A
Project Report on

BIT-TORRENT PROTOCOL FOR DATA MOVEMENT WITHIN A GRID

Submitted to the partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

Session 2015-16

SRM UNIVERSITY
NCR CAMPUS, Sikri Kalan, MODINAGAR

Submitted by:
Animesh Patni (1031230295)

Supervised by:
Dr. A. K. Soni
Professor
Department of CSE
SRM University

BONAFIDE CERTIFICATE

This is to certify that Project Report entitled "**Bit-torrent Protocol for Data Movement Within a Grid**", which is submitted by **Animesh Patni (1031230295)** in the partial fulfillment of the requirement for the award of degree **B.Tech** in **Department of Computer Science and Engineering** of **SRM University**, **NCR Campus, Modinagar, Ghaziabad** is a record of the candidate own work carried out by them under my own supervision. The matter embodied in this thesis is original and has not been submitted for the award of any other degree.

…….....

Dr. R. P. Mahapatra
HOD(CSE)
SRM University
NCR Campus
Modinagar,
Ghaziabad

….....

Dr. A. K. Soni
Supervisor
Professor (CSE)
SRM University,
NCR Campus
Modinagar, Ghaziabad

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

I would like to express my hearty gratitude to **Dr. (Prof.) Manoj Kumar Pandey, Director, SRM University NCR Campus, Modinagar**, under whose auspices I was able to work on our project work.

Also, I extend my sincere regards and thanks to **Dr. R. P. Mahapatra, Associate Dean & Head of the Department, Computer Science and Engineering, SRM University, NCR Campus, Modinagar**, for his thoughtful and clairvoyant suggestion.

I feel compelled to articulate my thankfulness to **Ms. Arnika** and **Mr. Manish Bhardwaj, Project Coordinators, Computer Science and Engineering Department, SRM University NCR Campus, Modinagar**, for their enlightening dexterity which was a perennial source of inspiration.

I would also like to express my special thanks to D**r. A. K. Soni**, **Professor and Project Supervisor** of **SRM University, NCR Campus, Modinagar**, for his insight and knowledge on the subject, who imbued us with the feeling to work assiduously.

I am also indebted to all the teaching and non-teaching staff members of our college for helping me directly or indirectly by all means throughout the course of our study and project work.

Finally, I take this opportunity to thank my friends for their moral support and help and extended thanks to my well wishers.

Animesh Patni (1031230295)

DECLARATION

I, Animesh Patni (1031230295) hereby declare that the work which is being presented in the project report **"**Bit-torrent Protocol for Data Movement within a Grid**"** is the record of authentic work carried out by me during the period from January '16 to May '16 and submitted by me in partial fulfillment for the award of the degree **"Bachelor of Technology** in **Computer Science and Engineering"** to **SRM University, NCR Campus, Ghaziabad (U.P.).** This work has not been submitted to any other University or Institute for the award of any Degree/Diploma.

(Signature)
Animesh Patni (1031230295)

ABSTRACT

Data-centric applications are still a challenging issue for Large Scale Distributed Computing Systems. The emergence of new protocols and softwares for collaborative content distribution over Internet offers a new opportunity for efficient and fast delivery of high volume of data. This paper presents an evaluation of the BitTorrent protocol for Data Communication within Grids. We first present a prototype of a generic subsystem dedicated to data management and designed to serve as a building block for any Desktop Grid System. Based on this prototype we conduct experimentations to evaluate the potential of BitTorrent compared to a classical approach based on FTP data server. The preliminary results obtained with a 8-node cluster measure the basic characteristics of BitTorrent in terms of latency and bandwidth and evaluate the scalability of BitTorrent for the delivery of large input files. Moreover, we show that BitTorrent has a considerable latency overhead compared to FTP but clearly outperforms FTP when distributing large files or files to a high number of nodes. Tests on a synthetic application show that Bit-Torrent significantly increases the communication/computation ratio of the applications eligible to run on a Desktop Grid System. In this thesis we describe and analyze a full featured and extensible implementation of BitTorrent for the OMNeT++ simulation environment.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

1. P2P - Peer to Peer
2. P2PP - Peer to Peer Protocol
3. BT - Bit Torrent
4. FTP - File Transfer Protocol
5. TCP - Transmission Control Protocol
6. UDP - User Datagram Protocol

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

One of the major characteristics of today's Internet is that a large fraction of its traffic is due to content distribution applications. This has prompted researchers to consider redesigning the Internet so as to best support content delivery between publishers and subscribers of information, rather than communication between endpoints. The primary means of content dissemination are currently represented by Peer-to-Peer (P2P) applications, where multiple entities (peers) collaborate in order to efficiently exchange content. The technique of swarming, which is the concurrent downloading of content from multiple peers while simultaneously uploading to multiple other peers (first presented in BitTorrent), has greatly contributed to this development.

1.1. About Bit-Torrent

BitTorrent is a P2P content distribution system, comprised of a set of network protocols for realizing communication between the participating entities. It utilizes a simple bartering scheme for reducing parasitic behaviour such as free-riding, where peers only download and never upload content. Each time a host wants to distribute a set of files through BitTorrent, it organizes all of them as a sequence of bytes, it logically splits the sequence into equal size pieces and calculates a hash value for each piece. Then, a server that is willing to host the file exchange (not the file itself) is located; this is the tracker. The content metadata and supporting information such as hash values, piece size, file size, tracker address and so forth, are recorded in a file with an arbitrary name that acts as a description and summary of the content. This

metafile can then be distributed over the Web so that search engines can match user queries with metafile data.

When presented with a metafile, a client connects to the indicated tracker and asks for a list of other hosts currently participating in that particular exchange; all these hosts comprise the swarm. Note that the tracker does not itself participate in the swarm. Subsequently, each client constructs and maintains a bitmap with the pieces that it has (a new client initially has nothing, while the original distributor has everything). Then, each client randomly contacts other clients and exchanges bitmaps with them. Based on the bitmaps and other available data, such as path delay or bandwidth, each client can freely select the peers it will exchange pieces with. In general, pieces are exchanged in a tit-for-tat fashion, but for bootstrapping purposes, peers occasionally give pieces for free.

The success of BitTorrent stems from two key characteristics. First, to its ability to distribute resource consumption among the participating entities, thus avoiding the bottlenecks of centralized distribution. Second, to its aptitude to avoid performance deterioration and service unavailability by enforcing cooperation. While BitTorrent has drawn strong interest from researchers, most studies have concentrated on the performance evaluation of the protocol and its potential variations via the study of real world trace data sets.

1.2. Reason for Simulation

This approach has significant advantages with respect to the reliability of the extracted results, but it is characterized by inflexibility: there is no control over the participating peer characteristics and major protocol variations cannot be studied

without first implementing and then deploying them. Moreover, the trace collection process is cumbersome and the data gathered may be incomplete. For instance, collecting information for peers behind firewalls is difficult, while gathering information about the swarm's size and structure might be hindered by the tracker protocol itself. Analytical studies are even more problematic due to the highly dynamic character of BitTorrent: peers dynamically enter and leave the swarm, establish and tear down connections, decide on the preferred pieces of a file and chose to exchange data with peers or not. Simulation appears to be a more promising alternative, as it allows fast prototyping, provides the ability to perform large scale experiments, and offers a common reference platform for experimentation. Nonetheless, current BitTorrent simulators either consider coarse-grained representations of the underlying network, thus reducing the realism of the simulation, or omit many important features of the BitTorrent protocols.

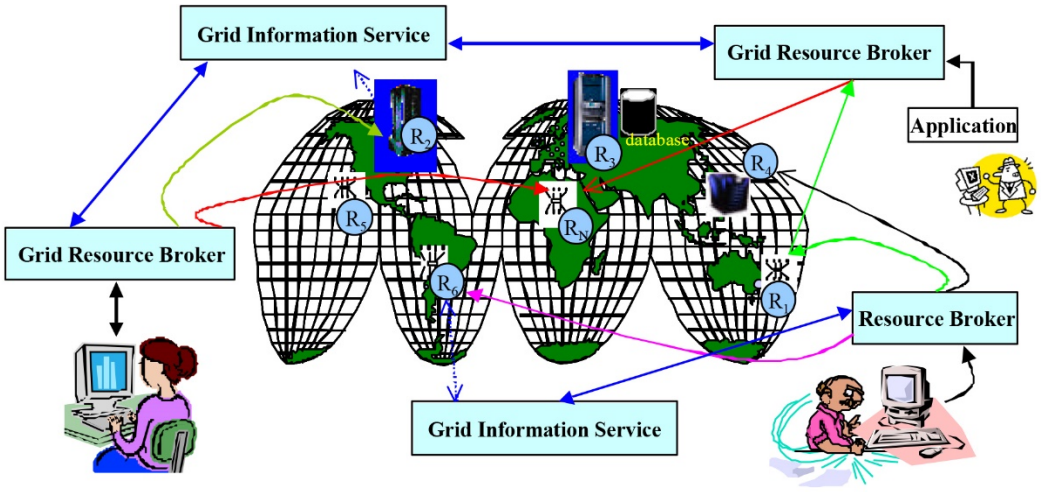

Figure 1: A generic view of World-Wide Grid computing environment.

1.3. Proposed Work

In this thesis, we present a full featured and extensible implementation of the BitTorrent protocol for the OMNeT++ simulation environment. We chose this platform due to its simplicity, its high degree of modularity, and the availability of several protocol implementations ranging from a complete TCP/IP protocol stack (provided by the INET framework) to a large set of overlay protocols (encapsulated in OverSim module). In the same vein, we present a churn generator that activates BitTorrent nodes in a network topology by following an arrival process derived from the analysis of actual BitTorrent traces.

CHAPTER 2

LITERATURE SURVEY

Owing to the unpopularity of the Bit-torrent protocol due to its usage for distribution of pirated content, not much research work has been done in this field. Nevertheless, the robust and resilient nature of the Bit-torrent protocol makes it a very attractive option for data communication, and has been used to a limited extent in large-scale web services to distribute files. Facebook uses it for file sharing, and Twitter for code deployments.

Recent developments in content distribution such as collaborative file distribution (BitTorrent, Slurpie, Digital Fountain) and P2P file sharing applications (EDonkey/Overnet, Kazaa), has proven to be both effective, viable and self-scalable. Today, a significant part of the Internet bandwidth is used by P2P traffic. The key idea, often featured as parallel download in the P2P applications, is to divide the file in a list of chunks. Immediately after a peer downloads a chunk from another peer, the peer serves the block for the other peers in the network, thus behaving itself as a server. Collaborative Content Distribution is a very active research topic and several promising strategies such as the use of network coding, are proposed to improve the performance of the system.

2.1. Paper 1

Title: Collaborative Data Distribution with Bit-Torrent for Computational Desktop Grids

Author: Baohua Wei, Gilles Fedak, Franck Cappello

Journal/Conference: International Symposium on Parallel and Distributed Computing (ISPDC'2005), Jul 2005, Lille/France, 2005. <inria-00001041>

Description:

- BitTorrent as a protocol for Data Diffusion in the context of Computational Desktop Grid has been investigated.
- A prototype was designed and found that even if Desktop Grid architectures often rely on centralized coordination components, they can easily integrate this P2P technology without fundamental changes on their model of security or deployment.
- An experimental performance evaluations of the protocol on a LAN cluster was conducted and it and showed that BitTorrent is efficient for large file transfers, scalable when the number of nodes increases but suffers from a high overhead when transmitting small files.
- Comparison with FTP-based centralized Desktop Grid on the execution of a multi-parametric application demonstrates that BitTorrent is able to execute application with a higher communication/computation ratio and to reduce the fault probabilities, which are two requirements for a broader use of Desktop Grid.

2.2. Paper 2

Title: A BitTorrent Module for the OMNeT++ Simulator

Author: Konstantinos Katsaros, Vasileios P. Kemerlis, Charilaos Stais and George Xylomenos.

Journal/Conference: Proceedings of the IEEE Mascots 2009, pp. 361–370

Description:

- An implementation of the BitTorrent set of protocols for the OMNeT++ simulation environment has been presented.
- The main target was to produce a realistic simulation environment that will enable the detailed evaluation of the protocol in fully controllable conditions.
- A set of tools, which enable the construction of realistic simulation scenarios that can capture the properties of the Internet-like network topologies and the real world deployment dynamics of BitTorrent participants.
- The goal is to enable simulation scenarios to be created where clients access the network over both symmetric and asymmetric links with various characteristics.

CHAPTER 3

BITTORRENT PROTOCOL

One of the difficulties faced during the implementation of the BitTorrent modules, was the lack of an official protocol specification. Despite the immense embrace of BitTorrent from both the user and research community, no formal protocol specification has been drawn up yet. The only authoritative document available, describes the entities involved in the protocols, the basic concepts, and the rudimentary transactions among them, but it lacks behavioral and implementation details. In effect, we had to resort to the unofficial BitTorrent Protocol Specification, which nevertheless does not constitute a formal and unambiguous source of information. In fact, several attributes of the protocols appear to be under dispute. In the remainder of this section we provide a detailed description of the protocols implemented, clarifying at the same time our approach in all cases of dispute.

3.1 The Tracker Protocol

As we already mentioned, the distribution of a new file with BitTorrent starts by publishing a .torrent metafile; this metafile is distributed to peers using an out-of band channel, usually by posting it on a web page. Trackers are responsible for aiding peers to discover each other and form a swarm. In most cases, each metafile is served by a single tracker, but recent extensions to the protocol (not implemented by us) allow multiple trackers for each file or even no trackers at all. This is the trackerless approach, which employs Distributed Hash Tables (DHTs) for decentralized peer discovery. Clients communicate with the tracker via a simple textbased protocol,

layered on top of HTTP/HTTPS, using the tracker's URL stored inside the metafile. During the download phase, each client communicates with the tracker and publishes its progress (in terms of total bytes downloaded/uploaded), as well as its contact details (e.g., IP address, TCP port, identification info). These parameters are passed from the client to the tracker using the standard HTTP GET method. Note that most of the information announced by the client is for statistical purposes; only the IP address and TCP port of a client are crucial. After each such message, called a tracker request, the tracker randomly selects a set of peers and returns their contact details in a bencoded dictionary. This is the tracker response. The tracker discovers these details via the tracker requests made by the clients. In this manner, over time the peers discover increasing subsets of the swarm.

3.2 The peer-wire Protocol

The peer-wire protocol provides the core BitTorrent functionality (i.e., interaction with remote peers). In the following we first present an overview of the protocol and then proceed with the details of its operation, focusing on the most important features available in our implementation.

3.2.1 Protocol Overview

After contacting the tracker, a client attempts to establish TCP connections with the peers listed in the tracker response. Upon connection establishment, the two peers exchange HANDSHAKE messages in order to verify each other's identity and ensure that they are interested in the same torrent. This handshake is then followed by an

exchange of BITFIELD messages that contain the bitfield of each client (i.e., the bitmap denoting the availability of each piece at the client).

Based on that information a client can determine whether it is interested in one or more pieces of the remote peer. Note that this exchange is optional when the client has no pieces, since it would result in the exchange of useless information, and is therefore avoided in our implementation, a client collects information regarding the availability of the pieces that it is still missing in the subset of the swarm explored thus far. Based on this, it then decides which pieces to request from each peer. In general, if a peer does not hold any pieces that the client does not already have, a NOT INTERESTED message is sent to that peer to indicate the lack of interest for its data.

3.2.2 Connections

A client periodically learns about other peers by utilizing the Tracker protocol and parsing the peer list returned. The client joins the swarm by establishing connections with some of those peers. However, as noted in, each connection incurs an increase in signaling traffic, especially for bit field maintenance via the exchange of HAVE messages. Thus, our implementation provides configurable lower and upper bounds for the number of established connections, using the minNumConnections and maxNumConnections configuration parameters.

3.2.3 Piece Downloading Strategy

The piece downloading strategy refers to the policy followed in the selection of the pieces that will be requested from a peer. It is an important aspect of BitTorrent as it heavily affects the diversity of the pieces available in each peer. A low degree of diversity would result in low interest for a peer's pieces, thus causing degraded application performance. We have implemented the two most prevalent piece downloading strategies: rarest first and random first. Based on the information gathered during the BITFIELD and HAVE message exchanges, the former strategy selects those pieces that appear less frequently in a client's set of connected peers. This selection is randomized among several of the less common pieces, according to the rarest list size configuration parameter, in order to avoid multiple peers converging on the same piece. This way, peers download pieces that most other peers probably want, therefore facilitating data exchange. However, rare pieces are present only in a few peers, and it is possible that downloading from them may be interrupted due to a choking decision. Clients with no pieces in their possession would therefore have to wait for an optimistic unchoking event from a peer holding the same rare piece in order to continue downloading. The latter strategy avoids this problem by selecting a random piece which is more likely to be available from multiple peers, so that a choking decision would not have such an adverse effect.

3.2.4 Queueing

REQUEST messages refer to specific blocks of a piece. This facilitates fine-grained data exchange by enabling queuing of data requests. As common piece sizes vary from 256 KB to 1 MB or even larger, per piece requests would result in a many

redundant retransmissions in the event of a choking decision during piece transfer. A window-based queuing mechanism is employed for these requests, otherwise propagation delays would dominate the total download time. Since the exact nature of the queueing policy is under dispute, we implemented a generic queueing mechanism in which the user can specify the exact size of the queue. In this mechanism a client may send to a peer up to request queue length request messages for blocks. Once a PIECE message has been received, the client may send the next REQUEST message. In case a piece has been requested in its entirety and the request queue is not full, the client chooses another desired piece from that peer's bitfield according to the piece selection strategy and starts sending REQUEST messages for its blocks.

3.2.5 Choking Algorithm

For the choking algorithm we followed the guidelines, along with the ability to tune the choking algorithm as desired. The user may select appropriate values for the time between (optimistic) choking decisions, using the choking interval and optUnchoking interval configuration parameters, and the maximum number of (optimistically) unchoked peers, using the downloaders and optUnchokedPeers configuration parameters (see Table I). To enable content providers to offer advanced seeding capabilities, the above parameters can be separately configured for such nodes via the seederDownloaders and seederOptUnchokedPeers parameters. The parameter newlyConnectedOptUnchokedProb is the probability that the most recently connected peer will be preferred over previously connected ones in an optimistic unchoke decision.

3.2.6 Super Speeding

The super seed feature is especially useful for content distribution as it helps the initial seeder to avoid excessive bandwidth consumption while fostering data exchange between participating peers. A super seeder does not inform its peers that is has all pieces available, masquerading as an ordinary client. Initially, it pretends to possess no pieces and only later informs them about the availability of an individual piece with a HAVE message, as if it had just completed downloading it. The seeder either selects a piece it has never uploaded before or, if all pieces have already been uploaded at least once, a piece that has been uploaded only a few times. After the piece has been downloaded by a peer, the seeder will not inform it of other pieces until it sees this piece marked as available in the bitfield of other peers, implying that the first peer has in turn uploaded that piece. Our module implements this feature in all clients, but only enables it at the initial seeder via the super seed mode configuration parameter (see Table I), since super seeding is not recommended for ordinary peers. Instead, ordinary peers act as regular seeders after downloading all pieces.

CHAPTER 4

CREATING SIMULATION SCENARIOS

Our implementation was developed as a stand-alone INET framework application, therefore, in order to run a BitTorrent simulation, a network topology must be provided and the appropriate modules (*e.g.*, TRACKER, TRACKER CLIENT and PEER-WIRE) must be loaded as submodules of the compound modules representing the peers and the tracker. While this procedure is sufficient for testing the modules, it is cumbersome to use when constructing realistic scenarios such as:

- Large-scale network topologies that require careful handling of node module placement and interconnection.
- Random introduction of clients into the simulation, both topologically and chronologically, as the network description files cannot capture such dynamics.

These considerations indicate that there is a need for globally controlled, network-wide *dynamic module loading* in the construction of the simulation scenario. Hence, we turned to the OverSim overlay simulation framework, which provides several of the features required to establish realistic and dynamic simulation scenarios. It must be stressed however that our BitTorrent implementation is not OverSim dependent: it can optionally employ several of the features provided by OverSim. In the following sections we present the OverSim features that we exploited along with our enhancements.

4.1 Topologies

One of the big concerns in creating realistic simulation platforms for BitTorrent is the underlying network topologies. OverSim provides various underlying network structures, both simple (*e.g.,* SIMPLEUNDERLAY) and composite (*e.g.,* IPV4UNDERLAY). However, both models present significant limitations in representing a realistic network substrate. The SIMPLEUNDERLAY model was designed to provide a simple and scalable network substrate specially tailored for simulations focusing on the functionality of higher layer protocols, such as the set of overlay routing schemes provided by OverSim. In this model, packets are directly exchanged between end hosts completely neglecting the functionality of the underlying protocol stack. Packet delivery is performed by simply considering the characteristics of the communicating end hosts' access links and samples of end to end propagation delays derived from CAIDA's Skitter project.

This lack of protocol functionality and step-by-step routing in SIMPLEUNDERLAY turned our attention to the more realistic IPV4UNDERLAY model. In we addressed two important limitations of this model. First, the model only provides a distinction between backbone and access routers neglecting the complex structures imposed by the existence of multiple autonomous administrative domains. Second, the model provides no support for routing policy weights.

The first is revealed when considering the locality properties (with respect to the consumption of ISP-specific resources) of data exchanges in P2P content distribution applications, such as BitTorrent. It has been shown that BitTorrent's network-agnostic peer-wire protocol has an adverse impact on capacity related ISP costs by allowing

downloads from peers residing in external domains, even when the desired data are already present locally.

This has triggered several research efforts that would benefit from a simulator providing the flexibility to study ISP level aspects of the protocol performance. Hence, while OverSim's IPV4UNDERLAY model provides no access to such information, we have further enhanced our topology conversion tool to also preserve the unique Autonomous System numbers [19] produced by BRITE and to export them to a separate configuration file so that each router, as well as each attached end host, can be assigned the corresponding AS number. Direct access to this information is provided to the simulation programmer, facilitating the investigation of the aforementioned locality properties and protocol inefficiencies, as well as the implementation of location-aware schemes.

Second, OverSim does not make any distinction between the uplink and downlink characteristics of access links (*i.e.*, bandwidth). However, this distinction is important for providing realistic networking environments, since typical current access technologies, such as ADSL, do present this asymmetry. This issue becomes more important due to the fact that bandwidth heterogeneity results in a systematic unfairness of the peerwire protocol, as the download rates achieved are based on the tit-for-tat mechanism. IPV4UNDERLAY model to support a range of *channel* 5 characteristics for the two directions of each access link. Specifically, the simulation programmer is able to specify different channel options for the uplink and downlink of each access link type, along with the fraction of the total access links across the entire network that each channel type is assigned to. In the measurements presented below, we have set these values as depicted. For example, 40% of the participating

peers can download data at a maximum rate of 8 Mbps and upload data with a maximum rate of 1 Mbps.

4.2. Host Deployment

Having established a realistic network topology, the next step is to deploy the corresponding entities on it. As far as the tracker is concerned, to avoid coupling the network topology description file with the application, we extended the OverSim IPV4UNDERLAYCONFIGURATOR module to dynamically introduce the BitTorrent tracker in the network. Regarding the initial seeder, we implemented a separate deployment scheme, since in many realistic scenarios the initial seeder has different characteristics from ordinary peers. For example, the content might be a new Linux distribution (*i.e.*,, an ISO image file), hosted by a dedicated server with a high capacity access link, while ordinary peers participate in the swarm through ADSL links. Protocol parameters may also be altered to achieve a differentiated behavior between the initial seeder and the peers. For example, the initial seeder may optimistically unchoke multiple peers to speedup the distribution of the offered file. This was again achieved by extending IPV4UNDERLAYCONFIGURATOR and ACCESSNET modules' functionality and employing a separate host description file for the initial seeder: BTHOSTSEEDER. Note that our extensions check whether the scenario is BitTorrent related before proceeding to deploy an initial seeder or a tracker, thus preserving the base functionality of the affected modules. Unlike the tracker and the initial seeder, peers need to be randomly introduced into the network both in a topological and chronological sense (*i.e.,* they need to be placed at random nodes in random points of time). The churn models provided by the OverSim

platform, together with the underlying IPV4UNDERLAY configuration mechanism, constitute flexible mechanism for dynamically deploying peers in the network. However, the churn models available in OverSim were not designed to reflect the arrival processes of real applications. Instead, they provide a generic mechanism for the arrival process and several distributions, which describe the duration of a peer's presence in the network. Since in BitTorrent this duration depends on protocol operation rather than on a predetermined distribution, we focused on modeling the arrival process. Using the OverSim churn generator mechanism, we implemented the BITTORRENTCHURN model that reproduces the arrival process of BitTorrent clients presented.

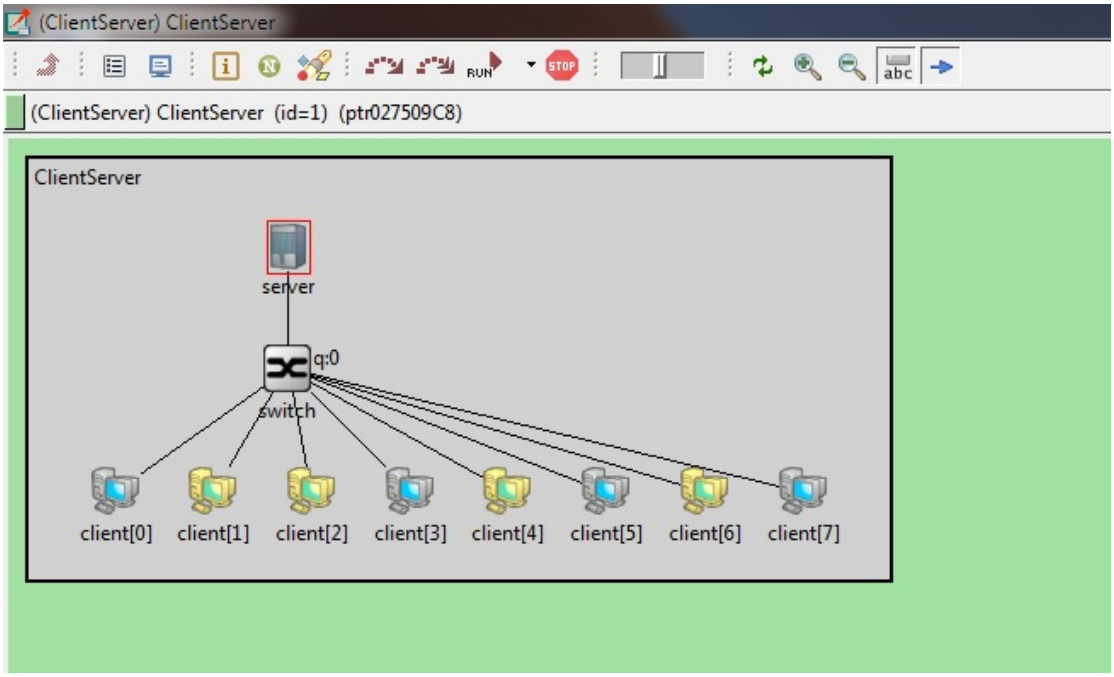

Fig. 2. Simulation of TCP network of 8-nodes using OMNeT++

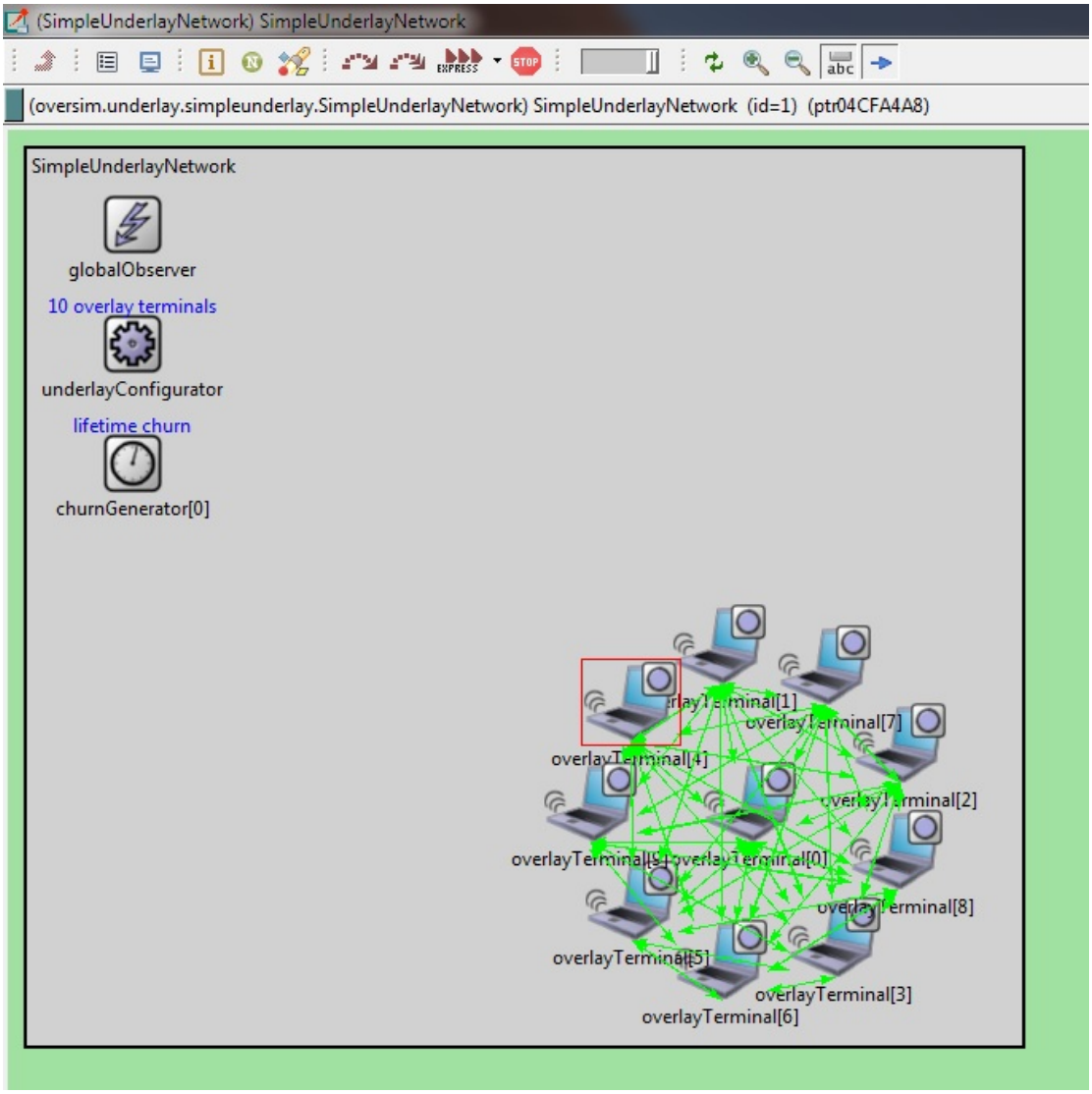

Fig. 3. Simulation of Bit-torrent network of 10-nodes using OMNeT++

CHAPTER 5

EXPERIMENTAL RESULTS

This section presents the experimental evaluation of Bit-Torrent for Computational Desktop Grids. We begin with a description of the experimental conditions, then we present the results obtained in our experiments.

5.1 Experiment Setup

The performance results reported in this paper were obtained with OMNeT++ v.4.2.2, INET v.20061020, and OverSim v.20080416 (patched with our modifications), on an Intel Core i3 4005U @ 1.7GHz processor with 4 GB of RAM running Windows 7 Ultimate 64-bit edition.

5.2 Basic performance of BitTorrent

This first experiment compares the performance of BitTorrent versus FTP when distributing a file to a set of 20 nodes. The file size varies from 1 to 250 MB. Figure 2 presents the minimum, maximum and average completion time in seconds for the file transfer. The time is measured on each receiver node and is averaged over 30 experiments. The close-up figure plots with a logarithmic scale, the latency in second for transferring small files (size between 10KB to 50MB).

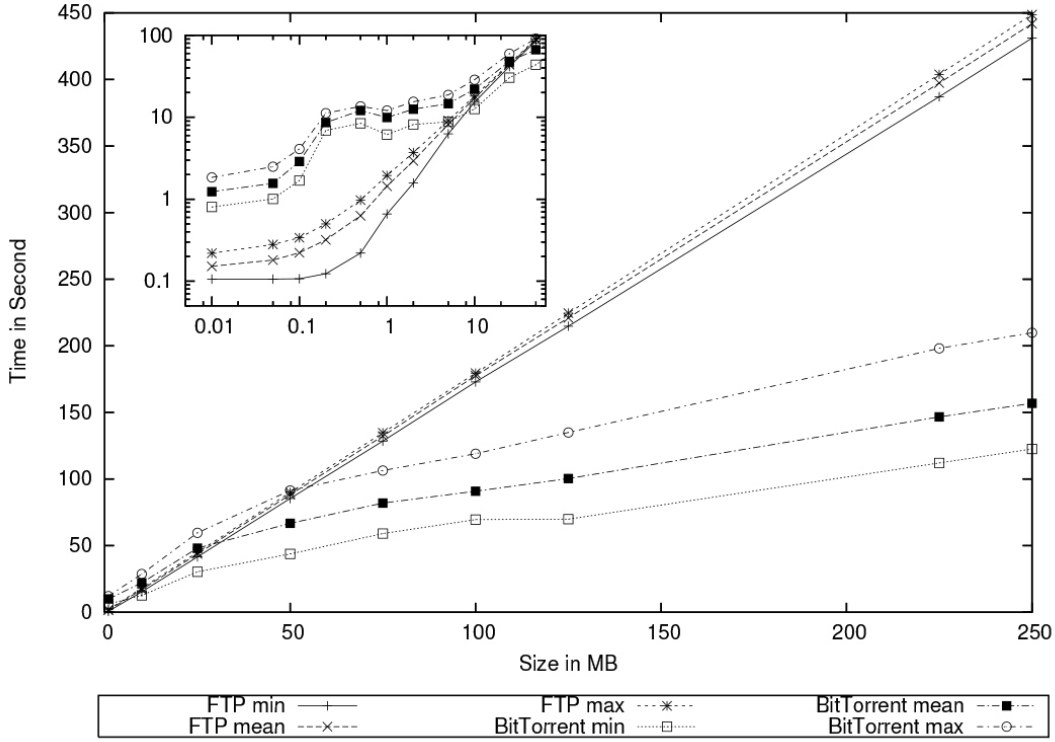

Fig. 4. The minimum, maximum and average completion time in second for the distribution of the file of a size varying from 1 to 250MB to 20 nodes. The figure presents a close-up of the latency at logarithmic scale for the distribution of small files, with a size comprised between 10K and 50MB

BitTorrent clearly outperforms FTP when the file size is greater than 20MB. After this crossover point, the curve grows softly with a slope approximately equals to 0.45. The cooperation between the nodes is effective to decrease the transfer time even if a modest number of hosts is involved (in this case 20 nodes). This shows a clear potential of using BitTorrent for large file transfer instead of FTP. The difference between the minimum, mean and maximum curves is discuss later in the paper.

If BitTorrent protocol seems appealing for large file, FTP is more efficient for small files transfer. Multi-parametric applications are often composed of a simulation code

associated with one or several configuration text files, which describe the parameters of the execution. Thus, this kind of studies implie the transfer of numerous small files.

When considering small file transfers, BitTorrent presents an overhead higher than FTP: respectively about 0.8 and 0.1 second. The BitTorrent overhead is due to the various steps the protocol imposes before actually starting the file communication.

First the downloader has to read the .torrent file to extract the information about the chunks and the tracker, next to contact the tracker to receive the list of peers. Finally the downloader needs to wait for the seeder or another peer to upload the chunks of file, with the additional constraint that upload queue is limited to 4 slots.

To cope with this overhead, Desktop Grids designers could:

1) use a dual protocol (FTP+BitTorrent) according to the size of the data,

2) or embody the small parameter file with the task description (this solution exists with XtremWeb).

Finally Figure 3 presents the bandwidth as measured locally on each node. We observe that the bandwidth for the FTP protocol is kept constant due to the fair sharing of the server bandwidth among the downloaders. In contrast, BitTorrent shows a noticeable improvement of the throughput when the file size increases. When the number of hosts is low, BitTorrent is effective for large file size.

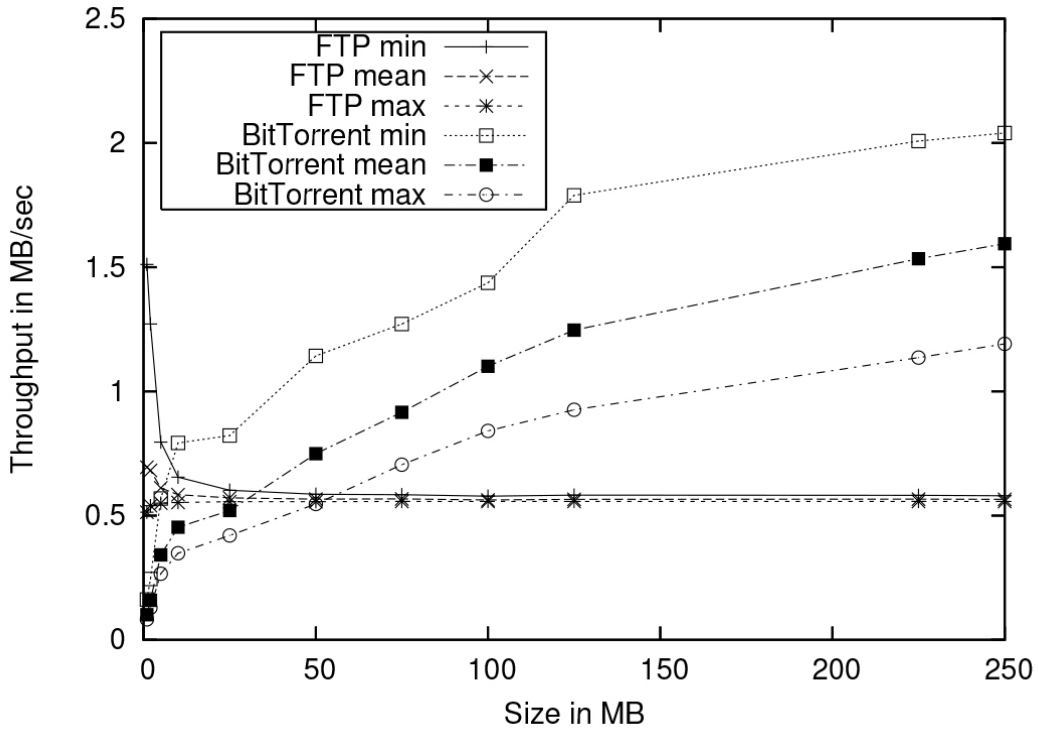

Fig. 5. Bandwidth (minimal, mean and maximal) in MB/s obtained when transferring a file to 20 nodes, with a size comprised between 1MB and 250MB

5.3 Communication Patterns of BitTorrent

The following experiment compares the profiles of the file distribution of the two protocols. We plot in the Figure 4 the download times when a file of 100MB is delivered to 20 nodes. The experience is repeated 30 times and each point on the plot represents the time to completion for the file transfer for one node during one run. The horizontal axis represents the measures for every run. The upper three curves (maximum, mean and minimum) refer to the FTP measurements and the lower three ones to the BitTorrent measurements.

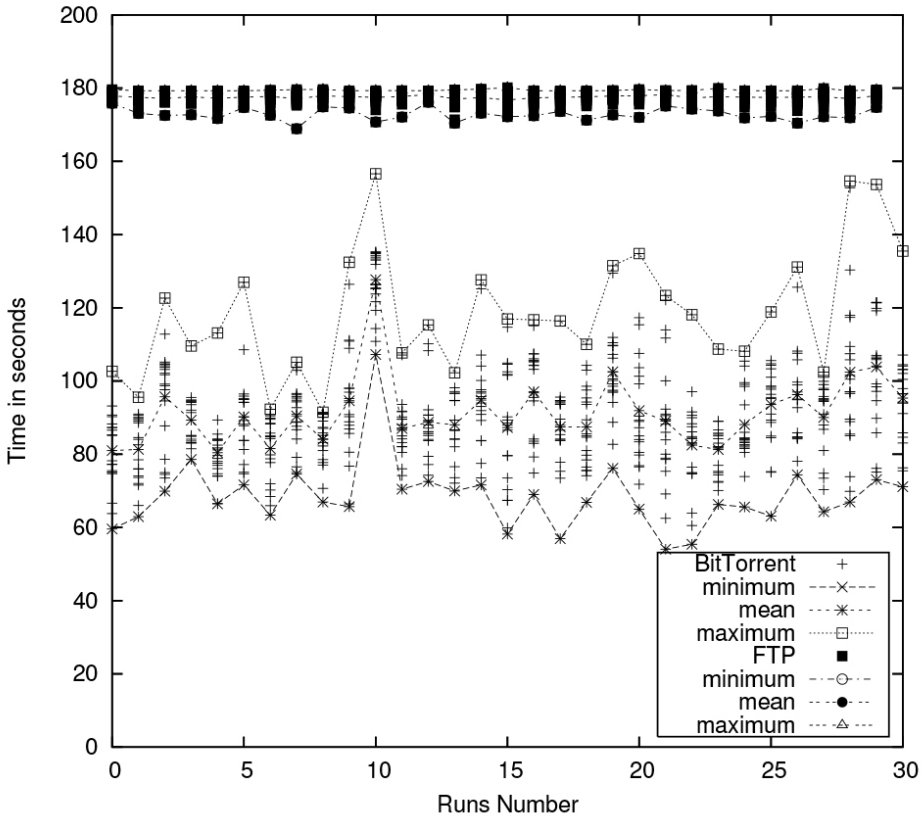

Fig. 6. Profiles of downloads for a 100 MB file by 20 machines: each point on the plot represents the time to completion for the file transfer for one node during one run. The three curves represent the maximum, mean and minimum for both FTP and BitTorrent.

The figure shows that: 1) the download time is always lower for BitTorrent 2) while the distributions of the download time for FTP are quite homogeneous, BitTorrent suffers from less consistent performance. With BitTorrent, the download time can vary from a factor 3 between the fastest and the slowest node.

In order to understand this variations we have instrumented the BitTorrent client. On each peer, we log every communications made to the other peers. The Figure 5 presents the communication pattern of the BitTorrent protocol when diffusing a file of

size 250 MB to 20 nodes. A vector represents a communication from a sender to a receiver (vertical axis) during a period of time (horizontal axis) with no more than

10 second idle. We discarded vectors with a volume of data is less than 1MB, however more than 99.45% of the total volume transmitted is presented.

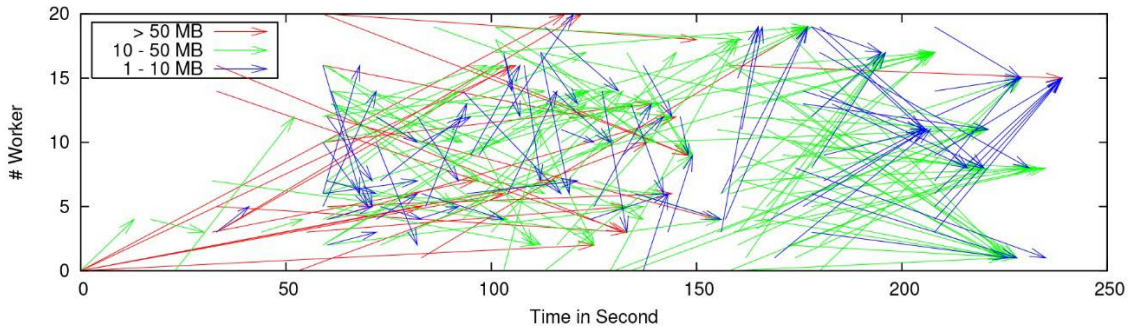

Fig. 7. Communication pattern of the BitTorrent protocol when diffusing a 250 MB file to 20 nodes. The color of the vector presents the volume of data transmitted.

The figure shows 1) nodes start to upload file chunks to other nodes before receiving the whole file, 2) largest communications are performed at the start of the diffusion and 3) the topology at the beginning of the diffusion represents a tree and a pipeline is organized to transmit the whole file to the last served nodes.

5.4 Scalability Evaluation

With this experiment, we evaluate the scalability of BitTorrent compared to FTP. The benchmark consists of a pool of nodes varying from 1 up to 64 which simultaneously download a 50MB file. The file is located on a central FTP server or on the first BitTorrent client. Each node measures the time to complete the download and the Figure 6 presents the average time to transfer the file, compared to the number of workers.

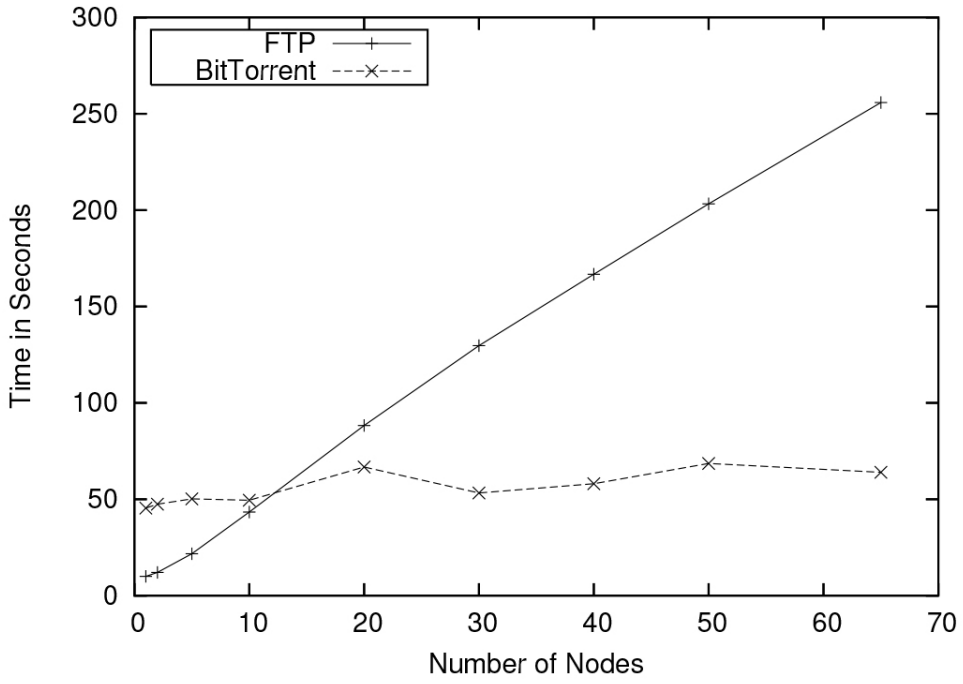

Fig. 8. Completion Time for the distribution of a 50MB file to a varying number of nodes. The vertical axe shows the time in second, the horizontal axe shows the number of nodes.

As seen in the figure the download time of BitTorrent remains stable as the number of resources increases while it linearly increases with FTP. This results shows that the scalability of BitTorrent is the one expected when the number of nodes is high. Other studies based on simulations implying up to 5000 nodes confirm this general trend. However, with a 50 MB file, there is a crossover point around 10 workers where FTP is more efficient than BitTorrent due to the overhead of BitTorrent.

5.5 Failure Rate

The last experiment confronts BitTorrent and FTP against node volatility. This experiment evaluates the transfer failure rate, that is the probability that a host will become unavailable before a transfer complete. The higher efficiency of BitTorrent permits to decrease the transfer failure rate by a factor 2.8.

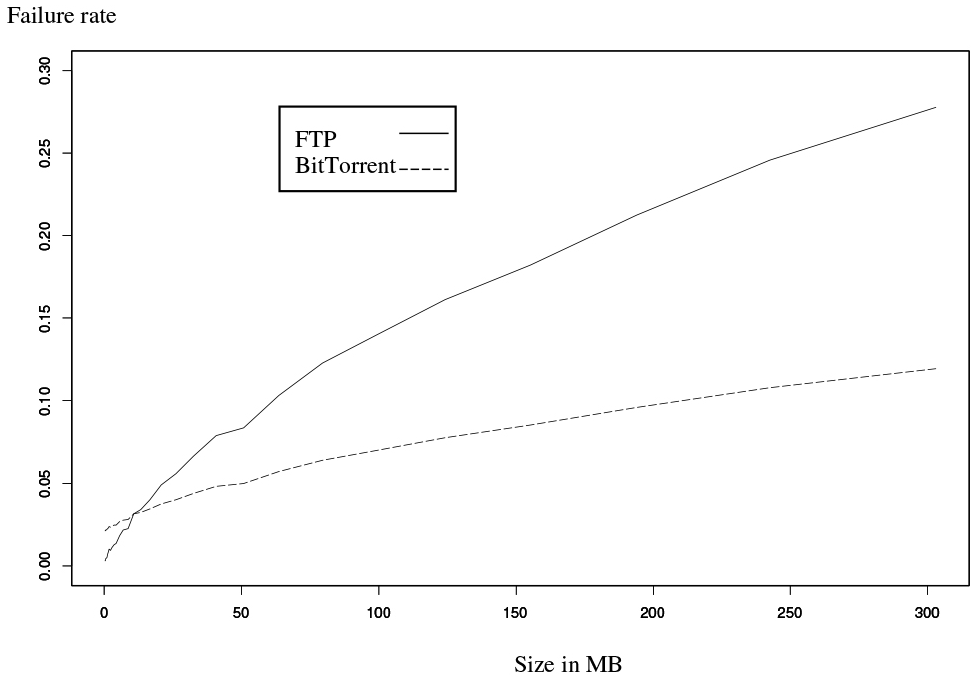

Fig. 9. Transfer failure rate versus transfer size (in MB)

CHAPTER 6

RESULTS

6.1 Related Work

Simulating the operation of BitTorrent is a difficult task due to the inherent protocol complexity and the lack of concrete specifications. The multitude of possible policies, such as those for piece selection and choking decisions, as well as parameter values, such as the choking interval and the number of unchoked peers, creates a highly dynamic environment. Hence, in order to isolate important protocol aspects, works such as ignore the influence of the underlying protocols and focus on the application logic. However, this design decision obviously incurs a non-negligible degree of inaccuracy. TCP dynamics, propagation delays and the potential queuing of packets in routers are important factors that can affect, for example, the perceived download rate in a client and therefore alter its choking decisions. Our implementation avoids this situation by providing almost all features specified in and by operating on top of full-fledged simulation platform. To the best of our knowledge, there is only one packet-level BitTorrent simulation module available implemented for ns-2. While this implementation shares our goal of providing a realistic simulation environment, our implementation provides several additional features, such as the entire Tracker protocol as well as the endgame mode. Furthermore, our implementation allows fine grained tuning of the protocols by providing several configuration parameters not available in, such as optUnchoking Interval.

Finally, we note that our implementation is not the first attempt to incorporate BitTorrent in the OMNeT++ simulation platform. A swarming-based module is presented in, but it only provides a bare-bones subset of the BitTorrent protocol features, since it lacks the tit-for-tat mechanism. This module was also developed using trivial network topologies with dedicated non-TCP connections among all pairs of peers. Likewise, a BitTorrent implementation for OMNeT++ that does not utilize the INET framework is presented at. The authors implemented most of the BitTorrent features (e.g., rarest first piece selection strategy, chocking), but their model suffers from the same deficiencies as do (i.e., the underlying protocol stack is missing and the corresponding peer modules are linked to each other using abstract connections). More importantly, critical parameters such as link delay/bandwidth and swarm inter arrival times are modeled using the probabilistic distributions provided by OMNeT++. In contrast, in our implementation we can use realistic and detailed network substrates, deploying the corresponding entities using a churn model derived from real life traces. According to the authors, the next major version of their implementation will incorporate a realistic underlay. However, this version is not yet available and it is not clear whether it will be assembled on top of the INET framework, or on a custom-made equivalent.

6.2 Future Work

Our plans for future extensions to the simulator primarily focus on enriching the set of extracted statistics. To this end, we aim to track the achieved download rate of each peer during its participation in the swarm. Our intention is to make this information, as well as the currently provided statistics, further categorizable with respect to the

access link capabilities of the peers so as to allow the fine grained analysis of the derived measurements. We also plan to take advantage of the enhanced topology model employed by providing support for the extraction of topology-aware metrics (i.e., metrics for specific areas of the network such as selected access networks). This will involve both per peer statistics (e.g., download time) and per area statistics (e.g., volume of data exchanged with neighboring access networks). Furthermore, we plan to enable the separate extraction of statistics for the operation of the initial seeders. Regarding the creation of simulation scenarios we intend to provide support for multiple initial seeders and to further enhance the tuning of the peer-wire protocol on these nodes by providing control over several parameters such as the choking interval and the number of optimistically unchoked peers. Such features, in combination with the seeder-specific statistics, are expected to enable the fine-grained investigation of the protocol performance from the perspective of content providers. Moreover, by taking advantage of the rich suite of overlay protocols provided by OverSim we currently investigate the design space for the implementation of the DHT-based trackerless extension.

6.3 Conclusion

In this paper we have presented an implementation of the BitTorrent set of protocols for the OMNeT++ simulation environment. Our main target was to produce a realistic simulation environment that will enable the detailed evaluation of the protocol in fully controllable conditions. Towards this direction we have faithfully implemented the only available protocol specification, trying to make our module resemble an actual BitTorrent implementation. Furthermore, we created a set of tools, which enable the

construction of realistic simulation scenarios that can capture the properties of the Internet-like network topologies and the real world deployment dynamics of BitTorrent participants. Our goal is to enable simulation scenarios to be created where clients access the network over both symmetric and asymmetric links with various characteristics. The results presented in this paper indicate that we can perform detailed medium-scale simulations of realistic Internet-like swarming scenarios in commodity hardware and, equally important, that is makes sense to do so.

REFERENCES

[1] PSIRP, “Publish-Subscribe Internet Routing Paradigm,” Jun 2009.

[2] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, “Should Internet service providers fear peer-assisted content distribution?” in Proc. of the Internet Measurement Conference (IMC), (Berkeley, CA, USA), pp. 63–76, Oct 2005.

[3] B. Cohen, “Incentives build robustness in BitTorrent,” in Proc. of the Workshop on the Economics of Peer-to-Peer Systems, (Berkeley, CA, USA), pp. 116–121, Jun 2003.

[4] BitTorrent.org, “BitTorrent Protocol Specification,” Jun 2009.

[5] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, and X. Zhang, “A performance study of BitTorrent-like peer-to-peer systems,” IEEE Journal on Selected Areas in Communications, vol. 25, no. 1, pp. 155–169, 2007.

[6] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, “The Bittorrent P2P file-sharing system: Measurements and analysis,” in Proc. of the International Workshop on Peer-to-Peer Systems (IPTPS), (Ithaca, NY, USA), pp. 205–216, Feb 2005.

[7] K. Katsaros, V. Kemerlis, C. Stais, and G. Xylomenos, “BitTorrent module for OmNEt++,” Jun 2009.

[8] I. Baumgart, B. Heep, and S. Krause, “OverSim: A flexible overlay network simulation framework,” in Proc. of the IEEE Global Internet Symposium, (Anchorage, AK, USA), pp. 79–84, Jan 2007.

[9] E. Zegura, K. Calvert, , and S. Bhattacharjee, “How to model an internetwork,” in Proc. of the IEEE INFOCOM, vol. 2, (San Francisco, CA, USA), pp. 594–602, Mar 1996.

[10] TheoryOrg, “BitTorrent Protocol Specification v1.0,” Jun 2009.

[11] EDonkey, .Edonkey, overnet homepage. January 2002, http://www.edonkey200.com/.

[12] FastTrack, .P2P Technology. KaZaA Media Desktop. January 2002, http://www.fasttrack.nu/.

[13] CAIDA, .CAIDA, the Cooperative Association for Internet Data Analysis: Top applications (bytes) for subinterface 0[0]: SD-NAP traf_c, 2002.

[14] D. Qiu and R. Srikant, .Modeling and Performance analysis of BitTorrent-like Peer-to-Peer Networks,. SIGCOMM Comput. Commun. Rev., vol. 34, no. 4, pp. 367.378, 2004.

[15] C. Gkantsidis and P. Rodriguez, .Network Coding for Large Scale Content Distribution,. in Proceedings of IEEE/INFOCOM 2005, Miami, USA, March 2005.

[16] M. Izal, G. Urvoy-Keller, E. W. Biersack, P. A. Felber, A. A. Hamra, and L. Garces-Erice, .Dissecting BitTorrent: Five Months in a Torrent's Lifetime, in Proceedings of Passive and Active Measurements (PAM), 2004.

[17] B. Kreaseck, L. Carter, H. Casanova, and J. Ferrante. On the interference of communication on computation in java, in the 3rd International Workshop on Performance Modeling, Evaluation and Optimization on Parallel and Distributed Systems (PMEO-PDS'04), Santa Fe, New Mexico, April 2004.

[18] J. A. Pouwelse, P. Garbacki, D. Epema, and H.J.Sips, .A Measurement Study of the BitTorrent Peer-to-Peer File Sharing System, Delft University of Technology, Tech. Rep. PDS-2004-007, 2004.

[19] B. N. L. Anthony Bellissimo, Prashant Shenoy, .Exploring the Use of BitTorrent as the Basis for a Large Trace Repository, University of Massachusetts, Tech. Rep., 2004.

[20] A. R. Bharambe, H. Cormac, and V. N. Padmanabhan, .Understanding and Deconstructing BitTorrent Performance. Microsoft Research, Tech. Rep., 2005.

[21] D. Kondo, M. Taufer, C. L. Brooks, H. Casanova, and A. Chien, Characterizing and evaluating desktop grids: An empirical study, in IPDPS 2004, IEEE/ACM International Parallel and Distributed ProcessingSymposium, Santa Fe, New Mexico, 2004.

[22] K. Eger, T. Hoßfeld, A. Binzenh¨ofer, and G. Kunzmann, “Efficient simulation of large-scale P2P networks: Packet-level vs. flow-level simulations,” in Proc. of the Workshop on the Use of P2P, GRID and Agents for the Development of Content Networks (UPGRADE), (Monterey, CA, USA), pp. 9–16, Jun 2007.

[23] UCB/LBNL/VINT, “The Network Simulator - ns - 2,” Jun 2009.

[24] K. De Vogeleer, D. Erman, and A. Popescu, “Simulating bittorrent,” in Proc. of the International Workshop on the Evaluation of Quality of Service through Simulation in the Future Internet (QoSim), (Marseille, France), Mar 2008.

[25] “BT-SIM a BitTorrent Simulator,” Jun 2009.