Validating an ‘ns’ Simulation Model of the DOCSIS Protocol

Jim Martin
Department of Computer Science
Clemson University
Clemson, SC 29634-0974
jim.martin@cs.clemson.edu

Abstract-- **The number of households and businesses using HFC cable networks for Internet access is rapidly approaching 20million in the United States. The cable industry has standardized on a single MAC and physical layer standard, the Data Over Cable System Interface Specification (DOCSIS). We have implemented a simulation model of the DOCSIS 1.1/2.0 MAC and physical layer using the ‘*ns*’ simulation package. In this paper we provide analytic and live network evidence that the simulation model is correct. To demonstrate the model, we provide the results of a brief simulation-based performance evaluation designed to provide insight as to how a best effort VoIP service (e.g., Vonage) performs under varying traffic loads compared to a VoIP service that utilizes DOCSIS QoS mechanisms.**

***Keywords*— Simulation, Broadband Access, HFC Cable Networks, TCP Performance**

A. Introduction

The cable industry has converged to the Data Over Cable System Interface Specification (DOCSIS) as the standard MAC protocol for data over HFC cable networks [1]. The DOCSIS standard was developed by the cable industry’s research consortium, CableLabs [2]. The standard defines the MAC and physical layers for sending data over hybrid fiber coaxial (HFC) cable networks. A multi-system operator (i.e., a cable operator otherwise known as an MSO) operates the cable modem termination system (CMTS) units that interact with cable modems (CMs) deployed at subscriber’s locations. A modern CMTS houses multiple ‘blades’ with each blade supporting one or more HFC domains (1 downstream channel with one or more upstream channels). 6Mhz (or larger) of bandwidth is allocated from the 88-860Mhz spectrum for each downstream channel and upstream channels are allocated from the 5 – 52Mhz frequency range.

In the downstream direction, a single sender (the CMTS) transmits to a set of CMs using a data rate ranging from 10Mbps to 50Mbps. IP packets sent downstream are divided into 180 byte MPEG frames. As in a LAN, each CM has a unique MAC address and will receive only frames that are addressed to its MAC address or to the broadcast address. In the upstream direction, multiple senders (CMs) share a channel offering data rates in the range of 5Mbps to 10Mbps. The upstream transmission model is a combination of contentionless shared access using time division multiple access (TDMA) and contention-free access using a reservation mechanism. IP packets that are sent upstream are encapsulated in a DOCSIS frame and transmitted during assigned slots. If a packet does not fit in the number of contiguous slots that were allocated it is fragmented into multiple frames. Traffic from CMs is classified in terms of service flows. As an example, a configuration that supports telephony would have 4 service flows: two for the upstream and downstream VoIP traffic and two for the upstream and downstream best effort traffic. DOCSIS maps service flows to one of several ATM-like services including best effort, unsolicited grant service (UGS, which is equivalent to ATM’s CBR service), and non-realtime polling (nrtPS, which is equivalent to ATM’s nrtVBR service). As in ATM, different performance guarantees are available for each service. The particular type of service determines how upstream bandwidth is allocated to CMs. UGS periodically provides grants to the CM, nrtPS periodically asks the CM if it needs bandwidth, and best effort allocates bandwidth on-demand using a contention-based request mechanism.

Prior analysis of the IEEE 802.14 standard, the predecessor to DOCSIS, found that TCP throughput over HFC networks is low primarily due to ACK compression [3]. While assumptions made by the authors (such as high loss rates in the upstream path) are no longer true, our recent results do confirm that DOCSIS induces ACK compression. More recent analysis has been performed using an Opnet model[4]. However most of these studies were limited in scope. Further, since source code for the model is not readily available, the model is of limited use to the research community. In order for the research community to participate in the advancement of this important broadband access technology, a fully functional, validated and documented simulation model is required.

In previous work, we presented a preliminary *'ns'* DOCSIS simulation model that we developed and showed that certain system parameters can significantly impact performance and dynamics of a DOCSIS system [5,6,7]. In this paper we extend our past work by providing a robust validation of the model[1]. We validate the model by comparing simulation results with live network measurements and with analytic results. To demonstrate the model, we explore the use of best effort VoIP services over DOCSIS access networks. Services such as Vonage and AT&T offer VoIP without engaging DOCSIS QoS mechanisms [8,9]. While this conflicts with the cable industry's direction for telephony, these best effort services offer an inexpensive and popular alternative.

The rest of this paper is organized as follows. The next section presents the operation and features of our DOCSIS model. We present an analytic model that captures the upstream behavior of DOCSIS. Then we present the results of a simulation analysis using our *'ns'* DOCSIS model involving hundreds of active CMs generating a mix of web, streaming, P2P and VoIP traffic. Finally, we provide conclusions and identify future work.

---

[1] A long version of this paper as well as the *'ns'* simulation code and documentation is available at http://www.cs.clemson.edu/~jmarty/docsis.html

B. Summary of the Simulation Model

The simulation model implements the DOCSIS architecture defined in [1] with the following restrictions: 1)CMs are limited to a single default best effort service flow and a single UGS or nrtPS flow; 2)the model is limited to one upstream channel for each downstream channel; 3)the model does not support dynamic service provisioning; 4)physical layer impairments are not modeled; 5)the model assumes that the CMTS and the CM clocks are synchronized.

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 bandwidth from the CMTS using a contention-based mechansim. When a CM is ready to transmit a request frame, it selects a random number within its backoff window which is determined by the backoff range value. While this value is sent to the CM in each MAP, our implementation never alters the statically configured value. After a CM transmits the request, if the next MAP does not contain either a grant or a grant pending indication from the CMTS, the CM assumes a collision occurs and increases the backoff window by a factor of two. The contention request cycle continues until it succeeds or it has tried a total of 16 times in which case the packet is dropped. A CM can request bandwidth sufficient to transport multiple IP packets in a single DOCSIS frame by issuing a concatenated bandwidth request. A further performance improvement, one which minimizes the frequency of contention-based bandwidth requests, is for a CM to piggyback a request for bandwidth on an upstream data frame. If a CM receives a grant for a smaller number of slots than were requested, the CM must fragment the data to fit into the assigned slots.

Figure 1 illustrates the upstream transmission of a 1500 byte IP datagram from a TCP source to a sink located outside the HFC network. Time progresses in the downwards direction. We assume collisions do not occur. The figure depicts an example configuration with a MAP size of 64 slots, an upstream channel capacity of 5.12Mbps and 5 ticks per slot. 85 slots are required to transport a 1500 byte IP packet. The small dark square box positioned at the beginning each MAP time represents the transmission of the MAP message in the downstream direction. Our model sends a MAP message 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 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 located in the third 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 and propagation time experienced by the data packet (we also refer to this delay as the $t_{request}$). Packets that arrive at a CM that is waiting for bandwidth to send packets that have already arrived will be queued. The size of the upstream CM queue is a configuration parameter.

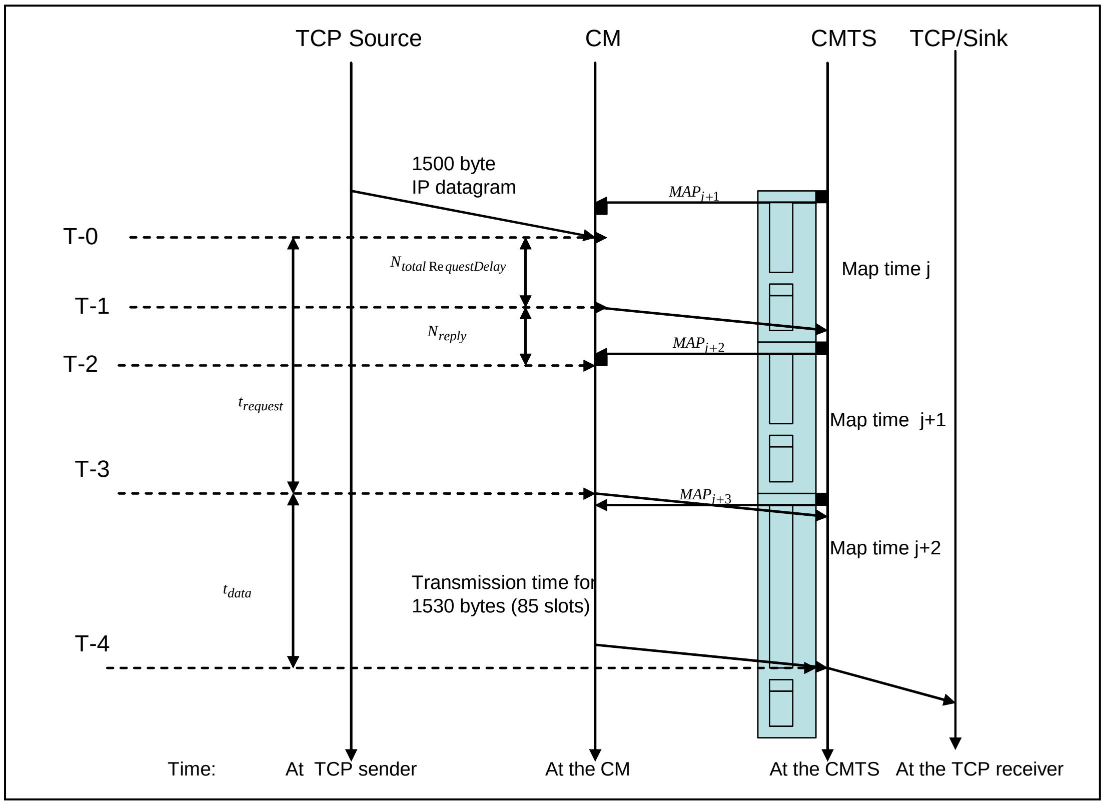


Figure 1. Upstream transmission scenario

The bandwidth scheduler runs at the CMTS node. It executes on every tick of a timer that is set to the MAP time frequency. The algorithm examines all existing requests for bandwidth and implements an earliest deadline first scheduling policy. All UGS service requests are scheduled and whatever bandwidth is left over is shared by best effort requests on a first-come-first-served basis. The scheduler supports dynamic MAP times by allowing a MAP to specify grants up to a configured maximum (known as the MAP lookahead). The scheduler will only do this if it can meet all QoS requirements. For example, a system with a configured MAP time of 64 slots will generate a MAP containing 100 slots when a 1500 byte IP packet is transmitted (85 slots for the IP packet, 3 slots for management frames and 12 slots for contention requests). When a MAP has been stretched to fit a bandwidth request, the scheduler always adds the configured number of management and contention slots to the MAP. The scheduler by default will assign all unused slots to be available for contention requests (this behavior can be changed).

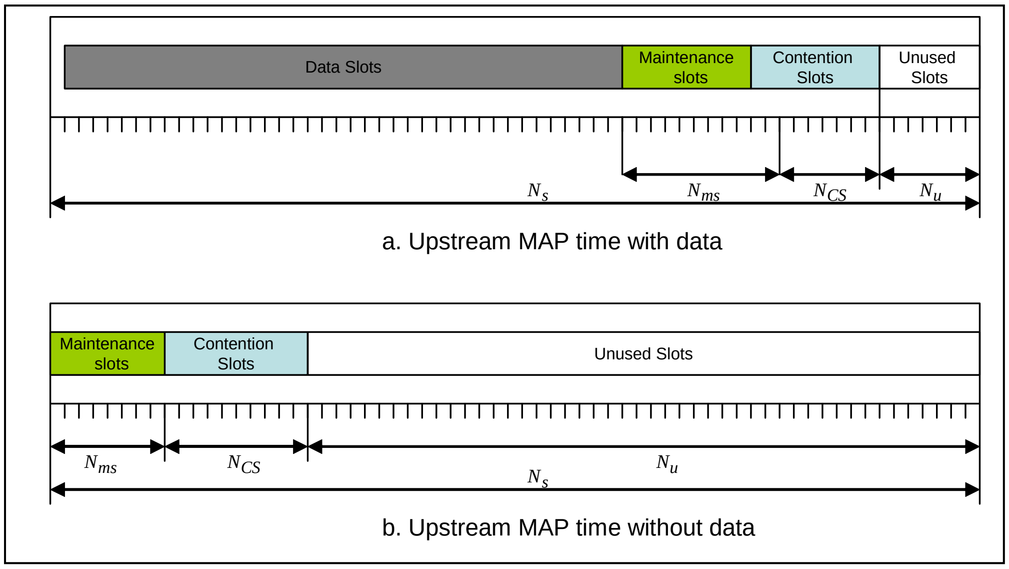


Figure 2. Example DOCSIS MAP layout

C. Upstream Analytic Model

The upstream behavior of DOCSIS is similar to slotted aloha with reservations[10,11]. Following the analysis presented in [12], we define the maximum application throughput, $T_{\max us}$, to be $\frac{D_{perCycle}}{t_{data} + t_{request}}$ where $D_{perCycle}$ is the number of user data packets sent upstream in one reservation request cycle, t_{data} is the upstream transmission and propagation time of the data and $t_{request}$ is the delay associated with the request process. In this section, we present an analytic model that computes the upstream throughput that could be obtained by a single flow. The analysis does not consider delay or loss caused by multiple flows.

Figure 1 illustrates the $t_{request}$ and t_{data} delays in an upstream operation. The $t_{request}$ represents the total delay experienced by the packet from arrival at the CM until when the first bit of the packet is transmitted upstream. The t_{data} is the transmission and propagation time of the upstream data frame. Figure 2a illustrates the MAP layout when it has data grants. Figure 2b illustrates the layout when there are no data grants. As mentioned, the scheduler will normally allocate unused slots for contention. We define the variable N_s as the number of slots in a MAP, the variable N_{ms} as the number of management slots and the variable N_{cs} as the minimum number of contention slots per MAP. Initially we assume that the CMTS does not allocate unused slots for contention requests.

In this paper we only consider the backlogged case where the CM always has 1500 byte IP packets to send. Figure 3 illustrates the *(j-1)'th* MAP time during which the CM transmits data and also the next contention request. We define $D_{total\,\mathrm{Re}questDelay}$ as the total number of slots delayed from when a packet arrives until when the packet is transmitted. The delay, which is $(t_4 - t_2)$ in Figure 3, includes a static portion, the configured number of management slots, N_{ms}, and a random portion associated with the CS backoff.

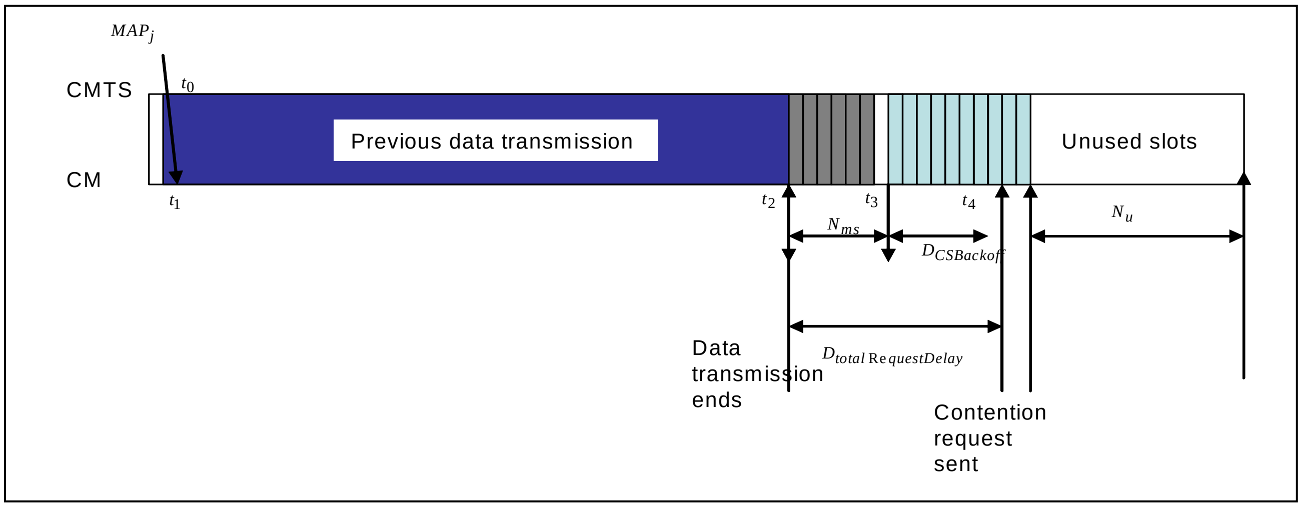


Figure 3. Backlogged contention request delays

Once an upstream packet arrives at the CM, the CM randomly computes the number of contention slots it will observe before transmitting a bandwidth request. We represent this delay with the random variable $D_{CSBackoff}$ whose range is determined by the parameter *backoffRange*[2]. The expected value is $E[D_{CSBackoff}] = \frac{backoffRange}{2}$. Depending on the value of the *backoffRange* and the N_{cs}, the backoff delay might extend over multiple MAP times. We define $E[N_u] = \max(N_s - N_{data} - N_{cs} - N_m, 0)$. This represents the number of unused slots in a map. In the backlogged case, the $D_{total\,RequestDelay}$ must consider three subcases depending on if the contention request gets sent during the current MAP, during the next MAP, or during a future MAP. We formulate the probabilities of being in each subcase.

Case a. If the selected contention slot is in the current MAP $mapTime_j$

[2] The actual range is $2^{backoffRange}$ but for simplicity we use *backoffRange*. This parameter specifies the maximum number of backoff contention slots the CM must count before attempting a contention request.

$p_a = 1$ if $backoffRange \le N_{cs}$

$p_a = \dfrac{N_{cs}}{backoffRange}$ if $backoffRange > N_{cs}$

Case b. If the contention slot is in the next MAP ($mapTime_j$)

$p_b = 0$ if $backoffRange \le N_{cs}$

$p_b = \min\left[\dfrac{backoffRange - N_{cs}}{backoffRange}, \dfrac{N_{cs}}{backoffRange}\right]$ if $backoffRange > N_{cs}$

Case c. If the selected contention slot falls in a MAP at least 2 MAP times in the future ($mapTime_{j+1}$ or later)

$$p_c = \max\left[\frac{backoffRange - 2N_{cs}}{backoffRange}, 0\right]$$

The expected delay from when the data transmission terminates until when the request is sent includes the weighted delays for each subcase.

$$E[D_{total\,\mathrm{Re}\,questDelay}] = p_a\left[N_{ms} + E[D_{CSBackoff}]\right] +$$
$$p_b\left[N_{ms} + N_{cs} + N_u + N_{ms} + N_{cs} - rem\left[\frac{E[D_{CSBackoff}]}{N_{cs}}\right]\right] +$$
$$p_c\left[N_{ms} + N_{cs} + N_u + \left[floor\left(\frac{backoffRange}{N_{cs}}\right)\right] * N_s + N_{ms} + N_{cs} - rem\left[\frac{E[D_{CSBackoff}]}{N_{cs}}\right]\right]$$

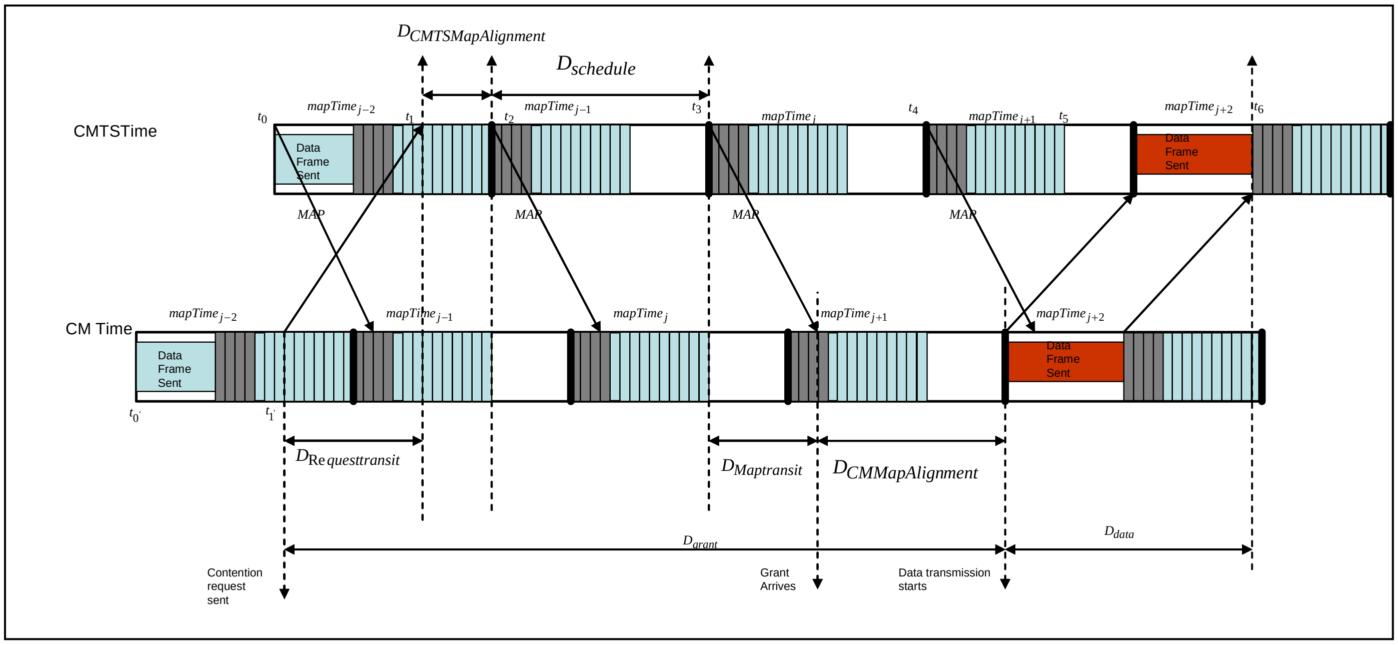


Figure 4. Backlogged grant delay

Figure 4 illustrates the additional components of delay once the contention request is sent. We define the variable D_{grant} to represent the delay from when the CM transmits the request until it begins transmitting the data frame. To help illustrate the impact of the propagation delay we show that the CM's view of the TDMA frame timing (labeled CM Time) must be properly adjusted for propagation delay. For example, the CMTS receives a contention request from a CM at time t_1. In order for the CM's request to arrive at the CMTS at time t_1 the CM must transmit at it's local time t_1'. The time difference $t_1 - t_1'$ reflects the propagation delay between the CM and the CMTS.

The expected value of D_{grant} consists of four delay components (all defined in units of slot times):

$$E[D_{grant}] = D_{\mathrm{Re}questtransit} + E[D_{CMTSMapAlignment}] + D_{schedule} + D_{Maptransit} + D_{CMMapAlignment}$$

- $D_{\mathrm{Re}questtransit}$: Represents the transmission and propagation time experienced by the contention request. A bandwidth request frame fits in one slot which leads to $D_{\mathrm{Re}questtransit} = t_{slot} + t_{prop}$.

- $D_{CMTSMapAlignment}$: Represents the number of slots from when the CMTS receives a request from a CM until the beginning of the next MAP. The value depends on if the contention request arrives during a MAP in which a data packet from the CM arrived (as in Figure 4) or it the CMs backoff value caused it to send the request in a subsequent MAP time. For the former case, the expected value is $D_{CMTSMapAlignment} = N_{cs}/2 + N_u$ (i.e., N_u is the number of unused slots in a MAP time when an upstream transmission occurs). For the latter case, the value is $D_{CMTSMapAlignment} = N_{cs}/2 + N_u + N_s - (N_{cs}/2 + N_{mgt})$. The first case occurs with probability $p = \min\left(\frac{N_{cs}}{backoffRange}, 1\right)$. Therefore,

$$E[D_{CMTSMapAlignment}] = p(N_{cs}/2 + N_u) + (1-p)(N_{cs}/2 + N_u + N_s - (N_{cs}/2 + N_{mgt}))$$

- $D_{schedule}$: Represents waiting time experienced by the request at the CMTS caused by scheduling delays. This delay will be a number of slots times that are integral multiples of the number of slots in a MAP. In other words, the granularity of the scheduling delays at the CMTS are in units of MAP times. For the analysis presented in this paper we assume that this delay is 0.
- $D_{Maptransit}$: Represents the transmission and propagation time (in slots) experienced by the MAP message in the downstream direction.
- $D_{CMMapAlignment}$: Represents the number of slots from when the MAP message arrives at the CM until the grant. This will be $D_{CMMapAlignment} = N_s - D_{Maptransit}$

We define T_{data} to represent the transmission and propagation delay of the data frame in the upstream direction. We assume that the grant is large enough to fit the data in a single frame (i.e., no fragmentation). We define $B_{request}$ as the total number of user data bytes delivered per data frame, $B_{overhead}$ as the total number of bytes of overhead associated with each frame and $B_{perslot}$ as the number of bytes that fit in each slot. N_{data}, the total number of slots consumed by the data transmission, is defined as:

$$N_{data} = \frac{B_{request} + B_{overhead}}{B_{perslot}}$$

The amount of time to send the frame from the CM to the CMTS is:

$$T_{data} = t_{slot} * N_{data} + t_{prop}$$

The total amount of time required to send the packet is:

$$T_{totalAccessDelay} = t_{request} + T_{data}$$

$$t_{totalAccessDelay} = E[D_{total\,\mathrm{Re}\,questDelay}] * t_{slot} + T_{data} + D_{grant} * t_{slot}$$

The maximum upstream throughput is:

$$T_{\max us} = \frac{B_{request}}{t_{request}}$$

Figure 5a and 5b illustrates the results of the analytic model and the simulation model respectively. The simulated network is illustrated in Figure 6. Only 1 FTP flow is active between CM-1 and the server S-1. No loss occurs and the upstream transmission queue at CM-1 is always filled with data ready to be sent upstream. We performed 9 runs varying the MAP time parameter from a value of .001 seconds to a maximum value of .012 seconds . Figure 5a was configured with a starting backoff range of 8 slots. This was increased to 64 slots for the experiments illustrated in Figure 5b. The analytic model accurately matches the simulation results in both cases. The results show that increasing the backoff range reduces throughput. This is because it increases the access delay. The advantage of this parameter is apparent when the system becomes heavily congested.

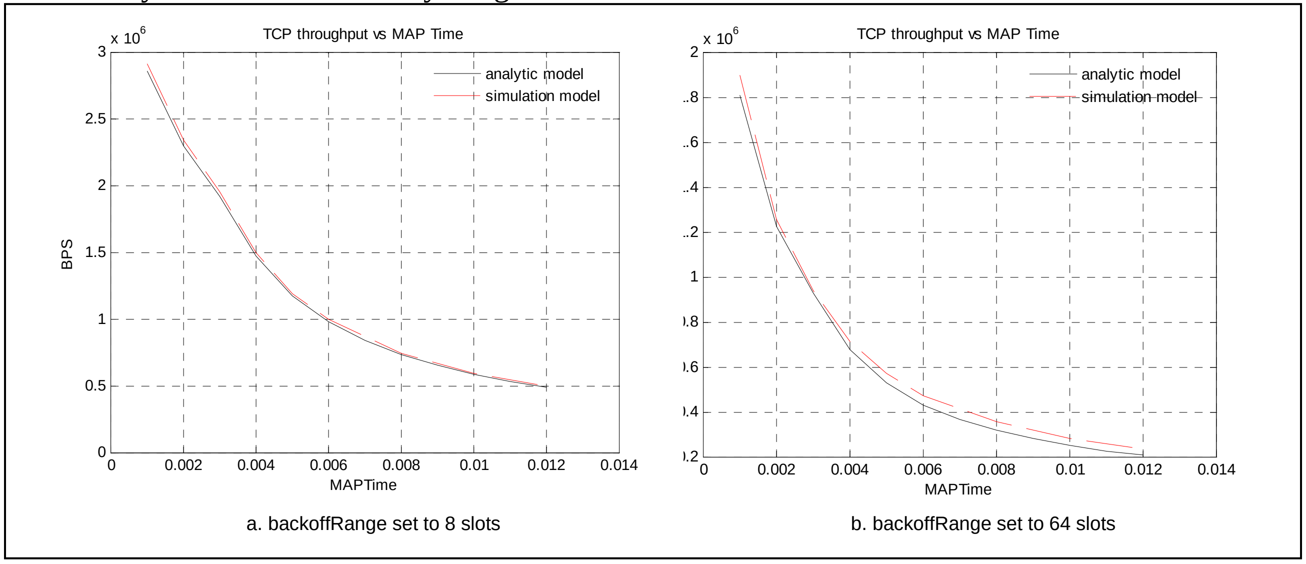
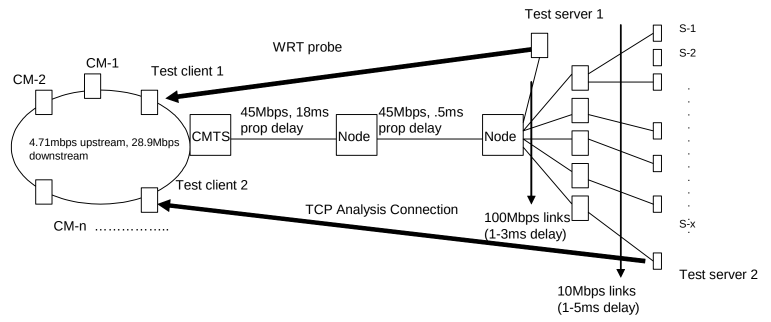


Figure 5a. backOffDelay = 8 Figure 5b. backOffDelay=64

Figure 5. Model results with fixed number of contention slots



Model Parameters

Upstream bandwidth 5.12Mbps
Preamble 80 bits
Downstream bandwidth 30.34Mbps
5 ticks per minislot
Default map time: 2 milliseconds (64 minislots per map)
Fragmentation Off, MAP_LOOKAHEAD = 255 slots
Concatonation ON
Backoff Start: 8 slots, Backoff stop: 128 slots
12 contention slots (minimum), 3 management slots
Simulation time: 200 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 6. Simulation network model and parameters

The simulation model supports a variable number of contention slots each MAP by allocating unused slots for contention requests. The analytic model becomes less complicted in this case. The number of contention slots increases by the number of unused slots in a MAP. We define a modified N_{cs}, $N'_{cs} = N_{cs} + N_u$ where $N_u = \max(N_s - N_{data} - N_{cs} - N_{ms}, 0)$. In this case, there are only two subcases: either the selected contention slot falls in the current MAP or it falls in a future MAP.

$$p_a = 1 \qquad \text{if } backoffRange \le N'_{cs}$$

$$p_a = \frac{N'_{cs}}{backoffRange} \qquad \text{if } backoffRange > N'_{cs}$$

$$E[D_{total\,\mathrm{Re}\,questDelay}] = p_a\lfloor N_{ms} + E[D_{CSBackoff}]\rfloor +$$
$$(1 - p_a)\left[E[D_{CSBackoff}] + (\,floor\left[\frac{E[D_{CSBackoff}]}{N_s - N_{ms}}\right] + 1)N_{ms}\right]$$

The computation of D_{grant} is the same as in the previous case with the exception of $D_{CMTSMapAlignment}$. There still are two cases to consider, the probability of each is $p = \min\left(\frac{N'_{cs}}{backoffRange}, 1\right)$.

$$E[D_{CMTSMapAlignment}] = p(N_s - N_d - N_{ms} - E[D_{CSBackoff}]) + (1 - p)(N_s - (E[D_{CSBackoff}] - N'_{cs}))$$

Figure 7a illustrates the results of the both the analytic and simulation models when unused slots are allocated for contention requests. The starting backoff delay range was 64 slots. Figure 7b is the identical experiment except piggybacking is enabled. As mentioned earlier, DOCSIS provides two mechanisms to avoid contention: piggybacking and concatenation. We have shown that a backlogged CM will send every other MAP time with or without piggybacking as long as the backoff range does not exceed the number of contention slots available each MAP time. However, as the backoff range is increased beyond the number of contention slots the CM will experience multiple MAP delays inbetween transmissions. Piggybacking allows the CM to send every other MAP independent of the backoff range. Our analytic model captures this effect by setting $D_{total\,\mathrm{Re}\,questDelaya}$ and $D_{transit}$ to 0. Figure 7b shows that, as expected, piggybacking can increase upstream throughput. The performance improvements attributed to piggybacking (and concatenation) are more significant in different traffic scenarios when many small packets are sent upstream (i.e., like TCP Ack packets). The analytic model can account for concatenation by increasing the amount of data transferred each cycle (i.e., $B_{request}$). For brevity, we do not show these results.

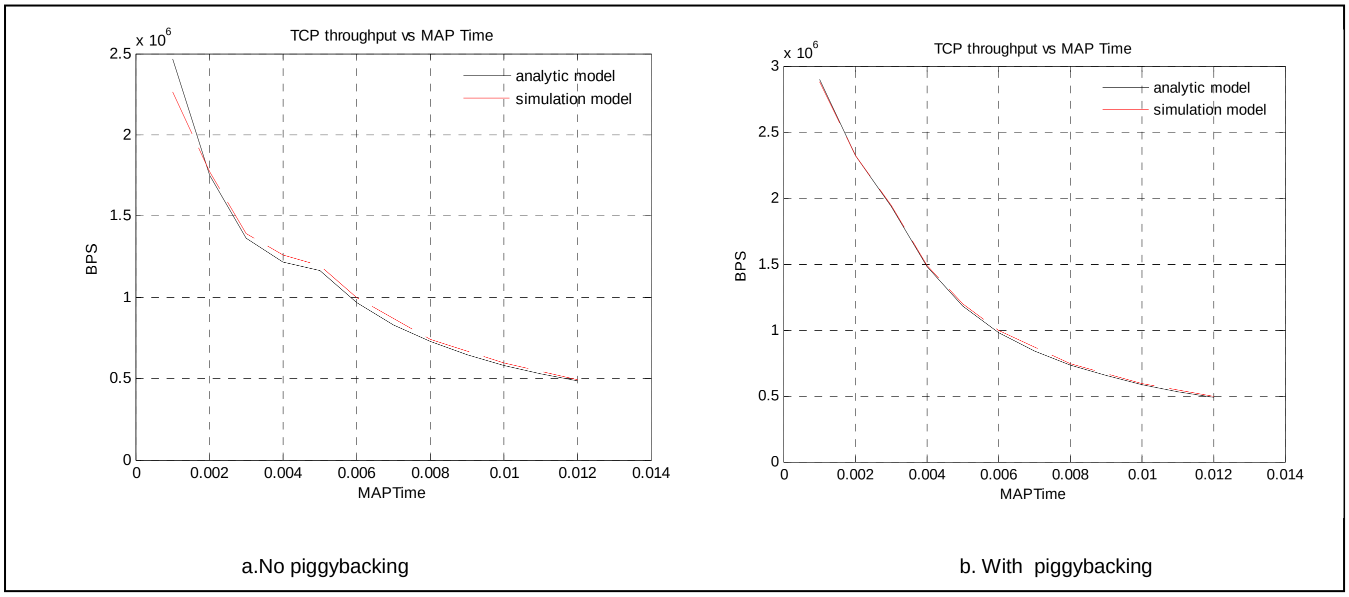


Figure7a. No piggybacking Figure 7b. With piggybacking

Figure 7. Model results with a dynamic number of contention slots and with piggybacking

D. Live Network Model Validation

To further validate the model, we compare simulation results with observations from a live cable network. We conducted experiments using a simple UDP echo application between a Linux machine (the client) located on the campus of Clemson and a second Linux machine (the server) located in a residential network connected to the Internet with Charter's cable Internet access service. The service provides 5Mbps downstream rate and 512Kbps rate upstream. The client sends a periodic stream of small (64 byte) UDP packets to the server that echoes the packets back. We obtain a tcpdump trace at both the client and the server.

Figure 8a through 8d visualizes both one-way streams at the respective send and receive sides. The client sends a packet every 2 milliseconds. The figures plot the distribution of the interpacket departure (or arrival) times. Figure 8a shows that at least 97% of the interpacket departure times are within 200 microseconds (the bin size) of their expected value. For the purposes of this experiment, the send time accuracy is sufficient. Figure 8b shows that packets traveling over the path were subject to both compression and delay. Figure 8c shows minor additional distortion caused by processing overhead at the Linux server. Figure 8d shows the impact of DOCSIS on the upstream UDP flow. The bandwidth consumed by in the upstream path is roughly 275Kbps (accounting for headers) which would not overwhelm the upstream channel. We confirm this by verifying that minimal loss occurs. Based on Figure 8d we conjecture that the DOCSIS network uses a MAP time of 2 milliseconds. The mode of .004 seconds represents the two MAP times that are required to send upstream data (even if piggybacking is used). The large mode at 0 seconds indicates that roughly 50% of the echo packets are being sent two at a time in a concatenated frame. Concatenated packets arrive at the client separated by the transmission time of the bottleneck link over the path between the client and the CMTS (which we estimate to be 45Mbps).

Figure 9a and 9b illustrates comparable simulation results. We configured the CM-1 node (refer to Figure 6) with a CBR traffic source that sends a 64 byte packet every .002 seconds. To model the observed randomness associated with UDP echo packets that arrive at the CM for upstream transmission (i.e., Figure 8c), we add an artificial jitter based on a normal distribution with a mean of 0 and a standard deviation of .0001. Figure 9a plots the interpacket departure time distribution from the CBR source and Figure 9b plots the interpacket arrival distribution at the UDP sink (S-1). Comparing Figure 9a with Figure 8c suggests that a normal distribution is correct although we could use a larger spread. Comparing Figure 9b to with Figure 8d confirms that the Charter network uses a MAP time of .002 seconds and that concatenation is common.

We next subject the server in the live cable network to a stream of periodic UDP packets sent every 5milliseconds. Figure 10a through 10d shows the four packet arrival and departure time distributions. Figure 10d again suggests the network is configured with a MAP time setting of .002 as some number of UDP packets are sent .002 milliseconds after the main mode of .004 seconds. Figure 11a and 11b show the results of a comparable simulation run. The CBR source is configured to add an artificial jitter based on a normal distribution with a mean of 0 and a standard deviation of .0005. To accurately model the randomness associated with Figure 10d we turned on 200 competing CMs that generated realistic amounts of Web traffic. Comparing Figures 11a with 10c suggest that the simulator accurately models the arrival process of packets sent from the echo server to the CM. Comparing Figure 11b with 10d shows that the simulation model behaves in a reasonable manner. Without adding the additional CM traffic the spread around the modes in Figure 11b was not visible with a 200 microsecond bin size. To determine if the live network is using piggybacking we repeated the simulation associated with Figures 11a and 11b but we disabled piggybacking. In Figure 11b we see roughly 4% of packet interarrival times were back-to-back implying a small amount of concatonation was occurring. When we repeat the experiment without piggybacking we saw the level of concatonation grow much larger (17%). From this experiment we conclude that if piggybacking were not used in the live network, we would see a significant level of concatonation. Since we do not see evidence of concatonation in Figure 10d we conclude that piggybacking was used.

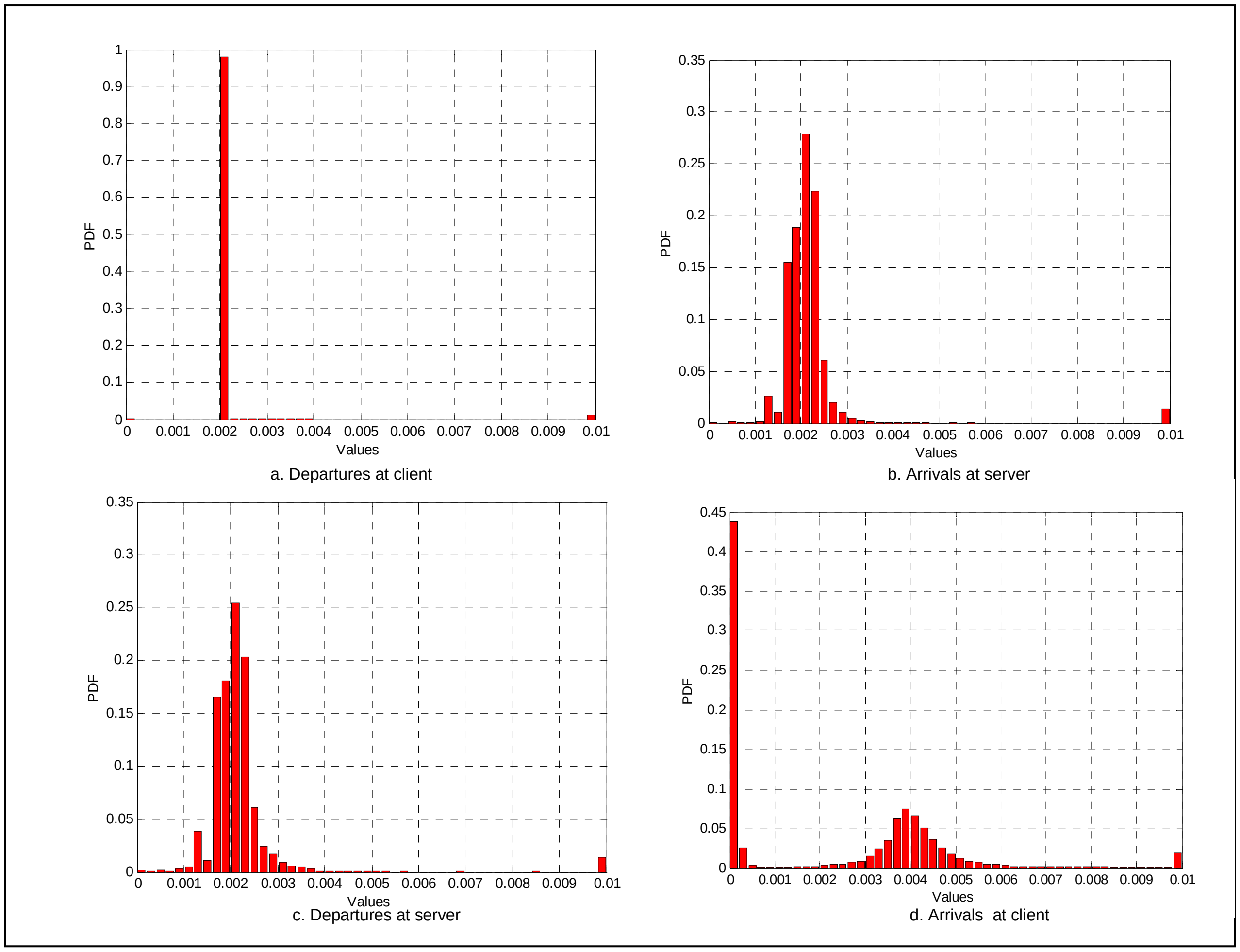


Figure 8. Cable network observed departure and arrival distributions (2ms packet spacing)

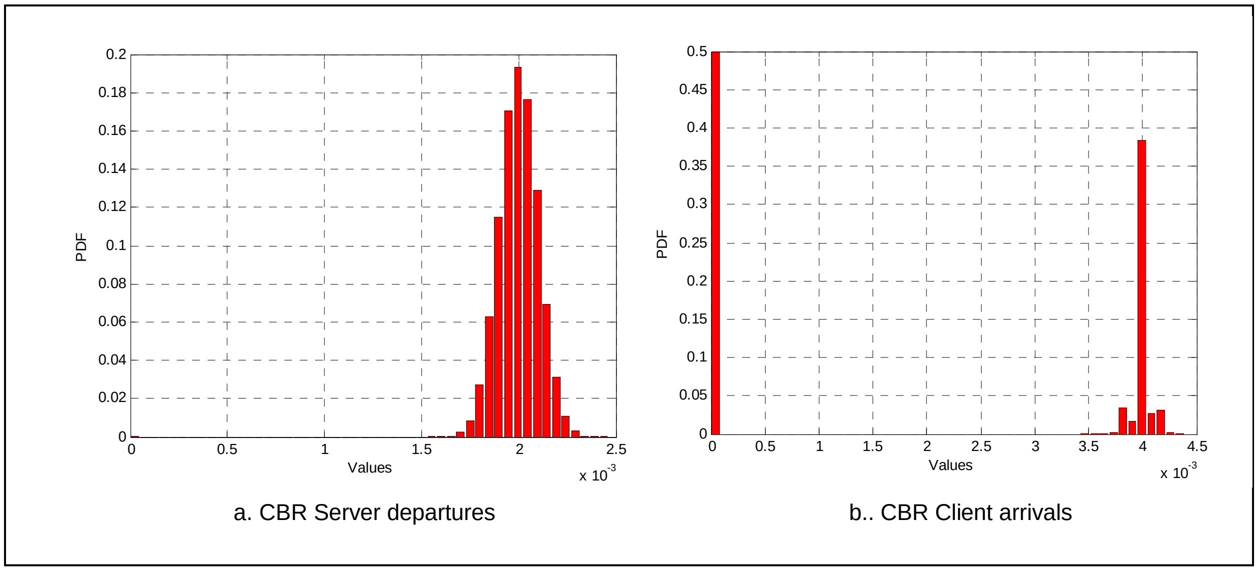


Figure 9. Simulation results (2ms packet spacing)

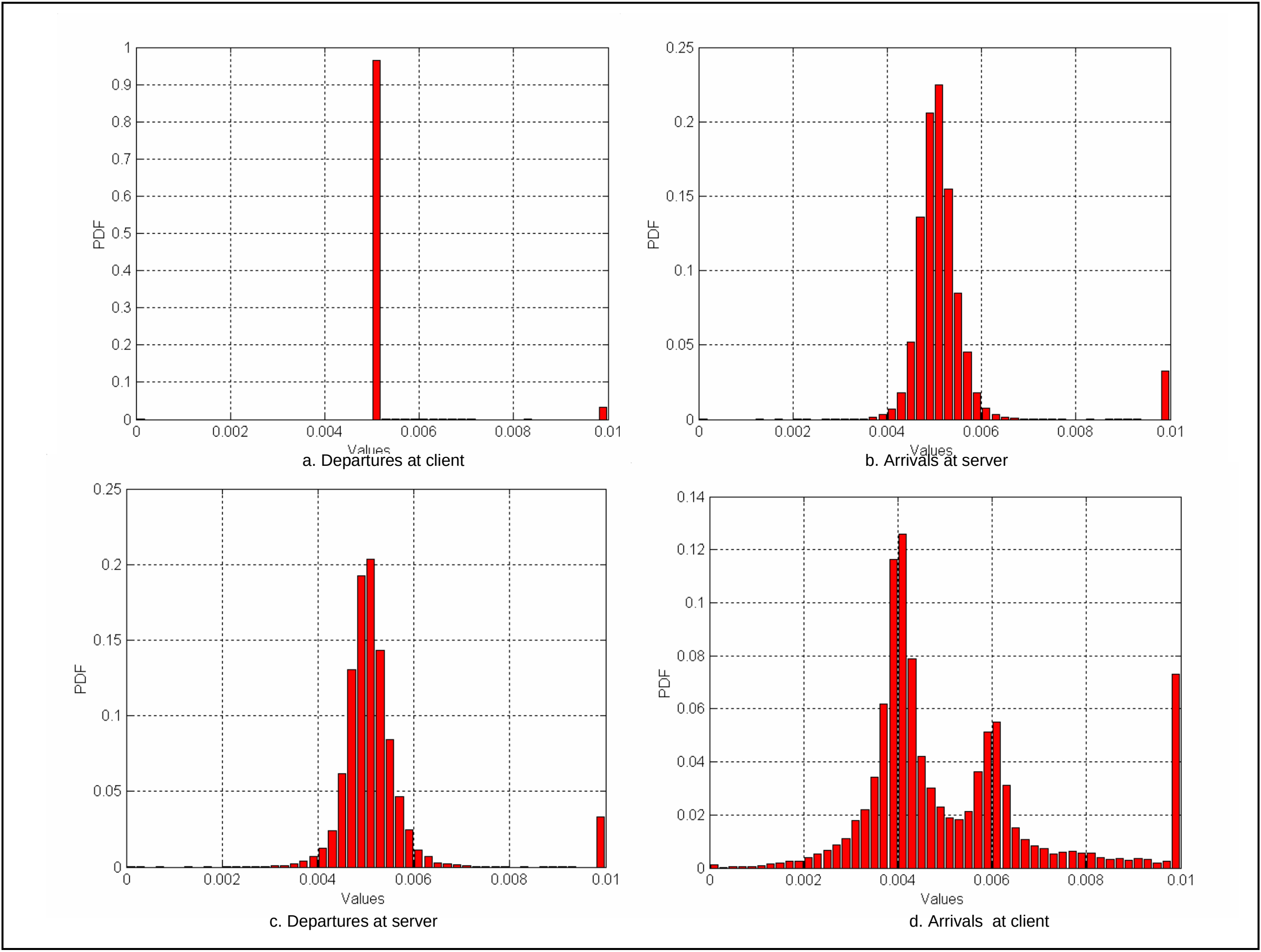
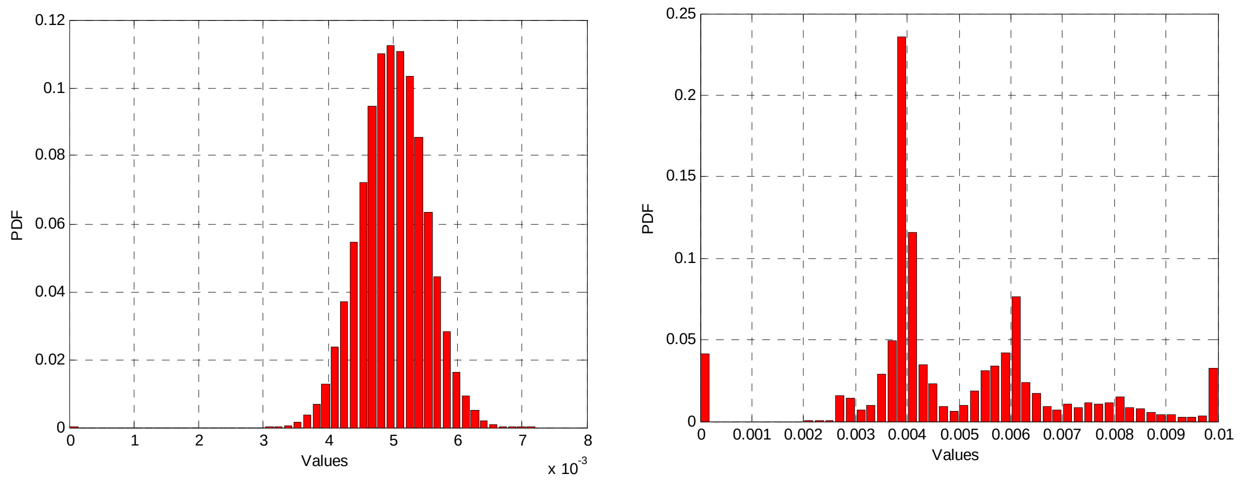


Figure 10. Cable network observed departure and arrival distributions (5ms packet spacing)



a. CBR Server departures b.. CBR Client arrivals

Figure 11 Simulation results (5ms packet spacing)

E. Performance Evaluation

For the analysis presented in this section, we rely on the simulation model illustrated in Figure 6. Up to '*n*' cable modems (CMs) generate upstream web requests to one or more servers (S-x nodes). Figure 6 describes the DOCSIS configuration and the web traffic model parameters. The web traffic model, which is based on the model described in [13], is designed so that each CM generates traffic that is similar to that produced by a broadband access subscriber. In addition to 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 [14]) 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. We also have a VoIP flow run between CM-1 and S-1. We model this flow using a 56Kbps CBR generator bound to a UDP sending agent located on CM-1. We performed the experiments twice: first by assigning the VoIP traffic to a UGS flow and second by classifying the VoIP stream as a best effort flow.

We define an experiment to consist of 5 simulation runs with each run configured with a larger number of CMs (i.e., 100 to 500). We perform 6 experiments with each experiment configured with a different MAP time setting (.001 to .01 seconds). We collect a set of statistics including the collision rate, channel utilizations, and CBR statistics (loss, latency and jitter) for each run.

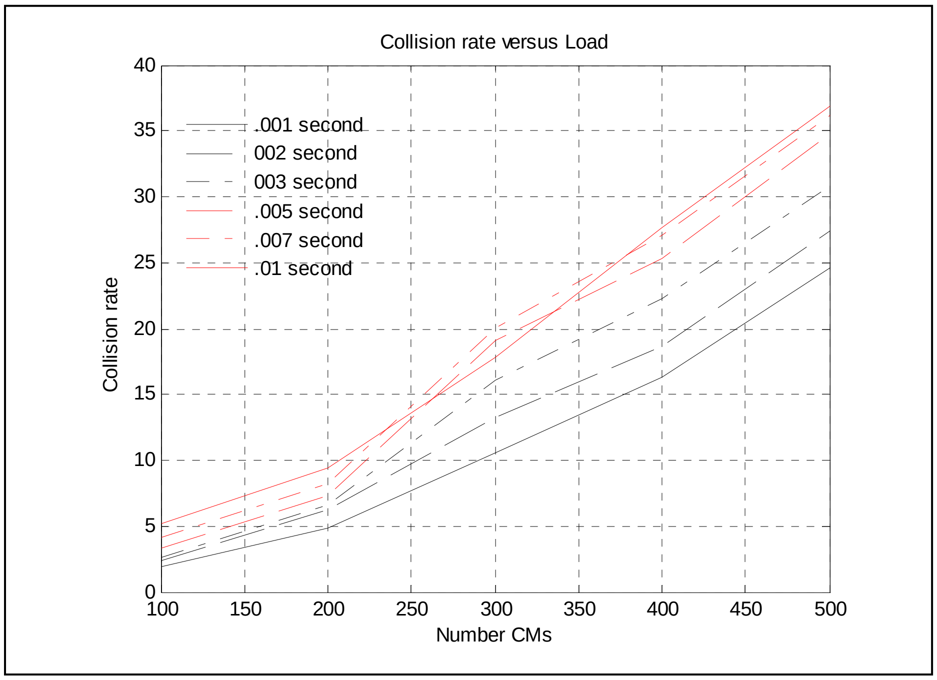


Figure 12. Upstream collision rates

In previous results [6] we found that the collision rate approached 70% as the load increases. The experiments described in this section are based on our most recent model which differs from our prior studies in two significant ways: 1)it allocates unused slots for contention requests; 2)the number of IP packets allowed in a concatenated frame is no longer limited to two. The results displayed in Figures 12 through 14 are based on the model parameters described in Figure 7 except that concatenation is disabled. Figure 12 shows that collision rates range from 2% to 37%. Allocating unused slots for contention requests signficantly improves performance from our previous study [6]. The collision rates were lowest for smaller MAP times. This is primarily because as the system gets busy the number of unused slots gets smaller. In the worst case, a MAP time will provide the minimum number of contention request slots (12 in the example) but the absolute number of contention request opportunities (i.e., the bandwidth allocated for contention requests) is greater for small MAP times. Figures 13 and 14 however shows that the MAP time has little impact on channel utilizations. Concatenation was disabled, however piggybacking was highly effective. 50%-90% of all packets sent upstream used a piggyback bandwidth request. We reran the experiments with concatenation enabled and saw similar results with the exception that extreme levels of TCP ACK compression occurred. However, since all nodes in the simulator were configured with adequate buffers, performance was not impacted by the compression.

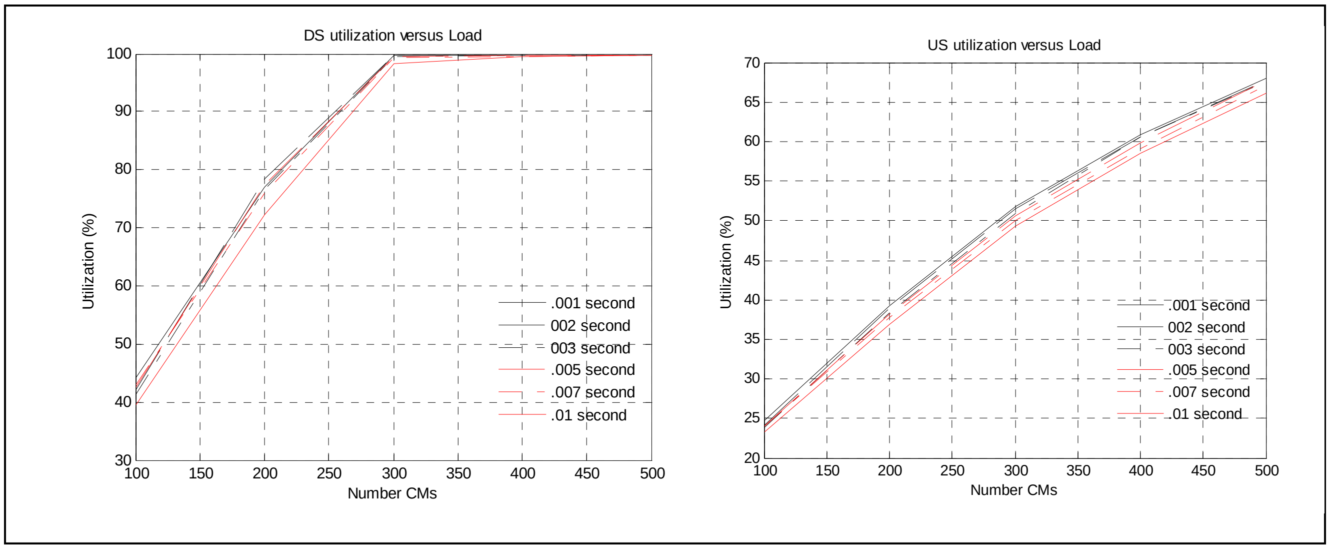
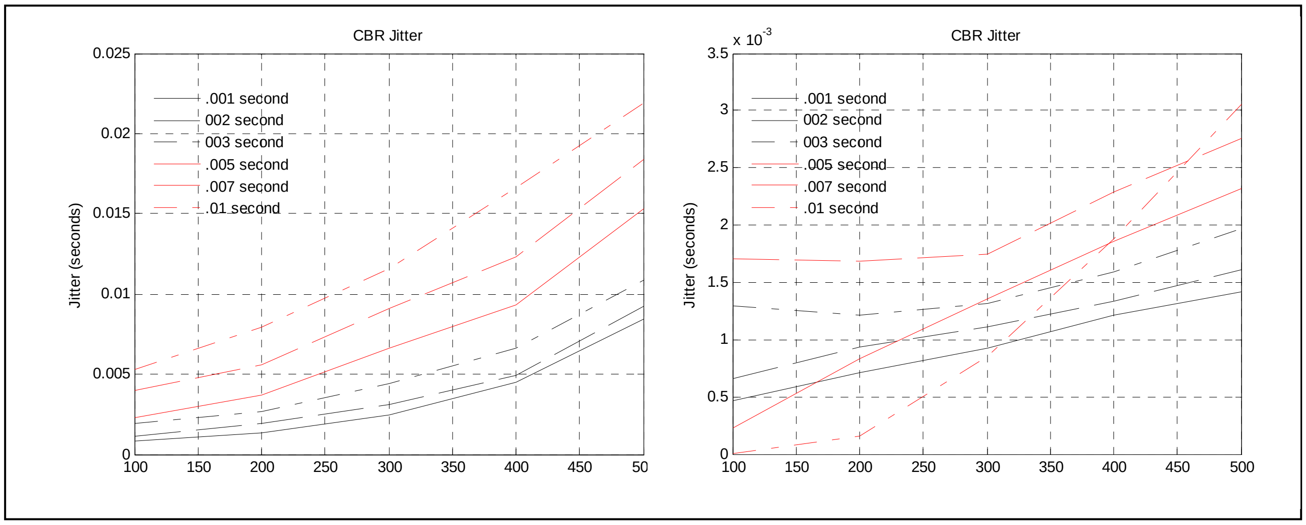


Figure 13a. Downstream Figure 13b. Upstream

Figure 13. Channel utilizations



a. Best Effort CBR flow b. UGS flow

Figure 14. Jitter experienced by a VoIP flow

Figure 14a illustrates the jitter associated with the simulated VoIP flow when mapped to a best effort service flow. For a reasonable MAP time setting of .002 seconds, the jitter ranged from 1 to 10 milliseconds. The associated end-to-end one-way delay ranged from 10 to 20 milliseconds. The loss rate experienced by the CBR flow never exceeded .35%. These results suggest that even a loaded DOCSIS network might not severely impact the perceived quality of a best effort VoIP call. This result is very limited however as there are many factors that could lead to poor call quality. Different subscriber traffic characteristics could substantially deteriorate network performance. Further, if the subscriber is generating other upstream traffic and the VoIP router does not isolate voice traffic from best effort traffic then latency and loss are likely to significantly increase. Figure 14b shows the results of the experiments when the VoIP session is mapped to a UGS flow. The jitter never exceeded 3 milliseconds, the one-way delay was less than 12 milliseconds and the loss rates were negligible.

F. Conclusions and Future Work

We have presented the design and validation of an '*ns*' DOCSIS simulation model. With analytic analysis and with analysis based on live cable network experiments, we have provided evidence that the simulation model behaves reasonably. To demonstrate the model, we examined the performance of a realistic scenario involving Web, P2P and VoIP traffic. In addition to providing insight into the behavior of a DOCSIS system, our performance analysis showed that a best effort VoIP flow can perform reasonably well even when subject to a significant amount of competing network traffic.

Our analytic modeling was limited to single flow scenarios. In the future we will consider the performance of a DOCSIS system subject to competing flows of a specified traffic intensity. Developing an accurate DOCSIS simulation model is the first step in this research project. The overall objective is to explore next generation cable network systems. Vendors are proposing DOCSIS enhancements that will drastically increase data rates. More complex modulation schemes, larger bandwidth allocations and upstream channel bonding will increase data rates to 200Mbps. The DOCSIS MAC protocol, however, will experience scaling issues as the data rates increase. We showed that upstream TCP throughput is limited to less than 3Mbps regardless of provisioned service rates. While future, higher capacity systems will be able to support more users sharing the channel, upstream packet rate limitations will make it difficult for DOCSIS networks to support a small number of high bandwidth applications (e.g., IPTV). To address this we are exploring a predictive data grant service that can efficiently support next generation broadband applications.

G. 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.com/specifications/ specifications20.html.

[2]CableLabs, www.cablelabs.com

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

[4]G.Chandrasekaran, M. Hawa, D. Petr, "Preliminary Performance Evaluation of QoS in DOCSIS1.1", ITTC-FY2003-TR-22736-01, Information and Telecommunication Technology Center, University of Kansas, Jan 2003.

[5] "The Network Simulator –ns2", UCB/LBNL/VINT http://www.isi.edu/nsnam/ns/

[6]J. Martin, "Modeling the DOCSIS 1.1/2.0 MAC Protocol", ICCCN03, October 2003.

[7]J. Martin, "The Impact of the DOCSIS 1.1/2.0 MAC Protocol on TCP", Proceedings of the IEEE Consumer Communications and Networking Conference, (Las Vegas, NV, Jan 2005).

[8] Vonage, www.vonage.com

[9] AT&T's VoIP, http://www.att.com/voip/

[10]N. Abramson, "The Aloha System – Another Alternative for Computer Communications", Proceedings of the Fall Joint Computer Conference, Jan 1970.

[11]S. Lam, L. Kleinrock, "Packet Switching in Multiaccess Broadcast Channel: Dynamic Control Procedures", IEEE Transactions on Communications, 23(9), 891-904, Sept 1975.

[12]D. Bertsekas, R. Gallager, Data Networks, Prentice-Hall, 1992.

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

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