Assessing the Impact of BitTorrent on DOCSIS Networks

James J. Martin and James M. Westall
Department of Computer Science
Clemson University
Clemson, SC 29634–0974
Email: jim.martin, westall@cs.clemson.edu

Abstract— **BitTorrent is a widely deployed peer-to-peer protocol that provides scalable file sharing capabilities. While BitTorrent applications contribute to the demand for high speed broadband access, they also contribute to the undesirable 80/20 effect wherein 80% of the bandwidth is consumed by 20% of the users. In this study we explore the impact that BitTorrent users can have on a DOCSIS cable network. We began the study by capturing packet traces of BitTorrent applications operating on two commercial DOCSIS cable networks. Next we developed for the *ns-2* simulation tool a configurable mix of BitTorrent, Web browsing, and VoIP workloads and verified that the behavior of the simulated BitTorrent workloads to be consistent with the behavior observed on the commercial network. In this simulated environment, we show that as few as 15 BitTorrent users can significantly reduce the service quality experienced by other subscribers.**

I. Introduction

The use of peer-to-peer (P2P) applications has grown to such an extent that P2P traffic often greatly exceeds HTTP traffic on Internet links. Although P2P networks are well known for the ethical and legal issues raised by their widespread use, they also pose interesting technical challenges in the area of bandwidth management, especially on shared medium access networks. In xDSL access networks, intelligent queuing algorithms in gateway routers can reduce the impact of P2P users, but in shared medium access networks such as DOCSIS we show that this impact can be a significant problem.

A. DOCSIS

The Data Over Cable Service Interface Specification (DOCSIS) defines a set of standards that support the transport of data over a cable network [1], [2]. The physical layer of a DOCSIS system is a shared access cable. Medium access is controlled by a head-end device called the Cable Modem Terminating System (CMTS). All packet flow is between the CMTS and the Cable Modems (CMs). Downstream bandwidth is shared in an asynchronous TDM fashion under the control of the CMTS via a possibly prioritized form of round robin scheduling.

Multiple CMs must contend for access to the upstream channel, and the channel allocation procedure is quite complex. The upstream channel is subdivided into transmission slots referred to as mini-slots. The capacity in bytes of a mini-slot on a given DOCSIS network is fixed and is in the range of 8 to 16 octets. Permission to transmit data in a block of one or more mini-slots must be granted to a CM by the CMTS. The CMTS grants mini-slot ownership by periodically transmitting a frame called the MAP on the downstream channel. In addition to ownership grants, the MAP also typically identifies some mini-slots as contention slots in which CMs may bid for for quantities of future mini-slots. To minimize collisions in the contention slots, a non-greedy backoff procedure is employed. Each CM is required to randomly select the contention slot in which it transmits a bid for mini-slots. When collisions do occur in contention slots, all parties that collide are required to employ an exponential backoff, doubling the size of the window of slots in which the next bid is randomly placed. Two additional facilities reduce contention. When a CM has a backlog of upstream traffic it may *piggyback* a request for additional mini-slots in a request field of the current frame header. The *concatenation* facility allows multiple (typically small) IP packets to be transmitted as a single logical upstream MAC layer protocol data unit.

B. BitTorrent

BitTorrent is a peer-to-peer (P2P) system that can consume tremendous amounts of bandwidth. For example, at Clemson University BitTorrent is the single largest consumer of outbound Internet bandwidth. This fact led us to conjecture that BitTorrent traffic would comprise a significant portion of the load on any network serving a significant population of teenage and young adult users, and subsequent analysis of network trace data provided by a large commercial cable operator has confirmed this.

The BitTorrent protocol has a particular characteristic that makes its use problematic in shared medium access networks with asymmetric provisioning such as DOCSIS. A user whose sole intent is to *download* a large file often consumes more *upstream* bandwidth than downstream in the process!

In this paper we characterize the impact that the upstream traffic generated as a side-effect of BitTorrent downloads can have on a DOCSIS network. Our traffic model is derived from traces of actual BitTorrent traffic captured on the Internet. The traffic model is then used to drive an *ns-2* simulation of a DOCSIS network from which the results are obtained.

The remainder of the paper is organized as follows. In section II we review the architecture and related performance studies of peer-to-peer systems. In section III we describe measurement studies designed to characterize bandwidth consumption of BitTorrent clients. The results of simulation analysis designed to assess the impact of BitTorrent on an HFC cable network are described in section IV. We end the paper with conclusions and our future directions.

II. Background and related work

In addition to file sharing, P2P has been used for grid computation [3], storage [4], web caching [5] and directory services [6], [7]. BitTorrent, however, is designed specifically for file distribution. The algorithms contained in BitTorrent were heavily influenced by the earlier P2P protocols and applications such as Gnutella [8] and Kazaa [9], [10].

A. BitTorrent

The BitTorrent protocol itself is described in [11], [12]. A downloadable file (or collection of related files) is commonly referred to as a *torrent*. Each torrent is partitioned into *pieces*, commonly 256 Kbytes in size. Different pieces of a torrent may be simultaneously downloaded from different peers. The efficacy of this approach was validated in [13] whose authors experimented with several parallel access schemes to download a file from multiple servers concurrently.

To initiate a BitTorrent download, a BitTorrent client must contact the *tracker system* for the torrent to be downloaded. Tracker system addresses may be retrieved from a collection of well-known web sites that provide indexing services and distribute small *.torrent* metadata files that identify both the tracker system for the torrent and the pieces that comprise it. The tracker system is responsible for disseminating the dynamically changing set of IP addresses of active BitTorrent client (peer) nodes that presently hold pieces of a torrent. However, the tracker system *does not* normally serve the data pieces of a torrent.

When a BitTorrent client contacts a tracker to initiate a download, the tracker system provides a list of IP address/port pairs identifying peers that have one or more pieces of the file. The client attempts to maintain connections with at least 20 peers (referred to as the peer set). If it can not, it asks the tracker for additional peers. The client exchanges pieces of the *torrent* that it has downloaded for pieces that it needs.

An objective of the BitTorrent protocol is to reward peers that are presently providing the highest download rates. When performing a download, a BitTorrent client will also upload requested pieces that it holds to at most five peers at a time. The first four of these are the peers presently providing the highest download rates, and the fifth is randomly selected. Therefore, after a BitTorrent client has successfully downloaded a piece of the torrent it may subsequently forward the piece to multiple peers. It is this effect that can cause the total amount of upstream traffic produced during a download to exceed the total amount of downstream traffic.

After a peer has downloaded the entire file, it becomes a *seed* and provides pieces of the file to any peer. The system is self-scalable in that as the number of downloaders increase, so does the number of nodes that can provide pieces of the torrent.

To enhance performance, BitTorrent further breaks pieces into smaller (typically 16 Kbytes) sub-pieces (sometimes called blocks) and maintains a strategy of having at least 5 concurrent block requests pipelined to a given peer. A piece selection algorithm in the client determines the order in which pieces are requested. This algorithm randomly selects the initial pieces. After one or more pieces are received, the algorithm switches to a rarest first algorithm which, based on information learned from its peers, attempts to the least available pieces first. When the peer has downloaded all but the last few pieces, it sends requests for these sub-pieces to all peers.

B. Related work

There has been a great deal of prior research on P2P protocols and systems. The majority of the studies have focused either on P2P deployments in the Internet [14]–[16] or on evaluating protocol issues [7], [17], [18]. There have been several analytic models proposed. Several notable efforts have been based on queuing theory [19] and on fluid flow models [20], [21].

The existing studies of BitTorrent indicate that the protocol is indeed scalable and robust [22]–[26]. In [24] the authors examined the log file from the tracker associated with the popular Redhat Linux 9 distribution torrent. They observed that peers continue to participate in the BitTorrent network as a seed for an average of 6.5 hours after the entire file has been downloaded. The authors also showed that the average download rate is over 500 Kbps and that nodes do consume symmetric amounts of bandwidth. The authors observed that 81% of all file downloads were incomplete. Of these aborted downloads, 90% had retrieved less than 10% of the file. The average download rate of the 19% of the sessions that completed was 1.3 Mbps which is larger than the average download rate of all sessions at 500 Kbps. The authors studied an individual peer by running an instrumented client. The client downloads a 1.7 Gbyte file in 4500 seconds. During the download period, the client interacted with roughly 40 peers. They found that the volume of traffic in the upstream and downstream directions at the client was correlated but throughputs were not correlated. 85% of the file was sent by only 20 peers including 8 seeds that provided 40% of the file. This set of 20 peers were not a part of the initial peer list provided by the tracker which suggests that to enhance performance NAT must not prevent nodes from connecting with a downloader.

The work that is most similar to ours performed an analysis of client-side BitTorrent nodes participating in a number of long download sessions (2 to 12 days in length) [27], [28]. The authors found that the average session transferred roughly 40 Mbytes and lasted for roughly 400 seconds. Session interarrivals can be modeled using the hyper-exponential distribution and session durations can be modeled by the lognormal distribution. Our modeling work differs in that we are interested in characterizing the bandwidth consumption of individual TCP connections.

III. Modeling BitTorrent traffic

BitTorrent traffic consists of two components: signaling and data. Signaling traffic is described in [27]. It is carried in low-rate TCP flows in which application payloads are smaller than the TCP header. It typically comprises a very small percentage of overall traffic, and we do not consider it in this study.

Although the measurement results presented in this paper are limited to downloads of a single torrent, we have analyzed traces associated with many torrents. These studies clearly demonstrate that no single model can reasonably reflect all of the operational modes that can be readily observed. Furthermore, it is the case that not all torrent downloads have adverse impact upon competing traffic. Of the three different modes of operation identified below, only one has particularly adverse impact upon a DOCSIS network.

The highest performance torrents have many seeds that are hosted by high speed access networks such as Internet-2. An example of such a torrent is the Fedora-Core 4 distribution of Linux. In one experiment, a torrent of size 2.6 Gbyte was downloaded by a DOCSIS client having a nominal 3 Mbps downstream limit at a sustained throughput of over 800 Kbps. Because Fedora-Core 4 had already been superceded by Fedora-Core 5, there was little demand for this torrent, and consequently the download produced virtually no upstream data traffic at all. The download of a torrent of this type, like the download of a UTube movie from a high speed server, generates only ACK flow in the upstream direction and thus has little impact on other users of a DOCSIS network.

At the other performance extreme we have encountered collections of music with few seeds and/or few peers in which the sustained download rate is less than 10 Kbps. Torrents having few downloaders or low bit rates also have little negative impact on competing traffic in a DOCSIS network because they do not produce high volume upstream traffic.

In contrast, shortly after a new torrent such as a new distribution of Linux (or a pirated popular movie) is first posted, there are many peers actively exchanging pieces of the torrent. When this occurs, it will be shown that more *upstream* traffic than downstream can be associated with a *download*. In this situation, competing traffic on a DOCSIS network can be significantly affected. Thus, it is this type of BitTorrent traffic that we seek to model.

A. BitTorrent traces

Because our objective was to demonstrate the potential adverse affects of BitTorrent traffic on a DOCSIS network, we obtained sets of traces of the download of a 4.3 Gbyte torrent that was being actively traded on the public Internet. According to statistics obtained from the tracker, this torrent consistently had hundreds of downloaders. We also observed that tens of peers entered and left our peer set over the course of a download.

The BitTorrent client was located on a home network connected to the Internet through Charter's high speed cable service. The service provides 3 Mbps downstream and 512 Kbps upstream. The BitTorrent client was version 4.04 from the original (and still popular) client developed by Cohen [2]. The client PC was a 2.8 MHz Intel machine with 1 Gbyte of RAM running WindowsXP Professional. We captured the traces on the client PC using Ethereal [29]. The home network connected to the cable modem through a broadband router that provided NAT and port filtering capabilities.

Trace File	Pkts / 10^6	TCP Cxs	Hours	Incoming Cx
Grp 1, DS 1	6.84	1552	15.3	blocked
Grp 1, DS 2	8.75	1997	16.4	blocked
Grp 2, DS 1	6.84	4596	8.3	open
Grp 2, DS 2	5.70	3267	6.2	open

TABLE I
Summary of BitTorrent trace files

Trace	Group	Dataset	Section
1	1	1	1
2	1	1	2
3	1	2	1
4	1	2	2
5	2	1	1
6	2	1	2
7	2	2	1
8	2	2	2

TABLE II
Trace ID Mapping Table

	Upstream			Downstream		
Trace	Secs	Bytes/10^6	Cxs	Secs	Bytes/10^6	Cxs
1	14941	572.0	658	18993	475.6	388
2	12563	581.1	245	15854	551.5	121
3	14447	594.1	685	17853	421.6	374
4	10806	533.4	341	12648	536.8	225
5	8605	494.9	1396	9534	517.1	1494
6	8297	487.9	1122	9401	542.0	1221
7	8307	463.7	1050	8160	569.4	976
8	7779	457.8	1177	8267	563.1	1223

TABLE III
Summary of BitTorrent trace files

B. Measurement data

Two groups of traces were captured. For Group 1, the broadband router was configured to block incoming TCP connections. For Group 2, unsolicited inbound TCP connections were permitted. All traces were terminated when the torrent download was complete and thus do not show the client operating as a seed. Each group contains traces of two complete downloads of the target 4.3 Gbyte torrent. The four downloads were performed serially. The two individual download traces belonging to Group 1 and Group 2 are referred to as Dataset 1 (DS1) and Dataset 2 (DS2) respectively. The characteristics of the four complete traces are summarized in Table I. This

Trace	US Kbps	DS Kbps	Ratio US:DS	US Flows	DS Flows
1	306	200	1.53	3.3, 1.9, 0.9	3.7, 1.3, 0.4
2	370	278	1.33	5.4, 2.9, 1.0	5.3, 1.8, 0.8
3	329	189	1.74	5.5, 3.2, 0.6	5.6, 1.1, 0
4	395	339	1.16	6.8, 3.8, 0.5	7.6, 1.65, 0.7
5	460	434	1.06	7.8, 5.7, 0.07	10.3, 4.0, 0.1
6	470	461	1.02	8.1, 5.6, 0.36	10.8, 3.8, 0.13
7	446	558	0.80	7.9, 5.2, 0.11	10.2, 4.7, 1.0
8	471	545	0.86	8.7, 5.5, 0.11	11.9, 4.5, 0.45

TABLE IV

AGGREGATE MEASUREMENT RESULTS

	Upstream			
Trace	Thpt(bps)	stdev	Loss %	Traffic %
1	41619	49410	0.99	85.8
2	70004	55384	1.20	87.3
3	54465	26184	0.56	81.0
4	37728	18670	0.99	70.3
5	49267	15533	0.50	78.5
6	50773	24045	3.00	79.2
7	43648	25754	1.06	84.6
8	44686	11489	0.27	78.0

	Reverse direction			
Trace	Thpt(bps)	Loss %	Traffic %	US:DS
1	13133	1.02	37.7	3.17
2	23992	0.23	43.6	2.92
3	24814	1.19	69.9	2.19
4	19954	2.13	39.2	1.89
5	36367	0.85	61.2	1.35
6	33082	1.36	50.9	1.53
7	46471	0.40	71.0	0.94
8	34996	0.44	53.7	1.28

TABLE V

TOP 10 UPSTREAM TCP CONNECTIONS IN EACH TRACE

	Downstream			
Trace	Thpt(bps)	stdev	Loss %	Traffic %
1	30297	30404	1.12	83.2
2	34338	36373	0.58	89.5
3	24718	17648	1.27	70.9
4	54253	68813	2.15	77.0
5	32984	13204	1.12	61.9
6	36905	16152	1.25	52.6
7	48707	24425	0.40	72.8
8	37954	13808	0.43	59.7

	Reverse direction			
Trace	Thpt(bps)	Loss %	Traffic %	US:DS
1	39366	2.16	77.3	1.30
2	51485	1.18	84.0	1.49
3	50976	0.58	77.3	2.08
4	28827	0.88	60.9	0.53
5	44325	0.52	78.3	1.35
6	47762	2.93	74.2	1.30
7	42315	1.03	82.8	0.87
8	38186	0.26	68.6	1.01

TABLE VI

TOP 10 DOWNSTREAM TCP CONNECTIONS IN EACH TRACE

table illustrates that the BitTorrent protocol works much more effectively when incoming TCP connections are permitted. The Group 2 downloads, using a much larger number of TCP connections, complete in approximately half the real time required by those in Group 1.

To observe changes in dynamics over the lifetime of the download and to characterize the difference in downstream and upstream behavior, we divide each of the four datasets into two sections. Each section consists of an upstream component and a downstream component. Each component consists of one million packet trace records. Section one commences at the start of the dataset, and section two starts at (aggregate) trace record 2,500,001. Thus section one data characterizes the initial phase of the download, and section two characterizes the steady state behavior. In the following discussion we refer to a section as trace n for $n = 1..8$. The mapping is shown in in Table II.

Characteristics of the upstream and downstream components of the eight trace sections are shown in table III. The *Secs* column is the elapsed time in seconds covered by the one million trace records of the component. Upstream and downstream times differ because the packet arrival rates differ. The columns labeled *Bytes* / 10^6 contain the number of bytes in millions that were carried by the one million packets belonging a component. The byte counts include application data, TCP and IP header data, and standalone ACKs. The *Cxs* columns show the number of unique destination IP address/port pairs that were observed in the upstream and downstream directions. The upstream and downstream counts differ because a single packet (e.g., a TCP reset or a SYN sent to a host that does not respond) in the upstream or downstream direction will cause the connection count to increase.

The first three columns of table IV provide the aggregate upstream and downstream bit rates in Kbps and the ratio of the amount of data transferred in the upstream direction to the downstream. In a traditional Web download of a large object one would expect to see a single upstream ACK for every two downstream segments yielding a downstream to upstream ratio of about 72:1. In contrast, the ratios observed here are close to 1:1. This overall ratio could be produced by similar numbers of upstream only or downstream only connections or by a set of connections that carry significant application data in both directions. The fact that the downstream bit rate appears constrained by the nominal bit rate (512 Kbps) of the upstream link is an indicator of significant bi-directional data flow, and additional evidence of this behavior will be provided subsequently. A particularly non-intuitive result is that blocking *incoming* TCP connections *raises* the upstream:downstream ratio.

To obtain the flow data shown in the last two columns of table IV each trace was partitioned into one-second timeslices. For each timeslice the number of unique TCP connections operating between 10 and 40 Kbps during the timeslice, between 40 and 100 Kbps, and greater than 100 Kbps were tallied. The averages of the three tallies over the lifetime of the trace appears in the last two columns. It can be seen in this table that Group 2 gains its big advantage over Group 1 in its ability to support more concurrent flows consuming less than 100 Kbps. Focusing on the connections that consumed between 40 Kbps and 100 Kbps, Group 1 exhibits an average of about 3 active connections in the upstream and between 1 and 2 connections in the downstream. Group 2 exhibits an average of between 5 and 6 connections in the upstream and just over 4 connections in the downstream.

Tables V and VI characterize the ten connections in each trace that transferred the most data in the upstream and downstream directions respectively. The throughput and loss values are the mean values of the ten connections. The column labeled Traffic % represents the percentage of total traffic that was carried by the top ten connections. For example, in trace 1 the top ten upstream connections, which represent only 0.6% of all upstream connections, carried 85.8% of all of the upstream traffic associated with the torrent download. The data in the bottom halves of these tables characterizes the reverse flows associated with the top ten connections.

It is not possible to derive the traffic percentage values from the throughput values. The throughput average of 41619 bps for Trace 1 indicates that if the top 10 upstream connections had been active for the duration of the entire trace they would have produced an aggregate upstream throughput of 416.19 Kbps. The facts that the actual aggregate throughput was only 306 Kbps and that there were many other upstream connections demonstrate the top 10 were definitely not active for the duration of the trace. The 85.8% figure was independently derived from the trace data.

The reverse flow throughput values and upstream:downstream ratios again demonstrate that individual BitTorrent connections are much more strongly symmetric than uni-directional Web object downloads where directional ratios of 72:1 would be normal. Note particularly that table VI shows that in six of the eight traces the top ten traces in the *downstream* direction actually carried more data *upstream* than downstream. It is precisely *this characteristic* that causes ten BitTorrent users to have a far more negative impact upon competing traffic with real time requirements than would ten users downloading very large Web objects.

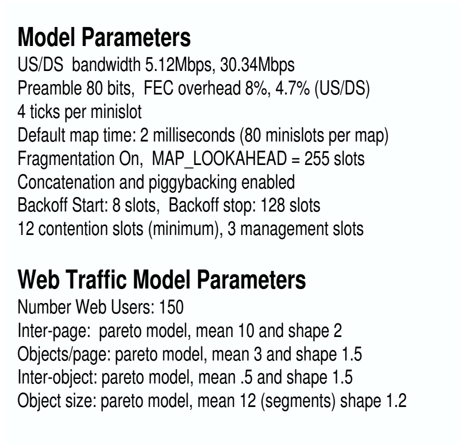
C. The simulated BitTorrent workload

We saw an enormous range of traffic characteristics between different torrents. Repeated downloads of the same torrent shows that performance depends strongly on the characteristics of the active set of peers and on the availability of unsolicited inbound connections. For the results reported in this paper, we configured the simulation to match the Group 1 traces. We estimated the RTT for the Group 1 traces by finding the time from when the client sends the first TCP segment (i.e., the SYN) until when the peer replies with the SYN-ACK segment. Using this method on traces 1-4, we found the average RTT to be approximately 0.9 seconds. The simulated BitTorrent workload that is used as described in the next section has the following characteristics:

- Three active (i.e., always-on) flows in both directions.
- An average loss rate in either direction of 0.93%.
- An average upstream TCP throughput of 51 Kbps.
- An upstream to downstream throughput ratio of about 2.5.
- An aggregate upstream bandwidth consumption of about 350 Kbps.
- An upstream to downstream aggregate bandwidth consumption ratio of approximately 1.5.

IV. Simulation of BitTorrent over DOCSIS

In previous work [30], [31], we described a simulation of the DOCSIS MAC protocol implemented for the *ns-2* simulation system. The results presented in this section were obtained by using that model augmented by a new BitTorrent traffic model that was derived from analysis of the BitTorrent traces described in the preceding section.


Model Parameters
US/DS bandwidth 5.12Mbps, 30.34Mbps
Preamble 80 bits, FEC overhead 8%, 4.7% (US/DS)
4 ticks per minislot
Default map time: 2 milliseconds (80 minislots per map)
Fragmentation On, MAP_LOOKAHEAD = 255 slots
Concatenation and piggybacking enabled
Backoff Start: 8 slots, Backoff stop: 128 slots
12 contention slots (minimum), 3 management slots

Web Traffic Model Parameters
Number Web Users: 150
Inter-page: pareto model, mean 10 and shape 2
Objects/page: pareto model, mean 3 and shape 1.5
Inter-object: pareto model, mean .5 and shape 1.5
Object size: pareto model, mean 12 (segments) shape 1.2

Fig. 2. DOCSIS and web client configuration

The simulated network is shown in figure 2. There are 400 cable modems (labeled CM-1 through CM-400) in the DOCSIS subnetwork. These share one downstream channel and one upstream channel. The DOCSIS subnetwork is connected by a simulated WAN to simulated Web and BitTorrent peers.

The BitTorrent users are assigned to the first CMs (i.e., beginning at CM1 through CM*numberBTs*). The Web users are assigned to the CMs following those assigned to BitTorrent users. In all simulations, 150 CMs generated web-like traffic.

The emulated web users generate requests to web servers (located at nodes S-1 through S-x) following the model described in [32] using the parameters shown in figure 2. The remaining CMs do not generate application data but they transmit periodic DOCSIS management messages to the CMTS. Refer to [30] for further details of the DOCSIS model. To realistically model the impact of BitTorrent during peak usage times, the background web traffic was calibrated to consume approximately 60% of the upstream capacity when no competing BitTorrent sources are active.

A. The BitTorrent Traffic Model

The BitTorrent traffic model consists of a set of TCP traffic sources and sinks placed at a configurable number of selected

CMs. The BitTorrent peers of these CMs are located on the nodes labeled BTServer-j. The aggregate measurement results showed that it is common for BitTorrent clients operating over cable networks to have 3-4 active duplex connections that transfer the majority of the data. Our model establishes 3 concurrent connections between the client and three BitTorrent peers that are randomly selected from the BTServer-j set. We use the *ns-2* TCP/Sack model which generates traffic in one direction. To model a full duplex TCP traffic, we establish two one-way TCP/Sack connections, one in each direction. We refer to these matching connections as TCP connection pairs. This approach is operationally different than true BitTorrent duplex flows because it generates bi-directional standalone ACK streams. Nevertheless, we argue that it is the volume of upstream traffic and not the fact that it carried in duplex flows in BitTorrent that is the primary impact on competing flows.

An exponential traffic generator is attached to each TCP traffic source. Previous research has found that the average torrent size is about 760 Mbytes [25]. The generator is configured to send an average of 1 Gbyte of data during its *on* state. When the transfer is complete, the generator pauses for a short time (1 second on average) and then begins the next download. The parameters associated with the exponential source are the packet size, the burst time, the idle time and the rate. During the *on* state a BitTorrent source sends at *rate* bps for *burst time*. We have implemented an adaptation algorithm that dynamically adjusts the rate parameter which in turn adjusts the traffic generator's TCP throughput.

To model variations in the speed of the local access networks of the BitTorrent peers, the upstream and downstream link capacity between between nodes BT_gateway and BTServer$j$ were randomly selected using a uniform distribution in the range of [64 Kbps,1 Mbps] and [1 Mbps,6 Mbps] respectively. These ranges capture the range of likely Internet access broadband connection speeds. The link propagation delay was also selected randomly in the range [20,100] milliseconds. The queue size in both directions at the BTServer$j$ nodes was uniformly distributed between [20,40] packets.

B. BitTorrent traffic adaptation algorithm

When a simulated BitTorrent connection is created, it is endowed with a target asymmetry ratio which is the desired ratio of upstream to downstream throughput. These values are randomly selected with a mean of approximately 1.7. Each direction is configured with an initial rate randomly selected between 50.0 Kbps and 500 Kbps. These rates govern the rate at which the generators attempt to inject data into the simulated network, but TCP feedback dynamics control the actual rate at which data is injected.

Using the *ns-2* TCL scripting support, a TCL control function is created for each upstream/downstream pair of TCP/Sack connections between a given BitTorrent client and a peer. The objective of the adaptation algorithm is to adapt the presently configured sending rate of a TCP connection pair to meet a target upstream to downstream throughput ratio. Periodically (we found five seconds to be satisfactory), the control algorithm runs and computes the observed upstream and downstream throughput and the observed asymmetry ratio for the preceding five seconds.

If the observed asymmetry ratio exceeds the target asymmetry ratio, there is excess throughput in the upstream direction. In this case the configured downstream throughput is multiplied by 1.125. If this causes the configured downstream throughput to exceed the maximum downstream throughput, it is clamped to the maximum downstream throughput, and the configured upstream throughput is divided by 2. Analogous adjustments are performed when the observed asymmetry ratio is too small.

C. Assessing BitTorrent Impact

To show the impact that BitTorrent users have on other users that are not running BitTorrent, we captured two application level metrics between machines not generating BitTorrent traffic. One metric estimates web browsing performance and the other estimates VoIP performance.

1) Assessing web browsing quality: It was reported in [33] that response times longer than one second are likely to exceed the threshold of human perceived satisfaction and that users become frustrated when page response time exceeds 10 seconds. We have developed a performance monitor in *ns-2* that monitors the time it takes to download an average size web object (roughly 15 Kbytes according to [32]) from a server. The typical web page consists of approximately 10 objects [32]. Based on these results, we define a performance guideline which assumes a network is likely to exceed the typical human user tolerance for poor performance when the average object download time exceeds 1 second.

The CM labeled test client 1 in figure 1 is used to monitor web response times (WRT). Periodically, test client 1 requests a simulated web object of 15 Kbytes in size from web server $S-1$. The WRT application located at the test client 1 CM obtains a response time sample for each iteration.

2) Assessing VoIP Quality: VoIP quality is determined by a complex relationship involving latency, loss, codec capability, and the jitter buffer size. The E-model is a mathematical model that captures this relationship to provide an estimate of call quality (referred to as the Mean Opinion Score or MOS) [34]. However, there are several 'rules-of-thumb' that provide useful guidelines. One well known rule-of-thumb requires that in order for a call to be 'toll quality', the 'mouth-to-ear' latency must be less than 150 milliseconds [35]. Further, telephony is considered unusable if the latency exceeds 400 milliseconds. The affect of packet loss on call quality is highly dependent on the codec, in particular on the packet loss concealment (PLC) technique. If PLC is not used, loss rates over 1% will prohibit toll quality calls [35].

For the results described in this paper, we assess VoIP quality using the maximum mouth-to-ear (MTE) guidelines. We approximate the average MTE delay as the sum of the average one-way packet latency and jitter. Our rationale is that an optimally configured VoIP client adjusts its playout buffer to match the level of the jitter observed over the path. We set the maximum tolerated MTE delay to be 150 milliseconds. Since the metric does not take into account impairment caused by packet loss, the assessment is a conservative performance guideline.

The CM labeled test client 2 is configured to send a periodic stream of packets in a manner similar to a VoIP flow. A CBR traffic agent was configured on test client 2 to send 350 bytes every 0.05 seconds. The bandwidth consumed is similar to the bandwidth consumed by a G.711 VoIP flow. Various measures were obtained from this flow including the latency, jitter, and loss rate. The jitter was computed by taking the average of per packet latency differences.

D. Experimental configuration

The DOCSIS simulation model settings are summarized in figure 2. Both piggybacking and concatenation were enabled. The physical layer bit rate of the upstream channel was set to 5.12 Mbps and the downstream channel set to 30.34 Mbps. Service rates were unconstrained so that the CMTS and each CM could send as fast as the channel bit rate and the DOCSIS MAC protocol allowed. Experiments consisted of five runs with each run lasting 500 seconds of simulated time.

E. Comparison with BitTorrent traces

Table VII shows the mean bandwidth consumed by simulated BitTorrent users in the upstream and downstream directions. It can be observed that the adaptation algorithm is successful in producing asymmetry ratios that are consistent with those observed in the traces. Because the simulated BitTorrent users are not rate limited, one might expect significant differences in throughput. However, the measured results in Table IV suggest an average upstream consumption by BitTorrent of 350 Kbps. The simulations show that performance is comparable when approximately 6 BitTorrent users are active. Therefore, we feel that the essential characteristics of workloads produced by the simulation are consistent with those observed in the traces.

BT-CMs	Upstream (bps)	Downstream (bps)	Upstream: Downstream
2	497676	250405	1.9
10	224258	134298	1.67
20	123769	89508	1.38
30	85677	78914	1.1
40	65483	61759	1.1

TABLE VII
PER HOST BITTORRENT THROUGHPUT

F. Channel utilization and BitTorrent performance

Figure 3 shows the upstream and downstream channel utilization as a function of the number of BitTorrent CMs. The results suggest that the upstream channel saturates when approximately 15 BitTorrent CMs are active. The downstream utilization never exceeds 31%.

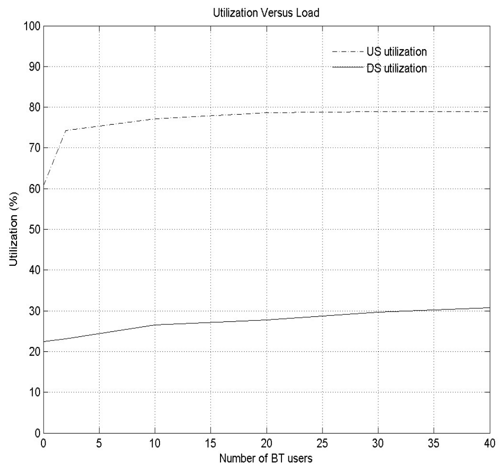


Fig. 3. Channel utilization

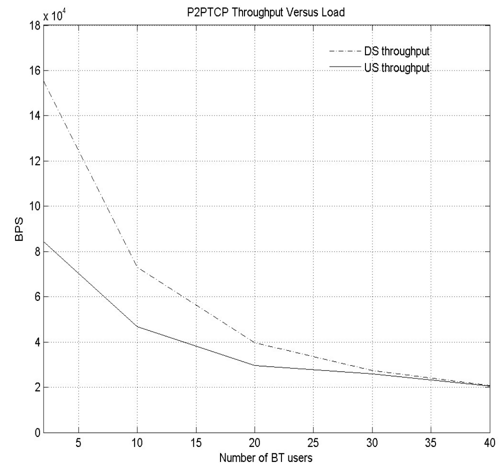


Fig. 4. BitTorrent Performance

Figure 4 shows the average TCP connection throughput in the upstream and downstream directions for the simulation experiments. The rate adaptation algorithm achieves approximately the desired level of asymmetry between the upstream and downstream BitTorrent throughput (the *asymLevel* was set to approximately 1.7). The ratio converges to a value of 1 (i.e., symmetric throughputs in the upstream and downstream directions) under extreme loads when the bottlenecked upstream channel effectively rate limits the downstream channel.

G. Impact on non-BitTorrent users

Figure 5 illustrates the mean of the web response times gathered by the CM identified as *Test client 1* in figure 1. The web response time statistic increased from a value of 0.25 seconds when no BitTorrent users were active to 0.65 seconds when 15 BitTorrent users were active. This suggests that 15 BitTorrent users can cause a drop in performance by a factor of 2.5. When the number of BitTorrent users exceeds 30 performance degrades beyond the 1 second metric threshold.

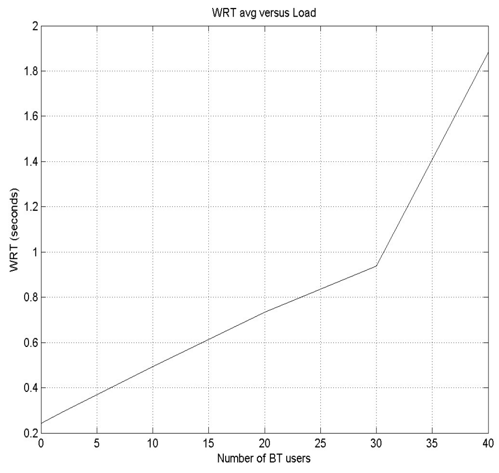


Fig. 5. Web response time

Figure 6 illustrates the latency and the jitter associated with an upstream VoIP-like flow. The latency more than triples when 10 BitTorrent users become active. The jitter increases from a value of 0.01 seconds to 0.06 seconds when 20 BitTorrent users are active. The effective mouth-to-ear delay exceeds the performance threshold of 150 milliseconds when 15 or more BitTorrent users are active.

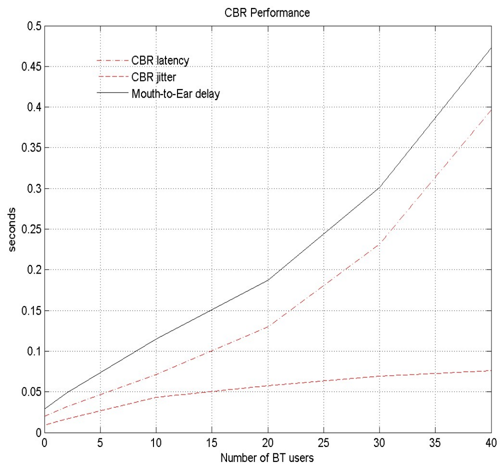


Fig. 6. VoIP performance assessment

V. Conclusion

In this paper we have assessed the impact of BitTorrent on DOCSIS cable networks. Specific contributions of our work include:

- We have demonstrated that downloads of widely traded torrents can produce large volumes of upstream traffic.
- We have developed an empirically derived model of such traffic.
- We have implemented a BitTorrent traffic generator in *ns-2* that produces traffic loads having characteristics consistent with traced traffic.
- We have provided simulation-based evidence that a small number of BitTorrent users in a DOCSIS-based network can significantly impact other users who are not running BitTorrent.

Our study focused on current DOCSIS deployments that involve upstream channel rates of 5.12 Mbps. The simulation results were based on a configuration that did not impose service rate restrictions on the CMs. We repeated the experiment with upstream services rates ranging from 256 Kbps to 2 Mbps and with corresponding downstream service rates ranging from 1 Mbps to 10 Mbps. There was no noticeable improvement until the upstream service rates were 256 Kbps or lower. The improvements were evident only in runs involving fewer than 10 BitTorrent users because, as shown in Table VII, BitTorrent consumes less than 256 Kbps when there are 10 or more competing users.

Cable system operators are moving towards deployments based on DOCSIS 3.0 equipment that will provide greater than 100 Mbps upstream channel rates. Applications such as BitTorrent that reward users for distributing content at high data rates will continue to consume a disproportionate share of available bandwidth. This fact motivates our continued research in developing innovative approaches for managing bandwidth in current and future broadband access networks.

In ongoing work we are seeking to develop algorithms that can reliably identify high bandwidth data flows associated with peer to peer protocols and manage their bandwidth consumption. Bandwidth management strategies under consideration involve the development of autonomic support in the DOCSIS MAC protocol for dynamically configuring the backoff range of each CM based upon the time varying characteristics of the network.

Acknowledgment

This work was supported in part by the ITR Program of the National Science Foundation under award ACI-0113139.

References

[1] C. Cable Television Labs Inc., "DOCSIS primer." [Online]. Available: http://www.cablemodem.com/primer/
[2] "Data-over cable service interface specifications- radio frequency interface specification sp-rfiv2.0." [Online]. Available: http://www.cablemodem.com/specifications/ specifications20.html
[3] "Entropia." [Online]. Available: http://www.entropia.com
[4] F. Dabek, M. Kaasheok, D. Karger, R. Morris, and I. Stoica, "Wide-area cooperative storage with cfs," in *Proceedings of SOSP01*, October 2001.
[5] S. Iyer, A. Rowstron, and P. Druschel, "Squirrel: A decentralized peer-to-peer web cache," in *Proceedings of PODC02*, October 2002.
[6] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network," in *Proceedings of ACM SIGCOMM 2001*, 2001.
[7] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishman, "Chord: A scalable peer-to-peer lookup protocol for internet applications," in *Proceedings of ACM SIGCOMM 2001*, 2001.
[8] I. Ivkovic, "Improving gnutella protocol: Protocol analysis and research proposals." [Online]. Available: http://www9.limewire.com/download/ivkovic_paper.pdf

[9] J. Liang, R. Kumar, and K. Ross, "Understanding kazaa." [Online]. Available: http://cis.poly.edu/ ross/papers/

[10] N. Leibowitz, M. Ripeanu, and A. Wierzbicki, "Deconstructing the kazaa network," in *Proceedings of the Third IEEE Workshop on Internet Applications / WIAPP'03*, June 2003.

[11] "Bittorrent documentation: Protocol." [Online]. Available: http://www.bittorrent.org/protocol.html

[12] B.Cohen, "Incentives build robustness in bittorrent," in *Proceedings from the Workshop on Economics of Peer-to-Peer Systems*, Berkeley CA, May 2004. [Online]. Available: http://www.bittorrent.org/bittorrentecon.pdf

[13] P. Rodriguez and E. Biersack, "Dynamic parallel access to replicated content in the internet," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, August 2002.

[14] S. Saroiu, P. Gummadi, and S. Gibble, "A measurement study of peer-to-peer file sharing systems," in *Proceedings of Multimedia Computing and Networking 2002*, 2002.

[15] T. Ng, Y. Chu, S. Rao, K. Sripanidkulchai, and H. Zhang, "Measurement-based optimization techniques for bandwidth-demanding peer-to-peer systems," in *Proceedings of Infocom 03*, July 2003.

[16] M. Ripeanu, "Peer-to-peer architecture case study: Gnutella network," 2001.

[17] M. Ripeanu, I. Foster, and A. Iamnitchi, "Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design," *IEEE Internet Computing Journal*, vol. 6, no. 1, 2002.

[18] L. Gao, J. Kurose, and D. Towsley, "Efficient schemes for broadcasting popular videos," in *Proceedings of Multimedia Systems*, vol. 8, no. 4, July 2002.

[19] Z. Ge, D. Figueiredo, S. Jaiswal, J. Kurose, and D. Towsley, "Modeling peer-to-peer file sharing systems," in *Proceedings of IEEE Infocom 2003*, July 2003.

[20] F. Clevenot and P. Nain, "A simple fluid model for the analysis of squirrel," in *Technical Report, Inria RR-4911*, 2003.

[21] F. Clevenot, P. Nain, and K. Ross, "Stochastic fluid models for cache clusters," in *Technical Report, Inria RR-4815*, 2003.

[22] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "The bittorrent p2p file-sharing system: Measurements and analysis," in *Proceedings of International Workshop on Peer-To-Peer Systems*, February 2005.

[23] K. Skevik, V. Goebel, and T. Plagemann, "Analysis of bittorrent and its use for the design of a p2p based streaming protocol for a hybrid cdn," Department of Informatics, University of OSLO, Tech. Rep., 2004.

[24] e. A. M. Izal, "Dissecting bittorrent: Five months in a torrent's lifetime," in *Proceedings of Passive and Active Measurements Workshop*, April 2004.

[25] A. Bellissimo, B. Levine, and P. Shenoy, "Exploring the use of bittorrent as the basis for a large trace repository," in *Technical report 04-41*, Department of Computer Science, University of Amherst, June 2004.

[26] A. Bharambe, C. Herley, and V.Padmanabhan, "Analyzing and improving bittorrent performance," in *Microsoft Research Technical Report MSR-TR-2005-03*, Februrary 2005.

[27] D. Erman, D. Ilie, A. Popescu, and A. Nilsson, "Measurement and analysis of bittorrent signaling traffic," in *Proceedings of Nordic Teletraffic Seminar*, August 2004.

[28] D. Erman, D. Ilie, and A. Popescu, "Bittorrent session characteristics and models," in *Procedings of HETNETS05*, July 2005.

[29] "The *ethereal* program." [Online]. Available: http://www.ethereal.org

[30] J. Martin and J. Westall, "Validating an 'ns' simulation model of the DOCSIS protocol." [Online]. Available: http://people.clemson.edu/ jmarty/papers/docsis-model.pdf

[31] T. N. Simulator. [Online]. Available: http://www-mash.cs.Berkeley.EDU/ns/

[32] P. Barford and M. Crovella, "Generating representative web workloads for network and server performance evaluation," in *Proceedings of Performance '98 / ACM SIGMETRICS '98*, July 1998.

[33] A. Bouch, A. Kuchinsky, and N. Bhatti, "Quality is in the eye of the beholder: Meeting users' requirements for internet quality of server," *ACM CHI Letters*, vol. 2, no. 1, April 2000.

[34] I. T. Union, "The e-model, a computational model for use in transmission planning," December 1998.

[35] D. Vleeschauwer, J. Janssen, G. Petit, and F. Poppe, "Quality bounds for packetized voice transport," in *Alcatel Telecommunications Review*, April 2000.

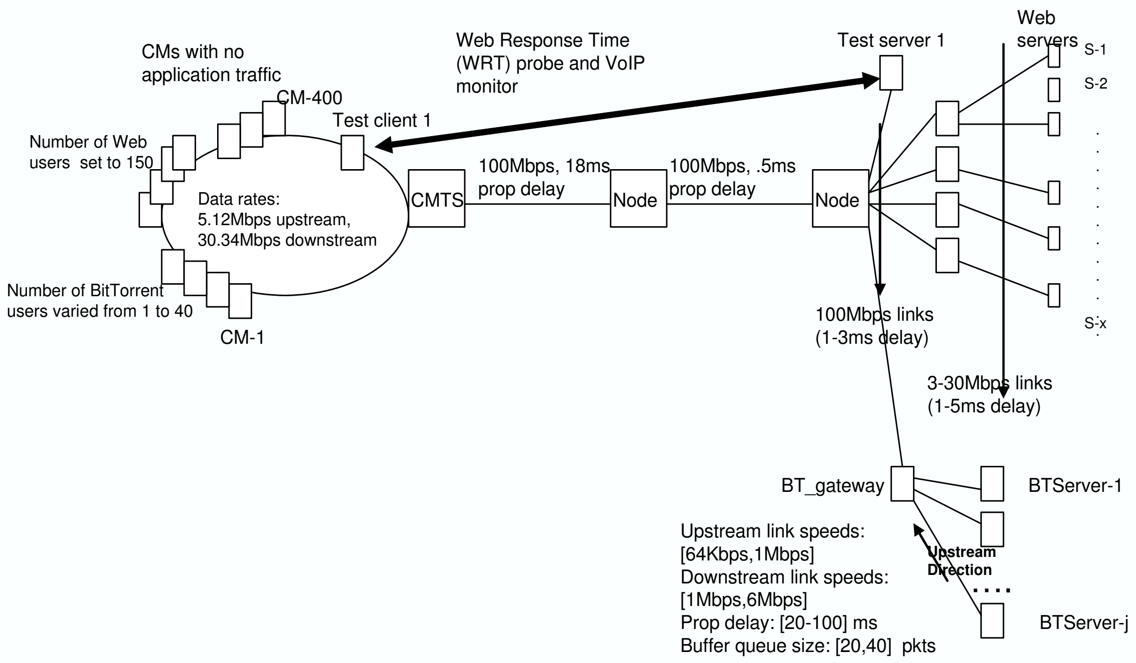

CMs with no application traffic
CM-400
Web Response Time (WRT) probe and VoIP monitor
Test server 1
Web servers
S-1
S-2
S-x
Test client 1
Number of Web users set to 150
Data rates: 5.12Mbps upstream, 30.34Mbps downstream
CMTS
100Mbps, 18ms prop delay
Node
100Mbps, .5ms prop delay
Node
Number of BitTorrent users varied from 1 to 40
CM-1
100Mbps links (1-3ms delay)
3-30Mbps links (1-5ms delay)
BT_gateway
BTServer-1
Upstream link speeds: [64Kbps,1Mbps]
Downstream link speeds: [1Mbps,6Mbps]
Prop delay: [20-100] ms
Buffer queue size: [20,40] pkts
Upstream Direction
BTServer-j

Fig. 1. Simulated network