sent. Specifically, it returns a string
  "NOTICE BOOTSTRAP ..." or "WARN BOOTSTR
  use this getinfo when they connect or a
  current bootstrap state.
"status/version/recommended"
  List of currently recommended versions.
"status/version/current"
  Status of the current version. One of:
  recommended, new in series, obsolete, u
"status/clients-seen"
  A summary of which countries we've seen
  formatted the same as the CLIENTS_SEEN
  Section 4.1.14. This GETINFO option is
  for bridge relays.
"status/fresh-relay-descs"
  Provides fresh server and extra-info de
```

```
      this is *not* the latest descriptors we
we
      would generate if we needed to make a r

    "net/listeners/*"

      A quoted, space-separated list of the l
      for connections of the specified type.
      network address...

        "127.0.0.1:9050" "127.0.0.1:9051"

      ... or local unix sockets...

        "unix:/home/my_user/.tor/socket"

      ... or IPv6 network addresses:

        "[2001:0db8:7000:0000:0000:dead:beef:

      [New in Tor 0.2.2.26-beta.]

    "net/listeners/or"

      Listeners for OR connections. Talks Tor
      tor-spec.txt.

    "net/listeners/dir"

      Listeners for Tor directory protocol, a

    "net/listeners/socks"

      Listeners for onion proxy connections t

    "net/listeners/trans"

      Listeners for transparent connections r
      pf or netfilter.

    "net/listeners/natd"

      Listeners for transparent connections r

    "net/listeners/dns"

      Listeners for a subset of DNS protocol

    "net/listeners/control"

      Listeners for Tor control protocol, des

    "net/listeners/extor"

      Listeners corresponding to Extended ORP
      pluggable transports. See proposals 180
```

```
"net/listeners/httptunnel"

  Listeners for onion proxy connections t
  tunnelling.

  [The extor and httptunnel lists were ac
  0.3.4.6-rc.]

"dir-usage"
  A newline-separated list of how many by
  each type of directory request. The for
     Keyword 1*SP Integer 1*SP Integer
  where the first integer is the number c
  is the number of requests answered.

  [This feature was added in Tor 0.2.2.1-
   Tor 0.2.9.1-alpha. Even when it existe
   useful output when the Tor client was
   INSTRUMENT_DOWNLOADS or RUNNING_DOXYGE

"bw-event-cache"
  A space-separated summary of recent BW
  from oldest to newest.  Each event is r
  tuple of "R,W", R is the number of byte
  bytes written.  These entries each repr
  of traffic.
  [New in Tor 0.2.6.3-alpha]

 "consensus/valid-after"
 "consensus/fresh-until"
 "consensus/valid-until"
  Each of these produces an ISOTime descr
  the current (valid, accepted) consensus
  [New in Tor 0.2.6.3-alpha]

"hs/client/desc/id/<ADDR>"
  Prints the content of the hidden servic
  the given <ADDR> which is an onion addr
  The client's cache is queried to find t
  the descriptor is described in section
  document.

  If <ADDR> is unrecognized or if not fou
  returned.

  [New in Tor 0.2.7.1-alpha]
  [HS v3 support added 0.3.3.1-alpha]

"hs/service/desc/id/<ADDR>"
  Prints the content of the hidden servic
  the given <ADDR> which is an onion addr
  The service's local descriptor cache is
  The format of the descriptor is describ
  rend-spec.txt document.

  If <ADDR> is unrecognized or if not fou
  returned.
```

```
  [New in Tor 0.2.7.2-alpha]
  [HS v3 support added 0.3.3.1-alpha]

"onions/current"
"onions/detached"
  A newline-separated list of the Onion (
  via the "ADD_ONION" command. The 'curre
  belonging to the current control connec
  returns Onion Services detached from th
  (as in, belonging to no control connect
  The format of each line is:
     HSAddress
  [New in Tor 0.2.7.1-alpha.]
  [HS v3 support added 0.3.3.1-alpha]

"network-liveness"
  The string "up" or "down", indicating w
  network is reachable.

"downloads/"
  The keys under downloads/ are used to (
  return either a sequence of newline-ter
  a "serialized download status" as follo

   SerializedDownloadStatus =
     -- when do we plan to next attempt t
     "next-attempt-at" SP ISOTime CRLF
     -- how many times have we failed sin
     "n-download-failures" SP UInt CRLF
     -- how many times have we tried to (
     "n-download-attempts" SP UInt CRLF
     -- according to which schedule rule
     "schedule" SP DownloadSchedule CRLF
     -- do we want to fetch this from an
     "want-authority" SP DownloadWantAuth
     -- do we increase our download delay
     -- or whenever we attempt fetching t
     "increment-on" SP DownloadIncrementO
     -- do we increase the download sched
     -- random?
     "backoff" SP DownloadBackoff CRLF
     [
       -- with an exponential backoff, wh
       "last-backoff-position" Uint CRLF
       -- with an exponential backoff, wh
       "last-delay-used UInt CRLF
     ]

  where

  DownloadSchedule =
    "DL_SCHED_GENERIC" / "DL_SCHED_CONSEN
  DownloadWantAuthority =
    "DL_WANT_ANY_DIRSERVER" / "DL_WANT_AU
  DownloadIncrementOn =
    "DL_SCHED_INCREMENT_FAILURE" / "DL_SC
```

```
      DownloadBackoff =
        "DL_SCHED_DETERMINISTIC" / "DL_SCHED_

      The optional last two lines must be pre
      "DL_SCHED_RANDOM_EXPONENTIAL" and must
      is "DL_SCHED_DETERMINISTIC".

      In detail, the keys supported are:

      "downloads/networkstatus/ns"
        The SerializedDownloadStatus for the
        whichever bootstrap state Tor is curr

      "downloads/networkstatus/ns/bootstrap"
        The SerializedDownloadStatus for the
        bootstrap time, regardless of whether

      "downloads/networkstatus/ns/running"

        The SerializedDownloadStatus for the
        running, regardless of whether we are

      "downloads/networkstatus/microdesc"
        The SerializedDownloadStatus for the
        whichever bootstrap state Tor is curr

      "downloads/networkstatus/microdesc/boot
        The SerializedDownloadStatus for the
        bootstrap time, regardless of whether

      "downloads/networkstatus/microdesc/runn
        The SerializedDownloadStatus for the
when
        running, regardless of whether we are

      "downloads/cert/fps"

        A newline-separated list of hex-encod
        certificates for which we have downlo

      "downloads/cert/fp/<Fingerprint>"
        A SerializedDownloadStatus for the de
        identity digest <Fingerprint> returne

      "downloads/cert/fp/<Fingerprint>/sks"
        A newline-separated list of hex-encod
        authority identity digest <Fingerprin
        downloads/cert/fps key.

      "downloads/cert/fp/<Fingerprint>/<SKDig
        A SerializedDownloadStatus for the ce
        digest <Fingerprint> returned by the
signing
        key digest <SKDigest> returned by the
/fp/<Fingerprint>/
        sks key.
```

```
        "downloads/desc/descs"
          A newline-separated list of hex-encoc
          [note, not identity digests - the Tor
          yet while downloading router descript
          using a NS-flavored consensus, a 551

        "downloads/desc/<Digest>"
          A SerializedDownloadStatus for the rc
          <Digest> as returned by the downloads
          process is not using a NS-flavored cc
returned.

        "downloads/bridge/bridges"
          A newline-separated list of hex-encoc
          the Tor process is not using bridges,

        "downloads/bridge/<Digest>"
          A SerializedDownloadStatus for the br
          digest <Digest> as returned by the dc
          the Tor process is not using bridges,

      "sr/current"
      "sr/previous"
        The current or previous shared random v
        consensus, base-64 encoded.  An empty v
        the consensus has no shared random valu

      "current-time/local"
      "current-time/utc"
        The current system or UTC time, as retu
        format.  (Introduced in 0.3.4.1-alpha.)

      "stats/ntor/requested"
      "stats/ntor/assigned"
        The NTor circuit onion handshake rephis
        assigned.  (Introduced in 0.4.5.1-alpha

      "stats/tap/requested"
      "stats/tap/assigned"
        The TAP circuit onion handshake rephist
        assigned.  (Introduced in 0.4.5.1-alpha

      "config-can-saveconf"
        0 or 1, depending on whether it is poss
        FORCE flag. (Introduced in 0.3.1.1-alph

      "limits/max-mem-in-queues"
        The amount of memory that Tor's out-of-
        Tor to allocate (in places it can see)
        and killing circuits. See the MaxMemIn(
        details. Unlike the option, this value
        may be adjusted depending on the availa
        the MaxMemInQueues option. (Introduced

    Examples:

       C: GETINFO version desc/name/moria1
```

```
S: 250+desc/name/moria=
S: [Descriptor for moria]
S: .
S: 250-version=Tor 0.1.1.0-alpha-cvs
S: 250 OK
```

EXTENDCIRCUIT

Sent from the client to the server. The format is:

```
"EXTENDCIRCUIT" SP CircuitID
                [SP ServerSpec *("," Se
                [SP "purpose=" Purpose]
```

This request takes one of two forms: either the Circu
request for the server to build a new circuit, or the C
a request for the server to extend an existing circuit
specified path.

If the CircuitID is 0, the controller has the option of p
the circuit. If it does not provide a path, Tor will sele
capacity nodes according to path-spec.txt.

If CircuitID is 0 and "purpose=" is specified, then the
are recognized: "general" and "controller". If not spe
"general".

If the request is successful, the server sends a reply
of the CircuitID of the (maybe newly created) circuit.
SP CircuitID CRLF.

SETCIRCUITPURPOSE

Sent from the client to the server. The format is:

"SETCIRCUITPURPOSE" SP CircuitID SP "purpose=" P

This changes the circuit's purpose. See EXTENDCIRC

SETROUTERPURPOSE

Sent from the client to the server. The format is:

"SETROUTERPURPOSE" SP NicknameOrKey SP Purp

This changes the descriptor's purpose. See +POSTD

NOTE: This command was disabled and made obsol
exist anymore, and is listed here only for historical i

ATTACHSTREAM

Sent from the client to the server. The syntax is:

"ATTACHSTREAM" SP StreamID SP CircuitID [SP "HO

This message informs the server that the specified s
specified circuit. Each stream may be associated wit
streams may share the same circuit. Streams can or
(that is, circuits that have sent a circuit status 'BUILT
GETINFO circuit-status request).

If the circuit ID is 0, responsibility for attaching the g

If HOP=HopNum is specified, Tor will choose the Ho
node, rather than the last node in the circuit. Hops a
permitted to attach to hop 1.

Tor responds with "250 OK" if it can attach the strea
exist, 555 if the stream isn't in an appropriate state
or 551 if the stream couldn't be attached for anothe

{Implementation note: Tor will close unattached str
after they are born. Let the developers know if that

{Implementation note: By default, Tor automatically
unless the configuration variable "__LeaveStreamsU
attach streams via TC when "__LeaveStreamsUnatta
between Tor and the controller, as both attempt to

{Implementation note: You can try to attachstream
connect or resolve request but hasn't succeeded ye
stream from its current circuit before proceeding wi

POSTDESCRIPTOR

Sent from the client to the server. The syntax is:

```
"+POSTDESCRIPTOR" [SP "purpose=" Purpose]
                  CRLF Descriptor CRLF ".
```

This message informs the server about a new descr be either "general", "controller", or "bridge", else we "general".

If Cache is specified, it must be either "no" or "yes", not specified, Tor will decide for itself whether it wa controllers must not rely on its choice.

The descriptor, when parsed, must contain a numbe fields for its nickname and identity.

If there is an error in parsing the descriptor, the serv descriptor" reply. If the descriptor is well-formed bu must reply with a 251 message whose body explains descriptor is added, Tor replies with "250 OK".

REDIRECTSTREAM

Sent from the client to the server. The syntax is:

"REDIRECTSTREAM" SP StreamID SP Address [SP Por

Tells the server to change the exit address on the sp changes the destination port as well. No remapping address.

To be sure that the modified address will be used, th stream event is received, and before attaching this s

Tor replies with "250 OK" on success.

CLOSESTREAM

Sent from the client to the server. The syntax is:

"CLOSESTREAM" SP StreamID SP Reason *(SP Flag) (

Tells the server to close the specified stream. The re RELAY_END reasons given in tor-spec.txt, as a decim servers SHOULD ignore unrecognized flags. Tor may flush any data that is pending.

Tor replies with "250 OK" on success, or a 512 if the if it doesn't recognize the StreamID or reason.

CLOSECIRCUIT

The syntax is:

```
"CLOSECIRCUIT" SP CircuitID *(SP Flag) (
Flag = "IfUnused"
```

Tells the server to close the specified circuit. If "IfUn circuit unless it is unused.

Other flags may be defined in the future; Tor SHOUL

Tor replies with "250 OK" on success, or a 512 if the if it doesn't recognize the CircuitID.

QUIT

Tells the server to hang up on this controller connec before authenticating.

USEFEATURE

Adding additional features to the control protocol s compatibility. Initially such features are added into T USEFEATURE can enable these additional features.

The syntax is:

```
"USEFEATURE" *(SP FeatureName) CRLF
FeatureName = 1*(ALPHA / DIGIT / "_" / "-
```

```
Feature names are case-insensitive.
```

Once enabled, a feature stays enabled for the durat
controller. A new connection to the controller must
feature.

Features are a forward-compatibility mechanism; ea
standard part of the control protocol. Once a featur
always-on. Each feature documents the version it wa
version in which it became part of the protocol.

Tor will ignore a request to use any feature that is a
response to an unrecognized feature.

EXTENDED_EVENTS

```
   Same as passing 'EXTENDED' to SETEVENTS;
   request the extended event syntax.

   This feature was first introduced in 0.1
   and part of the protocol in Tor 0.2.2.1-

VERBOSE_NAMES

   Replaces ServerID with LongName in event
   provides a Fingerprint for all routers,
   and a Nickname if one is known. LongName
   than ServerID, which only provides eithe

   This feature was first introduced in 0.1
   part of the protocol in Tor 0.2.2.1-alph
```

RESOLVE

The syntax is

```
"RESOLVE" *Option *Address CRLF
Option = "mode=reverse"
Address = a hostname or IPv4 address
```

This command launches a remote hostname lookup
(or reverse lookup if "mode=reverse" is specified). N
background: to see the answers, your controller will

see 4.1.7 below.

[Added in Tor 0.2.0.3-alpha]

PROTOCOLINFO

The syntax is:

"PROTOCOLINFO" *(SP PIVERSION) CRLF

The server reply format is:

"250-PROTOCOLINFO" SP PIVERSION CRLF *InfoLine

InfoLine = AuthLine / VersionLine / OtherLine

```
      AuthLine = "250-AUTH" SP "METHODS=" Auth
                       *(SP "COOKIEFILE=" Aut
      VersionLine = "250-VERSION" SP "Tor=" To

      AuthMethod =
       "NULL"           / ; No authentication
       "HASHEDPASSWORD" / ; A controller must
       "COOKIE"         / ; ... or supply the
       "SAFECOOKIE"       ; ... or prove knowl

      AuthCookieFile = QuotedString
      TorVersion = QuotedString

      OtherLine = "250-" Keyword OptArguments

     PIVERSION: 1*DIGIT
```

This command tells the controller what kinds of auth

Tor MAY give its InfoLines in any order; controllers M
they do not recognize. Controllers MUST ignore extr

PIVERSION is there in case we drastically change the
always be "1". Controllers MAY provide a list of the p
Tor MAY select a version that the controller does no

AuthMethod is used to specify one or more control
currently accepts.

AuthCookieFile specifies the absolute path and filen
Tor is expecting and is provided iff the METHODS fie
and/or "SAFECOOKIE". Controllers MUST handle esc

All authentication cookies are 32 bytes long. Control
non-32-byte-long file as an authentication cookie.

If the METHODS field contains the method "SAFECO
contain the same authentication cookie.

The COOKIE authentication method exposes the use
unintended information disclosure attack whenever
read access than the process that it has connected t
to a process other than Tor.) It is almost never safe t
has explicitly specified which filename to read an au
reason, the COOKIE authentication method has bee
from a future version of Tor.

The VERSION line contains the Tor version.

[Unlike other commands besides AUTHENTICATE, PI
once!) before AUTHENTICATE.]

[PROTOCOLINFO was not supported before Tor 0.2.

LOADCONF

The syntax is:

"+LOADCONF" CRLF ConfigText CRLF "." CRLF

This command allows a controller to upload the text
port. This config file is then loaded as if it had been

[LOADCONF was added in Tor 0.2.1.1-alpha.]

TAKEOWNERSHIP

The syntax is:

"TAKEOWNERSHIP" CRLF

This command instructs Tor to shut down when this
command affects each control connection that send
connections send the TAKEOWNERSHIP command t
when any of those connections closes.

(As of Tor 0.2.5.2-alpha, Tor does not wait a while fo
because of an exiting controller. If you want to ensu
should!--then send "SIGNAL SHUTDOWN" and wait f

This command is intended to be used with the __Ow
option. A controller that starts a Tor process which t
should 'own' that Tor process:

```
* When starting Tor, the controller shoul
  __OwningControllerProcess on Tor's comm
  cause Tor to poll for the existence of
  and exit if it does not find such a pro
  completely reliable way to detect wheth
  controller' is still running, but it sh
  most cases.)

* Once the controller has connected to To
  should send the TAKEOWNERSHIP command a
  connection.  At this point, *both* the
  the __OwningControllerProcess option ar
  exit when the control connection ends *
  detects that there is no process with t
  __OwningControllerProcess option.

* After the controller has sent the TAKEO
  should send "RESETCONF __OwningControll
  control connection.  This will cause To
  existence of a process with its owning
  will still exit when the control connec

[TAKEOWNERSHIP was added in Tor 0.2.2.28-be
```

AUTHCHALLENGE

The syntax is:

```
"AUTHCHALLENGE" SP "SAFECOOKIE"
                SP ClientNonce
                CRLF

ClientNonce = 2*HEXDIG / QuotedString
```

This command is used to begin the authentication r
authentication.

If the server accepts the command, the server reply

```
"250 AUTHCHALLENGE"
        SP "SERVERHASH=" ServerHash
        SP "SERVERNONCE=" ServerNonce
        CRLF

ServerHash = 64*64HEXDIG
ServerNonce = 64*64HEXDIG
```

The ClientNonce, ServerHash, and ServerNonce valu
way as the argument passed to the AUTHENTICATE
bytes long.

ServerHash is computed as:

```
HMAC-SHA256("Tor safe cookie authenticat
            CookieString | ClientNonce |

(with the HMAC key as its first argument)
```

After a controller sends a successful AUTHCHALLEN
sent on the connection must be an AUTHENTICATE
authentication string which that AUTHENTICATE cor

```
HMAC-SHA256("Tor safe cookie authenticat
            CookieString | ClientNonce |
```

[Unlike other commands besides AUTHENTICATE, A
only once!) before AUTHENTICATE.]

[AUTHCHALLENGE was added in Tor 0.2.3.13-alpha.]

DROPGUARDS

The syntax is:

"DROPGUARDS" CRLF

Tells the server to drop all guard nodes. Do not invo
increase vulnerability to tracking attacks over time.

Tor replies with "250 OK" on success.

[DROPGUARDS was added in Tor 0.2.5.2-alpha.]

HSFETCH

The syntax is:

```
"HSFETCH" SP (HSAddress / "v" Version "-'
         *[SP "SERVER=" Server] CRLF

HSAddress = 16*Base32Character / 56*Base3
Version = "2" / "3"
DescId = 32*Base32Character
Server = LongName
```

This command launches hidden service descriptor f
DescId.

HSAddress can be version 2 or version 3 addresses.
Version 2 addresses consist of 16*Base32Character a*
*56*Base32Character.

If a DescId is specified, at least one Server MUST als
is returned. If no DescId and Server(s) are specified,
descriptor fetch. If one or more Server are given, th
on each of them in parallel.

The caching behavior when fetching a descriptor us
normal Tor client behavior.

Details on how to compute a descriptor id (DescId)
1.3.

If any values are unrecognized, a 513 error is return
success, Tor replies "250 OK" then Tor MUST eventu
and HS_DESC_CONTENT events with the results. If S
emitted for each location.

Examples are:

```
C: HSFETCH v2-gezdgnbvgy3tqolbmjrwizlgm5
   SERVER=9695DFC35FFEB861329B9F1AB04C46
S: 250 OK

C: HSFETCH ajkhdsfuygaesfaa
S: 250 OK

C: HSFETCH vww6ybal4bd7szmgncyruucpgfkqa
S: 250 OK
```

[HSFETCH was added in Tor 0.2.7.1-alpha] [HS v3 su

ADD_ONION

The syntax is:

```
    "ADD_ONION" SP KeyType ":" KeyBlob
            [SP "Flags=" Flag *("," Flag)]
            [SP "MaxStreams=" NumStreams]
            1*(SP "Port=" VirtPort ["," Targe
            *(SP "ClientAuth=" ClientName [":
            *(SP "ClientAuthV3=" V3Key) CRLF

    KeyType =
     "NEW"     / ; The server should generate
     "RSA1024" / ; The server should use the
                   in as KeyBlob (v2).
     "ED25519-V3"; The server should use the
                   KeyBlob (v3).

    KeyBlob =
     "BEST"    / ; The server should generate
                   supported algorithm (KeyTy
                   [As of 0.4.2.3-alpha, ED25
     "RSA1024" / ; The server should generate
                   (KeyType == "NEW") (v2).
     "ED25519-V3"; The server should generate
                   (KeyType == "NEW") (v3).
     String      ; A serialized private key (

    Flag =
     "DiscardPK" / ; The server should not in
                     private key as part of t
     "Detach"    / ; Do not associate the new
                     to the current control c
     "BasicAuth" / ; Client authorization is
                     method (v2 only).
     "V3Auth"    / ; Version 3 client authori

     "NonAnonymous" /; Add a non-anonymous Si
                       checks this flag match
                       service anonymity mode
     "MaxStreamsCloseCircuit"; Close the circ
                               allowed is rea

    NumStreams = A value between 0 and 65535
                 streams that can be attached
Setting
                 it to 0 means unlimited whic

    VirtPort = The virtual TCP Port for the O
               HiddenServicePort "VIRTPORT" a

    Target = The (optional) target for the gi
             optional HiddenServicePort "TARG

    ClientName = An identifier 1 to 16 charac
                 characters in A-Za-z0-9+-_ (

    ClientBlob = Authorization data for the c
                 specific to the authorizatio
```

```
     V3Key = The client's base32-encoded x2551
             part of rend-spec-v3.txt section

   The server reply format is:

     "250-ServiceID=" ServiceID CRLF
     ["250-PrivateKey=" KeyType ":" KeyBlob CR
     *("250-ClientAuth=" ClientName ":" Client
     "250 OK" CRLF

     ServiceID = The Onion Service address wit
                 suffix
```

Tells the server to create a new Onion ("Hidden") Se
and algorithm. If a KeyType of "NEW" is selected, the
using the selected algorithm. The "Port" argument's
identical semantics to the corresponding HiddenSer

The server response will only include a private key i
generate a new keypair, and also the "DiscardPK" fla
"DiscardPK" flag is specified, there is no way to recre
corresponding Onion Service at a later date).

If client authorization is enabled using the "BasicAut
will not be accessible to clients without valid authori
"HidServAuth" option). The list of authorized clients
"ClientAuth" parameters. If "ClientBlob" is not specif
be randomly generated and returned.

Tor instances can either be in anonymous hidden se
single onion service mode. All hidden services on th
anonymity. To guard against unexpected loss of anc
ADD_ONION "NonAnonymous" flag matches the cu
The hidden service anonymity mode is configured u
HiddenServiceSingleHopMode and HiddenServiceN
options are 1, the "NonAnonymous" flag must be pr
options are 0 (the Tor default), the flag must NOT be

Once created the new Onion Service will remain act
removed via "DEL_ONION", the server terminates, o
originated the "ADD_ONION" command is closed. It
Onion Service on control connection close by specif

It is the Onion Service server application's responsib
connections if desired after the Onion Service is rem

(The KeyBlob format is left intentionally opaque, ho
currently the Base64 encoded DER representation c

newlines removed. For a "ED25519-V3" key is the Ba
of the 32-byte ed25519 secret scalar in little-endian

[Note: The ED25519-V3 format is not the same as, e
stores the concatenation of the 32-byte ed25519 ha
byte public key, and which derives the secret scalar
seed with SHA-512. Our key blinding scheme is inco
seeds, so we store the secret scalar alongside the PI
recomputing the public key when importing an ED2!

Examples:

```
     C: ADD_ONION NEW:BEST Flags=DiscardPK Po
     S: 250-
ServiceID=exampleoniont2pqglbny66wpovyvao3ylc
     S: 250 OK

     C: ADD_ONION RSA1024:[Blob Redacted] Por
     S: 250-ServiceID=sampleonion12456
     S: 250 OK

     C: ADD_ONION NEW:BEST Port=22 Port=80,80
     S: 250-
ServiceID=sampleonion4t2pqglbny66wpovyvao3ylc
     S: 250-PrivateKey=ED25519-V3:[Blob Redac
     S: 250 OK

     C: ADD_ONION NEW:RSA1024 Flags=DiscardPK
        ClientAuth=alice:[Blob Redacted] Clie
     S: 250-ServiceID=testonion1234567
     S: 250-ClientAuth=bob:[Blob Redacted]
     S: 250 OK

     C: ADD_ONION NEW:ED25519-V3 ClientAuthV3
     S: 250-
ServiceID=n35etu3yjxrqjpntmfziom5sjwspoydchme
     S: 250-ClientAuthV3=[Blob Redacted]
     S: 250 OK

   Examples with Tor in anonymous onion servic

     C: ADD_ONION NEW:BEST Flags=DiscardPK Po
     S: 250-
ServiceID=exampleoniont2pqglbny66wpovyvao3ylc
     S: 250 OK

     C: ADD_ONION NEW:BEST Flags=DiscardPK,No
     S: 512 Tor is in anonymous hidden servic

   Examples with Tor in non-anonymous onion se

     C: ADD_ONION NEW:BEST Flags=DiscardPK Po
     S: 512 Tor is in non-anonymous hidden se

     C: ADD_ONION NEW:BEST Flags=DiscardPK,No
     S: 250-
ServiceID=exampleoniont2pqglbny66wpovyvao3ylc
     S: 250 OK
```

[ADD_ONION was added in Tor 0.2.7.1-alpha.] [MaxS
were added in Tor 0.2.7.2-alpha] [ClientAuth was ad
[NonAnonymous was added in Tor 0.2.9.3-alpha.] [H
[ClientV3Auth support added 0.4.6.1-alpha]

DEL_ONION

The syntax is:

"DEL_ONION" SP ServiceID CRLF

```
ServiceID = The Onion Service address wi
            suffix
```

Tells the server to remove an Onion ("Hidden") Serv
"ADD_ONION" command. It is only possible to remo
on the same control connection as the "DEL_ONION
no control connection in particular (The "Detach" fla

If the ServiceID is invalid, or is neither owned by the
detached Onion Service, the server will return a 552

It is the Onion Service server application's responsib
connections if desired after the Onion Service has b

Tor replies with "250 OK" on success, or a 512 if the
arguments, or a 552 if it doesn't recognize the Servi

[DEL_ONION was added in Tor 0.2.7.1-alpha.] [HS v3

HSPOST

The syntax is:

```
"+HSPOST" *[SP "SERVER=" Server] [SP "HSA
          CRLF Descriptor CRLF "." CRLF

Server = LongName
HSAddress = 56*Base32Character
Descriptor =  The text of the descriptor
in rend-spec.txt section 1.3.
```

The "HSAddress" key is optional and only applies fo
returned if used with v2.

This command launches a hidden service descriptor
or more Server arguments are provided, an upload
parallel. If no Server options are provided, it behave
and will upload to the set of responsible HS director

If any value is unrecognized, a 552 error is returned
is an error in parsing the descriptor, the server mus

On success, Tor replies "250 OK" then Tor MUST eve
event with the result for each upload location.

Examples are:

```
C: +HSPOST SERVER=9695DFC35FFEB861329B9F
   [DESCRIPTOR]
   .
S: 250 OK
```

```
[HSPOST was added in Tor 0.2.7.1-alpha]
```

ONION_CLIENT_AUTH_ADD

The syntax is:

```
"ONION_CLIENT_AUTH_ADD" SP HSAddress
                        SP KeyType ":" Pr
                        [SP "ClientName="
                        [SP "Flags=" TYPE

HSAddress = 56*Base32Character
KeyType = "x25519" is the only one suppor
PrivateKeyBlob = base64 encoding of x2551
```

Tells the connected Tor to add client-side v3 client a
with "HSAddress". The "PrivateKeyBlob" is the x2551
this client, and "Nickname" is an optional nickname

FLAGS is a comma-separated tuple of flags for this r
supported flags are:

```
"Permanent" - This client's credentials s
filesystem.
              If this is not set, the cl
              and stored in memory.
```

If client auth credentials already existed for this serv

If Tor has cached onion service descriptors that it ha
(due to lack of client auth credentials), attempt to de
this command succeeds.

On success, "250 OK" is returned. Otherwise, the fol

```
     251 - Client auth credentials for this or
 replaced.
     252 - Added client auth credentials and s
 descriptor.
     451 - We reached authorized client capac
     512 - Syntax error in "HSAddress", or "Pr
     551 - Client with with this "Nickname" al
     552 - Unrecognized KeyType

   [ONION_CLIENT_AUTH_ADD was added in Tor 0.4
```

ONION_CLIENT_AUTH_REMOVE

The syntax is:

"ONION_CLIENT_AUTH_REMOVE" SP HSAddress

KeyType = "x25519" is the only one supported right

Tells the connected Tor to remove the client-side v3 service with "HSAddress".

On success "250 OK" is returned. Otherwise, the foll

```
     512 - Syntax error in "HSAddress".
     251 - Client credentials for "HSAddress"

   [ONION_CLIENT_AUTH_REMOVE was added in Tor
```

ONION_CLIENT_AUTH_VIEW

The syntax is:

"ONION_CLIENT_AUTH_VIEW" [SP HSAddress] CRLF

Tells the connected Tor to list all the stored client-si "HSAddress". If no "HSAddress" is provided, list all tl credentials.

The server reply format is:

```
"250-ONION_CLIENT_AUTH_VIEW" [SP HSAddres
*("250-CLIENT" SP HSAddress SP KeyType ":
              [SP "ClientName=" Nickname]
              [SP "Flags=" FLAGS] CRLF)
"250 OK" CRLF

HSAddress = The onion address under which
KeyType = "x25519" is the only one suppor
PrivateKeyBlob = base64 encoding of x2551
```

"Nickname" is an optional nickname for this client, w
ONION_CLIENT_AUTH_ADD command, or it's the file
are stored in the filesystem.

FLAGS is a comma-separated field of flags for this cl
are:

"Permanent" - This client's credentials are stored in

On success "250 OK" is returned. Otherwise, the foll

512 - Syntax error in "HSAddress".

[ONION_CLIENT_AUTH_VIEW was added in Tor 0.4.3

DROPOWNERSHIP

The syntax is:

"DROPOWNERSHIP" CRLF

This command instructs Tor to relinquish ownership
will not shut down when this control connection is c

This method is idempotent. If the control connection
this method returns successfully, and does nothing.

The controller can call TAKEOWNERSHIP again to re-

[DROPOWNERSHIP was added in Tor 0.4.0.0-alpha]

DROPTIMEOUTS

```
The syntax is:
  "DROPTIMEOUTS" CRLF
```

Tells the server to drop all circuit build times. Do no
increase vulnerability to tracking attacks over time.

Tor replies with "250 OK" on success. Tor also emits
right after this "250 OK".

[DROPTIMEOUTS was added in Tor 0.4.5.0-alpha.]

Replies

Reply codes follow the same 3-character format as u
defining a status, the second character defining a su
fine-grained information.

The TC protocol currently uses the following first ch

```
  2yz   Positive Completion Reply
     The command was successful; a new requ

  4yz   Temporary Negative Completion reply
     The command was unsuccessful but might

  5yz   Permanent Negative Completion Reply
     The command was unsuccessful; the clie
     that sequence of commands again.

  6yz   Asynchronous Reply
     Sent out-of-order in response to an ea

The following second characters are used:

  x0z   Syntax
     Sent in response to ill-formed or nons

  x1z   Protocol
     Refers to operations of the Tor Contro

  x5z   Tor
     Refers to actual operations of Tor sys

The following codes are defined:

   250 OK
   251 Operation was unnecessary
      [Tor has declined to perform the ope

   451 Resource exhausted

   500 Syntax error: protocol

   510 Unrecognized command
   511 Unimplemented command
   512 Syntax error in command argument
   513 Unrecognized command argument
   514 Authentication required
   515 Bad authentication

   550 Unspecified Tor error

   551 Internal error
            [Something went wrong inside T
             request couldn't be fulfilled

   552 Unrecognized entity
            [A configuration key, a stream
             mentioned in the command did

   553 Invalid configuration value
      [The client tried to set a configura
        incorrect, ill-formed, or impossib

   554 Invalid descriptor
```

```
555 Unmanaged entity

650 Asynchronous event notification
```

Unless specified to have specific contents, the huma
should not be relied upon to match those in this do

Asynchronous events

These replies can be sent after a corresponding SET
They will not be interleaved with other Reply elemen
command and its corresponding reply. For example

```
C: SETEVENTS CIRC
S: 250 OK
C: GETCONF SOCKSPORT ORPORT
S: 650 CIRC 1000 EXTENDED moria1,moria2
S: 250-SOCKSPORT=9050
S: 250 ORPORT=0

But this sequence is disallowed:

C: SETEVENTS CIRC
S: 250 OK
C: GETCONF SOCKSPORT ORPORT
S: 250-SOCKSPORT=9050
S: 650 CIRC 1000 EXTENDED moria1,moria2
S: 250 ORPORT=0
```

Clients MUST tolerate more arguments in an asynch
MUST tolerate more lines in an asynchronous reply
that expects a CIRC message like:

650 CIRC 1000 EXTENDED moria1,moria2

must tolerate:

```
650-CIRC 1000 EXTENDED moria1,moria2 0x
650-EXTRAMAGIC=99
650 ANONYMITY=high
```

If clients receives extended events (selected by USEI
0.1.2.2-alpha..Tor-0.2.1.x, and always-on in Tor 0.2.2
specified below may be followed by additional argu
lines will be of the form:

"650" ("-"/" ") KEYWORD ["=" ARGUMENTS] CRLF

Additional arguments will be of the form

SP KEYWORD ["=" (QuotedString / * NonSpDquote)

Clients MUST tolerate events with arguments and ke
SHOULD process those events as if any unrecognize
present.

Clients SHOULD NOT depend on the order of keywo
NOT depend on there being no new keyword=value
existing keyword=value arguments, though as of thi
extensions to this protocol should add new keyword
until all controllers have been fixed. At some point t
"MUST NOT".

Circuit status changed

The syntax is:

```
"650" SP "CIRC" SP CircuitID SP CircStat
     [SP "BUILD_FLAGS=" BuildFlags] [SP
     [SP "HS_STATE=" HSState] [SP "REND_
     [SP "TIME_CREATED=" TimeCreated]
     [SP "REASON=" Reason [SP "REMOTE_RE
     [SP "SOCKS_USERNAME=" EscapedUserna
     [SP "SOCKS_PASSWORD=" EscapedPasswo
     [SP "HS_POW=" HSPoW ]
     CRLF

 CircStatus =
          "LAUNCHED" / ; circuit ID ass
          "BUILT"    / ; all hops finish
          "GUARD_WAIT" / ; all hops fin
                         ;  circuit with
          "EXTENDED" / ; one more hop ha
          "FAILED"   / ; circuit closed
          "CLOSED"     ; circuit closed

 Path = LongName *("," LongName)
   ; In Tor versions 0.1.2.2-alpha throu
   ; VERBOSE_NAMES turned off and before
   ; is as follows:
   ; Path = ServerID *("," ServerID)

 BuildFlags = BuildFlag *("," BuildFlag)
 BuildFlag = "ONEHOP_TUNNEL" / "IS_INTER
             "NEED_CAPACITY" / "NEED_UPT

 Purpose = "GENERAL" / "HS_CLIENT_INTRO"
           "HS_SERVICE_INTRO" / "HS_SERV
           "CONTROLLER" / "MEASURE_TIMEO
           "HS_VANGUARDS" / "PATH_BIAS_T
           "CIRCUIT_PADDING"

 HSState = "HSCI_CONNECTING" / "HSCI_INT
           "HSCR_CONNECTING" / "HSCR_EST
           "HSCR_ESTABLISHED_WAITING" /
           "HSSI_CONNECTING" / "HSSI_EST
           "HSSR_CONNECTING" / "HSSR_JOI

 HSPoWType = "v1"
 HSPoWEffort = 1*DIGIT
 HSPoW = HSPoWType "," HSPoWEffort

 EscapedUsername = QuotedString
 EscapedPassword = QuotedString

 HSAddress = 16*Base32Character / 56*Bas
 Base32Character = ALPHA / "2" / "3" / "

 TimeCreated = ISOTime2Frac
 Seconds = 1*DIGIT
 Microseconds = 1*DIGIT

 Reason = "NONE" / "TORPROTOCOL" / "INTE
```

```
          "HIBERNATING" / "RESOURCELIMIT
          "OR_IDENTITY" / "OR_CONN_CLOSE
          "FINISHED" / "DESTROYED" / "NO
          "MEASUREMENT_EXPIRED"

The path is provided only when the circuit
hop.

The "BUILD_FLAGS" field is provided only f
and later.  Clients MUST accept build flag
Build flags are defined as follows:

   ONEHOP_TUNNEL   (one-hop circuit, used
   IS_INTERNAL     (internal circuit, not
   NEED_CAPACITY   (this circuit must use
   NEED_UPTIME     (this circuit must use

The "PURPOSE" field is provided only in ve
later, and only if extended events are ena
MUST accept purposes not listed above.  Pu
follows:

   GENERAL         (circuit for AP and/or
   HS_CLIENT_INTRO (HS client-side introdu
   HS_CLIENT_REND  (HS client-side rendezv
   HS_SERVICE_INTRO (HS service-side intro
   HS_SERVICE_REND (HS service-side rendez
   TESTING         (reachability-testing c
   CONTROLLER      (circuit built by a con
   MEASURE_TIMEOUT (circuit being kept aro
   HS_VANGUARDS    (circuit created ahead
                   HS vanguards, and later
   PATH_BIAS_TESTING (circuit used to prob
                   being deliberately clos
   CIRCUIT_PADDING (circuit that is being
                   true close time)

The "HS_STATE" field is provided only for
and only in versions 0.2.3.11-alpha and la
hidden-service circuit states not listed a
circuit states are defined as follows:

   HSCI_*      (client-side introduction-p
     HSCI_CONNECTING           (connecting
     HSCI_INTRO_SENT           (sent INTROD
     HSCI_DONE                 (received re

   HSCR_*      (client-side rendezvous-poi
     HSCR_CONNECTING           (connecting
     HSCR_ESTABLISHED_IDLE     (established
     HSCR_ESTABLISHED_WAITING  (introductio
     HSCR_JOINED               (connected t

   HSSI_*      (service-side introduction-
     HSSI_CONNECTING           (connecting
     HSSI_ESTABLISHED          (established
```

```
   HSSR_*      (service-side rendezvous-po
     HSSR_CONNECTING          (connecting
     HSSR_JOINED              (connected t

The "SOCKS_USERNAME" and "SOCKS_PASSWORD"
that were used by a SOCKS client to connec
initiate this circuit. (Streams for SOCKS
usernames and/or passwords are isolated or
IsolateSOCKSAuth flag is active; see Propo
0.4.3.1-alpha.]

The "REND_QUERY" field is provided only fo
circuits, and only in versions 0.2.3.11-al
MUST accept hidden service addresses in fo
specified above. [Added in Tor 0.4.3.1-alp

The "TIME_CREATED" field is provided only
later.  TIME_CREATED is the time at which
cannibalized. [Added in Tor 0.4.3.1-alpha.

The "REASON" field is provided only for FA
if extended events are enabled (see 3.19).
not listed above. [Added in Tor 0.4.3.1-al
tor-spec.txt, except for:

   NOPATH              (Not enough nodes t
   MEASUREMENT_EXPIRED (As "TIMEOUT", exce
                        open for measureme
                        would take to fini
   IP_NOW_REDUNDANT    (Closing a circuit
                        has become redunda
                        opened in parallel

The "REMOTE_REASON" field is provided only
TRUNCATE cell, and only if extended events
actual reason given by the remote OR for c
accept reasons not listed above.  Reasons
[Added in Tor 0.4.3.1-alpha.]
```

Stream status changed

The syntax is:

```
"650" SP "STREAM" SP StreamID SP Stream
    [SP "REASON=" Reason [ SP "REMOTE_F
    [SP "SOURCE=" Source] [ SP "SOURCE_
    [SP "PURPOSE=" Purpose] [SP "SOCKS_
    [SP "SOCKS_PASSWORD=" EscapedPasswo
    [SP "CLIENT_PROTOCOL=" ClientProtoc
    [SP "SESSION_GROUP=" SessionGroup]
    CRLF

StreamStatus =
         "NEW"          / ; New request
         "NEWRESOLVE"   / ; New request
         "REMAP"        / ; Address re-
         "SENTCONNECT"  / ; Sent a conn
         "SENTRESOLVE"  / ; Sent a reso
         "SUCCEEDED"    / ; Received a
         "FAILED"       / ; Stream fail
         "CLOSED"       / ; Stream clos
         "DETACHED"     / ; Detached fr
         "CONTROLLER_WAIT"  ; Waiting f
ATTACHSTREAM
                            ; (new in 0
         "XOFF_SENT"  ; XOFF has been s
                      ; (new in 0.4.7.5
         "XOFF_RECV"  ; XOFF has been r
                      ; (new in 0.4.7.5
         "XON_SENT"  ; XON has been sen
                     ; (new in 0.4.7.5-
         "XON_RECV"  ; XON has been rec
                     ; (new in 0.4.7.5-

 Target = TargetAddress ":" Port
 Port = an integer from 0 to 65535 incl
 TargetAddress = Address / "(Tor_intern

 EscapedUsername = QuotedString
 EscapedPassword = QuotedString

 ClientProtocol =
         "SOCKS4"      /
         "SOCKS5"      /
         "TRANS"       /
         "NATD"        /
         "DNS"         /
         "HTTPCONNECT" /
         "UNKNOWN"

 NymEpoch = a nonnegative integer
 SessionGroup = an integer

 IsoFields = a comma-separated list of

 IsoField =
         "CLIENTADDR" /
         "CLIENTPORT" /
         "DESTADDR" /
```

```
"DESTPORT" /
the name of a field that is va
```

The circuit ID designates which circuit this stream is unattached, the circuit ID "0" is given. The target ind is meant to resolve or connect to; it can be "(Tor_int the Tor program to talk to itself.

```
   Reason = "MISC" / "RESOLVEFAILED" / "CC
            "EXITPOLICY" / "DESTROY" / "DC
            "NOROUTE" / "HIBERNATING" / "I
            "CONNRESET" / "TORPROTOCOL" /
            "PRIVATE_ADDR"

The "REASON" field is provided only for FA
events, and only if extended events are en
accept reasons not listed above.  Reasons
except for:

   END          (We received a RELAY_END c
                 stream.)
   PRIVATE_ADDR (The client tried to conne
                 127.0.0.1 or 10.0.0.1 ove
   [XXXX document more. -NM]

The "REMOTE_REASON" field is provided only
cell, and only if extended events are enab
reason given by the remote OR for closing
reasons not listed above.  Reasons are as

"REMAP" events include a Source if extende

   Source = "CACHE" / "EXIT"

Clients MUST accept sources not listed abo
the Tor client decided to remap the addres
answer, and "EXIT" is given if the remote
the new address as a response.

The "SOURCE_ADDR" field is included with N
extended events are enabled.  It indicates
that requested the connection, and can be
requesting program.

   Purpose = "DIR_FETCH" / "DIR_UPLOAD" /
             "USER" /  "DIRPORT_TEST"

The "PURPOSE" field is provided only for N
only if extended events are enabled (see 3
purposes not listed above.  The purposes a

    "DIR_FETCH" -- This stream is generate
      fetching directory information.
    "DIR_UPLOAD" -- An internal stream for
      a directory authority.
    "DIRPORT_TEST" -- A stream we're using
      port to make sure it's reachable.
    "DNS_REQUEST" -- A user-initiated DNS
    "USER" -- This stream is handling user
      to Tor, but it doesn't match one of

The "SOCKS_USERNAME" and "SOCKS_PASSWORD"
that were used by a SOCKS client to connec
initiate this stream. (Streams for SOCKS c
```

```
   usernames and/or passwords are isolated or
   IsolateSOCKSAuth flag is active; see Propo

   The "CLIENT_PROTOCOL" field indicates the
client
   to initiate this stream. (Streams for clie
   protocols are isolated on separate circuit
   flag is active.)  Controllers MUST tolerat

   The "NYM_EPOCH" field indicates the nym ep
client
   initiated this stream. The epoch increment
   received. (Streams with different nym epoc
   circuits.)

   The "SESSION_GROUP" field indicates the se
   that a client used to initiate this stream
is
   different for each listener port, but this
listener
   via the "SessionGroup" option in torrc. (S
   groups are isolated on separate circuits.)

   The "ISO_FIELDS" field indicates the set o
   stream isolation is enabled for the lister
   initiate this stream.  The special values
   "DESTADDR", and "DESTPORT", if their corre
   present, refer to the Address and Port com
   Target fields.
```

OR Connection status changed

The syntax is:

```
    "650" SP "ORCONN" SP (LongName / Target)
              Reason ] [ SP "NCIRCS=" NumCircu

    ORStatus = "NEW" / "LAUNCHED" / "CONNECTE

        ; In Tor versions 0.1.2.2-alpha throu
        ; VERBOSE_NAMES turned off and before
        ; Connection is as follows:
        "650" SP "ORCONN" SP (ServerID / Targ
                  Reason ] [ SP "NCIRCS=" NumC
```

NEW is for incoming connections, and LAUNCHED is CONNECTED means the TLS handshake has finishec connection is being closed that hasn't finished its ha connections that have handshaked.

A LongName or ServerID is specified unless it's a NE know what server it is yet, so we use Address:Port.

If extended events are enabled (see 3.19), optional r
information is provided for CLOSED and FAILED eve

```
      Reason = "MISC" / "DONE" / "CONNECTREFU
               "IDENTITY" / "CONNECTRESET" /
               "IOERROR" / "RESOURCELIMIT" /

   NumCircuits counts both established and per

   The ORStatus values are as follows:

      NEW -- We have received a new incoming (
        the server-side handshake.
      LAUNCHED -- We have launched a new outgo
        starting the client-side handshake.
      CONNECTED -- The OR connection has been
        done.
      FAILED -- Our attempt to open the OR con
      CLOSED -- The OR connection closed in ar

   The Reason values for closed/failed OR conn

      DONE -- The OR connection has shut down
      CONNECTREFUSED -- We got an ECONNREFUSEI
         OR.
      IDENTITY -- We connected to the OR, but
         not what we expected.
      CONNECTRESET -- We got an ECONNRESET or
         connection with the OR.
      TIMEOUT -- We got an ETIMEOUT or similar
         with the OR, or we're closing the cor
         long.
      NOROUTE -- We got an ENOTCONN, ENETUNREA
         similar error while connecting to the
      IOERROR -- We got some other IO error or
      RESOURCELIMIT -- We don't have enough op
         descriptors, buffers, etc) to connect
      PT_MISSING -- No pluggable transport was
      MISC -- The OR connection closed for son

   [First added ID parameter in 0.2.5.2-alpha]
```

Bandwidth used in the last second

The syntax is:

```
"650" SP "BW" SP BytesRead SP BytesWritt
BytesRead = 1*DIGIT
BytesWritten = 1*DIGIT
Type = "DIR" / "OR" / "EXIT" / "APP" / .
Num = 1*DIGIT
```

BytesRead and BytesWritten are the totals. [In a futu breakdown of the connection types that used bandv yet).]

Log messages

The syntax is:

"650" SP Severity SP ReplyText CRLF

or

"650+" Severity CRLF Data 650 SP "OK" CRLF

Severity = "DEBUG" / "INFO" / "NOTICE" / "WARN"/ "I

Some low-level logs may be sent from signal handle signal-safe. These low-level logs include backtraces, code called by logging functions. Signal-safe logs are events.

Control port message trace debug logs are never se modifying control output when debugging.

New descriptors available

This event is generated when new router descriptor anything else) are received.

Syntax:

```
"650" SP "NEWDESC" 1*(SP LongName) CRLF
   ; In Tor versions 0.1.2.2-alpha throu
   ; VERBOSE_NAMES turned off and before
   ; is as follows:
   "650" SP "NEWDESC" 1*(SP ServerID) CR
```

New Address mapping

These events are generated when a new address m map cache, or when the answer for a RESOLVE com by a successful or failed DNS lookup, a successful o RESOLVE command, a MAPADDRESS command, the the TrackHostExits feature.

Syntax:

```
      "650" SP "ADDRMAP" SP Address SP NewAddr
        [SP "error=" ErrorCode] [SP "EXPIRES="
 Cached]
        [SP "STREAMID=" StreamId] CRLF

      NewAddress = Address / "<error>"
      Expiry = DQUOTE ISOTime DQUOTE / "NEVER"

      ErrorCode = "yes" / "internal" / "Unabl
      UTCExpiry = DQUOTE IsoTime DQUOTE

      Cached = DQUOTE "YES" DQUOTE / DQUOTE "N
      StreamId = DQUOTE StreamId DQUOTE
```

Error and UTCExpiry are only provided if extended e Error are mostly useless. Future values will be chos "Unable to launch resolve request" value is a bug in

Expiry is expressed as the local time (rather than UT compatibility; new code should look at UTCExpiry in is omitted.)

Cached indicates whether the mapping will be store notification in response to a RESOLVE command.

StreamId is the global stream identifier of the strea was resolved.

Descriptors uploaded to us in our role as

[NOTE: This feature was removed in Tor 0.3.2.1-alph

Tor generates this event when it's a directory autho a server descriptor.

Syntax:

```
"650" "+" "AUTHDIR_NEWDESCS" CRLF Actio
  Descriptor CRLF "." CRLF "650" SP "OK
Action = "ACCEPTED" / "DROPPED" / "REJE
Message = Text
```

The Descriptor field is the text of the server descript
we're accepting the descriptor as the new best valid
if we aren't taking the descriptor and we're complai
and "DROPPED" if we decide to drop the descriptor
field is a human-readable string explaining why we
newlines.)

Our descriptor changed

Syntax:

"650" SP "DESCCHANGED" CRLF

[First added in 0.1.2.2-alpha.]

Status events

Status events (STATUS_GENERAL, STATUS_CLIENT, a
on occurrences in the Tor process pertaining to the
Generally, they correspond to log messages of seve
log messages in that their format is a specified inter

Syntax:

```
"650" SP StatusType SP StatusSeverity SP
                                        [SP

StatusType = "STATUS_GENERAL" / "STATUS_
StatusSeverity = "NOTICE" / "WARN" / "ER
StatusAction = 1*ALPHA
StatusArguments = StatusArgument *(SP St
StatusArgument = StatusKeyword '=' Statu
StatusKeyword = 1*(ALNUM / "_")
StatusValue = 1*(ALNUM / '_')  / QuotedS

StatusAction is a string, and StatusArgu
keyword=value pairs on the same line.  V
strings, or quoted strings.

These events are always produced with EX
VERBOSE_NAMES; see the explanations in t
for details.

Controllers MUST tolerate unrecognized a
unrecognized arguments, MUST tolerate mi
tolerate arguments that arrive in any or

Each event description below is accompan
controllers.  These recommendations are
is required to implement them.
```

Compatibility note: versions of Tor before 0.2.0.22-r "STATUS_SERVER" as "STATUS_SEVER". To be compa accept both.

Actions for STATUS_GENERAL events can be as follo

```
CLOCK_JUMPED
"TIME=NUM"
  Tor spent enough time without CPU cycl
  its circuits and will establish them a
  happens when a laptop goes to sleep an
  also happens when the system is swapp
  starving. The "time" argument specifie
  thinks it was unconscious for (or alte
  seconds it went back in time).

  This status event is sent as NOTICE se
  severity if Tor is acting as a server

  {Recommendation for controller: ignore
  know what the user should do anyway. H

DANGEROUS_VERSION
"CURRENT=version"
"REASON=NEW/OBSOLETE/UNRECOMMENDED"
"RECOMMENDED=\"version, version, ...\""
  Tor has found that directory servers c
  the Tor software.  RECOMMENDED is a co
  of Tor versions that are recommended.
  of Tor is newer than any recommended v
  this version of Tor is older than any
  UNRECOMMENDED if some recommended vers
  some are older than this version. (The
  "OLD" from Tor 0.1.2.3-alpha up to and

  {Controllers may want to suggest that
  UNRECOMMENDED versions.  NEW versions
  simply be development versions.}

TOO_MANY_CONNECTIONS
"CURRENT=NUM"
  Tor has reached its ulimit -n or whate
  descriptors or sockets.  CURRENT is th
  currently has open.  The user should r
  this. The "current" argument shows the
  open.

  {Controllers may recommend that the us
  increase it for them.  Recommendations
  OS-appropriate way and automated when

BUG
"REASON=STRING"
  Tor has encountered a situation that
  and the developers would like to learn
  the controller can explain this to the
  file a bug report?

  {Controllers should log bugs, but shou
  bug appears frequently.}

CLOCK_SKEW
```

```
  SKEW="+" / "-" SECONDS
  MIN_SKEW="+" / "-" SECONDS.
  SOURCE="DIRSERV:" IP ":" Port /
         "NETWORKSTATUS:" IP ":" Port /
         "OR:" IP ":" Port /
         "CONSENSUS"
    If "SKEW" is present, it's an estima
    time declared in the source.  (In ot
    the past, the value is -3600.)  "MIN
    bound.  If the source is a DIRSERV,
    connection to a dirserver.  If the s
    decided we're skewed because we got
    the future.  If the source is OR, th
    cell from a connection to another re
    CONSENSUS, we decided we're skewed b
    consensus from the future.

    {Tor should send this message to con
    skew is so high that it will interfe
    Controllers shouldn't blindly adjust
    accurate source of skew info (DIRSER
    unauthenticated.}

BAD_LIBEVENT
"METHOD=" libevent method
"VERSION=" libevent version
"BADNESS=" "BROKEN" / "BUGGY" / "SLOW"
"RECOVERED=" "NO" / "YES"
   Tor knows about bugs in using the con
   version of libevent.  "BROKEN" libeve
   "BUGGY" libevents might work okay; "S
   fine, but not quickly.  If "RECOVERED
   switch to a more reliable (but probab

   {Controllers may want to warn the use
   generally it's the fault of whoever b
   not much the user can do besides upgr
   binary.}

DIR_ALL_UNREACHABLE
  Tor believes that none of the known di
  reachable -- this is most likely becau
  down or otherwise not working, and mig
  user why Tor appears to be broken.

  {Controllers may want to warn the user
  action is generally not possible.}

Actions for STATUS_CLIENT events can be as

BOOTSTRAP
"PROGRESS=" num
"TAG=" Keyword
"SUMMARY=" String
["WARNING=" String]
["REASON=" Keyword]
["COUNT=" num]
```

```
["RECOMMENDATION=" Keyword]
["HOST=" QuotedString]
["HOSTADDR=" QuotedString]

  Tor has made some progress at establis
  Tor network, fetching directory inforn
  circuit; or it has encountered a probl
  status event is especially useful for
  or with connectivity problems.

  "Progress" gives a number between 0 ar
  the bootstrapping process we are. "Sum
  be displayed to the user to describe t
  will tackle, i.e., the task it is work
  status event. "Tag" is a string that c
  recognize bootstrap phases, if they wa
  than just blindly displaying the summa
  for the current tags that Tor issues.

  The StatusSeverity describes whether t
  phase (severity notice) or an indicati
  problem (severity warn).

  For bootstrap problems, we include the
  summary values as we would for a norma
  also include "warning", "reason", "cou
  key/value combos. The "count" number t
  problems there have been so far at thi
  string lists one of the reasons allowe
  "warning" argument string with any hir
  why it's having troubles bootstrapping

  The "reason" values are long-term-stab
  identify particular issues in a bootst
  strings, on the other hand, are human-
  SHOULD NOT rely on the format of any w
  the possible values for "recommendatio
  "warn" -- if ignore, the controller ca
  a pile of problems to show the user if
  the controller should alert the user t
  there's a bootstrapping problem.

  The "host" value is the identity diges
  trying to connect to; the "hostaddr" i
  where 'address' is an ipv4 or ipv6 add

  Currently Tor uses recommendation=igno
  nine bootstrap problem reports for a g
  uses recommendation=warn for subsequen
  phase. Hopefully this is a good balanc
  occasional errors and reporting seriou

ENOUGH_DIR_INFO
  Tor now knows enough network-status do
  descriptors that it's going to start t
 [Newer versions of Tor (0.2.6.2-alpha a
  If the consensus contains Exits (the t
```

```
    both exit and internal circuits. If nc
    circuits.]

    {Controllers may want to use this ever
    progress to their users, but should nc
    to tell them so.}

  NOT_ENOUGH_DIR_INFO
    We discarded expired statuses and serv
    below the desired threshold of directc
    try to build any circuits until ENOUGH

    {Controllers may want to use this ever
    progress to their users, but should nc
    to tell them so.}

  CIRCUIT_ESTABLISHED
    Tor is able to establish circuits for
    only be sent if we just built a circu
    that is, prior to this event we didn't
    establish circuits.

    {Suggested use: controllers can notify
    ready for use as a client once they se
    controllers should also have a timeout
    this event hasn't arrived, to give tip
    On the other hand, hopefully Tor will
    if it can identify the problem.]}

  CIRCUIT_NOT_ESTABLISHED
  "REASON=" "EXTERNAL_ADDRESS" / "DIR_ALL_
    We are no longer confident that we car
    keyword provides an explanation: which
    our lack of confidence.

    {Controllers may want to use this ever
    progress to their users, but should nc
    to do so.}
    [Note: only REASON=CLOCK_JUMPED is imp

  CONSENSUS_ARRIVED
     Tor has received and validated a new
     (This event can be delayed a little w
     is received, if Tor needs to fetch ce

  DANGEROUS_PORT
  "PORT=" port
  "RESULT=" "REJECT" / "WARN"
    A stream was initiated to a port that'
    vulnerable-plaintext protocols. If the
    refused the connection; whereas if it'

    {Controllers should warn their users w
    happen to know that the application us
    correctly (e.g., because it is part of
    might also want some sort of interface
    their RejectPlaintextPorts and WarnPla
```

```
    DANGEROUS_SOCKS
    "PROTOCOL=" "SOCKS4" / "SOCKS5"
    "ADDRESS=" IP:port
      A connection was made to Tor's SOCKS p
      approaches that doesn't support hostna
      If the client application got this add
      it may be leaking target addresses via

      {Controllers should warn their users w
      happen to know that the application us
      correctly (e.g., because it is part of

    SOCKS_UNKNOWN_PROTOCOL
      "DATA=string"
      A connection was made to Tor's SOCKS p
      for something other than the SOCKS pro
      using Tor as an HTTP proxy?   The DATA
      sent to Tor on the SOCKS port.

      {Controllers may want to warn their us
      indicates a misconfigured application.

    SOCKS_BAD_HOSTNAME
     "HOSTNAME=QuotedString"
      Some application gave us a funny-looki
      it is broken? In any case it won't wor
      should know.

      {Controllers may want to warn their us
      usually indicates a misconfigured appl

 Actions for STATUS_SERVER can be as follows

    EXTERNAL_ADDRESS
    "ADDRESS=IP"
    "HOSTNAME=NAME"
    "METHOD=CONFIGURED/CONFIGURED_ORPORT/DIR
            INTERFACE/GETHOSTNAME"
      Our best idea for our externally visib
      'HOSTNAME' is present, we got the new
      method is 'CONFIGURED', the IP was giv
      configuration option.  If the method i
was
      given verbatim in the ORPort configura
      'RESOLVED', we resolved the Address co
IP.
      If the method is 'GETHOSTNAME', we res
IP.
      If the method is 'INTERFACE', we got t
      interfaces to get the IP.  If the meth
      server told us a guess for what our IP

      {Controllers may want to record this i

    CHECKING_REACHABILITY
    "ORADDRESS=IP:port"
```

```
"DIRADDRESS=IP:port"
  We're going to start testing the reach
  or directory port.

  {This event could affect the controlle
  the controller should not interrupt th

REACHABILITY_SUCCEEDED
"ORADDRESS=IP:port"
"DIRADDRESS=IP:port"
  We successfully verified the reachabil
  directory port (depending on which of
  given.)

  {This event could affect the controlle
  the controller should not interrupt th

GOOD_SERVER_DESCRIPTOR
  We successfully uploaded our server de
  of the directory authorities, with no

  {Originally, the goal of this event wa
  has accepted the descriptor, so there
  about it." But since some authorities
  harder to get certainty than we had th
  is equivalent to ACCEPTED_SERVER_DESCR
  should just look at ACCEPTED_SERVER_DE
  this event for now.}

SERVER_DESCRIPTOR_STATUS
"STATUS=" "LISTED" / "UNLISTED"
  We just got a new networkstatus consen
  it or not in it has changed. Specifica
  if we're listed in it but previous to
  we were listed in a consensus; and sta
  thought we should have been listed in
  the last one), but we're not.

  {Moving from listed to unlisted is not
  alarm. The relay might have failed a t
  or the Internet might have had some ro
  feature is mainly to let relay operato
  has successfully been listed in the co

  [Not implemented yet. We should do thi

NAMESERVER_STATUS
"NS=addr"
"STATUS=" "UP" / "DOWN"
"ERR=" message
   One of our nameservers has changed st

   {This event could affect the controll
   the controller should not interrupt t

NAMESERVER_ALL_DOWN
   All of our nameservers have gone down
```

```
             {This is a problem; if it happens oft
             coming up again, the user needs to co
             nameservers.}

          DNS_HIJACKED
             Our DNS provider is providing an addr
             "NOTFOUND"; Tor will treat the addres

             {This is an annoyance; controllers ma
             DNS provider is not to be trusted.}

          DNS_USELESS
             Our DNS provider is giving a hijacked
             websites; Tor will not try to be an e

             {Controllers could warn the admin if
             exit node: the admin needs to configu
             Alternatively, this happens a lot in
             (hotels, universities, coffeeshops) w

          BAD_SERVER_DESCRIPTOR
          "DIRAUTH=addr:port"
          "REASON=string"
             A directory authority rejected our de
             include malformed descriptors, incorr
             and so on.

             {Controllers should warn the admin, a

          ACCEPTED_SERVER_DESCRIPTOR
          "DIRAUTH=addr:port"
             A single directory authority accepted
             // actually notice

            {This event could affect the controlle
            the controller should not interrupt th

          REACHABILITY_FAILED
          "ORADDRESS=IP:port"
          "DIRADDRESS=IP:port"
            We failed to connect to our external O
            successfully.

            {This event could affect the controlle
            controller should warn the admin and s
     take.}

          HIBERNATION_STATUS
          "STATUS=" "AWAKE" | "SOFT" | "HARD"
            Our bandwidth based accounting status
            relaying traffic/rejecting new connect

            {This event could affect the controlle
            controller MAY inform the admin, thoug
            explicitly enabled for a reason.}
```

```
[This event was added in tor 0.2.9.0-a
```

Our set of guard nodes has changed

Syntax:

```
"650" SP "GUARD" SP Type SP Name SP Stat
Type = "ENTRY"
Name = ServerSpec
  (Identifies the guard affected)
Status = "NEW" | "UP" | "DOWN" | "BAD" |
```

The ENTRY type indicates a guard used for connecti

The Status values are:

```
"NEW"  -- This node was not previously us
          picked it as one.
"DROPPED" -- This node is one we previous
          no longer consider it to be a m
"UP"   -- The guard now seems to be reach
"DOWN" -- The guard now seems to be unrea
"BAD"  -- Because of flags set in the cor
          configuration, this node is now
"BAD_L2" -- This layer2 guard has expirec
          consensus. This node is removec
"GOOD" -- Because of flags set in the cor
          configuration, this node is now

Controllers must accept unrecognized types
```

Network status has changed

Syntax:

"650" "+" "NS" CRLF 1*NetworkStatus "." CRLF "650"

The event is used whenever our local view of a relay
we get a new v3 consensus (in which case the entrie
see in the NEWCONSENSUS event, below), but it also
relay as up or down in our local status, for example

[First added in 0.1.2.3-alpha]

Bandwidth used on an application stream

The syntax is:

```
"650" SP "STREAM_BW" SP StreamID SP Byte
        Time CRLF
BytesWritten = 1*DIGIT
BytesRead = 1*DIGIT
Time = ISOTime2Frac
```

BytesWritten and BytesRead are the number of byte
since the last STREAM_BW event on this stream.

Note that from Tor's perspective, *reading* a byte on a
wrote the byte. That's why the order of "written" vs "
events compared to bw events.

The Time field is provided only in versions 0.3.2.1-al
created the bandwidth event.

These events are generated about once per second
for streams that have not written or read. These eve
Tor (such as on a SOCKSPort, TransPort, or so on). T
streams.

Per-country client stats

The syntax is:

```
"650" SP "CLIENTS_SEEN" SP TimeStarted S
IPVersions CRLF
```

We just generated a new summary of which countri
The controller could display this for the user, e.g. in
give them a sense that they are actually being usefu

Currently only bridge relays will receive this event, b
sufficiently aggregate and sanitize the client counts
sending these events in other cases too.

TimeStarted is a quoted string indicating when the r
UTCS).

The CountrySummary keyword has as its argument
set of "countrycode=count" pairs. For example (with
TimeStarted="2008-12-25 23:50:43" CountrySumma

The IPVersions keyword has as its argument a comr family=count" pairs. For example, IPVersions=v4=16

Note that these values are rounded, not exact. The r
description of "geoip-client-origins" in dir-spec.txt.

New consensus networkstatus has arrive

The syntax is:

```
"650" "+" "NEWCONSENSUS" CRLF 1*NetworkS
"OK" CRLF
```

A new consensus networkstatus has arrived. We inc
the consensus. NEWCONSENSUS is a separate even
here represents every usable relay: so any relay *not*
longer recommended.

[First added in 0.2.1.13-alpha]

New circuit buildtime has been set

The syntax is:

```
"650" SP "BUILDTIMEOUT_SET" SP Type SP '
   "TIMEOUT_MS=" Timeout SP "XM=" Xm SP
   "CUTOFF_QUANTILE=" Quantile SP "TIMEO
   "CLOSE_MS=" CloseTimeout SP "CLOSE_RA
   CRLF
Type = "COMPUTED" / "RESET" / "SUSPENDEI
Total = Integer count of timeouts storec
Timeout = Integer timeout in millisecond
Xm = Estimated integer Pareto parameter
Alpha = Estimated floating point Paredo
Quantile = Floating point CDF quantile c
TimeoutRate = Floating point ratio of ci
CloseTimeout = How long to keep measuren
CloseRate = Floating point ratio of meas
```

A new circuit build timeout time has been set. If Typ
the value based on historical data. If Type is "RESET"
changes have caused Tor to reset the timeout back
is "SUSPENDED", Tor has detected a loss of network
changed the timeout value to the default until the n
Tor has decided to discard timeout values that likely
down. If type is "RESUME", Tor has decided to resum

The Total value is the count of circuit build times Tor capped internally at the maximum number of build (NCIRCUITS_TO_OBSERVE).

The Timeout itself is provided in milliseconds. Intern nearest second before using it.

[First added in 0.2.2.7-alpha]

Signal received

The syntax is:

"650" SP "SIGNAL" SP Signal CRLF

Signal = "RELOAD" / "DUMP" / "DEBUG" / "NEWNYM

A signal has been received and actions taken by Tor mapping to Unix signals, is as defined in section 3.7. signals other than those listed here; controllers MUS

If Tor chose to ignore a signal (such as NEWNYM), th some options (like ReloadTorrcOnSIGHUP) may affe

Note that the HALT (SIGTERM) and SHUTDOWN (SIG any event.

[First added in 0.2.3.1-alpha]

Configuration changed

The syntax is:

StartReplyLine *(MidReplyLine) EndReplyLine

```
StartReplyLine = "650-CONF_CHANGED" CRLF
MidReplyLine = "650-" KEYWORD ["=" VALUE
EndReplyLine = "650 OK"
```

Tor configuration options have changed (such as via KEYWORD and VALUE specify the configuration opti configuration options contain only the KEYWORD.

Circuit status changed slightly

The syntax is:

```
    "650" SP "CIRC_MINOR" SP CircuitID SP Ci
          [SP "BUILD_FLAGS=" BuildFlags] [SP
          [SP "HS_STATE=" HSState] [SP "REND_
          [SP "TIME_CREATED=" TimeCreated]
          [SP "OLD_PURPOSE=" Purpose [SP "OLI

    CircEvent =
             "PURPOSE_CHANGED" / ; circuit pu
 changed
             "CANNIBALIZED"      ; circuit ca

   Clients MUST accept circuit events not list
```

The "OLD_PURPOSE" field is provided for both PURI
events. The "OLD_HS_STATE" field is provided when
provided and is a hidden-service-related purpose.

Other fields are as specified in section 4.1.1 above.

[First added in 0.2.3.11-alpha]

Pluggable transport launched

The syntax is:

```
    "650" SP "TRANSPORT_LAUNCHED" SP Type SP
    Type = "server" | "client"
    Name = The name of the pluggable transpo
    TransportAddress = An IPv4 or IPv6 addre
                       transport is listenin
    Port = The TCP port on which it is liste

    A pluggable transport called 'Name' of t
    successfully and is now listening for co
```

Bandwidth used on an OR or DIR or EXIT c

The syntax is:

```
      "650" SP "CONN_BW" SP "ID=" ConnID SP "T
               SP "READ=" BytesRead SP "WRITTE

      ConnType = "OR" /  ; Carrying traffic wi
                           either be our own (
                           relaying within the
                 "DIR" / ; Fetching tor descri
                           descriptors we're m
                 "EXIT"  ; Carrying traffic be
                           external destinatio

      BytesRead = 1*DIGIT
      BytesWritten = 1*DIGIT

   Controllers MUST tolerate unrecognized conn
```

BytesWritten and BytesRead are the number of byte last CONN_BW event on this connection.

These events are generated about once per second generated for connections that have not read or wri if TestingTorNetwork is set.

[First added in 0.2.5.2-alpha]

Bandwidth used by all streams attached t

The syntax is:

```
      "650" SP "CIRC_BW" SP "ID=" CircuitID SP
               "WRITTEN=" BytesWritten SP "TIM
               "DELIVERED_READ=" DeliveredByte
               "OVERHEAD_READ=" OverheadBytesR
               "DELIVERED_WRITTEN=" DeliveredB
               "OVERHEAD_WRITTEN=" OverheadByt
               "SS=" SlowStartState SP
               "CWND=" CWNDCells SP
               "RTT=" RTTMilliseconds SP
               "MIN_RTT=" RTTMilliseconds CRLF
      BytesRead = 1*DIGIT
      BytesWritten = 1*DIGIT
      OverheadBytesRead = 1*DIGIT
      OverheadBytesWritten = 1*DIGIT
      DeliveredBytesRead = 1*DIGIT
      DeliveredBytesWritten = 1*DIGIT
      SlowStartState = 0 or 1
      CWNDCells = 1*DIGIT
      RTTMilliseconds= 1*DIGIT
      Time = ISOTime2Frac
```

BytesRead and BytesWritten are the number of byte
since the last CIRC_BW event. These bytes have not
can include invalid cells, dropped cells, and ignored
values include the relay headers, but not circuit hea

Circuit data that has been validated and processed l
two categories: delivered payloads and overhead. D
DeliveredBytesWritten are the total relay cell payloa
event, not counting relay cell headers or circuit head
OverheadBytesWritten are the extra unused bytes a
be the fixed CELL_LEN bytes long.

The sum of DeliveredBytesRead and OverheadBytes
and the same is true for their written counterparts.
cell bytes on the circuit that have been validated by
cell headers. Subtracting this sum (plus relay cell he
BytesWritten) value gives the byte count that Tor ha
errors, or has otherwise decided to ignore.

The Time field is provided only in versions 0.3.2.1-al
created the bandwidth event.

The SS, CWND, RTT, and MIN_RTT fields are present
congestion control to an onion service or Exit hop (a
congestion control hops are not examined here). SS
in slow start (1), or not (0). CWND is the size of the c
of cells. RTT is the N_EWMA smoothed current RTT v
RTT value of the circuit. The SS and CWND fields app
the circuit. The slow start state and CWND values of

These events are generated about once per second
for circuits that had no attached stream writing or r

[First added in 0.2.5.2-alpha]

[DELIVERED_READ, OVERHEAD_READ, DELIVERED_W
were added in Tor 0.3.4.0-alpha]

[SS, CWND, RTT, and MIN_RTT were added in Tor 0.4

Per-circuit cell stats

The syntax is:

```
"650" SP "CELL_STATS"
         [ SP "ID=" CircuitID ]
         [ SP "InboundQueue=" QueueID SP
         [ SP "InboundAdded=" CellsByTyp
         [ SP "InboundRemoved=" CellsByT
              "InboundTime=" MsecByType
         [ SP "OutboundQueue=" QueueID S
         [ SP "OutboundAdded=" CellsByTy
         [ SP "OutboundRemoved=" CellsBy
              "OutboundTime=" MsecByType
CellsByType, MsecByType = CellType ":" 1
                          0*( "," CellTy
CellType = 1*( "a" - "z" / "0" - "9" / '
```

```
Examples are:

   650 CELL_STATS ID=14 OutboundQueue=19403
       OutboundAdded=create_fast:1,relay_ea
       OutboundRemoved=create_fast:1,relay_
       OutboundTime=create_fast:0,relay_ear
   650 CELL_STATS InboundQueue=19403 Inbour
       InboundAdded=relay:1,created_fast:1
       InboundRemoved=relay:1,created_fast:
       InboundTime=relay:0,created_fast:0
       OutboundQueue=6710 OutboundConn=18
       OutboundAdded=create:1,relay_early:1
       OutboundRemoved=create:1,relay_early
       OutboundTime=create:0,relay_early:0
```

ID is the locally unique circuit identifier that is only i node.

Inbound and outbound refer to the direction of cell either to origin (inbound) or from origin (outbound).

InboundQueue and OutboundQueue are identifiers queues of this circuit. These identifiers are only unic OutboundQueue is chosen by this node and matche the circuit.

InboundConn and OutboundConn are locally uniqu connection. OutboundConn does not necessarily m the circuit.

InboundQueue and InboundConn are not present if OutboundQueue and OutboundConn are not prese node.

InboundAdded and OutboundAdded are total numb inbound and outbound queues. Only present if at le

InboundRemoved and OutboundRemoved are total
processed from inbound and outbound queues. Inb
total waiting times in milliseconds of all processed c
least one cell was removed from a queue.

These events are generated about once per second
for circuits that have not added or processed any ce
TestingTorNetwork is set.

[First added in 0.2.5.2-alpha]

Token buckets refilled

The syntax is:

```
      "650" SP "TB_EMPTY" SP BucketName [ SP '
               "READ=" ReadBucketEmpty SP "WRI
               "LAST=" LastRefill CRLF

      BucketName = "GLOBAL" / "RELAY" / "ORCON
      ReadBucketEmpty = 1*DIGIT
      WriteBucketEmpty = 1*DIGIT
      LastRefill = 1*DIGIT

   Examples are:

      650 TB_EMPTY ORCONN ID=16 READ=0 WRITTEN
      650 TB_EMPTY GLOBAL READ=93 WRITTEN=93 L
      650 TB_EMPTY RELAY READ=93 WRITTEN=93 LA
```

This event is generated when refilling a previously e
"GLOBAL" and "RELAY" keywords are used for the gl
BucketName "ORCONN" is used for the token bucke
MUST tolerate unrecognized bucket names.

ConnID is only included if the BucketName is "ORCC

If both global and relay buckets and/or the buckets
out of tokens at the same time, multiple separate ev

ReadBucketEmpty (WriteBucketEmpty) is the time ir
was empty since the last refill. LastRefill is the time i

If a bucket went negative and if refilling tokens didn
be multiple consecutive TB_EMPTY events for each r
contained zero tokens or less. In such a case, ReadB
capped at LastRefill in order not to report empty tim

These events are only generated if TestingTorNetwc

[First added in 0.2.5.2-alpha]

HiddenService descriptors

The syntax is:

```
    "650" SP "HS_DESC" SP Action SP HSAddress
             [SP DescriptorID] [SP "REASON="
             [SP "HSDIR_INDEX=" HSDirIndex]

    Action =  "REQUESTED" / "UPLOAD" / "RECEI
              "FAILED" / "CREATED"
    HSAddress = 16*Base32Character / 56*Base3
    AuthType = "NO_AUTH" / "BASIC_AUTH" / "ST
    HsDir = LongName / Fingerprint / "UNKNOWN
    DescriptorID = 32*Base32Character / 43*Ba
    Reason = "BAD_DESC" / "QUERY_REJECTED" /
/
             "UNEXPECTED" / "QUERY_NO_HSDIR"
    Replica = 1*DIGIT
    HSDirIndex = 64*HEXDIG

    These events will be triggered when requ
    not found in the cache and a fetch or upl
performed.

    If the fetch was triggered with only a De
    command for instance), the HSAddress only
    since there is no way to know the HSAddre
    the value will be "UNKNOWN".

    If we already had the v0 descriptor, the
    will be ignored and a "HS_DESC" event wit
    generated.

    For HsDir, LongName is always preferred.
    list at the time event is sent, Fingerpri

    If Action is "FAILED", Tor SHOULD send Re
    values of Reason are:
       - "BAD_DESC" - descriptor was retrieve
       - "QUERY_REJECTED" - query was rejecte
       - "UPLOAD_REJECTED" - descriptor was r
       - "NOT_FOUND" - HS descriptor with giv
       - "UNEXPECTED" - nature of failure is
       - "QUERY_NO_HSDIR" - No suitable HSDir
       - "QUERY_RATE_LIMITED" - query for thi

    For "QUERY_NO_HSDIR" or "QUERY_RATE_LIMIT
    "UNKNOWN" which was introduced in tor 0.3
    respectively.

    If Action is "CREATED", Tor SHOULD send R
Replica
    field contains the replica number of the
Replica
    number is specified in rend-spec.txt sect
    descriptor ID of the descriptor.

    For hidden service v3, the following appl

       The "HSDIR_INDEX=" is an optional fiel
```

```
        which contains the computed index of t
        uploaded to or fetched from.

        The "DescriptorID" key is the descript
 index
        value at the "HsDir".

        The "REPLICA=" field is not used for t
        doesn't use the replica number in the

        Because client authentication is not y
        field is always "NO_AUTH".

    [HS v3 support added 0.3.3.1-alpha]
```

HiddenService descriptors content

The syntax is:

```
    "650" "+" "HS_DESC_CONTENT" SP HSAddress
              Descriptor CRLF "." CRLF "650"

    HSAddress = 16*Base32Character / 56*Base3
    DescId = 32*Base32Character / 32*Base64Ch
    HsDir = LongName / "UNKNOWN"
    Descriptor = The text of the descriptor t
                 rend-spec.txt section 1.3 (v
                 section 2.4 (v3) or empty st
```

This event is triggered when a successfully fetched H that descriptor is then replied. If the HS_DESC event RECEIVED action.

If a fetch fails, the Descriptor is an empty string and HS_DESC event should be used to get more informa

If the fetch fails for the QUERY_NO_HSDIR or QUERY HS_DESC event, the HsDir is set to "UNKNOWN". Thi 0.4.1.0-alpha respectively.

It's expected to receive a reply relatively fast as in it' over the Tor network. This can be between a couple hard limit). But, in any cases, this event will reply eit empty one.

[HS_DESC_CONTENT was added in Tor 0.2.7.1-alpha

Network liveness has changed

Syntax:

```
    "650" SP "NETWORK_LIVENESS" SP Status CR
    Status = "UP" /  ; The network now seems
             "DOWN" /  ; The network now see

 Controllers MUST tolerate unrecognized stat

 [NETWORK_LIVENESS was added in Tor 0.2.7.2-
```

Pluggable Transport Logs

Syntax:

"650" SP "PT_LOG" SP PT=Program SP Message

```
     Program = The program path as defined 
               configuration option. Tor acc
     Message = The log message that the PT s
               process minus the "LOG" strin
               specified in pt-spec.txt sect
               Transport Log Message".

  This event is triggered when tor receives

  Example:

     PT (obfs4): LOG SEVERITY=debug MESSAGE=

   the resulting control port event would be

     Tor: 650 PT_LOG PT=/usr/bin/obs4proxy S
to bridge A"

  [PT_LOG was added in Tor 0.4.0.1-alpha]
```

Pluggable Transport Status

Syntax:

"650" SP "PT_STATUS" SP PT=Program SP TRANSPOI

```
      Program = The program path as defined 
                configuration option. Tor acc
      Transport = This value indicates a hint
                  name or the protocol used 
      Message = The status message that the P
                process minus the "STATUS" st
                specified in pt-spec.txt sect
                Transport Status Message".

   This event is triggered when tor receives 

   Example:

      PT (obfs4): STATUS TRANSPORT=obfs4 CONN

   the resulting control port event would be:

      Tor: 650 PT_STATUS PT=/usr/bin/obs4prox
CONNECT=Success

   [PT_STATUS was added in Tor 0.4.0.1-alpha]
```

Implementation notes

Authentication

If the control port is open and no authentication ope
user that connects to the control port. This is genera

If the 'CookieAuthentication' option is true, Tor write
"control_auth_cookie" into its data directory (or to a
'CookieAuthFile' option). To authenticate, the contro
read the contents of the cookie file:

- Current versions of Tor support cookie authen

```
  using the "COOKIE" authentication method:
  contents of the cookie file, encoded in h
  authentication method exposes the user ru
  unintended information disclosure attack
  has greater filesystem read access than t
  connected to.  (Note that a controller ma
  other than Tor.)  It is almost never safe
  controller's user has explicitly specifie
  an authentication cookie from.  For this
  authentication method has been deprecated
  Tor before some future version of Tor.

* 0.2.2.x versions of Tor starting with 0.2

  Tor after 0.2.3.12-alpha, support cookie
  "SAFECOOKIE" authentication method, which
  information about the contents of the coo
```

If the 'HashedControlPassword' option is set, it mus
password. The salted hash is computed according to
(OpenPGP), and prefixed with the s2k specifier. This
prefixed by the indicator sequence "16:". Thus, for e
encode to:

```
  16:660537E3E1CD49996044A3BF558097A981F53
     ++++++++++++++++**^^^^^^^^^^^^^^^^^^^
        salt                           hashed
                        indicator
```

You can generate the salt of a password by calling

```
'tor --hash-password <password>'
```

or by using the example code in the Python and Jav
under this scheme, the controller sends Tor the orig
the password, either as a quoted string or encoded

Don't let the buffer get too big

With old versions of Tor (before 0.2.0.16-alpha), if y
them queue up on the buffer, the Tor process will c

Newer Tor versions do not have this 16 MB buffer li
numbers of events unread, Tor may still run out of
about buffer size.

Backward compatibility with v0

The 'version 0' control protocol was replaced in Tor
0.2.0.x. Every non-obsolete version of Tor now supp

For backward compatibility with the "version 0" con
whether the third octet of the first command is zer
is in use.)

This compatibility was removed in Tor 0.1.2.16 and

Tor config options for use by cor

Tor provides a few special configuration options for
not saved to disk by SAVECONF. Most can be set an
GETCONF commands, but some (noted below) can
command line.

Generally, these options make Tor unusable by disa
operations. Unless a controller provides replaceme
not correctly handle user requests.

__AllDirActionsPrivate

```
  If true, Tor will try to launch all direc
  anonymous connections.  (Ordinarily, Tor
  requests related to hidden services.)  Th
  directory access, and may stop Tor from w
  yet have enough directory information to

  (Boolean. Default: "0".)

__DisablePredictedCircuits

  If true, Tor will not launch preemptive '
  streams to attach to.  (It will still lau
  for hidden services.)

  (Boolean. Default: "0".)

__LeaveStreamsUnattached

  If true, Tor will not automatically attac
  instead, the controller must attach them
  controller does not attach the streams, t

  (Boolean. Default: "0".)

__HashedControlSessionPassword

  As HashedControlPassword, but is not save
  SAVECONF.  Added in Tor 0.2.0.20-rc.

__ReloadTorrcOnSIGHUP

  If this option is true (the default), we
  every time we get a SIGHUP (from the cont
  Otherwise, we don't.  This option exists
  their options from getting overwritten wh
  some other reason (for example, to rotate

  (Boolean.  Default: "1")

__OwningControllerProcess

  If this option is set to a process ID, To
  whether a process with the specified PID
  does not.  Added in Tor 0.2.2.28-beta.  T
  is documented in section 3.23 with the re
  command.

  Note that this option can only specify a
  the TAKEOWNERSHIP command which can be se
  connections.

  (String.  Default: unset.)

__OwningControllerFD

  If this option is a valid socket, Tor wil
```

```
      connection on this socket.  Added in Tor

      This socket will be an owning controller,
      TAKEOWNERSHIP.  It will be automatically
      should only be used by other programs tha

      This option cannot be changed via SETCONF
      via the command line.

      (Integer. Default: -1.)

    __DisableSignalHandlers

      If this option is set to true during star
      any signal handlers to watch for POSIX s
      command will still work.

      This option is meant for embedding Tor in
      the controlling process would rather hand

      This option cannot be changed via SETCONF
      via the command line.

      (Boolean. Default: 0.)
```

Phases from the Bootstrap statu

```
  [For the bootstrap phases reported by Tor pr
   Section 5.6.]
```

This section describes the various bootstrap phases should not assume that the percentages and tags lis even that the tags will stay in the same order. Some reported) if the associated bootstrap step is already necessary. Only "starting" and "done" are guarantee

Current Tor versions enter these phases in order, m earlier phases, for example, if the network fails.

Overview of Bootstrap reporting

Bootstrap phases can be viewed as belonging to one

1. Initial connection to a Tor relay or bridge
2. Obtaining directory information
3. Building an application circuit

Tor doesn't specifically enter Stage 1; that is a side e
taking. Tor could be making a connection to a fallba
making a connection to a guard candidate. Either or
of bootstrap reporting.

Stage 2 might involve Tor contacting directory serve
directory information from a previous session. Larg
there is already enough cached directory informatic
reporting progress in Stage 2 until Stage 1 is comple

Tor defers this reporting because Tor can already ha
build circuits, yet not be able to connect to a relay. V
misleadingly see Tor stuck at a large amount of prog
fundamental as making a TCP connection to any rel

Tor also doesn't specifically enter Stage 3; that is a s
some purpose or other. In a typical client, Tor builds
latency for application connection requests. In Stage
to relays or bridges that it did not connect to in Stag

Phases in Bootstrap Stage 1

Phase 0: tag=starting summary="Starting"

Tor starts out in this phase.

```
 Phase 1:
 tag=conn_pt summary="Connecting to pluggabl
[This phase is new in 0.4.0.x]
```

Tor is making a TCP connection to the transport plu
use this pluggable transport to make its first connec

```
 Phase 2:
 tag=conn_done_pt summary="Connected to plug
[New in 0.4.0.x]
```

Tor has completed its TCP connection to the transpo

```
 Phase 3:
 tag=conn_proxy summary="Connecting to proxy
[New in 0.4.0.x]
```

Tor is making a TCP connection to a proxy to make i

```
Phase 4:
tag=conn_done_proxy summary="Connected to p
[New in 0.4.0.x]
```

Tor has completed its TCP connection to a proxy to bridge.

```
Phase 5:
tag=conn summary="Connecting to a relay"
[New in 0.4.0.x; prior versions of Tor had a
sometimes but not always corresponded to co
```

Tor is making its first connection to a relay. This mig or proxy connection that Tor has already establishe

```
Phase 10:
tag=conn_done summary="Connected to a relay
[New in 0.4.0.x]

Tor has completed its first connection to a

Phase 14:
tag=handshake summary="Handshaking with a r
[New in 0.4.0.x; prior versions of Tor had a

Tor is in the process of doing a TLS handsh

Phase 15:
tag=handshake_done summary="Handshake with
[New in 0.4.0.x]

Tor has completed its TLS handshake with a
```

Phases in Bootstrap Stage 2

```
Phase 20:
tag=onehop_create summary="Establishing an
[prior to 0.4.0.x, this was numbered 15]
```

Once TLS is finished with a relay, Tor will send a CRE circuit for retrieving directory information. It will ren CREATED_FAST cell back, indicating that the circuit is

```
Phase 25:
tag=requesting_status summary="Asking for n
[prior to 0.4.0.x, this was numbered 20]
```

Once we've finished our one-hop circuit, we will star
networkstatus consensus. We'll stay in this phase ur
back, indicating that we've established a directory co

```
 Phase 30:
 tag=loading_status summary="Loading network
[prior to 0.4.0.x, this was numbered 25]
```

Once we've established a directory connection, we v
consensus document. This could take a while; this p
the "progress" keyword to indicate partial progress.

This phase could stall if the directory server we pick
networkstatus consensus so we have to ask another
don't find it valid.

Phase 40: tag=loading_keys summary="Loading autl

Sometimes when we've finished loading the networ
don't have all the authority key certificates for the k
point we put the consensus we fetched on hold and
signatures.

Phase 45 tag=requesting_descriptors summary="As

Once we have a valid networkstatus consensus and
start asking for relay descriptors. We stay in this pha
'connected' relay cell in response to a request for de

```
[Some versions of Tor (starting with 0.2.6.2
 0.4.0.x): Tor could report having internal
 5.6]
```

Phase 50: tag=loading_descriptors summary="Loadi

We will ask for relay descriptors from several differe
make up the bulk of the bootstrapping, especially fo
stay in this phase until we have descriptors for a sig
listed in the networkstatus consensus (this can be b
Tor's configuration and network consensus parame
opportunity to use the "progress" keyword to indica

```
[Some versions of Tor (starting with 0.2.6.2
 0.4.0.x): Tor could report having internal
 5.6]

 Phase 75:
 tag=enough_dirinfo summary="Loaded enough c
 circuits"
[New in 0.4.0.x; previously, Tor would misle
 "conn_or" tag once it had enough directory
```

Phases in Bootstrap Stage 3

```
 Phase 76:
 tag=ap_conn_pt summary="Connecting to plugg
 circuits"
[New in 0.4.0.x]
```

This is similar to conn_pt, except for making connec
that Tor needs to use to build application circuits.

```
  Phase 77:
  tag=ap_conn_done_pt summary="Connected to p
circuits"
 [New in 0.4.0.x]
```

This is similar to conn_done_pt, except for making c
bridges that Tor needs to use to build application ci

```
 Phase 78:
 tag=ap_conn_proxy summary="Connecting to pr
[New in 0.4.0.x]
```

This is similar to conn_proxy, except for making con
bridges that Tor needs to use to build application ci

```
 Phase 79:
 tag=ap_conn_done_proxy summary="Connected t
[New in 0.4.0.x]
```

This is similar to conn_done_proxy, except for makir
bridges that Tor needs to use to build application ci

```
 Phase 80:
 tag=ap_conn summary="Connecting to a relay
[New in 0.4.0.x]
```

This is similar to conn, except for making connectior
Tor needs to use to build application circuits.

```
  Phase 85:
  tag=ap_conn_done summary="Connected to a re
 [New in 0.4.0.x]
```

This is similar to conn_done, except for making conr
bridges that Tor needs to use to build application cir

```
  Phase 89:
  tag=ap_handshake summary="Finishing handsha
circuits"
 [New in 0.4.0.x]
```

This is similar to handshake, except for making conr
bridges that Tor needs to use to build application cir

```
  Phase 90:
  tag=ap_handshake_done summary="Handshake f
circuits"
 [New in 0.4.0.x]
```

This is similar to handshake_done, except for makin
bridges that Tor needs to use to build application cir

```
  Phase 95:
  tag=circuit_create summary="Establishing a
 [prior to 0.4.0.x, this was numbered 90]
```

Once we've finished our TLS handshake with the firs
trying to make some 3-hop circuits in case we need

```
 [Some versions of Tor (starting with 0.2.6.2
  0.4.0.x): Tor could report having internal
  5.6]
```

Phase 100: tag=done summary="Done"

A full 3-hop circuit has been established. Tor is read
now.

```
 [Some versions of Tor (starting with 0.2.6.2
  0.4.0.x): Tor could report having internal
  5.6]
```

Bootstrap phases reported by ol

These phases were reported by Tor older than 0.4.0
Section 5.5.

```
[Newer versions of Tor (0.2.6.2-alpha and la
 If the consensus contains Exits (the typica
 exit and internal circuits. When bootstrap
 to handle an application requesting an exit
 World Wide Web.
```

If the consensus does not contain Exits, Tor will only
earlier statuses will have included "internal" as indic
completes, Tor will be ready to handle an applicatio
hidden services at ".onion" addresses.

If a future consensus contains Exits, exit circuits ma

Phase 0: tag=starting summary="Starting"

Tor starts out in this phase.

Phase 5: tag=conn_dir summary="Connecting to dir

Tor sends this event as soon as Tor has chosen a dir
authorities if bootstrapping for the first time or afte
relays listed in its cached directory information othe

Tor will stay at this phase until it has successfully es
directory server. Problems in this phase generally ha
network connection, or because the local firewall is

Phase 10: tag=handshake_dir summary="Finishing h

This event occurs when Tor establishes a TCP conne
a directory server (or its https proxy if it's using one)
TLS handshake with the relay or authority is finished

Problems in this phase generally happen because To
sophisticated MITM attacks on it, or doing packet-le
handshake.

Phase 15: tag=onehop_create summary="Establishin

Once TLS is finished with a relay, Tor will send a CRE
circuit for retrieving directory information. It will ren
CREATED_FAST cell back, indicating that the circuit is

Phase 20: tag=requesting_status summary="Asking

Once we've finished our one-hop circuit, we will star
networkstatus consensus. We'll stay in this phase ur
back, indicating that we've established a directory co

Phase 25: tag=loading_status summary="Loading ne

Once we've established a directory connection, we w
consensus document. This could take a while; this p
the "progress" keyword to indicate partial progress.

This phase could stall if the directory server we pick
networkstatus consensus so we have to ask another
don't find it valid.

Phase 40: tag=loading_keys summary="Loading autl

Sometimes when we've finished loading the networ
don't have all the authority key certificates for the k
point we put the consensus we fetched on hold and
signatures.

```
Phase 45
tag=requesting_descriptors summary="Asking
                                    [ for
```

Once we have a valid networkstatus consensus and
start asking for relay descriptors. We stay in this pha
'connected' relay cell in response to a request for de

```
[Newer versions of Tor (0.2.6.2-alpha and la
 If the consensus contains Exits (the typica
 descriptors for both exit and internal path
 for descriptors for internal paths. In this
 include "internal" as indicated above.]

 Phase 50:
 tag=loading_descriptors summary="Loading re
                                  paths]"
```

We will ask for relay descriptors from several differe
make up the bulk of the bootstrapping, especially fo
stay in this phase until we have descriptors for a sig
listed in the networkstatus consensus (this can be b
Tor's configuration and network consensus parame
opportunity to use the "progress" keyword to indica

```
[Newer versions of Tor (0.2.6.2-alpha and la
 If the consensus contains Exits (the typica
 descriptors for both exit and internal path
 download descriptors for internal paths. In
 include "internal" as indicated above.]
```

Phase 80: tag=conn_or summary="Connecting to th

Once we have a valid consensus and enough relay c
and start trying to build some circuits. This step is si
the only difference is the context.

If a Tor starts with enough recent cached directory i
event will be for the conn_or phase.

```
[Newer versions of Tor (0.2.6.2-alpha and la
 If the consensus contains Exits (the typica
 exit and internal circuits. If not, Tor wil
 In this case, this status will include "int

 Phase 85:
 tag=handshake_or summary="Finishing handsha
                          circuit]"
```

This phase is similar to the "handshake_dir" phase, l
connection to a Tor relay and we have already reach
this phase until we complete a TLS handshake with a

```
[Newer versions of Tor (0.2.6.2-alpha and la
 If the consensus contains Exits (the typica
 a handshake with the first hop if either an
 this case, it won't specify which type. If
 Tor will only build internal circuits. In t
 include "internal" as indicated above.]
```

Phase 90: tag=circuit_create summary="Establishing

Once we've finished our TLS handshake with the firs
trying to make some 3-hop circuits in case we need

```
[Newer versions of Tor (0.2.6.2-alpha and la
 If the consensus contains Exits (the typica
 exit and internal circuits. If not, Tor wil
 In this case, this status will include "int
```

Phase 100: tag=done summary="Done"

A full 3-hop circuit has been established. Tor is read
now.

```
[Newer versions of Tor (0.2.6.2-alpha and la
 If the consensus contains Exits (the typica
 exit and internal circuits. At this stage,
 an application requesting an exit circuit t
 Wide Web.
```

If the consensus does not contain Exits, Tor will only earlier statuses will have included "internal" as indic ready to handle an application requesting an intern addresses.

If a future consensus contains Exits, exit circuits ma

How Tor Version Numbe

The Old Way

Before 0.1.0, versions were of the format:

MAJOR.MINOR.MICRO(status(PATCHLEVEL))?(-cvs)?

where MAJOR, MINOR, MICRO, and PATCHLEVEL are
an alpha release), "rc" (for a release candidate), or ".
"a.b.c" was equivalent to "a.b.c.0". We compare the
micro, status, patchlevel, cvs), with "cvs" preceding r

We would start each development branch with a fin
first pre-release would be "0.0.8pre1", followed by (f
"0.0.8pre2", "0.0.8pre3-cvs", "0.0.8rc1", "0.0.8rc2-cvs
0.0.8. The stable CVS branch would then be versione
bugfix release would be "0.0.8.1".

The New Way

Starting at 0.1.0.1-rc, versions are of the format:

```
MAJOR.MINOR.MICRO[.PATCHLEVEL][-STATUS_TAG][
```

The stuff in parentheses is optional. As before, MAJ(
are numbers, with an absent number equivalent to
distinguishable purely by those four numbers.

The STATUS_TAG is purely informational, and lets yo
release is: "alpha" is pretty unstable; "rc" is a release
that we have a final release. If the tag ends with "-cv
development snapshot that came after a given relea
that differ only by status tag, we compare them lexi
whitespace.

The EXTRA_INFO is also purely informational, often
commit this version came from. It is surrounded by
whitespace. Unlike the STATUS_TAG this never impa
compared. EXTRA_INFO may appear any number of
parse EXTRA_INFO entries.

Now, we start each development branch with (say) 0
increments consistently as the status tag changes, f
0.1.1.3-alpha, 0.1.1.4-rc, 0.1.1.5-rc. Eventually, we re
0.1.1.7.

Between these releases, CVS is versioned with a -cvs
0.1.1.1-alpha-cvs, and so on. But starting with 0.1.2.
started using the "-dev" suffix instead of the "-cvs" s

Version status

```
Sometimes we need to determine whether a To
experimental, or neither, based on a list o
logic is as follows:

 * If a version is listed on the recommende
   "recommended".

 * If a version is newer than every recomme
   is "experimental" or "new".

 * If a version is older than every recomme
   "obsolete" or "old".

 * The first three components (major,minor,
   are its "release series".  If a version
   versions with the same release series, a
   than all such recommended versions, but
   _every_ recommended version, then the ve

 * Finally, if none of the above conditions
   "un-recommended."
```

Tor Bandwidth File Form

juga
teor

This document describes the format of Tor's Bandw

It is a new specification for the existing bandwidth fi
It also specifies new format versions 1.1.0 and later,
with 1.0.0 parsers.

Since Tor version 0.2.4.12-alpha, the directory autho
called "V3BandwidthsFile" generated by Torflow [1].
described in Torflow's README.spec.txt. We also su
specification.

Scope and preliminaries

The key words "MUST", "MUST NOT", "REQUIRED", "
"SHOULD NOT", "RECOMMENDED", "MAY", and "OP
interpreted as described in RFC 2119.

Acknowledgements

The original bandwidth generator (Torflow) and forr
suggested to write this specification while contributi
generator implementation.

This specification was revised after feedback from:

Nick Mathewson (nickm) Iain Learmonth (irl)

Outline

The Tor directory protocol (dir-spec.txt [3]) sections
bandwidth measurements, to refer to what here is c

A Bandwidth File contains information on relays' ba
bandwidth generators, previously known as bandwi

Format Versions

1.0.0 - The legacy Bandwidth File format

```
1.1.0 - Adds a header containing informatic
        file. Document the sbws and Torflow

1.2.0 - If there are not enough eligible re
        SHOULD contain a header, but no rel
        existing behaviour.)

        Adds scanner and destination countr
        Adds new KeyValue Lines to the Head
        statistics about the number of rela
        Adds new KeyValues to Relay Bandwid
        bandwidth values (averages and desc

1.4.0 - Adds monitoring KeyValues to the he

        RelayLines for excluded relays MAY
        file for diagnostic reasons. Simila
        eligible relays, the bandwidth file

        Diagnostic relay lines SHOULD be ma
        Tor SHOULD NOT use their bandwidths

        Also adds Tor version.
1.5.0 - Removes "recent_measurement_attempt
1.6.0 - Adds congestion control stream ever
1.7.0 - Adds ratios KeyValues to the relay
        KeyValues to the header.
1.8.0 - Adds "dirauth_nickname" KeyValue to
1.9.0 - Allows "node_id" KeyValue without t
the
        hexdigit.
All Tor versions can consume format versior
```

All Tor versions can consume format version 1.1.0 a
0.3.5.1-alpha warn if the header contains any KeyVa

```
Tor versions 0.4.0.3-alpha, 0.3.5.8, 0.3.4.
understand "vote=0". Instead, they will vot
that sbws puts in diagnostic relay lines:
  * 1 for relays with "unmeasured=1", and
  * the relay's measured and scaled bandwic
```

Format details

```
The Bandwidth File MUST contain the follow
- Header List (exactly once), which is a pa
  - Header Lines (one or more times), then
- Relay Lines (zero or more times), in an a
If it does not contain these sections, pars
```

Definitions

The following nonterminals are defined in Tor direct
2.1.3.:

```
    bool
    Int
    SP (space)
    NL (newline)
    KeywordChar
    ArgumentChar
    nickname
    hexdigest (a '$', followed by 40 hexadec
      ([A-Fa-f0-9]))

  Nonterminal defined section 2 of version-sp

    version_number

  We define the following nonterminals:

    Line ::= ArgumentChar* NL
    RelayLine ::= KeyValue (SP KeyValue)* NL
    HeaderLine ::= KeyValue NL
    KeyValue ::= Key "=" Value
    Key ::= (KeywordChar | "_")+
    Value ::= ArgumentCharValue+
    ArgumentCharValue ::= any printing ASCII
    Terminator ::= "=====" or "===="
                   Generators SHOULD use a 5-
    Timestamp ::= Int
    Bandwidth ::= Int
    MasterKey ::= a base64-encoded Ed25519 pu
                  padding characters omitted.
    DateTime ::= "YYYY-MM-DDTHH:MM:SS", as in
    CountryCode ::= Two capital ASCII letters
                    ISO 3166-1 alpha-2 plus '
                    (eg the destination is in
    CountryCodeList ::= One or more CountryCo
                        ([A-Z]{2}(,[A-Z]{2})*
```

Note that key_value and value are defined in Tor dir
to KeyValue and Value here.

Tor versions earlier than 0.3.5.1-alpha require all lin
less. The previous limit was 254 characters in Tor 0.
ignore longer Lines.

Note that directory authorities are only supported c
versions, so we expect that line limits will be remove

Header List format

It consists of a Timestamp line and zero or more He

All the header lines MUST conform to the HeaderLir
line.

The Timestamp line is not a HeaderLine to keep con
File format.

Some header Lines MUST appear in specific positior
Lines can appear in any order.

If a parser does not recognize any extra material in
ignored.

If a header Line does not conform to this format, th

It consists of:

Timestamp NL

[At start, exactly once.]

The Unix Epoch time in seconds of the most recent

If the generator implementation has multiple threac
independently, it SHOULD take the most recent time
oldest value. This ensures all the threads continue r

If there are threads that do not run continuously, th
timestamp calculation.

If there are no recent results, the generator MUST N

It does not follow the KeyValue format for backwarc

"version" version_number NL

[In second position, zero or one time.]

The specification document format version. It uses s

This Line was added in version 1.1.0 of this specifica

Version 1.0.0 documents do not contain this Line, ar
to be "1.0.0".

"software" Value NL

[Zero or one time.]

The name of the software that created the documer

This Line was added in version 1.1.0 of this specifica

Version 1.0.0 documents do not contain this Line, ar "torflow".

"software_version" Value NL

[Zero or one time.]

The version of the software that created the docum version_number, a git commit, or some other versio

This Line was added in version 1.1.0 of this specifica

"file_created" DateTime NL

[Zero or one time.]

The date and time timestamp in ISO 8601 format ar created.

This Line was added in version 1.1.0 of this specifica

"generator_started" DateTime NL

[Zero or one time.]

The date and time timestamp in ISO 8601 format ar started.

This Line was added in version 1.1.0 of this specifica

"earliest_bandwidth" DateTime NL

[Zero or one time.]

The date and time timestamp in ISO 8601 format ar bandwidth was obtained.

This Line was added in version 1.1.0 of this specifica

"latest_bandwidth" DateTime NL

[Zero or one time.]

The date and time timestamp in ISO 8601 format ar generator bandwidth result.

This time MUST be identical to the initial Timestamp

This duplicate value is included to make the format

This Line was added in version 1.1.0 of this specifica

"number_eligible_relays" Int NL

[Zero or one time.]

The number of relays that have enough measureme
file.

This Line was added in version 1.2.0 of this specifica

"minimum_percent_eligible_relays" Int NL

[Zero or one time.]

The percentage of relays in the consensus that SHO
bandwidth file.

If this threshold is not reached, format versions 1.3.
any relays. (Bandwidth files always include a header

Format versions 1.4.0 and later SHOULD include all
even if this threshold is not reached. But these relay
does not vote on them. See section 1.4 for details.

The minimum percentage is 60% in Torflow, so sbws

This Line was added in version 1.2.0 of this specifica

"number_consensus_relays" Int NL

[Zero or one time.]

The number of relays in the consensus.

This Line was added in version 1.2.0 of this specifica

"percent_eligible_relays" Int NL

[Zero or one time.]

The number of eligible relays, as a percentage of the

```
    This line SHOULD be equal to:
        (number_eligible_relays * 100.0) /
    to the number of relays in the consensu

    This Line was added in version 1.2.0 of

  "minimum_number_eligible_relays" Int NL

    [Zero or one time.]
```

The minimum number of relays that SHOULD be inc
minimum_percent_eligible_relays for details.

```
    This line SHOULD be equal to:
        number_consensus_relays * (minimum_

    This Line was added in version 1.2.0 of

  "scanner_country" CountryCode NL

    [Zero or one time.]

    The country, as in political geolocatic

    This Line was added in version 1.2.0 of

  "destinations_countries" CountryCodeList

    [Zero or one time.]
```

The country, as in political geolocation, or countries
are located. The destination Web Servers serve the
measure the bandwidth.

This Line was added in version 1.2.0 of this specifica

"recent_consensus_count" Int NL

[Zero or one time.].

The number of the different consensuses seen in th
is 5 by default.)

```
        Assuming that Tor clients fetch a conse
        and that the data_period is 5 days, the
        between:
            data_period * 24 / 2 =  60
            data_period * 24     = 120

        This Line was added in version 1.4.0 of

      "recent_priority_list_count" Int NL

        [Zero or one time.]
```

The number of times that a list with a subset of rela
been created in the last data_period days. (data_per

```
        In 2019, with 7000 relays in the networ
 be
        approximately:
            data_period * 24 / 1.5 = 80
        Being 1.5 the approximate number of hou
        priority list of 7000 * 0.05 (350) rela
        in a priority list is the 5% (0.05).

        This Line was added in version 1.4.0 of

      "recent_priority_relay_count" Int NL

        [Zero or one time.]
```

The number of relays that has been in in the list of r
the last data_period days. (data_period is 5 by defau

```
        In 2019, with 7000 relays in the networ
 be
        approximately:
            80 * (7000 * 0.05) = 28000
        Being 0.05 (5%) the fraction of relays
        the approximate number of priority list
        "recent_priority_list_count").

        This Line was added in version 1.4.0 of

      "recent_measurement_attempt_count" Int NL

        [Zero or one time.]
```

The number of times that any relay has been queue
data_period days. (data_period is 5 by default.)

In 2019, with 7000 relays in the network, the Value c
the same as "recent_priority_relay_count", assuming

a relay for each relay that has been prioritized unles
implementation issues.

This Line was added in version 1.4.0 of this specifica

"recent_measurement_failure_count" Int NL

[Zero or one time.]

The number of times that the scanner attempted to
data_period days (5 by default), but the relay has no
network or implementation issues.

This Line was added in version 1.4.0 of this specifica

"recent_measurements_excluded_error_count" Int N

[Zero or one time.]

The number of relays that have no successful meas
(5 by default).

(See the note in section 1.4, version 1.4.0, about exc

This Line was added in version 1.4.0 of this specifica

"recent_measurements_excluded_near_count" Int N

[Zero or one time.]

The number of relays that have some successful me
days (5 by default), but all those measurements wer
was too short (by default 1 day).

(See the note in section 1.4, version 1.4.0, about exc

This Line was added in version 1.4.0 of this specifica

"recent_measurements_excluded_old_count" Int NL

[Zero or one time.]

The number of relays that have some successful me
measurements are too old (more than 5 days, by de

Excludes relays that are already counted in recent_r

(See the note in section 1.4, version 1.4.0, about exc

This Line was added in version 1.4.0 of this specifica "recent_measurements_excluded_few_count" Int NL

[Zero or one time.]

The number of relays that don't have enough recen
than 2 measurements in the last 5 days, by default).

Excludes relays that are already counted in recent_r
and recent_measurements_excluded_old_count.

(See the note in section 1.4, version 1.4.0, about exc

This Line was added in version 1.4.0 of this specifica

"time_to_report_half_network" Int NL

[Zero or one time.]

The time in seconds that it would take to report mea
network, given the number of eligible relays and the
by default).

(See the note in section 1.4, version 1.4.0, about exc

This Line was added in version 1.4.0 of this specifica

"tor_version" version_number NL

[Zero or one time.]

The Tor version of the Tor process controlled by the

This Line was added in version 1.4.0 of this specifica

"mu" Int NL

[Zero or one time.]

The network stream bandwidth average calculated a

This Line was added in version 1.7.0 of this specifica

"muf" Int NL

[Zero or one time.]

The network stream bandwidth average filtered calc

This Line was added in version 1.7.0 of this specifica

KeyValue NL

[Zero or more times.]

"dirauth_nickname" NL

[Zero or one time.]

The dirauth's nickname which publishes this V3Banc

This Line was added in version 1.8.0 of this specifica

There MUST NOT be multiple KeyValue header Lines
parser SHOULD choose an arbitrary Line.

If a parser does not recognize a Keyword in a KeyVa

Future format versions may include additional KeyV
Lines will be accompanied by a minor version increr

Implementations MAY add additional header Lines a
be updated to avoid conflicting meanings for the sa

Parsers MUST NOT rely on the order of these additi

Additional header Lines MUST NOT use any keyword
measurements format. If there are, the parser MAY

Terminator NL

[Zero or one time.]

The Header List section ends with a Terminator.

In version 1.0.0, Header List ends when the first rela
the next section.

Implementations of version 1.1.0 and later SHOULD

Tor 0.4.0.1-alpha and later look for a 5-character ter
line. sbws versions 0.1.0 to 1.0.2 used a 4-character
1.0.3.

Relay Line format

It consists of zero or more RelayLines containing rel
their KeyValues are in arbitrary order.

There MUST NOT be multiple KeyValue pairs with th
there are, the parser SHOULD choose an arbitrary V

There MUST NOT be multiple RelayLines per relay ic
master_key_ed25519). If there are, parsers SHOULD
the file, choose an arbitrary RelayLine, or ignore bot

If a parser does not recognize any extra material in
be ignored.

Each RelayLine includes the following KeyValue pair

"node_id" hexdigest

[Exactly once.]

The fingerprint for the relay's RSA identity key.

```
Note: In bandwidth files read by Tor ve
      0.3.4.1-alpha, node_id MUST NOT b
      These authority versions are no l
```

Current Tor versions ignore master_key_ed25519, s
relay Line.

Implementations of version 1.1.0 and later SHOULD
master_key_ed25519. Parsers SHOULD accept Lines

From version 1.9.0 of this specification, "node_id" d
dollar sign before the hexdigit.

"master_key_ed25519" MasterKey

[Zero or one time.]

The relays's master Ed25519 key, base64 encoded,
ambiguity with KeyValue "=" character.

This KeyValue pair SHOULD be present, see the not

This KeyValue was added in version 1.1.0 of this spe

"bw" Bandwidth

[Exactly once.]

The bandwidth of this relay in kilobytes per second.

No Zero Bandwidths: Tor accepts zero bandwidths, implementations. Therefore, implementations SHOU Instead, they SHOULD use one as their minimum ba bandwidths, the parser MAY ignore them.

Bandwidth Aggregation: Multiple measurements ca scheme, such as a mean, median, or decaying avera

Bandwidth Scaling: Torflow scales bandwidths to kil implementations SHOULD use kilobytes per second

If different implementations or configurations are u their measurements MAY need further scaling. See scaling, and one possible scaling method.

MaxAdvertisedBandwidth: Bandwidth generators M bandwidth based on the MaxAdvertisedBadwidth. A limits the bandwidth-avg in its descriptor. bandwidt MaxAdvertisedBandwidth, BandwidthRate, RelayBa RelayBandwidthBurst. Therefore, generators MUST its descriptor's bandwidth-avg. This limit needs to b because generators may scale consensus weights b Generators SHOULD NOT limit measured bandwidt observed, because that penalises new relays.

sbws limits the relay's measured bandwidth to the b

Torflow partitions relays based on their bandwidth. the minimum of all descriptor bandwidths, including (MaxAdvertisedBandwidth) and bandwidth-observe in each partition against each other, which implicitly to the bandwidths of similar relays.

Torflow also generates consensus weights based or bandwidth and the minimum of all descriptor band measurement). So when an operator reduces the M Torflow reduces that relay's measured bandwidth.

KeyValue

[Zero or more times.]

Future format versions may include additional KeyV

KeyValue pairs will be accompanied by a minor vers

Implementations MAY add additional relay KeyValu
SHOULD be updated to avoid conflicting meanings f

Parsers MUST NOT rely on the order of these additi

Additional KeyValue pairs MUST NOT use any keywo
there are, the parser MAY ignore conflicting keywor

Implementation details

Writing bandwidth files atomica

To avoid inconsistent reads, implementations SHOU
the file is transferred from another host, it SHOULD
renamed to the V3BandwidthsFile path.

sbws versions 0.7.0 and later write the bandwidth fi
temporary symlink to that location, then atomically
V3BandwidthsFile path.

Torflow does not write bandwidth files atomically.

Additional KeyValue pair definiti

KeyValue pairs in RelayLines that current implemen

Simple Bandwidth Scanner

sbws RelayLines contain these keys:

"node_id" hexdigest

As above.

"bw" Bandwidth

As above.

"nick" nickname

[Exactly once.]

The relay nickname.

Torflow also has a "nick" KeyValue.

"rtt" Int

[Zero or one time.]

The Round Trip Time in milliseconds to obtain 1 byt

This KeyValue was added in version 1.1.0 of this spe
version 1.3.0 or 1.4.0 of this specification.

"time" DateTime

[Exactly once.]

The date and time timestamp in ISO 8601 format ar
bandwidth was obtained.

This KeyValue was added in version 1.1.0 of this spe
"measured_at".

"success" Int

[Zero or one time.]

The number of times that the bandwidth measurem

This KeyValue was added in version 1.1.0 of this spe

"error_circ" Int

[Zero or one time.]

The number of times that the bandwidth measurem
circuit failures.

This KeyValue was added in version 1.1.0 of this spe
"circ_fail".

"error_stream" Int

[Zero or one time.]

The number of times that the bandwidth measurem
stream failures.

This KeyValue was added in version 1.1.0 of this spe

"error_destination" Int

[Zero or one time.]

The number of times that the bandwidth measurem
destination Web server was not available.

This KeyValue was added in version 1.4.0 of this spe

"error_second_relay" Int

[Zero or one time.]

The number of times that the bandwidth measurem
sbws could not find a second relay for the test circu

This KeyValue was added in version 1.4.0 of this spe

"error_misc" Int

[Zero or one time.]

The number of times that the bandwidth measurem
other reasons.

This KeyValue was added in version 1.1.0 of this spe

"bw_mean" Int

[Zero or one time.]

The measured bandwidth mean for this relay in byt

This KeyValue was added in version 1.2.0 of this spe

"bw_median" Int

[Zero or one time.]

The measured bandwidth median for this relay in by

This KeyValue was added in version 1.2.0 of this spe

"desc_bw_avg" Int

[Zero or one time.]

The descriptor average bandwidth for this relay in b

This KeyValue was added in version 1.2.0 of this spe

"desc_bw_obs_last" Int

[Zero or one time.]

The last descriptor observed bandwidth for this rela

This KeyValue was added in version 1.2.0 of this spe

"desc_bw_obs_mean" Int

[Zero or one time.]

The descriptor observed bandwidth mean for this re

This KeyValue was added in version 1.2.0 of this spe

"desc_bw_bur" Int

[Zero or one time.]

The descriptor burst bandwidth for this relay in byte

This KeyValue was added in version 1.2.0 of this spe

"consensus_bandwidth" Int

[Zero or one time.]

The consensus bandwidth for this relay in bytes per

This KeyValue was added in version 1.2.0 of this spe

"consensus_bandwidth_is_unmeasured" Bool

[Zero or one time.]

If the consensus bandwidth for this relay was not ob
bandwidth authorities, this KeyValue is True or False

This KeyValue was added in version 1.2.0 of this spe

"relay_in_recent_consensus_count" Int

[Zero or one time.]

The number of times this relay was found in a conse
(Unless otherwise stated, data_period is 5 by defaul

This KeyValue was added in version 1.4.0 of this spe

"relay_recent_priority_list_count" Int

[Zero or one time.]

The number of times this relay has been prioritized
data_period days.

This KeyValue was added in version 1.4.0 of this spe

"relay_recent_measurement_attempt_count" Int

[Zero or one time.]

The number of times this relay was tried to be meas

This KeyValue was added in version 1.4.0 of this spe

"relay_recent_measurement_failure_count" Int

[Zero or one time.]

The number of times this relay was tried to be meas
it was not possible to obtain a measurement.

This KeyValue was added in version 1.4.0 of this spe

"relay_recent_measurements_excluded_error_count

[Zero or one time.]

The number of recent relay measurement attempts
if they are in the last data_period days (5 by default)

(See the note in section 1.4, version 1.4.0, about exc

This KeyValue was added in version 1.4.0 of this spe

"relay_recent_measurements_excluded_near_count

[Zero or one time.]

When all of a relay's recent successful measuremen
that was too short (by default 1 day), the relay is exc
number of recent successful measurements for the
reason.

(See the note in section 1.4, version 1.4.0, about exc

This KeyValue was added in version 1.4.0 of this spe

"relay_recent_measurements_excluded_old_count"

[Zero or one time.]

The number of successful measurements for this re
data_period days, 5 by default).

Excludes measurements that are already counted ir
relay_recent_measurements_excluded_near_count.

(See the note in section 1.4, version 1.4.0, about exc

This KeyValue was added in version 1.4.0 of this spe

"relay_recent_measurements_excluded_few_count"

[Zero or one time.]

The number of successful measurements for this re
relay did not have enough successful measurement

Excludes measurements that are already counted ir
relay_recent_measurements_excluded_near_count (
relay_recent_measurements_excluded_old_count.

(See the note in section 1.4, version 1.4.0, about exc

This KeyValue was added in version 1.4.0 of this spe

"under_min_report" bool

[Zero or one time.]

If the value is 1, there are not enough eligible relays
bandwidth authorities MAY NOT vote on this relay. (
their behaviour based on the "under_min_report" ke

If the value is 0 or the KeyValue is not present, there
file.

Because Tor versions released before April 2019 (se
versions) ignore "vote=0", generator implementatio
for under_min_report relays. Using the same bw val
understand "vote=0" or "under_min_report=1" prod
weights too much. It also avoids flapping when the r

This KeyValue was added in version 1.4.0 of this spe

"unmeasured" bool

[Zero or one time.]

If the value is 1, this relay was not successfully meas
MAY NOT vote on this relay. (Current Tor versions d
on the "unmeasured" key.)

If the value is 0 or the KeyValue is not present, this r

Because Tor versions released before April 2019 (se
versions) ignore "vote=0", generator implementatio
relays. Using the minimum bw value makes authorit

or "unmeasured=1" produce votes that don't chang

This KeyValue was added in version 1.4.0 of this spe

"vote" bool

[Zero or one time.]

If the value is 0, Tor directory authorities SHOULD ig
bandwidth file. They SHOULD vote for the relay the
that is not present in the file.

This MAY be the case when this relay was not succes
the Bandwidth File, to diagnose why they were not r

If the value is 1 or the KeyValue is not present, Tor d
relay's bw value in any votes for that relay.

Implementations MUST also set "bw=1" for unmeas
change the bw for under_min_report relays. (See th
and "under_min_report" for more details.)

This KeyValue was added in version 1.4.0 of this spe

"xoff_recv" Int

[Zero or one time.]

The number of times this relay received `XOFF_RECV`
in the last data_period days.

This KeyValue was added in version 1.6.0 of this spe

"xoff_sent" Int

[Zero or one time.]

The number of times this relay received `XOFF_SENT`
in the last data_period days.

This KeyValue was added in version 1.6.0 of this spe

"r_strm" Float

[Zero or one time.]

The stream ratio of this relay calculated as explaine

This KeyValue was added in version 1.7.0 of this spe

"r_strm_filt" Float

[Zero or one time.]

The filtered stream ratio of this relay calculated as e

This KeyValue was added in version 1.7.0 of this spe

Torflow

Torflow RelayLines include node_id and bw, and oth

References:

1. https://gitlab.torproject.org/tpo/network-healt
2. https://gitlab.torproject.org/tpo/network-healt /main/NetworkScanners/BwAuthority/READMI Torflow specification is outdated, and does no See section A.1. for the format produced by To
3. The Tor Directory Protocol
4. How Tor Version Numbers Work In Tor
5. https://semver.org/

Sample data

The following has not been obtained from any real r

Generated by Torflow

This an example version 1.0.0 document:

```
1523911758
node_id=$68A483E05A2ABDCA6DA5A3EF8DB5177638A2
measured_at=1523911725 updated_at=1523911725
pid_error_sum=4.11374090719 pid_bw=57136645 p
circ_fail=0.2 scanner=/filepath
node_id=$96C15995F30895689291F455587BD94CA427
measured_at=1523911623 updated_at=1523911623
pid_error_sum=3.96703337994 pid_bw=47422125 p
circ_fail=0.0 scanner=/filepath
```

Generated by sbws version 0.1.0

```
1523911758
version=1.1.0
software=sbws
software_version=0.1.0
latest_bandwidth=2018-04-16T20:49:18
file_created=2018-04-16T21:49:18
generator_started=2018-04-16T15:13:25
earliest_bandwidth=2018-04-16T15:13:26
====

bw=380 error_circ=0 error_misc=0 error_stream
master_key_ed25519=YaqV4vbvPYKucElk297eVdNArD
node_id=$68A483E05A2ABDCA6DA5A3EF8DB5177638A2
time=2018-05-08T16:13:26
bw=189 error_circ=0 error_misc=0 error_stream
master_key_ed25519=a6a+dZadrQBtfSbmQkP7j2ardC
node_id=$96C15995F30895689291F455587BD94CA427
time=2018-05-08T16:13:36
```

Generated by sbws version 1.0.3

```
1523911758
version=1.2.0
latest_bandwidth=2018-04-16T20:49:18
file_created=2018-04-16T21:49:18
generator_started=2018-04-16T15:13:25
earliest_bandwidth=2018-04-16T15:13:26
minimum_number_eligible_relays=3862
minimum_percent_eligible_relays=60
number_consensus_relays=6436
number_eligible_relays=6000
percent_eligible_relays=93
software=sbws
software_version=1.0.3
=====

bw=38000 bw_mean=1127824 bw_median=1180062 de
desc_bw_obs_last=17230879 desc_bw_obs_mean=14
error_stream=1
master_key_ed25519=YaqV4vbvPYKucElk297eVdNArD
node_id=$68A483E05A2ABDCA6DA5A3EF8DB5177638A2
time=2018-05-08T16:13:26
bw=1 bw_mean=199162 bw_median=185675 desc_bw_
desc_bw_obs_last=836165 desc_bw_obs_mean=8580
error_stream=0
master_key_ed25519=a6a+dZadrQBtfSbmQkP7j2ardC
node_id=$96C15995F30895689291F455587BD94CA427
time=2018-05-08T16:13:36
``

<a id="bandwidth-file-spec.txt-A.3.1"></a>

### When there are not enough eligible measur
enough-measured }

```text
1540496079
version=1.2.0
earliest_bandwidth=2018-10-20T19:35:52
file_created=2018-10-25T19:35:03
generator_started=2018-10-25T11:42:56
latest_bandwidth=2018-10-25T19:34:39
minimum_number_eligible_relays=3862
minimum_percent_eligible_relays=60
number_consensus_relays=6436
number_eligible_relays=2960
percent_eligible_relays=46
software=sbws
software_version=1.0.3
=====
```

Headers generated by sbws vers
```

```
1523911758
version=1.2.0
latest_bandwidth=2018-04-16T20:49:18
destinations_countries=TH,ZZ
file_created=2018-04-16T21:49:18
generator_started=2018-04-16T15:13:25
earliest_bandwidth=2018-04-16T15:13:26
minimum_number_eligible_relays=3862
minimum_percent_eligible_relays=60
number_consensus_relays=6436
number_eligible_relays=6000
percent_eligible_relays=93
scanner_country=SN
software=sbws
software_version=1.0.4
=====
```

Generated by sbws version 1.1.0

```
1523911758
version=1.4.0
latest_bandwidth=2018-04-16T20:49:18
destinations_countries=TH,ZZ
file_created=2018-04-16T21:49:18
generator_started=2018-04-16T15:13:25
earliest_bandwidth=2018-04-16T15:13:26
minimum_number_eligible_relays=3862
minimum_percent_eligible_relays=60
number_consensus_relays=6436
number_eligible_relays=6000
percent_eligible_relays=93
recent_measurement_attempt_count=6243
recent_measurement_failure_count=732
recent_measurements_excluded_error_count=969
recent_measurements_excluded_few_count=3946
recent_measurements_excluded_near_count=90
recent_measurements_excluded_old_count=0
recent_priority_list_count=20
recent_priority_relay_count=6243
scanner_country=SN
software=sbws
software_version=1.1.0
time_to_report_half_network=57273
=====

bw=1 error_circ=1 error_destination=0 error_m
error_stream=0 master_key_ed25519=J3HQ24kOQWa
nick=snap269 node_id=$DC4D609F95A52614D1E69C7
relay_recent_measurement_attempt_count=3
relay_recent_measurements_excluded_error_cour
relay_recent_measurements_excluded_near_count
relay_recent_consensus_count=3 relay_recent_p
time=2019-03-16T18:20:57 unmeasured=1 vote=0
bw=1 error_circ=0 error_destination=0 error_m
error_stream=2 master_key_ed25519=h6ZB1E1yBFW
nick=relay node_id=$C4544F9E209A9A9B99591D548
relay_recent_measurement_attempt_count=3
relay_recent_measurements_excluded_error_cour
relay_recent_measurements_excluded_few_count=
relay_recent_priority_list_count=3 success=1
unmeasured=1 vote=0
```

Scaling bandwidths

Scaling requirements

```
Tor accepts zero bandwidths, but they trigg
implementations. Therefore, scaling methods
following checks:
 * If the total bandwidth is zero, all rela
 bandwidths.
 * If the scaled bandwidth is zero, it shou
```

Initial experiments indicate that scaling may not be
their measured bandwidths are similar enough alre

A linear scaling method

If scaling is required, here is a simple linear bandwic
that all bandwidth votes contain approximately the

```
1. Calculate the relay quota by dividing th
   in all votes, by the number of relays wi
   votes. In the public tor network, this i
   April 2018. The quota should be a conser
   adjusted for all generators on the netwo

2. Calculate a vote quota by multiplying th
   of relays this bandwidth authority has m
   bandwidths for.

3. Calculate a scaling factor by dividing t
   total unscaled measured bandwidth in thi
   authority's upcoming vote.

4. Multiply each unscaled measured bandwidt
   factor.
```

Now, the total scaled bandwidth in the upcoming vo
quota.

Quota changes

If all generators are using scaling, the quota can be g
needed. Smaller quotas decrease the size of uncom
decrease the size of consensus diffs and compresse
is too small, some relays may be over- or under-wei

Torflow aggregation

Torflow implements two methods to compute the b
bandwidth measurements: with and without PID co
here is without PID control (see Torflow specificatio

In the following sections, the relays' measured banc
bandwidth authority has measured for the relays th
bandwidth authority's upcoming vote.

```
1. Calculate the filtered bandwidth for eac
  - choose the relay's measurements (`bw_j`
    than the mean of the measurements for t
  - calculate the mean of those measurement

  In pseudocode:

    bw_filt_i = mean(max(mean(bw_j), bw_j))

2. Calculate network averages:
  - calculate the filtered average by divic
    relays' filtered bandwidth by the numbe
    measured (`n`), ie, calculate the mean
    filtered bandwidth.
  - calculate the stream average by dividir
    relays' measured bandwidth by the numbe
    measured (`n`), ie, calculate the mean
    measured bandwidth.

   In pseudocode:

     bw_avg_filt_ = bw_filt_i / n
     bw_avg_strm = bw_i / n

3. Calculate ratios for each relay:
  - calculate the filtered ratio by dividir
    bandwidth by the filtered average
  - calculate the stream ratio by dividing
    bandwidth by the stream average

  In pseudocode:
```

r_filt_i = bw_filt_i / bw_avg_filt r_strm_i = bw_i / bw_a

```
4. Calculate the final ratio for each relay
  The final ratio is the larger between the
  stream bandwidth's ratio.

  In pseudocode:

    r_i = max(r_filt_i, r_strm_i)

5. Calculate the scaled bandwidth for each
  The most recent descriptor observed bandw
  multiplied by the ratio

  In pseudocode:

    bw_new_i = r_i * bw_obs_i
```

<<In this way, the resulting network status consensu weighted proportional to how much faster the node network.>>

Tor Directory List Forma

Tim Wilson-Brown (te

Scope and Preliminaries

This document describes the format of Tor's directo
coded into the tor binary. There is currently one list:
list is also parsed by other libraries, like stem and m
implementations can use this list to bootstrap from
information.

The FallbackDir feature was introduced by proposal
in Tor version 0.2.4.7-alpha. The first hard-coded list

The hard-coded fallback directory list is located in th

```
src/app/config/fallback_dirs.inc
```

In Tor 0.3.4 and earlier, the list is located at:

```
src/or/fallback_dirs.inc
```

This document describes version 2.0.0 and later of t

Legacy, semi-structured versions of the fallback list
through Tor 0.3.1.9. We call this format version 1. St
this legacy format.

Format Overview

A directory list is a C code fragment containing an ar
double-quoted C string constant is a valid torrc Fallb
various data fields.

Directory lists do not include the C array's declaratio
Entries in directory lists do not include the FallbackD
the including C code.

Directory lists also include C-style comments and wh
may be significant, but the amount of whitespace is

whitespace is not significant to the C compiler or To
parsers MAY rely on the distinction between newlin
whitespace characters in the list are newlines and s

The directory entry C string constants are split over
Structured C-style comments are used to provide ac
is not used by Tor, but may be of interest to other li

The order of directory entries and data fields is not

Acknowledgements

The original fallback directory script and format was
uses code written by gsathya & karsten.

This specification was revised after feedback from D
Learmonth ("irl").

Format Versions

The directory list format uses semantic versioning: h

```
    In particular:
      * major versions are used for incompatib
        removing non-optional fields
      * minor versions are used for compatible
        fields
      * patch versions are for bug fixes, like
        incorrectly-formatted Summary item

    1.0.0 - The legacy fallback directory list

    2.0.0 - Adds name and extrainfo structured
            comments to make the list easier t
list
            comment to the header.

    3.0.0 - Modifies the format of the source
```

Future Plans

Tor also has an auth_dirs.inc file, but it is not yet in t
formats for authorities and fallback directory mirror
changes to tor so that it parses this format. (We will
information to this format.) See #24818 for details.

We want to add a torrc option so operators can opt-
mirrors. This gives us a signed opt-in confirmation. (
whitelist entries, and do other checks.) We need to v
some changes to tor and the fallback update script.

Format Details

Directory lists contain the following sections:

- List Header (exactly once)
- List Generation (exactly once, may be empty)
- Directory Entry (zero or more times)

Each section (or entry) ends with a separator.

Nonterminals

The following nonterminals are defined in the Onior

- dir_address
- fingerprint
- nickname

See https://metrics.torproject.org/onionoo.html#de

The following nonterminals are defined in the "Tor c
spec.txt:

```
Keyword
ArgumentChar
NL      (newline)
SP      (space)
bool    (must not be confused with Onior
```

We derive the following nonterminals from Onionoc

```
    ipv4_or_port ::= port from an IPv4 or_ac

      The ipv4_or_port is the port part of a
      Onionoo or_addresses list.

    ipv6_or_address ::= an IPv6 or_addresses

      The ipv6_or_address is an IPv6 address
      or_addresses list. The address MAY be
      IPv6 address format.

  A key-value pair:

    value ::= Zero or more ArgumentChar, exc
                * a double quotation mark (D
                * the C comment terminators

              Note that the C++ comment ("//
              not excluded, because they are
              base64 values.

    key_value ::= Keyword "=" value

  We also define these additional nontermina

    number ::= An optional negative sign ("-
               numeric characters ([0-9]), w
               (".", followed by one or more

    separator ::= "/*" SP+ "=====" SP+ "*/"
```

List Header

The list header consists of a number of key-value pa

List Header Format

```
"/*" SP+ "type=" Keyword SP+ "*/" SP\* N

  [At start, exactly once.]

  The type of directory entries in the l
  an error if this is not the first line
  is anything other than "fallback".

"/*" SP+ "version=" version_number SP+ '

  [In second position, exactly once.]

  The version of the directory list form

  version_number is a semantic version,
  section for details.

  Version 1.0.0 represents the undocumen
  format(s). Version 2.0.0 and later are
  specification.

"/*" SP+ "timestamp=" number SP+ "*/" SP

  [Exactly once.]

  A positive integer that indicates when
  generated. This timestamp is guarantee
  version 2.0.0 and later directory list

  The current timestamp format is YYYYMM

"/*" SP+ "source=" Keyword ("," Keyword)

  [Zero or one time.]

  A list of the sources of the directory

  As of version 3.0.0, the possible sour
    * "offer-list" - the fallback_offer_
scripts
                     repository.
    * "descriptor" - one or more signed
                     "offer-fallback-dir
                     implemented in tick
    * "fallback"   - a fallback_dirs.inc
                     Used in check_exist

  Before #24839 is implemented, the defa
  transition to signed offers, it will b
  Afterwards, it will be "descriptor".

  In version 2.0.0, only one source name
  and the deprecated "whitelist" source
  "offer-list".
```

```
    This line was added in version 2.0.0 o
    of this line was modified in version 3

  "/*" SP+ key_value SP+ "*/" SP* NL

    [Zero or more times.]

    Future releases may include additional
    rely on the order of these additional
    will be accompanied by a minor version

  separator SP* NL

    The list header ends with the section
```

List Generation

The list generation information consists of human-r
and origin of this directory list. It is contained in zero
contain multi-line comments and uncommented C c

In particular, this section may contain C-style comm
character. It may also be entirely empty.

Future releases may arbitrarily change the content o
on a version increment when the format changes.

List Generation Format

In general, parsers MUST NOT rely on the format of

Parsers MAY rely on the following details:

The list generation section MUST NOT be a valid dire

The list generation summary MUST end with a sectio

separator SP* NL

There MUST NOT be any section separators in the li
terminating section separator.

Directory Entry

A directory entry consists of a C string constant, and
C string constant is a valid argument to the DirAutho

section also contains additional key-value fields in C

The list of fallback entries does not include the direc
separate list. (The Tor implementation combines the
applies the DirAuthorityFallbackRate to their weight

Directory Entry Format

```
If a directory entry does not conform t
be ignored by parsers.

DQUOTE dir_address SP+ "orport=" ipv4_o
  "id=" fingerprint DQUOTE SP* NL

  [At start, exactly once, on a single 

  This line consists of the following f

  dir_address

    An IPv4 address and DirPort for thi
    Onionoo. In this format version, al
    are guaranteed to be non-zero. (For
    that they are not equal to "0.0.0.0

  ipv4_or_port

    An IPv4 ORPort for this directory, 
    format version, all IPv4 ORPorts ar

  fingerprint

    The relay fingerprint of this direc
    All relay fingerprints are guarante
    digits.

Note:

  Each double-quoted C string line that
  starts with space inside the quotes. 
  Tor implementation.

DQUOTE SP+ "ipv6=" ipv6_or_address DQUO

  [Zero or one time.]

  The IPv6 address and ORPort for this 
  Onionoo. If present, IPv6 addresses a
  non-zero. (For IPv6 addresses, this m
  "[::]".)

DQUOTE SP+ "weight=" number DQUOTE SP* 

  [Zero or one time.]

  A non-negative, real-numbered weight 
  The default fallback weight is 1.0, a
  DirAuthorityFallbackRate is 1.0 in le
  recent Tor versions.

  weight was removed in version 2.0.0, 
  may be of interest to libraries imple
  behaviour.
```

```
DQUOTE SP+ key_value DQUOTE SP* NL

  [Zero or more times.]

  Future releases may include additional
  C string constants. Parsers MUST NOT r
  additional fields. Additional data fie
  minor version increment.

"/*" SP+ "nickname=" nickname* SP+ "*/"

  [Exactly once.]

  The nickname for this directory, as de
  empty nickname indicates that the nick

  The first fallback list in the 2.0.0 f
  they were all empty.

"/*" SP+ "extrainfo=" bool SP+ "*/" SP*

  [Exactly once.]

  An integer flag that indicates whether
  extra-info documents. Set to 1 if the
  cached extra-info documents in its des
  created. 0 indicates that it did not,
  available.

  The first fallback list in the 2.0.0 f
  they were all zero.

"/*" SP+ key_value SP+ "*/" SP* NL

  [Zero or more times.]

  Future releases may include additional
  comments. Parsers MUST NOT rely on the
  fields. Additional data fields will be
  increment.

separator SP* NL

  [Exactly once.]

  Each directory entry ends with the sep

"," SP* NL

  [Exactly once.]

  The comma terminates the C string cons
  constants separated by whitespace or c
  the C compiler.)
```

Usage Considerations

This section contains recommended library behavio
directory lists.

Caching

The fallback list typically changes once every 6-12 m
the state of the fallback directory entries when the l
change their details over time.

Libraries SHOULD parse and cache the most recent
build or release processes. Libraries MUST NOT retr
they are deployed or executed.

The latest fallback list can be retrieved from:

https://gitlab.torproject.org/tpo/core/tor/-/raw/mair
/fallback_dirs.inc?ref_type=heads

Libraries MUST NOT rely on the availability of the se

The list can also be retrieved using:

```
git clone https://gitlab.torproject.org/tpo/c
```

If you just want the latest list, you may wish to perfo

Retrieving Directory Information

Some libraries retrieve directory documents directly
The directory authorities are designed to support To
choose to rate-limit library access. Libraries MAY pro
they are not intended to support anonymous opera
vector.)

Libraries SHOULD consider the potential load on the
sources can meet their needs.

Libraries that require high-uptime availability of Tor
investigate the following options:

```
* OnionOO: https://metrics.torproject.or
  * Third-party OnionOO mirrors are also
* CollecTor: https://collector.torprojec
* Fallback Directory Mirrors
```

Onionoo and CollecTor are typically updated every h
update their own directory information at random i

Fallback Reliability

The fallback list is typically regenerated when the fa
Libraries SHOULD NOT rely on any particular fallbac
of fallbacks being available.

Libraries that use fallbacks MAY wish to query an au
fail. For example, Tor clients try 3-4 fallbacks before

Sample Data

A sample version 2.0.0 fallback list is available here:

https://trac.torproject.org/projects/tor/raw-attachm
/fallback_dirs_new_format_version.4.inc

A sample transitional version 2.0.0 fallback list is ava

https://raw.githubusercontent.com/teor2345/tor/fal
/or/fallback_dirs.inc

Sample Fallback List Header

```
/*type=fallback */
/* version=2.0.0 */
/* =====*/
```

Sample Fallback List Generation

```
/*Whitelist & blacklist excluded 1326 of 1513
/* Checked IPv4 DirPorts served a consensus w
/*
Final Count: 151 (Eligible 187, Target 392 (1
Excluded: 36 (Same Operator 27, Failed/Skippe
Bandwidth Range: 1.3 - 40.0 MByte/s
*/
/*
Onionoo Source: details Date: 2017-05-16 07:0
URL:
https:onionoo.torproject.orgdetails?fields=f
Clast_changed_address_or_port%2Cconsensus_we
addresses%2Cdir_address%2Crecommended_versior
atform&flag=V2Dir&type=relay&last_seen_days=-
*/
/*
Onionoo Source: uptime Date: 2017-05-16 07:00
URL: https:onionoo.torproject.orguptime?first
type=relay&last_seen_days=-0
*/
/* ===== \*/
```

Sample Fallback Entries

```
"176.10.104.240:80 orport=443 id=0111BA9B6046
/*nickname=foo */
/* extrainfo=1 */
/* ===== */
,
"5.9.110.236:9030 orport=9001 id=0756B7CD4DF0
" ipv6=\[2a01:4f8:162:51e2::2\]:9001"
/* nickname= */
/* extrainfo=0 */
/* =====*/
,
```

Tor network parameters

This file lists the recognized parameters that can ap
directory consensus.

Network protocol parameters

"circwindow" -- the default package window that circ
started out at 1000 cells, but some research indicate
fewer cells in transit in the network at any given tim
First-appeared: Tor 0.2.1.20

"UseOptimisticData" -- If set to zero, clients by defau
data to servers until they have received a RELAY_CO
1 First-appeared: 0.2.3.3-alpha Default was 0 before
alpha; now always on.

"usecreatefast" -- Used to control whether clients us
first hop of their circuits. Min: 0, Max: 1. Default: 1. F
Removed in 0.4.5.1-alpha; now always off.

"min_paths_for_circs_pct" -- A percentage threshold
believe they have enough directory information to b
total fraction of bandwidth-weighted paths that the
for more information. Min: 25, Max: 95, Default: 60 I

"ExtendByEd25519ID" -- If true, clients should incluc
generating EXTEND2 cells. Min: 0. Max: 1. Default: 0.

"sendme_emit_min_version" -- Minimum SENDME v
255. Default 0. First appeared: 0.4.1.1-alpha.

"sendme_accept_min_version" -- Minimum SENDME
255. Default 0. First appeared: 0.4.1.1-alpha.

"allow-network-reentry" -- If true, the Exit relays allo
network to re-enter. If false, any exit connections go
ORPort and DirPort is denied and the stream is term
appeared: 0.4.5.1-alpha.

Performance-tuning parameters

"CircuitPriorityHalflifeMsec" -- the halflife parameter will send the next cell. Obeyed by Tor 0.2.2.10-alpha 0.2.2.7-alpha and 0.2.2.10-alpha recognized a "CircP mishandled it badly.) Min: 1, Max: 2147483647 (INT3 appeared: Tor 0.2.2.11-alpha

```
"perconnbwrate" and "perconnbwburst" -- 
separate token bucket for every client OF
that connection independently. Typically
performance experiments around trac entry
running Tor 0.2.2.16-alpha and later. (No
0.2.2.7-alpha through 0.2.2.14-alpha look
bwconnburst, but then did the wrong thing
details.)
Min: 1, Max: 2147483647 (INT32_MAX), Defa
    BandwidthRate/BandwidthBurst).
First-appeared: 0.2.2.7-alpha
Removed-in: 0.2.2.16-alpha
```

"NumNTorsPerTAP" -- When balancing ntor and TAP handshakes should we perform for each TAP hands 10. First-appeared: 0.2.4.17-rc

"circ_max_cell_queue_size" -- This parameter detern allowed per circuit queue. Min: 1000. Max: 214748 First-appeared: 0.3.3.6-rc.

"KISTSchedRunInterval" -- How frequently should th decide which data to write to the network? Value in 100. Default: 2 First appeared: 0.3.2

"KISTSchedRunIntervalClient" -- How frequently sho to decide which data to write to the network, on clie The client value needs to be much lower than the re First appeared: 0.4.8.2

Voting-related parameters

"bwweightscale" -- Value that bandwidth-weights ar defaults to 10000. Min: 1 First-appeared: 0.2.2.10-al

"maxunmeasuredbw" -- Used by authorities during maximum value to give for any Bandwidth= entry fo three measurements.

(Note: starting in version 0.4.6.1-alpha there was a b instead look at a parameter called "maxunmeasurdl fixed in 0.4.9.1-alpha and in 0.4.8.8. Until all relays a this parameter must not be set, or it must be set to

First-appeared: 0.2.4.11-alpha

"FastFlagMinThreshold", "FastFlagMaxThreshold" -- for the cutoff for routers that should get the Fast fla prevent the threshold for getting the Fast flag from FastFlagMinThreshold: Min: 4. Max: INT32_MAX: Def Max: INT32_MAX: Default: INT32_MAX First-appeare

"AuthDirNumSRVAgreements" -- Minimum number required for a fresh shared random value to be writ applies on the first commit round of the shared ran INT32_MAX. Default: 2/3 of the total number of dira

Circuit-build-timeout parameter

"cbtdisabled", "cbtnummodes", "cbtrecentcount", "c "cbtquantile", "cbtclosequantile", "cbttestfreq", "cbtı "cbtmaxopencircs", and "cbtinitialtimeout" -- see "2. behavior" in path-spec.txt for a series of circuit builc

Directory-related parameters

"max-consensus-age-to-cache-for-diff" -- Determine hours) relays should try to cache in order to serve d

"try-diff-for-consensus-newer-than" -- This paramete can be (in hours) before a client should no longer try default 72)

Pathbias parameters

"pb_mincircs", "pb_noticepct", "pb_warnpct", "pb_ex "pb_scalecircs", "pb_scalefactor", "pb_multfactor", "p "pb_extremeusepct", "pb_scaleuse" -- DOCDOC

Relay behavior parameters

"refuseunknownexits" -- if set to one, exit relays lool
ask to open an exit stream, and refuse to exit if they
is to make it harder for people to use them as one-h
details. Min: 0, Max: 1 First-appeared: 0.2.2.17-alpha

"onion-key-rotation-days" -- (min 1, max 90, default

"onion-key-grace-period-days" -- (min 1, max onion-

Every relay should list each onion key it generates fo
generating it, and then replace it. Relays should con
previous onion key for an additional onion-key-grac
(Introduced in 0.3.1.1-alpha; prior versions of tor ha
days.)

"AllowNonearlyExtend" -- If true, permit EXTEND cel
cells. Min: 0. Max: 1. Default: 0. First-appeared: 0.2.3

"overload_dns_timeout_scale_percent" -- This value
timeout over N seconds we accept before reporting
by a factor of 1000 in order to be able to represent
of 1000 means 1%. Min: 0. Max: 100000. Default: 10
Deprecated: 0.4.7.3-alpha-dev

"overload_dns_timeout_period_secs" -- This value is
timeout measurements (the N in the "overload_dns
For this amount of seconds, we will gather DNS stat
assessment on the overload general signal with rega
2147483647. Default: 600 First-appeared: 0.4.6.8 De

"overload_onionskin_ntor_scale_percent" -- This valu
onionskin ntor drop over N seconds we accept befo
state. It is scaled by a factor of 1000 in order to be a
example, a value of 1000 means 1%. Min: 0. Max: 10
0.4.7.5-alpha

"overload_onionskin_ntor_period_secs" -- This value
onionskin ntor overload measurements (the N in th
"overload_onionskin_ntor_scale_percent" paramete
gather onionskin ntor statistics and at the end, we'll
general signal. Min: 0. Max: 2147483647. Default: 21
alpha

"assume-reachable" -- If true, relays should publish

make a connection to their IPv4 ORPort. Min: 0. Max
alpha.

"assume-reachable-ipv6" -- If true, relays should pul
cannot make a connection to their IPv6 ORPort. Min
0.4.5.1-alpha.

"exit_dns_timeout" -- The time in milliseconds an Ex
considers the DNS timed out. The corresponding lib
Max: 120000. Default: 1000 (1sec) First appeared: 0.

"exit_dns_num_attempts" -- How many attempts *aft*
timing-out DNS query before calling it hopeless? (Ea
"exit_dns_timeout" independently). The correspond
0. Max: 255. Default: 2 First appeared: 0.4.7.5-alpha.

V3 onion service parameters

"hs_intro_min_introduce2", "hs_intro_max_introduc
INTRODUCE2 cells allowed per circuits before rotati
between these two values). Min: 0. Max: INT32_MAX

"hs_intro_min_lifetime", "hs_intro_max_lifetime" -- M
seconds that a service should keep an intro point fo
between these two values). Min: 0. Max: INT32_MAX

"hs_intro_num_extra" -- Number of extra intro point
concept comes from proposal #155. Min: 0. Max: 12

"hsdir_interval" -- The length of a time period, *in mir*
[TIME-PERIODS]. Min: 30. Max: 14400. Default: 1440

"hsdir_n_replicas" -- Number of HS descriptor replica

"hsdir_spread_fetch" -- Total number of HSDirs per r
to fetch a descriptor. Min: 1. Max: 128. Default: 3.

"hsdir_spread_store" -- Total number of HSDirs per r
descriptor to. Min: 1. Max: 128. Default: 4

"HSV3MaxDescriptorSize" -- Maximum descriptor siz
Default: 50000

"hs_service_max_rdv_failures" -- This parameter det
rendezvous attempt an HS service can make per int

First-appeared: 0.3.3.0-alpha.

"HiddenServiceEnableIntroDoSDefense" -- This para
start using a rate-limiting defense if they support it.
when no `DOS_PARAMS` extension is sent in the ESTAE
Default: 0. First appeared: 0.4.2.1-alpha.

"HiddenServiceEnableIntroDoSBurstPerSec" -- Defau
token bucket for the introduction point rate-limiting
when no `DOS_PARAMS` extension is sent in the ESTAE
INT32_MAX. Default: 200 First appeared: 0.4.2.1-alpl

> Note that the above parameter is slightly misnam
> "per second".

"HiddenServiceEnableIntroDoSRatePerSec" -- Defau
used for token bucket for the introduction point rat
value when no `DOS_PARAMS` extension is sent. Min: (
appeared: 0.4.2.1-alpha.

Denial-of-service parameters

Denial of Service mitigation parameters. Introduced

"DoSCircuitCreationEnabled" -- Enable the circuit cre

"DoSCircuitCreationMinConnections" -- Minimum th
before a client address can be flagged as executing

"DoSCircuitCreationRate" -- Allowed circuit creation
once the minimum concurrent connection threshol

"DoSCircuitCreationBurst" -- The allowed circuit crea
the minimum concurrent connection threshold is re

```
"DoSCircuitCreationDefenseType" -- Defens
detected client address for the circuit c
    1: No defense.
    2: Refuse circuit creation for the le
      "DoSCircuitCreationDefenseTimePeric
```

"DoSCircuitCreationDefenseTimePeriod" -- The base
activated for.

"DoSConnectionEnabled" -- Enable the connection D

"DoSConnectionMaxConcurrentCount" -- The maxin
connection from a client IP address.

```
"DoSConnectionDefenseType" -- Defense typ
client address for the connection mitigat
    1: No defense.
    2: Immediately close new connections.
```

"DoSRefuseSingleHopClientRendezvous" -- Refuse e
single hop clients.

Padding-related parameters

"circpad_max_circ_queued_cells" -- The circuitpaddir
padding cells if more than this many cells are in the
Max: 50000. Default 1000. First appeared: 0.4.0.3-al

"circpad_global_allowed_cells" -- This is the number
before the 'circpad_global_max_padding_percent' pa
65535. Default: 0

"circpad_global_max_padding_pct" -- This is the max
cells, specified as a percent. If the global ratio of pac
circuits exceeds this percent value, no more padding
lower. 0 means no limit. Min: 0. Max: 100. Default: 0

"circpad_padding_disabled" -- If set to 1, no circuit p
all current padding machines will cease padding imr

"circpad_padding_reduced" -- If set to 1, only circuit
"reduced"/"low overhead" will be used. (Currently n
"reduced overhead"). Min: 0. Max: 1. Default: 0

"nf_conntimeout_clients"

- The number of seconds to keep never-used cir
  to use. Note that the actual client timeout is ra
  to twice this value.
- The number of seconds to keep idle (not curre
  open and available. (We do this to ensure a su
  which is the ultimate goal.)
- This value is also used to determine how long,
  attempt to keep building predicted circuits for

2.1.1.) This behavior was originally added to w
limitations, but it serves as a reasonable defau

- For all use cases, reduced padding clients use
- Implementations MAY mark circuits held open (half the consensus value) as "not to be used f becoming a distinguisher. Min: 60. Max: 86400

"nf_conntimeout_relays" -- The number of seconds t kept open. Min: 60. Max: 604800. Default: 3600

"nf_ito_low" -- The low end of the range to send pad Max: 60000. Default: 1500

"nf_ito_high" -- The high end of the range to send pa nf_ito_high == 0, padding will be disabled. Min: nf_it

"nf_ito_low_reduced" -- For reduced padding clients: padding when inactive, in ms. Min: 0. Max: 60000. D

"nf_ito_high_reduced" -- For reduced padding client padding, in ms. Min: nf_ito_low_reduced. Max: 6000

"nf_pad_before_usage" -- If set to 1, OR connections them for any application traffic. If 0, OR connections begins. Min: 0. Max: 1. Default: 1

"nf_pad_relays" -- If set to 1, we also pad inactive rel 1. Default: 0

"nf_pad_single_onion" -- DOCDOC

Guard-related parameters

(See guard-spec.txt for more information on the voc

"UseGuardFraction" -- If true, clients use `GuardFrac` in order to decide how to weight guards when pickin First appeared: 0.2.6

"guard-lifetime-days" -- Controls guard lifetime. If ar sampled more than this many days ago, it should be Min: 1. Max: 3650. Default: 120. First appeared: 0.3.(

"guard-confirmed-min-lifetime-days" -- Controls con confirmed more than this many days ago, it should

Min: 1. Max: 3650. Default: 60. First appeared: 0.3.0

"guard-internet-likely-down-interval" -- If Tor has be
long (in seconds), assume that the internet connecti
as unproven. Min: 1. Max: INT32_MAX. Default: 600.

"guard-max-sample-size" -- Largest number of guar
their sample. Min: 1. Max: INT32_MAX. Default: 60. F

"guard-max-sample-threshold-percent" -- Largest ba
that clients should try to collect in their sample. Min
appeared: 0.3.0

"guard-meaningful-restriction-percent" -- If the clien
many guards that the available guard bandwidth is l
treat the guard sample as "restricted", and keep it ir
Default: 20. First appeared: 0.3.0

"guard-extreme-restriction-percent" -- Warn the use
exclude so many guards that the available guard ba
of the total. Min: 1. Max: 100. Default: 1. First appea
which would have no meaningful effect. MAX lowere

"guard-min-filtered-sample-size" -- If fewer than this
sample after filtering out unusable guards, the clien
the sample (if allowed). Min: 1. Max: INT32_MAX. De

"guard-n-primary-guards" -- The number of confirm
as "primary guards". Min: 1. Max: INT32_MAX. Defau

```
"guard-n-primary-guards-to-use", "guard-r
-- number of primary guards and primary c
client should be willing to use in parall
won't get used unless the earlier ones ar
"guard-n-primary-guards-to-use":
   Min 1, Max INT32_MAX: Default: 1.
"guard-n-primary-dir-guards-to-use"
   Min 1, Max INT32_MAX: Default: 3.
First appeared: 0.3.0
```

"guard-nonprimary-guard-connect-timeout" -- Wher
guards, if a guard doesn't answer for more than this
guards as usable. Min: 1. Max: INT32_MAX. Default:

"guard-nonprimary-guard-idle-timeout" -- When tryi
guard doesn't answer for more than this long in sec
INT32_MAX. Default: 600 First appeared: 0.3.0

"guard-remove-unlisted-guards-after-days" -- If a gu
consensus for at least this many days, remove it fro
Default: 20. First appeared: 0.3.0

Obsolete parameters

"NumDirectoryGuards", "NumEntryGuards" -- Numb
by default. If NumDirectoryGuards is 0, we default t
NumDirectoryGuards: Min: 0. Max: 10. Default: 0 Nu
Default: 3 First-appeared: 0.2.4.23, 0.2.5.6-alpha Rer

"GuardLifetime" -- Duration for which clients should
Min: 30 days. Max: 1826 days. Default: 60 days. First
in: 0.3.0.

"UseNTorHandshake" -- If true, then versions of Tor
by default. Min: 0, Max: 1. Default: 1. First-appeared

"Support022HiddenServices" -- Used to implement a
timestamps to hidden services by default to sending
absent, or is set to 1, clients with the default configu
they do not. Min: 0, Max: 1. Default: 1. First-appeare

Tor Project SSH protocol

The SSH protocol provides various extension faciliti

The Tor Project has defined some extensions, using
facility. The Tor Project uses names ending `@spec.t`

Id(s)	Namespace	
`ed25519-expaneded@`	Public key algorithm (in SSH/OpenSSH key file)	E e p
`x25519@`	Public key algorithm (in SSH/OpenSSH key file)	X

Registration process

New entries may be added to this table after peer re
via gitlab merge request.

The specification links may be to external document
Specifications. Or, they may be links to specific secti
Proposals. External links should be dated, for ease c

Ideally, before a protocol is deployed, its specificatic
Specifications (and the link in the table adjusted).

Interpretation

This section uses the notation and conventions from
RFC4251 s5), not those from the rest of the Tor Spec

Interpreting the table

For example, the row for `x25519@` indicates that:

- The Tor Project has assigned `x25519@spec.tor`

- In the namespace of public key algorithms - se
  found within an SSH/OpenSSH format key file.
- The meaning of this name is summarised as "X
- The full details can be found at the linked text,
  section, below.

The registered names resemble email addresses, bu
mail to them will not be delivered.

For further information, consult the linked specificat

SSH key types for the Arti keysto

The Arti keystore stores private keys in OpenSSH ke
keys, in SSH format). But it needs to store some key
protocols. So the following key types are defined.

These are in the namespace of Public Key Algorithm
meaningful in OpenSSH format private key files (Op
SSH public key files (RFC4716 3.4).

In each case we specify/reference

- the name of the "public key algorithm" (RFC42
- the underlying cryptographic algorithm(s),
- the public key data ("key/certificate data" in RF
  key data
- the private key data (see Encoding of the priva

Encoding of the public key data

OpenSSH `PROTOCOL.key` does not clearly state the c
`publickey1` / `publickey2` / `publickeyN` fields in the
(`PROTOCOL.key` s1), so we state it here.

Each `publickey` consists of the encoded public key
"Certificates and public keys are encoded as follows

So the overall format of this part of the file is:

```
uint32       number of keys, N
    string       publickey1, where the c
        string       public key algorith
        byte[]       public key data (al
    ... keys 2 to N-1 inclusive, each as
    string       publickeyN (as for publ
```

Encoding of the private key data

OpenSSH `PROTOCOL.key` defines the representation
"using the same rules as used for SSH agent". Howe
the SSH agent protocol is only available as an Intern
and, the actual encoding used by OpenSSH is hard t
document our understanding here.

The contents of each `privatekey1` / `privatekey2` / `p`

```
string       public key algorithm name (RF
byte[]       public key data (algorithm-sp
byte[]       private key data (algorithm-s
```

> Note that this depends on the reader knowing th
> key data must be self-delimiting, since the file for
> length field. Although here we discuss only algori
> private key data, are each a single `string`, that is
> the `ssh-rsa` algorithm's key data formats are se
> surrounding overall length.
>
> Note also that the encrypted section does not sep
> or their total length. The reader must keep readin
> end, or something that looks like padding (startin

`x25519@spec.torproject.org`

These refer to keys for X25519, ie, Diffie-Hellman on
s5.

The public key data is:

```
string        wrapper for the following
    byte[32]      the u-coordinate encod
```

The private key data is:

```
string        wrapper for the following
   byte[32]        the scalar k encoded a
```

k MUST be stored as the true scalar value. So if the p
random bytes according to the procedure describec
random bytes as an integer scalar". the value storec
the value *after* the transformation. If a stored value
represent a valid scalar according to RFC7748 s5 (i.e
rejected; if it is not rejected, it MUST NOT be used u
clamped.

Keys whose `string` wrapper is not of the expected

> The `string` wrapper is useless, but the same wra
> SSH for ed25519 public keys (RFC8709 s4). and fo
> agent protocol (draft-miller-ssh-agent-04 4.2.3). V
> consistency (and implementation convenience).

> X25519 keys are interconvertible with ed25519 ke
> store the ed25519 form instead, and convert on l
> different key type to avoid needing conversions c
> type punning and accidental key misuse: using th
> algorithms is a poor idea.

`ed25519-expanded@spec.torproject.`

These refer to the expanded form of private keys fo

This key type appears within OpenSSH private key fi
`expanded@spec.torproject.org` algorithm name is
(`PROTOCOL.key` section 3, `privatekey1` etc.) but als
section 1, `publickey1` etc.).

> In `PROTOCOL.key` we interpret the requirement th
> private keys to include the requirement that the p
> must be the same.

> In the Arti keystore a private key file whose filena
> may contain either a standard ed25519 keypair w
> `ed25519-expanded@spec.torproject.org` keypai

`ed25519-expanded@spec.torproject.org` SHOULD
files. Software which is aware of this key type MUST
and SHOULD reject them on loading. (Software hand
MAY, and probably will, process such files without c

> These rules are because public keys should alway
> the private key is only available as `ed25519-expa`
> information about the key generation process to
> certification and verification.

> Arti will provide a utility to convert anomalous RF
> keys declared to be of type `ed25519-expanded@sp`
> conforming files containg `ed25519` keys. In other
> anomalous files.

The public key data is the same as for the official `ed`

```
    string          wrapper for the following
        byte[32]        the actual public key,
```

(Reference: RFC8032 3.2.)

The private key data is as follows:

```
    string          wrapper for the following
        byte[32]        ENC(s) as per RFC8032
        byte[32]        `h_b...h_(2b-1)` as pe
 nonce"
```

(References: `ENC(s)` in RFC8032 3.2; `h_b || ... |`

Keys whose `string` wrapper is not of the expected

> As with `x25519`, the `string` wrapper is useless. '
> reasons.

This private key format does not provide a way to co
(unexpanded, standard) ed25519 private key `k`.

> The ed25519 standards define the private key for
> is then hashed and processed into parameters us
> for verification. The `ed25519-expanded@spec.tor`
> keys which can be used with the ed25519 signatu
> corresponding working public key is known, but f
> of `k` (the "real ed25519" private key). Allowing su
> one wishes to find private keys whose public keys
> example when trying to find a `.onion` domain fo
> is also used where blinded ed25519 keys need to

Glossary

The Tor Project

This document aims to specify terms, notations, and
the Tor specification documents and other documen

This glossary is not a design document; it is only a r

This glossary is a work-in-progress; double-check its
authoritatively. ;)

Preliminaries

The key words "MUST", "MUST NOT", "REQUIRED", "
"SHOULD NOT", "RECOMMENDED", "MAY", and "OP
interpreted as described in RFC 2119.

Commonly used Tor configuratio

ORPort - Onion Router Port DirPort - Directory Port

Tor network components

Relays, aka OR (onion router)

[Style guide: prefer the term "Relay"]

Specific roles

Exit relay: The final hop in an exit circuit before traff
to external servers.

Non-exit relay: Relays that send and receive traffic c

Entry relay: The first hop in a Tor circuit. Can be eith depending on the client's configuration.

Guard relay: A relay that a client uses as its entry fo
are rotated more slowly to prevent attacks that can
many guards.

Bridge: A relay intentionally not listed in the public T
circumventing entities (such as governments or ISPs
Tor. Currently, bridges are used only as entry relays

Directory cache: A relay that downloads cached dire
authorities and serves it to clients on demand. Any
bandwidth is high enough.

Rendezvous point: A relay connecting a client to a hi
three-hop circuit, meeting at the rendezvous point.

Client, aka OP (onion proxy)

[Style: the "OP" and "onion proxy" terms are deprec

Authorities

Directory Authority: Nine total in the Tor network, o
Directory authorities define and serve the consensu
network." This document contains a "router status"
network. Directory authorities also serve router des
microdescriptors, and the microdescriptor consens

Bridge Authority: One total. Similar in responsibility
bridges.

Fallback directory mirror: One of a list of directory c
software. (When a client first connects to the netwo
it asks a fallback directory. From then on, the client
listed in the directory information it has.)

Hidden Service

A hidden service is a server that will only accept inc

service protocol. Connection initiators will not be ab
hidden service, allowing the hidden service to receiv
content, etc, while preserving its location anonymity

Circuit

An established path through the network, where cry
the ntor protocol or TAP (Tor Authentication Protoco
Circuits can differ in length depending on their purp

Origin Circuit -

Exit Circuit: A circuit which connects clients to destir
example, if a client wanted to visit duckduckgo.com,
circuit.

Internal Circuit: A circuit whose traffic never leaves t
could connect to a hidden service via an internal circ

Edge connection

```
 2.7. Consensus: The state of the Tor network,
      decided by a vote from the network's dir
      fetch the consensus from directory autho
      directories, or directory caches.

 2.8. Descriptor: Each descriptor represents t
     relay in the Tor network. The descriptor
     address, public keys, and other data. Rel
     descriptors to directory authorities, who
     summary of them in the network consensus.
```

Tor network protocols

Link handshake

The link handshake establishes the TLS connection o
send Tor cells. This handshake also authenticates th

using Tor cells.

Circuit handshake

Circuit handshakes establish the hop-by-hop onion
their application traffic. The client does a pairwise k
each individual relay in the circuit. For every hop ex
tunnel through existing hops in the circuit. Each cell
version (with a "2" suffix), e.g., CREATE2.

CREATE cell: First part of a handshake, sent by the ir

CREATED cell: Second part of a handshake, sent by t

EXTEND cell: (also known as a RELAY_EXTEND cell) F
through an existing circuit. The last relay in the circu
the payload in a CREATED cell to the chosen next hc

EXTENDED cell: (also known as a RELAY_EXTENDED
tunneled through an existing circuit. The last relay ir
CREATED cell from the new last hop relay and encry
to tunnel back to the client.

Onion skin: A CREATE/CREATE2 or EXTEND/EXTEND
of the TAP or ntor key establishment handshake.

Hidden Service Protocol

Directory Protocol

General network definitions

Leaky Pipe Topology: The ability for the origin of a ci
addressed to any hop in the path of a circuit. In Tor,
using the 'recognized' field of relay cells.

Stream: A single application-level connection or requ
'Stream' can currently carry the contents of a TCP cc

directory request.

Channel: A pairwise connection between two Tor re
Circuits are multiplexed over Channels. All channels
connections.

About the Tor Specificat

The canonical, official, versions of these documents
maintained by the Tor Project.

Only the Tor Specifications themselves are approve
drafts.

When linking to the Specifications, consider using o
of Permalinks.

Source code

The Specifications and Proposals are maintained by

Corrections and clarifications are welcome. To prop
the Proposals process

Building

The documents are in Markdown and formatted wit
HTML:

```
cargo install mdbook
git clone https://gitlab.torproject.org/tpo/c
cd torspec
bin/build_html
```

The output is then in `html/` .

Source code structure, and outp

There are two mdbook books here:

- **The Tor Specifications**: source code in `specs`
- **Proposals**: source code in `proposals/` , forma

Each book's source files are listed, and the chapter
is pretty restrictive; see the mdbook documentation

Editing advice

To edit these specs, clone the git repository and edi
directory. These files will match the URLs of their co
edit `tor-spec/flow-control.html`, you'll be lookin
`spec/flow-control.md`.

We have started a style guide for writing new parts
preliminary. You should feel free to edit it!

Tor specifications: style

Audience

The primary intended audiences are:

- Programmers: implementors of the Tor Protoc
  existing implementations, and people writing r
- Researchers: people analysing the security of t
  academic research and practical security inves
- Expert users who wish to fully understand the
  includes users of clients and relays.
- Directory authority operators, and others with
  of the Tor network.

Scope and intentions

These notes apply to our specifications. When possi
proposals, to make proposals easier to merge into o

As of 2023, our existing specifications have been wr
so you should not expect to find these guidelines to
you read them. Instead, these guidelines are for doc

These notes are not terribly well organized. We shou
meant to be a living document.

Other sources

There are a number of other style guides used in To
they do not suit our needs, we should try to get the

- Community team guidelines
- Tor project glossary
- Specifications glossary

(Please add more if you know about them!)

As we refine the guidelines in this file, we should att
more project-wide guides, if they are suitable.

Starting notes

We are moving towards using semantic newlines: pu
subclause, on its own line, in the Markdown source
consistently, and we won't reject contributions that

Vocabulary

We use these terms freely:

- Channel
- Circuit
- Stream

We try not to say "connection" without qualification
that can be called a "connection".

Similarly, don't say "session" without qualification e
what we mean.

Prefer "relay" to "node" or "server".

Prefer "service" and "client" when talking about onic

Refer to arti as arti and the C tor implementation as
mention. Subsequently you can call it `tor` or "C tor"

Avoid "AP" and "OP" and "OR"; those are not in curr

Documenting keys

TODO: Explain our new key documentation convent

Documentating data encodings

We have two competing legacy schemes for docume
an ad-hoc format the looks like this:

```
    * FIELD_1     [4 bytes]
    * FIELD_2     [1 byte]
```

The other is a somewhat C-like format based on the

```
struct message {
   u32 field_1;
   u8 field_2;
}
```

Neither of these is really great. We should find som

Writing explanations

> When you are writing an explanation in the midd
> a good idea to put it in quoted text, like this.

Managing links

We're in the early stages of this spec organization, b
long term maintainability.

Please think about how to keep links working in the
link to a file, make sure that the file's name is reaso
consider adding a redirect from the file's old name.
more information about how.)

If you want to link to a specific section within a file,
defined anchor that makes sense. The syntax to def
this:

```
## Heading with a long title that you want sh
```

If you need to change a heading, make sure that yo
before, so that links will still work.

Finally, when you're looking for specific sections (e.g
section 5.2.3") you can look for the HTML anchors th

example, if you want to find `dir-spec.txt` section 2
`id="dir-spec.txt-2.1.3"></a>` .

Permalinks

These URLs at `spec.toprorject.org` are intended t

`/address-spec`
: `https://spec.torproject.org/address-spec`

`/bandwidth-file-spec`
: `https://spec.torproject.org/bandwidth-fi`
Bandwidth File spec)

`/bridgedb-spec`
: `https://spec.torproject.org/bridgedb-spe`

`/cert-spec`
: `https://spec.torproject.org/cert-spec` (Ec

`/collector-protocol`
: `https://gitlab.torproject.org/tpo/networ`
`/master/src/main/resources/docs/PROTOCOL?`
CollecTor's File Structure)

`/control-spec`
: `https://spec.torproject.org/control-spec`

`/dir-spec`
: `https://spec.torproject.org/dir-spec` (Tor

`/dir-list-spec`
: `https://spec.torproject.org/dir-list-spe`

`/ext-orport-spec`
: `https://spec.torproject.org/ext-orport-s`
transports)

`/gettor-spec`
: `https://gitlab.torproject.org/tpo/core/t`
`/text_formats/gettor-spec.txt?ref_type=he`

`/padding-spec`
: `https://spec.torproject.org/padding-spec`

`/path-spec`
: `https://spec.torproject.org/path-spec` (Tc

`/pt-spec`
: `https://spec.torproject.org/pt-spec` (Tor I
version 1)

`/rend-spec`
: `https://spec.torproject.org/rend-spec` (Tc
Specification, latest version)

`/rend-spec-v2`
: `https://gitlab.torproject.org/tpo/core/t`
`spec-v2.txt?ref_type=heads` (Tor Onion Serv
2 (Obsolete))

`/rend-spec-v3`

https://spec.torproject.org/rend-spec (Tc
Specification, Version 3 (Latest))

/socks-extensions

https://spec.torproject.org/socks-extens
protocol)

/srv-spec

https://spec.torproject.org/srv-spec (Tor
Specification)

/tor-fw-helper-spec

https://gitlab.torproject.org/tpo/core/to
helper-spec.txt?ref_type=heads (Tor's (little

/tor-spec

https://spec.torproject.org/tor-spec (Tor

/torbrowser-design

https://2019.www.torproject.org/projects/
Implementation of the Tor Browser)

/version-spec

https://spec.torproject.org/version-spec

/tor-design

https://svn.torproject.org/svn/projects/d
The Second-Generation Onion Router)

/walking-onions

https://spec.torproject.org/proposals/323
(Walking Onions specifications)