CS 136 Pset 2

Peer-to-Peer Simulation

September 28, 2018

Problem 2 a).

- In any round, if there are fewer than 3 peers who request pieces from us and allocated bandwidth to us in the previous round, we distribute all our bandwidth evenly among the requesters and the peer we randomly select via optimistic unchoking.
- We have a maximum of 4 slots, but we do not necessarily use all 4 in each round. We do however, use allocate all our bandwidth in each round.
- We shuffle the peers list and requests list to increase randomness and break symmetry.
- We make our client "nearly-memoryless" in that we only care about the actions of the previous round and not the actions of any round prior to that.

Problem 2 b).

- Our tournament client is based off of the BitTyrant client described in the textbook, since it provides one of the best ways to accurately estimate the amount of bandwidth needed to dictate reciprocation and the amount of reciprocation we can expect to gain.
- We increase α and decrease γ from the textbook's default values so that we tend towards overestimating the amount we need to upload to each peer. This makes sure that we are more likley to get reciprocation each time we upload.
- We also take any unused bandwith and use it for optimistic bandwith. Even though this is a very small fraction of our total bandwith, it allows us to counter other BitTyrant strategies by becoming uploaders without uploading much to them and without utilizing a significant portion of our own bandwith.
- We also decrease the number of rounds it takes for our client to decrease the estimated upload needed for reciprocation. This, in combination with our lower value for γ, allows us to create a more continuous and smooth set of values for estimated upload required, which is ideally a better estimate for the actual values of upload bandwith required. We decrease the number of rounds required from three, per the textbook model, to only two rounds.

Problem 2 c).

- BitTyrant does slightly worse than the standard BitTorrent clients in a population of 10 BitTorrent standard clients and 1 BitTyrant clients. On average, it downloads about 1 to 2 rounds slower than the standard client. Based on the code below, the BitTyrant client finishes in, on average, 29 rounds:

```
python sim.py --iters=5 --loglevel=info --num-pieces=32
    --blocks-per-piece=32 --min-bw=32 --max-bw=64 --max-round=1000
    Seed,2 ArmlB1Std,10 ArmlB1Tyrant,1
```

- Tourney does slightly better than the standard BitTorrent client in a population of 10 BitTorrent standard clients and 1 Tourney client. On average, it is about 1 round faster than the standard client. Based on the code below, the Tourney client finishes in, on average, 26.6 rounds:

```
python sim.py --iters=5 --loglevel=info --num-pieces=32
    --blocks-per-piece=32 --min-bw=32 --max-bw=64 --max-round=1000
    Seed,2 ArmlB1Std,10 ArmlB1Tourney,1
```

- PropShare also does very slightly better than the standard BitTorrent client in a population of 10 BitTorrent standard clients and 1 PropShare client. However, it is on average only about half a round faster than the BitTorrent standard client. Based on the code below, the PropShare client finishes in, on average, 27.8 rounds.

```
python sim.py --iters=5 --loglevel=info --num-pieces=32
    --blocks-per-piece=32 --min-bw=32 --max-bw=64 --max-round=1000
    Seed,2 ArmlB1Std,10 ArmlB1PropShare,1
```

Problem 2 d).

- A population of 10 BitTyrant clients does significantly worse than a population of 10 Standard clients, finishing in about 2 rounds more on average. Ten BitTyrant clients finishes in about 30 rounds, according to the code below:

```
python sim.py --iters=5 --loglevel=info --num-pieces=32
    --blocks-per-piece=32 --min-bw=32 --max-bw=64 --max-round=1000
    Seed,2 ArmlB1Tyrant,10
```

  Meanwhile, a population of 10 Tourney clients does even worse, finishing in nearly 3 rounds more than a population of standard BitTorrent clients. Ten Tourney clients finishes in about 31 round on average, according to the code below:

```
python sim.py --iters=5 --loglevel=info --num-pieces=32
    --blocks-per-piece=32 --min-bw=32 --max-bw=64 --max-round=1000
    Seed,2 ArmlB1Tourney,10
```

- A population of 10 PropShare clients does similar to a population of 10 standard BitTorrent clients. On average, they finish in less than 1 round of each other. Ten PropShare clients finishes in about 28 rounds on average, according to the code below:

```
python sim.py --iters=5 --loglevel=info --num-pieces=32
    --blocks-per-piece=32 --min-bw=32 --max-bw=64 --max-round=1000
    Seed,2 ArmlB1PropShare,10
```

Problem 2 e).

Firstly, implementing the actual clients gave us greater insight into how each of the clients work and some of the advantages and disadvantages of each client. For example, we realized that some small factors like the initialization of the BitTyrant estimated data and the behavior of the BitTorrent standard model depends on what it chooses to do if less than n peers upload to it.

Additionally, we realized that there is no sure-fire way of designing a client that works better than the BitTorrent standard client or the BitTyrant client. Although it is possible to conjecture what factors will cause a client to be better, it is very difficult to actually design a good client without sufficient testing of parameters and theories. For example, a client that does well in a population of different clients may perform significantly worse in another population of similarly-designed clients.

Peer-to-peer systems and other game-theoretic scenarios often come with tradeoffs. For example, a population of BitTyrant clients who look to selfishly optimize their own download times may cause delays for the entire group as a whole in obtaining all the pieces, similar to a *tragedy of the commons* situation. It's important to consider the specific goals of each client when designing these systems and deciding between different tradeoffs.

Additionally, a concern that is not mentioned in the textbook is the actual runtime of the algorithms. Because our Tourney client is more complicated than the prescribed clients (Standard, Tyrant, and PropShare), it takes longer to run in the simulation. Although the difference is barely noticeable in our small-scale simulation, more complicated clients can have large repurcussions in actual Peer to Peer applications.

This argument is a perfect segue into our last insight into the design of efficient clients. Designing the clients in a certain way can have significant impact on the actual implementation of these clients. While we seek to design clients that will theoretically perform the best, we must also take into account how simple the implementation of these clients can be since these simplifications can have significant impact on the clients' performance.

Problem 3 a).

- There are more than two players in the BitTorrent game, and so players do not always play against the same opponent. On average, every pair of players interacts with each other at most a few times over the span of the game, depending on the bandwith of uploading, the number of peers and the size of files.
- There are more than two actions for each peer. Peers can vary the bandwidth they provide to others, and only need to interact if one peer has a file that the other peer wants.
- Additionally, peers also have many different additional strategic options. These include strategic piece revelation, strategic piece requesting, and only sharing the most common pieces with peers in the swarm.
- Peers can decide to leave at any time if they already have the pieces they need or if none of the other peers have a block that is desired by the agent.

Problem 3 b).

- The payoffs in the BitTorrent game vary depending on the bandwidth provided by the uploader, whereas the payoff when both agents cooperate in the Prisoner's Dilemma are fixed.
- Optimistic unchoking in the BitTorrent client randomly rewards those who did not cooperate, while TfT punishes those who don't cooperate. Thus, strategies like BitThief are successful where clients don't necessarily have to upload anything in to gain utility.
- If more than three people upload to you, you only choose the top 3 agents to upload to. Essentially, cooperation in the last round does not guarantee cooperation in the current round. Thus, choosing the amounto of bandwith to devote to each peer is important in strategically achieving one of the top 3 spots for each peer an agent uploads to.

Problem 3 c).

- There might exist another client that is strictly dominant over the BitTorrent client, in which case you would choose to use the other client.
- Your preferences might not align with the strategy of the BitTorrent client, which is to minimize the time it takes to download all the pieces. For example, even though BitThief takes on average 2-4 times longer than the reference client, it can be a better option for users who prefer not to upload.