Design Issues of the prTorrent File Sharing Protocol

Suman Deb Roy †Tyler Knierim ¤Shanieka Churchman Wenjun Zeng

Department of Computer Science
University of Missouri-Columbia
† Truman State University
¤ Lincoln University

***Abstract*-The absence of piece rarity in the BitTorrent Unchoking Algorithm exposes it to various exploitations and hinders optimized performances. prTorrent is a file sharing protocol based on BitTorrent which considers rarity of pieces swapped during unchoking for optimized performance within the swarm. This paper discusses the philosophies, issues and structure behind development of the prTorrent p2p protocol from scratch. The protocol was developed in Java to promote portability in simulation as well as emulation. We hope this work is a guide to students and developers planning to build a p2p simulator from the very base and alleviate design differences between simulation and emulation.**

I. Introduction

prTorrent [1] is a file sharing protocol based on BitTorrent that uses piece rarity (PR) during unchoking in selecting the best peers to trade pieces with. PR is dynamic and, unlike availability, its value changes with the current swarm conditions. This provides improved performance by reducing exploitations and promoting a Nash Equilibrium within the swarm. However, it also requires increased circumspection in developing such a protocol.

Motivation

This work describes issues and structure in building a p2p simulator and emulator using a single platform like Java. Most simulators in the market are built upon various layered platforms [4], which inhibit portability and make the design cumbersome. Moreover, their design makes actual emulation quite difficult to implement. In the next section, we discuss critical ideas in developing a p2p protocol from scratch. The simulator allows testing of Starvation and flash crowd scenarios [2] in addition to normal performance measurements.

II. The Design

A. Basic Framework

prTorrent divides a p2p network into three domains of influence: the piece, the peer and the swarm. For most p2p systems the basic framework should consist of building the piece, peer and connection entity classes (Fig. 1). We can then build a main module to run all of the required entities for the simulation. Global Availability measures the availability of a particular piece across other swarms and is considered a random number, since the simulator simulates a single swarm. We embrace a bottom-up development approach.

B. Piece Class

We begin with the smallest domain of influence – the piece class. Every piece has a piece number and availability attached

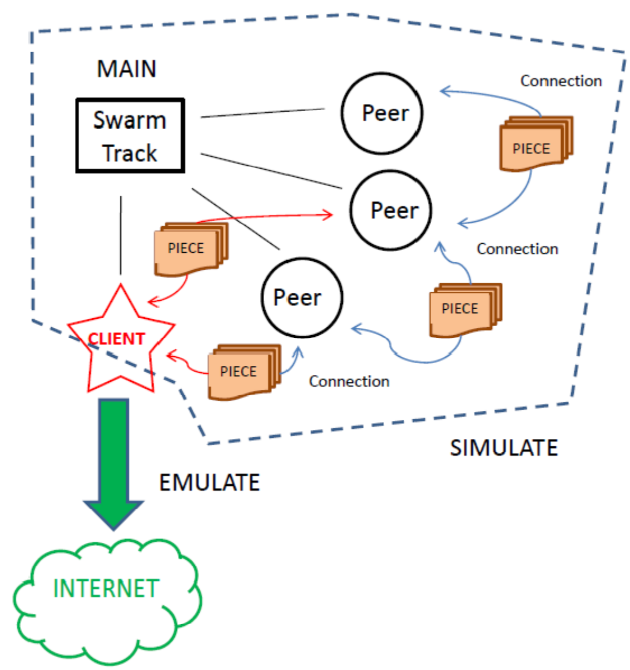

Figure 1. The basic framework

to it. Since availability is just one of the parameters in realizing PR, note that two instances of the same piece have the same availability, but might have different PRs. This is fundamental to piece swapping in prTorrent.

C. Peer Class

The typical data members for the peer class includes the amount of pieces the peer has, their download and upload speeds, neighbor information and arrays for the set of pieces and neighboring peers in the swarm. Completion Factor (CF) judges how far a peer has proceeded towards its goal of downloading all the pieces. CF ranges from 0-1. A very high or very low value of CF means the peer becomes less choosy during unchoking. Each peer has an array to store the upload speed from their neighbors as well as the PR of the last piece downloaded.

When deciding to download a piece, the peer should first check with its neighboring peers on the piece's availability. If none of the neighbors have the piece, it should sort the other peers in its list by their upload speeds. If however the piece is available with neighbors, then the sorting should be done based first on the PR of uploaded pieces in the previous round. In either case, the formula (3) used for unchoking in [1] should be used.

D. *Connection Class*

The connection class is perhaps the most different when emulating than for the simulator. In either case however, a point to point connection must be established between the two peer clients. In the simulator this can be realized by a simple reading from and writing to socket code. It is required to generate a TCP connection handler that will manage sending and receiving of the pieces. There should also be a data buffer taking care of write/read lengths which can be detached as applicable. The connection handler can be easily implemented by treating one node as the server and another as the client. Hence, all we need is a parent (belonging to class TCP Server), a socket connection and a listener. Following this, modules should be written to handle the received messages, reading data from the connection and handling the message queue. The message queue can be written by using a Runnable writer [3] and a separate *sendmessage* function.

E. *Client Class*

The *client* is the prTorrent peer that we focus on. We perform the client's unchoking using prTorrent and test its performance among a swarm of BitTorrent peers. This is different from [1] where we simulated all peers following prTorrent. This design feature enables our protocol to be easily emulated in a true network scenario. This will also facilitate testing the effects of an exploitation environment or starvation on prTorrent peers. The upload bandwidth of the client is taken as an input parameter before the simulation begins.

F. *Main Class*

The main class simulates the swarm domain of influence. As such it needs to decide the piece size and then divide the input file size into necessary number of pieces. The main class also performs the task of the tracker for the simulator purposes. However, for the emulator, we would need a separate class to interact with the tracker of the network. The main class should contain a vector of peers present in the swarm and the pieces that each of them hold. However, this is where our design differs from most other simulators. We actually run two layered simulations- one for all the peers except the prTorrent *client* performing unchoking among themselves using various other BitTorrent clients and one for the interaction of the prTorrent *client* with these other peers. Therefore, for emulation, we just need to remove the first simulation layer and our protocol will be ready to connect to other peers on the Internet.

III. Theoretical Analysis

We are testing the simulator in situations similar to our original tests for prTorrent. In terms of the Discount Parameter (DP) Exploitation referred in [1], we found that we can mathematically predict the minimum numbers of peers required to remain fair in order to maintain a DP exploitation scenario within the swarm. The minimal value can be reached by realizing that uploading in p2p is a collective action. This means that aims of the whole swarm together are best served if every peer uploads, but it is not in the best private interest of the individual peers. Hence, if the collective payoff for each peer from uploading is p(n) and the payoff of a DP exploiter is s(n), then the minimal 'n' peers required to avoid DP exploitation is obtained when:

$$p(n+1) = s(n)$$

The Nash Equilibrium for fairness will be maintained as long as p(n+1) is greater than s(n). The mathematical formulation of this collective payoff in p2p is still an open challenge.

Let us consider the DP (δ) = 1/ (1+ r), where r is rate of decrease of the piece value with each successive transfer. However, with DP exploitation, there is a probability 'p' attached to whether further interaction will occur or whether the exploiter will defect. Hence, $p\,\delta = 1/(1 + R)$, where R is the effective rate of return on a future payoff. This value of 'p' is judged by the PR of the requested piece. It clearly shows that if 'p' is less, cooperative behavior maintained by BitTorrent Tit-for-Tat will disintegrate since:

$$R = \frac{1 - p\,\delta}{p\,\delta}$$

IV. Conclusion

The prTorrent p2p simulator was built with the perspective of easing the transfer from simulators to emulators. We hope this will provide guidance to new developers on the structured development of a p2p network. It also contains theoretical insights that prTorrent can estimate a Nash Equilibrium that mitigates exploitation and promotes optimized performances.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. 0649158.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] S. Deb Roy and W. Zeng, "prTorrent: On Establishment of Piece Rarity in the BitTorrent Unchoking Algorithm", *In IEEE P2P'09*, Sept 2009.

[2] K. Eger, T. Hobfield, A. Binzenhofer and G, Kunzmann, "Efficient simulation of large-scale p2p networks: packet-level vs flow level simulations", *In Proc. Of 2nd Workshop on Use of P2P, GRID and agents for development of content networks.*

[3] http://java.sun.com/j2se/1.4.2/docs/api/java/lang/Runnable.html

[4] S. Naicken, B. Livingston, A. Basu, S. Rodhetbhai, I. Wakeman and D. Chalmers, "The state of peer-to-peer simulators and simulations", *Computer Communication Review, 37(2):95-98, 2007.*