Bittorrent over I2P - I2P

13-16 minutes

This page was last updated in 2024-11 and is accurate for router version 0.9.64.

There are several bittorrent clients and trackers on I2P. As I2P addressing uses a Destination instead of an IP and port, minor changes are required to tracker and client software for operation on I2P. These changes are specified below. Note carefully the guidelines for compatibility with older I2P clients and trackers.

This page specifies protocol details common to all clients and trackers. Specific clients and trackers may implement other unique features or protocols.

We welcome additional ports of client and tracker software to I2P.

General Guidance for Developers

Most non-Java bittorrent clients will connect to I2P via SAMv3. SAM sessions (or inside I2P, tunnel pools or sets of tunnels) are designed to be long-lived. Most bittorrent clients will only need one session, created at startup and closed on exit. I2P is different from Tor, where circuits may be rapidly created and discarded. Think carefully and consult with I2P developers before designing your application to use more than one or two simultaneous sessions, or to rapidly create and discard them. Bittorrent clients must not create a unique session for every connection. Design your client to use the same session for announces and client connections.

Also, please ensure your client settings (and guidance to users about router settings, or router defaults if you bundle a router) will result in your users contributing more resources to the network than they consume. I2P is a peer-to-peer network, and the network cannot survive if a popular application drives the network into permanent congestion.

Do not provide support for bittorrent through an I2P outproxy to the clearnet as it will probably be blocked. Consult with outproxy operators for guidance.

The Java I2P and i2pd router implementations are independent and have minor differences in behavior, feature support, and defaults. Please test your application with the latest version of both routers.

i2pd SAM is enabled by default; Java I2P SAM is not. Provide instructions to your users on how to enable SAM in Java I2P (via /configclients in the router console), and/or provide a good error message to the user if the initial connect fails, e.g. "ensure that I2P is running and the SAM interface is enabled".

The Java I2P and i2pd routers have different defaults for tunnel quantities. The Java default is 2 and the i2pd default is 5. For most low- to medium-bandwidth and low- to medium-connection counts, 3 is sufficient. Please specify the tunnel quantity in the SESSION CREATE message to get consistent performance with the Java I2P and i2pd routers.

I2P supports multiple signature and encryption types. For compatibility, I2P defaults to old and inefficient types, so all clients should specify newer types.

If using SAM, the signature type is specified in the DEST GENERATE and SESSION CREATE (for transient) commands. All clients should set SIGNATURE_TYPE=7 (Ed25519).

The encryption type is specified in the SAM SESSION CREATE command or in i2cp options. Multiple encryption types are allowed. Some trackers support ECIES-X25519, some support ElGamal, and some support both. Clients should set i2cp.leaseSetEncType=4,0 (for ECIES-X25519 and ElGamal) so that they may connect to both.

DHT support requires SAM v3.3 PRIMARY and SUBSESSIONS for TCP and UDP over the same session. This will require substantial development effort on the client side, unless the client is written in Java. i2pd does not currently support SAM v3.3. libtorrent does not currently support SAM v3.3.

Without DHT support, you may wish to automatically announce to a configurable list of known open trackers so that magnet links will work. Consult with I2P users for information on currently-up open trackers and keep your defaults up-to-date. Supporting the i2p_pex extension will also help alleviate the lack of DHT support.

For more guidance to developers on ensuring your application uses only the resources it needs, please see the SAMv3 specification and our guide to bundling I2P with your application. Contact I2P or i2pd developers for further assistance.

Announces

Clients generally include a fake port=6881 parameter in the announce, for compatibility with older trackers. Trackers may ignore the port parameter, and should not require it.

The ip parameter is the base 64 of the client's Destination, using the I2P Base 64 alphabet [A-Z][a-z][0-9]-~. Destinations are 387+ bytes, so the Base 64 is 516+ bytes. Clients generally append ".i2p" to the Base 64 Destination for compatibility with older trackers. Trackers should not require an appended ".i2p".

Other parameters are the same as in standard bittorrent.

Current Destinations for clients are 387 or more bytes (516 or more in Base 64 encoding). A reasonable maximum to assume, for now, is 475 bytes. As the tracker must decode the Base64 to deliver compact responses (see below), the tracker should probably decode and reject bad Base64 when announced.

The default response type is non-compact. Clients may request a compact response with the parameter compact=1. A tracker may, but is not required to, return a compact response when requested. Note: All popular trackers now support compact responses and at least one requires compact=1 in the announce. All clients should request and support compact responses.

Developers of new I2P clients are strongly encouraged to implemenent announces over their own tunnel rather than the HTTP client proxy at port 4444. Doing so is both more efficient and it allows destination enforcement by the tracker (see below).

There are no known I2P clients or trackers that currently support UDP announce/responses.

Non-Compact Tracker Responses

The non-compact response is just as in standard bittorrent, with an I2P "ip". This is a long base64-encoded "DNS string", probably with a ".i2p" suffix.

Trackers generally include a fake port key, or use the port from the announce, for compatibility with older clients. Clients must ignore the port parameter, and should not require it.

The value of the ip key is the base 64 of the client's Destination, as described above. Trackers generally append ".i2p" to the Base 64 Destination if it wasn't in the announce ip, for compatibility with older clients. Clients should not require an appended ".i2p" in the responses.

Other response keys and values are the same as in standard bittorrent.

Compact Tracker Responses

In the compact response, the value of the "peers" dictionary key is a single byte string, whose length is a multiple of 32 bytes. This string contains the concatenated 32-byte SHA-256 Hashes of the binary Destinations of the peers. This hash must be computed by the tracker, unless destination enforcement (see below) is used, in which case the hash delivered in the X-I2P-DestHash or X-I2P-DestB32 HTTP headers may be converted to binary and stored. The peers key may be absent, or the peers value may be zero-length.

While compact response support is optional for both clients and trackers, it is highly recommended as it reduces the nominal response size by over 90%.

Destination Enforcement

Some, but not all, I2P bittorrent clients announce over their own tunnels. Trackers may choose to prevent spoofing by requiring this, and verifying the client's Destination using HTTP headers added by the I2PTunnel HTTP Server tunnel. The headers are X-I2P-DestHash, X-I2P-DestB64, and X-I2P-DestB32, which are different formats for the same information. These headers cannot be spoofed by the client. A tracker enforcing destinations need not require the ip announce parameter at all.

As several clients use the HTTP proxy instead of their own tunnel for announces, destination enforcement will prevent usage by those clients unless or until those clients are converted to announcing over their own tunnel.

Unfortunately, as the network grows, so will the amount of maliciousness, so we expect that all trackers will eventually enforce destinations. Both tracker and client developers should anticipate it.

Announce Host Names

Announce URL host names in torrent files generally follow the I2P naming standards. In addition to host names from address books and ".b32.i2p" Base 32 hostnames, the full Base 64 Destination (with [or without?] ".i2p" appended) should be supported. Non-open trackers should recognize their own host name in any of these formats.

To preserve anonymity, clients should generally ignore non-I2P announce URLs in torrent files.

Client Connections

Client-to-client connections use the standard protocol over TCP. There are no known I2P clients that currently support uTP communication.

I2P uses 387+ byte Destinations for addresses, as explained above.

If the client has only the hash of the destination (such as from a compact response or PEX), it must perform a lookup by encoding it with Base 32, appending ".b32.i2p", and querying the Naming Service, which will return the full Destination if available.

If the client has a peer's full Destination it received in a non-compact response, it should use it directly in the connection setup. Do not convert a Destination back to a Base 32 hash for lookup, this is quite inefficient.

Cross-Network Prevention

To preserve anonymity, I2P bittorrent clients generally do not support non-I2P announces or peer connections. I2P HTTP outproxies often block announces. There are no known SOCKS outproxies supporting bittorrent traffic.

To prevent usage by non-I2P clients via an HTTP inproxy, I2P trackers often block accesses or announces that contain an X-Forwarded-For HTTP header. Trackers should reject standard network announces with IPv4 or IPv6 IPs, and not deliver them in responses.

PEX

I2P PEX is based on ut_pex. As there does not appear to be a formal specification of ut_pex available, it may be necessary to review the libtorrent source for assistance. It is an extension message, identified as "i2p_pex" in the extension handshake. It contains a bencoded dictionary with up to 3 keys, "added", "added.f", and "dropped". The added and dropped values are each a single byte string, whose length is a multiple of 32 bytes. These byte strings are the concatenated SHA-256 Hashes of the binary Destinations of the peers. This is the same format as the peers dictionary value in the i2p compact response format specified above. The added.f value, if present, is the same as in ut_pex.

DHT

DHT support is included in the i2psnark client as of version 0.9.2. Preliminary differences from BEP 5 are described below, and are subject to change. Contact the I2P developers if you wish to develop a client supporting DHT.

Unlike standard DHT, I2P DHT does not use a bit in the options handshake, or the PORT message. It is advertised with an extension message, identified as "i2p_dht" in the extension handshake. It contains a bencoded dictionary with two keys, "port" and "rport", both integers.

The UDP (datagram) port listed in the compact node info is used to receive repliable (signed) datagrams. This is used for queries, except for announces. We call this the "query port". This is the "port" value from the extension message. Queries use I2CP protocol number 17.

In addition to that UDP port, we use a second datagram port equal to the query port + 1. This is used to receive unsigned (raw) datagrams for replies, errors, and announces. This port provides increased efficiency since replies contain tokens sent in the query, and need not be signed. We call this the "response port". This is the "rport" value from the extension message. It must be 1 + the query port. Responses and announces use I2CP protocol number 18.

Compact peer info is 32 bytes (32 byte SHA256 Hash) instead of 4 byte IP + 2 byte port. There is no peer port. In a response, the "values" key is a list of strings, each containing a single compact peer info.

Compact node info is 54 bytes (20 byte Node ID + 32 byte SHA256 Hash + 2 byte port) instead of 20 byte Node ID + 4 byte IP + 2 byte port. In a response, the "nodes" key is a single byte string with concatenated compact node info.

Secure node ID requirement: To make various DHT attacks more difficult, the first 4 bytes of the Node ID must match the first 4 bytes of the destination Hash, and the next two bytes of the Node ID must match the next two bytes of the destination hash exclusive-ORed with the port.

In a torrent file, the trackerless torrent dictionary "nodes" key is TBD. It could be a list of 32 byte binary strings (SHA256 Hashes) instead of a list of lists containing a host string and a port integer. Alternatives: A single byte string with concatenated hashes, or a list of strings alone.

Datagram (UDP) Trackers

UDP tracker support in clients and trackers is not yet available. Preliminary differences from BEP 15 are described below, and are subject to change. Contact the I2P developers if you wish to develop a client or tracker supporting datagram announces.

See Proposal 160.

Additional Information

- I2P bittorrent standards are generally discussed on zzz.i2p.
- A chart of current tracker software capabilities is also available there.
- The I2P bittorrent FAQ
- DHT on I2P discussion