2. Analysis: Provide your answers based on test runs with the clients you programmed. When asked for comparative performance results, provide some evidence (e.g., simple statistics) for the results you are reporting/ describing.

(a) For the standard client explain what assumptions or decisions you had to make beyond those specified in Chapter 5.

Assumption:

1. We evenly split the bandwidth in to 4 unchoking slots: 3 are taken by top 3 peers that from whom the user has highest download rate; the rest 1 is optimistic unchoking slot that belongs to the peer who is randomly selected by the system.
2. If first 3 slots are not occupied, i.e., the system could not find enough peers that the agent has downloaded blocks from, we make the system to randomly select peers in the neighborhood to occupy the empty slot.

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

(c) Outperforming the standard client:

i. How does the BitTyrant client do in a population of standard clients?

ii. How does the Tourney client do in a population of standard clients?

iii. How does the PropShare client do in a population of standard clients?

Look at the relative ranking of the clients and the percentage improvement (or impairment) in the number of rounds it takes the client to get the complete file.

(d) Overall performance of populations:

i. How does a population of only BitTyrant clients perform? What about a population of only Tourney clients?

ii. How does a population of only PropShare clients do?

Look at the time it takes to get the file out to all clients (i.e., when does the last client complete downloading the whole file), as well as the average download time for the individual clients.

(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.)

3. Theory:

(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.

1) BitTorrent network has more than 2 players (peers).

2) BitTorrent peers have more than just two actions. They can decide how much upload bandwidth to make available to each peer. In Prisoner’s dilemma, the decisions are only sharing (cooperate) and not sharing (defect).

3) In BitTorrent, one player (peer) might quit the game after they finish downloading while other peers are still in the game downloading. Also one peer’s payoff (any bandwidth that received) depends on both the seeder and the actions of other peers in that seeder’s neighborhood. While in Prisoner’s dilemma, player’s payoff only depends on the other player’s action.

(b) State three ways in which the BitTorrent reference client is different from the tit-for-tat strategy in a repeated Prisoner’s Dilemma.

1) Besides rewarding a certain number of peers that give most download to the user, that user would also give an additional optimistic unchoking slot to a random peer from its neighborhood and let that random selected peer to download. TfT strategy does not reward random actions.

2) In tit-for-tat, as long as both players adopt TfT and both cooperate, they would get the best payoff. As for BitTorrent reference client, cooperation (uploading that piece) does not promise the agent to be selected because they are also competing with other agents in the neighborhood. Defect (not sharing) does not necessarily mean that that agent has no hope to get any payoff (get sharing) because they could be randomly selected to the optimistic unchoking slot.

3) Also cooperation’s payoff in TfT depends on how much upload bandwidth the seeder is willing to make available to each cooperator. In TfT, cooperation payoff is usually fixed and does not depend on any player.

(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.

1) First, that client