UNDERSTANDING THE IMPACT OF BITTORRENT ON CABLE NETWORKS

Jim Martin
Clemson University

Terry Shaw
CableLabs

Abstract

BitTorrent is a peer-to-peer (P2P) protocol for sharing media (audio, video and software) files that accounts for 18% of traffic on cable high-speed data networks. In order to better understand the network dynamics of traffic associated with this protocol, we analyzed roughly 1 Gbytes of network trace data from BitTorrent clients running in three different environments. Based on this data, we performed a simulation-based analysis of a DOCSIS®-based cable network involving a varying number of simulated BitTorrent users and web browsing users. Our results suggest that as few as 10 BitTorrent clients can double the access delay experienced by other non-BitTorrent users.

INTRODUCTION

BitTorrent is a peer-to-peer (P2P) protocol that provides scalable file sharing capabilities. As in other P2P systems, BitTorrent provides an overlay network that runs over the Internet. BitTorrent is unique from other P2P protocols in several aspects. The most notable of these are: 1) the use of swarming; and 2) the use of incentives to prevent free-riding. BitTorrent, frequently the dominant application in a network, poses a dilemma for cable network service providers. In general, P2P applications are broadband applications that contribute to the demand for high speed broadband access. This usage is driven by younger Internet users. In a recent study, it was estimated that about 58% of American teenagers with broadband access have downloaded audio and/or video files from the Internet using P2P networks[1]. However, BitTorrent can consume tremendous amounts of bandwidth in both the upstream and downstream directions. The primary incentive used by BitTorrent to prevent free-riding is that the protocol mandates that a file-downloader must make some content available for upload. One study from several cable operators found that 18% of all traffic is BitTorrent [WSJ05]. The government and the cable industry are addressing the evolving economic and policy issues. In this study we explore the impact that BitTorrent users can have on a cable network. We provide insight to this issue by presenting an analysis of BitTorrent involving live network measurement and simulation[2].

In addition to file sharing, P2P has been used for grid computation [Entropia], storage [DKK+01], web caching [IRD02] and directory services [RFH+01, SMK+01]. BitTorrent, however, is designed specifically for file distribution. The algorithms contained in BitTorrent were heavily influenced by previous P2P protocols and applications. We briefly overview two such protocols, Gnutella and Kazaa, before presenting BitTorrent.

[1] In a recent study, it was estimated that about 66% of American teenagers have downloaded audio and/or video files from the Internet. It was also found that 30% of teenagers admit to currently using P2P networks and 28% admit to using P2P networks in the past [PIP05].

[2] The observations that we report represent preliminary work. We expect our findings and conclusions to evolve as the study progresses.

Gnutella

Gnutella is an overlay network superimposed on the Internet [MAR+04,IVK]. Unlike the centralized Napster approach, Gnutella is decentralized because meta-data (i.e., the indexes of where files are located) is distributed throughout the network. At initialization time, a Gnutella client accesses a host server to obtain a list of peers currently in the network. After this step, peers learn about the network and available content using Ping and Pong messages. Once a client has identified a file to download, it sends the request to the network using a query message. Query replies will contain information needed for the client to download the file.

Kazaa

Few details of the Kazaa protocol are available. Several 'black box' measurement studies do however provide some insight [LKR04,LRW03]. There are two types of nodes: Super Nodes (SN) and Ordinary Nodes (ON). Each ON must be told or must learn the address of at least one SN. SNs maintain lists of other SNs that can be thousands large. Each SN maintains records for files located at the ONs in its domain. ONs contact their SN on initialize, providing metadata that describes the content located at the ON. When an ON requests a file, the SN looks locally and also propagates the request to other SNs in the overlay network. To improve performance, files can be stored in fragments on multiple nodes and they can be transferred in parallel.

BitTorrent

While other P2P systems provide distributed object location in addition to file downloading, BitTorrent assumes that the user has a URL of the file to be downloaded. BitTorrent is therefore a protocol for distributing files. Files are broken up into chunks and downloaded in pieces. This concept is borrowed from [RB02] whose authors experimented with several parallel access schemes to download a file from multiple servers concurrently. BitTorrent files are represented by a torrent which is a meta-information file that identifies the chunks associated with a file and a tracker address that knows about the file. Downloaders, also referred to as clients or peers, download the torrent that is available on a web site. The download engages the BitTorrent client software that was installed on its machine which contacts the tracker that was identified in the torrent to obtain a number of IP address/port pairs corresponding to peers that have one or more chunks of the file. The client periodically reports its state to the tracker (about every 30 minutes). 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. When a client first connects to a tracker, the tracker attempts to reverse connect to it. If it succeeds, the client is added to the list of peers for the torrent maintained by the tracker. This ensures that the peers in the client's peer set will accept incoming connections. At least one of the peers must contain all of the file. Once a peer has downloaded the entire file, it becomes a seed and provides chunks of the file to lechers for free. The system is self-scalable in that as the number of downloaders increase, the number of nodes that can function as seeds also increase.

BitTorrent breaks a file into chunks, or pieces. To enhance performance, BitTorrent further breaks pieces into small (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. The client contains a piece selection algorithm that decides the next piece to obtain. The algorithm has three modes of operation: starting, downloading, or finishing. When the client starts it has nothing to upload. The algorithm randomly selects the initial pieces. Once one or more pieces are received, the algorithm switches to a rarest first algorithm. Based on information learned from its peers, the client chooses to download the chunk that is least available. This ensures that peers will have pieces that they will need. Once the peer has downloaded all but the last few pieces, it might experience significant delay if the remaining pieces are located at nodes that are available only over low-bandwidth paths. To avoid this, it sends requests for these sub-pieces to all peers.

A peer actively controls its downstream bandwidth consumption by granting upload requests to peers from which it is currently downloading a piece. In essence, it provides a 'tit-for-tat' bartering approach to distributed resource management. By choosing to upload to peers that provide the best download rates, freeloaders will perform poorly. A peer effectively 'chokes' another peer by denying download requests from a peer. Snubbing is when a peer is choked by all peers with which it was formerly downloading.

A BitTorrent peer uploads only to a limited number of peers (usually four). Peers monitor download rates and decide who to choke over time intervals of every ten seconds (note that some implementations might use a different rechoke period). When the next piece is to be downloaded, the peer that has been performing the worst will likely be replaced by a higher performing peer. To probe the network in search of better performing peers, BitTorrent supports an 'optimistic unchoke.' Every third rechoke period, it unchokes a peer by uploading to it regardless of its download rate. Therefore, a BitTorrent client is allowed to upload to a maximum of five peers, four that implement tit-for-tat bartering to maximize downstream throughput and a fifth that probes the network searching for better performing peers.

BitTorrent is unique from other P2P protocols in several aspects: 1) the use of swarming; 2) the use of incentives to prevent free-riding. BitTorrent, frequently the dominant application in a network, poses a dilemma for cable network service providers. It is a broadband application which contributes to the demand for high-speed broadband access. However, BitTorrent can consume tremendous amounts of bandwidth. In this study, we explore the impact that BitTorrent users can have on a cable network. We provide insight to this issue by presenting an analysis of BitTorrent involving live network measurement and simulation. This paper is organized as follows. First, we overview related performance studies of peer-to-peer systems. We then describe our modeling efforts: a measurement study designed to characterize bandwidth consumption of BitTorrent clients and a simulation analysis designed to assess the impact of BitTorrent on an HFC cable network. We end the paper with conclusions and our future directions.

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 [SGG02, NCR+03, RIP01] or on evaluating protocol issues [RFK+01, SMK+01,GKT03]. There have been several analytic models proposed. Several notable efforts have been based on

queuing theory [GFJ+03] and on fluid flow models [CN03,CNR03].

The existing studies of BitTorrent indicate that the protocol is indeed scalable and robust [PGE+05, SGP04, IUB+04, BLS04, BHP05]. In [IUB+04] 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. Upon completion, the client remained on line as a seed for 13 additional hours. During this period, roughly 90 leechers were served (but only 4 at a time while the others are choked). 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.

In [PGE05], the authors examined the access of 60000 files and find an average download bandwidth of 240 Kbps and that only 17% of the nodes stay on line for one or more hours after they complete the download. The authors suggested that there might be a shortage of seeds in typical usage scenarios. The authors assessed performance based on availability, integrity, responsiveness to flash crowds and on download performance. To obtain an availability data point, they tracked the activity associated with a popular file from its initial offering until when it died (i.e., when it is no longer available). During the file's three month lifetime, 90155 peers downloaded at least one piece. Of the 53883 peers that were not behind firewalls, only 17% have an uptime longer than 1 hour after they finish downloading. By trying to upload corrupt files but failing due to the moderator's inspection, the authors deduce the BitTorrent network is relatively pollution free. By correlating system activity with the introduction of a new file (The Lord of The Rings III) that caused a flash crowd effect, the authors observe that the system remained stable.

In [QS04], the authors applied fluid flow modeling techniques to BitTorrent to obtain a model that predicts average number of seeds, downloaders, and download time.

To assess the effectiveness of BitTorrent, the authors in [BP05] defined link utilization as the aggregate capacity of all nodes participating in the overlay network. Fairness is defined in terms of the relative number of pieces served by a node to the number downloaded. Another aspect of fairness is if all nodes, in particular seeds, are equally utilized. Finally, a diversity measure assesses how effectively pieces are distributed throughout the overlay network.

In [BLS04] the authors analyzed BitTorrent activity over a four month period involving thousands of torrents. They observed a mean and median file size of 760 and 600 Mbytes, respectively. The mean and median session duration was 13.25 and 8.4 hours, respectively. The authors plan on developing a BitTorrent application for distributing large network traces to the research community.

In [SGP04] the authors instrumented the standard BitTorrent client and monitored the activity associated with a popular tracker. They observed an average download rate of 200 Kbps among all clients that were not behind a firewall. The authors proposed a streaming content delivery network based on a BitTorrent-like protocol. They pointed out that two areas that need improvement for this application are better protection from freeloaders and better support for nodes behind firewalls.

MODELING BITTORRENT

To the best of our knowledge no one has characterized in detail the workload generated by a BitTorrent client. As this was required for our simulations, we performed an analysis involving network traces of real BitTorrent clients.

BitTorrent Traces

We obtained three sets of traces. They are summarized in Table 1. All involved downloading the same torrent (a 4.3 GByte file actively traded on the BitTorrent network) using version 4.04 of the BitTorrent client at [BT]. The client machine was a WindowsXP machine. According to the statistics at the tracker, the torrent consistently had hundreds of downloaders and tens of peers. In our experiments, firewalls prevented remote peers from initiating TCP connections with the client. The BitTorrent client allows the user to specify the maximum bandwidth to be consumed by the host. We set this to the maximum value of 45 Mbps.

We refer to the three data sets as Set1, Set2 and Set3 respectively. Two of the sets (Set1 and Set2) are from a machine on Clemson University's campus and the other set (Set3) is from a machine located at the author's home. To help manage P2P traffic generated by students, Clemson deployed a Packeteer bandwidth management device that limits individual TCP sessions (except for web traffic) to a maximum rate of 64 Kbps[PACK]. Set1 traces were subject to this control. For Set2, we configured the Packeteer device so that packets associated with the IP address of the BitTorrent client had high priority and were not subject to rate limits. The speed of the access link connecting Clemson's gigabit Ethernet campus network to the Internet is 100 Mbps. Set 3 traces were from a machine connected to the Internet by DSL with service rates of 1.5 Mbps downstream and 256 Kbps upstream.

To calibrate our trace and analysis methodology we performed a set of TCP transfers between the DSL connected machine and the host located on campus. We transferred 500 Kbytes ten times from the DSL connected machine to the campus machine (i.e., upstream from the perspective of the BitTorrent client). Then, we transferred 500 Kbytes ten times from campus to the DSL connected machine. For these traces, and for all traces described in this report, all TCP packets sent and received by the DSL connected host were captured using tcpdump[TCPD]. Table 2 summarizes the calibration results. The top row shows the average of a set of performance measures collected from each

of the 10 upstream transfers and the bottom row shows the downstream results. The columns labeled 'AvgCx BW', 'Avg TCP RTT', and 'Avg TCP LR' indicate the average TCP throughput, round trip time and loss rate, respectively, experienced by the 10 TCP connections. The column labeled 'Bottleneck Link Speed Estimate' examines the interarrival times between ACK packets that arrive at the client and, based on a packet-pair algorithm, estimates the bottleneck link speed in the upstream direction [KLD+04]. Since this flow will not be classified as P2P traffic by the Packeteer device, we expect that the bottleneck link speed in both the upstream and downstream directions to be the DSL access link speeds. The results from Table 2 confirm this although the estimated bottleneck bandwidth is roughly 20% lower than the actual. This is due to framing overhead that is not included in the bandwidth estimate. The TCP throughput was slightly less than the bandwidth estimate. This is because of additional overhead due to TCP. The uncongested RTT over the path based on a 56 byte Ping probe is roughly .11 seconds. The mean TCP RTT of 1.16 seconds observed in the upstream connections along with no packet losses implies that sustained queuing exists over the path, presumably at the DSL router's upstream queue. Further analysis shows that the TCP congestion window stays at a maximum value of 32 Kbytes that was set by the receiver's advertised window. The transmission time of 32 Kbytes over a 256 Kbps link is roughly 1 second making the observed RTT reasonable. The algorithm that we use to obtain the average TCP RTT is based on previous work [MNR03]. It requires a send side trace and consequently we were not able to obtain a RTT result for downstream connections.

The first two traces of each set were taken while the client was downloading the torrent. The third trace was taken once the client became a seed. While downloading, data flowed in both directions over most of the TCP connections. Our analysis tools operate only on the data sent in the upstream direction. While seeding, data flowed primarily in the upstream direction. Each trace in Set1 and Set2 contains about 400000 packets comprising roughly 60 unique connections of which 20 to 30 were active throughout the lifetime of the trace. The Set3 traces each contained about 60000 packets comprising slightly more (40-60) active connections while downloading and significantly more once the client became a peer (180 connections). More than half of the connections in all sets transferred a very small amount of data (presumably BitTorrent messages).

Table 3 summarizes the observed behavior of upstream TCP transfers contained in the Set1 traces. The columns labeled 'Avg Aggregate US BW' and 'Avg Aggregate DS BW' show the aggregate bandwidth consumed in the upstream and downstream direction, respectively, by the client during the trace. As with all traces described in this report, more bandwidth is consumed in the upstream direction than in the downstream. The column labeled 'US/DS BW ratio' captures this. The mean ratio for the 4,5,7,8 traces was 2.5 and increased to 24 once the client became a peer (i.e., traces 6 and 9).

The Set1 results clearly show the impact of the Packeteer device. Without the rate limits imposed by Packeteer we would expect to see a bottleneck bandwidth estimate in the 1 to 5 Mbps range (i.e., typical broadband access downstream service rates). As expected, with the Packeteer device engaged, the estimated

bottleneck bandwidth was close to the configured service rate of 64 Kbps. The transmission time of a 1500 byte packet over a 64 Kbps link is .188 seconds making the observed TCP RTT reasonable. The average upstream and downstream bandwidth consumption for each trace is significant, especially once the client becomes a seed. The average upstream TCP throughput varied widely although never exceeded 50,000 bps.

The last four columns in Table 3 indicate the number of concurrent transfers. We define an active connection to be a connection that sends data in a one-way direction at a rate that is greater than a threshold level. We used thresholds of 10 Kbps, 40 Kbps and 100 Kbps. With Set1, since TCP connections are limited to 64 Kbps, we did not see any connections with a transfer rate greater than 100 Kbps. We see many low rate (10 Kbps) flows and smaller number of flows that consume at least 40 Kbps. Once the client became a peer, we observed 28 concurrent 10 Kbps upstream flows and 22 concurrent 40 Kbps flows.

In the Set2 traces we see an order of magnitude increase in the bottleneck link speed which confirms that the Packeteer service rates are no longer in place. The bottleneck link speed ranged from a minimum of 256 Kbps to 10 Mbps. The RTT is reasonable except for Trace 22 which had an RTT of .802 seconds. It's unclear why this value is so high. We see roughly 5 to 7 active high speed flows in the upstream direction. In the downstream direction, the number of active high speed flows was very low. Presumably, peers were not capable (or willing) of sending at very high rates.

The Set3 traces also seem reasonable although the upstream bottleneck link estimates were off by almost 50%. The error associated with packet-pair estimate increases if the senders are not constantly sending. The average upstream TCP throughput is low because the connection was bidirectional and but there were periods of time when data was flowing in only one direction. This "idle" time is reflected in the TCP throughput estimate. The TCP RTT matches the value we saw in the calibration tests (refer to Table 2). Once the client becomes a seed the RTT becomes significantly higher. However, the TCP loss rate gets lower. Further investigation is required to explain this.

Simulation Analysis

In prior work we implemented a simulation model of a DOCSIS network using the NS-2 simulation tool [MW05, ns2]. We have extended this model to support a simple BitTorrent traffic model. Figure 1 illustrates the simulated network. There are 200 cable modems (labeled CM-1 through CM-n) that share one downstream channel and one upstream channel. Fifty CMs are configured with the client side of a web traffic model. A simulated web user at the CM generates requests to web servers (nodes S-1 through S-x) following the model described in [BC98]. Figure 2 identifies the DOCSIS network configuration parameters and the web model settings that were used in the experiments. Refer to [MW05] for further details of the model.

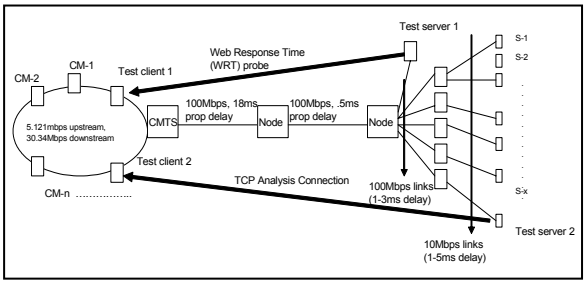

Figure 1. Simulation Network Model

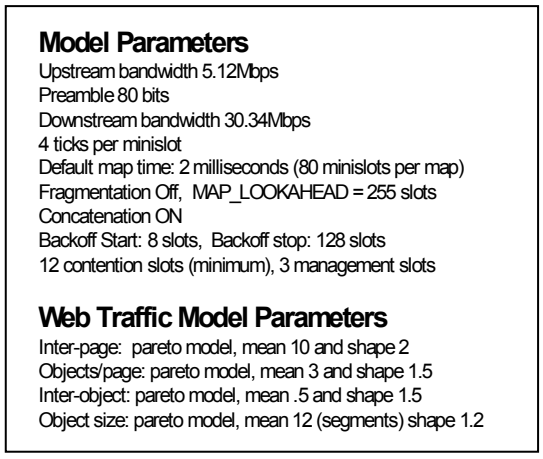

Figure 2. DOCSIS Network and Web Traffic Model Simulation Parameters

A varying number of the CMs are configured with the client side of a BitTorrent traffic model. In the model one or more TCP connections transfer data in the upstream direction and one or more additional TCP connections transfer data in the downstream direction. An on/off traffic source is attached to each TCP sender associated with the client. For US traffic, the senders are located at the designated CM and interact with the TCP sinks located at the servers (S-1 through S-x). For DS traffic, TCP senders are located at the servers and the sinks are attached to the CMs. A BitTorrent source periodically transfers a large amount of data to the server. The message size is exponentially distributed with a mean of 1 Gbyte. This value is based on a measurement study that found an average torrent size of about 800 Mbytes [BLS04]. The off time of the traffic source is exponentially distributed with a mean of 2 seconds. To model *n* concurrent TCP connections at a BitTorrent client, we create n TCP flows between the server and the client but reduce the amount of data sent by each traffic source by a factor of *1/n.*

We ran four sets of simulation experiments. In each set, the number of simulated BitTorrent clients in the HFC network was increased from 0 to 50 in increments of 10 over 6 different runs. The four sets are summarized as follows:

Symmetric: For each BitTorrent client, there was one upstream and one downstream flow.

Downstream asymmetric: The total number of upstream BitTorrent file transfers is limited to five streams. We increase the ratio of downstream BitTorrent traffic to upstream traffic by increasing the number of downstream connections from 5 to 50.

Upstream asymmetric: The total number of downstream BitTorrent file transfers is limited to five streams while the number of upstream connections increases from 5 to 50.

Symmetric and parallel connections: For each BitTorrent client, instead of one upstream flow and one downstream flow, we adjust the traffic source accordingly and use four concurrent connections upstream and four concurrent connections downstream.

To calibrate our simulation model with observed behavior, we compare the symmetric simulation results (summarized in Table 6) with the results from measured traces 7 and 8 shown in Table 5. The average TCP connection throughput, the loss rates and the aggregate US bandwidth are similar. The most significant difference is that the aggregate downstream bandwidth is much higher in the simulation. This is due in part because our BitTorrent model does not capture the application 'tit-for-tat' dynamics, nor does the model support user specified rate limits that are available on most BitTorrent clients. However, the more significant reason for the asymmetry is

because our simulation network model assumes peers are connected to the network with very high speed links (10Mbps).

Summary of simulation results

For each of the four sets of simulation experiments, we obtain a number of performance measures that fall into two categories.

- Those that assess performance of CMs that are not running BitTorrent. We ran several network performance monitor applications on the Test client 1 and Test server 1 nodes. These nodes were not running the BitTorrent or the web traffic generators. For brevity we report only the results of a CBR flow between these two nodes. The objective of the monitor is to simulate a best effort VoIP flow and to monitor the UDP packet jitter, latency and loss. The CBR source is attached to the Test client 1 node and is configured to send a 350 byte message every .05 seconds (i.e., 56Kbps). Figures 3, 4 and 5 visualize these results for each of the four sets of experiments.

- Those that assess DOCSIS network dynamics. Figure 9 visualizes the mean collision rates reported by the CM's. Figures 6, 7 and 8 provide insight to how the CMs obtained upstream bandwidth. Figure 6 plots the percentage that contention requests were used for all packets (IP packets, fragments and management messages) that were sent upstream. Figure 7 plots the percentage of upstream packets that were sent in a concatenated frame. Figure 8 plots the average number of packets per concatenated frame.

Figures 3 through 9 plot the performance measures. Each figure contains four plots representing the results of the experiments. The following observations apply to all four sets:

- As expected, the load on the upstream channel increased linearly as the number of BitTorrent clients grew. Over a 5.12 Mbps upstream channel, as few as 10 BitTorrent clients caused the access delay experienced by other users to double. When 10 BitTorrent clients were active, the US channel was 55% utilized. This result was observed with no service rate limits and also when 384 Kbps upstream service rates were imposed.

- As the BitTorrent load increased, the majority of bandwidth requests was accomplished using concatenation with an average of 3.5 IP packets (primarily TCP acknowledgement packets) inserted into each concatenated frame. If concatenation was disabled, most bandwidth requests would rely on piggybacking which is not as efficient and would lead to significantly higher mean access delays.

We note the following differences between the experiments:

- The UDP packet jitter and latency is double for the symmetric, parallel connections compared to the symmetric, single connection runs (i.e., Figures 3 and 4). At the same time, the collision rate is lower by roughly 20% once the network becomes congested (Figure 9). More study is required to explain this behavior. However, the result suggests that BitTorrent's use of concurrent connections might have a greater detrimental impact on real-time traffic such as voice and video.

- All statistics confirm that performance is significantly worse when BitTorrent traffic is bi-directional rather than asymmetric.

- The downstream asymmetric configuration delivers upstream data by concatenation less often (by 50%) than the other sets. This is because the US utilization was lower than in the symmetric experiments.

CONCLUSIONS

The objective of this study was to gain insight in how varying levels of BitTorrent traffic on an HFC network impacts other CMs and on the dynamics of a DOCSIS network. Different parameters and model assumptions will change the results. In particular, the availability and locality of the torrent, peer behaviors, and network loads all determine the impact on the network and subsequently, on subscriber's perceived performance. Our goal was to obtain sufficient data permitting us to build a simulation model of a BitTorrent traffic generator. Our simulation model is most similar to the behavior observed in a residential network. We configured four concurrent upstream and downstream flows that consumed symmetric levels of bandwidth.

Based on simulation we have shown that a small number of BitTorrent users can impact other users, even when CMs are provisioned with low upstream service rates. Cable companies can no longer rely on static rules of thumb when provisioning an HFC network. This issue becomes more urgent if the provider plans to reliably support best effort VoIP and video applications. The provider must monitor network performance and adapt the network as needed.

There are many directions for future work including further development of our BitTorrent traffic model and subsequent analysis in cable environments. The impact that high-bandwidth applications such as BitTorrent have on other subscribers becomes more problematic as service rates increase. Therefore, we are developing adaptive bandwidth management techniques that implement fairness policies through a combination of performance and monetary incentives.

REFERENCES

[BC98] P. Barford, M. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation", ACM SIGMETRICS '98, July, 1998.

[BHP05] A Bharambe,C. Herley, V. Padmanabhan, "Analyzing and Improving BitTorrent Performance", Microsoft Research Technical Report MSR-TR-2005-03, Feb 2005.

[BITPR] BitTorrent Documentation: Protocol available at http://www.bittorrent.com/protocol.html

[BLS04] A Bellissimo, B. Levine, P. Shenoy, "Exploring the Use of BitTorrent as the Basis for a Large Trace Repository", Technical report 04-41, Department of Computer Science, University of Amherst, June 2004.

[BT] BitTorrent client and tracker software, http://www.bittorrent.com

[BTSPE] The BitTorrent Specification: http://wiki.theory.org/BitTorrentSpecification

[CN03] F. Clevenot, P. Nain, "A Simple Fluid Model for the Analysis of SQUIRREL", Technical Report, Inria RR-4911,2003.

[CNR03] F. Clevenot, P. Nain, K. Ross, "Stochastic Fluid Models for Cache Clusters", Technical Report, Inria RR-4815, 2003.

[COH03] B.Cohen, "Incentives Build Robustness in BitTorrent", Workshop on Economics of Peer-to-Peer Systems, Berkeley CA, May 2004. Available at http://www.bittorrent.com/bittorrentecon.pdf

[DKK+01] F. Dabek, M. Kaasheok, D. Karger, R. Morris, I Stoica, "Wide-area Cooperative Storage with CFS", SOSP01, Oct 2001.

[ENTRO]Entropia, http://www.entropia.com

[GFJ+03] Z. Ge, D. Figueiredo, S. Jaiswal, J. Kurose, D. Towsley, "Modeling Peer-to-Peer File Sharing Systems", IEEE Infocom 2003.

[GKT02] L. Gao, J. Kurose, D. Towsley, "Efficient Schemes for Broadcasting Popular Videos", Multimedia Systems, Vol 8, No4, July 2002.

[IRD02] S. Iyer, A. Rowstron, P. Druschel, "Squirrel: A Decentralized Peer-to-Peer Web Cache", PODC02, 2002.

[IUB+04] M Izal, et. Al., "Dissecting BitTorrent: Five Months in a Torrent's Lifetime",Passive and Active Measurements, April 2004.

[IVK] I. Ivkovic, "Improving Gnutella Protocol: Protocol Analysis and Research Proposals", unpublished report available at http://www9.limewire.com/download/ ivkovic_paper.pdf.

[KBB+04] T. Karagiannis, A. Broidi, N. Brownlee, K. Claffy, M. Faloutsos, "Is P2P Dying or just Hiding?", IEEE Globecom, November 2004.

[KLD+04] S. Kang, X. Liu, M. Dai, D. Loguinov," Packet-pair Bandwidth Estimation: Stochastic Analysis of a Single Congested Node", ICNP-04, 2004.

[LKR04] J. Liang, R. Kumar, K. Ross, "Understanding KaZaA", unpublished, available at http://cis.poly.edu/ ~ross/papers/.

[LRW03] N. Leibowitz, M. Ripeanu, A. Wierzbicki, "Deconstructing the KaZaA Network", Proceedings of the Third IEEE Workshop on Internet Applications (WIAPP'03), June 2003.

[Mark02] E. Markatos, "Tracing a large-scale Peer to Peer System: an hour in the life of Gnutella", CCGrid 2002.

[MNR03] J. Martin, A. Nilsson, I. Rhee, "Delay-based Congestion Avoidance for TCP", IEEE/ACM Transactions on Networking, 11(3), 356-369 (2003).

[MW05] J. Martin, J. Westall, "Validating an 'ns' Simulation Model of the DOCSIS Protocol", under review, available at: http://people.clemson.edu/~jmarty/papers/docsis-model.pdf.

[NCR+03] T. Ng, Y. Chu, S. Rao, K. Sripanidkulchai, H. Zhang, "Measurement-Based Optimization Techniques for Bandwidth-DemandingPeer-to-Peer Systems, Infocom, 2003.

[NS2] The Network Simulator. Available at : http://www-mash.cs.Berkeley.EDU/ns/.

[PACK] Packeteer bandwidth management devices, http://www.packeteer.com

[PGE+05] J. Pouwelse, P. Garbacki, D. Epema, H. Sips, "The BitTorrent P2P File-Sharing System: Measurements and Analysis", International Workshop on Peer-To-Peer Systems, Feb 2005.

[PIP05] A. Lenhart, M. Madden, "Teen Content Creators and Consumers", Pew Internet & American Life Project, November 2, 2005.

[QS04] D. Qiu, R. Srikant, "Modeling and Performance Analysis of BitTorrent-like Peer-to-Peer Networks", Proceedings of the

2004 conference on Applications, technologies, architectures, and protocols for computer communications, 2004.

[RB02] P. Rodriguez, E. Biersack, "Dynamic Parallel Access to Replicated Content in the Internet", IEEE/ACM Transactions on Networking, Vol 10, no 4, Aug 2002.

[RFH+01] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, "A Scalable Content Addressable Network", ACM SIGCOMM 2001

[RFI02] M. Ripeanu, I. Foster, A. Iamnitchi, "Mapping the Gnutella Network: Properties of Large-scale Peer-to-Peer Systems and Implications for System Design", IEEE Internet Computing Journal, 6(1), 2002.

[RIP01] M. Ripeanu, Peer-to-Peer Architecture Case Study: Gnutella Network, 2001.

[SGG02] S. Saroiu, P. Gummadi, S. Gibble, "A Measurement Study of Peer-to-Peer File Sharing Systems", Proceedings of Multimedia Computing and Networking 2002.

[SMK+01] I Stoica, R. Morris, D. Karger, M. Kaashoek, H. Balakrishman, "Chord: A Scalable Peer-to-Peer Lookup Prototocol for Internet Applications", ACM SIGCOMM 2001.

[SGP04] K. Skevik,V. Goebel, T. Plagemann, "Analysis of BitTorrent and its use for the Design of a P2P based Streaming Protocol for a Hybrid CDN", Technical report, Department of Informatics, University of OSLO, 2004.

[TCPD] tcpdump program available at http://www.tcpdump.org

[YV04] X. Yang, G. Veciana, "Service Capacity of Peer-to-Peer Networks", IEEE Infocom

[VY03] G. Veciana, X. Yang, "Fairness, Incentives and Performance in Peer-to-Peer Networks", Proceedings of the Forty-first Annual Allerton Conference on Communication, Control and Computing, Oct 2003.

[WSJ05] P. Grant, J. Drucker, "Phone, Cable Firms Rein in Consumers's Internet Use", The Wall Street Journal, October 21, 2005.

Set	Comment	US bandwidth	DS bandwidth	# Traces in Set
1	Campus, but rate limited	64 Kbps	64 Kbps	3
2	Campus, high speed	100 Mbps	100 Mbps	6
3	DSL home network	256 Kbps	1.5 Mbps	3

Table 1. Summary of the three data sets

Trace Number	US Bottleneck Link Speed Estimate	AvgCx BW (bps)	Avg TCP RTT	Avg Loss Rate (%)
Calibration US	214484	210161	1.164	0
Calibration DS	1221614	1123897	*	0

Table 2. Residential Network Calibration Results

Trace Number	US Bottle-neck Link Speed Estimate	AvgCx BW (bps)	Avg TCP RTT	Avg Loss Rate (%)	Avg aggregate US BW (bps)	Avg Aggregate DS BW (bps)	US/DS BW ratio	US > 100Kbps, 40Kbps, 10Kbps flows	DS > 100Kbps, 40Kbps, 10Kbps flows
1	54873	18655	.554	2.43	326079	129626	2.5	0, 2.8, 7.5	0, .2, 4.9
2	55523	36705	.407	.431	462271	312500	1.48	0, 6, 10.1	0, 2.6, 8.3
3 (seed)	63655	47509	.447	.96	1379270	30947	45	0, 22, 28.5	0, 0, 0

Table 3. Set1 BitTorrent Client Measurement Results

Trace Number	US Bottle-neck Link Speed Estimate	AvgCx BW (bps)	Avg TCP RTT	Avg Loss Rate (%)	Avg Aggregate US BW (bps)	Avg Aggregate DS BW (bps)	US/DS BW ratio	US > 100Kbps, 10Kbps flows	DS > 100Kbps, 10K bps flows
4	3948836	191714	.366	2.6	2180090	1292200	1.69	5.8,10.0	4.9,8.4
5	3809208	211886	.361	2.9	2525240	885066	2.85	5.8, 12.3	3,10.8
6 (seed)	2425737	203143	.802	1.1	2671200	111122	24	6.4, 16.1	0,3.7

Table 4. Set2 BitTorrent Client Measurement Results

Trace Number	US Bottle-neck Link Speed Estimate	AvgCx BW (bps)	Avg TCP RTT	Avg Loss Rate (%)	Avg Aggregate US BW (bps)	Avg Aggregate DS BW (bps)	US/DS BW ratio	US > 100Kbps, 40Kbps, 10Kbps flows	DS > 100Kbps, 40Kbps, 10Kbps flows
7	123797	23347	.989	4.35	189623	161176	1.18	.1, 1.2, 3.8	.1, 1.2, 3.8
8	111115	17840	1.11	4.8	201635	156835	1.28	0, 1.2,4.7	0, .8, 4.4
9 (seed)	140998	20869	3.93	.013	214107	5851	37	0, .4, 6.2	0, 0, 0

Table 5. Set3 BitTorrent Client Measurement Results

Set Identifier	**Avg TCP Cx BW**	**Avg TCP Cx RTT**	**Avg TCP Cx LR**	**Aggregate US BW**	**Aggregate DS BW**	**US/DS BW ratio**	**Number of concurrent US and DS flows**
Symmetric	60000	.05	4%	160 Kbps	1.4 Mbps	.11	1
Symmetric and parallel cxs	17500	.2	6.5%	185 Kbps	1.3 Mbps	.14	4

Table 6. Summary of the Symmetric Simulations

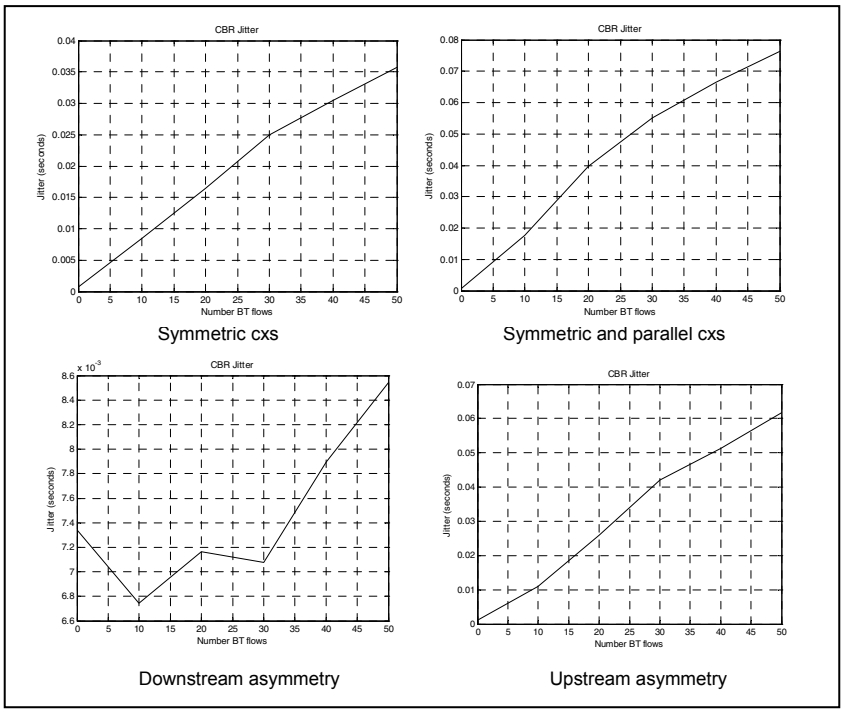

Figure 3. CBR Jitter as BitTorrent Load Increases

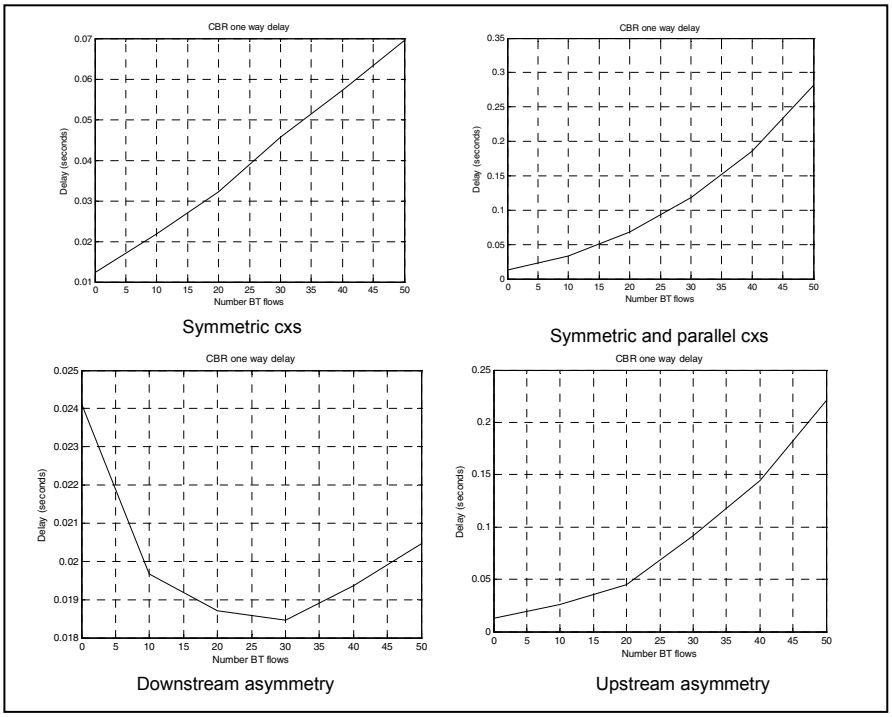

Figure 4. CBR One-Way Latency as BitTorrent Load Increases

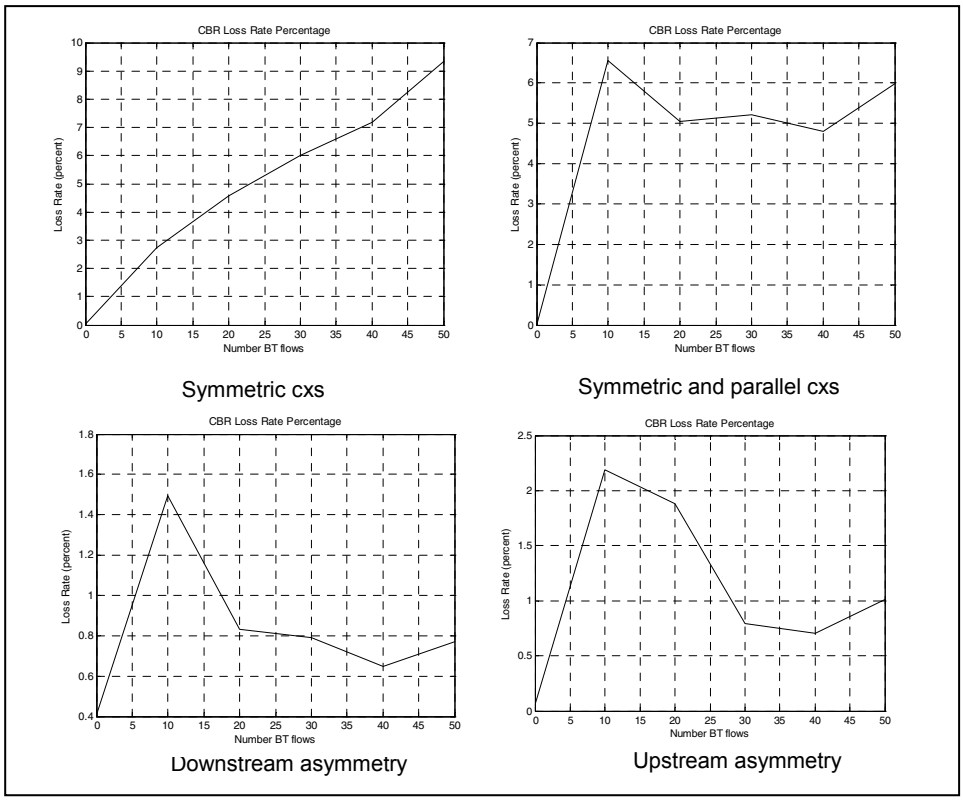

Figure 5. CBR Loss as BitTorrent Load Increases

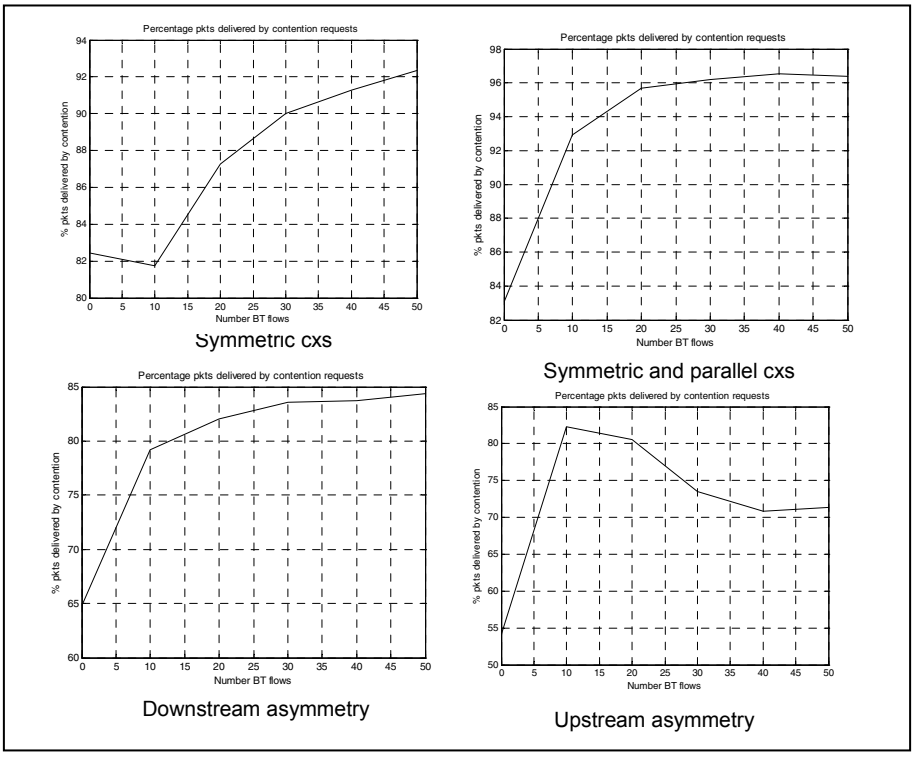

Figure 6. Percent Contention Requests as BitTorrent Load Increases

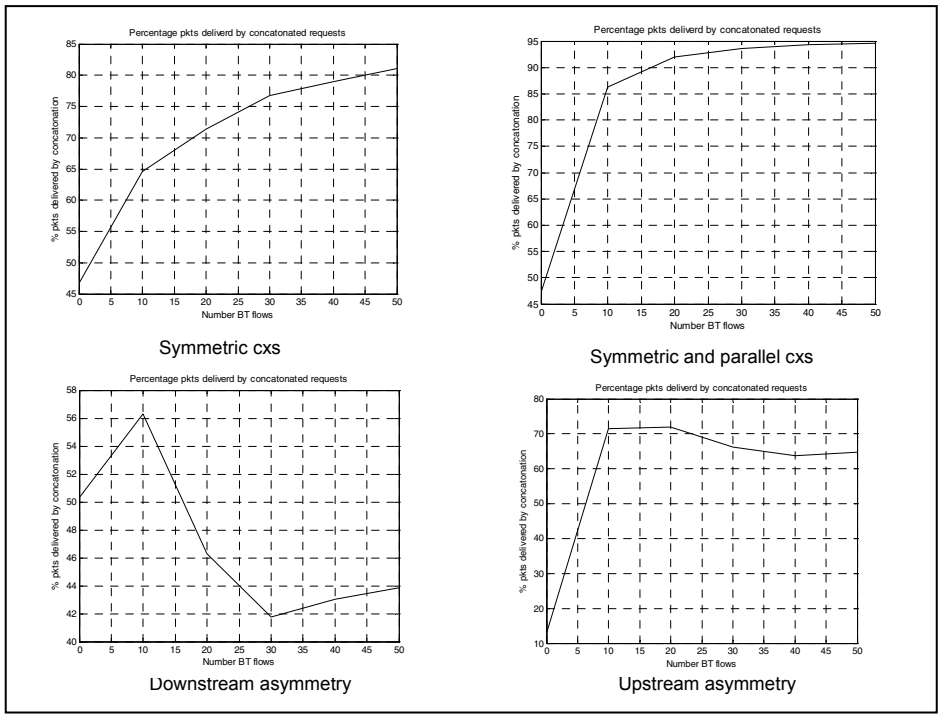

Figure 7. Percent Concatenation Requests as BT Load Increases

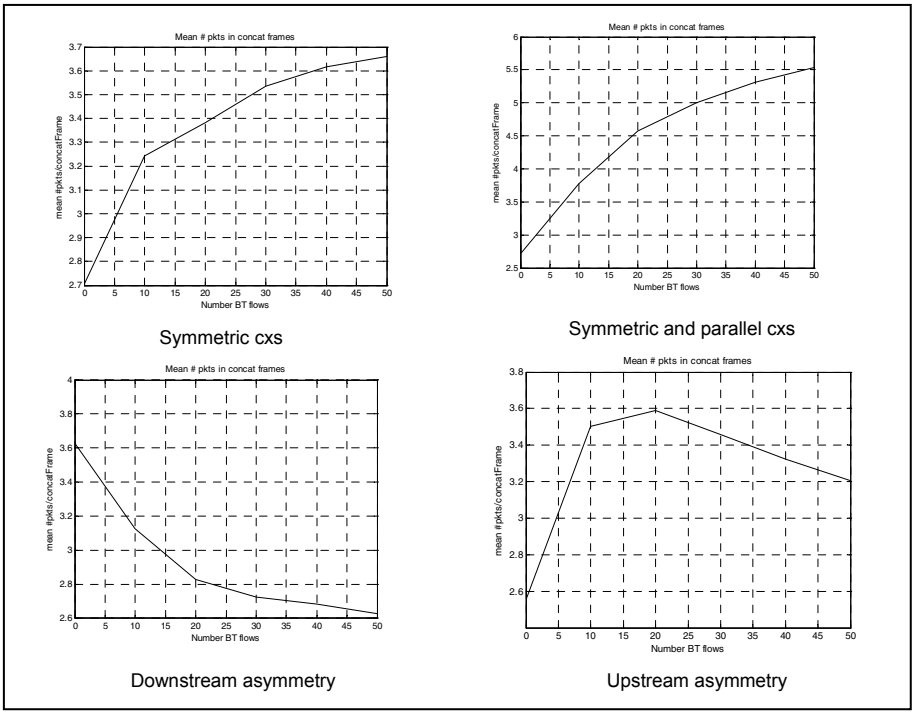

Figure 8. Mean Number of Pkts in Concatenated Frames as BitTorrent Load Increases

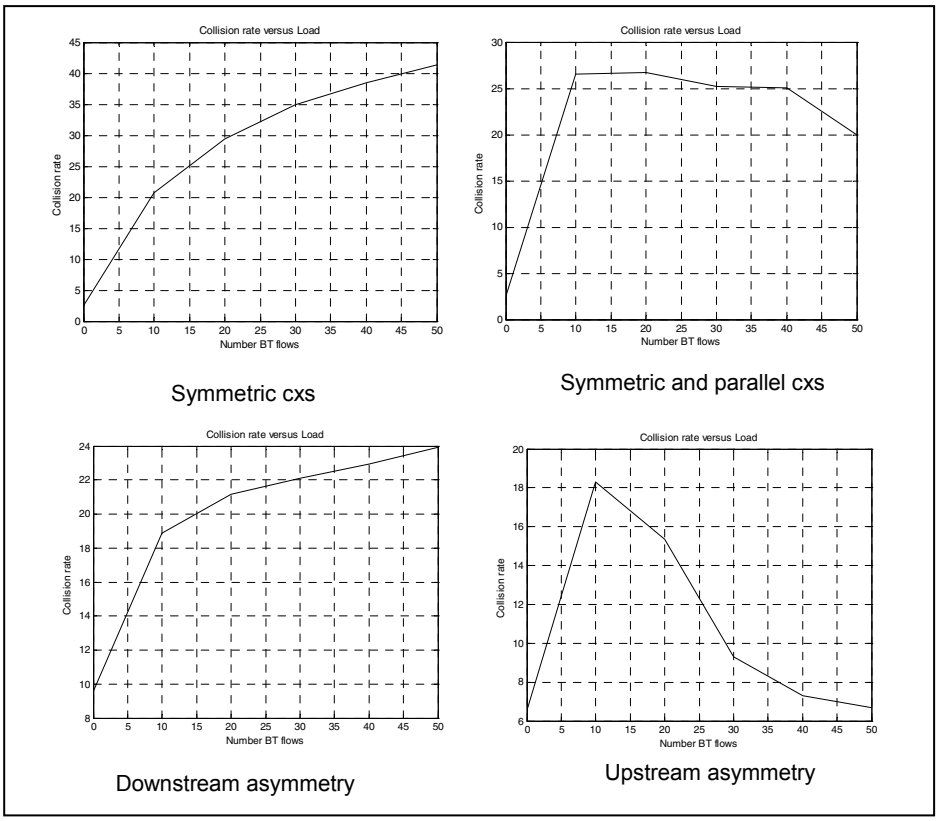

Figure 9. Collision rate as BitTorrent load increases