K254 Group Members: Tara Balan, Brian Kazinduka
February 3, 2025

Problem Set 2

Analysis

A. For the standard client explain what assumptions or decisions you had to make beyond those specified in Chapter 5. (For example, details about unblocking that are not specified in Chapter 5, and how to handle possible corner cases.)

As specified in chapter 5, the client should use a rarest-first algorithm to decide which pieces to request. Right now, the client has a random approach to requests. Specifically, it requests a piece held by the fewest peers in its neighborhood from the set of pieces that it needs and are held by at least one peer. This prioritizes pieces that are least available by reducing their bottleneck and therefore risk that they will disappear. Additionally, the reference client should use a Tit-for-Tat upload strategy (ie. if you upload, I'll upload) as well as optimistic unblocking, which unblocks a randomly selected interested peer every three rounds. To do so, we check the top users and sort by who is contributing the most, and match them with one another to ensure that users are mutually benefitting.

In order to implement this, we must make some additional assumptions outside of those listed in chapter 2. We assume that the network has an ample amount of users logged on at each time to share files with one another. In terms of the file sharing process, we also assume that all clients follow the standard handshake process, in which they send a message that includes the peer id and the information hash which can verify the content of the file, and receives a handshake response before sending out another message to their peers. For this to work, we have to trust that the hash tracking is reliable, else we would be unable to effectively match peers with one another. To implement the tit-for-tat strategy and subsequently optimistic unblocking, we have to assume that users do want to benefit from one another and are willing to comply with this tit-for-tat strategy.

The error checking and comments built into the pseudocode also provided insight into some of the assumptions we will have to make. One of those insights was that we should focus on symmetry breaking, which is what to do when clients swarm around the same pieces. The rarest-first method helps mitigate some of these conflicts, as removing some elements of randomization helps to prioritize some users over others in a fair manner when cases like this do arise.

B. Write a concise summary of the strategies you used for the tournament client, and why you chose them.

For the tournament client, we implemented a similar strategy to that of the reference client but with a few tweaks. We still used a rarest first prioritization for downloading pieces, but we also wanted to prioritize completing pieces that are partially downloaded, first finishing pieces with fewer blocks left and then moving to ones with a greater number of remaining blocks. Doing so ensures that the speed of the software does not slow down and incentivises peers to continue file sharing. It also ensures that partially downloaded files do not get lost in the downloading cycle and therefore dropped from the platform altogether. It also improves conflicts that arise from clients downloading the same pieces multiple times, or trying to download the same rare pieces. These improvements are done via symmetry breaking, which delegates pieces to peers by sorting them by their peer IDs, and introducing randomization into the clients such that the top peers are no longer overloading at the peers with the smallest IDs are prioritized.

The tournament client also improves the optimistic unblocking methodology. Initially we were not specifying an amount or upload bandwidth for the randomly selected peer, not doing so could give them too much power if they are an invaluable peer. Allocating about a 7% bandwidth gives them a bit of power in case they are a valuable contributor, but still ensure that the rest of the uploading bandwidth (the other 93%) is distributed among other clients proportionally.

Building off of that proportional bandwidth allocation, our tournament competitor normalizes the bandwidth allocation to ensure that all of the available bandwidth is used. Additionally, we strive for fairness to encourage peer retention by allocating a large amount of the remaining bandwidth to the top contributors. This pulled from the method used for the PropShare client and ensures client fairness and encourages peers who are contributing the most to continue contributing frequently.

C. Outperforming the standard client

The BitTyrant client is able to upload more blocks on average than the standard clients. It also takes a greater number of rounds to complete the uploads. However, this should not be the behavior of the Tyrant--it should take fewer rounds as it is designed to prioritize the fastest peers. The Tourney client uploaded far fewer blocks on average than the standard client in a comparable number of rounds. This also is not expected behavior, as the prioritization of partially completed blocks and rarest blocks should have resulted in more completed blocks. The PropShare client uploaded on average roughly the same amount of blocks as the standard client. As the population of standard clients grew, the amount of completed rounds grew proportionally to one another. This may be indicative that the bandwidth allocation should be increased.

D. Overall performance of populations

A population of only BitTyrant clients on average uploads far fewer blocks than either the PropShare or the Tourney client, and takes many more rounds. The PropShare and Tourney clients were comparable in that they took roughly the same amount of rounds to upload a similar amount of files. The BitTyrant client may be taking longer because it spends a greater amount of

time searching for the top contributors. The Tourney client may be taking longer than the Propshare client because it is also focusing on finding partially downloaded pieces in addition to rare pieces. While this will eventually reduce a bottle neck and swarm around pieces, making the program overall faster, perhaps when only a small number of rounds are occurring the effects are not heavily felt.

E. Write a paragraph about what you learned from these exercises about BitTorrent, game theory, and programming strategic clients? (We aren't looking for any particular answers here, but are looking for evidence of real reflection.)

These exercises emphasized the strategy involved with BitTorrent and file sharing platforms and now to factor fairness into these algorithms. While no method will truly be fair, the symmetry breaking we implemented as well as the upload bandwidth allocations in propshare makes sure that those who help the most also benefit the most. Because of this, BitTorrent is a prime example of a tit-for-tat prisoner's dilemma. Similar to what we saw with the prisoner's dilemma--the Nash Equilibrium is for both to defect but both will actually benefit more if they cooperate--the socially optimal choice here would be for both users to upload, but in the Nash Equilibrium both players would choose not to upload. In addition, to create our tournament client, we simply combined and slightly modified aspects of the Tyrant and PropShare clients to make one client that optimizes both requests and uploads in a fair and efficient manner such that peers will choose to stay in the network. Doing so allowed us to understand a bit of the logic behind how some users may choose to game the system; they may choose to modify the bandwidth allocations via a propshare client so they are still able to download highly sought after pieces more easily, or even change the optimistic unblocking methods to give randomly selected players more power if their goal is to increase the number of diverse users in the network. Implementing the Tyrant also gave insights into how users may try to game the system by abusing their uploading power (ie. uploading the same thing repeatedly) to gain greater downloading power.

Theory Questions

A. State three ways in which the peer-to-peer file sharing game of the BitTorrent network is different from a repeated Prisoner's dilemma.

One difference we can notice is the optimistic unlocking mechanism of BitTorrent which allows for peer exploration. A second difference is that in BitTorrent the client is "constantly trying to download from every peer in its neighborhood," as opposed to the fixed opponents in PD (i.e. P1, P2). Lastly is that as opposed to the single-decision model of the PD, actions like sharing and free-riding occur continuously and independently in BitTorrent.

B. State three ways in which the BitTorrent reference client is different from the tit-for-tat strategy in a repeated Prisoner's Dilemma.

One difference we can see is that uploads are based on contribution and not just retaliation, as the reference client prioritizes the top three peers who provide the most upload capacity. A second difference is that unlike Tit-for-Tat (TfT), which strictly mirrors an opponent's past actions, BitTorrent also includes an optimistic unblocking slot to periodically unblock a random peer. Lastly, defection in TfT is met with immediate retaliation, whereas BitTorrent gradually deprioritizes uncooperative peers rather than instantly blocking them, giving them a chance to resume cooperation.

C. Explain two reasons why just having a BitTorrent client that is a best response to itself is insufficient for this client to form an equilibrium in a peer-to-peer system. (Hint: you can think about both theoretical reasons and practical reasons. For example, what could happen if there is another client that is also a best-response to itself? What if different users care about different objectives?)

In a peer-to-peer system, a BitTorrent client being a best response to itself is insufficient for equilibrium because other clients may adopt different strategies, disrupting cooperation. The reading states that the reference client follows a tit-for-tat (TfT) strategy, but TfT is "not a subgame-perfect Nash equilibrium," meaning deviations can occur. If we imagine a situation where another client defects by free-riding—where it "only tries to download (Defect)"—cooperation will be stifled, leading to worse outcomes overall. Equilibrium also requires stability across interactions, but BitTorrent's optimistic unblocking mechanism means peers occasionally cooperate without immediate reciprocation, allowing room for strategic exploitation.