BitTorrent

Masood Khosroshahy

July 2009

Tech. Report

Copyright © 2009 Masood Khosroshahy, All rights reserved.
www.masoodkh.com

Contents

BitTorrent protocol [1, 2] has emerged as the most popular P2P protocol over the past years. The core BitTorrent protocol has been designed and implemented by Bram Cohen in 2001. The protocol is especially useful for distributing large popular files (like open-source operating system distributions) as its performance improves as the number of interested connected peers increases. The way in which BitTorrent operates lessens the burden (hardware costs and bandwidth resources) of servers hosting the files and distributes that burden among all the peers currently connected, reducing costs significantly for original content distributors as a result. Connected peers share the task of serving the content to newly-connected peers and a "tit-for-tat" mechanism ensures fairness among all the peers. This method of content sharing also improves redundancy in the overlay network (formed around that specific content), as a probable malfunctioning of the original content provider does not render the content unavailable. In what follows, we explain the functionality of the BitTorrent protocol and its various system components.

1 Basic Concepts

BitTorrent technology has some specific terminologies which are briefly introduced in this section. The system dynamics and how these concepts relate to each other is treated later on.

Torrent The "system intelligence" is stored in a meta-data file called "Torrent" in text format. It is called meta-data since it does not contain any portion of the content to be shared; it only contains "information" about the content and how a new peer could join the system to acquire that content. This file is usually distributed by web servers hosting torrent files. After being downloaded to the peer's computer, it is read by a BitTorrent client to start the process of getting the actual content.

Seed A "Seed" is a peer who has a complete copy of the content whose information, or "meta-data", appears in the Torrent file. "To Seed" refers to the initial process of making the content available by the original distributor.

Leech-Leecher A "Leech" or "Leecher" is a peer who does not have a complete copy of the content. A newly-joined peer to the network is necessarily a Leecher, since it does not have any pieces of the content to share (or upload) and it can only download at first.

Swarm The term "Swarm" refers to the group of the peers (Seeds and Leechers) which are interconnected. A swarm exists around a content whose information has been mentioned in the Torrent meta-data file.

Tracker "Tracker" is a server which coordinates the swarm. The URL address of the Tracker is mentioned in the Torrent meta-data file. When a new peer wants to acquire the content, it contacts the Tracker for information about the swarm. The Tracker responds with information, more specifically, with IP addresses and port numbers of the peers (Seeds and Leechers) in the swarm. During the transfer, peers send statistics about their state and how much they have downloaded/uploaded to the Tracker.

Distributed Copy The Leechers in a swarm can collectively have a "Distributed Copy" of the content. It happens that, at some point, there is no longer a Seed connected to the swarm and as along as there is at least one Distributed Copy in the swarm, every Leecher can eventually acquire a complete copy of the content.

Choke-Unchoke Each peer in the swarm can express their desire, or lack thereof, to transmit a content piece to a requesting peer with "Unchoke" message, or "Choke" message, respectively. A peer in a swarm maintains TCP connections with a number of other peers (usually 50 peers) in the swarm. By default, a peer sends content upstream to four peers at a time; hence the peer sends "Unchoke" messages to these four peers and will send "Choke" messages to all the other peers.

Interested-Not Interested Corresponding to the "Choke-Unchoke" messages in the upstream direction, there are "Interested-Not Interested" messages in the downstream direction. A Leecher that is interested in a piece that the other peer has sends an "Interested" message on that connection. If the other peer does not have any pieces that the Leecher lacks, the Leecher obviously sends a "Not Interested" message.

Piece-Chunk The person who creates the Torrent meta-data file and first Seeds the content, decides on a "Piece" size. The content is divided into equally-sized Pieces except probably the last piece which is the last remaining part. The Piece size is usually chosen 256KB; however, the user is free to choose any other amount (The larger the file, the bigger the Piece

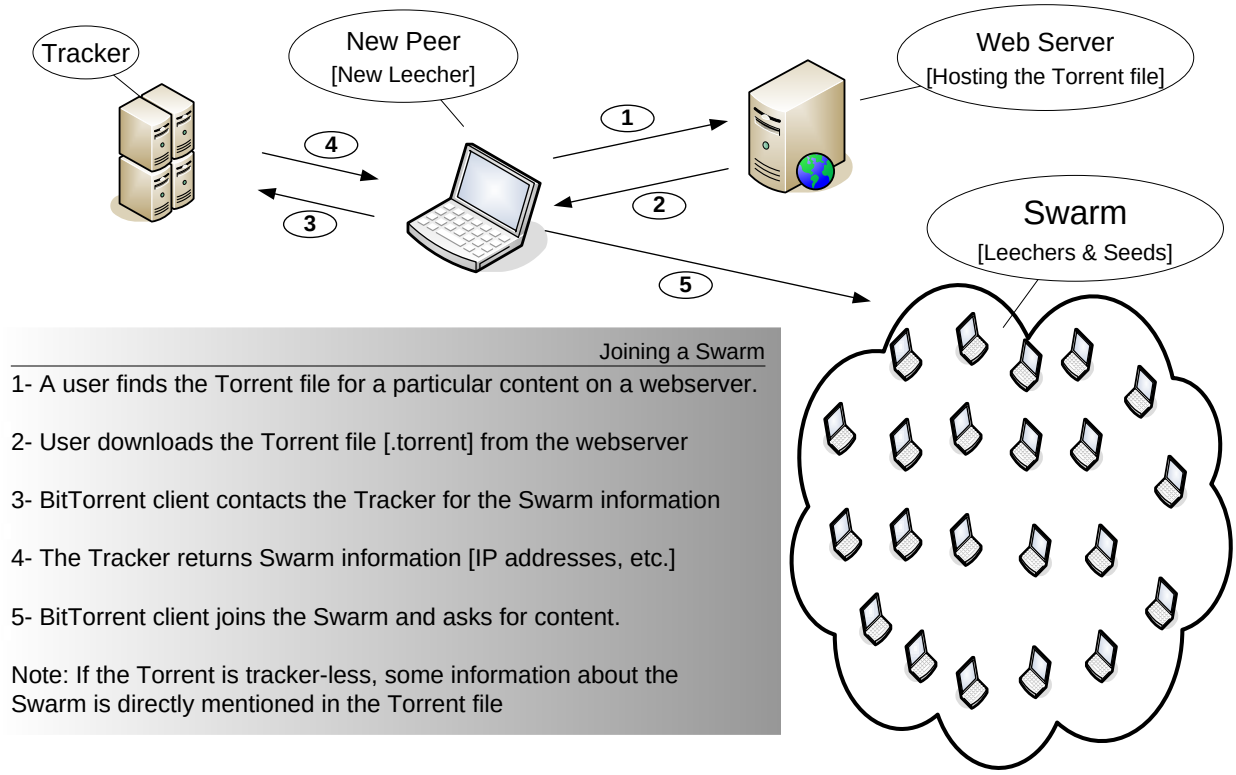

Figure 1: A new peer joining the swarm

size is chosen)[1]. The Piece unit is further divided into what is called "sub-piece" or "chunk" (typically 16 KB) which is nothing more than a byte range within a piece. The content is being requested among peers in a swarm using Piece numbers and chunk numbers, within those Pieces, as identifiers.

2 Mechanics

A new peer (a new Leecher) who wants to acquire the content follows the procedure depicted in Figure 1 and explained in the following. A user first searches (using ordinary web search engines) for the torrent file of a specific content (video clip, music file, etc.) that he wishes to acquire.

[1]What Piece size leads to a better performance is a matter of debate in the research community and, currently, it is chosen arbitrarily. This parameter along with some other BitTorrent parameters is probably not being chosen optimally in the majority of cases.

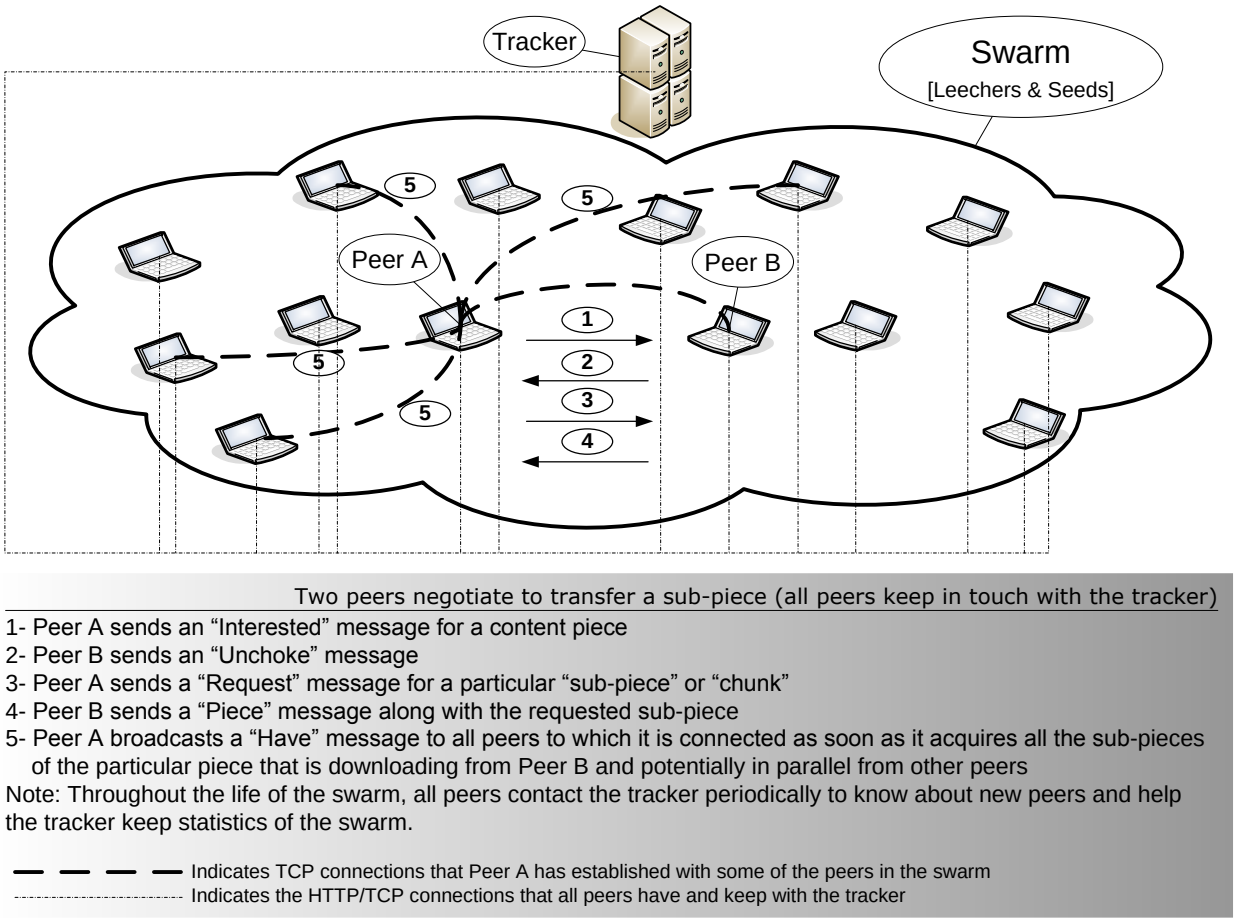

Figure 2: Interactions between two peers and between all peers and the tracker

User then downloads the found torrent file from the webserver which has hosted the file to his personal computer. Afterwards, user opens the downloaded torrent file using the BitTorrent client. The BitTorrent client reads the torrent file and extracts the necessary information including the address of the Tracker. The BitTorrent client then contacts the Tracker for the swarm information, in response to which, the Tracker returns the requested information (peers' IP addresses, etc.). Finally, the BitTorrent client contacts the peers and joins the swarm and asks for the content.

A transfer of content pieces between two peers in a swarm happens when both parties send correct signaling messages; the procedure has been depicted in Figure 2. For a transfer to happen, a requesting peer (Peer A) should send an “Interested” message for a particular content piece. The corresponding serving peer (Peer B) should respond with an “Unchoke” message to the requesting

peer. Thereafter, the requesting peer (Peer A) sends a "Request" message for a particular sub-piece and the serving peer (Peer B) responds with the "Piece" message along with the content itself. A peer (Peer A) may be downloading in parallel various sub-pieces of a particular piece from different peers. In any case, as soon as a peer (Peer A) acquires all the sub-pieces of a particular piece, it broadcasts a "Have" message (for that piece) to all the connected peers.

Signaling between connected peers happens in the same TCP connection that the content transfer occurs and since the TCP connection is bi-directional, a single TCP session is used for the transmission of content/signaling in both directions. Each peer knows exactly which peer has which pieces in its possession, thanks to receiving a "bitfield"(a table of binary numbers with each number indicating whether or not the peer holds the respective piece) during handshake and to receiving "Have" messages from other peers all along its presence in the swarm.

2.1 Protocols: Tracker and Peer

The "Peer-to-Tracker" protocol is a standard HTTP protocol, based on which, the peer sends GET requests and the Tracker issues GET responses. In the GET requests, peers also inform the Tracker about the state in which they currently are, helping the Tracker keep overall statistics about the swarm. All peers in the swarm contact periodically the Tracker using the "Peer-to-Tracker" HTTP connections, as depicted in Figure 2. Contrary to the standard HTTP "Peer-to-Tracker" protocol, the "Peer-to-Peer" protocol is a BitTorrent-specific protocol implemented directly on top of TCP. For each TCP connection with a remote peer (Peer B), a peer (Peer A) maintains a four-tuple state information: 1) Whether or not Peer A is "interested" in any piece of Peer B 2) Whether or not Peer B is "interested" in any piece of Peer A 3) Whether or not Peer A has "choked" Peer B 4) Whether or not Peer B has "choked" peer A.

2.2 Piece Selection

BitTorrent protocol has a relatively sophisticated "Piece Selection" algorithm [3]. "Piece Selection" algorithm is used when the BitTorrent client needs to decide which content piece to ask for and to ask for it from which peer. The algorithm of a new Leecher goes through the following three modes (in the mentioned order) in the process of acquiring the content: "Random First Piece", "Rarest First" and "Endgame Mode". The "Piece Selection" algorithm is in mode "Random First

Piece" when the client starts to download the content. According to this mode, the client only concentrates on getting its first complete piece with which it can start trading pieces with other peers in the swarm. As a result, the client does not take into account the piece availability in the swarm and prefers to choose a piece with a high availability which translates into fast downloads since the sub-pieces can be acquired in parallel from different peers.

After having acquired the first piece, the algorithm switches mode to "Rarest First". In this mode, the client tries to locate the rarest content pieces in the swarm and sends requests to download them. It keeps the pieces with high availability for later. Downloading first the rarest content pieces, hence replicating them in the swarm, helps increase the longevity of the swarm by reducing the probability of losing some content pieces due to some peers (who hold those rare pieces) going offline. As soon as there are only few content pieces left for the client to acquire the whole content, the "Piece Selection" algorithm switches mode again, for the last time, to "Endgame Mode". In this mode, the client broadcasts requests for sub-pieces to the whole swarm, not just to a single remote peer which helps speed up the download completion.

2.3 Peer Selection (Application-level Routing)

The "Peer Selection" algorithm [3] deals with how a BitTorrent client chooses which peer to download from and which peer to upload to. Note that the client has already established TCP connections with some of the peers in the swarm (typically 50 peers; the ones whose information has been sent by the Tracker). TCP connections remain open regardless of whether or not content is being exchanged between both parties at the two ends of the connections. "Peer Selection" algorithm (which accomplishes a sort of "Application-layer Routing") serves the sole purpose of maximizing the download rate for the client. Overall, the algorithm works against the free riders, i.e. the ones which do not upload (or contribute) enough, and ensures a decent download rate for everyone in the swarm. A BitTorrent client allows uploading to a remote peer (unchokes the remote peer) based on the current download rate from that peer. Typically, at any given instant, the client unchokes four connections. Every ten seconds, the choices of which remote peers to unchoke are revisited and the client may make new selections[2].

This method of selecting which peers to upload to presents two problems: 1) There will be no

[2] As the connections are TCP-based, changing the unchoked peers list too often (less than 10 seconds) will not allow the connections to saturate.

way of discovering whether or not other peers (with lower download rates at the moment) could be better partners if the client starts uploading to those peers. 2) New peers (new Leechers) will have no way of joining the swarm, since at first, they have nothing to send so the download rate from them will remain zero and they will never be selected to upload to. To tackle these two issues, the notion of "Optimistic Unchoke" has been introduced in the "Peer Selection" algorithm and it works as follows. Every thirty seconds, the BitTorrent client unchokes one connection (uploads data to) at random, regardless of current download rate from that peer[3]. This ensures that new peers can have a way to join the swarm and the client itself will be able to explore other peers to potentially find better peers to trade pieces with.

2.4 Tracker-less Torrent (Distributed Hash Table)

Apart from BitTorrent protocol operating with the existence of a Tracker, there exists another mode of operation: Tracker-less mode based on Distributed Hash Table (DHT) technology. The DHT implementation in BitTorrent is implemented over UDP and is based on Kademlia [4] which is a "Key, Value" storage and lookup system for peer-to-peer networks. Each BitTorrent peer has a DHT node with a node ID (selected randomly from a 160-bit space), a routing table and a storage for the contact information of the swarms for the Torrents that it knows about. A Torrent's `info_hash` (which uniquely identifies the content) is also a 160-bit value and considered to be in the same address space as node IDs. For choosing where to store a particular Torrent's swarm contact information, an XOR operation (the chosen distance metric) is carried out between node IDs and the `info_hash` of the Torrent. The nodes, with node IDs that are closest to the `info_hash`, will store the contact information of the swarm (the peers who are currently downloading the Torrent's content), with contact information being Node ID, IP address and UDP port number.

Each node's routing table contains more entries about the nodes that are "closer" to them. Note that the measure of "closeness" is the result of the XOR operation between Node IDs. In BitTorrent's implementation, for each i ($0 \leq i < 160$, because of the 160-bit address space), a node's routing table has eight entries for nodes with node IDs of distance between 2^i and 2^{i+1} from its own node ID. This means that a node does an XOR operation between its node ID and all other nodes' IDs, divides the XOR operation results into 160 categories ($[2^0, 2^1]$, $[2^1, 2^2]$, $[2^2, 2^3]$ and so

[3] Thirty seconds is somewhat around the value necessary to allow the upload rate to the selected peer (by "Optimistic Unchoke") to saturate and the other peer starts sending some needed pieces and saturates its transfer.

on) and selects eight nodes per category. The way in which categories are chosen implies that each node has more information about the nodes which are "close" to itself (i.e. smaller XOR operation result) and have less entries about the nodes that are far away. To fill and update their routing tables, nodes use the information contained in the DHT queries originating from new peers wanting to join a swarm as well as the information in other types of DHT queries received from the other nodes in the network [5].

When a new Torrent meta-file is announced by a peer, the announcement travels through the network and gets stored only in nodes with node IDs close to the Torrent's `info_hash`. In the same manner, when a new peer wants to acquire information about the swarm and start downloading the content, it compares the Torrent's `info_hash` with the node IDs which it has in its routing table (and with node IDs that it will learn about later by querying other nodes) and tries to reach the nodes with closest node IDs to the Torrent's `info_hash` (to start, some of close node IDs are already stored in the Torrent's meta-data file). These close nodes will have the contact information of the peers in the corresponding swarm.

References

[1] Bram Cohen. The bittorrent protocol specification, ver. 11031. `http://www.bittorrent.org/beps/bep_0003.html`. Consulted on June 20, 2008.

[2] Bittorrent protocol specification v1.0. `http://wiki.theory.org/BitTorrentSpecification`. Consulted on June 15, 2008.

[3] Bram Cohen. Incentives build robustness in bittorrent. 1st Workshop on Economics of Peer-to-peer Systems, P2PECON'03, 2003.

[4] Peter Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *IPTPS*, 2002.

[5] Andrew Loewenstern. Distributed hash table (dht) protocol. `http://bittorrent.org/beps/bep_0005.html`. Consulted on June 20, 2008.