blob: 021532e4d65d3af96fdb215e0d16ca960ca30673 (plain)

```
                      Tor directory protocol, version 2

0. Scope and preliminaries

   This directory protocol is used by Tor version 0.1.1.x and 0.1.2.x.  See
   dir-spec-v1.txt for information on earlier versions, and dir-spec.txt
   for information on later versions.

0.1. Goals and motivation

   There were several problems with the way Tor handles directory information
   in version 0.1.0.x and earlier.  Here are the problems we try to fix with
   this new design, already implemented in 0.1.1.x:
      1. Directories were very large and use up a lot of bandwidth: clients
         downloaded descriptors for all router several times an hour.
      2. Every directory authority was a trust bottleneck: if a single
         directory authority lied, it could make clients believe for a time an
         arbitrarily distorted view of the Tor network.
      3. Our current "verified server" system is kind of nonsensical.

      4. Getting more directory authorities would add more points of failure
         and worsen possible partitioning attacks.

   There are two problems that remain unaddressed by this design.
      5. Requiring every client to know about every router won't scale.
      6. Requiring every directory cache to know every router won't scale.

   We attempt to fix 1-4 here, and to build a solution that will work when we
   figure out an answer for 5.  We haven't thought at all about what to do
   about 6.

1. Outline

   There is a small set (say, around 10) of semi-trusted directory
   authorities.  A default list of authorities is shipped with the Tor
   software. Users can change this list, but are encouraged not to do so, in
   order to avoid partitioning attacks.

   Routers periodically upload signed "descriptors" to the directory
   authorities describing their keys, capabilities, and other information.
   Routers may act as directory mirrors (also called "caches"), to reduce
   load on the directory authorities.  They announce this in their
   descriptors.

   Each directory authority periodically generates and signs a compact
   "network status" document that lists that authority's view of the current
   descriptors and status for known routers, but which does not include the
   descriptors themselves.

   Directory mirrors download, cache, and re-serve network-status documents
   to clients.

   Clients, directory mirrors, and directory authorities all use
   network-status documents to find out when their list of routers is
   out-of-date.  If it is, they download any missing router descriptors.
   Clients download missing descriptors from mirrors; mirrors and authorities
   download from authorities.  Descriptors are downloaded by the hash of the
   descriptor, not by the server's identity key: this prevents servers from
   attacking clients by giving them descriptors nobody else uses.

   All directory information is uploaded and downloaded with HTTP.

   Coordination among directory authorities is done client-side: clients
   compute a vote-like algorithm among the network-status documents they
   have, and base their decisions on the result.

1.1. What's different from 0.1.0.x?

   Clients used to download a signed concatenated set of router descriptors
   (called a "directory") from directory mirrors, regardless of which
   descriptors had changed.

   Between downloading directories, clients would download "network-status"
   documents that would list which servers were supposed to running.

   Clients would always believe the most recently published network-status
```

document they were served.

Routers used to upload fresh descriptors all the time, whether their keys and other information had changed or not.

1.2. Document meta-format

Router descriptors, directories, and running-routers documents all obey the following lightweight extensible information format.

The highest level object is a Document, which consists of one or more Items. Every Item begins with a KeywordLine, followed by one or more Objects. A KeywordLine begins with a Keyword, optionally followed by whitespace and more non-newline characters, and ends with a newline. A Keyword is a sequence of one or more characters in the set [A-Za-z0-9-]. An Object is a block of encoded data in pseudo-Open-PGP-style armor. (cf. RFC 2440)

More formally:

```
Document ::= (Item | NL)+
Item ::= KeywordLine Object*
KeywordLine ::= Keyword NL | Keyword WS ArgumentsChar+ NL
Keyword = KeywordChar+
KeywordChar ::= 'A' ... 'Z' | 'a' ... 'z' | '0' ... '9' | '-'
ArgumentChar ::= any printing ASCII character except NL.
WS = (SP | TAB)+
Object ::= BeginLine Base64-encoded-data EndLine
BeginLine ::= "-----BEGIN " Keyword "-----" NL
EndLine ::= "-----END " Keyword "-----" NL
```

The BeginLine and EndLine of an Object must use the same keyword.

When interpreting a Document, software MUST ignore any KeywordLine that starts with a keyword it doesn't recognize; future implementations MUST NOT require current clients to understand any KeywordLine not currently described.

Other implementations that want to extend Tor's directory format MAY introduce their own items. The keywords for extension items SHOULD start with the characters "x-" or "X-", to guarantee that they will not conflict with keywords used by future versions of Tor.

2. Router operation

ORs SHOULD generate a new router descriptor whenever any of the following events have occurred:

- A period of time (18 hrs by default) has passed since the last time a descriptor was generated.
- A descriptor field other than bandwidth or uptime has changed.
- Bandwidth has changed by at least a factor of 2 from the last time a descriptor was generated, and at least a given interval of time (20 mins by default) has passed since then.
- Its uptime has been reset (by restarting).

After generating a descriptor, ORs upload it to every directory authority they know, by posting it to the URL

```
http://<hostname:port>/tor/
```

2.1. Router descriptor format

Every router descriptor MUST start with a "router" Item; MUST end with a "router-signature" Item and an extra NL; and MUST contain exactly one instance of each of the following Items: "published" "onion-key" "signing-key" "bandwidth".

A router descriptor MAY have zero or one of each of the following Items, but MUST NOT have more than one: "contact", "uptime", "fingerprint", "hibernating", "read-history", "write-history", "eventdns", "platform", "family".

For historical reasons some options are prefixed with "opt ", for instance "opt fingerprint". This "opt " prefix should be ignored with the second word used as the keyword instead.

Additionally, a router descriptor MAY contain any number of "accept", "reject", and "opt" Items. Other than "router" and "router-signature", the items may appear in any order.

The items' formats are as follows:

"router" nickname address ORPort SocksPort DirPort

Indicates the beginning of a router descriptor. "address" must be an IPv4 address in dotted-quad format. The last three numbers indicate the TCP ports at which this OR exposes functionality. ORPort is a port at which this OR accepts TLS connections for the main OR protocol; SocksPort is deprecated and should always be 0; and DirPort is the port at which this OR accepts directory-related HTTP connections. If

```
170         any port is not supported, the value 0 is given instead of a port
171         number.
172
173      "bandwidth" bandwidth-avg bandwidth-burst bandwidth-observed
174
175         Estimated bandwidth for this router, in bytes per second.  The
176         "average" bandwidth is the volume per second that the OR is willing to
177         sustain over long periods; the "burst" bandwidth is the volume that
178         the OR is willing to sustain in very short intervals.  The "observed"
179         value is an estimate of the capacity this server can handle.  The
180         server remembers the max bandwidth sustained output over any ten
181         second period in the past day, and another sustained input.  The
182         "observed" value is the lesser of these two numbers.
183
184      "platform" string
185
186         A human-readable string describing the system on which this OR is
187         running.  This MAY include the operating system, and SHOULD include
188         the name and version of the software implementing the Tor protocol.
189
190      "published" YYYY-MM-DD HH:MM:SS
191
192         The time, in GMT, when this descriptor was generated.
193
194      "fingerprint"
195
196         A fingerprint (a HASH_LEN-byte of asn1 encoded public key, encoded in
197         hex, with a single space after every 4 characters) for this router's
198         identity key. A descriptor is considered invalid (and MUST be
199         rejected) if the fingerprint line does not match the public key.
200
201      "hibernating" 0|1
202
203         If the value is 1, then the Tor server was hibernating when the
204         descriptor was published, and shouldn't be used to build circuits.
205
206      "uptime"
207
208         The number of seconds that this OR process has been running.
209
210      "onion-key" NL a public key in PEM format
211
212         This key is used to encrypt EXTEND cells for this OR.  The key MUST be
213         accepted for at least 1 week after any new key is published in a
214         subsequent descriptor.
215
216      "signing-key" NL a public key in PEM format
217
218         The OR's long-term identity key.
219
220      "accept" exitpattern
221      "reject" exitpattern
222
223         These lines describe the rules that an OR follows when
224         deciding whether to allow a new stream to a given address.  The
225         'exitpattern' syntax is described below.  The rules are considered in
226         order; if no rule matches, the address will be accepted.  For clarity,
227         the last such entry SHOULD be accept *:* or reject *:*.
228
229      "router-signature" NL Signature NL
230
231         The "SIGNATURE" object contains a signature of the PKCS1-padded
232         hash of the entire router descriptor, taken from the beginning of the
233         "router" line, through the newline after the "router-signature" line.
234         The router descriptor is invalid unless the signature is performed
235         with the router's identity key.
236
237      "contact" info NL
238
239          Describes a way to contact the server's administrator, preferably
240          including an email address and a PGP key fingerprint.
241
242      "family" names NL
243
244          'Names' is a space-separated list of server nicknames or
245          hexdigests. If two ORs list one another in their "family" entries,
246          then OPs should treat them as a single OR for the purpose of path
247          selection.
248
249          For example, if node A's descriptor contains "family B", and node B's
250          descriptor contains "family A", then node A and node B should never
251          be used on the same circuit.
252
253      "read-history" YYYY-MM-DD HH:MM:SS (NSEC s) NUM,NUM,NUM,NUM,NUM... NL
254      "write-history" YYYY-MM-DD HH:MM:SS (NSEC s) NUM,NUM,NUM,NUM,NUM... NL
255
256          Declare how much bandwidth the OR has used recently. Usage is divided
257          into intervals of NSEC seconds.  The YYYY-MM-DD HH:MM:SS field
258          defines the end of the most recent interval.  The numbers are the
259          number of bytes used in the most recent intervals, ordered from
260          oldest to newest.
261
262      "eventdns" bool NL
```

```
263
264         Declare whether this version of Tor is using the newer enhanced
265         dns logic.  Versions of Tor without eventdns SHOULD NOT be used for
266         reverse hostname lookups.
267
268         [All versions of Tor before 0.1.2.2-alpha should be assumed to have
269          this option set to 0 if it is not present.  All Tor versions at
270          0.1.2.2-alpha or later should be assumed to have this option set to
271          1 if it is not present.  Until 0.1.2.1-alpha-dev, this option was
272          not generated, even when eventdns was in use.  With 0.2.0.1-alpha, the
273          old 'dnsworker' logic has been removed, rendering this option of
274          historical interest only.]
275
276 2.2. Nonterminals in router descriptors
277
278    nickname ::= between 1 and 19 alphanumeric characters, case-insensitive.
279    hexdigest ::= a '$', followed by 20 hexadecimal characters.
280       [Represents a server by the digest of its identity key.]
281
282    exitpattern ::= addrspec ":" portspec
283    portspec ::= "*" | port | port "-" port
284    port ::= an integer between 1 and 65535, inclusive.
285       [Some implementations incorrectly generate ports with value 0.
286        Implementations SHOULD accept this, and SHOULD NOT generate it.]
287
288    addrspec ::= "*" | ip4spec | ip6spec
289    ipv4spec ::= ip4 | ip4 "/" num_ip4_bits | ip4 "/" ip4mask
290    ip4 ::= an IPv4 address in dotted-quad format
291    ip4mask ::= an IPv4 mask in dotted-quad format
292    num_ip4_bits ::= an integer between 0 and 32
293    ip6spec ::= ip6 | ip6 "/" num_ip6_bits
294    ip6 ::= an IPv6 address, surrounded by square brackets.
295    num_ip6_bits ::= an integer between 0 and 128
296
297    bool ::= "0" | "1"
298
299    Ports are required; if they are not included in the router
300    line, they must appear in the "ports" lines.
301
302 3. Network status format
303
304    Directory authorities generate, sign, and compress network-status
305    documents.  Directory servers SHOULD generate a fresh network-status
306    document when the contents of such a document would be different from the
307    last one generated, and some time (at least one second, possibly longer)
308    has passed since the last one was generated.
309
310    The network status document contains a preamble, a set of router status
311    entries, and a signature, in that order.
312
313    We use the same meta-format as used for directories and router descriptors
314    in "tor-spec.txt".  Implementations MAY insert blank lines
315    for clarity between sections; these blank lines are ignored.
316    Implementations MUST NOT depend on blank lines in any particular location.
317
318    As used here, "whitespace" is a sequence of 1 or more tab or space
319    characters.
320
321    The preamble contains:
322
323       "network-status-version" -- A document format version.  For this
324          specification, the version is "2".
325       "dir-source" -- The authority's hostname, current IP address, and
326          directory port, all separated by whitespace.
327       "fingerprint" -- A base16-encoded hash of the signing key's
328          fingerprint, with no additional spaces added.
329       "contact" -- An arbitrary string describing how to contact the
330          directory server's administrator.  Administrators should include at
331          least an email address and a PGP fingerprint.
332       "dir-signing-key" -- The directory server's public signing key.
333       "client-versions" -- A comma-separated list of recommended client
334         versions.
335       "server-versions" -- A comma-separated list of recommended server
336         versions.
337       "published" -- The publication time for this network-status object.
338       "dir-options" -- A set of flags, in any order, separated by whitespace:
339           "Names" if this directory authority performs name bindings.
340           "Versions" if this directory authority recommends software versions.
341           "BadExits" if the directory authority flags nodes that it believes
342               are performing incorrectly as exit nodes.
343           "BadDirectories" if the directory authority flags nodes that it
344               believes are performing incorrectly as directory caches.
345
346    The dir-options entry is optional.  The "-versions" entries are required if
347    the "Versions" flag is present.  The other entries are required and must
348    appear exactly once. The "network-status-version" entry must appear first;
349    the others may appear in any order.  Implementations MUST ignore
350    additional arguments to the items above, and MUST ignore unrecognized
351    flags.
352
353    For each router, the router entry contains:  (This format is designed for
354    conciseness.)
355
```

```
      "r" -- followed by the following elements, in order, separated by
          whitespace:
          - The OR's nickname,
          - A hash of its identity key, encoded in base64, with trailing =
            signs removed.
          - A hash of its most recent descriptor, encoded in base64, with
            trailing = signs removed.  (The hash is calculated as for
            computing the signature of a descriptor.)
          - The publication time of its most recent descriptor, in the form
            YYYY-MM-DD HH:MM:SS, in GMT.
          - An IP address
          - An OR port
          - A directory port (or "0" for none")
      "s" -- A series of whitespace-separated status flags, in any order:
          "Authority" if the router is a directory authority.
          "BadExit" if the router is believed to be useless as an exit node
             (because its ISP censors it, because it is behind a restrictive
             proxy, or for some similar reason).
          "BadDirectory" if the router is believed to be useless as a
             directory cache (because its directory port isn't working,
             its bandwidth is always throttled, or for some similar
             reason).
          "Exit" if the router is useful for building general-purpose exit
             circuits.
          "Fast" if the router is suitable for high-bandwidth circuits.
          "Guard" if the router is suitable for use as an entry guard.
          "Named" if the router's identity-nickname mapping is canonical,
             and this authority binds names.
          "Stable" if the router is suitable for long-lived circuits.
          "Running" if the router is currently usable.
          "Valid" if the router has been 'validated'.
          "V2Dir" if the router implements this protocol.
      "v" -- The version of the Tor protocol that this server is running.  If
          the value begins with "Tor" SP, the rest of the string is a Tor
          version number, and the protocol is "The Tor protocol as supported
          by the given version of Tor."  Otherwise, if the value begins with
          some other string, Tor has upgraded to a more sophisticated
          protocol versioning system, and the protocol is "a version of the
          Tor protocol more recent than any we recognize."

      The "r" entry for each router must appear first and is required. The
      "s" entry is optional (see Section 3.1 below for how the flags are
      decided). Unrecognized flags on the "s" line and extra elements
      on the "r" line must be ignored.  The "v" line is optional.

   The signature section contains:

      "directory-signature" nickname-of-dirserver NL Signature

      Signature is a signature of this network-status document
      (the document up until the signature, including the line
      "directory-signature <nick>\n"), using the directory authority's
      signing key.

   We compress the network status list with zlib before transmitting it.

3.1. Establishing server status

   (This section describes how directory authorities choose which status
   flags to apply to routers, as of Tor 0.1.1.18-rc. Later directory
   authorities MAY do things differently, so long as clients keep working
   well.  Clients MUST NOT depend on the exact behaviors in this section.)

   In the below definitions, a router is considered "active" if it is
   running, valid, and not hibernating.

   "Valid" -- a router is 'Valid' if it is running a version of Tor not
   known to be broken, and the directory authority has not blacklisted
   it as suspicious.

   "Named" -- Directory authority administrators may decide to support name
   binding.  If they do, then they must maintain a file of
   nickname-to-identity-key mappings, and try to keep this file consistent
   with other directory authorities.  If they don't, they act as clients, and
   report bindings made by other directory authorities (name X is bound to
   identity Y if at least one binding directory lists it, and no directory
   binds X to some other Y'.)  A router is called 'Named' if the router
   believes the given name should be bound to the given key.

   "Running" -- A router is 'Running' if the authority managed to connect to
   it successfully within the last 30 minutes.

   "Stable" -- A router is 'Stable' if it is active, and either its
   uptime is at least the median uptime for known active routers, or
   its uptime is at least 30 days. Routers are never called stable if
   they are running a version of Tor known to drop circuits stupidly.
   (0.1.1.10-alpha through 0.1.1.16-rc are stupid this way.)

   "Fast" -- A router is 'Fast' if it is active, and its bandwidth is
   in the top 7/8ths for known active routers.

   "Guard" -- A router is a possible 'Guard' if it is 'Stable' and its
   bandwidth is above median for known active routers. If the total
```

bandwidth of active non-BadExit Exit servers is less than one third of the total bandwidth of all active servers, no Exit is listed as a Guard.

"Authority" -- A router is called an 'Authority' if the authority generating the network-status document believes it is an authority.

"V2Dir" -- A router supports the v2 directory protocol if it has an open directory port, and it is running a version of the directory protocol that supports the functionality clients need. (Currently, this is 0.1.1.9-alpha or later.)

Directory server administrators may label some servers or IPs as blacklisted, and elect not to include them in their network-status lists.

Authorities SHOULD 'disable' any servers in excess of 3 on any single IP. When there are more than 3 to choose from, authorities should first prefer authorities to non-authorities, then prefer Running to non-Running, and then prefer high-bandwidth to low-bandwidth. To 'disable' a server, the authority *should* advertise it without the Running or Valid flag.

Thus, the network-status list includes all non-blacklisted, non-expired, non-superseded descriptors.

4. Directory server operation

All directory authorities and directory mirrors ("directory servers") implement this section, except as noted.

4.1. Accepting uploads (authorities only)

When a router posts a signed descriptor to a directory authority, the authority first checks whether it is well-formed and correctly self-signed. If it is, the authority next verifies that the nickname in question is not already assigned to a router with a different public key.
Finally, the authority MAY check that the router is not blacklisted because of its key, IP, or another reason.

If the descriptor passes these tests, and the authority does not already have a descriptor for a router with this public key, it accepts the descriptor and remembers it.

If the authority _does_ have a descriptor with the same public key, the newly uploaded descriptor is remembered if its publication time is more recent than the most recent old descriptor for that router, and either:
- There are non-cosmetic differences between the old descriptor and the new one.
- Enough time has passed between the descriptors' publication times. (Currently, 12 hours.)

Differences between router descriptors are "non-cosmetic" if they would be sufficient to force an upload as described in section 2 above.

Note that the "cosmetic difference" test only applies to uploaded descriptors, not to descriptors that the authority downloads from other authorities.

4.2. Downloading network-status documents (authorities and caches)

All directory servers (authorities and mirrors) try to keep a fresh set of network-status documents from every authority. To do so, every 5 minutes, each authority asks every other authority for its most recent network-status document. Every 15 minutes, each mirror picks a random authority and asks it for the most recent network-status documents for all the authorities the authority knows about (including the chosen authority itself).

Directory servers and mirrors remember and serve the most recent network-status document they have from each authority. Other network-status documents don't need to be stored. If the most recent network-status document is over 10 days old, it is discarded anyway.
Mirrors SHOULD store and serve network-status documents from authorities they don't recognize, but SHOULD NOT use such documents for any other purpose. Mirrors SHOULD discard network-status documents older than 48 hours.

4.3. Downloading and storing router descriptors (authorities and caches)

Periodically (currently, every 10 seconds), directory servers check whether there are any specific descriptors (as identified by descriptor hash in a network-status document) that they do not have and that they are not currently trying to download.

If so, the directory server launches requests to the authorities for these descriptors, such that each authority is only asked for descriptors listed in its most recent network-status. When more than one authority lists the descriptor, we choose which to ask at random.

If one of these downloads fails, we do not try to download that descriptor from the authority that failed to serve it again unless we receive a newer network-status from that authority that lists the same descriptor.

Directory servers must potentially cache multiple descriptors for each router. Servers must not discard any descriptor listed by any current network-status document from any authority. If there is enough space to store additional descriptors, servers SHOULD try to hold those which clients are likely to download the most. (Currently, this is judged based on the interval for which each descriptor seemed newest.)

Authorities SHOULD NOT download descriptors for routers that they would immediately reject for reasons listed in 3.1.

4.4. HTTP URLs

"Fingerprints" in these URLs are base16-encoded SHA1 hashes.

The authoritative network-status published by a host should be available at:
   http://<hostname>/tor/status/authority.z

The network-status published by a host with fingerprint
<F> should be available at:
   http://<hostname>/tor/status/fp/<F>.z

The network-status documents published by hosts with fingerprints
<F1>,<F2>,<F3> should be available at:
   http://<hostname>/tor/status/fp/<F1>+<F2>+<F3>.z

The most recent network-status documents from all known authorities, concatenated, should be available at:
   http://<hostname>/tor/status/all.z

The most recent descriptor for a server whose identity key has a fingerprint of <F> should be available at:
   http://<hostname>/tor/server/fp/<F>.z

The most recent descriptors for servers with identity fingerprints
<F1>,<F2>,<F3> should be available at:
   http://<hostname>/tor/server/fp/<F1>+<F2>+<F3>.z

(NOTE: Implementations SHOULD NOT download descriptors by identity key fingerprint. This allows a corrupted server (in collusion with a cache) to provide a unique descriptor to a client, and thereby partition that client from the rest of the network.)

The server descriptor with (descriptor) digest <D> (in hex) should be available at:
   http://<hostname>/tor/server/d/<D>.z

The most recent descriptors with digests <D1>,<D2>,<D3> should be available at:
   http://<hostname>/tor/server/d/<D1>+<D2>+<D3>.z

The most recent descriptor for this server should be at:
   http://<hostname>/tor/server/authority.z
 [Nothing in the Tor protocol uses this resource yet, but it is useful for debugging purposes. Also, the official Tor implementations (starting at 0.1.1.x) use this resource to test whether a server's own DirPort is reachable.]

A concatenated set of the most recent descriptors for all known servers should be available at:
   http://<hostname>/tor/server/all.z

For debugging, directories SHOULD expose non-compressed objects at URLs like the above, but without the final ".z".
Clients MUST handle compressed concatenated information in two forms:
  - A concatenated list of zlib-compressed objects.
  - A zlib-compressed concatenated list of objects.
Directory servers MAY generate either format: the former requires less CPU, but the latter requires less bandwidth.

Clients SHOULD use upper case letters (A-F) when base16-encoding fingerprints. Servers MUST accept both upper and lower case fingerprints in requests.

5. Client operation: downloading information

Every Tor that is not a directory server (that is, those that do not have a DirPort set) implements this section.

5.1. Downloading network-status documents

Each client maintains an ordered list of directory authorities. Insofar as possible, clients SHOULD all use the same ordered list.

For each network-status document a client has, it keeps track of its publication time *and* the time when the client retrieved it. Clients consider a network-status document "live" if it was published within the last 24 hours.

Clients try to have a live network-status document hours from *every* authority, and try to periodically get new network-status documents from each authority in rotation as follows:

If a client is missing a live network-status document for any

```
   authority, it tries to fetch it from a directory cache.  On failure,
   the client waits briefly, then tries that network-status document
   again from another cache.  The client does not build circuits until it
   has live network-status documents from more than half the authorities
   it trusts, and it has descriptors for more than 1/4 of the routers
   that it believes are running.

   If the most recently _retrieved_ network-status document is over 30
   minutes old, the client attempts to download a network-status document.
   When choosing which documents to download, clients treat their list of
   directory authorities as a circular ring, and begin with the authority
   appearing immediately after the authority for their most recently
   retrieved network-status document.  If this attempt fails (either it
   fails to download at all, or the one it gets is not as good as the
   one it has), the client retries at other caches several times, before
   moving on to the next network-status document in sequence.

   Clients discard all network-status documents over 24 hours old.

   If enough mirrors (currently 4) claim not to have a given network status,
   we stop trying to download that authority's network-status, until we
   download a new network-status that makes us believe that the authority in
   question is running.  Clients should wait a little longer after each
   failure.

   Clients SHOULD try to batch as many network-status requests as possible
   into each HTTP GET.

   (Note: clients can and should pick caches based on the network-status
   information they have: once they have first fetched network-status info
   from an authority, they should not need to go to the authority directly
   again.)

5.2. Downloading and storing router descriptors

   Clients try to have the best descriptor for each router.  A descriptor is
   "best" if:
      * It is the most recently published descriptor listed for that router
        by at least two network-status documents.
        OR,
      * No descriptor for that router is listed by two or more
        network-status documents, and it is the most recently published
        descriptor listed by any network-status document.

   Periodically (currently every 10 seconds) clients check whether there are
   any "downloadable" descriptors.  A descriptor is downloadable if:
      - It is the "best" descriptor for some router.
      - The descriptor was published at least 10 minutes in the past.
        (This prevents clients from trying to fetch descriptors that the
        mirrors have probably not yet retrieved and cached.)
      - The client does not currently have it.
      - The client is not currently trying to download it.
      - The client would not discard it immediately upon receiving it.
      - The client thinks it is running and valid (see 6.1 below).

   If at least 16 known routers have downloadable descriptors, or if
   enough time (currently 10 minutes) has passed since the last time the
   client tried to download descriptors, it launches requests for all
   downloadable descriptors, as described in 5.3 below.

   When a descriptor download fails, the client notes it, and does not
   consider the descriptor downloadable again until a certain amount of time
   has passed. (Currently 0 seconds for the first failure, 60 seconds for the
   second, 5 minutes for the third, 10 minutes for the fourth, and 1 day
   thereafter.)  Periodically (currently once an hour) clients reset the
   failure count.

   No descriptors are downloaded until the client has downloaded more than
   half of the network-status documents.

   Clients retain the most recent descriptor they have downloaded for each
   router so long as it is not too old (currently, 48 hours), OR so long as
   it is recommended by at least one networkstatus AND no "better"
   descriptor has been downloaded.  [Versions of Tor before 0.1.2.3-alpha
   would discard descriptors simply for being published too far in the past.]
   [The code seems to discard descriptors in all cases after they're 5
   days old. True? -RD]

5.3. Managing downloads

   When a client has no live network-status documents, it downloads
   network-status documents from a randomly chosen authority.  In all other
   cases, the client downloads from mirrors randomly chosen from among those
   believed to be V2 directory servers.  (This information comes from the
   network-status documents; see 6 below.)

   When downloading multiple router descriptors, the client chooses multiple
   mirrors so that:
     - At least 3 different mirrors are used, except when this would result
       in more than one request for under 4 descriptors.
     - No more than 128 descriptors are requested from a single mirror.
     - Otherwise, as few mirrors as possible are used.
   After choosing mirrors, the client divides the descriptors among them
```

randomly.

After receiving any response client MUST discard any network-status documents and descriptors that it did not request.

6. Using directory information

Everyone besides directory authorities uses the approaches in this section to decide which servers to use and what their keys are likely to be. (Directory authorities just believe their own opinions, as in 3.1 above.)

6.1. Choosing routers for circuits.

Tor implementations only pay attention to "live" network-status documents. A network status is "live" if it is the most recently downloaded network status document for a given directory server, and the server is a directory server trusted by the client, and the network-status document is no more than 1 day old.

For time-sensitive information, Tor implementations focus on "recent" network-status documents. A network status is "recent" if it is live, and if it was published in the last 60 minutes. If there are fewer than 3 such documents, the most recently published 3 are "recent." If there are fewer than 3 in all, all are "recent.")

Circuits SHOULD NOT be built until the client has enough directory information: network-statuses (or failed attempts to download network-statuses) for all authorities, network-statuses for at more than half of the authorities, and descriptors for at least 1/4 of the servers believed to be running.

A server is "listed" if it is included by more than half of the live network status documents. Clients SHOULD NOT use unlisted servers.

Clients believe the flags "Valid", "Exit", "Fast", "Guard", "Stable", and "V2Dir" about a given router when they are asserted by more than half of the live network-status documents. Clients believe the flag "Running" if it is listed by more than half of the recent network-status documents.

These flags are used as follows:

- Clients SHOULD NOT use non-'Valid' or non-'Running' routers unless requested to do so.

- Clients SHOULD NOT use non-'Fast' routers for any purpose other than very-low-bandwidth circuits (such as introduction circuits).

- Clients SHOULD NOT use non-'Stable' routers for circuits that are likely to need to be open for a very long time (such as those used for IRC or SSH connections).

- Clients SHOULD NOT choose non-'Guard' nodes when picking entry guard nodes.

- Clients SHOULD NOT download directory information from non-'V2Dir' caches.

6.2. Managing naming

In order to provide human-memorable names for individual server identities, some directory servers bind names to IDs. Clients handle names in two ways:

When a client encounters a name it has not mapped before:

> If all the live "Naming" network-status documents the client has claim that the name binds to some identity ID, and the client has at least three live network-status documents, the client maps the name to ID.

When a user tries to refer to a router with a name that does not have a mapping under the above rules, the implementation SHOULD warn the user. After giving the warning, the implementation MAY use a router that at least one Naming authority maps the name to, so long as no other naming authority maps that name to a different router. If no Naming authority maps the name to a router, the implementation MAY use any router that advertises the name.

Not every router needs a nickname. When a router doesn't configure a nickname, it publishes with the default nickname "Unnamed". Authorities SHOULD NOT ever mark a router with this nickname as Named; client software SHOULD NOT ever use a router in response to a user request for a router called "Unnamed".

6.3. Software versions

An implementation of Tor SHOULD warn when it has fetched (or has attempted to fetch and failed four consecutive times) a network-status for each authority, and it is running a software version not listed on more than half of the live "Versioning" network-status documents.

6.4. Warning about a router's status.

```
821
822     If a router tries to publish its descriptor to a Naming authority
823     that has its nickname mapped to another key, the router SHOULD
824     warn the operator that it is either using the wrong key or is using
825     an already claimed nickname.
826
827     If a router has fetched (or attempted to fetch and failed four
828     consecutive times) a network-status for every authority, and at
829     least one of the authorities is "Naming", and no live "Naming"
830     authorities publish a binding for the router's nickname, the
831     router MAY remind the operator that the chosen nickname is not
832     bound to this key at the authorities, and suggest contacting the
833     authority operators.
834
835     ...
836
837 6.5. Router protocol versions
838
839     A client should believe that a router supports a given feature if that
840     feature is supported by the router or protocol versions in more than half
841     of the live networkstatus's "v" entries for that router.  In other words,
842     if the "v" entries for some router are:
843         v Tor 0.0.8pre1               (from authority 1)
844         v Tor 0.1.2.11                (from authority 2)
845         v FutureProtocolDescription 99 (from authority 3)
846     then the client should believe that the router supports any feature
847     supported by 0.1.2.11.
848
849     This is currently equivalent to believing the median declared version for
850     a router in all live networkstatuses.
851
852 7. Standards compliance
853
854     All clients and servers MUST support HTTP 1.0.
855
856 7.1. HTTP headers
857
858   Servers MAY set the Content-Length: header.  Servers SHOULD set
859   Content-Encoding to "deflate" or "identity".
860
861   Servers MAY include an X-Your-Address-Is: header, whose value is the
862   apparent IP address of the client connecting to them (as a dotted quad).
863   For directory connections tunneled over a BEGIN_DIR stream, servers SHOULD
864   report the IP from which the circuit carrying the BEGIN_DIR stream reached
865   them.
866
867   Servers SHOULD disable caching of multiple network statuses or multiple
868   router descriptors.  Servers MAY enable caching of single descriptors,
869   single network statuses, the list of all router descriptors, a v1
870   directory, or a v1 running routers document.  XXX mention times.
871
872 7.2. HTTP status codes
873
874   XXX We should write down what return codes dirservers send in what
875   situations.
876
877
```