Tiers of Service for Data Access in a HFC Architecture

Kathleen M. Nichols and Mark Laubach
Com21, Inc.
750 Tasman Drive
Milpitas, CA USA 95035
+1 (408) 953-9100 / +1 (4085)953-9299 (tax)
nichols@com21.com / laubach@com21.com
URL http://www.com21.com/

Abstract

This paper discusses the Hybrid Fiber-Coaxial cable TV environment as a data delivery service providing subscribers with specific levels of service. We overview HFC for data delivery and Com21's UPSTREAMS architecture for HFC data delivery using an ATM cell based format. We discuss the evaluation of levels of service for individual subscribers and the need for both new simulation traffic models and for changing the common approach of evaluating communications systems with simulation models that have only one-way paths. Results and a new simulation traffic model, based on web browsing, are presented. Finally, we give some early results on scheduling at different qualities of service, previously presented in [1]. Our scheduling algorithm is based on the Class-Based Queueing work of Floyd and Jacobson [14] applied at the MAC layer to ATM cells.

1.0 Introduction

The IEEE 802.14 Cable TV Media Access Control [2] and Physical Protocol Working Group and the ATM Forum's Residential Broadband Working Group, for ATM over HFC, are both working on standardization of broadband data delivery. Many companies are participating in the standards process, some of these with cable modems already announced or on the market. The IEEE 802.14 Working Group is chartered with providing a single medium access control (MAC) and multiple PHY standard for cable TV networks. 802.14 must support IEEE 802 layer services and must also be *ATM Friendly.* The expected data interface of choice in the home is a 10 Mbps ethernet and attached devices are expected to be those which support the TCP/IP protocol suite.

The results presented in this paper are based on a Com21-developed protocol called UPSTREAMS: The Upstream Protocol for Sharing Transmission Resources among Entities using an ATM-based Messaging System [6]. This protocol has many features that are expected to be in the future IEEE 802.14 standard. In the next section, we overview Com21's approach, in section 3.0 we discuss some considerations for providing differentiated levels of service, section 4.0 presents our recent work on appropriate simulation traffic models to evaluate these architectures, and section 5.0 presents simulation results on scheduling for quality of service.

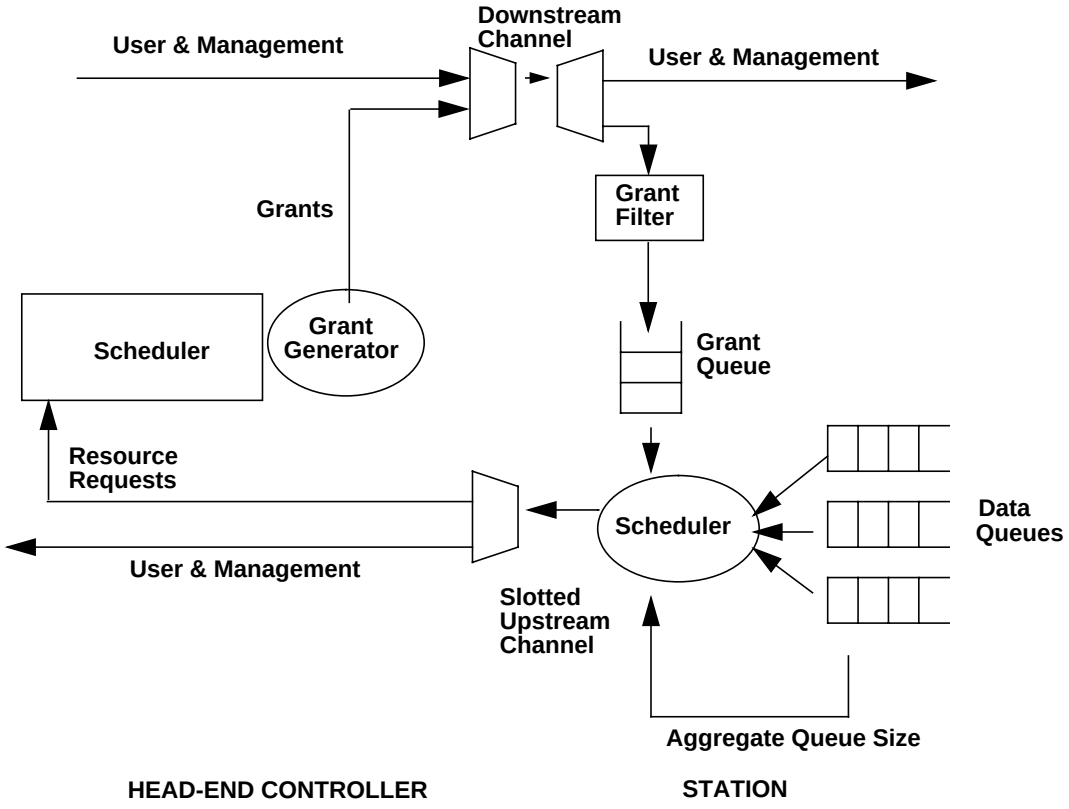
2.0 Overview of the UPSTREAMS protocol

In the HFC environment, stations cannot communicate directly, making it impossible to use a DQDB or CSMA/CD MAC layer. A slotted ALOHA technique might be used, but the standards call for supporting both connection-oriented traffic and connectionless traffic and high utilization of the upstream bandwidth is desired. UPSTREAMS uses a mac layer where all stations are precisely ranged to a time-slotted channel that is shared among the subscriber stations under the control of the head-end. The precise time-slotting prevents stations from interfering with one another's transmissions and the head-end control allows for dynamic allocation of the bandwidth among the subscriber units to provide the level of service that a particular type of traffic requires and for which a particular subscriber has paid. Subscriber terminals communicate with the head-end controller to request access to the upstream channel and the head-end controller's scheduling algorithm allocates these time slots among the stations using the requests from the stations and knowledge about the level of service subscribed to at a particular station. This architecture has the advantage of reducing the complexity in the subscriber's unit to keep its cost down.[3, 4]

The basic information unit that is sent in each upstream time slot is a 53 byte ATM cell augmented with 1 byte of management information, plus FEC and guardband bytes. The raw downstream channel bandwidth is 30 Mbps with a usable data payload of 23.9 Mbps. The downstream channel is shared by all the subscriber units for both user and management traffic. The raw upstream channel bandwidth is 2.56 Mbps with a usable data payload of 1.92 Mbps. However, multiple upstream channels can be employed on each cable, ultimately achieving a symmetric data rate if required. The standards call for supporting from 50-2000 stations on a single cable.

The head-end controls the upstream channel by issuing *grants* that specify the station(s) and type of messages that can be sent in each upstream slot. Grants are may be issued to groups or all stations to sign on with the head-end (*invite grants*) or to make requests for bandwidth for reactive traffic needs (*contention grants*). Grants are issued directly to individual stations (*direct grants*) to send data in a particular time slot on the upstream channel. Since multiple stations can send in any invite or contention slot, random access algorithms are used for these slots to resolve contention when it occurs. Data is only sent in the direct grant slots. Multiple grants (up to 15) can be packed into a single downstream cell. A block diagram of the request and grant flow is shown in figure 1. General issues for the head-end scheduler, without considering quality of service, are: 1) grants must reach a station on the downstream in sufficient time for it to respond on the upstream channel, 2) don't schedule too far in advance (though system delays must be accounted for), and 3) avoid using contention when possible.

FIGURE 1. Request and Grant Flow



A scheduling algorithm is applied to the flow of resource requests using information about the type of subscription service of each traffic flow. Grants are allocated to specific upstream cells based on this scheduler input. The head-end scheduler block in figure 1 implements the service model while the scheduler at the station schedules based on simple priority.

3.0 Considerations in Providing Tiers of Service

Today's internet users fall into several categories and have different expectations and requirements for their service level. The casual user needs a lower performance than the work-related user, and the work-at-home user is likely to be willing to pay a higher price for a higher level of service. In between these two types of users may be recreational users who are willing to pay some premium for increased service or small business or home businesses who may also wish to pay for some kind of increased service level. The marketplace will help to sort out what kinds of users would like what kinds of service and at what cost, but we must first be able to quantify what a user expects as "increased service" and be able to deliver it.

From a technical perspective, we've approached this problem in three ways: providing the mechanisms to deliver quality of service, developing an adequate traffic model to study these mechanisms in simulation, and getting a clearer picture of what a user expects a better level of service to provide. We're finding this to be something of an iterative process and continue to work on our understanding of the problem and its solutions. This paper is a report on our current status on this problem. Our studies have all been carried out with simulation models of UPSTREAMS.

We started with a QoS model with three categories of traffic: CBR, where each connection receives a constant bit rate delivered at minimum possible jitter; CIR, a *committed information rate* service with a minimum guaranteed rate, a maximum bound, and no guarantee on jitters[1]; and a best effort (BE) category which receives available bandwidth only. We take the CIR definition from frame relay as "The information transfer rate which the network is committed to transfer under normal conditions. The rate is averaged over a minimum increment of time." The BE traffic may or may not be bounded, both as a class and by individual STU. Traffic models for the first category include low bit-rate alarm information, telephony, and conventional videoconferencing. Traffic models for the second category include classes of p*remium data service* like work-at-home data traffic or Internet videoconferencing. Traffic models for the third category include electronic mail. Another important traffic model is web-surfing, but it is less clear to which category it belongs and we will discuss this further later in the paper.

The first obvious criterion for differentiating tiers of service is bandwidth. We concentrated on this criterion in our first work on scheduling for QoS and reported on it in [1] and present part of this work in section 6.0. We are considering borrowing from the IETF's integrated services working group to define premium service similarly to a controlled load service in that users should not see behavior appreciably different whether the system is load or not. In this case we should add another important criterion, the latency the user sees. It is expected that this is related to bandwidth, but the two are not completely correlated.

We made our initial explorations of bandwidth-differentiated QoS using "traffic blasting" sources. More recently, we have been exploring the proper way to evaluate our HFC architecture so that we can predict the performance a subscriber will see at a particular level of service and set bounds on the provisioning of users on a channel. It is an accepted fact that users will produce traffic and consume system bandwidth in bursts, but it is more difficult to find simulation traffic models of this true user behavior. We quickly learned that we can evaluate only the coarsest features of our architecture with simple traffic models. In the next section we discuss our work on traffic models.

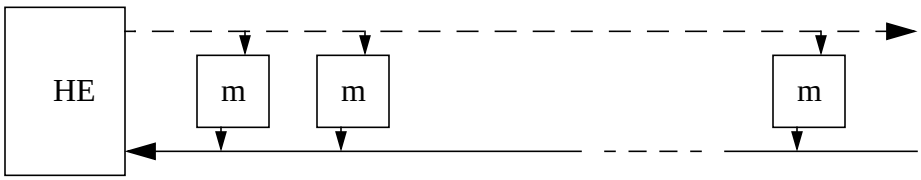
4.0 Appropriate Traffic Models for Cable Data Systems

Our simulation model is similar in its general architecture to the "Common Simulation Framework" model that Mil3, Inc. created for the 802.14 working group members who use that company's Opnet product [2]. Figure 2 shows a block diagram of our model. The simulation includes a model of Com21's UPSTREAMS protocol. The headend module (HE) handles scheduling of the upstream channel and control of the contention resolution algorithm. Each station (or cable modem, m) is modeled as a separate module and includes both a model of the cable modem and one or more application models that source traffic in the upstream direction. The upstream bus timing is modeled in detail, with time slots and propagation delays. The downstream bus is not modeled except as messages passed from the headend to the stations. The downstream messages *do* include the major components of downstream delay, most notably

1. CIR is similar to Available Bit Rate (ABR) when the Minimum Cell Rate (MCR) is non-zero.

FEC and interleaving delays. Variants of this basic model are being used by most of the participants in the 802.14 working group

FIGURE 2.Block diagram of the system simulated



How best shall we use simulation to determine performance of cable data systems? This question is particularly important given the unique architectural features of these systems. The 802.14 working group has used a number of one-way traffic source models to measure performance [8]. The “one-way” nature means that the sources will create traffic (packets, cells) according to some pattern or probability distribution and the output is queued at the cable modem until it can be sent. If the traffic level is too high for the available bandwidth, it will queue up and cause large backlogs in an infinite queue system (as in most models) or be discarded in a finite queue system. Large backlogs are problematic because they will show long packet or cell delays that primarily reflect these queuing delays and because many vendors’ systems (as Com21’s) use “piggyback” schemes that will completely remove these backlogged stations from the contention process and thus mitigate the critical loading on the contention resolution mechanism. Most damning of all is that there are very few real applications that function this way. Nearly all of the applications that are expected to be used in the home are based on the TCP/IP transport layer protocol and they will not have more that one window of data outstanding (a maximum of 16-64 Kbytes, depending on the implementation). In the real world of computer networking, feedback is a major part of the system dynamics. The true interaction of the application, transport protocol, and MAC layer protocol in the presence of bursts and congestion is a complex one. Can one-way models with no feedback give a clear picture of the kinds of traffic loads our systems will experience?

A careful use of backpressure from the modem queues to the traffic sources might be used to reduce the sending of data that would not be sent in a real system. The problem of a proper source traffic model remains. Exponentially distributed packet arrivals (the popular poisson process) do not well represent most traffic processes (e.g., [9]). Their usefulness as source models for evaluating a system can also be questioned. We will explore some traffic sources to evaluate cable data systems.

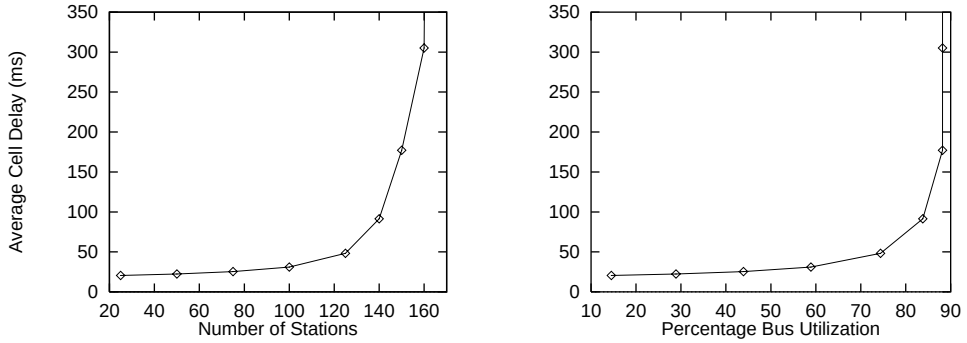
Exponentially distributed source models. First, consider a simplified version of the “scenario A” model suggested in [8]. In our model, the number of packet sizes selected is reduced, but the distribution is similar. UPSTREAMS is ATM cell-based, so source models are programmed to produce cells directly at each packet-time; 60% of the packets are 2 cells long, 30% are 11 cells, and10% are 32 cells. The average packet interarrival time can be varied to create different average bit rates per source. About 12% of the channel will be dedicated to contention opportunities, so only about 1.7 Mbps is available for data payload. If sources are configured at 10 Kbps, the system will be saturated around 150-160 stations. If the average source rate is reduced to accommodate a larger number of simultaneously active stations, the interarrival time becomes stretched out in time and the likelihood of stations competing for the upstream bandwidth at the same time becomes very small. We start by examining results for the 10Kbps sources.

Results from simulations using this traffic source model are shown in figure 3. There is a smooth curve showing that architectural features dominate the delay at lighter loads, first the delay is due to the round trip time to get an allocation of upstream bandwidth from the headend, then some extra delays due to contention resolution come in. At the larger numbers of stations, the system becomes limited by the upstream bandwidth bottleneck and delays grow toward infinity as the queue sizes. The general shape of this result can be predicted by simple analysis, so it is pleasing to use a traffic model whose behavior we can predict. A more important question is what insights into the architecture have we gained from running these simulations? We can measure cell delay[1], bus utilization[2], average queue sizes, and some other statistics, but we cannot decide what these values mean to a typical application.In fact, mea-

1. It is also possible to reassemble cells into packets and measure packet delay, but they will follow a very similar curve.
2. The maximum bus utilization is 88% since at least 12% of the bus is dedicated to contention opportunities.

surement studies have shown that the *average* bandwidth per user is quite low and that most applications have some sort of bursty behavior. In this case, we can support more simultaneous users, but we might see very different behavior than that of exponentially distributed sources.

FIGURE 3. Results for exponentially distributed sources



A large segment of the user population for cable data systems is expected to be spending its on-line time accessing world wide web pages. A model of this application would be helpful in evaluating architectural features for our products. We would expect traffic sources in such a model to look quite different from the exponentially distributed traffic sources and this may lead to different architectural insights. There is a growing body of research work on the measurement and characterization of world wide web traffic.

Deng's WWW traffic source model. Building on the WWW measurement and characterization work done by Cunha et. al. in [10], Deng carried out further measurements of what is expected to be the most popular applications used from home and created a one-way model of web surfing [11] from the client side. Deng created an ON-OFF type model which can be outlined as:

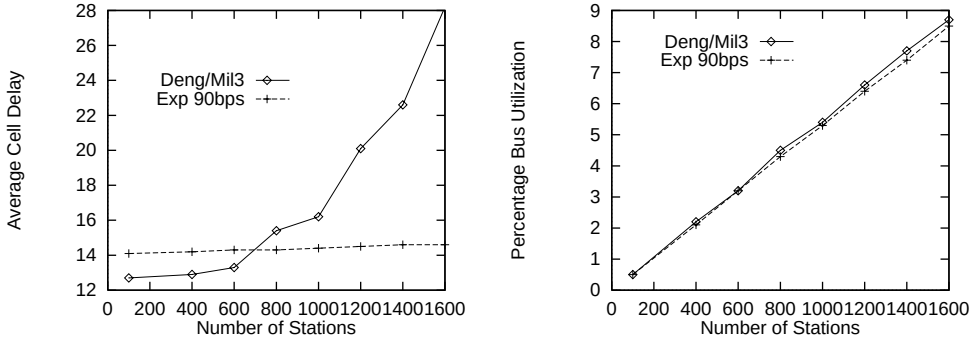
1. Assume starting in the ON (browsing) state. The state changes to OFF at a delay (in seconds) taken from a weibull distribution with parameters ($e^{4.5}$, 0.88).
2. A URL request packet of size 250 bytes is produced after a delay picked from a weibull distribution with parameters ($e^{1.5}$, 0.5).
3. Repeat step 2 until the state changes from ON to OFF.
4. Return to the ON state after a delay taken from a pareto distribution with parameters (60, 0.5).

The advantage of this traffic model is that is allows us to model systems of more than one thousand web-surfing sources, since it is bursty. That is, individual sources may be inactive for long periods of time, then send their message. Simulations show that the average source rate is about 90bps. Use of these distributions ensures that at least 60 seconds are spent in the OFF state, and at least 90 seconds are spent in the ON state. How large are the bursts during the ON state? Each message is fixed at 250 bytes (6 ATM cells for our MAC) in the Opnet code. The minimum time between these messages is about 4.5 seconds, so the maximum source rate during the ON period is about 450bps, but this would correspond to one message interarrival time being picked at 4.5 seconds. All 6 cells arrive at the cable modem's queue at the same time. If they are sent in the minimum possible time to make a contention request to the headend and the cells are sent consecutively at the upstream bus rate of 200 microseconds per cell, the data can burst at a rate of about 200 Kbps. This clearly produces a bursty model, though the burst sizes are not large. The burst rate could be increased by choosing larger or random message sizes, but two important questions to consider are 1) how well does this model the dynamics of WWW browsing and 2) does this model produce additional information about the system being simulated that cannot be learned using the exponential sources. Note that it is not possible to use this model to make any measurements of what kind of performance the user would see with respect to the application.

To examine the value of Deng's traffic source as a simulation model, note that we can repeat the experiment using exponential sources, adjusting their average interarrival time to give an average source data rate of 90bps. This allows us to compare the bursty, one-way WWW model with the exponentially distributed source traffic. In figure 4, the average cell delays for the two traffic models are plotted. Although they appear different across this range, note that

these values occur within a very limited range, between 12 and 28 milliseconds. The channel utilization results are nearly identical. In fact, there is very little difference in the effect of the two traffic source models on the system under study. The differences in average cell delays might come from the slightly larger contention waits for the less smoothly distributed Deng model. Each message of 6 cells will have to wait at least one round trip time for the headed to send a grant and may need to contend for the upstream channel.In lightly loaded systems, the delay is dominated by the round trip time and the scheduling delay; in more heavily loaded systems it is dominated by the contention wait. Neither of these two models produced significant collisions on the upstream channel.

FIGURE 4. Results for Deng/Mil3 web browsing sources



This one-way WWW model does not offer any additional insights about system behavior. Although a model like Deng's may be useful for creating analytical models, in this age of fast CPUs, it seems like an unnecessary simplification of the actual web browsing process for a simulated traffic source.

A more detailed browser model for simulation. Neither of these models gives us a sufficient feel for what is actually going on in the system nor does it represent a real application well. Since real applications enter into a transport protocol level conversation with a remote server, they will not continue to fill a cable modem's queue beyond the amount proscribed by the transport protocol. With a reasonably fast desktop machine and a reasonably fast simulator, it seems that more of the web browsing "conversation" might be simulated, leading to more insights on system behavior and architectural decisions and the ability to measure some quantities that pertain to application-level performance.

A browser traffic source model should start with the typical process initiated when a user makes a request for a particular URL[1]:

1. User request for a URL results in opening of a TCP connection: a 40 byte SYN packet is sent.

2. The SYN is received by the remote server, it returns a SYN acking the connection after some service time delay. Upon its receipt, the URL, typically 100-200 bytes, is sent to the remote server.

3. After the service time delay, the remote server returns the requested page data including other URLs referenced on that page. We assume that the typical case is for this information to fit into two packets thus causing a 40 byte packet to be sent which is both ACK for the two packets and FIN for this initial TCP connection. When browsing in a graphics off mode, the page would be displayed and done at this time. Otherwise, at this same time, SYNs are sent to open parallel connections for all the URL references found in the page. A typical number for such references is four.

4. After the service time delay the remote server(s) return SYNs acking the connections and the URLs, as above 100-200 bytes, are sent from the requesting client.

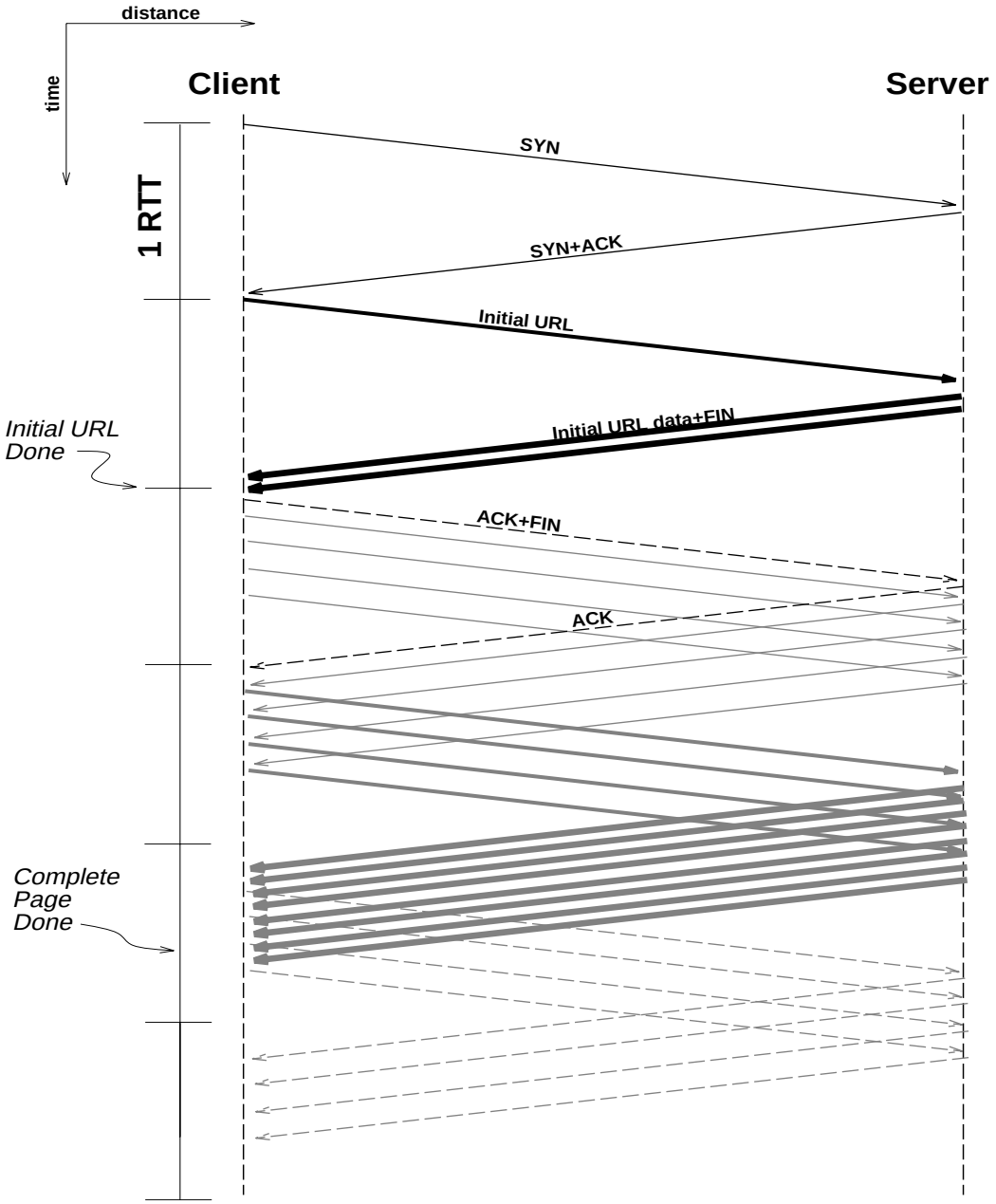
5. After the service time delay, the remote server(s) returns the documents requested and FINs for the connections and the requesting client ACKs the data and the FINs. The number and spacing of ACKs depends on the size of the documents and, for larger documents, on the particular implementation of TCP. When the FINs have all been received by the client, it will be able to display the total page.

1. The author is indebted to Van Jacobson for suggesting most of this model.

6. The user (or controlling process) delays some “think time” before making the next request, at which time, return to step 1.

A diagram of this message exchange is shown in figure 5.

FIGURE 5. Diagram of web browsing message exchange



For a web-browsing user, the performance measures of interest concern how long it takes to “see something”. These have been split into two measures, first, the time from when a user requests a particular new URL until that page has been completely fetched (and is thus available for display). This is referred to here as the “initial URL’s data” time and happens at step 3, above. The second measure is when all the data associated with all the URLs in the page have been received and referred to as “all data” and happens at step 5 above.

We now have a model and some figures of merit that have meaning at the application level. To fully implement these as a simulation model requires having a fully functional two-way model of our system. It also requires getting representative values for all of the web browsing parameters and implementing a model of the transport layer protocol and its messages in the simulation. All of these are important end goals and are planned for the future, but are not done at this time.

In order to get initial results without having to build a complete simulation model with remote servers and a downstream path, an intermediate approach was taken. The real process described above was used as the basis for the browsing source simulation model. At each stage of the process outlined above, the browser source module sends a message to the cable modem module then *waits until the modem signals that its queue is empty* before it adds the server delay and goes on to the next step. The wait time for the remote server is a parameter and is set at a constant value to which is added a value uniformly distributed within +/-1ms. The model is instrumented to collect the "first page time" of step 3 and the "total page time" of step 5. The upstream transmission delay of the last cell is not accounted for, nor is the downstream transmission delay, but these are not the quantities of interest.

Although many of the parameters would typically vary, several were fixed across simulation runs for some initial experience with this type of model. First, all URL sizes were fixed at the same value. The results in this document used URL sizes of 3 cells (97-144 bytes)[1], the "first page" information was assumed to fit in two packets which could be ACK'd with a single 40 byte packet, four URLs embedded per page, and each of these four document sizes was selected from the pareto distribution given in [10].

Although part of the transport protocol behavior is modeled as outlined above, the simplifying assumption was made to send all the ACKs for the four documents fetched in the second stage at the same time. The number of ACKs was computed as the document size divided by 1460 bytes per packet and that result divided by 2 packets per ACK. This clearly ignores the transport protocol interactions and probably causes the time for "all data" to arrive at the application to be underestimated by at least one round trip time (or "server time") due to ignoring slow start. A more complete model of the interaction of the browsing client with the transport protocol and the downstream traffic is crucial to determining the total throughput and latencies the user sees, but we defer the inclusion of these features until a complete two-way path is implemented in the simulation model.

A more difficult problem is picking the average "think time" per user; that is, the average amount of time a user waits before selecting another page. Although [11] has some measurements of this value, it is unclear how well these apply here since the authors were trying for a more simple on-off model. It should be possible to extract this information from the traces located at [12]. We have recently learned that researchers at Georgia Tech are pursuing a model similar to ours and plan to do some measurements [13]. If this work is made publicly available, these values may be more applicable to our model. In the meantime, how should one best select this time? Real think times tend to be fairly long and vary widely. This can be problematic for simulation. So, this initial model was set to have random think times uniformly distributed between one and 30 seconds, a very conservative (pessimistic) assumption.

In figure 6, we show the results of two sets of browser runs. One uses a server time of 20 milliseconds and the other a server time of 60 milliseconds. A value of 20 milliseconds is picked to correspond to the time to access data located or cached at the headend. The adjusted average response time is just the response time with the constant server times subtracted (two times the server time for the initial URL, four times the server time for all data). Using this time makes it easier to compare the relative performance of the different server times or locations and isolates the part of the delay contained within our system. To get an idea of the delays the user would see, add the server times to the adjusted values.

Figure 7 shows that the server time makes no difference until the largest numbers of stations and, even then, the difference is quite small. This is due to the dominance of the think time over the data transfer time. The 20 ms server is the more conservative assumption on traffic load, so that is used as our baseline. A set of simulation runs was performed with the server time of 20 ms and all URL sizes set to take 5 cells (193-240 bytes) and compared to the baseline simulations in figure 7. Here there is a real difference between the two scenarios, the larger URL resulting in a

1. One set of run was done with URL sizes of 5 cells (193-240 bytes) to show the differences.

greater bandwidth use and moving the curves to the left. This also shows that URL size is an important part of the burst behavior in this model and a bit of examination shows why.

FIGURE 6. Results from more detailed web-browsing model.

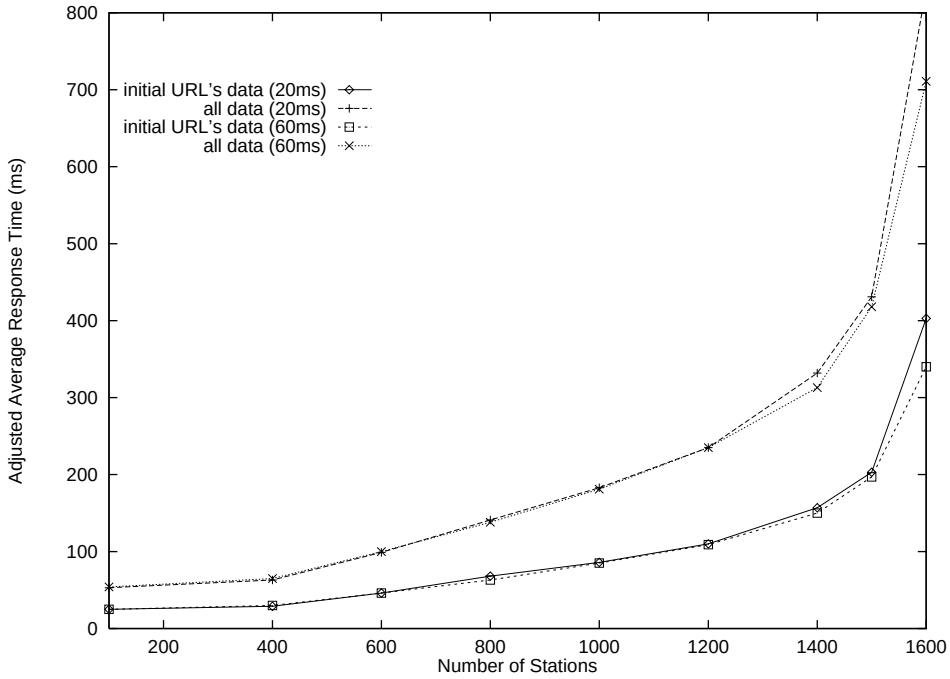
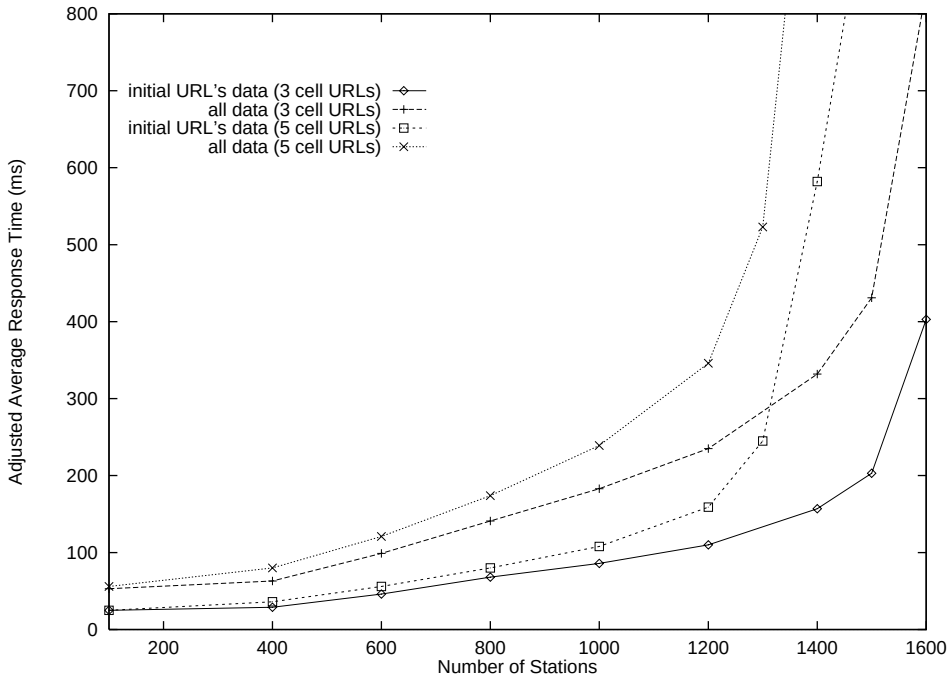


FIGURE 7. Results with larger URL size (5 cells vs. 3 cells)



There are three times that the browser source potentially sends a larger number of cells: when the four SYNs and the previous FIN are sent, when the four URLs are sent, and when the ACKs are sent for the four documents obtained from those URLs. In the first case, a source always sends 10 cells. In the second case, the source sends 12 cells for the 3-cell URL scenario and 20 for the 5-cell scenario. The minimum document size is 128 bytes, so the third case could easily be only 8 cells. Although the third case may sometimes exceed 20 cells, it appears that the second case, the one

dependent on the number of URLs, dominates. Of course, real URLs are small, but vary. This behavior could easily be simulated, but it appears the shape of the resulting curves would lie somewhere near the curves of figure 7.

The average source rate for a browser source is about 750bps, five times that of Deng's model. This is due in part to the short think time, in part to the burstier behavior than Deng's model (further discussed below). Altering the think time (or off time) distribution makes it possible to increase the traffic levels while preserving the application behavior, making it easier to exercise the architecture.[1] With the backpressure indication from the cable modem's data queue to the traffic source, we can also keep track of the rate that each burst of data is being sent at. The average data bursts are between 120-130 Kbps for the simulations with smaller numbers of stations and can decrease to 20 Kbps for the saturated runs. This information can help in understanding the dynamics of the system and can be plotted against the response times to help determine what level of burst bandwidth a station will need for adequate performance.

The results of simulations with the browser model showed the "good news" of being able to support a large number of simultaneous users without significant delays being introduced into the time to present a web page. Using this traffic model, it can be seen that the curves start up due to the effects of contention and there are no false benefits of excessive piggybacking when queues are long. This makes it easier to test architectural features.

Comparing the models. All the models used so far in this report are compared in figure 8 by plotting cell delay vs. number of stations. Two additional data sets were added. One uses exponentially distributed sources with 750bps rates to match the browser rates, the other uses Deng's model with no off time to get the maximum possible source rates. It is not possible to get average source rates higher than 265bps from Deng's model without further altering the model dynamics. The delays for the Deng model with no off state are slightly lower than the delays with the off state, but this could have to do with the larger number of simulation points (all simulations were run for 600 seconds). This shows that the Deng model does not have sufficient burstiness to stress the modeled system.

FIGURE 8. Average cell delays for all sets of results

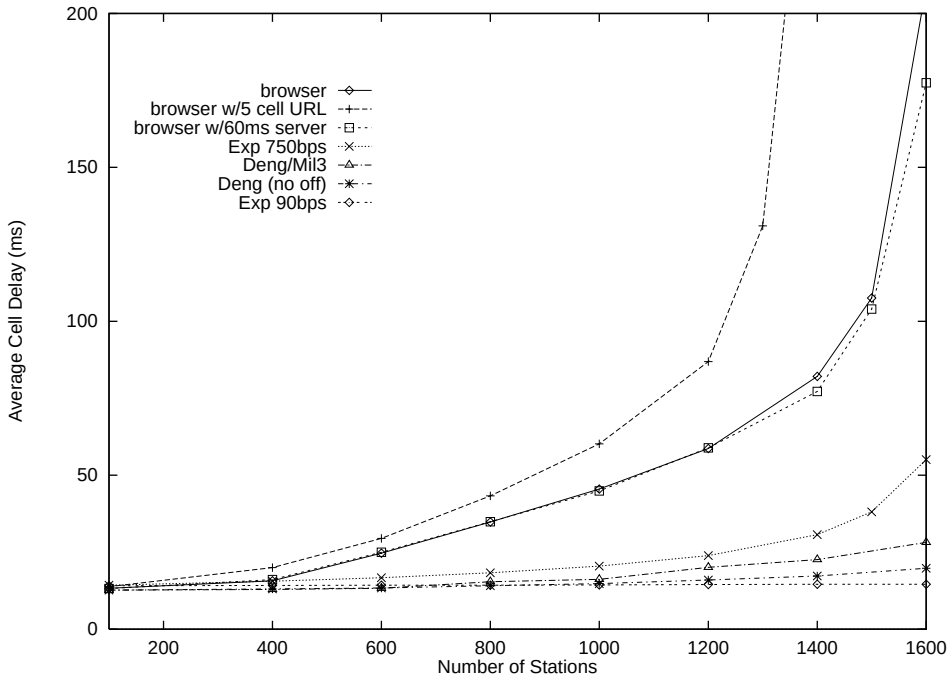


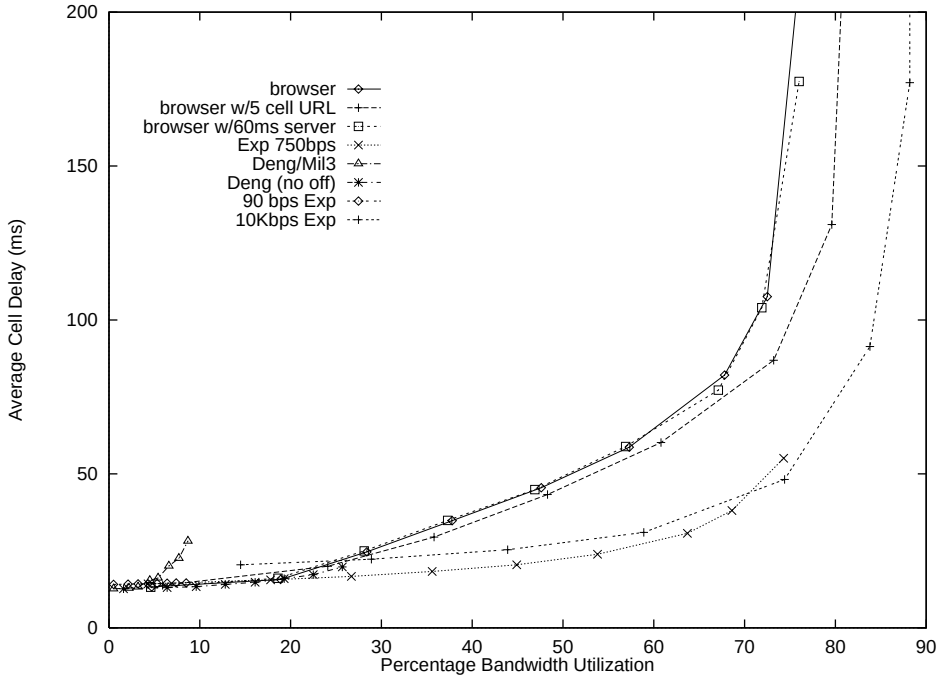
Figure 8 shows that the one-way models produce manifestly uninteresting behavior for architectural evaluation. The Deng/Mil3 model has a very low average cell delay across all runs, nearly identical to the exponential model. This kind of "gentle" behavior gives good simulation numbers, but does not provide a traffic model sufficient to exercise the system's contention mechanism or to stress scheduling algorithms. The browser model represents the actual

1. Automatic browsing programs may, in fact, exhibit short think times.

browsing client-server “conversation” with more fidelity and produces more interesting behavior. When we want to stress a system by producing high volumes of traffic, we would be better served by constant bit rate sources transmitting at a data rate sufficient for our purposes (“bandwidth blasters”).

All models are compared by plotting average cell delay vs. the bus utilization in figure 9. In figure 9, it is possible to see the characteristic curve of the exponential sources does not change much even when the source rates vary. Note the similarity in the 10Kbps and the 750bps curves. The curve for the Deng/Mil3 source never has a high bandwidth utilization, but the delay shows that there are more contentions at the lower bandwidth utilization due to the number of sources involved compared to the other models’ number of sources at that utilization. The Deng/Mil3 curve with no off state lies closer to the other models since the utilization is increased, but the highest bandwidth utilization is 26%. The browser model with 5-cell URLs gets more bandwidth utilization at the higher cell delays because the average message size is longer. The results shown here reflect some architectural tuning that was done after the first uses of the browser model. This tuning made it possible to push the curves to the right. The other models do not sufficiently stress the system to permit exploration of options.

FIGURE 9. Average cell delay vs. bus utilization



Summary and current status of our simulation traffic models. Using our browser model, we were able to learn much more about the potential behavior of a home network. We were able to more appropriately assess the importance of rapidly resolving contention and developing algorithms to reduce contention. Our investigations convinced us that we must model our system with all the salient features of the application and transport-layer conversations if we are to design architectures that will function well when deployed in the real world. We are quite aware that this traffic model is not yet fully adequate in this respect, but we are pleased with the insights its use has provided.

5.0 Preliminary work on Quality of Service scheduling

In this section we overview the initial work we did on scheduling for QoS and reported on in [1]. At that time, we had only applied bandwidth allocation in the upstream direction, but we are also adapting the algorithm to bandwidth allocation in the downstream direction. We are interested in discovering the effects of using a controlled bandwidth access link in both directions on a two-way browsing model.

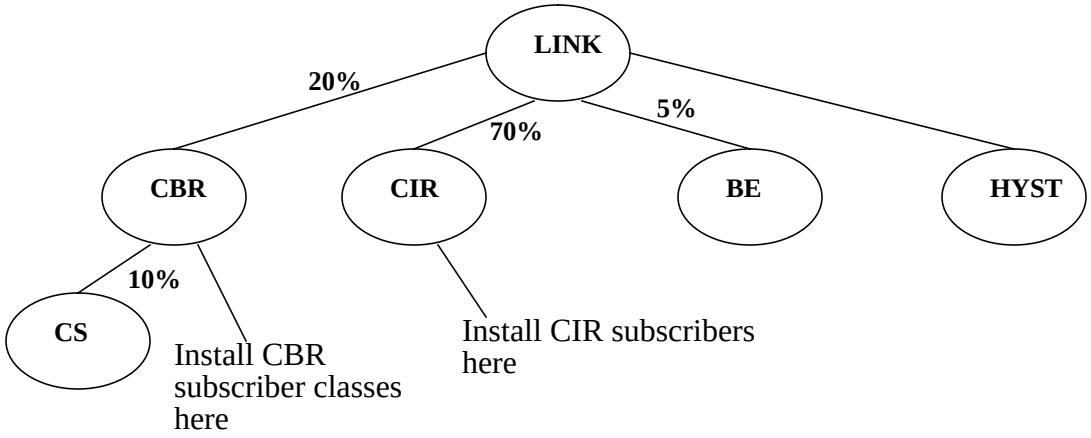
One of our goals in a scheduler for grant allocation is one that can include our three basic tiers of service, CBR, CIR, and BE, in an integrated fashion. Our scheduler is based on the class-based queueing (CBQ) algorithm developed by Floyd and Jacobson [14]. The concepts of classes, a class hierarchy, and priorities can be mapped to our tiers of ser-

vice. Still, CBQ was meant for packet-level scheduling and we needed to schedule grants based on requests for allocations to send cells, undelineated by packet boundaries. Tuning is required to select the correct number of requests to schedule at a time and our scheduler works in an environment where it is necessary to schedule well in advance (due to round-trip delays) and to schedule a small block of upstream cells at one time. To keep that block size, and hence scheduler reaction time, small the scheduler must be time-efficient, but must also be extensible, as the number of active STUs[1] on an cable can vary widely. CBQ was selected since it appears it will work under these circumstances.

The grant scheduler works on ATM cells or *shreds* of packets rather than full packets as in CBQ, thus our algorithm is *shredded* CBQ, or SCBQ[2]. This scheduler has all the basic features we wanted, integrating all our requirements into a single scheduling mechanism, and has good sharing performance, as will be seen in the results. However, several aspects will require further tuning.

SCBQ classes and parameters. Following [14], we can draw a diagram of our class hierarchy, which is headed by the link (see figure 10). Under the link there are four classes, CBR, CIR, BE, and HYST. CBR is priority 1, CIR and BE have the same priority, and HYST is the lowest priority possible. The CBR class includes both an administrative class that sends contention grants every ten slots (CS) and subscriber CBR, and is allocated a total of 20% of the bandwidth. The BE class has an aggregate allocation of 5% of the link, but may "overdraft" the unused link bandwidth. The subscriber CIR classes are installed under CIR and set up with both a guaranteed allocation and a maximum allocation. CIR itself cannot "overdraft" the link. In the experiments, constant bit rate source ("bandwidth blasters") were used to drive all the STUs. In most cases, we were driving at higher data rates in order to isolate scheduler effects from contention effects.

FIGURE 10. Class hierarchy



In all the experiments, the CS class takes up 10% of link, reducing the available payload or data bandwidth to 1.73 Mbps. Each upstream ATM cell is sent in a time slot of duration 200 microseconds, giving a data payload of 1.92 Mbps. All simulations were run for 30 seconds of simulated time. The plots shown here are of the average instantaneous utilization.

Simulation results. In the first experiment 10 STUs are configured at *modem-quality* CIR and are overdriving their allocation. The service level is a minimum guaranteed rate of 28Kbps and a maximum of 200 Kbps each. Each source comes on at one second intervals into the simulation, sources traffic at a bit-rate of 384 Kbps and turns off at one second intervals before the end of the simulation. With 1.73 Mbps available for sharing, each should get 173 Kbps, which is under the individual maximum limit. Results in figure 11 show that, indeed there is equal sharing of the channel. In figure 12 a close up of the STU utilizations (without the aggregate) shows that the STUs get more of the channel bandwidth when there are fewer than 10 STUs, but that each is limited at 200 Kbps. Although we configured

1. We use HE to denote head-end and STU to denote the station, or subscriber terminal unit.

2. M. Laubach would like to thank Steve Deering for inspiring the name for our variant of CBQ.

our CBR service in an *at most* rate delivery mode, the average contention slot spacing was 2.0 milliseconds, as desired, with a very small variance. As the source bandwidth exceeds the allocation, the uncontrolled sources fill the infinite STU queues; note the effects of the backlogged STUs emptying their buffers.

FIGURE 11. Ten controlled sources sharing the upstream channel with their aggregate

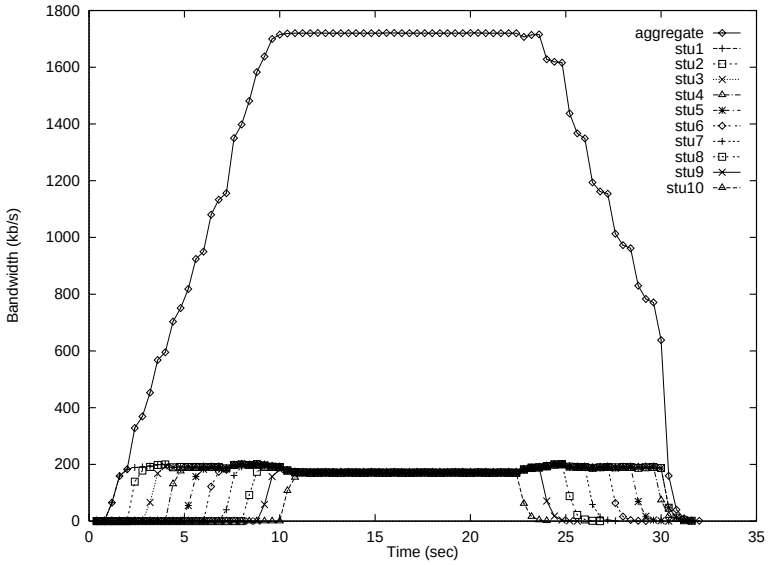
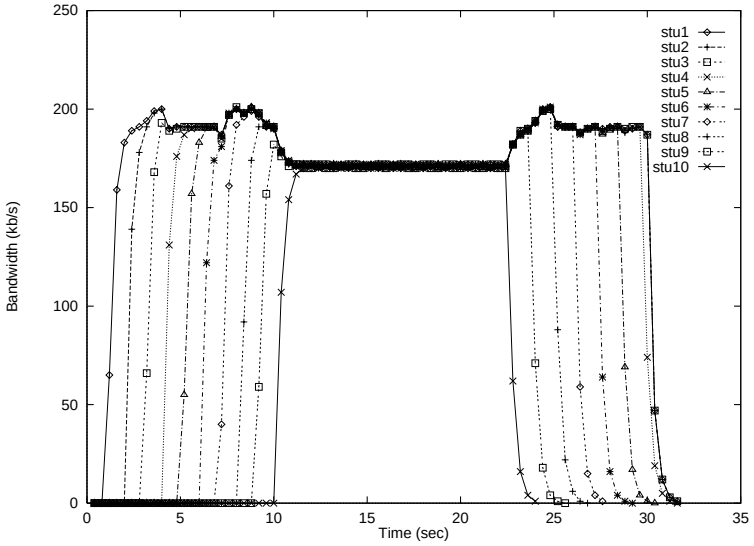


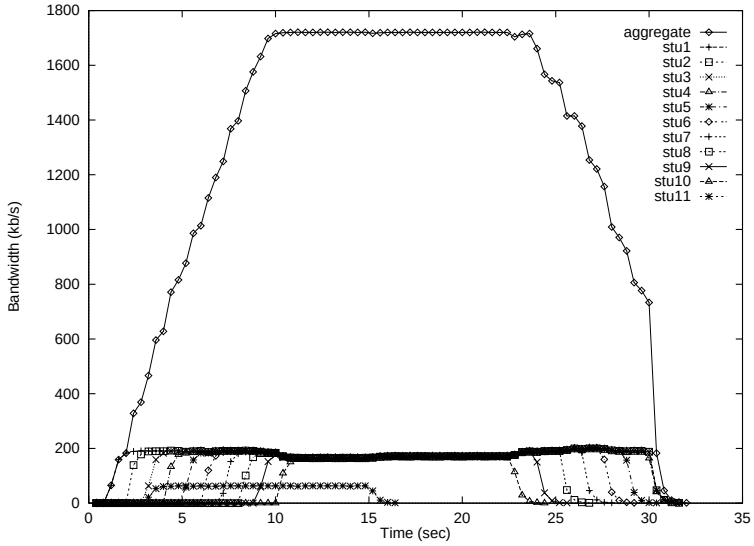
FIGURE 12. STU-only plot of the same experiment as in figure 11.



In the next experiment (figure 13), a constant bit rate subscriber stream of 64 Kbps was added, provisioned as a single cell every 6 milliseconds. When the CBR stream comes on at 3 seconds into the simulation it reduces the overdraft bandwidth available to the 10 CIR STUs.

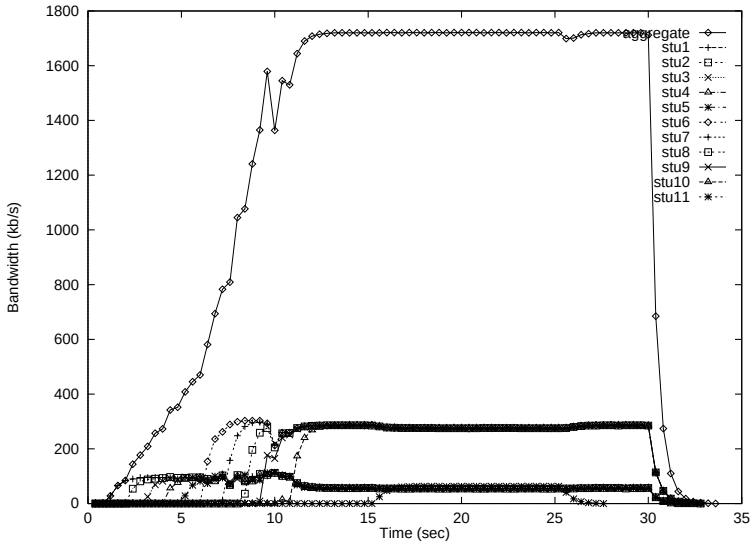
What happens when best effort service is part of the mix? Recall, that this is a *true* best effort service where no minimum (other than an allotment of 1% of the link bandwidth for the entire class) rate is guaranteed per subscription flow, but no maximum is enforced either. Any spare bandwidth can be used by waiting traffic of this class. The STUs subscribed at BE have sources sending cells at 96 Kbps and the CIR serviced STUs have sources sending at 384 Kbps. The CIR service here is lower bounded at 128 Kbps and upper bounded at 300 Kbps. Each BE STU's source comes on at one second intervals from the simulation start, then the CIR serviced sources turn on. A CBR source sends data between 15 to 25 seconds into the simulation. Figure 14 shows the individual STUs with the aggregate

FIGURE 13. Example of figure 12 with a CBR (64 Kbps) source added



bandwidth value, ramping up as additional sources start to send. In Figure 15 the same experiment is shown with just the STU utilizations plotted to show clearly that the BE service is reduced to accommodate the CBR service.

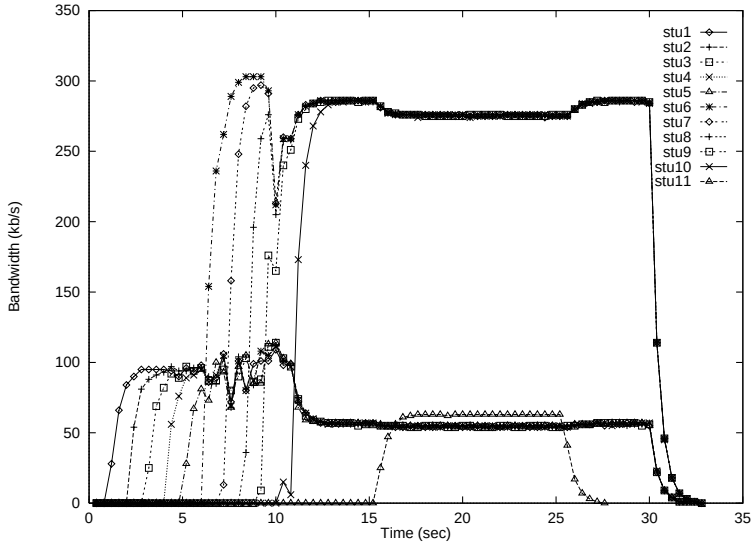
FIGURE 14. Five BE STUs, five CIR STUs, and a CBR STU



We examine the sharing behavior in figures 14 and 15. Before the CBR source is turned on, there is 1.73 Mbps that has 657 Kbps of allocated bandwidth (the 128 Kbps of each CIR service plus the 5% of the link allocated to best effort) being used. This leaves the rest to be shared equally between the six classes. Thus, each CIR STU should get 295 Kbps and the BE class gets 253 Kbps, to share equally among its 5 members, at about 51 Kbps. In the plots, this appears to be the case. When the CBR source comes on, one-sixth of its bandwidth is taken from each CIR STU and one-sixth is taken from the BE class, spread equally among the 5 members. Although the plot is not at this level of resolution, the results appear to agree.

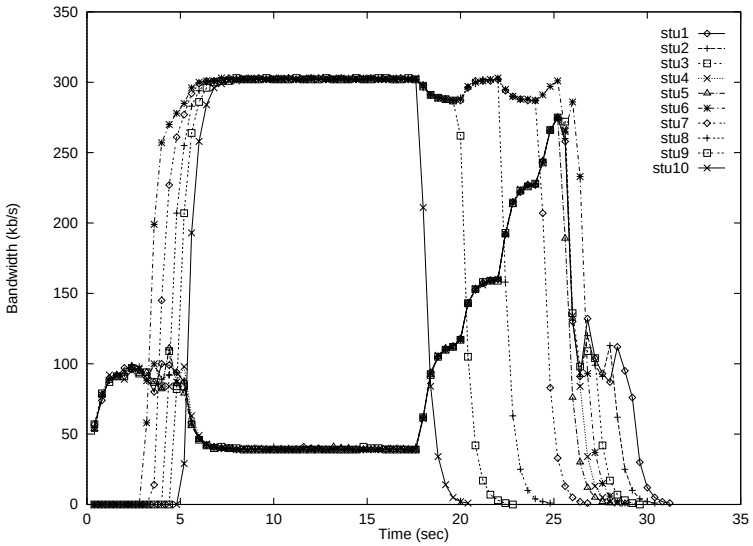
Next, we simulated 5 BE STUs and 5 CIR STUs configured as in the previous example, but with CIR given a preferential weighting in the round robin allocation of excess bandwidth such that the BE class gets 2/3 the excess allocation of a CIR class. If the excess bandwidth were shared equally, CIR would have 295 Kbps and BE 51 Kbps. With weighting, each CIR class could get an additional 177 Kbps and the BE class only 119 Kbps. Since 177 Kbps would put the CIR classes over their 300 Kbps upper bound, that leaves an additional 25 Kbps for the BE class to draw on.

FIGURE 15. As in figure14 without aggregate traffic



This means the 5 BE classes share (86+119+25) Kbps to get 46 Kbps each, as in figures 16. The high usage causes the aggregate (not shown here) to stay flat at 1.73 Mbps between 6 seconds and 25 seconds, then sources start to turn off.

FIGURE 16. Five CIR, five BE, with a preferential weight given to overdraft in CIR vs. BE



6.0 Conclusions and Future Work

Our investigations convinced us that we must model our system with all the salient features of the application and transport-layer conversations if we are to design architectures that will function well when deployed in the real world. We found that the commonly used one-way exponentially distributed traffic source models were not useful for stressing bandwidth level QoS differentiation, for exploring the dynamics of the contention resolution algorithm or creating burst traffic effects. Since desktop computers are fast enough to simulate detailed system dynamics, we can't think of any justification to use a one-way model, except as a modular step toward the final full-feedback design.

Our next step is to complete work on the simulation model of a complete two-way path and to add a transport protocol model and an ftp source type. The latter should be possible by modifying parts of the models distributed with **ns**.

We plan to ascertain whether our browser results hold with a more complete model, then to continue work on scheduling for QoS and on provisioning of channels.

Our future SCBQ work will be in three categories: further algorithm refinement, further testing of SCBQ, and turning the simulation code into efficient runtime code. Further modifications to the allocation algorithm are needed to reduce CBR jitter and to allow properly sized bursts of cells from stations that are just becoming active. We also need to apply the SCBQ scheduler to the downstream traffic. In addition, we need to do further work on our hysteresis class once we have installed appropriate two-way traffic models. We also want to make the modifications to deliver multiple QoS streams to a single STU. Finally, our SCBQ simulation code has departed significantly from the original LBNL prototype code [5] and needs to be reworked for runtime efficiency.

Acknowledgments

K. Nichols would like to thank Van Jacobson for discussions clarifying CBQ, for assistance with the web-browsing model, and discussions of results.

References

[1] K.M. Nichols and M. Laubach, *On Quality of Service in an ATM-based HFC Architecture*, IEEE ATM '96 Workshop, San Francisco, CA, August 25-27, 1996

[2] See http://walkingdog.com for further information about IEEE 802.14.

[3] M. Laubach, *To Foster Residential Area BroadBand Internet Technology*, Connexions, February 1996 Vol 10, no. 2, p18-30.

[4] D. Sala and J.O. Limb, *A Protocol for Efficient Transfer of Data over Fiber/Cable Systems*, Proceedings of Infocom 1996, p 904.

[5] *LBNL's CBQ code*, available at http://www-nrg.ee.lbl.gov/floyd/cbq.html

[6] M. Laubach, *The UPSTREAMS Protocol for HFC Networks*, proposed to IEEE 802.14 Working Group, contribution number IEEE802.14-95/152R1, January, 1996.

[7] Available at `http://www-nrg.ee.lbl.gov/ns/`

[8] J. Limb, et. al., "Performance Evaluation Process for MAC Protocols", IEEE 802.14 working group document 96/83R2

[9] V. Paxson and S. Floyd, "Wide Area Traffic: The Failure of Poisson Modeling", IEEE Transactions on Networking, June 1995, pp. 226-244.

[10] C.R. Cunha, A. Bestavros, M.E. Crovella, "Characteristics of WWW Client-based Traces", Boston University Computer Science Technical Report BU-CS-95-010, July 18, 1995.

[11] S. Deng, "Empirical Model of WWW Document Arrivals at Access Link", *to be presented at* IEEE ICC'96.

[12] Traces at `ftp://cs-ftp.bu.edu/techreports/95-010-web-client-traces.tar.gz`

[13] S.U. Khaunte, J.O. Limb, "An Empirical Model for the World Wide Web Client Traffic at the Upstream Channel Access Point", Georgia Tech Computer Science Department draft report

[14] S. Floyd and V. Jacobson, *Link-sharing and Resource Management Models for Packet Networks*, IEEE/ACM Transactions on Networking, Vol. 3 No. 4, pp. 365-386, August 1995.