Hybrid Content Distribution Network with a P2P Based Streaming Protocol

Saumay Pushp[1] and Priya Ranjan[2]

[1] Dept. of Computer Science and Engineering, Kanpur Institute of Technology, Kanpur, UP-208001, India
saumaypushp@gmail.com
[2] Dept. of Electrical Engineering, Indian Institute of Technology, Kanpur, UP-208016, India
ranjanp@iitk.ac.in

Abstract. Multicast has been used as a one-to-many approach to deliver information; it is based on the idea that if one packet of data should be transmitted to several recipients, the information should be sent by the origin just one time. In this paper, we propose the use of IP based Pragmatic General Multicast (PGM) to distribute content and to make distribution more efficient; we combine it with a P2P approach. We focus on the problem of data redundancy (at each node), congestion and contention and show how severely it impacts the network economics and the experience of end-user and hence leads to low traffic load and redundancy.

Keywords: Multicast, Peer to Peer, Congestion, Contention.

1 Introduction

Since 20 years, internet has seen an exponential increase in its growth. With more and more people using it, efficient data delivery over the internet has become a key issue. Peer-to-peer (P2P)/efficient data sharing based networks have several desirable features for content distribution, such as low costs, scalability, and fault tolerance. While the invention of each of such specialized systems has improved the user experience, some fundamental shortcomings of these systems have often been neglected. These shortcomings of content distribution systems have become severe bottlenecks in scalability of the internet. The need to scale content delivery systems has been continuously felt and has led to development of thousand-node clusters, global-scale content delivery networks, and more recently, self-managing peer-to-peer structures. These content delivery mechanisms have changed the nature of Internet content delivery and traffic. Therefore, to exploit full potential of the modern Internet, there is a requirement for a detailed understanding of these new mechanisms and the data they serve. In this work, we focus on the problem of redundancy of data being transmitted using several state of the art content distribution systems and show how severely it impacts the network economics and the experience of end-user. We base our findings on real world large scale measurement studies conducted over Emulab, which is a network test bed hosted by the University of Utah.

I. Tomkos et al. (Eds.): BROADNETS 2010, LNICST 66, pp. 40–54, 2012.

1.1 The Problem of Data Redundancy

Consider the scenario shown in Figure 1**.** The network topology contains a file server which hosts a file to be downloaded by 9 clients. The file server is connected to a core router which is then connected to three other access routers. All the clients are connected to the access routers.

Each 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 with file server as the source have to be started. This means that the server opens 9 different sockets to serve each TCP connection and essentially transmits the same data through each of these sockets. Thus, nine exact copies of the file available at server are sent across the link connecting the file server and the core router. The core router in turn sends 3 copies of the same data on each of the access links. Now imagine the scenario where the number of interested clients increases from nine to say around a few hundreds. This is common in case of new files (like movies) getting hosted on websites or critical security patches being made available by software companies. In that case, too much of server bandwidth and bandwidth of access routers is wasted. This leads to each client getting low download rates and bad user experience. We call this problem as the problem of data redundancy and work towards solving this by proposing a Hybrid content distribution network (CDN) which leverages the basic BitTorrent protocol.

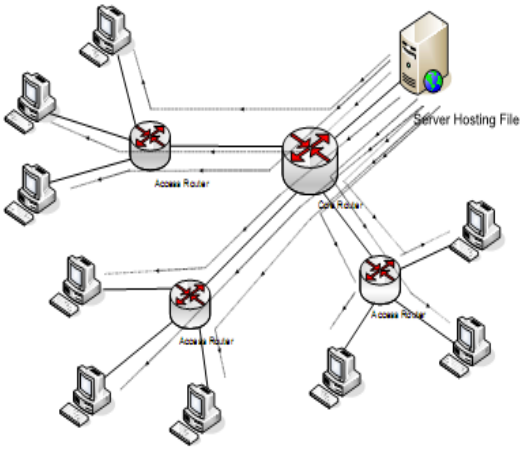

Fig. 1. Concurrent downloads cause heavy load on server bandwidth and network resources

1.2 BitTorrent Protocol

There are several systems that focus on file sharing; one widely deployed is BitTorrent **[1].** BitTorrent is a "tit-for-tat" file sharing system whose operation is described in this section. The basic idea behind BitTorrent is to divide a file or set of files in several pieces also called fragments. BitTorrent distinguishes two kinds of

peers, that is, down loaders and seeds. Down loaders are peers that have some fragments of the file, while seeds have a complete copy of the file. Down loaders and seeds that share a torrent form a swarm. A torrent file is usually published on the Internet as a text file, it contains the following information:

- Number of pieces, for each piece a checksum is created to guarantee its integrity, this checksum is created using the SHA1 hash function and included in this torrent file.
- The URL of a Tracker A tracker is responsible of keeping track of the down loaders and seeds that are in the swarm. When a peer needs to know which other peers or seeds are currently connected, it makes an HTTP request to the tracker asking for IP addresses and ports of other peers. In other words, the tracker is responsible to keep track of membership.

Since the file is broken into fragments, peers may share different fragments with other peers. As mentioned before, a peer is aware of other peers by querying the tracker; once it has their IP addresses, it can establish TCP connections with some of them to download or upload data. Each peer is responsible for keeping upload and downloads rates statistics of the connections it has established. This maximizes its download rate by downloading from whoever it can and make a decision of which peers to upload using a tit-for-tat approach. With this information, if one peer is not providing fragments, it may be choked, which is temporary refusal to upload to other peer.

1.3 Some Drawbacks of BitTorrent Protocol

BitTorrent protocol often sustains to following drawbacks:

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

1.4 BitTorrent Location Aware Protocol

As mentioned above, the original BitTorrent protocol can lead to peers geographically distant from one another exchanging data when peers close by are also present, leading to suboptimal performance. A location-aware BitTorrent protocol has been proposed in **[8].** However, the proposal is in a very lose form with no real world implementation or performance results**.** It requires each BitTorrent client to supply its approximate geographical location (longitude and latitude) when contacting the tracker to get the peer list. The tracker knows geographical locations of all down loaders and thus returns the list of peers to the original requesters which are closer to it, instead of returning a random list (as in case of the original Bit Torrent tracker).

Several issues arise here. Firstly, this protocol is not compatible with the original BitTorrent protocol and requires changes at the trackers. Secondly, assuming that the geographical location of a client would be known is not realistic. Thirdly, clients located close to each other geographically may not be having a fast network link between them and might be separated by several hops in terms of routing. Finally, absence of any implementation of this protocol makes one skeptical about the relative performance gain of it.

2 Performance Study of Content Distribution Models

Earlier, we talked about the content distribution models, including the Peer-to-Peer Systems model. With the help of an example scenario, we also illustrated the problem of same data being re-transmitted over internet links, leading to degraded performance and higher running costs. Now, we present the results of a large scale experimental study to understand the performance of each of the content distribution models. The study was conducted using the Emulab [9] emulation facility.

2.1 Experimental Setup

- **Network Topology:** The first step towards performing experiments on Emulab is to specify the network topology and the specification of hardware and software on each node of the network. This is done with the help of a topology specification script written in tcl programming language, in a format identical to that of NS-2 [10] (the program code is mentioned in 2.4). Internet can be assumed to be composed of following two entities.

Backbone Network: It consists of the high bandwidth, high delay, and long distance network links, which typically run across continents and countries. These backbone links are generally hosted by various Internet Service Providers (ISPs) and account for the main cost in running the internet.

High Speed LANs: Most organizations today have access to high speed local area networks (LANs) which in turn are connected to the backbone internet via particular nodes (routers). Such LANs are generally error-free and congestion free and are administered by the local organizations. Since the major cost in running Internet is in maintaining the backbone network, the ISPs are generally concerned about transferring the data across backbone links in the most cost-effective manner. The cost for a link is proportional to the amount of data (or the number of bytes) transferred across the link. In this study, we try to understand the typical amount of traffic which the ISPs need to transfer to support the different content distribution models. Also, as we show in this study, most of the current models end-up sending the same data again and again over the same links. We are interested in designing a hybrid CDN structure which restricts such retransmissions. Figure 2 illustrates the network topology used for this performance study on Emulab.

The internet backbone is made up of four core routers, named coreRouter0, coreRouter1, coreRouter2 and coreRouter3. Each of the core Routers run on the Red Hat Linux 9.0 Standard operating system. The four core routers are all connected to each other in a symmetrical manner and thus there are total six core links named corelink0 ... corelink5. Each of the core links is a 10Mb link with a 20 ms end-to-end delay and a Drop Tail queue. 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 runs the Red Hat Linux 9.0 version of operating system. The link between a router and a core router is a 2Mb 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.

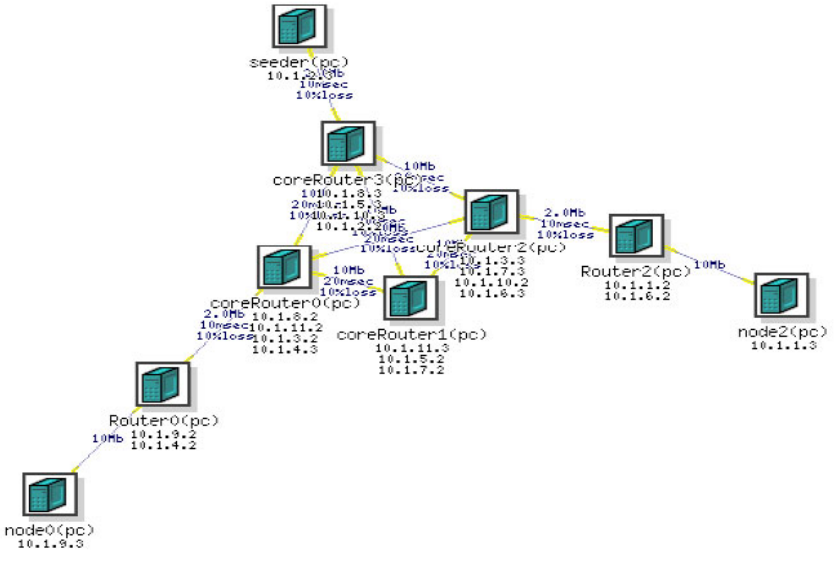

Fig. 2. The experimental setup used for the performance study

- **Performance Metrics:** In this study, we are concerned about quantifying the amount of data transmitted over backbone links in the various content distribution models. Thus, we measure two key metrics in each experiment run, for each link, in each direction:

  Number of Bytes: This represents the raw amount of data transferred over a link in a particular direction.

  Stress: This represents the ratio of number of total packets transmitted over a link and the number of unique packets transmitted over the link. For example, a stress of 2 represents a case where each packet is transferred twice over a link. As mentioned earlier, the running cost of a link for the ISP

is proportional to the raw amount of data transferred over a link. A higher link stress refers to the case where higher redundant transmissions of the same data are happening over the link, thus wasting the bandwidth.

Emulab has simple support for tracing and monitoring links and LANs. For example, to trace a link:

```
set link0 [$ns duplex-link $nodeB $nodeA 30Mb 50ms DropTail]
 $link0 trace
```

The default mode for tracing a link (or a LAN) is to capture just the packet headers (first 64 bytes of the packet) and store them to a tcpdump output file. In addition to capturing just the packet headers, one may also capture the entire packet:

```
$link0 trace packet
```

By default, all packets traversing the link are captured by the tracing agent. To narrow the scope of the packets that are captured, one may supply any valid tcpdump style expression:

```
$link0 trace monitor "icmp or tcp"
```

One may also set the **snaplen** for a link or LAN, which sets the number of bytes that, will be captured by each of the trace agents:

```
$link0 trace_snaplen 128
```

In our experiments, we set the snaplen to 1600 bytes. For each link (say link0, between nodeA and nodeB), 2 trace files of interest are generated by tcpdump: **trace nodeA-link0. recv** and **trace nodeB-link0.recv.** Here, the first trace file stores the packets sent by nodeA to nodeB over link0, while the second file stores the packets sent by nodeB to nodeA over link0. To analyze the tcpdump trace files, we modified a well-known tool tcptrace. We added a module in the tcptrace code to calculate the MD5 checksum of payload of each tcp packet and store the checksums of all payloads in a file. The number of checksums is equal to the total number of packets transmitted over a link. We then calculate the number of unique checksums in the file, which represents the number of unique packets transmitted. The ratio of these two gives the link stress. Also, the total number of bytes from payloads of all tcp packets on a link can be easily calculated from tcptrace.

In our experiment we designed a BitTorrent client which supports the following:

- Must support a console based interface to allow remote execution over Emulab nodes.
- We preferred it to be in java so that Datagram sockets could be used to extend it to support IP multicast [11].

2.2 Performance Evaluation

Performance evaluation of Peer to Peer and WWW model is illustrated below:

- **Link Statistics for the File Download Using BitTorrent**

Link	Direction	No. of Bytes	Stress
coreLink0	coreRouter0 − > coreRouter1	4.1 MB	5.712
coreLink0	coreRouter1 − > coreRouter0	3.6 MB	6.078
coreLink1	coreRouter2 − > coreRouter1	3.9 MB	5.429
coreLink1	coreRouter1 − > coreRouter2	3.1 MB	5.896
coreLink2	coreRouter0 − > coreRouter2	2.8 MB	6.368
coreLink2	coreRouter2 − > coreRouter0	4.0 MB	5.221
coreLink3	coreRouter0 − > coreRouter3	15 KB	2.800
coreLink3	coreRouter3 − > coreRouter0	0.8 MB	1.544
coreLink4	coreRouter1 − > coreRouter3	16 KB	3.620
coreLink4	coreRouter3 − > coreRouter1	1.2 MB	1.855
coreLink5	coreRouter2 − > coreRouter3	15 KB	2.880
coreLink5	coreRouter3 − > coreRouter2	0.9 MB	1.366
link0	coreRouter0 − > router0	8.5 MB	6.630
link0	router0 − > coreRouter0	6.9 MB	7.013
link1	coreRouter1 − > router1	9.3 MB	7.155
link1	router1 − > coreRouter1	6.7 MB	7.300
link2	coreRouter2 − > router2	6.8 MB	6.516
link2	router2 − > coreRouter2	8.0 MB	6.933
link3	coreRouter3 − > seeder	47 KB	3.184
link3	seeder − > coreRouter3	3.0 MB	2.797

Fig. 3. The experimental setup used for the performance study

In the P2P model, clients download the file in a collaborative manner. Instead of depending only on the seeder for the file download, each client fetches data packets from other clients as well. Thus, in this case, clients have TCP connections between them, in addition to TCP connections with the seeder. In P2P model, since clients are also responsible for uploading packets to other clients, thus the uplink capacity is also used in P2P model as compared to the www model which is shown in figure 4 using Wget [12].

All the links see data transfers of the order of 4-6 MB, not like the case of WWW model, where several links had to transfer as much data as 14 MB. Data transfer happens in both directions (uplink and downlink). The other important observation is regarding the link stress. We observe that link stress values are smaller in case of the core links. This means that there is lesser number of duplicate packet transmissions happening over the internet links, thus avoiding the wastage of resources. This is due to the fact that each client observes the data pieces which are available with other clients and fetches them as well, instead of fetching pieces always from the seeder.

Link	Direction	No. of Bytes	Stress
coreLink0	coreRouter0 − > coreRouter1	0	0
coreLink0	coreRouter1 − > coreRouter0	0	0
coreLink1	coreRouter2 − > coreRouter1	0	0
coreLink1	coreRouter1 − > coreRouter2	0	0
coreLink2	coreRouter0 − > coreRouter2	0	0
coreLink2	coreRouter2 − > coreRouter0	0	0
coreLink3	coreRouter0 − > coreRouter3	1.3 KB	12.000
coreLink3	coreRouter3 − > coreRouter0	14.5 MB	11.585
coreLink4	coreRouter1 − > coreRouter3	1.3 KB	12.000
coreLink4	coreRouter3 − > coreRouter1	14.5 MB	4.573
coreLink5	coreRouter2 − > coreRouter3	1.3 KB	13.000
coreLink5	coreRouter3 − > coreRouter2	14.5 MB	4.485
link0	coreRouter0 − > router0	14.5 MB	11.585
link0	router0 − > coreRouter0	1.3 KB	12.000
link1	coreRouter1 − > router1	14.5 MB	4.573
link1	router1 − > coreRouter1	1.3 KB	12.000
link2	coreRouter2 − > router2	14.5 MB	4.485
link2	router2 − > coreRouter2	1.3 KB	13.000
link3	coreRouter3 − > seeder	3.9 KB	37.000
link3	seeder − > coreRouter3	43.6 MB	8.315

Fig. 4. Link Statistics for file download using Wget

2.3 IP-Multicast as Content Distribution Model

IP Multicast is a particularly attractive alternative for content distribution in such scenarios. All the clients can initially send IGMP request messages to join a multicast group and the source can multicast the data on this group. Since routers are aware of the physical topology and positions of clients, the data traverses the shortest path to reach each of the clients, guaranteeing optimal download time. Although such an approach is promising, it is not viable in today's Internet because of lack of support of IP Multicast on Internet. This means that two nodes on the Internet do not necessarily have a route between them which is IP Multicast enabled **There are several reasons why IP Multicast is not available on the Internet**. These include:

• Most routers on the Internet lack support for IP Multicast. Recollect that to support IP Multicast, a router needs to perform several additional operations like duplication of packets with PIM, IGMP support, Multicast forwarding etc. The routers available on Internet simply do not have resources or capabilities to perform all such operations. Upgrading such existing routers is clearly infeasible.
• Congestion control schemes are not well defined for multicast.
• Pricing policies in multicast are not clear. Hence, there are no incentives for the ISPs to be interested in deploying multicast support in the networks.

Therefore, it is almost clear that utilizing IP-level multicast for large scale content distribution in above mentioned scenarios is not feasible. The problem of IP Multicast

as an unreliable protocol is that it works over UDP. This means that there is no guarantee that a packet multicast over UDP will be successfully received by other clients. 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 by clients would be received by all other clients on the island. The clients which have low received buffer or which are busy with other operations often are unable to completely receive packets sent over multicast. We tackle the above problem in providing more efficient data sharing through the concept of 3-way Hand shake [13] and propose a method which co-exist with the standard BitTorrent protocol and leverage IP Multicast to distribute downloaded pieces to other BitTorrent clients on the same network.

2.4 Program Code

The NS-2 script used in the experiment is shown below:

```
#generated by Netbuild 1.03
set ns [new Simulator]
source tb_compat.tcl

set coreRouter0 [$ns node]
set coreRouter1 [$ns node]
set coreRouter2 [$ns node]
set coreRouter3 [$ns node]
set seeder [$ns node]
set Router0 [$ns node]
set node0 [$ns node]
set Router2 [$ns node]
set node2 [$ns node]

set link0 [$ns duplex-link $coreRouter0 $coreRouter3 10Mb
20ms DropTail]
tb-set-link-loss $link0 0.1
set link1 [$ns duplex-link $coreRouter0 $coreRouter1 10Mb
20ms DropTail]
tb-set-link-loss $link1 0.1
set link2 [$ns duplex-link $coreRouter0 $coreRouter2 10Mb
20ms DropTail]
tb-set-link-loss $link2 0.1
set link3 [$ns duplex-link $coreRouter1 $coreRouter3 10Mb
20ms DropTail]
tb-set-link-loss $link3 0.1
set link4 [$ns duplex-link $coreRouter1 $coreRouter2 10Mb
20ms DropTail]
tb-set-link-loss $link4 0.1
set link5 [$ns duplex-link $coreRouter2 $coreRouter3 10Mb
20ms DropTail]
tb-set-link-loss $link5 0.1
```

```
set link6 [$ns duplex-link $coreRouter3 $seeder 2Mb 10ms
DropTail]
tb-set-link-loss $link6 0.1
set link7 [$ns duplex-link $Router0 $coreRouter0 2Mb 10ms
DropTail]
tb-set-link-loss $link7 0.1
set link8 [$ns duplex-link $Router2 $coreRouter2 2Mb 10ms
DropTail]
tb-set-link-loss $link8 0.1


set lan0 [$ns make-lan "$Router0 $node0 " 10Mb 0ms]
set lan1 [$ns make-lan "$Router2 $node2 " 10Mb 0ms]



$ns rtproto Static
$ns run
#netbuild-generated ns file ends.
```

2.5 Our Approach

We used a highly modular approach to the problem. We figured out that there are basically 5 parts to the program:

1. **Database Manager:** This takes care of the list of chunks of different files available on the network.
2**. Chunk Maker/Assembler:** This creates chunks of a file and maintains a mechanism for testing the integrity of each chunk. It also assembles the chunks into a complete file when all the chunks of a file have been downloaded.
3. **Chunk Sender/Receiver:** This communicates on a single port with another host on a defined port and transfers file reliably. This throws back problems if encountered in the process or flags a success message if it is successful.
4. **User Interface:** This is where the user interacts with the program. We have 2 such interfaces, one is a GUI and another is a console one. Here the user can ask to share a file on the network and fetch a file from the network.
5. **The Head:** This interacts with every other part and decides what to do when. It basically deploys the work to other modules and also performs a 3-way handshake before a communication begins on a defined port using the Chunk Sender/Receiver.

Multicast packets on an island can be lost or delayed due to two things:

1. The clients and links on a LAN show abnormal behavior (due to load or miss-configuration) leading to random packet losses.
2. There is congestion on the LANs due to other heavy traffic being exchanged by clients, e.g., VoIP etc.

To overcome this problem we applied the method of 3-way handshake which is illustrated below:

The tracker when requested to fetch a file from the network, it does the following:

a. Asks the Managed Hash Table for the information of the locations of the chunks.
b. Now for each chunk, it contacts the Tracker of another host sending a Type1 packet requesting a chunk.
c. The peer host's tracker sends back a packet which can be:

• Type2 packet: This says that the peer host has accepted the request and it is designating a port for sending the chunk.
• Type3a packet: This says that the peer host does not have the chunk requested and thus is negating the connection.
• Type3b packet: This says that the peer host has the chunks but currently does not have any free ports to take the request.

d. If Type3a is received then the tracker tries to request the file from another source, if available.
e. If Type3b is received then the tracker would look for other sources and if it runs out of other sources it ask the same host after some time.
f. If Type2 packet is received, the Tracker sends a Type4 packet that carries the information about which port of this user would be listening for the packets and starts the Downloader.
g. On reception of Type4, the Up-loader is called.
h. If on any of these communications, a timeout is faced, it is gracefully handled
Thus we achieve a 3-way handshake similar to TCP for starting up the chunk transfer.

3 Results

For the sake of completeness, the topology is shown in Fig 2. There are two types of links in this topology:

Core links: which serve the traffic across the internet by connecting the **core routers**; and **Access links**; which are used to provide internet access to the islands consisting of various high-speed LAN's. Since the two types of links carry different type of traffic, we show the evaluation of both types separately.

Each island in our experimental topology consists of 3 high-speed (10 Mbps) LANs. All the LANs are connected to each other via the access router (i.e., router0, router1 or router2). Each of the access routers runs the Red Hat operating system. In order to allow IP Multicast across different LANs on the same island, we run mrouted **[14]** on each of the access routers. The mrouted utility is an implementation of the Distance-Vector Multicast Routing Protocol (DVMRP), an earlier version of which is specified in RFC-1075 **[15].**It maintains topological knowledge via a distance-vector routing protocol (like RIP, described in RFC-1058 **[16]**), upon which it implements a multicast datagram forwarding algorithm called Reverse Path Multicasting. The mrouted 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 sub networks 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 reaches all other LANs on the same island, provided there 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.

The seeder serves a file of size 1 MB. All the results reported in this section have been obtained after proper averaging over 5 to 10 runs of each experiment. Figure 5 below shows the comparison between the bandwidth utilization of BitTorrent client and the Hybrid CDN. Note that the steeper the plot is, the faster is the completion of download for all the clients. In the above figure 100 % of clients complete their download within 30 seconds while using Hybrid CDN. It takes about 60 seconds for all the nodes to complete their download using Bit-Torrent. Figure 6 show the effect of random link losses, a random packet loss module is installed in each of the LANs, whose packet loss rate can be configured. We varied the packet loss rate of each LAN from 0% to 5% and repeated the experiments for each case to measure the various performance metrics for both Hybrid CDN and BitTorrent. Experimental Client's download time increase after 4% packet loss, due to the fact that during multicasting maximum of the packets get lost and they are retransmitted using unicasting. However, random packet loss percentages as high as 4% are quite rare in most LANs today and thus represent an unnatural scenario. With the more common scenarios, Hybrid CDN is shown to have a better performance over BitTorrent.

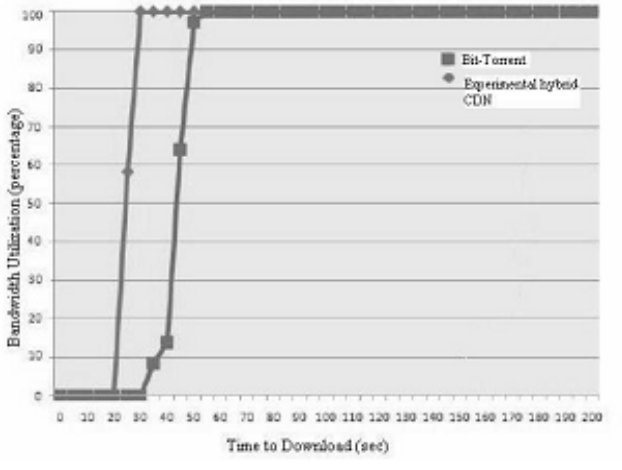

Fig. 5. Cumulative Distribution Function of time for download by each client

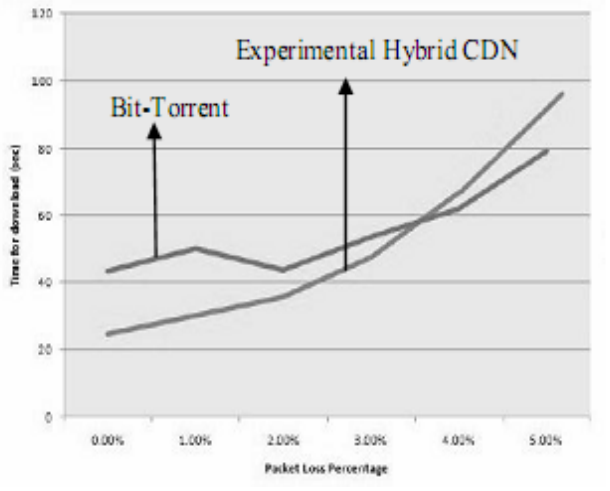

Fig. 6. Time for download with Packet Loss Percentage of each LAN

Finally, Figure 7 shows the variation of average link stress with packet loss percentage. Stress on the internet links increases with random packet loss due to the higher number of TCP retransmissions to deliver data across the LAN network. Note that more retransmission means same data packets traversing internet links again and again. 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 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 8 shows the variation of average time for download over all clients with increasing value of CBR traffic rates.

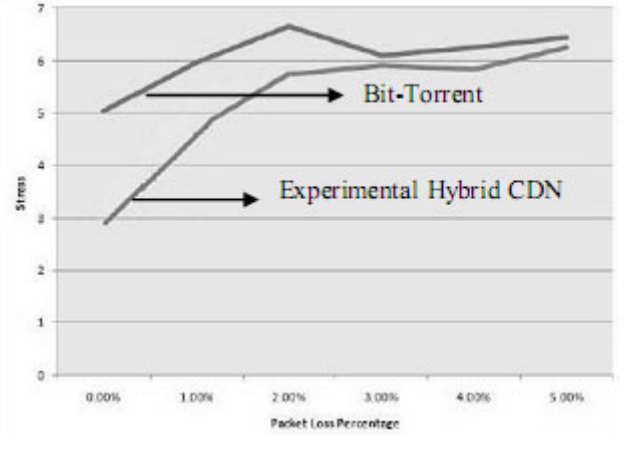

Fig. 7. Link stress with Packet Loss Percentage of each LAN

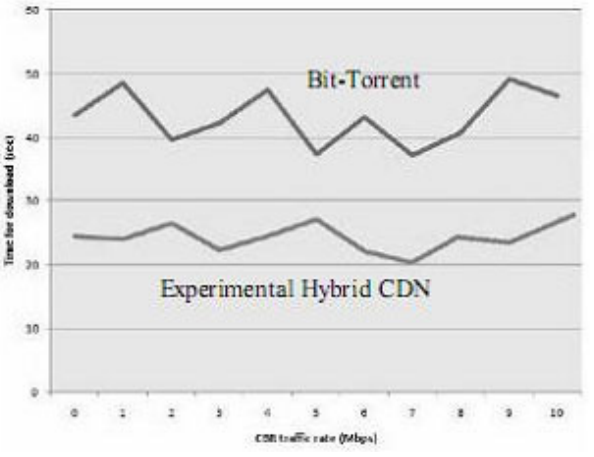

Fig. 8. Time for download with varying congestion level

4 Conclusion

We obtained the following three important results with the Hybrid CDN Model:

- Reduction in download time of each client using Hybrid CDN by 48% over Bit- Torrent and by 86% over WWW Protocol.
- Reduction in traffic load on Internet links and ISPs.
- Reduction in the wastage of resources like bandwidth due to redundant packet.

Downloading time is the most critical performance metric for normal Internet users, whose experience with the system is largely determined by how fast they can Download file from the Internet. Also, recent applications of Peer-to-Peer systems like distributing the software updates and the images of operating systems, etc., over large networks spread across a geographically distributed area depend heavily on the download time for each computer. The delay in download in [**7**] can be overcome by the use of Hybrid CDN like model leveraging the IP-Multicast and Bit-torrent protocol applicability. Most ISPs today observe heavy traffic load on their Internet links due to increasing number of users using Peer-to-Peer file sharing systems. Due to competition, ISPs are forced to reduce tariff continuously resulting in reduction in the margins of profit. However, with more users migrating to a system like Hybrid CDN, the load on ISP resources (Internet links) can be reduced by as much as 65%, for the comparable amount of downloads by end clients. Thus, the profit margins of ISPs can be increased heavily if they encourage more users to switch to such type of system. The load on access links is also reduced by similar proportions, the island owners have to pay for the Internet access links, on the basis of the usage of such links. With reduced usage of access links, the Internet consumption bills for island owners can be reduced

considerably, which in turn will be a motivation for them to enable IP Multicast support on their networks requiring software (and in some cases hardware) upgrades. Thus, such models are economically sustainable.

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

Standard compliance: The proposed method is interoperable with BitTorrent protocol. It only requires changes at the end client level, unlike other solutions, which would need network wide support.

Actual Implementation: In place of theoretical results or Network simulations, we resorted to actually implementing a prototype system of our method and have evaluated it on a large scale real network.

References

1. Cohen, B.: Incentives build robustness in Bit-Torrent. In: Proc. of the First Workshop on the Economics of Peer-to-Peer Systems (2003)
2. Leibowitz, N., Bergman, A., Ben-Shaul, R., Havit, A.: Are file swapping networks cacheable? Characterizing p2p traffic. In: Proc. of the 7th Int. WWW Caching Workshop (2002)
3. Karagiannis, T., Rodriguez, P., Papagiannaki, K.: Should internet service providers fear peer-assisted content distribution. In: Proc. of IMC, pp. 63–76 (2005)
4. Wierzbicki, A., Leibowitz, N., Ripeanu, M., Wozniak, R.: Cache replacement policies revisited: the case of p2p traffic. In: Proc. of IEEE International Symposium on Cluster Computing and the Grid (CCGrid), pp. 182–189 (2004)
5. Saleh, O., Hefeeda, M.: Modeling and caching of peer-to-peer traffic. In: Proc. of IEEE International Conference on Network Protocols, pp. 249–258 (2006)
6. Vahdat, A., Yocum, K., Walsh, K., Mahadevan, P., Kostic, D., Chase, J., Becker, D.: Scalability and accuracy in a large-scale network emulator. In: Proc. of OSDI (2002)
7. BitTorrent used to update workstationsm, http://torrentfreak.com/university-usesutorrent-080306
8. BitTorrent location-aware protocol 1.0 specification, http://wiki.theory.org/BitTorrentLocation-awareProtocol1.0Specification
9. Emulab documentation, http://www.emulab.net/doc.php3
10. The network simulator - ns-2, http://www.isi.edu/nsnam/ns
11. IP-Multicast, http://www.cisco.com/en/US/docs/internetworking/technology/handbook/IPMulti.html
12. Gnu Wget, http://www.gnu.org/software/wget
13. 3-way Handshake, http://www.inetdaemon.com/tutorials/internet/tcp/3-way_handshake.html
14. How to set up Linux for multicast routing, http://www.jukie.net/bart/multicast/Linux-MroutedMiniHOWTO.html
15. Distance vector multicast routing protocol, http://www.ietf.org/rfc/rfc1075.txt
16. Routing information protocol, http://www.ietf.org/rfc/rfc1058.txt