

# MTorrent: A Multicast Enabled BitTorrent Protocol

Piyush Agrawal Hitesh Khandelwal and R. K. Ghosh

**Abstract**—In this paper we address the problem of *repetitive data transmission in BitTorrent*, and propose a new protocol called *MTorrent* to restrict the same by exploiting *IP Multicast functionality* wherever available. A prototype system of our protocol have been implemented, and large scale measurement studies were conducted over Emulab network consisting of 44 nodes, spread over 9 LANs and 36 end clients. *MTorrent* leads to 44% reduction in download time, 65% reduction in traffic load on the Internet links and 40% reduction in download of redundant packets when compared to the BitTorrent. *MTorrent* is interoperable with BitTorrent, and requires only a few changes at the client end.

## I. INTRODUCTION

The recent wide-spread use of P2P applications such as SETI (setiathome.ssl.berkeley.edu), Napster (www.napster.com), and Gnutella (gnutella.wego.com) indicate that there are many potential benefits of deploying a fully distributed P2P system. Two of the obvious advantages are: (i) more resilience, and (ii) higher availability through large scale replication of content at large numbers of peers. However, in a heterogeneous environment such as the Internet, the performance is an equally important consideration.

Multicast is extremely useful for building highly distributed applications, It provides marked improvements in performance over that of unicast transmissions. A number of interesting non-trivial real-time multi-media applications have been developed in recent years by leveraging multicast services [18], [30], [23]. Though the importance of multicast communication is well recognized, the problem in its deployment lies in concerns over its requirements for fair amount of states information [6], and also the fact that it should be supported by routers from Internet Service Providers (ISPs). The researchers have divided opinions on the question whether multicast should be implemented at the IP layer or at the end systems [22], [11], [8].

The lack of support for IP multicast in today's Internet has been a major road block for building many multicast applications. In fact, the development of ALM protocol can be entirely attributed to the above scenario. There have been fair amount of research to leverage application layer multi-casting (ALM) [30], [13], [12] for reducing transmission latencies in P2P applications [27], [7]. ALM is implemented on an overlay network which uses Pastry [20], Chord [24], CAN [19] or

Tapestry [33] for routing and location schemes. Since, the physical network is hidden, ALM increases delay in deliver packets compared to IP multicast. It will also likely to increase the stress<sup>1</sup> on Internet (core) links as a node's neighbor may not necessarily be its topological neighbor in the underlying physical network. Furthermore, multicast in overlay networks is implemented by maintaining trees. It treats all nodes having equal power and same bandwidth. The root of the tree can potentially be subject to overload and may become a single point of failure.

In Internet, IP multicast is deployed in a limited scale on LANs, and by a few ISPs [31]. We believe IP multicast can be leveraged in content delivery not only to reduce transmission latency but also to reduce stresses on internet links. It can lead to a win-win situation for ISPs as well as individual enterprises. With the reduction in stresses on the core links, ISPs can add more subscribers. On the other hand, the subscribers could substantially bring down the cost for repetitive data transfers by supporting IP multicast functionality at the access links.

Keeping the performance improvement of IP multicast over unicast in view, in this paper we propose a new content distribution protocol called *MTorrent*. It co-exists with the standard BitTorrent [10] protocol, and exploits the IP Multicast functionality, wherever available, to eliminate repetitive data transmissions. *MTorrent* uses a unique combination of IP multicast and unicast for torrent download. It does not rely on IP multicast in the Internet links. Any packet which is to be downloaded from the Internet outside a LAN (or a local island) is received by unicast. Since the local islands are likely to be under the control of a single administrative authority, multicast functionality can be turned on for these networks. This way IP multicast can be leveraged only over the access links to optimize stresses on the core links. *MTorrent* protocol has been implemented and extensive experiments were carried out over the network testbed provided by Emulab [28] to find out the performance improvement over BitTorrent.

Initially, experiments were carried out on Emulab environment to determine the effect of data repetitiveness in BitTorrent and WWW protocols. It was found that on most commonly configured network topologies, BitTorrent suffers from severe data repetitiveness. On the basis of this key observation, we designed *MTorrent*. The key features of *MTorrent* include the simplicity of its design and inter-operability with other BitTorrent clients. Furthermore, it does not require any modification on the tracker part of the BitTorrent protocol. Both the protocols are compared with respect to data repetitiveness. The results of the experiments indicate that *MTorrent* reduces

Piyush Agrawal is with the Department of Electrical Engineering, Stanford University, CA 94305, USA. Email: piyushag@stanford.edu

Hitesh Khandelwal is with the Department of Computer Science, Purdue University, IN 47907, USA. Email: hitesh@purdue.edu

R. K. Ghosh is with Indian Institute of Technology, Department of CSE, Kanpur 208016, India. Email: rkg@cse.iitk.ac.in,

<sup>1</sup>Stress is defined by the ratio of number of total packets and number of unique packets traversing over a link

the download time by 44%, load on Internet links by 65%, and download of redundant data packets by as much as 40%.

This paper is organized as follows. Section II deals with related work. Section III explains the problem of data repetitiveness. Section IV presents the details of experimental setup over Emulab. Section V deals with implementation of content distribution models, and provides the motivation behind our new protocol MTorrent. Section VI deals with the design and implementation of MTorrent, and its comparisons against BitTorrent as well as WWW-based content distribution models. The paper concludes with section VII.

## II. RELATED WORK

P2P traffic is found to be highly repetitive. In absence of IP multicast functionality, caching [15] has been used reduce data repetitiveness in transmission. There have been good deal of research highlighting the importance and feasibility of caching P2P traffic [15], [14]. Also most of the current P2P protocols are not ISP-friendly, because they impose unnecessary traffic on the Internet links [14]. Some researchers suggest deploying smart caching techniques or making P2P protocols locality-aware [29], [21].

In a P2P system, peers collaborate to form a distributed system for the purpose of exchanging data. A file that one peer downloads is often made available for upload at other peers. In 'pure' P2P networks:

- Peers act as equals, merging the roles of client and server
- There is no central server managing the network
- There is no central router

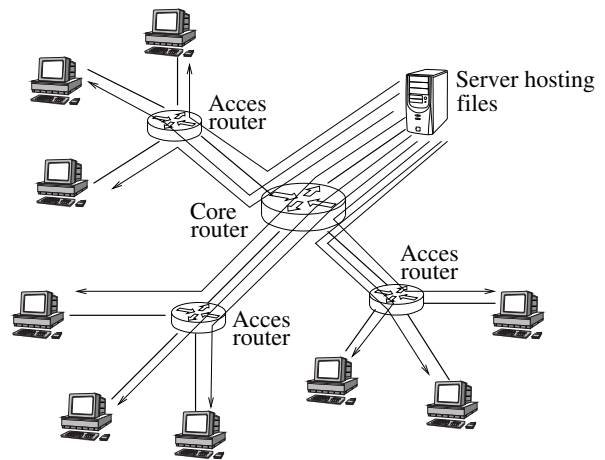
Some examples of pure P2P application layer networks designed for file sharing are Gnutella and Freenet [9]. There are also countless hybrid P2P systems, which share the following characteristics:

- A central server that keeps information of peers and responds to requests for that information.
- Peers are responsible for hosting available resources (as the central server does not have them), for letting the central server know what resources they want to share, and for making its shareable resources available to requesting peers.

A popular example of hybrid peer-to-peer application layer network is the BitTorrent protocol.

BitTorrent uses the tit-for-tat method for seeking *pareto* efficiency. With BitTorrent based file downloads, when multiple clients concurrently download a file, they upload pieces of this file to each other, redistributing the cost of uploads among the downloaders. Although BitTorrent is a successful P2P protocol, it has following notable drawbacks:

- For small files, BitTorrent tends to show higher latency and overhead.
- Though several downloaders, downloading a file, might be located physically close to each other (e.g. several clients on a LAN downloading a software patch), the tracker returns a random list of peers to which a new downloader should connect. This leads to wastage of resources because of redundant downloads of same pieces by peers close-by.



**Fig. 1:** Concurrent downloads cause heavy load on server bandwidth and network resources

To counter the problem of geographically distant peers exchanging data with other peers in BitTorrent protocol, a location-aware BitTorrent protocol has been proposed [32]. However, the proposal is in a very loose form with no real world implementation or performance results.

## III. REPETITIVE TRANSMISSION OF DATA

Consider the scenario depicted by figure 1. A file server hosts a file to be downloaded by 9 clients. It is connected to a core router which in turn is connected to three other access routers. Each client is connected to one of these access routers.

A client establishes an independent TCP connection to the file server to fetch the file. If all the clients need to download the file at the same time, nine parallel TCP connections will be sourced at the file server. The server should open 9 different sockets to serve each of above TCP connections. Each core router in turn sends 3 copies of the same data on each of the access links to clients.

Now imagine the scenario when the number of interested clients increases from nine to say around a few hundreds. This is a very common scenario, specially when new files (like movies) are installed for hosting on websites or critical security patches are made available by the software developers. The problem of *data repetitiveness* due to attempts of multiple clients to download the same content leads to significant drop in download speed for individual clients.

## IV. EXPERIMENTAL SETUP

In order to examine the problem of repetitive data transmission in commonly used content distribution models like WWW and BitTorrent, which are used over the Internet, we designed an experimental set up over Emulab<sup>2</sup>. Emulab provides integrated access to a wide range of experimental environments, including Emulations, Live-Internet Experimentations and also Simulations.

<sup>2</sup><http://www.emulab.net>

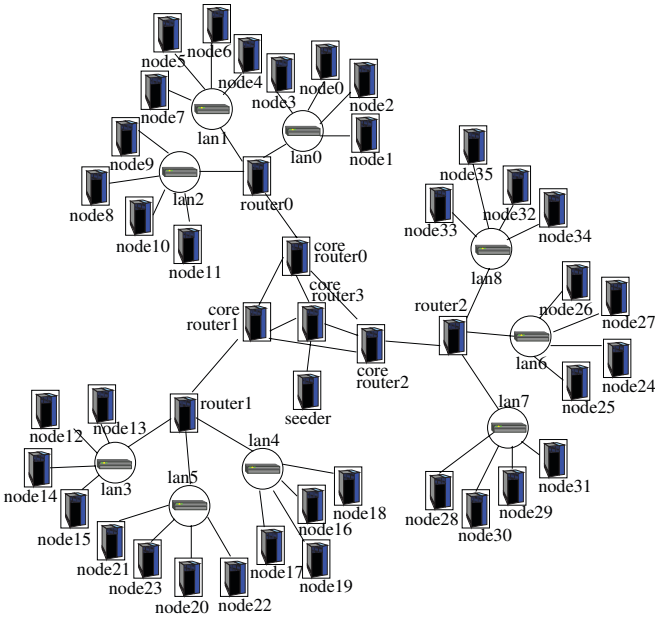


Fig. 2: The network topology used for the performance study

### A. Network Topology

MTorrent is one of the few proposed extensions to BitTorrent protocol which was actually implemented and tested on real nodes. We preferred doing a moderate scale emulations (real experiments) over large scale simulations using NS-2 [3]. This approach exposed a number of issues which originate while building real systems.

The Internet can be assumed to be composed of two entities:

- 1) *Backbone Network*: It consists of the high bandwidth, high delay, long distance network links, which typically run across continents and countries. These backbone links are generally hosted by various ISPs and account for the major cost in running the Internet.
- 2) *High Speed LANs*: Most organizations today have access to high speed LANs which in turn are connected to the backbone Internet via routers. LANs are generally error-free, congestion free and administered by the local organizations.

The two entities described above are connected through access links between the access routers which are part of high speed LANs with the core routers present in Backbone network. These access links are maintained by ISPs who charge their customers for data transmission over these links.

Figure 2 illustrates the network topology we used for studying performance of the content distribution models on Emulab. A total of 200 nodes were available on Emulab for any experiment. In reality we were able to grab just about 40-48 nodes by writing a script (a practice very strongly discouraged by Emulab). So, any experiment with large number of nodes or a sufficiently large representative topology was not possible under Emulab environment. We wanted the experimental topology that has at least three LANs with one not having IP multicast functionality. This explain why we settled for a sythetically constructed topology consisting of just 44 nodes in our experiments in Emulab environment.

The Internet backbone is made up of four core routers, named as *coreRouter0*, *coreRouter1*, *coreRouter2* and *coreRouter3*. Each of the core routers run the Red Hat Linux 9.0 standard operating system. The four core routers are connected to each other in a symmetrical manner. The links between these core routers are named as *corelink0*, ..., *corelink5*. Similarly, the links between core routers and the access routers are named as *link0*, ..., *link3*. The correspondence of the links with their respective names and labels are as shown in Table I. Note that labels are, odd numbers, for representing data transfer in forward direction over the respective links. Since, all links are symmetrical, data transfer also takes place in backward direction. To represent traffic in backward direction on a link  $l$  we represent the same by label  $l + 1$ .

TABLE I: Mapping of links with names and labels.

Links	Connecting nodes	Label	Flow of traffic
coreLink0	(coreRouter0, coreRouter1)	1	forward
	(coreRouter1, coreRouter0)	2	backward
coreLink1	(coreRouter2, coreRouter1)	3	forward
	(coreRouter1, coreRouter2)	4	backward
coreLink2	(coreRouter0, coreRouter2)	5	forward
	(coreRouter2, coreRouter0)	6	backward
coreLink3	(coreRouter0, coreRouter3)	7	forward
	(coreRouter3, coreRouter0)	8	backward
coreLink4	(coreRouter1, coreRouter3)	9	forward
	(coreRouter3, coreRouter1)	10	backward
coreLink5	(coreRouter2, coreRouter3)	11	forward
	(coreRouter3, coreRouter2)	12	backward
link0	(coreRouter0, router0)	13	forward
	(router0, coreRouter0)	14	backward
link1	(coreRouter1, router1)	15	forward
	(router1, coreRouter1)	16	backward
link2	(coreRouter2, router2)	17	forward
	(router2, coreRouter2)	18	backward
link3	(coreRouter3, seeder)	19	forward
	(seeder, coreRouter3)	20	backward

Each of the core link is a 10 Mb link with a 20 ms end-to-end delay and a Drop Tail queue. For convenience in showing the performance plots, the links are represented by their labels instead of names. Three of the core routers (*coreRouter0*, *coreRouter1* and *coreRouter2*) are each connected to a set of three high speed LANs via routers (*router0*, *router1* and *router2*). Each of the three routers run on FreeBSD 6.0. The link between a router and a core router is a 2 Mb link with a 10 ms end-to-end delay and a Drop Tail queue. Each router is in turn connected to three 10 Mbps LANs (for example, *router0* is connected to *lan0*, *lan1* and *lan2*). Each LAN is composed of 4 end nodes and a switch. The nodes are named from *node0* to *node 35* (total 36 end-nodes/clients).

A dedicated node (named *seeder*) is connected to *coreRouter3* via a 2Mb link with a 10 ms end-to-end delay and a Drop Tail queue. This node is used for initially hosting the file which is to be distributed.

### B. Performance Metrics

We measure the following two key metrics in each experimental run, for each link, in each direction:

**TABLE II:** Link Statistics for file download using BitTorrent

Link	Flow direction	No. of Bytes	Stress
coreLink0	Forward	4.1 MB	5.712
	Backward	3.6 MB	6.078
coreLink1	Forward	3.9 MB	5.429
	Backward	3.1 MB	5.896
coreLink2	Forward	2.8 MB	6.368
	Backward	4.0 MB	5.221
coreLink3	Forward	15 KB	2.800
	Forward	0.8 MB	1.544
coreLink4	Forward	16 KB	3.620
	Backward	1.2 MB	1.855
coreLink5	Forward	15 KB	2.880
	Backward	0.9 MB	1.366
link0	Forward	8.5 MB	6.630
	Backward	6.9 MB	7.013
link1	Forward	9.3 MB	7.155
	Backward	6.7 MB	7.300
link2	Forward	6.8 MB	6.516
	Backward	8.0 MB	6.933
link3	Forward	47 KB	3.184
	Backward	3.0 MB	2.797

- 1) *Number of Bytes*: It represents the raw amount of data transferred over a link in a particular direction.
- 2) *Stress*: It represents the ratio of the number of total packets transmitted over a link against the number of unique packets transmitted over the link.

## V. IMPLEMENTATION OF BITTORRENT ON EMULAB

For our experiments on BitTorrent, we needed a client which supports at least the following features:

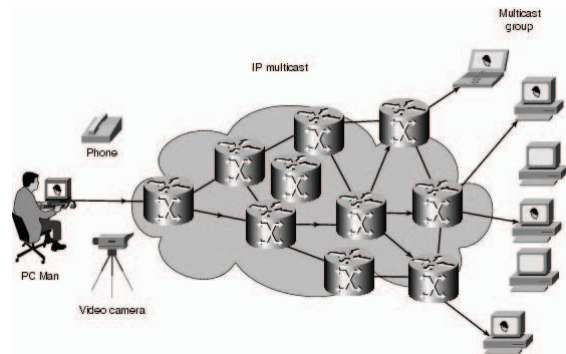
- A console based interface to allow remote execution over Emulab nodes.
- Implemented preferably in C/C++, so that BSD socket could be used to extend for supporting IP multicast functionality.

Many exhaustive comparisons of different BitTorrent implementations are available [17], [25], [5]. We found CTorrent [1] to be a light weight BitTorrent client, written in C++ and being actively supported by developer community. So, we based our experiments with the P2P model on CTorrent.

For each of the content distribution models, we initially hosted a 1 MB file on the *seeder* node, which is then downloaded by all 36 end nodes, using one of the models. The evaluations for each model is presented below. It shows several important trends. Most notable among these

- All the links participate in data transfers to the order of 4-6 MB. But data transfer occurs in both directions (uplink and downlink).
- The other important observation is regarding the link stress. The link stress values are smaller in the case of core links. This means that there are less number of duplicate packet transmissions over the Internet links.

Table II show that the Internet links suffer from non-ideal stress values. Since the coordination (if any) among the downloading clients is at a very high level (application layer), redundant downloads of packets do happen. This is particularly the case when several downloading clients are located on the



**Fig. 3:** Multicast transmission sends a single multicast packet addressed to all intended recipients (Source: [2])

same or near-by LANs, and individual clients unaware of their physical proximity. Consequently, in the case of BitTorrent, although the clients collaborate, they may be separated from one another by several Internet backbone links.

## VI. MTORRENT

We propose to augment the BitTorrent protocol with a simple yet efficient coordination protocol, through which redundant downloads of pieces in BitTorrent could be avoided at the level of local islands or LANs, thus saving network resources and speeding the download process.

In particular, we utilize the IP Multicast functionality available in some of the networks to share the pieces downloaded by BitTorrent with the local peers, instead of letting them download those pieces from the remote peers located in different islands. The underlying idea is that the design of the protocol should be such that it can be deployed on LANs and be interoperable with BitTorrent without requiring any extra support for traffic over core links.

We could have used an application level multicast (ALM) mechanism instead of relying on LAN being enabled with IP multicast functionality. Our decision in this regard are influenced by the following reasons. ALM is usually implemented on an overlay network which relies on efficient implementation of distributed hash table [20], [24], [19], [33] for routing and location schemes. Also explained at the beginning, ALM increases not only the latency for packet delivery, but also the stresses on core links. The tree for multicast tree maintained by the underlying ALM mechanism could also be a source of the single point failure.

We believe a LAN administrator will become favorable to idea of enabling IP multicast functionality when they realize the benefit of avoiding data repetitiveness in the Internet traffic. Therefore, for the simplicity in protocol implementation, we have assumed LANs to have IP multicast functionality.

Figure 3 demonstrates how the data from one source is delivered to the several interested recipients using IP Multicast. All the clients can initially send IGMP request messages to join a multicast group, and the source (or the seeder) multicasts the data on this group. Since the routers are aware of the physical topology and the positions of the clients, the data traverses through shortest paths to reach the clients,

guaranteeing optimal download time and stress of 1 on the links. Although such an approach is promising, it is not viable in today's Internet because of lack of support for IP Multicast on Internet.

In section IV, we observed that bulk data download happens using P2P systems and there are several instances where multiple clients downloading the same contents exist within the same island (a collection multicast enabled LANs), for example, an enterprise, or a university network. These clients are unaware of each others presence, and fetch data packets from outside the island over BitTorrent. We propose to use IP Multicast as an enabling technology to anonymously share downloaded packets with the local clients on the same island. This forms the basis of our proposed protocol, which we call *MTorrent*. It leads to a win-win situation for all stake holders. The users experience improvements in download time. The LAN administrators see twin improvements in terms of users satisfaction and a reduction in data repetitiveness in downloads. The ISPs witness improvements in terms of the volume of traffic due to user satisfaction.

#### A. Overview

We associate a class D IP Multicast address with each file to be downloaded. This IP address can be embedded in the torrent file available on the web server. The BitTorrent client reads information available in the torrent file, which comprises of the URL of the tracker, the piece size, checksums of the pieces, etc. We have modified the client to also read the embedded class D IP address.

Conceptually, the class D IP address associated with a file is the multicast address of the group of clients who are interested in this file. Before the download starts, each client tries to join the multicast group by sending IGMP messages in its own island. If the network to which the client is connected is IP Multicast enabled, the client successfully joins the group, and thus becomes a part of the island. Otherwise, the client stays as a isolated island, which means it does not receive or send data via IP multicast.

Note that the class D address is used by all the clients in different islands, which are connected to each other by normal unicast links, without any support for IP Multicast. Due to this, several groups may be formed on the Internet, who share the same class D IP address.

In *MTorrent*, each client participates in the normal download process using unicast. As soon as a piece has been successfully downloaded over unicast by a client, it multicasts the piece over the local multicast group (corresponding to this file). Since other clients on the same island must have also joined this group, they receive the pieces on the multicast socket, in addition to the normal pieces on the unicast socket.

When some client is in the process of downloading a file, it is very likely that other clients may also be anticipating the pieces of the same file over unicast links at the same time. Each client has a helper thread, independent of original BitTorrent client thread, which periodically checks for the receipt of any multicast packets. If the helper thread detects that a piece is now available over the multicast socket, it tries to

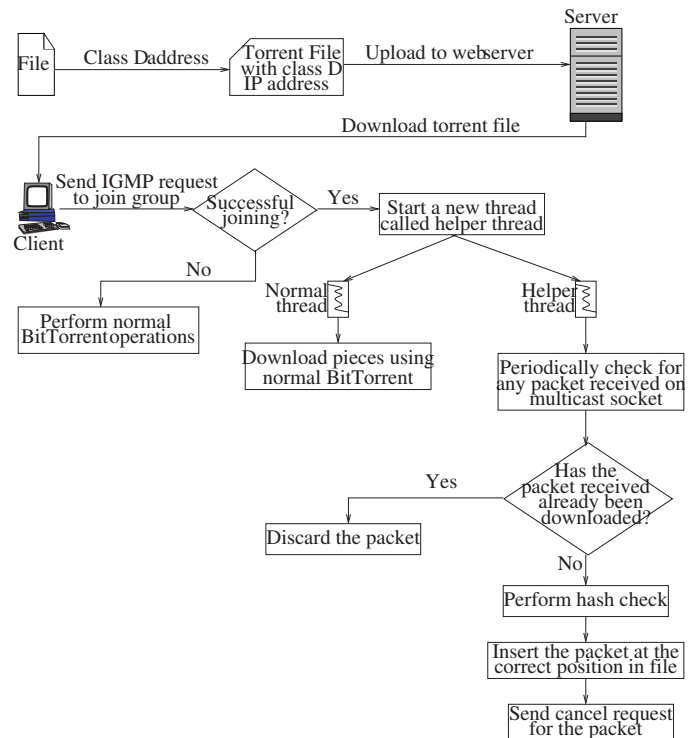


Fig. 4: Flow chart showing *MTorrent* operations

verify the integrity of the piece by calculating and comparing the checksum with that read from the torrent file. If the checksum matches, the helper thread places the received piece at its correct location on the disk and updates the local data structures to indicate that this piece need not be downloaded by unicast again. The helper thread also immediately issues cancellation of all requests for the received piece which may have been sent to other clients over the unicast link to avoid receiving the same piece from other peers by unicast.

Figure 4 summarizes the sequence of operations performed while uploading the torrent file on web server and while downloading of file by a *MTorrent* client.

One problem with IP Multicast is that unlike TCP, it is an unreliable protocol which works over UDP. Since IP Multicast does not have any mechanisms for rate control and checking packet losses (due to random errors etc.), it is not necessary that pieces shared over multicast by clients would be received by *all* other clients on the island. The clients which have small receive buffers or which are busy with other operations often are unable to completely receive packets sent over multicast.

However, a simple observation makes the operation of *MTorrent* much easier. Since it has been designed to be a module completely independent from the standard BitTorrent, the clients which are unable to successfully receive multicast pieces can subsequently receive them using the normal unicast connections. Furthermore, as the fraction of the clients which are unable to receive multicast pieces is very small, the benefits obtained from multicast packets more than offsets the cost of fetching some packets over unicast. Consequently the overall download process becomes much more efficient and cost effective.

One criticism about our experiment could be it subsumes a structured peer-to-peer dynamics by assuming that many peers will be interested to download the same file at the same time. However, in an environment typical to ours, this criticism does not stand. Indeed, we found a sizeable number of peers online at the same time during weekends on our LAN. This structuredness in behavior with respect to internet traffic at our LAN is primarily due to the fact that ours is fully residential institute. The students stay in dorms which are provided with LAN connection. They get free time for Internet browsing during weekends. Our logic about extension of structured behavior is that the area served by an ISP is much larger than the institute LAN. So, P2P dynamics would perhaps exhibit above structured behavior more often than not in a 24-hour clock.

### B. Experimental Issues

We evaluate the performance of MTorrent on a topology, same as that used in section IV. All the LANs in each island are connected to each other via the access router (i.e., router0, router1 or router2). Each of the access routers run the FreeBSD 6.10 operating system. In order to allow IP Multicast across different LANs on the same island, we run `mrouterd` [26] on each of the access routers.

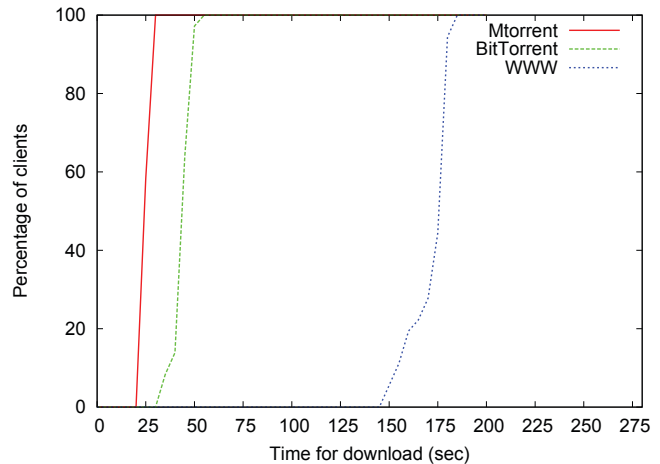
The `mrouterd` utility is an implementation of the Distance-Vector Multicast Routing Protocol (DVMRP), an earlier version of which is specified in RFC-1075 [16]. It maintains topological knowledge via a distance-vector routing protocol (like RIP, described in RFC-1058 [4]), upon which it implements a multicast datagram forwarding algorithm called Reverse Path Multicasting.

The `mrouterd` utility forwards a multicast datagram along a shortest (reverse) path tree rooted at the subnet on which the datagram originates. The multicast delivery tree may be thought of as a broadcast delivery tree that has been pruned back so that it does not extend beyond those subnetworks that have members of the destination group. Hence, datagrams are not forwarded along those branches which have no listeners of the multicast group. The IP time-to-live of a multicast datagram can be used to limit the range of multicast datagrams. Thus, any multicast packet in one of the LANs reach all other LANs on the same island, provided their are clients on the other LANs who have subscribed to the corresponding multicast group.

Also, we set the *TTL* value of multicast packets to 3 to allow them to cross multiple levels of multicast enabled routers. Note that a *TTL* value of 1 means that packets are limited to the same subnet.

### C. Performance Metrics

As already explained, there are two types of links, namely, (i) *core links*: which serve the traffic across the Internet by connecting the core routers; and (ii) *access links*: which are used to provide Internet access to the islands consisting of various high-speed LANS. Since the traffic patterns on two types of links are of interest to separate stake holders namely, ISPs and Internet users, we show the evaluation of these



**Fig. 5:** Cumulative distribution function of download time for each client

patterns separately. Apart from the two metrics number of bytes transferred and the stress, which have been already introduced in section IV-B, we also record time to download files for each client.

### D. Comparison of MTorrent, BitTorrent and WWW

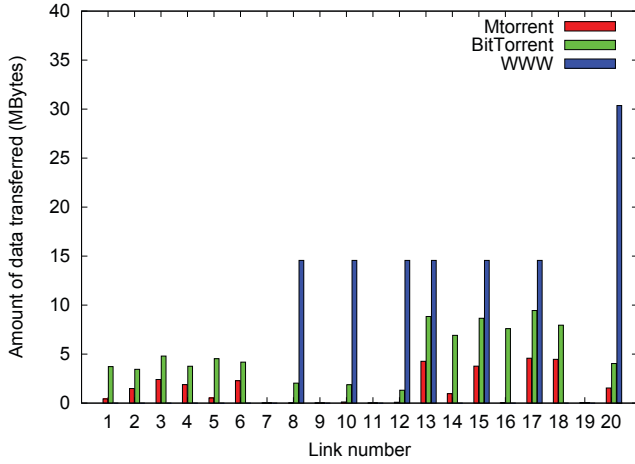
For all the experiments in this section, we run a seeder and a tracker at the seeder node (connected to `coreRouter3` in the topology shown in Figure 2. The seeder serves a file of size 1.2 MB. All the results reported in this section have been obtained after proper averaging over 5 to 10 runs of each experiment.

Figure 5 shows the *Cumulative Distribution Function (CDF)* of the time for download at each client. X-axis shows the time for download in seconds while Y-axis represents the cumulative percentage of clients which completed their download till this time.

Figure 5 shows that 100% of clients complete their download within 30 seconds by using MTorrent protocol. In contrast, it takes about 60 seconds for all the nodes to complete their download using BitTorrent. Both MTorrent and BitTorrent distribute download load on several links, and the clients collaborate with each other in fetching data. Furthermore data packets fetched from outside the island are instantaneously shared with other clients in the same island via IP Multicast in the case of MTorrent. Consequently the wait time for each packet is reduced considerably.

The performance of WWW protocol is the worst. It takes about 180 seconds for all nodes to complete the download.

WWW model is not relevant to our experiments. But, initially, in order to assess the importance of torrent downloads, we carried some experiments to compare the performance of BitTorrent against the WWW protocol. Since, the results of those experiments are already available to us, we have included them here for the sake of completeness. It also provides a three-way performance comparison among the protocols. To download data using WWW model, we used GNU `wget` ([www.gnu.org/software/wget](http://www.gnu.org/software/wget)). Since `wget` is a non-interactive



**Fig. 6:** Amount of data transferred over each link using MTorrent, BitTorrent or WWW

command line tool, it could be called easily from scripts, cron jobs, terminals without X-window support.

**TABLE III:** Download time (in seconds) statistics.

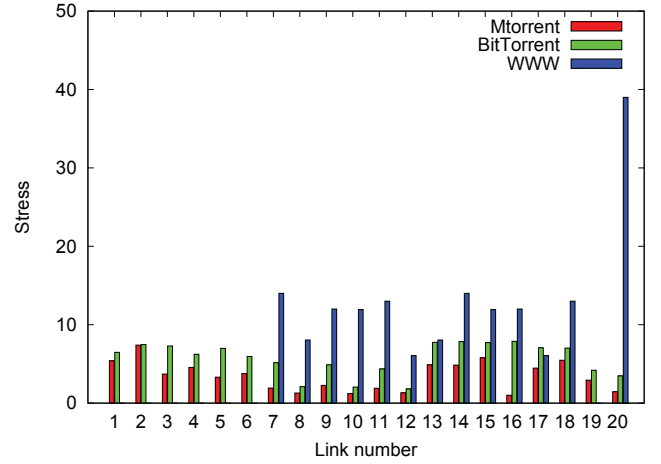
	Minimum	Maximum	Average
MTorrent	22.5	26.3	24.6
BitTorrent	34.2	51.0	43.4
WWW	147.4	181.2	171.2

Table III reports the minimum, maximum and the average download times over all the 36 clients using WWW, BitTorrent and MTorrent protocols. Based on the reported values, we conclude that the improvement in time for download of files using the proposed MTorrent protocol is about 44% over the conventional BitTorrent protocol and about 86% over the WWW HTTP protocol.

Figure 6 shows the amount of data transferred over each of the links using MTorrent, BitTorrent and the WWW protocol. From Figure 6, it is clear that the amount of data transferred over the access links is much higher as compared to the core links. Also, in case of WWW protocol, some of the links observe zero utilization because in that case, each client has a fixed single route to the seeder, and some of the links do not lie on any of such paths. Also, in case of WWW protocol, there is a clear difference in the utilization of up/down links due to the asymmetric nature of download.

All the links have to carry comparatively lesser amount of data in the of MTorrent protocol as opposed to either BitTorrent or WWW. This is explained by the fact that in MTorrent, packets *once* downloaded in an island, normally need not be downloaded again from the Internet. Thus, the amount of bytes transferred over Internet links is lower.

Figure 7 shows the stress observed over each link using MTorrent, BitTorrent and the WWW protocol. It is clear from this figure that the stress on each of the links is much lower with MTorrent protocol, as compared to that with either BitTorrent or the WWW protocol. This proves that MTorrent is successful in reducing the duplicate download of packets over



**Fig. 7:** Stress on link using MTorrent, BitTorrent or WWW.

the Internet links by effectively sharing downloaded packets with local clients.

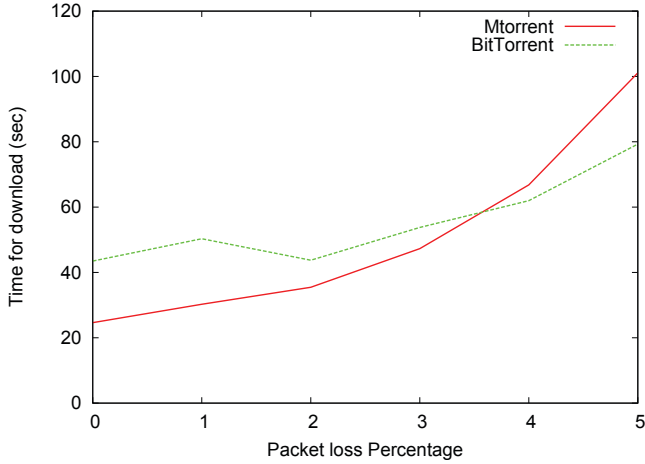
Table IV shows the total amount of data transfers over all type of links as well as the respective stress statistics obtained using MTorrent, BitTorrent and WWW protocols. It reports minimum, maximum and average over all the 20 links.

**TABLE IV:** Volume of data transfers and stress statistics on various links using different protocols.

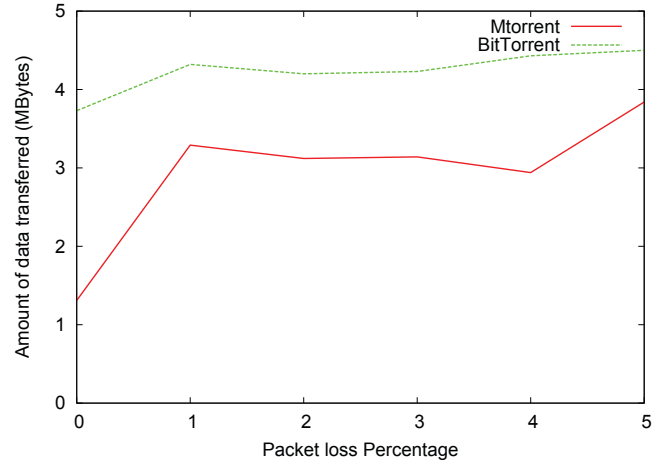
Protocol Type	Data transfer (in sec)			Stress statistics			
	Min	Max	Avg.	Min	Max	Avg.	
BitTorrent	Core	0.01	4.8	2.4	1.8	7.4	5.0
	Access	0.4	9.4	6.6	3.4	7.8	6.6
MTorrent	Core	0.007	2.4	0.7	1.2	7.3	3.1
	Access	0.01	0.4	2.4	1	5.7	3.8
WWW	Core	0	14.5	3.6	0	14	5.4
	Access	0.001	30.3	9.2	0	39	13.0

The amount of data transferred over core links indicates the traffic which ISPs need to support while traffic over the access links is important for the LAN administrators. The file being downloaded by the clients is of size 1.2 MB. Therefore, in the case of MTorrent, on an average, each access link transfers about 2.4 MB of data to serve 12 concurrent downloading clients on the corresponding island. This means only two copies of the original file are downloaded by all 12 clients collectively. As opposed to this, each access link, on an average, transfers about 6.6 MB of data using BitTorrent. It means approximately 5 copies of the same file are downloaded in each island. The performance of WWW is the worst with around 8 copies of the original file getting downloaded in each island. Based on the statistics shown in Table IV, we conclude that an improvement of about 65% over BitTorrent and about 75% over WWW protocol in amount of data transferred is achievable using the proposed MTorrent protocol.

The last three columns of Table IV show the stress statistics on all types of links using the 3 protocols. The minimum stress achieved on an access link is as low as 1 using MTorrent protocol, which is the theoretical lower bound on stress that



**Fig. 8:** Time for download with packet loss percentage in each LAN.



**Fig. 9:** Average amount of data transfers with packet loss % in each LAN.

any link can have. Thus, in one of the islands (island containing LAN0, LAN1 and LAN2 in topology shown in figure 2), exactly one copy of the original file was downloaded using the proposed MTorrent protocol, which is the *best performance achievable by any content distribution protocol!*. Based on our results (in Table IV) we conclude that MTorrent is about 40% more effective over BitTorrent and about 60% more effective over WWW protocol in reducing the redundant traffic load on Internet links.

#### E. MTorrent and Multicast Unreliability

As discussed in section VI-A, there is no guarantee that multicast packets reach their destinations, the effectiveness of MTorrent, which shares packets with local clients via IP Multicast needs to be examined. Multicast packets on an island can be lost or delayed due to following reasons:

- The clients and links on a LAN show abnormal behaviour (due to load or misconfiguration) leading to random packet losses.
- There is congestion on the LANs due to other ongoing heavy traffic exchanges among the clients, e.g., VoIP etc.

**Effect of random link losses:** To model the random behavior of the clients and the links, a random packet loss module is installed in each of the LANs, whose packet loss rate can be configured. The packet loss rate for each LAN was varied in the range 0–5%, and one experiment was performed for each percentage unit to measure the three metrics mentioned earlier in section VI, separately for MTorrent and BitTorrent.

Figure 8 shows the average time for download over all the clients with increasing values of packet loss percentage. As observed earlier, at 0% packet loss (no artificial random packet losses injected), MTorrent completes the download much faster than BitTorrent. However, some multicast packets might still be lost due to receiver buffer overflow, etc. As the random packet loss percentage increases, the effectiveness of sharing packets via IP Multicast decreases. As consequence, the failed packets have to be obtained by the clients via normal

unicast procedure. Thus, the time for download in the case of MTorrent increases with increasing packet loss percentage. However, the download time in the case of BitTorrent also increases with random losses because even unicast packets are dropped; and hence more TCP retransmissions occur resulting in increasing the download time.

At packet loss percentage of more than 4%, the time for download using MTorrent is higher than that using BitTorrent. This is because at such high loss rates, most multicast packets are lost and thus, these packets have to be fetched using normal unicast procedure. In addition, the module which sends and receives multicast packets incur extra overhead on the client, which exceeds any possible gains in time for download due to the successful multicast packets received. However, *random* packet loss percentages as high as 4% are quite rare in most LANs today and thus represent an unnatural scenario.

Figure 9 shows the average amount of data transferred on all Internet links with increasing values of packet loss percentage. The amount of data transferred increases in case of both MTorrent as well as BitTorrent. Still, MTorrent performs better than BitTorrent by downloading lesser amount of data from Internet.

Finally, Figure 10 shows the variation of average link stress with packet loss percentage. The stress on the Internet links increases with random packet loss due to the higher number of TCP retransmissions to deliver data across islands. Note that more retransmissions mean same data packets traversing Internet links again and again.

**Effect of congestion:** To model the scenario of congestion in each island, we start a Constant-Bit Rate (CBR) traffic source on each of the LANs which send the traffic to one of the clients on another LAN in the same island. Thus, each island has 3 CBR traffic sources. The rate of CBR traffic for each source is varied from 0 Mbps to 10 Mbps to model the severity of congestion.

Figure 11 shows the variation of average time for download over all clients with increasing value of CBR traffic rates. Although MTorrent consistently performs the download faster

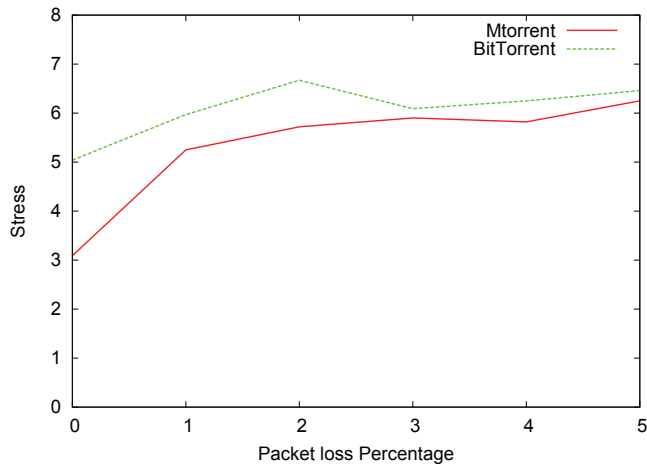


Fig. 10: Link stress with packet loss % of each LAN

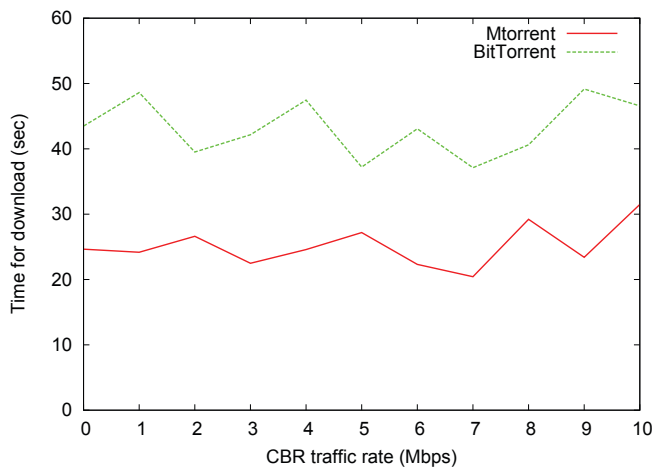


Fig. 11: Time for download with varying congestion level

than BitTorrent even at high levels of congestion, we do not observe any clear dependence of congestion level on the effectiveness of IP Multicast and thus MTorrent.

## VII. CONCLUSIONS

In this paper we studied the problem of repetitive data transmission over Internet in widely used content distribution protocols and proposed a new protocol called MTorrent which leverages IP multicast to eliminate the data repetitiveness over Internet links. We obtained the following three important results with MTorrent:

- Reduction in download time of each client using MTorrent by 44% over BitTorrent and by 86% over WWW protocol.
- Reduction in traffic load on Internet links and access links (maintained by ISPs) by 65% and 75% over BitTorrent and WWW protocols respectively.
- Reduction in the wastage of resources like bandwidth due to redundant packet downloads by 40% over BitTorrent and 60% over WWW protocol.

Most ISPs today observe heavy traffic load on their Internet links due to increasing number of users using P2P file sharing systems. With more and more users migrating to a system like MTorrent, the load on ISP resources (Internet links) can be reduced by as much as 65% for the comparable amount of downloads by end clients. The load on access links is also reduced by similar proportions. It may be noted that all our experiments are with respect to small file size (1MB only). As the file size increases, the number of pieces will also increase. So, we expect the relative gain over BitTorrent by using MTorrent will be more pronounced.

Finally, our work on MTorrent is distinct from other similar research because of the following reasons:

- **Standard compliance.** The proposed MTorrent protocol is interoperable with BitTorrent protocol. It only requires changes at the client level, unlike other solutions, which would need network wide support and upgrades for both hardware and software.
- **Simplicity.** The protocol is easy to understand and implement, thus can be readily integrated with most of the widely used BitTorrent clients like Azureus and BitComet, etc.
- **Actual Implementation.** In place of theoretical results or network simulations, we resorted to actually implementing a prototype system of our protocol and have evaluated it on a large scale real network.

## REFERENCES

- [1] Enhanced CTorrent. [www.rahul.net/dholmes/ctorrent](http://www.rahul.net/dholmes/ctorrent).
- [2] IP multicast. [www.cisco.com/en/US/docs/internetworking/technology/handbook/IP-Multi.html](http://www.cisco.com/en/US/docs/internetworking/technology/handbook/IP-Multi.html).
- [3] The network simulator- ns-2. [www.isi.edu/nsnam/ns](http://www.isi.edu/nsnam/ns).
- [4] Routing information protocol. [www.ietf.org/rfc/rfc1058.txt](http://www.ietf.org/rfc/rfc1058.txt).
- [5] C. Herley A. Bharambe and V. N. Padmanabhan. Analyzing and improving a bittorrent networks performance mechanisms. In *IEEE INFOCOM, Barcelona, Spain, 2006*.
- [6] Kevin C. Almeroth. The evolution of multicast: From the mbone to interdomain multicast to internet2 deployment. *IEEE Network*, January 2000.
- [7] Miguel Castro, Micael B. Jones, Anne-Marie Lermarrec, Antony Rowstron, Marvin Theimer, Helen Wang, and Alec Wolman. An evaluation of scalable application-level multicast built using peer-to-peer overlays. In *Proceedings of IEEE INFOCOM 2003*, volume 2, pages 1510–1520, 30 March-3 April 2003.
- [8] Y. Chu, S.G. Rao, and H. Zhang. A case for end system multicast. *IEEE Journal on Selected Areas in Communication*, 20(8):1456–1471, 2002.
- [9] I Clarke, O Sandberg, B Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability, LNCS-2009*, pages 46–77, Berkeley, California, June 2001.
- [10] B. Cohen. Incentives build robustness in BitTorrent. In *Proceedings of the First Workshop on the Economics of Peer-to-Peer Systems*, 2003.
- [11] S. E. Deering. Multicast routing in datagram internetworks and extended lans. *ACM Transaction on Computer Systems*, 8:85–110, 1990.
- [12] Huaqun Guo, Lek Heng Ngoh, and Wai Choong Wong. Application-level multicast using DINPeer in P2P networks. In *Proceedings of 4th International Conference on Networking, Part II*, volume 3421 of *Lecture Notes in Computer Science*, pages 754–761, 2005.
- [13] J. Jannotti, D.K. Gifford, K.L. Johnson, M.F. Kaashoek, and Jr J.W. OToole. Overcast: Reliable multicasting with an overlay network. In *Fourth Symposium on Operating System Design and Implementation (OSDI)*, 2000.
- [14] T. Karagiannis, P. Rodriguez, and K. Papagiannaki. Should internet service providers fear peer-assisted content distribution. In *Proceedings of IMC*, pages 63–76, 2005.

- [15] N. Leibowitz, A. Bergman, R. Ben-Shaul, and A. Shavit. Are file swapping networks cacheable? characterizing P2P traffic. In *Proceedings of the 7th International WWW Caching Workshop*, 2002.
- [16] T. Pusateri. Distance vector multicast routing protocol. [www.ietf.org/rfc/rfc1075.txt](http://www.ietf.org/rfc/rfc1075.txt).
- [17] D. Qiu and R. Srikant. Modeling and performance analysis of BitTorrent like peer-to-peer networks. In *Proceedings of ACM SIGCOMM*, pages 367–378, New York, NY, USA, 2004. ACM.
- [18] Bob Quinn and Kevin Almeroth. IP multicast application: Challenges and solutions. <http://www.ietf.org/rfc/rfc3170.txt>, 2001.
- [19] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable contentaddressable network. In *ACM SIGCOMM*, pages 161–172, San Deigo, CA, 2001.
- [20] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, Heidelberg, Germany, 2001.
- [21] Osama Saleh and Mohamed Hefeeda. Modeling and caching of peer-to-peer traffic. In *ICNP '06: Proceedings of the 2006 IEEE International Conference on Network Protocols*, pages 249–258, Washington, DC, USA, 2006. IEEE Computer Society.
- [22] J. Saltzer, D. Reed, and D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):195206, 1984.
- [23] Kamil Sarac and Kevin C. Almeroth. Monitoring ip multicast in the internet: Recent advances and ongoing challenges. *IEEE Communication Magazine*, 43(10):85–91.
- [24] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM*, pages 149–160, San Deigo, CA, 2001.
- [25] Y. Tian, D. Wu, and K. W. Ng. Modeling, analysis and improvement for bittorrent-like file sharing networks. In *Proceedings of IEEE INFOCOM, Barcelona, Spain*, pages 1–11, 2006.
- [26] Bart Trojanowski. How to set up linux for multicast routing. [www.jukie.net/~bart/multicast/Linux-Mrouted-MiniHOWTO.html](http://www.jukie.net/~bart/multicast/Linux-Mrouted-MiniHOWTO.html).
- [27] V. Venkataraman, P. Francis, and J. Calandrino. Chunkyspread: Multitree unstructured peer-to-peer multicast. In *The 5th International Workshop on Peer-to-Peer Systems*, 2006.
- [28] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. An integrated experimental environment for distributed systems and networks. volume 36, pages 255–270, New York, NY, USA, 2002. ACM.
- [29] A. Wierzbicki, N. Leibowitz, M. Ripeanu, and R. Wozniak. Cache replacement policies revisited: the case of P2P traffic. In *Proceedings of IEEE International Symposium on Cluster Computing and the Grid (CCGrid)*, pages 182–189, 2004.
- [30] Xiaotao Wu, Krishna Kishore Dhara, and Venkatesh Krishnaswamy. Enhancing application-layer multicast for p2p conferencing. In *4th IEEE Consumer Communications and Networking Conference CCNC 2007*, pages 986–990, 2007.
- [31] B. Zhang, S. Jamin, and L. Zhang. Universal ip multicast delivery. In *Proceedings of the International Workshop on Networked Group Communication (NGC)*, 2002.
- [32] Lie Zhang. Study of the location-awareness in BitTorrent-like Networks. In *Proceedings of 7th International Conference on Computer-Aided Industrial Design and Conceptual Design, CAIDCD'06*, pages 1–7, November 2006.
- [33] Ben Y. Zhao, Ben Y. Zhao, John Kubiatowicz, John Kubiatowicz, Anthony D. Joseph, and Anthony D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical report, University California, Berkeley, 2001.