Final Project Proposal

Members

Nicole Glabinski (nicolekg)
Everardo Rosales (erosales)
Kimberli Zhong (kimberli)

The Problem

Downloading large files over the internet from a single host poses a few problems. It requires large amounts of bandwidth and resources for the host. If the host crashes or the network fails, the download will fail with no possibility of continuing the download, leading to additional wasted bandwidth.

Plan

BitTorrent has been the defacto solution for distributing large files over the internet as it addresses the increased burden on hosts. It is one of the largest peer to peer systems on the internet today. BitTorrent allows peers to share pieces of the files amongst each other rather than relying on the host to send them all parts of the file.

To distribute a file, the host simply needs to run a tracker and seed the file on one initial peer. The tracker is tasked with letting peers discover each other, and the peers connect directly to each other to find the pieces of the file they need.

Design

In BitTorrent, clients download the file in small pieces from many peers at once, while also sending pieces of files they have to other peers. A server known as a “tracker” keeps track of which clients are currently uploading and downloading a file. Peers connect to the tracker to discover peers.

The first goal of our project is to implement our own version of the BitTorrent protocol[1] in Go. The key components required for this are enumerated below.

Tracker

BitTorrent trackers are responsible for letting peers know where to find fellow peers. A tracker only maintains state for a single torrent.

---

[1] http://bittorrent.org/beps/bep_0003.html

Maintained State
The tracker simply needs to maintain a map of peers with a peer id, their IP address and port. This can be updated whenever a new client connects to the tracker requesting the locations of its peers.

Communication with Peers
When a peer connects to the tracker, the tracker will send it at most 50 known peers at random. This will occur through simple GET requests and responses.

Peers

Each peer runs a BitTorrent client, which can receive and transmit torrent files. The torrent file (described in more detail below) specifies the tracker for the file, which the client connects to in order to find out what peers it can download the file data from. Peers send various messages between each other, such as “have” messages when they successfully download a piece and “request” messages to request a piece of the file. Each file that the client is seeding or downloading will have its own process.

Maintained State
A peer needs to store each piece of the file that it has along with its SHA1 hash on persistent storage. They also need to maintain a bitfield which is a simplified map of the pieces of the file we have or need, 1 for present 0 for missing.

Peers also need to maintain a map of peers they have discovered and communicated with. This map would map a peer to their IP address and bitfield. These bitfields are updated whenever a peer receives a have message from a fellow peer. They are used to quickly identify if that peer has a piece that this peer is interested in.

Communication with Peers
Peers communicate with each other over TCP. They can send heartbeats, “have” messages, “request” messages, as well as data messages.

Communication with Tracker
If a client wishes to download or seed a file, it will need to request the list of peers from the tracker. Additionally, it will need to send periodic heartbeats to ensure that it remains on the tracker’s list of peers.

Metainfo file

The metainfo file or torrent file maintains the location (URL) of the trackers as well as a dictionary of file meta information. This meta information includes the name of the file, the length of the file, and a string *pieces*. The pieces string is a concatenated string of all 20-byte SHA1 hash values, one per piece.

Extensions

Depending on how long that takes us, we hope to add a number of extensions to improve the availability and fault-tolerance of our BitTorrent implementation. Various modifications to BitTorrent have been created to reduce the reliance on the tracker. For example, Peer Exchange (PEX) allows peers that have already connected to continue sharing data without the tracker as well as to learn about new peers that their current peers are connected to.

According to a study on BitTorrent availability[2], including multiple trackers in the torrent file and increasing the availability of the trackers themselves is helpful for increasing availability of the file. If a tracker fails, a peer can simply reconnect to a different tracker in the list. Multiple trackers in a torrent share their peer lists with each other. However, in the case of certain failures or sequences of peer arrival and departure, the swarm of peers may be split into sub-swarms with limited or no connection to each other, which will hurt performance. With the addition of PEX, the swarm is less likely to split into sub-swarms. Additionally, peers which are already connected to at least one other peer can continue downloading the file even if all trackers in the torrent file fail.

[2] https://people.cs.umass.edu/~arun/papers/BT_availability.pdf