WIKIPEDIA

Torrent file

In the BitTorrent file distribution system, a **torrent file** or **meta-info file** is a computer file that contains metadata about files and folders to be distributed, and usually also a list of the network locations of trackers, which are computers that help participants in the system find each other and form efficient distribution groups called *swarms*.[1] Torrent files are normally named with the extension `.torrent`.

Torrent files

Filename extension	`.torrent`
Internet media type	`application/x-bittorrent`
Standard	BEP-0003 (v1),[1] BEP-0052 (v2)[2]

A torrent file acts like a table of contents (index) that allows computers to find information through the use of a torrent client. With the help of a torrent file, one can download small parts of the original file from computers that have already downloaded it. These "peers" allow for downloading of the file in addition to, or in place of, the primary server. A torrent file does not contain the content to be distributed; it only contains information about those files, such as their names, folder structure, sizes, and cryptographic hash values for verifying file integrity.

The torrent system has been created to ease the load on central servers, as instead of having individual clients fetch files from the server, torrent can crowd-source the bandwidth needed for the file transfer and reduce the time needed to download large files. Many free/freeware programs and operating systems, such as the various Linux distributions offer a torrent download option for users seeking the aforementioned benefits. Other large downloads, such as media files, are often torrented as well.

Background

Typically, Internet access is asymmetrical, supporting greater download speeds than upload speeds, limiting the bandwidth of each download, and sometimes enforcing bandwidth caps and periods where systems are not accessible. This creates inefficiency when many people want to obtain the same set of files from a single source; the source must always be online and must have massive outbound bandwidth. The torrent protocol addresses this by decentralizing the distribution, leveraging the ability of people to network "peer-to-peer", among themselves.

Each file to be distributed is divided into small information chunks called *pieces*. Downloading peers achieve high download speeds by requesting multiple pieces from different computers simultaneously in the swarm. Once obtained, these pieces are usually immediately made available for download by others in the swarm. In this way, the burden on the network is spread among the downloaders, rather than concentrating at a central distribution hub or cluster. As long as all the pieces are available, peers (downloaders and uploaders) can come and go; no one peer needs to have all the chunks or to even stay connected to the swarm in order for distribution to continue among the other peers.

A small torrent file is created to represent a file or folder to be shared. The torrent file acts as the key to initiating downloading of the actual content. Someone interested in receiving the shared file or folder first obtains the corresponding torrent file, either by directly downloading it or by using a magnet link. The user then opens that file in a torrent client, which automates the rest of the process. In order to learn the internet locations of peers who may be sharing pieces, the client connects to the trackers named in the torrent file,

and/or achieves a similar result through the use of distributed hash tables. Then the client connects directly to the peers in order to request pieces and otherwise participate in a swarm. The client may also report progress to trackers, to help the tracker with its peer recommendations.

When the client has all the pieces, the torrent client assembles them into a usable form. They may also continue sharing the pieces, elevating their status to that of *seeder* rather than an ordinary peer.

File struct

A torrent file contains a list of files and integrity metadata about all the pieces, and optionally contains a large list of trackers.

A torrent file is a bencoded dictionary with the following keys (the keys in any bencoded dictionary are lexicographically ordered):

- `announce`—the URL of the high tracker
- `info`—this maps to a dictionary whose keys are very dependent on whether one or more files are being shared:
  - `files`—a list of dictionaries each corresponding to a file (only when multiple files are being shared). Each dictionary has the following keys:
    - `length`—size of the file in bytes.
    - `path`—a list of strings corresponding to subdirectory names, the last of which is the actual file name
  - `length`—size of the file in bytes (only when one file is being shared though)
  - `name`—suggested filename where the file is to be saved (if one file)/suggested directory name where the files are to be saved (if multiple files)
  - `piece length`—number of bytes per piece. This is commonly 2^8 KiB = 256 KiB = 262,144 B.
  - `pieces`—a hash list, i.e., a concatenation of each piece's SHA-1 hash. As SHA-1 returns a 160-bit hash, `pieces` will be a string whose length is a multiple of 20 bytes. If the torrent contains multiple files, the pieces are formed by concatenating the files in the order they appear in the `files` dictionary (i.e., all pieces in the torrent are the full piece length except for the last piece, which may be shorter).

All strings must be UTF-8 encoded, except for `pieces`, which contains binary data.

A torrent is uniquely identified by an *infohash*, a SHA-1 hash calculated over the contents of the `info` dictionary in bencode form. Changes to other portions of the torrent do not affect the hash. This hash is used to identify the torrent to other peers via DHT and to the tracker. It is also used in magnet links.

BitTorrent v2

The BitTorrent v2 protocol (BEP-0052) introduces a new definition of the torrent file.[2] The basic structure is:

- `announce`—the URL of the tracker
- `info`—this maps to a dictionary whose keys are dependent on whether one or more files are being shared:
  - `name`—suggested directory name where the files are to be saved
  - `piece length`—number of bytes per piece. This is commonly 2^8 KiB = 256 KiB = 262,144 B. In v2, it must be a power of 2.
  - `meta version`—number, "2".
  - `file tree`—a tree of dictionaries. Each key represents a directory name or a file name. The file is
    - `length`—size of the file in bytes (only when one file is being shared though)

    - `piece root`—For non-empty files this is the root hash of a merkle tree with a branching factor of 2, constructed from 16KiB blocks of the file.
- `piece layers`—A dictionary of strings, containing the new type of merkle root hashes for each piece.

The new format uses SHA-256 in both the piece-hashing and the *infohash*, replacing the broken SHA-1 hash. The "btmh" magnet link would contain the full 32-byte hash, while communication with trackers and on the DHT uses the 20-byte truncated version to fit into the old message structure.[2] It is possible to construct a torrent file with only updated new fields for a "v2" torrent, or with both the old and new fields for a "hybrid" format. However, as a torrent would have different infohashes in v1 and v2 networks, two swarms would form, requiring special handling by the client to merge the two.[3]

A core feature of the new format is its application of merkle trees, allowing for 16KiB blocks of a piece to be individually verified and re-downloaded. Each file now always occupy whole piece sizes and have an independent merkle root hash, so that it's possible to find duplicate files across unrelated torrent files of any piece length. The file size is not reduced (assuming piece size stays the same; v2's tree structure allows larger pieces with fewer ill effects), but the `info` dictionary required for magnet links are (only in v2-only torrents).[3]

Extensions

A torrent file can also contain additional metadata defined in extensions to the BitTorrent specification.[4] These are known as "BitTorrent Enhancement Proposals." Examples of such proposals include metadata for stating who created the torrent, and when.

Accepted extensions

These extensions have been deployed in one or more implementations as well as having been proven useful through consistent and widespread use. While they may require minor revisions, they are largely considered to be complete, only awaiting the blessing of Bram Cohen in order to be elevated to the status of Final/Active Process.

Distributed hash tables

BEP-0005[5] extends BitTorrent to support distributed hash tables, specifically Mainline DHT.

A trackerless torrent dictionary does not have an `announce` key. Instead, a trackerless torrent has a `nodes` key:

```
{
  # ...
  'nodes': [["<host>", <port>], ["<host>", <port>], ...],
  # ...
}
```

For example,

```
  'nodes': [["127.0.0.1", 6881], ["your.router.node", 4804]],
```

The specification recommends that `nodes` "should be set to the K closest nodes in the torrent generating client's routing table. Alternatively, the key could be set to a known good node such as one operated by the person generating the torrent."

Multiple trackers

BEP-0012[6] extends BitTorrent to support multiple trackers.

A new key, `announce-list`, is placed in the top-most dictionary (i.e., with `announce` and `info`)

```
{
  # ...
  'announce-list': [['<tracker1-url>']['<tracker2-url>']],
  # ...
}
```

HTTP seeds

BEP-0019[7] is one of two extensions allowing HTTP seeds to be used in BitTorrent.

In BEP-0019, a new key `url-list`, is placed in the top-most list. The client uses the links to assemble ordinary HTTP URLs – no server-side support is required. This feature is very commonly used by open source projects offering software downloads. Web seeds allow smart selection and simultaneous use of mirror sites, P2P or HTTP(S), by the client. Doing so reducing the load on the project's servers while maximizing download speed. MirrorBrain automatically generates torrents with web seeds.

Private torrents

BEP-0027[8] extends BitTorrent to support private torrents.

A new key, `private`, is placed in the `info` dictionary. This key's value is 1 if the torrent is private:

```
{
  # ...
  'info': {
    # ...
    'private': 1,
    # ...
  },
  # ...
}
```

Private torrents are to be used with a *private tracker*. Such a tracker restricts access to torrents it tracks by checking the peer's IP, refusing to provide a peer list if the IP is unknown. The peer itself is usually registered to the tracker via a gated online community; the private tracker typically also keep statistics of data transfer for use in the community.

Decentralized methods like DHT, PeX, LSD are disabled to maintain the centralized control. A private torrent can be manually edited to remove the private flag, but doing so will change the info-hash (deterministically), forming a separate "swarm" of peers. On the other hand, changing the tracker list will not change the hash. The flag does not offer true privacy, instead operating as a gentlemen's agreement.

Draft extensions

These extensions are under consideration for standardization. Most are already widely adopted as *de facto* standards.

HTTP seeds

BEP-0017[9] extends BitTorrent to support HTTP seeds, later more commonly termed "web seeds" to be inclusive of HTTPS.

In BEP-0017, a new key, `httpseeds`, is placed in the top-most list (i.e., with `announce` and `info`). This key's value is a list of web addresses where torrent data can be retrieved. Special server support is required. It remains at Draft status.

```
{
  # ...
  'httpseeds': ['http://www.site1.com/source1.php', 'http://www.site2.com/source2.php'],
  # ...
}
```

Merkle trees

BEP-0030[10] extends BitTorrent to support Merkle trees (originally implemented in Tribler). The purpose is to reduce the file size of torrent files, which reduces the burden on those that serve torrent files.

A torrent file using Merkle trees does not have a `pieces` key in the `info` list. Instead, such a torrent file has a `root_hash` key in the `info` list. This key's value is the root hash of the Merkle hash:

```
{
  # ...
  'info': {
    # ...
    'root hash': <binary SHA1 hash>,
    # ...
  },
  # ...
}
```

BitTorrent v2 uses a different type of Merkle tree.[3]

Examples

Single file

A de-bencoded torrent file (with `piece length` 256 KiB = 262,144 bytes) for a file `debian-503-amd64-CD-1.iso` (whose size is 678 301 696 bytes) might look like:

```
{
    'announce': 'http://bttracker.debian.org:6969/announce',
    'info':
    {
        'length': 678301696,
        'name': 'debian-503-amd64-CD-1.iso',
        'piece length': 262144,
        'pieces': <binary SHA1 hashes>
    }
}
```

Note: `pieces` here would be a 51 KiB value ($\left\lceil \frac{\texttt{length}}{\texttt{piece length}} \right\rceil \times 160 = 414080 \text{ bits}$).

Multiple files

A de-bencoded torrent file (with `'piece length'` 256 KiB = 262144 B) for two files, `111.txt` and `222.txt`, might look like:

```
{
    'announce': 'http://tracker.example.com/announce',
    'info':
```

```
    {
        'files':
        [
            {'length': 111, 'path': ['111.txt']},
            {'length': 222, 'path': ['222.txt']}
        ],
        'name': 'directoryName',
        'piece length': 262144,
        'pieces': <binary SHA1 hashes>
    }
}
```

Hybrid, multiple files

See also

- Glossary of BitTorrent terms
- Magnet links

References

1. "BEP-0003: The BitTorrent Protocol Specification" (http://www.bittorrent.org/beps/bep_0003.html). Bittorrent.org. Archived (https://web.archive.org/web/20190726102940/http://www.bittorrent.org/beps/bep_0003.html) from the original on 2019-07-26. Retrieved 2009-10-22.
2. "bep_0052.rst_post" (http://bittorrent.org/beps/bep_0052.html). *bittorrent.org*. Archived (https://web.archive.org/web/20201112030826/http://bittorrent.org/beps/bep_0052.html) from the original on 2020-11-12. Retrieved 2023-02-09.
3. "BitTorrent v2" (https://blog.libtorrent.org/2020/09/bittorrent-v2/). *Libtorrent*. September 2020. Archived (https://web.archive.org/web/20201030011550/https://blog.libtorrent.org/2020/09/bittorrent-v2/) from the original on 2020-10-30. Retrieved 2023-02-09.
4. "BEP-0000: Index of BitTorrent Enhancement Proposals" (http://www.bittorrent.org/beps/bep_0000.html). Bittorrent.org. Archived (https://web.archive.org/web/20100211001952/http://www.bittorrent.org/beps/bep_0000.html) from the original on 2010-02-11. Retrieved 2009-10-22.
5. "BEP-0005: DHT Protocol" (http://www.bittorrent.org/beps/bep_0005.html). Bittorrent.org. Archived (https://web.archive.org/web/20100213010149/http://www.bittorrent.org/beps/bep_0005.html) from the original on 2010-02-13. Retrieved 2009-10-22.
6. "BEP-0012: Multitracker Metadata Extension" (http://www.bittorrent.org/beps/bep_0012.html). Bittorrent.org. Archived (https://web.archive.org/web/20121227233108/http://bittorrent.org/beps/bep_0012.html) from the original on 2012-12-27. Retrieved 2009-10-22.
7. "bep_0019.rst_post" (http://www.bittorrent.org/beps/bep_0019.html). *www.bittorrent.org*.
8. "BEP-0027: Private Torrents" (http://www.bittorrent.org/beps/bep_0027.html). Bittorrent.org. Archived (https://web.archive.org/web/20130324181003/http://www.bittorrent.org/beps/bep_0027.html) from the original on 2013-03-24. Retrieved 2009-10-22.
9. "BEP-0017: HTTP Seeding" (http://www.bittorrent.org/beps/bep_0017.html). Bittorrent.org. Archived (https://web.archive.org/web/20131213074432/http://www.bittorrent.org/beps/bep_0017.html) from the original on 2013-12-13. Retrieved 2009-10-22.
10. "BEP-0030: Merkle hash torrent extension" (http://www.bittorrent.org/beps/bep_0030.html). Bittorrent.org. Archived (https://web.archive.org/web/20090914143238/http://www.bittorrent.org/beps/bep_0030.html) from the original on 2009-09-14. Retrieved 2009-10-22.

External links

- Official BitTorrent Specification (http://www.bittorrent.org/beps/bep_0003.html)
- BitTorrent Definition (https://techobservatory.com/what-are-torrents-and-how-do-they-work/#tobs-bittorrent)