19th Annual IEEE International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems

EbitSim: An Enhanced BitTorrent Simulation Using OMNeT++ 4

Pedro Evangelista, Marcelo Amaral,
Charles Miers, Walter Goya, Marcos Simplicio, Tereza Carvalho
Escola Politécnica, University of São Paulo, São Paulo, SP, Brazil
{pevangelista,mamaral,cmiers,wgoya,mjunior,carvalho}@larc.usp.br

Victor Souza
Ericsson Research, Packet Technologies
CDNs and Cloud Computing, Kista, Sweden
victor.souza@ericsson.com

***Abstract*—The BitTorrent protocol is one of the most successful P2P applications, being largely studied by the research community. Nevertheless, studying the dynamics of a large BitTorrent network presents several challenges, such as difficulty in acquiring network traces or building measurement experiments. Evaluation through simulation is usually utilized for studying BitTorrent networks, yet only a few BitTorrent simulation models are available for the research community. In this article, we present an extensible framework for developing BitTorrent simulations, focusing on realism and without losing on scalability. We developed an accurate version of the protocol by analyzing the source code of mainstream BitTorrent clients and by discussing directly with their developers. The simulation model was developed with the OMNeT++ Framework, inheriting its high extensibility, and with the INET Framework, for accuracy of the underlying network model. We also took into account the effects of multitasking in our model, since BitTorrent applications acquires content from several sources simultaneously, and utilized real world traces for modeling the processing times. We present an analysis of our simulator regarding performance aspects and BitTorrent-related results.**

I. Introduction

The popularity and success of the BitTorrent protocol resulted in a continuous growth of BitTorrent traffic on the Internet. Studies performed by Ipoque, an European provider of deep packet inspection (DPI) solutions, shows that P2P traffic accounts for the majority of the Internet traffic, and that BitTorrent is by far the most popular P2P protocol [1]. Consequently, the study of the BitTorrent protocol has become an important matter within the research community.

There are many studies focusing on simulation models with varying levels of abstraction for evaluating BitTorrent: from detailed ones, which take into account the underlying network structure [2], to high level models that completely abstract the network [3]. The main advantage of simulation over other methods is that it enables the study of a large number of elements at low cost, allowing a deep analysis of the network's behavior when some parameter changes. However, despite the abundance of works about BitTorrent, there are few tools available for researchers to perform simulation-based studies. In order to tackle this issue, in this work we develop a BitTorrent simulation with the popular OMNeT++ Framework framework.

Our focus is on building a detailed and realistic model that can still simulate a high number of network elements. In order to achieve this goal, and knowing that the official specification of BitTorrent [4] does not present a clear definition of the dynamics of the its protocol [5], we studied modern BitTorrent clients such as Azureus and also discussed directly with their developers. Aiming to obtain realistic results, we also utilized the INET Framework for modeling some protocols of link, network and transport layers in detail. In this paper, we present experimental results concerning the performance of the simulation and results regarding the BitTorrent protocol.

The rest of this document is organized as follows. Section II gives a brief description of BitTorrent's engine. The model implementation is detailed in section III. In section IV, the scenario utilized and the results obtained from the experiments are presented. Section V discusses the related work. Finally, section VI provides our final considerations and presents some possibilities for future work.

II. BitTorrent and problem definition

BitTorrent is a swarming P2P protocol designed for the distribution of content between a large number of peers. Aiming to keep a fair usage of resources, the protocol relies on the tit-for-tat mechanism, which consists basically in providing higher download rates to altruistic peers, while penalizing egoistic behavior [6]. The basic entities that compose a BitTorrent network are Torrent Repository Server, Trackers and Peers, as shown in Figure 1. If a peer has the whole content, it is called Seeder; otherwise, it is called a Leecher. In order to facilitate the explanations, the local peer will be called Client and the peers connected to it will be called simply Peer.

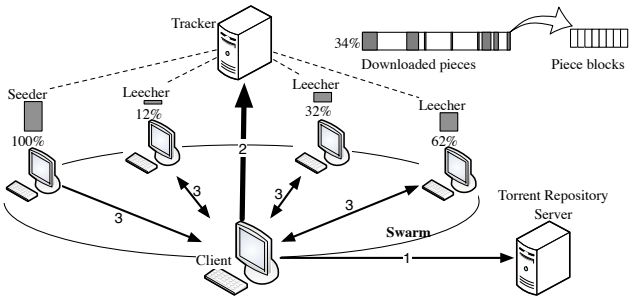

Figure 1: BitTorrent Architecture.

1526-7539/11 $26.00 © 2011 IEEE
DOI 10.1109/MASCOTS.2011.46

IEEE computer society

When a client wants to acquire some content, the first step (1) is to get a meta-data file from a public web server hereafter called Torrent Repository Server. This file contains the address of one or more Trackers and information about the content [4], [7]. The next step (2) performed by the Client is to announce to the Tracker its desire to join the swarm, which is answered with a list of Peers sharing the same content. This announce is performed periodically, so that the Tracker can keep the list of Peers updated. Finally, the last step (3) is to try to establish connections with other peers in order to start sending and receiving content. These connections operate over TCP and use the PeerWire Protocol [7].

BitTorrent employs three mechanisms for improving content sharing in the Swarm: 1) division of the content into pieces, used to allow concurrent download and improve bandwidth usage [8]; 2) selection of piece for download, in order to avoid problems such as piece starvation or stalling of download completion (the so-called *Rarest First* and *End-game mode* algorithms, respectively); and 3) limitation on the number of Peers downloading data from the Client via the so-called *Choking Algorithm*, aiming to enforce the tit-for-tat mechanism and to ensure a reasonable use of the uplink resources.

The official specification [4] and other works about BitTorrent [5], [7] present a detailed description of the protocol components, but the behavioral aspects are unclear and usually left open for the programmers to define. Our simulation aims to implemented the official BitTorrent protocol behavior as close to the real implementations as possible. Therefore, we used an unofficial specification from [7], researched related articles and talked with BitTorrent clients's developers from the #bittorrent IRC channel at irc.freenode.net. One result from this work is the description of dynamics of the PeerWire Protocol through state machines, shown in Figures 2, 3 and 4. There are two possible types of transitions in the state machines: the processing of a PeerWire message or the result of some event from the application. Furthermore, an action can be either the sending of a PeerWire message or a processing in the application.

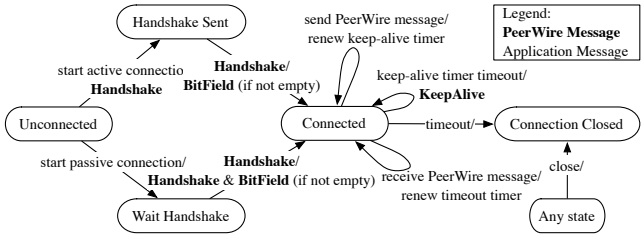

Figure 2: Connection State Machine

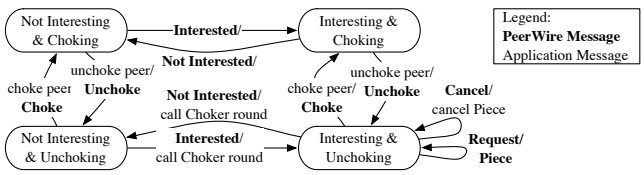

Figure 3: Upload State Machine

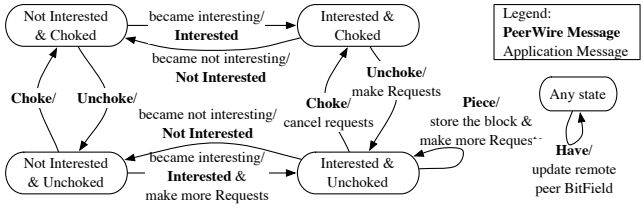

Figure 4: Download State Machine

The Connection State Machine deals with the establishment and maintenance of the connection, the Download State Machine keeps information about the content updated and requests the desired blocks from the Peer, and the Upload State Machine is responsible for choking or unchoking the Peer and also for responding to block requests.

It is possible to identify a complementarity between the download and upload mechanisms, as they exchange messages with each other. The exception to this behavior are messages that update the Client's view on the Peer's content.

We analyzed the source code of the official BitTorrent Client version 4.0.0 [9] in order to define the *Choking Algorithm*, since the official specification is not clear about some of its aspects. This algorithm is applied every 10 seconds, which configures a *choking round*. Extra calls to the algorithm will also happen every time an unchoked peer becomes interested or not interested in some content, or when one of the peers occupying the upload slot disconnects. These extra calls do not count as choking rounds.

1 At the beginning of every three choking rounds (i.e., every 30 seconds), one peer that is choked and interested is selected at random and unchoked. This is the optimistically unchoked peer, which will stay unchoked until the next optimistically unchoked peer is selected.

2 Each time the algorithm runs, the three interested, fastest downloaders are chosen for unchoking. However, the choice list will not contain snubbed peers, which are those peers that did not upload any blocks in the last 30 seconds.

If the optimistic peer is among the three fastest peers, another peer from all of the connected peers is selected at random and unchoked. This process is repeated until an interested peer is unchoked. Nonetheless, if the optimistic peer leaves the fastest peer set, it will return to its optimistic state. Not interested peers that are unchoked will cause an extra call of the algorithm the instant they become interested. In this manner, the peer list is quickly recomputed.

III. Simulation Model Design and Implementation

Our simulation was developed using OMNeT++ Framework and INET Framework. The former is a framework for developing event simulations with facilities for network models, while the latter supplements the former with several Internet protocols. The behavioral aspects of the BitTorrent model are coded in C++, while other aspects such as network elements are provided by the frameworks.

The model can be separated into network, Client and Tracker. The network is modeled with INET, being composed

of routers connected by Ethernet links and communicating over TCP/IP. Static, configurable network topologies can be easily created with the facilities provided by OMNeT++ Framework, facilitating the modeling of different experiments. Both the Client and the Tracker models are modeled as TCP applications attached to the TCP layer of INET hosts. They communicate with the TCP module from the INET host using sockets, much like a real TCP application. An overview of the model architecture can be seen in Figure 5.

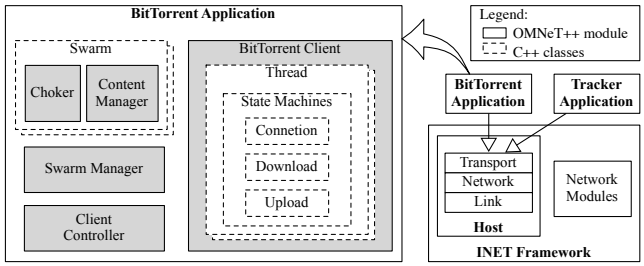

Figure 5: Architecture of the simulation model.

The Tracker Application is a simple HTTP server which can serve multiple connections, extended from a TCP-based server application from INET.

The BitTorrent Application, on the other hand, concentrates most of the complexity of the simulation model, being composed of several simple modules that implement different aspects of the BitTorrent protocol. We chose to break the model into smaller parts in order to provide higher extensibility and clarity to the model. A BitTorrent Application module contains one instance of the Client Controller, the Swarm Manager and the BitTorrent Client modules. Moreover, there is one Choker and one Content Manager modules for each Swarm in which the client participates.

1) Client Controller: This module acquires the *.torrent* file and start the application, acting as the user of a real BitTorrent application. This module controls the time the Client enters the swarm and how long it keeps seeding after download completion. In addition, it is possible to compare different user behaviors simply by changing to a different implementation of the Client Controller, assuming this implementation follow the same interface.

2) Swarm Manager: The task of requesting a list of peers to the Tracker is performed by this module. For each Swarm joined, the Swarm Manager will dynamically create a Choker and Content Manager, which allows for multiple Swarms at the same time.

3) BitTorrent Client: The BitTorrent Client establishes the connection between the Client and the Peer. Each connection is associated with a thread that controls the communication between Peers. This thread is a model abstraction and does not correspond to a real multi-thread application. In event-driven simulations, parallel processing is achieved by having events occurring without advancing the simulation clock.

Each thread has access to the Choker and Content Manager modules relative to the Swarm and utilizes the three state machines described in section II to implement the PeerWire Protocol. Events from the Choker and the Content Manager modules result in application transitions issued to the Upload State Machine and Download State Machine, respectively. The threads process the related application and PeerWire transitions in a multitasking way, sharing one single processor, in order to offer more realism. A round-robin scheduling is used in order to choose which thread will process the message. In order to take into account the system's processing time, we simply apply a delay to the response of the messages.

The processing time is taken from a histogram built from real world traces. These values were taken by watching the time between the arrival of a message and the sending of the response. Because of that, the processing time of application transitions is already accounted by the histogram.

4) Choker and Content Manager: The Content Manager is responsible for maintaining an updated view of how the content is distributed among the connected Peers and for keeping track of which pieces are needed by the Client. It does so by keeping all connected Peers' BitFields (which contains information about what pieces they already own), as well as a BitField for the Client's content. Interest in other Peers is determined by comparing BitFields. The Choker will enforce fairness by controlling which Peers are allowed to download data from the Client.

IV. Experimental Results

The scenario adopted to measure the simulation's performance is a static network composed of one Tracker and several Peers connected to a router in a star topology, thus providing the connectivity between all elements. This quite simple scenario was chosen because we focused our efforts in modeling BitTorrent accurately and in building an extensible simulation. Furthermore, it avoids introducing unnecessary complexity, such as peers joining and leaving the swarm dynamically.

The file size used in the experiments was 20MB and the network bandwidth was set to 100Mbps. The number of Peers varied from 100 to 1000 in steps of 100. Furthermore, the time a peer enters a swarm was defined by an exponential distribution with an average value of 1 second. This corresponds to the simulation of a flash crowd in the swarm, since a great number of peers join in a short amount of time. The simulation was run on an Intel i5-760 2.8GHz with 4GB of RAM memory running Ubuntu Maverick 10.10. We employed OMNeT++ v4.1 and INET v.20100323.

Execution time and memory consumption grow in a fairly linear fashion, as shown in Table I, demonstrating it is feasible to meet system requirements for large-scale simulations with today's regular hardware.

Table I: Execution Performance

Number of Peers	Elapsed Time (s)	Time Growth	Memory Usage (MB)	Memory Growth Rate
200	492,60	2,56	31	81,92
400	1038,63	2,80	47	81,92
600	1505,21	2,33	62	81,92
800	2185,02	3,01	78,5	87,04
1000	2693,57	2,08	95	81,92

It is possible to infer that the simulation execution time increases at a rate of 2.78 seconds per peer, with the goodness of fit shown by an R^2 of 0.9975. The rate of memory growth is $79,9$kB per peer, with an R^2 of 0.9996. The memory footprint of the simulation core engine is 15MB.

Another experiment performed had as goal determining the influence of the time a peer enters a swarm over the download time. The experiment configuration was set as ten repetitions with 1000 peers and 10% seeders. Figure 6 shows the results.

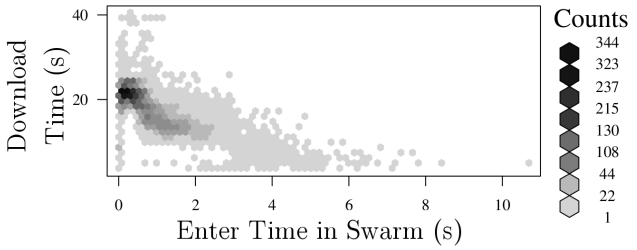

Figure 6: Effect of join instant on the download time

An important aspect of this graph is the upward line close to the origin, which shows that the first peers have a lower download time. This is a consequence of these peers connecting mostly with seeders. The darker area in the graph shows peers that connected with other recently connected peers. Since they do not own any content, they must wait for a peer that is connected to a Seeder to complete the download of some pieces. Finally, peers that connect later will encounter other peers already owning some part of the content. Thus, they can start requesting pieces as soon as they establish connection.

V. Related work

The popularity of BitTorrent resulted in many studies focused on simulations. GPS [2] presents a general P2P simulation framework developed in Java. The authors were concerned with building scalable simulations presenting good accuracy, without resorting to the full TCP stack. PeerSim [10] project is a popular general purpose P2P simulator also created with Java, which covers the implementation of some popular protocols. The simulation provides two engines: one using cycle-based simulation that does not model any of the network details, and another event-based engine that models the transport layer. BIT-SIM [11] was specifically developed as a simulation tool for the research community and uses OMNeT++ version 3 as a way to create a modular and extensible simulator. This simulator provides two models for the underlying network: the first, aimed at accuracy, uses the TCP/IP protocol stack fully implemented by INET framework; the other, focused on scalability, is a simple network implemented by the OverSim framework, in which the hosts exchange messages directly. BitTorrent Simulator [3] project consists of a message-based BitTorrent simulation developed upon the OMNeT++ as part of the Routing in Overlay Networks (ROVER) project. The differences presented by our simulation are:

1)The modeling of a more realistic processing mechanism, while only BitTorrent Simulator mentions about the processing time but chooses a simple delay model for the PeerWire messages;

2)The modularity and facility of customization inherited from the OMNeT++ Framework, since GPS and PeerSim simulations do not use any publicly available framework;

3)High number of simulated nodes, as only BitTorrent Simulation and GPS show results with thousands of nodes, albeit none of them model the underlying network in detail; and

4)The capacity to model multiple concurrent swarms, a feature that is not present in the other simulators.

VI. Considerations and Future Work

In this paper we present a BitTorrent simulation framework implemented using OMNeT++ and INET, focused on the realism of the model. Our simulator is an interesting tool for research in the area, and presents some appealing features such as the multiprocessing of messages and the support for multiple swarms. It also inherits aspects such as configuration easiness and extensibility from the OMNeT++ simulation architecture, facilitating its use. The experiments performed using the simulator shows that there is no need of expensive hardware for simulating the BitTorrent model, which is detailed in this document.

Our future work includes some improvements over the current version of the simulator. One example refers to the fact that this version consists only on a statically defined network and, thus, it does not give support to scenarios in which peers can be dynamically created and destroyed. Additionally, statistical analysis could be empowered by the acquisition and exhibition of extra information, such as Peer download/upload rates and link bandwidth usage.

Acknowledgment

This work was supported by the Innovation Center, Ericsson Telecomunicações S.A., Brazil.

References

[1] H. Schulze and K. Mochalski, "Internet study 2008/2009," 2009. [Online]. Available: http://www.ipoque.com/resources/internet-studies/internet-study-2008_2009

[2] W. Yang and N. Abu-Ghazaleh, "GPS: a general peer-to-peer simulator and its use for modeling BitTorrent," in *MASCOTS*, 2005, p. 425–432.

[3] K. D. Vogeleer, D. Erman, and A. Popescu, "Simulating BitTorrent," in *Simutools 2008*, ser. Simutools '08. Belgium: ICST, 2008, p. 2:1–2:7.

[4] B. Cohen, "The BitTorrent protocol specification," 2008. [Online]. Available: http://bittorrent.org/beps/bep_0003.html

[5] M. Konrath, M. Barcellos, and R. Mansilha, "Attacking a swarm with a band of liars: evaluating the impact of attacks on BitTorrent," in *P2P2007*, 2007, pp. 37–44.

[6] B. Cohen, "Incentives build robustness in BitTorrent," in *P2P Econ*, 2003. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.14.1911

[7] "Bittorrent specification wiki." [Online]. Available: http://wiki.theory.org/BitTorrentSpecification

[8] P. Marciniak, N. Liogkas, A. Legout, and E. Kohler, "Small Is Not Always Beautiful," in *IPTPS'2008*, Florida United States, 2008.

[9] B. Cohen, "Bittorrent client 4.0.0." [Online]. Available: http://download.bittorrent.com/dl/

[10] M. Jelasity, A. Montresor, G. P. Jesi, and S. Voulgaris, "The Peersim simulator." [Online]. Available: http://peersim.sf.net

[11] K. Katsaros, V. Kemerlis, C. Stais, and G. Xylomenos, "A BitTorrent module for the OMNeT++ simulator," in *MASCOTS*, 2009, pp. 1–10.