Project Milestone 3: Presentation

Benjamin Krueger. Patrick Carra

April 13, 2021

Implementing a Bittorrent-Like filesharing protocol.

CS4730 Networking Semester Project

Original Objective was to implement a Bittorrent v1 protocol client.
Original sub Objective was to investigate a newer hash algorithm to use in place of sha1.
Due to time restratints, only the first goal was realized.

Bittorrent v1 Specification

Bittorrent uses the following entities:

- A web server for proxying tracker web traffic and hosting metadata files
- A Bittorrent "tracker" that keeps track of clients
- A client that can act as the original downloader/seeder
- A "swarm" of downloaders/uploaders that act as "peers" for hosting a file

Bittorrent v1 Specification (cont.)

File Hosting Protocol:

Generally files are "hosted" on a torrent via the following proceedure"

1. Start tracker process
2. Start web server process
3. Use torrent client to generate a metainfo file known as a .torrent to be served from the url of the tracker
4. Link the metainfo file to a webpage where people can download it.
5. Start a torrent client on the host that already has the original file

Bittorrent v1 Protocol (cont.)

Bittorrent Downloading procedure

- Download .torrent file
- Open .torrent with client
- Select download location
- Torrent downloads
- Client will usually "seed" or continue uploading to peers in the swarm until the process exits.

What is a .torrent file?

.torrent files are Metadata that keep track of:

- Annouce: the http url of the tracker keeping track of a list of peers
- Info: a dictionary with the following keys
- Name: name of torrent
- piece length: number of bytes per file "piece" in tracker
- pieces: a string that is the sha1 hash of every piece concatenated together.

How is a .torrent file encoded?

Torrent files are bencoded

- B-encoding is a data serialization format for space efficient encoding of datatypes such as strings, lists and dictionaries.
- Bencode is very similar to json or xml
- Bencode is more efficient than other serialization formats at the expense of being not human readable.
- Do not try to edit bencode with a text editor
- Torrent files are just bencoded dictionaries with information about the tracker and file to be downloaded.

How does the client protocol work?

To download from a swarm with a .torrent file a bt client does the following:

1. The client opens the file and decodes the torrent file from bencode.
2. The client will contact the tracker with http GET request for the infohash, or the sha1 of the info dict in the .torrent.
3. The tracker will respond with a binary encoded list of peers that are in that swarm
4. The client will begin downloading pieces from the swarm per the Bittorrent sharing algorithm (more on that later)

Bittorrent Peer2Peer sharing algorithm

After contacting a tracker and recieving a peer list, the Bittorrent client will follow the following protocol:

- Clients start tcp connections with peers and start an initial handshake
- Clients keep track of 2 states for each connected peer:
- Choked: or whether or not the remote peer has choked the client
- interested: wheter or not the remote peer is interested in something the client has.

P2P protocol: TCP handshake

The tcp handshake between 2 clients has the following features

- handshake: pstrln pstr reserved infohash peerid
- pstrlen :length of pstr
- pstr: protocol identifier
- reserved: 8 byte bit field that can manipulate behavior of protocol
- infohash: infohash from torrent
- peerid: 20 byte identifier for the client

P2P protocol: udp messages

After handshaking, the clients will exchange udp messages of the following format

- length: lendgh of message
- messageid: single byte for message type
- payload: actual payload of information for the message

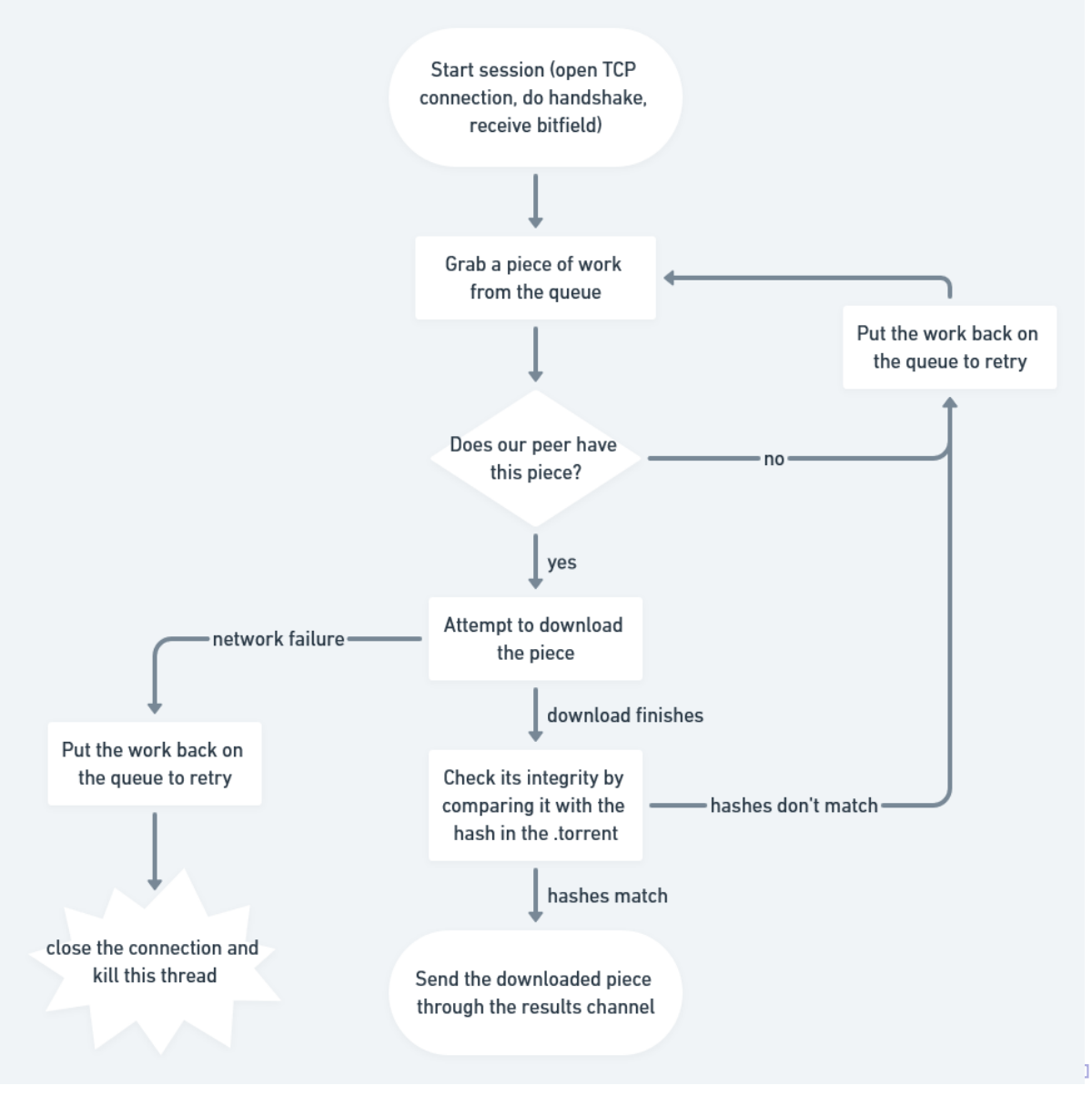
P2P protocol: message types

The UDP messages have the following functions

- keep-alive: keep alive the connection between peers, specified with a prefix length of 0
- choke: client is choked
- unchoke: client is unchoked
- interested: client is interested in something the other client has
- not interested: not interested
- have: client has the piece at index ¡piece index¿
- request: client requests a block of pieces starting at some index
- piece: sends the piece as a payload
- cancel: cancel a block request

p2p sharing algorithms

Here is a flowchart for the general algorithm bt utilizes for p2p sharing.



Our Implementation

How did we decide to go about implementing a p2p protocol?

- Due to time constraints, a full BT client implementation was not acheivable
- Instead, a far simpler to implement but functionally similar protocol was devised
- This sharing program is written in go

Implementation

Core components of our protocol

- Centralized metadata "tracker" tracking peers with file
- Files distributed via message passing algorithm
- HTTP used for client to tracker communication
- Metadata serialized via JSON instead of Bencode (better library support)
- Stripped down messages and passing algorithm
- Only supports one file at a time

Thank you for watching!