BitTorrent Simulation

1.0 Introduction:

BitTorrent is a protocol for distributing files. It identifies content by URL and is designed to integrate seamlessly with the web. Its advantage over plain HTTP is that when multiple downloads of the same file happen concurrently, the downloaders upload to each other, making it possible for the file source to support very large numbers of downloaders with only a modest increase in its load.

2.0 System design:

There are two distinct pieces of software – Peer and Tracker. Each peer is both a server and a client. As a server, it offers a file that it wants to share. It registers the file with the tracker. The tracker knows all files to be shared and which peer each file is located at. As a client, a peer may register with the tracker for downloading a file. The tracker knows, for each shared file, which peers want to download the file. These peers form a downloading group. The tracker informs each peer of a downloading group about other group members as well as the file owner. Each peer must establish a TCP connection with the file owner and additional TCP connections or UDP with up to three peers in the same downloading group. During file download, the owner breaks the file into chunks of 100 KB. It will send each chunk to one of the peers in the downloading group. The peers then communicate amongst themselves. Each peer finds out which neighbors have chunks that it does not have, and downloads those chunks from these neighbors. This process repeats until all peers have all chunks, from which the entire file is reconstructed and stored locally.

Main function:

1. Initial peers with its storage capacity and bandwidth
2. Initialize the Target File
3. Initialize map of Peers
4. Distribute Files Among Peers
5. Initialize the tracker
6. Allocate tracker for each peer
7. Initialize torrent files, one per peer
8. Start the peer thread with thread sleep

Tracker:

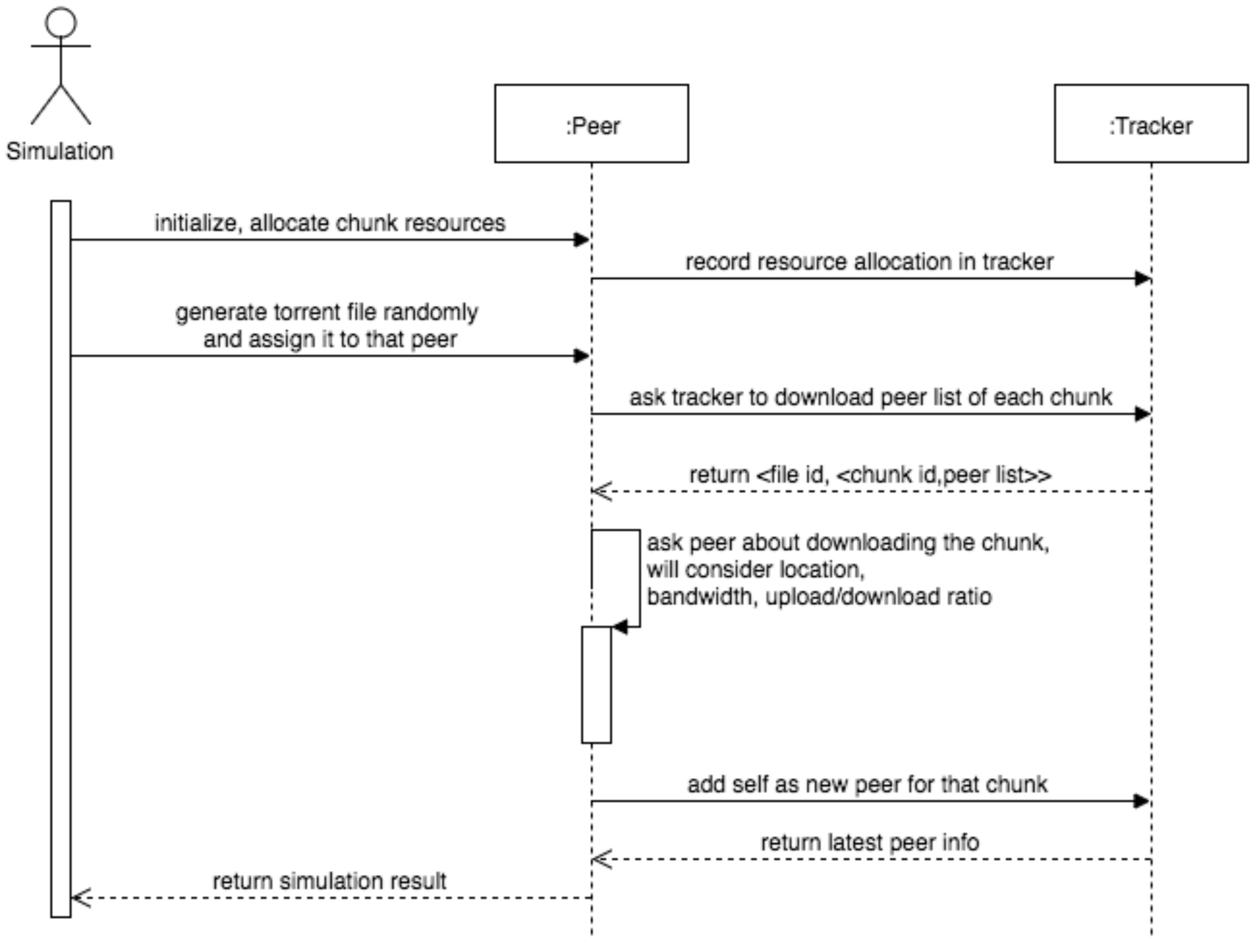
It has the following functionalities:

1. Keep track of the list of online peers, the files they share, and the downloading group for each file.
2. Inform each file owner of its downloading group and inform each peer of other members in the same downloading group. Commands supported:
   a. register-peer : allow a peer to register a shared file with the tracker.
   b. list : allow a peer to query for the list of shared files.
   c. register-group : allow a peer to register with the tracker for downloading a file.

Peer:

Each peer supports two kinds of functionalities: uploading as a server and downloading as a client . “register- peer ”, which will register a shared file with the tracker. Each peer contacts its neighbors to find the chunks available, and it downloads those that it does not have. As a peer downloads, it should print in its control window which chunks it is receiving and from which peers. After it completes download, it should print a summary about the list of chunks it has received and the peers from which the chunks are downloaded.

The following is the sequence diagram of the system.


Simulation
:Peer
:Tracker
initialize, allocate chunk resources
record resource allocation in tracker
generate torrent file randomly and assign it to that peer
ask tracker to download peer list of each chunk
return <file id, <chunk id,peer list>>
ask peer about downloading the chunk, will consider location, bandwidth, upload/download ratio
add self as new peer for that chunk
return latest peer info
return simulation result

sequence diagram

3.0 Creativity point:

1. we use bandwidth and location to pick the peer and choke the free downloader as refuse or agree
2. we use the Handshake to connect two peers
3. Choking:

There are several criteria a good choking algorithm should meet. It should cap the number of simultaneous uploads for good TCP performance. It should avoid choking and unchoking quickly, known as 'fibrillation'. It should reciprocate to peers who let it download. Our deployed choking algorithm avoids fibrillation by only changing who's choked once every ten seconds. It does reciprocation and number of uploads capping by unchoking the four peers which it has the best download rates from and are interested. Peers which have a better upload rate but aren't interested get unchoked and if they become interested the worst uploader gets choked. If a downloader has a complete file, it uses its upload rate rather than its download rate to decide who to unchoke.

4.0 Randomization in simulation

To increase randomization in our simulation, the file size, peer number, peer capacity, peer location, peer bandwidth and chunk size are generated by random number. Also, eventual consistency is utilized in this project and thus the latency is involved in the tracker's info. This is handled already.

Reference:

http://bittorrent.org/beps/bep_0003.html
https://github.com/javierfigueroa/p2p-downloader