DOCSIS PERFORMANCE ISSUES

Jim Martin
Department of Computer Science, Clemson University

Abstract

We have developed a model of DOCSIS using the 'ns' simulation package. We identify a set of possible DOCSIS performance issues which includes complex interactions between downstream TCP connections and upstream MAC operation, vulnerabilities caused by MAC level denial-of-service attacks and fairness issues. We summarize our ideas involving bandwidth management to address the issues.

INTRODUCTION

The Data over Cable (DOCSIS) Service Interface Specification defines the Media Access Control (MAC) layer as well as the physical communications layer that is used in the majority of hybrid fiber coaxial cable networks that offer data services [1]. A Cable Modem Termination System (CMTS) interfaces with hundreds or possibly thousands of Cable Modem's (CMs). The original DOCSIS MAC interface (version 1.0) provides a best effort service with simple prioritization capabilities. DOCSIS 1.1, which is currently being deployed, adds a set of ATM-like services along with the necessary QoS mechanisms. The follow on standard, version 2.0, enhances the physical layer communication methods with higher upstream data rates and improved tolerance to bursts of noise.

The CMTS makes upstream CM bandwidth allocations based on CM requests and QoS policy requirements. The upstream channel is divided into '*minislots*' (referred to as *slots*) which, depending on system configuration, contain between 8 to 32 bytes of data. The CMTS periodically sends a '*MAP*' message to all CMs on a downstream channel that indicates upstream bandwidth allocation over the next '*MAP time*'. The MAP provides slot assignments for particular CMs in the form of data grants, provides opportunities for CMs to request upstream bandwidth using a contention-based request process and identifies which slots are to be used for system overhead.

A critical component of the DOCSIS MAC layer is the upstream bandwidth allocation algorithm. The DOCSIS specification purposely does not specify these algorithms so that vendors can develop their own solutions. However, all upstream bandwidth management algorithm will share a set of basic system parameters such as the amount of time in the future that the scheduler considers when making allocation decisions (we refer to this parameter as the *MAP_TIME*), the amount of upstream bandwidth allocated for contention-based bandwidth requests and the range of collision backoff times. These parameters are crucial for ensuring good performance at high load levels.

We have developed a model of the DOCSIS MAC and physical layer using the '*ns*' simulation package [2]. In previous work we reported on the impact of several DOCSIS operating parameters on TCP/IP performance [3]. In this paper we extend those results by looking in greater detail at the impact that the MAC layer has on TCP performance when using the DOCSIS best effort service. We show that the interaction between DOCSIS and TCP exposes a possible denial-of-service vulnerability. By exploiting the inefficient, contention-based bandwidth request mechanism, a hacker can severely impact network performance. We demonstrate fairness issues involving TCP and video streaming protocols that are 'TCP-friendly'. Most streaming video applications do not respond to network congestion. The Internet community has addressed this by developing the Datagram Congestion Control Protocol (DCCP) which provides an unreliable datagram transport service that includes TCP-compatible congestion control algorithm referred to as the TCP Friendly Rate Control (TFRC) protocol. While DOCSIS impacts downstream TCP performance, it does not impact the performance of TFRC (at least to the same degree). This causes TFRC flows to steal bandwidth from similarly configured TCP connections. We summarize our ideas on how bandwidth management can address these issues. We propose a bandwidth management algorithm that addresses fairness issues that include controlling TCP unfriendly flows and also subscribers that consume a disproportionate amount of bandwidth.

This paper is organized as follows. The next section presents the operation and features of our DOCSIS model. We present experimental results illustrating the performance issues. We then present our bandwidth management algorithm. We end the paper with a discussion of related work, present conclusions and identify future work.

SUMMARY OF THE MODEL

The model implements the DOCSIS architecture defined in [1]. Packets sent over the downstream channel are broken into 188 byte MPEG frames each with 4 bytes of header and trailer. The model accounts for physical layer overhead including framing bits and forward error correction data. The downstream channel supports an optional token bucket-based service rate. Each SID service queue is treated in a first come first serve manner. Depending on traffic dynamics, queueing can occur at either the SID queue or the downstream transmission queue. The maximum size of either queue is a simulation parameter.

All CMs receive periodic MAP messages from the CMTS that identify future upstream scheduling opportunities over the next MAP time. If provisioned with a periodic grant, a CM can send at its next data grant opportunity. For best effort traffic, a CM must request upstream bandwidth from the CMTS using a contention-based mechanism. To improve efficiency, a CM can request bandwidth to transport multiple IP packets in a single DOCSIS frame by issuing a concatenated request. Further, a CM can piggyback a request for bandwidth on an upstream data frame. If a CM receives a grant for a smaller number of minislots than were requested, the CM must fragment the data to fit into the assigned slots. Our model supports concatenation, piggybacking and fragmentation.

Figure 1 illustrates the MAP layout used in our model. The first slot at the left of the MAP represents time 0 in the MAP time. Data slots are placed at the beginning of the MAP and contention slots are placed at the end. Figure 2 illustrates the upstream transmission of a 1500 byte IP datagram from a TCP source directly connected to a CM to a sink connected to the CMTS. In Figure 2, time progresses in the downwards direction. We assume collisions do not occur. Assuming a MAP size of 80 slots, an upstream channel capacity is 5.12Mbps and there are 4 ticks per slot, 96 slots are required to transport the entire packet. The small dark square box positioned at the beginning each MAP time in the figure represents the transmission of the MAP message in the downstream direction. Our model sends the MAP at the beginning of each MAP time. Each MAP describes the slot assignments for the next MAP time. The IP packet arrives at the CM during the *j'th* MAP time at time T-0. The CM sends the bandwidth request message at time T-1 and receives the data grant at time T-2. The grant is allocated in the *j+2* MAP time. The CM sends the frame at Time T-3 and is received by the

CMTS at time T-4. The time between T-3 and T-0 is the access delay which represents the total time a packet is delayed over the DOCSIS network not including transmission or propagation time. The model can be configured to allocate a specific number of contention request slots each MAP. Or, in addition to a minimum number of contention request slots, all unused slots can be designated for contention requests.

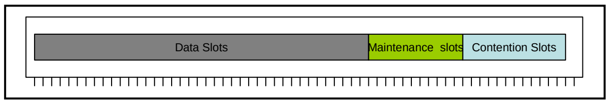

Figure 1. MAP layout

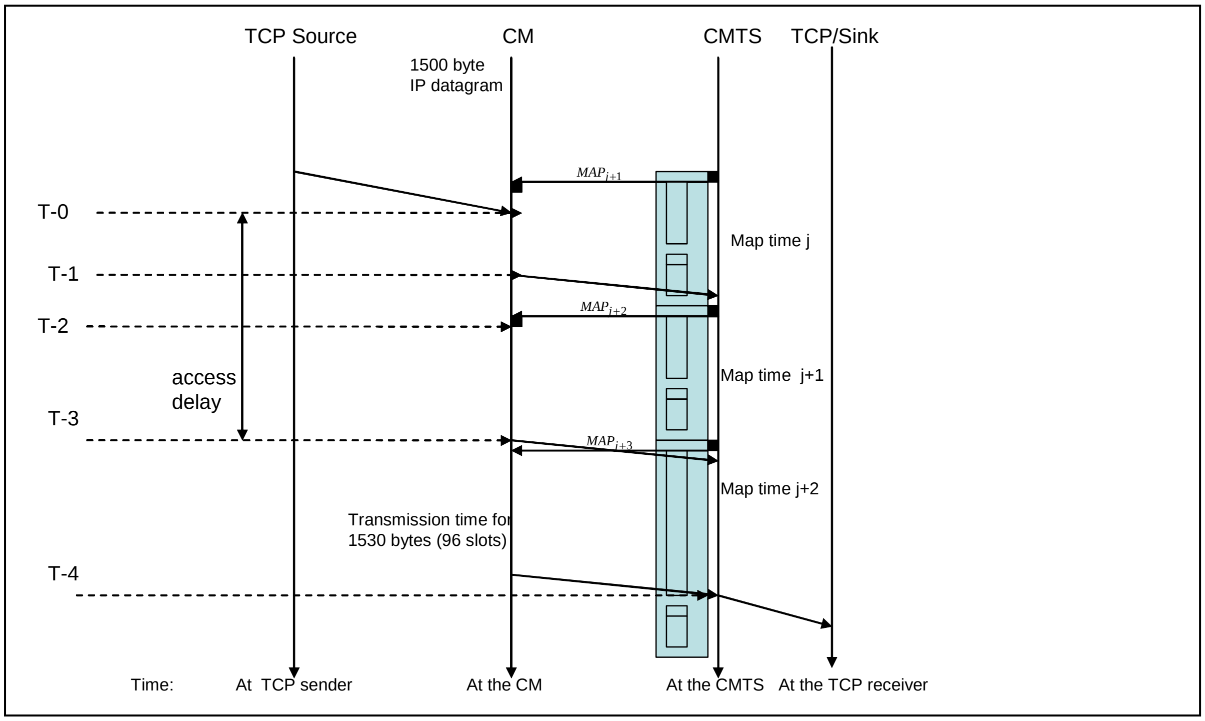

Figure 2. Upstream operation

IMPACT OF DOCSIS ON TCP

The results we report were based on simulation experiments using the network shown in Figure 3. The DOCSIS parameters were based on optimal configuration parameters that we found in a previous study [3]. A set of user nodes were attached to the CMs and a set of server nodes were located in the wired network. The traffic generators utilized realistic traffic models consisting of a combination of web, P2P and streaming traffic. The network and web traffic models were based on the "flexbell" model defined in [4]. In addition to downstream web traffic, we configure 5% of the CMs to generate downstream low speed UDP streaming traffic (i.e., a 56Kbps audio stream), 2% of the CMs to generate downstream high speed UDP streaming traffic (i.e., a 300Kbps video stream) and 5% of the CMs to generate downstream P2P traffic. The P2P model (based on [5]) incorporates an exponential on/off TCP traffic generator that periodically downloads on average 4Mbytes of data with an average idle time of 5 seconds between each download. The downstream transmission queue at the CMTS was configured to hold a maximum of 50 packets. We limited the number of packets that can be concatenated in a single frame to two. The DOCSIS and Web traffic simulation parameters are shown in Figure 4.

We varied two parameters in the experiments, the MAP_TIME and the number of CMs. For a given MAP_TIME setting, we varied the number of CMs from 100 to 500. We do this for six MAP_TIME settings ranging from .001 to .01 seconds.

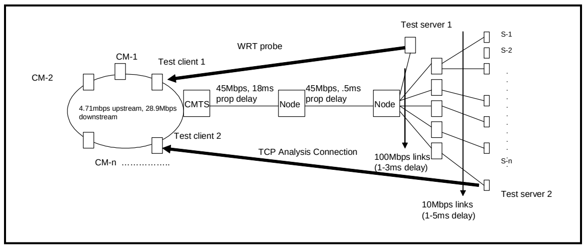

Figure 3. Simulated network

We obtained the following statistics for each run:

Collision rate: Each time a CM detects a collision it increments a counter. The collision rate is the ratio of the number of collisions to the total number of upstream packets transmissions attempted.

Downstream and upstream channel utilization: At the end of a run, the CMTS computes the ratio of the total bandwidth consumed to the configured raw channel bandwidth. The utilization value reflects the MAC and physical layer overhead including FEC bits.

Average upstream access delay: All CMs keep track of the delay from when an IP packet arrives at the CM in the upstream direction until when it actually gets transmitted. This statistic is the mean of all of the samples.

Web response time: a simple TCP client server application runs between test client 1 and the test server 1. Test server 1 periodically sends 20Kbytes of data to test client 1. With each iteration, the client obtains a response time sample. The iteration delay is set at 2 seconds. At the end of the test, the mean of the response times is computed. The mean web response time (WRT) can be correlated to end user perceived quality by using a very coarse rule of thumb that says end users are bothered by lengthy download times when the mean WRT metric value exceeds 1 second. We do not claim this to be an accurate measure of end user quality of experience. Instead, it is a convenient, reproducible performance reference.

Model Parameters
Upstream bandwidth 5.12Mbps
Preamble 80 bits
Downstream bandwidth 30.34Mbps
4 ticks per minislot
Default map time: 2 milliseconds (80 minislots per map)
Fragmentation Off, MAP_LOOKAHEAD = 255 slots
Concatonation ON
Backoff Start: 8 slots, Backoff stop: 128 slots
12 contention slots, 3 management slots
Simulation time: 1000 seconds

Web Traffic Model Parameters
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

Figure 4. Simulation parameters

Web Congestion Experiment Results

Figures 5a and 5b plot the channel utilization as the load increases. The downstream utilization reaches a maximum of about 64% with a MAP_TIME setting of .001 second. In this case, 12 contention slots per MAP is sufficient. For larger MAP_TIME values, the downstream utilization ramps up to its maximum value and then decreases at varying rates as the load increases. As the collision rate grows, downstream TCP connection throughput decreases. Limiting each MAP to 12 contention slots results in fewer total contention request opportunities as the MAP_TIME grows. This explains the high collision rates and reduced downstream utilization for the runs with large MAP_TIME settings.

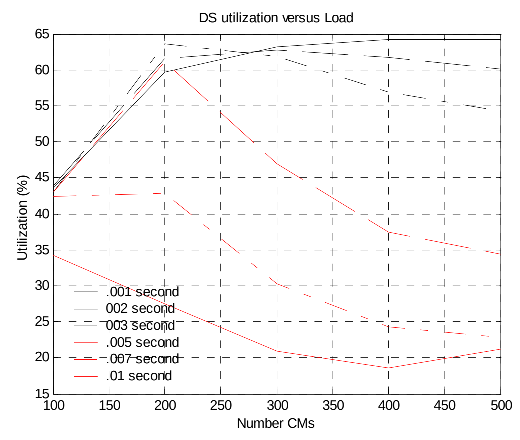

Figure 5a. Downstream channel utilizations

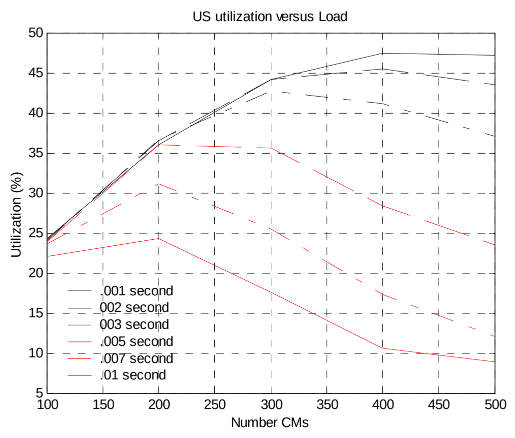

Figure 5b. Upstream channel utilizations

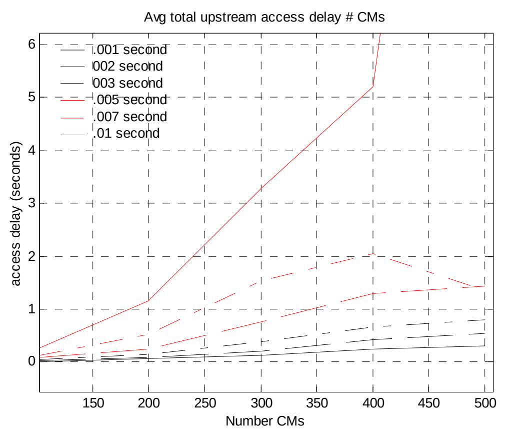

Figure 6a. Upstream access delay

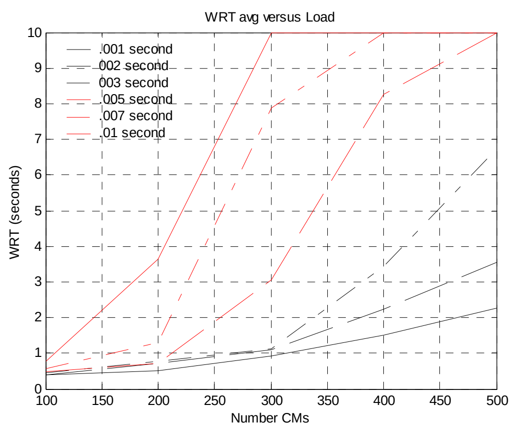

Figure 6b. Web response time metric results

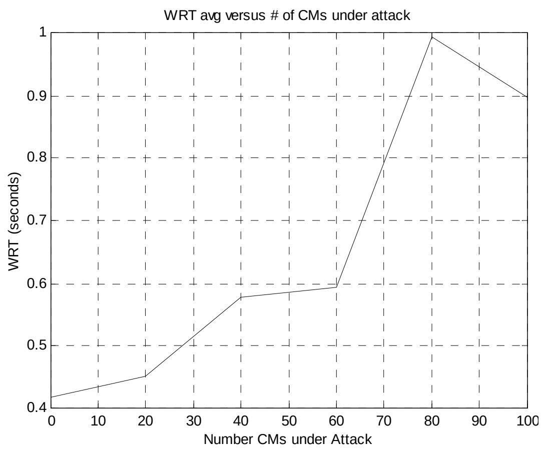

Figure 7a. WRT results without rate control

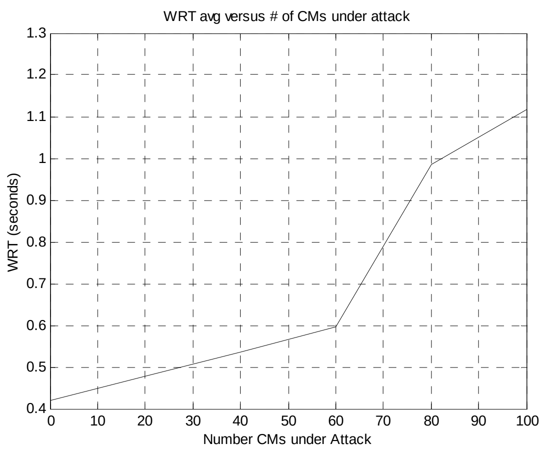

Figure 7b. WRT with 2Mbps DS rate control

Figure 6a shows that the average upstream access delay becomes very large at high loads when configured with large MAP_TIME settings. Even for lower MAP_TIME values, the access delay was significant. For a MAP_TIME of .002 seconds, the access delay exceeded .5 seconds at the highest load level. To assess the impact of the cable network on end-to-end performance we monitored web response times. Using the rule of thumb described earlier, Figure 6b suggests that for MAP_TIME settings less than .005, up to 300 users can be active before performance becomes bothersome to end users.

When 100 users are active, the collision rate is about 50%. What makes this result alarming is that the web traffic model accounts for the heavy tailed distribution associated with web user idle times. Consequently, the number of users actually competing for bandwidth at any given time is much less than 100. As the load increased, the collision rate approached 90% depending on the MAP_TIME setting.

When the dominant application is web browsing, the majority of data travels in the downstream direction. At high loads, the network can become packet rate bound causing ACK packets accumulate in the CM upstream queues waiting for transmission opportunities. Piggybacking is of limited benefit since ACKs that arrive back-to-back are sent in a concatenated frame. Concatenation can be helpful although it drastically increases the level of '*ACK compression*' experienced by downstream TCP data transfers [3]. ACK compression occurs when a network causes TCP acknowledgement packets to 'bunch' at some point leading to bursty TCP send behavior which in turn contributes to higher loss rates and poor network utilization.

We repeated the study using different parameters and features of the model. The results are virtually identical if we turn on a downstream service rate of 2Mbps, if we turn on ACK filtering or if we allocate all unused slots for contention requests. However, if we increase the downstream transmission queue size at the CMTS (the point where loss occurs) from 50 to 300 packets, loss no longer occurs and downstream utilization approaches 75%. While a larger buffer improves performance, the important result is that the downstream TCP traffic is subject to extreme levels of ACK compression caused by DOCSIS.

DoS VULNERABILTIES

In this section we show that it is possible for a hacker to take advantage of the inefficient contention-based upstream bandwidth allocation process by initiating a denial-of-service (DoS) attack. To accomplish the DoS attack, a host located outside the DOCSIS network must learn the IP address of a number of CMs that share the same upstream channel. The attacker transmits either a ping or a TCP SYN packet to the targeted CMs at a frequency that depends on how many CMs are under attack. The objective of the attack is to cause a large number of contention-based requests resulting in high collision rates and subsequently poor network performance. This type of attack has been identified in 802.11 networks where an attacker stimulates stations to initiate RTS/CTS exchanges leading to dramatically reduce network efficiency [15].

We simulate an attack using the network model illustrated in Figure 3. The configuration was identical to that described in Figure 4. We set the MAP_TIME to .002 second. There were 100 CMs but the number of CMs under attack was varied. The collision rate increased from 48% to 68% as the number of CMs under attack increased from 0 to 100. The downstream utilization dropped from 45% to 10%. Figure 7a shows that the web response times increased by a factor of 3. The web response time monitor was located at a CM that was not under attack (i.e., test client

1 in Figure 3). In a separate experiment, we included the test client 1 CM in the attack and found that the CM was not able to complete a single web response time sample. Not surprisingly, a CM subject to a flooding attack effecttively makes the access network unavailable to the subscriber.

We ran the denial-of-service experiment a second time with downstream service rates set to 2Mbps. The results were virtually identical to the previous results. Figure 7b shows the average web response times from test client 1 when this node was not under attack also increased by almost a factor of 3. The result suggests that a 2Mbps downstream service rate will not protect the network from the attack.

FAIRNESS ISSUES

There are several fairness issues that can arise in a DOCSIS network primarily caused by upstream packet rate limitations. The first issue is that DOCSIS exhibits bias against TCP connections running over paths with small MTU sizes. If two CMs are each transporting data from separate but identically configured TCP connections with the exception that one connection has a negotiated MSS of 512 bytes and the other connection uses an MSS of 1492 bytes, the connection that generates the larger packets will consume more bandwidth than the other connection.

A second issue, applicable to the downstream direction, involves streaming video protocols, such as TFRC, that claim to be TCP-friendly. Because DOCSIS systems can be packet rate bound, rate-based protocols such as TFRC that do not require an ACK stream to clock new data can consume larger amounts of bandwidth than comparable TCP connections.

To demonstrate this second issue, we modified the previous web scenario experiment by adding an FTP-like TCP flow between one of the CMs and a server and a similar FTP-like TFRC flow between another CM and server pair. The MAP_TIME was .002 seconds and the number of contention request slots per map was set to 12. We performed six runs, increasing the number of CMs from 0 to 500. Figure 8 plots the TCP and TFRC connection throughputs for each run. The TFRC flow obtains roughly 3-7 times the bandwidth of the TCP flow depending on the number of CMs. When there are just the two CMs competing (this is the 0 point on the x-axis of Figure 8), the TFRC and TCP flows achieve a throughput of 18Mbps and 6 Mbps respectively. The TFRC flow by itself (i.e., if we do not run the competing TCP connection) obtains about 22Mbps while the TCP flow by itself obtains about 12.5 Mbps. If the channel bandwidths increase, the maximum TCP throughput does not change (because TCP throughput is packet rate limited in the upstream direction rather than limited by downstream bandwidth). The TFRC flow does not have this limitation and can consume higher downstream bandwidths.

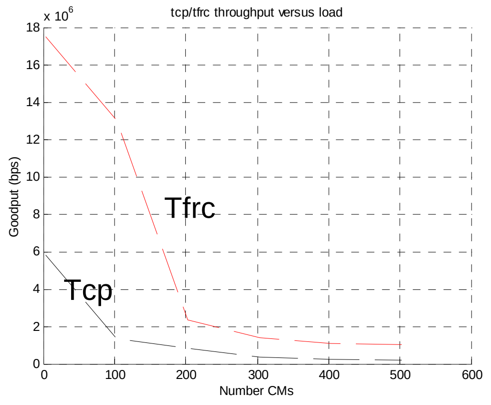

Figure 8. TCP and TFRC throughput

BANDWIDTH MANAGEMENT

In our current research, we are exploring bandwidth management to address these issues. One component of our work is to develop a protocol aware scheduling algorithm that predicts future CM upstream bandwidth needs and provides unsolicited grants. A second component is to develop bandwidth management algorithms that manage bandwidth based on a particular policy or service. For example, as an alternative to a pay-per-use policy, a provider might desire a policy where subscribers that consume large amounts of bandwidth in either the upstream or downstream directions are ‘punished’ by being placed in a state of reduced service rates for a given time period.

We have prototyped such an algorithm in our simulation model. The motivation for the algorithm is that future cable services will offer much higher service rates, possibly on the order of tens of megabits per second. To manage fairness issues or to facilitate new service options, dynamic bandwidth management is required. The objective of our algorithm is to prevent the large number of well behaved subscribers from adverse affects caused by a few high bandwidth users (referred to as ‘heavy-hitters’). The algorithm has three components: detecting poor quality of service observed by normal user, identifying the heavy-hitters and regulating the heavy users to solve the problem.

The algorithm runs at the CMTS and does not require any changes at the CM nodes. The algorithm can be used on the downstream channel or the upstream channel (or both). We have applied the algorithm to manage downstream bandwidth. The majority of subscribers are well behaved in the sense that they consume a reasonable amount of bandwidth over large time periods. A few subscribers are not well behaved (i.e., the heavy-hitters) and consume a disproportionate amount of bandwidth over large time scales. To simplify the discussion, we assume that there is one user per CM.

Detecting poor quality of service observed by normal user

We use the WRT metric described earlier to characterize the quality of service perceived by a subscriber. A node attached to a CM node periodically sends a request to an HTTP server for a 20Kbyte object. The time taken for this download is monitored periodically and is averaged over a time-scale defined by the parameter *WRTM* (WRT monitor interval). We assume that when the average of the WRT samples over a *WRTM* time period approaches 1 second users will perceive poor quality. Once this situation is detected, the algorithm identifies the heavy-hitters that are contributing to congestion in the network.

Identifying heavy-hitters in the cable network

The algorithm maintains the average bandwidth rate (*ABR*) of all active users. The time interval over which the rate is averaged is defined by the parameter *TAVG*. An active user is a user whose *ABR* is not zero over a *TAVG* amount of time. Based on the maximum channel capacity and the number of active users present, a fair bandwidth rate (*FBR*) of each user is calculated using the following equation:

FBR = (Maximum channel capacity)/(Number of Active users).

The number of active users present in the network is based on samples averaged over a period of time defined by the parameter *TNUS* (time for number users sample). Any user whose *ABR* is above a threshold based on the *FBR* is considered to be a heavy-hitter. The threshold value is represented by the parameter *THUSR* .

The timescale parameters, *TAVG* and *TNUS*, allows a cable service provider to implement different policies. For example, a small *TAVG* on the order of minutes, can be used to ensure that TFRC flows consume a fair share of bandwidth. A cable service provider might want to detect users who operate servers (e.g., peer-to-peer or web servers). This can be handled by setting the *TAVG* to days. What makes the algorithm unique is the fact that a heavy-hitter is not punished unless it is impacting other users. The extent of the punishment is determined by algorithm parameters.

Regulating the heavy users

Once a heavy-hitter has been identified, the next step is to regulate it to improve network performance. We implemented the policy that heavy-hitters never get more than the fair share of the bandwidth. The rate regulation continues for a configurable period of time (*TREG*). The rationale for 'punishing' the heavy-hitters by limiting their bandwidth to the fair share is to ensure that they no longer impact well-behaved users. Since we only consider the number of active users in calculating the fair share, it is possible that the channel might be under-utilized. For instance, assume there are 100 users using 30Kbps bandwidth. The fair share will be 300Kbps for a 30 Mbps channel. While being punished, a heavy-hitter can consume a maximum of 300Kbps even though additional bandwidth might be available.

Simulation verification

We demonstrate the algorithm using the simulation network in Figure 3. We configured 154 CMs to generate the traffic mix described in Figure 4. We configured 6 additional CMs to maliciously consume large amounts of downstream bandwidth using UDP traffic sources. All the heavy-hitters were started and stopped at the same time. When the simulation starts, the 100 web users are started. The heavy-hitters start at around 3000 seconds collectively generating around 35Mbps of traffic. Figure 9 plots the aggregate downstream bandwidth over the experiment. At time 3000 seconds we see the aggregate bandwidth increase as the heavy-hitters start. The algorithm smoothly adapts subscriber rates to the penalized value. The subscriber will be in the penalty state for about 4 hours. All web traffic stops time 13500 except for traffic generated by the 6 heavy-hitters. As the algorithm detects available bandwidth, it allocates more bandwidth to the heavy-hitters. If there are no other users, the fair share allocated to the heavy-hitters will consume all available bandwidth. More likely there will be other users in which case the *FBR* will limit the heavy-hitters but not affect well behaved users.

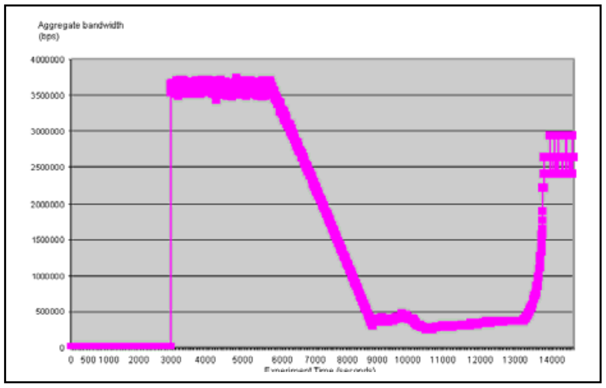

Figure 9. Bandwidth management algorithm

RELATED WORK

While the intent of the IEEE's 802.14 effort was to provide ATM services over a hybrid fiber coaxial (HFC) medium, the operation of the MAC layer is similar to that supported by DOCSIS. Therefore, prior 802.14 research is relevant. The work in [6] found that TCP throughput over an 802.14 network is low primarily due to ACK compression. The authors propose two solutions: one involving piggybacking and a second involving TCP rate smoothing by controlling the ACK spacing. The authors found that piggybacking can help reduce the burstiness associated with the ACK stream in certain situations. However it is limited in its abilities to effectively match offered load over a range of operating conditions. The author's second solution is to control the TCP sending rate by measuring the available bandwidth and calculating an appropriate ACK rate and allowing the CM to request a periodic grant that provides sufficient upstream bandwidth to meet the required ACK rate. We distinguish our work by focusing on the latest DOCSIS standards (1.1 and 2.0) and using more realistic traffic loads.

The observation in [7] is that an HFC network presents difficulties for TCP due to the asymmetry and due to high loss rates (possibly as high as 10-50%). Due to the problems of TCP/Reno in these environments[8,9,10], the authors propose a faster than fast retransmit operation where a TCP sender assumes that a packet is dropped when the first duplicate ACK is received (rather than the usual triple duplicate ACK indication). The motivations behind [7] are not relevant with the latest DOCSIS standards as DOCSIS 2.0 provides nearly symmetric access links with low packet loss rates as long as the plant is well engineered.

The performance of TCP over asymmetric paths has been thoroughly studied [11,12,13]. A network exhibits asymmetry with respect to TCP performance if achieved throughput is not solely a function of the link and traffic characteristics of the forward direction but in fact depends on the impact of the reverse direction. Most of the prior work was focused on highly asymmetric paths with respect to bandwidth where the normalized asymmetry level (i.e., the ratio of raw bandwidths to the ratio of packet sizes in both directions) typically would be on the order of 2-4 [11]. In DOCSIS, depending on the service rate configuration, the level of bandwidth asymmetry is small (or nonexistent). Instead, DOCSIS exhibits packet rate asymmetry due to low upstream packet rates with respect to downstream capacity. However the problem symptoms are similar. Various methods have been proposed to alleviate the TCP over asymmetric path problems including header compression and modified upstream queue policies(drop-from-front, ACK prioritization, ACK filtering) [11,12,13,14]. Some of these ideas can be applied to DOCSIS. For example, a CM that supports ACK filtering could drop 'redundant' ACKs that are queued. We have implemented this and found that while it does increase the acknowledgement rate, it also increases the level of ACK compression. ACK reconstruction could be implemented in the CMTS to prevent the increased level of ACK compression from affecting performance. We plan on addressing this in the future.

CONCLUSIONS

Using simulation we have identified several issues. First we saw that DOCSIS can affect the ACK stream in the upstream direction resulting in bursty downstream dynamics. Second, we have identified a possible DoS vulnerability in DOCSIS. Taking advantage of the inefficiency associated with upstream packet transmissions, a hacker can negatively impact network performance by periodically stimulating (e.g., by ping or TCP SYN packets) a number of CMs at a frequency that depends on the number of CMs under attack. The signature for this attack would be different than that of traditional flooding attacks as the amount of bandwidth consumed in the downstream direction is low. Finally, we illustrated that a *TCP-friendly* protocol turns out to be *TCP-unfriendly* in a DOCSIS environment because the model of TCP behavior incorporated by TFRC fails to accurately capture how TCP performs in a DOCSIS environment.

We presented an algorithm that is designed to help manage user traffic in DOCSIS networks. While the algorithm might not be appropriate for todays networks that rely on low service rates or that involves penalties for bandwidth misuse, our work is intended for future higher speed cable access networks that are likely to offer service rates on the order of tens of Mbps. In these environments intelligent bandwidth management will be required.

REFERENCES

1. Cable Television Labs Inc. , CableLabs, "Data-Over Cable Service Interface Specifications- Radio

Frequency Interface Specification", SP-RFIv2.0, available at http://www.cablemodem.comspecificions/specifications20.html.

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

3. J. Martin, N. Shrivastav, "Modeling the DOCSIS 1.1/2.0 MAC Protocol", Proceedings of the 2003 International Conference on Computer Communications and Networks", Dallas TX, October 2003.

4. A. Feldmann, et. Al., "Dynamics of IP Traffic: A study of the role of variability and the impact of control", SIGCOM99.

5. S. Saroiu, P. Gummadi, S. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems", Multimedia Computing and Networking (MMCN), Jan 2002.

6. R. Cohen, S. Ramanathan, "TCP for High Performance in Hybrid Fiber Coaxial Broad-band Access Networks", IEEE/ACM Transactions on Networking, Vol. 6, No. 1, February 1998.

7. O. Elloumi, et. Al., "A Simulation-based Study of TCP Dynamics over HFC Networks", Computer Networks, Vol. 32, No. 3, pp 301-317, 2000.

8. O. Elloumi, et. Al., "Improving Congestion Avoidance Algorithms in Asymmetric Networks", IEEE ICC 97, June 1997.

9. K. Fall, S. Floyd, "Simulation-based Comparisons of Tahoe, Reno and SACK TCP", CCR, Vol 26, No. 3, July 1996.

10. J. Hoe, "Improving the Startup Behavior of a Congestion Control Scheme for TCP", SIGCOMM 96, August 1996.

11. H. Balakrishnan, et. Al., "The Effects of Asymmetry on TCP Performance", ACM/IEEE International Conference on Mobile Computing and Networking, Sept. 1997.

12. T. Lakshman, U. Madhow, B. Suter, "Window-based error recovery and flow control with a slow acknowledgement channel: a study of TCP/IP performance", INFOCOM97, April 1997.

13. V Jacobson, "Compressing TCP/IP Headers for Low-Speed Serial Links", Feb 1990, RFC 1144.

14. L. kalampoukas, A Varma, K. Ramakrishnan, "Improving TCP Throughput over Two-Way Asymmetric Links: Analysis and Solutions", SIGMETRICS 98, June 1998.

15. Saikat Ray, Jeffrey B. Carruthers, and David Starobinski, "RTS/CTS-induced congestion in ad-hoc wireless LANs," in IEEE Wireless Communication and Networking Conference (WCNC), March 2003.