Validating an ‘ns’ Simulation Model of the DOCSIS Protocol – Extended Version

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

Abstract— The number of households and businesses using HFC cable networks for Internet access is rapidly approaching 40 million 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). The emerging IEEE 802.16 broadband wireless access MAC protocol is based on DOCSIS. We have implemented a simulation model of the DOCSIS MAC 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 increasing use of broadband DSL and cable access networks is a major driver of the continuing robust growth of the Internet. According to a PEW survey, nearly 65% of adults access the Internet at least once each day [8]. There are 66 million households equipped with broadband access, and that approximately 54% of these households use cable.

The technology driving Hybrid Fiber Coaxial (HFC) cable networks is advancing at breathtaking speed. Since 1998, the cable industry has converged to a set of standards,
collectively referred to as Data-Over-Cable Service Interface Specification (DOCSIS), for supporting data over HFC cable networks [1,2]. The DOCSIS standards cover the physical, MAC layers, security, operations system support (OSS), equipment interfaces, and equipment validation. Sophisticated modulation techniques along with channel bonding will increase data rates from the current tens of Mbps to hundreds of Mbps in both the upstream and downstream directions.

Our research in DOCSIS-based shared medium networks is motivated by the observation that physical layer capabilities are advancing at a faster pace than advances in the MAC protocol. The current DOCSIS protocol is an optimized version of slotted Aloha with reservations proposed by Roberts in 1973 [11]. Additional mechanisms such as Unsolicited Grant Service (UGS), piggybacking, and concatenation enhance performance and the ability to provide QoS guarantees. Our research agenda explores the continued evolution of DOCSIS to better support current and next generation shared medium networks. Aspects of this research are applicable to 802.16 (WiMAX) networks as its channel allocation mechanism is also DOCSIS-based, but the focus in this paper is HFC networks.

DOCSIS systems are extremely complex. The DOCSIS HFC cable specification is a 500 page document. Due to the complexity and the cost, there are no open source DOCSIS platforms that are available to researchers. In the Internet community, academics can introduce new protocols or protocol enhancements through the IETF’s RFC process. This is not possible in the industry-centric HFC cable and WiMAX
communities where the standards are developed in members-only industry consortiums. As a result, the evolution of DOCSIS is being directed by industry with almost no involvement from academia. To address this we have developed a simulation model of the DOCSIS MAC protocol for the popular ‘ns’ simulator tool.

In previous work, we presented a preliminary ‘ns’ DOCSIS simulation model and showed that certain system parameters can significantly impact performance and dynamics of a DOCSIS system [12,13,14]. This motivated our analysis efforts of live DOCSIS networks. It has been pointed out that research that relies solely on simulation lacks credibility[29]. Recent publications note that the wireless research is particularly sensitive to physical layer assumptions [25,26,27]. The authors of [26] indicate that in addition to valid channel models, the choice of experimental scenarios is as crucial. The validation presented in this paper addresses the concerns by providing grounding the simulation model with data obtained from live cable network systems.

Validating a model of a DOCSIS system is difficult for a number of reasons. First the specification leaves significant room for implementation decisions. Second, the behavior of the system is very sensitive to parameter selection. Third, the academic studies of the physical layer characteristics of HFC and WiMAX channels have not been published. In the validation presented in this paper, we address the first two issues and leave the physical model development and validation for future work. To focus the validation on the MAC protocol we have configured the DOCSIS MAC layer to operating over the existing ‘ns’ LAN physical link. To validate the simulation model
we developed a set of simple analytic models that confirm basic upstream and downstream behavior. To validate the implementation decisions we made in the simulation models (and also related assumptions made in the analytic models), we analyze two live DOCSIS systems. Using a home network that connects to the Internet by Charter’s cable network, we conduct a measurement analysis designed to help us deduce implementation choices and system configuration settings of a deployed system. In addition, we have built a DOCSIS testbed using equipment donated by Cisco. By conducting further tests, we are able to learn additional implementation details that further helps us validate the simulation model operation.

The rest of this paper is organized as follows. After a brief background on DOCSIS, we present the operation and features of the DOCSIS model. We present an analytic model that captures the upstream behavior of DOCSIS and use it to demonstrate the simulation model works as intended. Next we describe the analysis involving an actual broadband access network and also involving our DOCSIS testbed. Then we present the results of a simulation analysis using the model involving hundreds of active CMs generating a mix of web, streaming, P2P and VoIP traffic. To demonstrate the model, we explore the use of best effort VoIP services over DOCSIS access networks. Companies such as Vonage and AT&T offer VoIP without engaging DOCSIS QoS mechanisms [15,16]. While this conflicts with the cable industry’s direction for telephony, best effort VoIP offers an inexpensive and popular alternative. Finally, we highlight related work and end the paper with conclusions and identify future work.
The cable industry has converged to the Data Over Cable System Interface Specification (DOCSIS) as the standard physical layer MAC protocols for data over HFC cable networks [1]. The DOCSIS standard was developed by the cable industry’s research consortium, CableLabs [2]. A multi-system cable operator (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 (one downstream channel with four upstream channels). Six Mhz (or greater) bandwidth is allocated from the 88-860 Mhz spectrum for each downstream channel and upstream channels are allocated from the 5 – 52 Mhz frequency range.

In the downstream direction, a single sender (the CMTS) transmits to a set of CMs using a data rate ranging from 10 Mbps to 50Mbps. IP packets sent downstream are divided into 188-byte MPEG frames. 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 that supports data rates in the range of 5 Mbps to 10 Mbps. The upstream transmission model is shared access using time division multiple access (TDMA) with a random access contention-based 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 by *service flow*. For example, a configuration that supports telephony and best effort data would typically 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 constant bit rate service), and non-realtime polling (nrtPS, which is equivalent to ATM’s non-realtime variable bit rate 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.

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 (unsolicited) 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 mechanism. 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 a grant or a grant pending from the CMTS, the CM assumes a collision occurs and doubles the size of the backoff window. 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. To further reduce contention, a CM is permitted piggyback a request for bandwidth on an upstream data frame.

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. An upstream channel capacity of 5.12 Mbps with a map time of 64 slots is assumed. To account for FEC overhead we reduce the upstream channel capacity by 8%. This approximation was suggested in http://www.cisco.com/warp/public/109/data_thruput_docsis_world_19220.shtml. The DOCSIS framing overhead adds an additional 30 bytes to an IP packet. A system tick of 6.25 microseconds and an effective channel capacity of 4.71Mbps leads to 18 bytes of data per slot for a total of 85 slots required for a 1500 byte IP packet.

The small dark square box positioned at the beginning of each MAP time represents the transmission of the MAP message in the downstream direction. The CMTS (in our model) sends a MAP message at the beginning of each MAP time. Each MAP message 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 $t_{\text{request}}$). 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 with variable MAP times](image)

C. 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.

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.

Continuing with the preceding example, a system that is configured for a nominal MAP time of 64 slots could define a MAP time that contains 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 allocates all unused slots for contention requests, but this behavior is configurable.

In prior work we found that the size of the MAP has tremendous impact on the behavior of DOCSIS [12,13,14]. Equally as important is the contention request allocation strategy and the use of fragmentation, concatenation and piggybacking.
The number of slots in a MAP

The number of management slots in a MAP

The number of contention request slots in a MAP

The number of unused slots in a MAP

The number of slots allocated for data grant

The number of slots in a MAP

The number of management slots in a MAP

The number of contention request slots in a MAP

The number of unused slots in a MAP

The number of slots allocated for data grant

### D. UPSTREAM ANALYTIC MODEL

The upstream behavior of DOCSIS is similar to slotted aloha with reservations[17,18,33]. Following the analysis presented in [19], we define the maximum upstream application throughput, $T_{\text{max,us}}$, to be $\frac{D_{\text{perCycle}}}{t_{\text{data}} + t_{\text{request}}}$ where $D_{\text{perCycle}}$ is the amount of user data sent upstream in one reservation request cycle, $t_{\text{data}}$ is the upstream transmission and propagation time of the data, and $t_{\text{request}}$ 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 application flow. The analysis does not consider loss or collision that might be caused by competing traffic. In this paper we only consider the backlogged case where a single CM always has 1500 byte IP packets to send. The model does support piggybacking and concatenation.

Figure 1 illustrates the $t_{request}$ and the $t_{data}$ 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 the propagation time of the upstream data frame. Figures 2a and 2b illustrates the MAP layout when there is a data grant and when there is not a grant respectively. Table 1 defines the variables used for slot allocations within a MAP. We perform the analysis with three slightly different assumptions. First, we assume that the CMTS does not allocate unused slots for contention requests. Second, we assume that the CMTS does allocated unused slots for contention requests. In the first two approaches we assume that piggybacking is not used. In the third and final analysis, we add piggybacking and concatenation.

**Analysis 1. no piggybacking, no concatenation, fixed number of contention slots**

The $t_{request}$ has two components, the $D_{total\ Request\ Delay}$ and the $D_{grant}$. The former is the total number of slots delayed from when a packet arrives until when the request for

---

1 While the single-flow assumption is clearly unrealistic in a real network, it allows us to compare the basic operation of the MAC protocol with live systems under controlled conditions. We are extending the analysis to include operation under realistic loads.
bandwidth is transmitted. For the backlogged case, this delay is simply the time from when the previous data transmission ends until when the bandwidth request is sent ($t_4 - t_2$ in Figure 3). The $D_{\text{total RequestDelay}}$ includes a static portion (the configured number of management slots $N_{ms}$) and a random portion (the collision backoff $D_{CSBackoff}$). The $D_{\text{grant}}$ represents the delay from when the CM transmits the request until it begins transmitting the data frame.

Figure 3. Backlogged contention request delays

Figure 3 illustrates the $(j-1)^{th}$ MAP time during which the CM transmits data and also the next contention request. Once a packet arrives at the CM for upstream transmission, 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 $D_{CS\text{Backoff}}$ whose range is determined by the parameter \( backoffRange \). The expected value is $E[D_{CS\text{Backoff}}] = \frac{backoffRange}{2}$. Depending on the value of the \( backoffRange \) and the \( N_{cs} \), the backoff delay might extend over multiple MAP times. The number of unused slots in a map is represented as \( N_u = \max(N_s - N_{data} - N_{cs} - N_m, 0) \). In the backlogged case, the \( D_{\text{total RequestDelay}} \) 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 each subcase.

**Case a.** If the selected contention slot is in the current MAP \( mapTime_{j-1} \)

\[
p_a = 1 \quad \text{if} \quad backoffRange \leq N_{cs}
\]
\[
p_a = \frac{N_{cs}}{backoffRange} \quad \text{if} \quad backoffRange > N_{cs}
\]

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

\[
p_b = 0 \quad \text{if} \quad backoffRange \leq N_{cs}
\]
\[
p_b = \min\left[\frac{backoffRange - N_{cs}}{backoffRange}, \frac{N_{cs}}{backoffRange}\right] \quad \text{if} \quad backoffRange > N_{cs}
\]

**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 is the weighted delays for each subcase.

---

\[2\] 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.
\[ E[D_{\text{total Request Delay}}] = p_a \left[ N_{ms} + E[D_{\text{CSBackoff}}] \right] + \]
\[ p_b \left[ N_{ms} + N_{cs} + N_u + N_{ms} + N_{cs} - \text{rem} \left( \frac{E[D_{\text{CSBackoff}}]}{N_{cs}} \right) \right] + \]
\[ p_c \left[ N_{ms} + N_{cs} + N_u + \left\lfloor \frac{E[D_{\text{CSBackoff}}]}{N_{cs}} \right\rfloor \cdot N_s + N_{ms} + N_{cs} - \text{rem} \left( \frac{E[D_{\text{CSBackoff}}]}{N_{cs}} \right) \right] \]
Figure 4. Backlogged delay from the time the request is sent until the transmission completes.

Figure 4 illustrates the additional components of delay once the contention request is sent. The example portrayed in the figure assumes the selected contention request slot falls in the first MAP time following the previous data transmission (i.e., case a described earlier). To help illustrate the impact of the propagation delay, Figure 4 shows that the CM’s view of the TDMA frame timing (labeled CM Time) must be properly adjusted for propagation delay. For example, in order for a CM’s contention request sent in the (j-2)nd MAP time \((\text{mapTime}_{j-2})\) to arrive at the CMTS at time \(t_r\), the CM must transmit at its local time \(t_r\). The time difference, \(t_r - t_r\), reflects the propagation delay \((t_{prop})\) between the CM and the CMTS. The CMTS must also account for propagation delay when computing future slot assignments. For example in Figure 4 at time \(t_2\) the when the CMTS generates the MAP, it must know how far in the future the MAP’s first time slot must be for. In other words, by the time the MAP arrives at the
CM, the slot assignments must still be in the future. The model assumes the CMTS allocates a data grant in the earliest MAP time possible.

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

$$E[D_{grant}] = D_{RequestTrans} + E[D_{CMTSMapAlignment}] + D_{Schedule} + D_{MapTransit} + D_{CMMapAlignment}$$

We summarize each component as follows:

- **$D_{RequestTrans}$**: Represents the transmission and propagation time experienced by the contention request. A bandwidth request frame fits in one slot which leads to

$$D_{RequestTrans} = \text{ceil} \left( \frac{t_{slot} + t_{prop}}{t_{slotTime}} \right).$$

- **$D_{CMTSMapAlignment}$**: Represents the number of slots from when the CMTS received a request from a CM until the beginning of the next MAP time. 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 if the request arrived during a MAP time which did not include a data grant. For the former case, the expected value is

$$D_{CMTSMapAlignment} = \frac{N_{cs}}{2} + N_u.$$  

For the latter case, the value is

$$D_{CMTSMapAlignment} = \frac{N_{cs}}{2} + N_u + N_s - \left( \frac{N_{cs}}{2} + N_{mg} \right).$$  

The first case occurs with probability $p = \min \left( \frac{N_{cs}}{\text{backoffRange}}, 1 \right)$. Therefore,

$$E[D_{CMTSMapAlignment}] = p \left( \frac{N_{cs}}{2} + N_u \right) + (1 - p) \left( \frac{N_{cs}}{2} + N_u + N_s - \left( \frac{N_{cs}}{2} + N_{mg} \right) \right)$$
• \( D_{\text{schedule}} \): 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.

• \( D_{\text{Maptransit}} \): Represents the transmission and propagation time (in slots) experienced by the MAP message in the downstream direction.

• \( D_{\text{CMMapAlignment}} \): Represents the number of slots from when the MAP message arrives at the CM until the grant. This will be \( D_{\text{CMMapAlignment}} = N_s - D_{\text{Maptransit}} \)

We define \( \tau_{\text{data}} \) 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 \( B_{\text{request}} \) as the total number of user data bytes delivered per data frame, \( B_{\text{overhead}} \) as the total number of bytes of overhead associated with each frame and \( B_{\text{perslot}} \) as the number of bytes that fit in each slot. \( N_{\text{data}} \), the total number of slots consumed by the data transmission, is defined as:

\[
N_{\text{data}} = \frac{B_{\text{request}} + B_{\text{overhead}}}{B_{\text{perslot}}}
\]

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

\[ T_{\text{data}} = t_{\text{slot}} \times N_{\text{data}} + t_{\text{prop}} \]

\[ T_{\text{totalAccessDelay}} = t_{\text{request}} + T_{\text{data}} \]

\[ t_{\text{totalAccessDelay}} = E[D_{\text{totalRequestDelay}}] \times t_{\text{slot}} + T_{\text{data}} + D_{\text{grant}} \times t_{\text{slot}} \]

The maximum upstream throughput is:

\[ T_{\text{max,us}} = \frac{B_{\text{request}}}{t_{\text{totalAccessDelay}}} \]

Figures 5a and 5b compare the analytic model results with results from a simple simulation experiment. The simulated network is illustrated in Figure 6. One FTP flow is active between CM-1 and the server S-1. The maximum TCP window configuration setting was 11 packets which was sufficient to keep the upstream transmission queue at CM-1 always filled with data but to avoid queue overflow. We disable piggybacking and concatenation in the simulation model so that it matches the assumptions made in the analytic model. We performed 9 runs varying the MAP time parameter from a value of .001 seconds to a maximum value of .012 seconds. Each run lasted 1000 simulated seconds with the TCP throughput result representing the mean throughput over the lifetime of the simulated connection.
The first set of results, illustrated in Figure 5a, used a `backoffRange` of 8 slots. This was increased to 64 slots for the second set of results illustrated in Figure 5b. The analytic model accurately matches the simulation results in both cases. The results suggest that increasing the `backoffRange` reduces throughput. Intuitively this makes sense since a higher `backoffRange` increases the `D_{total RequestDelay}` resulting in a lower throughput. The DOCSIS specification requires the CMTS to send the current `backoffRange` in each MAP message. A CMTS could, for example, increase the `backoffRange` as the collision rate increases in an effort to implement congestion avoidance.

![Figure 5a. backOffDelay = 8](image1.png) ![Figure 5b. backOffDelay=64](image2.png)

**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
Concatenation 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
Object size/page: pareto model, mean 3 and shape 1.5
Inter-object: pareto model, mean 3 and shape 1.5
Object size: pareto model, mean 12 (segments) shape 1.2

Figure 6. Simulation network model and parameters

Analysis 2. no piggybacking, no concatenation, variable number of contention slots

The simulation model supports either a fixed number or a variable number of contention slots within each MAP. The latter is supported by allocating unused slots for contention requests in addition to a fixed minimum number of contention slots in each MAP. The analytic model becomes less complicated when adapted for a variable number of contention slots. The number of contention slots increases by the number of unused slots in a MAP. We define a modified $N_{cs}$ as $N_{cs}' = N_{cs} + N_u$ 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.
\[p_a = 1 \quad \text{if} \quad \text{backoffRange} \leq N'_{cs}\]

\[p_a = \frac{N'_{cs}}{\text{backoffRange}} \quad \text{if} \quad \text{backoffRange} > N'_{cs}\]

\[E[D_{\text{total RequestDelay}}] = p_a[N_{ms} + E[D_{CSBackoff}]] + (1 - p_a)[E[D_{CSBackoff}]] + (\text{floor}\left(\frac{E[D_{CSBackoff}]}{N_{cs} - N_{ms}}\right) + 1)N_{ms}\]

The computation of \(D_{\text{grant}}\) is the same as in the previous case with the exception of \(D_{\text{CMTSMapAlignment}}\). There still are two cases to consider, the probability of each is

\[p = \min\left(\frac{N'_{cs}}{\text{backoffRange}}, 1\right)\] and \(1 - p\) respectively:

\[E[D_{\text{CMTSMapAlignment}}] = p(N_{s} - N_{d} - N_{ms} - E[D_{CSBackoff}]) + (1 - p)(N_{s} - (E[D_{CSBackoff}] - N'_{cs}))\]

Figure 7a illustrates the results of the analytic and simulation models when unused slots are allocated for contention requests. The \textit{backoffRange} was 64 slots.

\textbf{Analysis 3. piggybacking, no concatenation, variable number of contention slots}

As mentioned earlier, DOCSIS provides two mechanisms to avoid contention: piggybacking and concatenation. We have shown that a backlogged CM in an uncongested network will send every other MAP time with or without piggybacking as long as the \textit{backoffRange} does not exceed the number of contention slots available each MAP time. However, as the \textit{backoffRange} is increased the CM might experience multiple MAP delays between transmissions. Piggybacking allows the CM to send every other MAP independent of the \textit{backoffRange}.
Our analytic model captures this effect by setting $D_{total\ Request\ Delay}$ and $D_{transit}$ to 0. Figure 7b is the identical scenario except piggybacking was enabled. As expected, piggybacking can increase upstream throughput. Piggybacking is effective in backlogged scenarios. If CM traffic consists primarily of Internet web browsing sessions, most upstream traffic consists of bursts of TCP ACK packets. Concatenation is very effective at increasing upstream efficiency in this situation. 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 however we confirmed that the analytic and simulation models match when concatenation is enabled.

Figure 7a. No piggybacking

Figure 7b. With piggybacking

Figure 7. Model results with a dynamic number of contention slots and with piggybacking
E. **Live Network Model Validation**

To further validate the simulation model, we compare simulation results with observations from two live cable networks. The first network involved a pair of machines located at Clemson University and at a residential network connected to the Internet by Charter’s cable network. The second network involved a DOCSIS testbed located in our research lab.

*Network 1: Charter’s cable network*

We conducted experiments using a simple UDP echo application between a Linux machine (the client) located on the campus of Clemson University 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 a 5 Mbps downstream rate and a 512 Kbps upstream rate. The client sends a periodic stream of small (64 byte) UDP packets to the server which will echo the packets back. We obtain a tcpdump trace at both the client and the server [10].

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 roughly 97% of the interpacket departure times are within 200 microseconds (the bin size) of their expected value. This jitter is attributed to random delays that occur in the operating system. For the purposes of this experiment, the jitter associated with the
stream generated by the client is acceptable. 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 upstream bandwidth consumed is roughly 275 Kbps (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 Charter has configured their network for 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 about 50% of the echo packets are being sent back-to-back over the upstream channel in a concatenated frame. Concatenated packets arrive at the client separated by the transmission time of the bottleneck link over the path between the CMTS and the client (which we estimate to be 45 Mbps).

Figure 9a and 9b illustrates comparable simulation results. We configured the CM-1 node (shown in Figure 6) with a CBR traffic source that sends a 64 byte packet every .002 seconds. To model the random delay observed in Charter’s network associated with UDP echo packets that arrive at the CM for upstream transmission (i.e., Figure 8c), we add an artificial delay to each packet before transmitting at the server. The delay is 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 the spread needs to be larger. Based on the similarity between Figures 9b and 8d and because the simulation model was configured to use a MAP time of .002 seconds as well as concatenation, we conclude that the Charter network is configured similarly.

We next subject the server in the live cable network to a stream of periodic UDP packets sent every 5 milliseconds. 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. Comparing Figure 11a with Figure 10c shows that this fits the observed data well. However in order to obtain the spread around the modes at .004 and .006 seconds (in Figure 11d), we turned on 200 competing CMs that generated realistic amounts of Web traffic. Comparing Figure 11b with 10d shows that the simulation model behaves in a reasonable manner. Without the competing CM traffic the spread around the modes in Figure 11b was not visible with a 200 microsecond bin size.

To determine if Charter's network is using piggybacking, we repeated the simulation associated with Figures 11a and 11b but we disabled piggybacking. With piggybacking, as illustrated in Figure 11b, we see roughly 4% of packet interarrival times were back-to-back implying a small amount of concatenation was occurring.
When we repeat the experiment without piggybacking we saw the level of concatenation grow much larger (17%). We conclude that Charter’s network does use piggybacking because if it did not we would see a much higher amount of concatenation.

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)
Figure 11 Simulation results (5ms packet spacing)

**Network 2: DOCSIS testbed**

The testbed consists of four CMs (two Cisco uBR905 and two Motorola SurfBoard), a cisco uBR7110 CMTS and a simple RF plant. Each cable device connects to private, 100 Mbps Fast Ethernet networks. We installed Linux machines on all the Fast Ethernet networks. The downstream service rate is 10Mbps and the upstream service rates are 1Mbps. We repeated the experiments conducted in the Charter network. Figure 12 illustrates the results when the client (located on the Linux machine connected to the CMTS) sends a UDP packet to the Linux machine connected to one of the CMs every .002 seconds. The most striking difference is that the amount of variation in packet latency experienced by packets in the downstream direction is significantly less in the testbed. The CMTS has been configured to use a MAP time of .002 seconds which confirms our conjecture that the Charter network is based on a .002
second MAP time. As observed in Figure 8d, Figure 12d suggests that half the echoed UDP packets sent in the upstream direction are sent in concatenated DOCSIS frames. The CMTS has been configured to use concatenation. Figure 13 illustrates the interpacket departures and arrivals when the client sends a UDP packet every .005 seconds. Comparing Figure 13d with Figure 10d shows the same two modes at .0045 and .006 seconds. The CMTS has been configured to use piggybacking and therefore this confirms our conjecture that the Charter network does use piggybacking.

In previous work we concluded that the bottleneck in a DOCSIS system is the upstream channel and in particular its ability to transport packets at a high rate of speed. This upstream packet rate limitation impacts both downstream and upstream throughput. In the downstream direction, TCP throughput is limited by the rate at which TCP ACK packets can be sent over the upstream channel. For a MAP time of 0.002 seconds and a TCP/IP packet size of 1500 bytes, we found that the maximum application level downstream throughput has been shown to be limited to less than 6 Mbps. To confirm this we conducted a simple experiment using the testbed. We enabled a single TCP flow in the downstream direction and observed a maximum TCP throughput of 3.6 Mbps. We configured the identical network using our simulation model and also saw a maximum downstream throughput of 3.6 Mbps. Only by hand tuning system parameters, in particular disabling piggybacking and forcing DOCSIS to concatenate large number of ACK packets in a single upstream frame, could we approach a downstream throughput of 10 Mbps. Unfortunately concatenation can significantly
impact TCP dynamics by perturbing the TCP ACK spacing which has been shown to possibly lead to higher loss rate [22,23].

Figure 12  DOCSIS testbed observed departure and arrival distributions (2ms packet spacing)
Figure 13  DOCSIS testbed observed departure and arrival distributions (5ms packet spacing)

F. 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 [20], 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 56 Kbps audio stream), 2% of the CMs to generate downstream high speed UDP...
streaming traffic (i.e., a 300 Kbps video stream) and 5% of the CMs to generate
downstream P2P traffic. The P2P model (based on [21]) incorporates an exponential
on/off TCP traffic generator that periodically downloads on average 4 MBytes 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 56 Kbps CBR generator
bound to a UDP sending agent located on CM-1. We performed each experiment
twice, first assigning the VoIP sessions to a UGS service and second assigning VoIP
sessions to a best effort service.

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. All runs within
an experiment used the same MAP time. The MAP time was varied for each
experiment over a range from .001 to .01 seconds. All runs within an experiment used
the same MAP time. We collect a set of statistics including the collision rate, channel
utilizations, and CBR statistics (loss, latency and jitter) for each run.
In previous results [13] we found that the collision rate approached 70% as the load increased. The experiments described in this section are based on our most recent model which differs from our prior studies in three significant ways: 1) unused slots are allocated for contention requests; 2) the number of IP packets allowed in a concatenated frame is no longer limited to two; 3) the buffer size at the CMTS downstream queue is increased from 50 packets to 300 packets. The results displayed in Figures 14 through 16 are based on the model parameters described in Figure 6 except that concatenation is disabled. Figure 14 graphs six curves, one for each of the six experiments. The results show that collision rate ranged from 2% to 37%. Collision rates were lowest for the runs with smaller MAP times. As the system becomes busy the number of unused slots gets smaller which reduces the number of
contention request slots. In other words, the bandwidth allocated for contention slots is greater for small MAP times.

Figures 15 and 16 show that the MAP time has little impact on channel utilizations. Piggybacking was highly effective in this scenario. 50%-90% of all packets sent upstream used a piggyback bandwidth request. The runs with large MAP times were able to take advantage of piggybacking more than the runs with small MAP times because there is more time for packets to accumulate while waiting for a data grant. We reran the experiments with concatenation enabled and saw similar results with the exception that extreme levels of TCP ACK compression occurred. Since all nodes in the simulator were configured with adequate buffers, performance was not impacted by the bursty traffic dynamics caused by the ACK compression. However, it has been shown that ACK compression leads to higher loss rates and that it makes it difficult for protocols that estimate bottleneck bandwidths or that monitor packet delays to operate correctly [22,23,24].

![Figure 15a. Downstream](image1)

![Figure 15b. Upstream](image2)

Figure 15. Channel utilizations
a. Best Effort CBR flow                    b. UGS flow
Figure 16. Jitter experienced by a VoIP flow
Figure 16a 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 a congested 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. Further,
different subscriber traffic characteristics or DOCSIS network configuration could
substantially deteriorate network performance. Figure 16b 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.

G. RELATED WORK

In 1994 the IEEE 802.14 working group was chartered to develop a MAC layer that
would support both ATM and IP over HFC networks[9]. The upstream channel was
TDMA with a slot size of 8 bytes. ATM’s CBR, VBR, ABR and UBR services were
supported over the HFC network. Primarily due to time constraints, the cable industry supported the DOCSIS standard. However, because DOCSIS is based on 802.14, prior analysis of the IEEE 802.14 standard is relevant. In [3], the authors found that TCP throughput over an 802.14-based HFC network is low primarily due to ACK compression. 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. Other early studies looked at TCP enhancements when operating over HFC networks suffering from high channel error rates and the 802.14 contention protocol [4,5,6]. More recent analysis has been performed using an Opnet model of DOCSIS[7]. Most of these studies were limited in scope or are now simply outdated. Further, the Opnet DOCSIS model source code is not freely available limiting its use to capacity planning rather than protocol evolution. In order for the research community to participate in the advancement of DOCSIS-based broadband access networks, a fully functional, validated and documented simulation model is required.

H. 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 as expected. 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.
Because the analytic model is based on our understanding of the DOCSIS protocol, comparing results obtained from it to results obtained from the simulation model validates that we have implemented the DOCSIS protocol as we understand it correctly. To validate the model for correctness we relied on the analysis of live systems to provide confirmation that our implementation behaves correctly. The results provide confidence that our implementation, especially with respect to the MAP time duration and the usage of concatenation and piggybacking, is correct.

Finally, 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.

I. REFERENCES


