

Improving Network Simulation with Feedback

Kathleen M. Nichols
Bay Networks
Santa Clara, CA 95052-8185
knichols@baynetworks.com

Abstract

Since the publication of [1], it has become well known that real network traffic does not follow a poisson process and that feedback mechanisms in a network affect the traffic patterns. Yet much modeling and simulation work employs exponential interarrivals in open-loop traffic sources. Historically, these sources were chosen because they are amenable to analysis and easy to simulate, but they have serious shortcomings. Even when the source distribution is changed to reflect the burstiness of real traffic, we found that open-loop models fail to reproduce the detailed time structure and the crucial effects of feedback on traffic sources.

In this paper, evaluation of a Hybrid Fiber Coax (HFC) architecture is used to illustrate the inadequacy of traditional open-loop modeling. In addition, a promising intermediate technique of using "local feedback" is shown. Finally, using feedback models, the interaction of the HFC architecture with TCP dynamics was studied using web-browsing and file transfer applications. These studies led to a novel architectural improvement that would have been missed in an open-loop model.

1.0 Introduction

For many years network architectures have been studied in simulation by driving a model of the network with traffic sources where the message sizes and interarrival times are drawn from a random distribution, usually an exponential model. Since the publication of Leland et. al. [1], researchers have consistently found that real, measured data traffic does not follow a poisson distribution and some papers have additionally suggested probability

distributions that capture the burstiness of real traffic [3,9,10] to replace the poisson arrivals. Despite the mounting evidence that poisson models are overly optimistic, significant amounts of network modeling and evaluation continue to be performed using such sources. Indeed, these models are widely used in industry and shipped with most commercial simulators. Our work adds to the body of evidence against poisson traffic sources and we have identified a further problem with commonly used simulation traffic models, their open-loop character. By *open-loop*, we mean the use of traffic sources which do not alter their behavior due to feedback from the network. These may be driven by a random or deterministic process or from a traffic trace or other measurement data. Traffic sources designed to respond to network feedback cannot be said to obey a particular apriori probability distribution since packet output will depend on information received in feedback. For example, in a TCP conversation the acknowledgements from a receiver are affected by network conditions experienced and in turn affect the sender's rate. When feedback is used, it is useful to think of *traffic* models rather than *source* models.

We make the case by focusing on the evaluation of Hybrid Fiber Coax (HFC) architectures. The performance evaluation criteria [2] submitted to the IEEE 802.14 Cable TV Media Access Control Working Group [8], one of the standards bodies working on broadband data delivery specifies open-loop traffic sources driven by exponential interarrivals. Another open-loop model, based on measured traffic and using a heavy-tailed interarrival distribution [3], was also used by the IEEE 802.14WG. We applied both these models and found them inadequate to evaluate crucial architectural features. To more appropriately evaluate HFC architectures, we began to use traffic models that respond to network feedback and found that

open-loop and feedback traffic sources have very different network behavior characteristics. Our results show the failure of the open-loop models to accurately capture system dynamics. We identify two major problems with the use of traditional models: inaccurate conclusions about the system architecture and the lack of user-level performance metrics. To size system installations, we need a clear idea of how load variation affects the performance those users perceive. Although the shortcomings of open-loop traffic sources may be exacerbated due to the special characteristics of HFC architectures, clear general differences emerged.

Section 2 describes the HFC architecture we are simulating, section 3 covers the application of traditional open-loop source models to this architecture, and section 4 presents simulation results with feedback source models, comparing results for all simulations. In section 5 we show results obtained using closed-loop models of web-browsing and file transfer. We evaluate the performance effects of some TCP options and a novel architectural feature for HFC architectures. Section 6 summarizes.

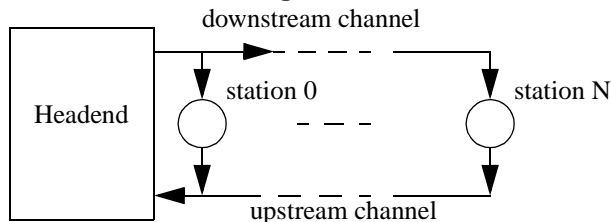
2.0 The Simulated HFC Architecture

Hybrid Fiber-Coaxial cable TV networks are beginning to be utilized as a data delivery service to provide subscribers with Internet access. It is important to understand the behavior of these networks under expected data traffic loads in order to make appropriate architectural decisions and to provision installed systems. Background on the challenges of the HFC environment and of the on-going standards work in this area can be found elsewhere [4,5,8]; only salient architectural features are given here. The results presented in this paper are based on an HFC MAC protocol [6] having many features expected to be in the IEEE 802.14 standard.

A block diagram of an HFC network is shown in figure 1. Data is sent from the head-end to the stations (*downstream* channel) on a separate frequency from that on which the data is sent from the stations to the head-end (*upstream* channel). The propagation delay of the cable together with techniques to reduce channel noise impairment introduces delays in the round-trip path between head-end and stations of several milliseconds. In the upstream, a time-slotted channel is shared among the subscriber stations under the control of the head-end. Stations communicate with the head-end controller to request

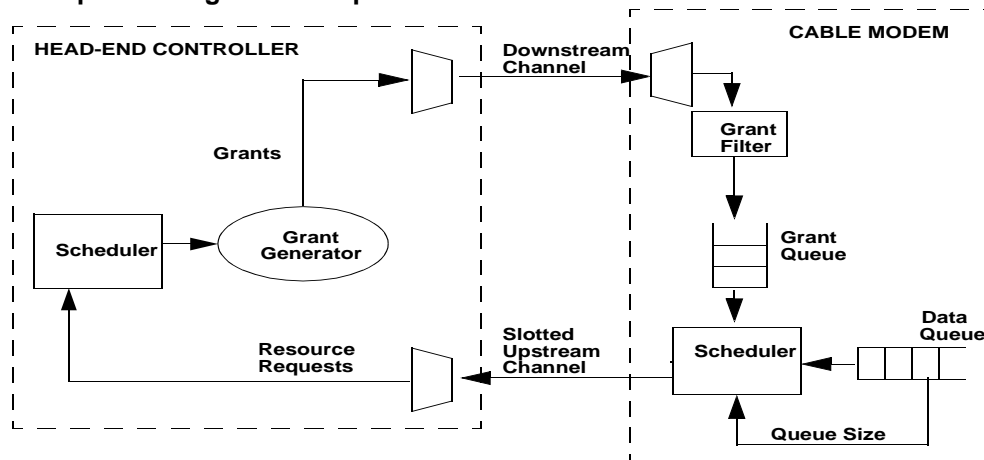
access to the upstream channel and the head-end controller's scheduling algorithm allocates these time slots using the requests from the stations and knowledge about the level of service of a particular subscriber. Stations can't communicate directly, so it's not possible to use a DQDB or CSMA MAC layer. The basic information unit 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. The standards call for supporting 50-2000 stations on a single cable.

FIGURE 1. Block diagram of an HFC network



The headend controls the upstream channel by issuing *grants* that specify the station(s) and type of messages that can be sent in each upstream slot. Contention grants are issued to groups or to all stations to send requests for bandwidth. In response to successfully transmitted requests, direct grants are issued to individual stations to send data in a particular time slot on the upstream channel. Multiple grants (up to 15) can be packed into a single downstream cell. Random access algorithms are used to resolve contention when it occurs during bandwidth requests. Data is only sent in the directly granted slots. A block diagram of the request and grant flow is shown in figure 2. A scheduling algorithm is applied to the flow of resource requests using information about the type of service of each traffic flow and grants are then allocated to specific upstream cells. The head-end scheduler implements a sophisticated allocation model while the scheduler at the station schedules based on simple priority [13]. A scheduling period of 3.2 milliseconds was used throughout this paper. The cable modem sends its queue length in every upstream cell to be used by the head-end scheduler.

FIGURE 2. Simplified Diagram of Request and Grant Flow



This makes it possible to *piggyback* data requests that arrive in the queue while it is being serviced without having to use a contention grant. This is desirable because requests made through contention are subject to collision, the resolution of which is costly in delays. Further, the physical effects of collisions on the channel may cause errors on other channels in the cable.

3.0 Traditional and Open-Loop Traffic Models applied to Cable Data Systems

Exponentially distributed packet arrivals (the popular poisson process) have been shown to represent traffic processes quite poorly and to give optimistic results (e.g., [9]), but it is often asserted that poisson modeling can be useful in evaluating a system during analysis and simulation. We begin by employing these traffic sources to model our HFC network. In the simulation, a headend module handles scheduling of the upstream channel and control of the contention resolution algorithm. Each subscriber station is modeled separately and includes both a model of the cable modem and an application model that sources traffic in the upstream direction. The upstream bus timing is modeled in detail, with time slots and propagation delays. The downstream bus was modeled only as messages passed from the head-end to the stations. The downstream messages include the major components of downstream delay, most notably FEC and interleaving delays, a total of 4.2 milliseconds in the simulations here. Variants of this basic simulation model were used by most of the participants in the 802.14 WG.

3.1 The simulator used in this paper.

Most of the 802.14 WG used Mil3's Opnet simulator. Initially, we followed this lead, but later found we needed something more "lightweight" and modifiable in order to carry out the kinds of investigations we needed. We gave up the friendly user interface and on-call support line for free source code, complete with bugs. This approach isn't for everyone, but we've been satisfied with the results. We selected Lawrence Berkeley Laboratories' *ns* simulator [14] as appearing the most germane. Our reasons included its TCP simulation model and its use of C++, a language we were quite familiar with. Subsequently, we found we had to rewrite the TCP model and made major changes to the simulator's architecture, but that's the advantage of having source code. The major output from our simulator was trace files which we post-process in a number of ways to aid in analysis. We could create traces that looked very much like the output of a *tcpdump* on a real network, as well as other custom data. For the adventurous, further information on how to follow in our footsteps is included in section 6.0.

3.2 Exponentially distributed source models.

We implemented the "batch poisson" model suggested in [2] where the number of packet sizes selected is distributed randomly as follows: 60% of the packets are 64 bytes, 6% 128 bytes, 4% 256 bytes, 2% 512 bytes, 25% 1024 bytes, and 3% 1518. Packet interarrival times are exponentially distributed and the average can be varied.

About 12% of the channel will be dedicated to contention opportunities, leaving 1.7 Mbps available for data payload. Unallocated slots are designated as contention slots. [2] suggests performance modelers plot delay-throughput curves for 10, 50, and 200 stations, varying the arrival rate and using average packet delays; 200 stations are shown in figure 3. Since HFC networks are specified for up to 2000 connected stations, we ran a second set of simulations with the per station average bit-rate set to just under 1 Kbps and the number of stations varied from 100 to 1400. These results are labeled “fixed rate srcs”. In figure 4 the median packet delays are shown. Median packet values are presented for consistency with other measures in this paper and are preferred for their stability. Note the differences at high throughput.

FIGURE 3. Average values for batch poisson

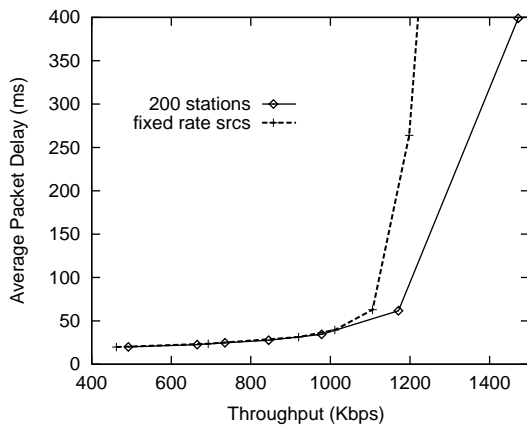
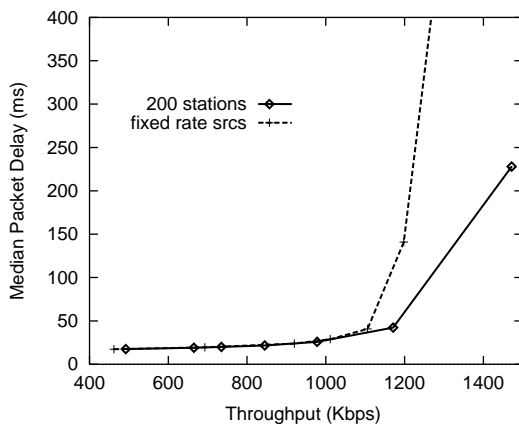


FIGURE 4. Median values for batch poisson



Architectural features dominate the delay at lighter loads, specifically the round trip time to get an allocation of upstream bandwidth from the headend, and as the load increases extra delays are introduced due to contention resolution. At the highest loads, the system becomes limited by the upstream bandwidth bottleneck and delays grow with station queue size. The two curves are initially coincident and diverge at higher throughput levels. We hypothesize that this is caused by the smaller likelihood of stations competing for the same contention grants when there are fewer stations and that more piggybacking occurs in the 200 station model. Since this model increases the load by decreasing the interarrival time, more packets arrive to a full queue and are thus piggybacked onto earlier requests. This hypothesis is verified by computing the ratio of all requests sent in contention slots to the number of total packets sent, a value that is 1.0 at light loads, 0.93 for the 1 Kbps sources at the highest throughputs, falling to 0.53 for the 200 source model. Nearly half of all packets are being piggybacked for the 200 source model!

The shape of these curves can be predicted by simple analysis. This is useful in model development as it helps validate the model’s integrity, but the purpose of simulation is architectural insight. Spreading the traffic load across a larger number of clients results in different contention behavior, something that would have been missed if we’d relied only on the 200 station models. Could we be missing other behaviors? Measurement studies consistently show low *average* bandwidths per user and bursty application behavior. The next model was developed using measurements of such an application.

3.3 Deng's WWW traffic source model.

As in the larger internet, a large segment of the cable modem user population is expected to be spending on-line time accessing world wide web (WWW) pages. Web clients look quite different from poisson sources. Building on the WWW measurement and characterization work done by Cunha et. al. [10], Deng proposed a one-way client model of web browsing [3] and made an Opnet simulation model available to the 802.14 WG. It is an ON-OFF traffic source following these steps:

1. Start 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). (Our model staggered the initial ON states to avoid synchronization.)

2. A URL request packet of size 250 bytes is produced after a delay picked from a Weibull distribution with parameters $(e^{1.5}, 0.5)$.

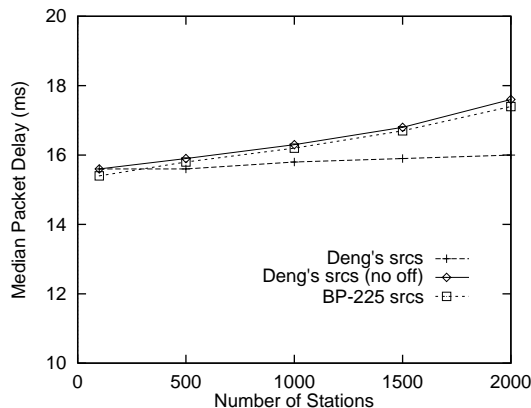
3. Repeat step 2 until the state changes from ON to OFF.

4. Return to the ON state after a delay taken from a pareto distribution with parameters $(60, 0.5)$.

Note that at least 60 seconds are spent in the OFF state, and at least 90 seconds are spent in the ON state. During the ON state, the minimum time between the fixed 250 byte (or 6 ATM cell) message is about 4.5 seconds, so the maximum source rate during the ON period is about 450 bps. This clearly produces a bursty model, though the burst sizes are not large. Though it models the user's actions, it is not possible to use this model to measure the delays a user would see as the number of clients increases. The emphasis is on reproducing the load the users put on the network in the upstream direction rather than on quantifying what the users see.

We next consider 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? We experimented with this source model in our own simulator. In simulation the average source rate is about 34 bps. Although the number of stations varies from 100 to 2000, the average packet delay remains quite low, around 16 ms across all runs. This produces uninteresting results and the long OFF times require long simulation runs (3600 simulation seconds) for stable results ("Deng's srcs" in fig. 5).

FIGURE 5. Deng's WWW model



We are interested in the system dynamics under plausible, stressful loads and observe that only OFF times depend on the subject population. Thus we do away with the OFF times entirely and rerun the set of simulations. In this case, the average source load is about 225 bps and simulations can be shortened to 600 seconds. We compare it to batch poisson sources with an average data rate of 225bps (BP-225) to evaluate the utility of the bursty model. The delay curves for these experiments (figure 5) all have a very limited range (compare to figure 3). Little difference can be seen between the effects of each of these models on the architecture. None produced significant collisions on the upstream channel contention slots. The throughput curves are even less interesting so are omitted.

3.4 Summary

The open-loop web client model doesn't stress the modeled system. Further the upstream traffic due to the TCP protocol is not modeled, so it underestimates the upstream channel traffic. The batch poisson model can be made to drive the system into overload but we question whether the excessive piggybacking and its affect on contention dynamics well represents reality. We are concerned with how HFC architectures behave under the range of light through heavy loads and wish to determine where unacceptable overloads occur. We are convinced of the value of performing evaluations using realistic, bursty traffic, but it's unclear how to scale up Deng's measurement-based distribution since an increased source rate produces essentially the same behavior as in the batch poisson model. We need a bursty, realistic model that can be meaningfully scaled.

Also, although we can measure packet delays, upstream throughput, average queue sizes, etc., we cannot determine what these values mean to a typical user. It's difficult to say what a tripling or quadrupling of millisecond packet delays will mean to a user operating on timescales closer to seconds. System overloads are determined by the performance a user sees and we need to relate architectural features to that. To capture user level performance and create stressful but realistic source traffic, we moved to an approach based on a more faithful representation of the expected applications.

4.0 Adding Feedback to Traffic Models

4.1 Discussion of traffic models for HFC

What applications are most important for HFC networks? In the words of one anonymous reviewer, traffic mix “is dependent heavily on the country looked at or the persons using it.” We chose to begin with web-browsing traffic for several reasons. Web-browsing is thought by many to be the major application of the target users (thus the use of Deng’s model by 802.14). It is a network-stressful traffic pattern since it is bursty and has short transfers that seldom reap much advantage from piggybacking. Further, it is an interactive application so delay times matter. Interactive gaming has turned out to be a very important use of HFC networks and we encourage future studies with such a model. Long-lived FTPs are less interesting since they will capitalize on the request-grant architecture. The problem there is to prevent such applications from capturing most of the channel bandwidth (subject to economic factors). This is possible in HFC networks and we have published an approach [13].

Continuing with WWW modeling and our desire to more faithfully represent network traffic, we observe that the web browsing client-server conversation might be simulated, leading to a scalable model, more insights on system behavior and the ability to measure some quantities that pertain to application-level performance. Most of our work focused on web-browsing using http1.0. Our experiments and the published work of others indicates that this leads to more pessimistic assessments than using http1.1. In simulation, such pessimism can be useful to balance the necessary simplifications we make that cause results to be more optimistic than real-world measurements.

4.2 A more detailed client model for simulation

A typical process initiated when a user makes a request for a particular URL is:

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 which returns a SYN acking the connection after some service time delay. Upon receipt of the SYN, the URL, typically 200-250 bytes, is sent to the remote server. (An ACK of this SYN may either be piggybacked with the URL or sent alone.)

3. After the service time delay, the remote server returns the requested page data including the in-lined URLs referenced on that page. As the data packets are received, ACK(s) and FIN are sent for this initial TCP connection. (When browsing in a “graphics off” mode, the page would be displayed and done at this time.) At the same time, SYNs are sent to open parallel connections for all the in-line URLs found on the page.

4. After the service time delay the remote server(s) return SYNs acking the connections and the in-lined URLs 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.

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

For a web-browsing user, the performance measure of interest is how long it takes to “see something”. The complete page can be displayed once all the data associated with all the URLs in the page have been received (step 5; step 3 for “graphics off”). Ignoring host and display time, this measure is a rough estimate of the delay a user sees in a real system.

4.3 Assumptions

We now have a model with figures of merit that have meaning at the user level. To implement it, we must model the downstream channel and add a model of transport layer behavior. We also need representative values for the web browsing parameters. In addition to the pioneering work of Cunha et al [10], work on measurement and modeling of WWW clients is continuing by other researchers. Mah at U.C. Berkeley has done measurements on some campus network segments and used these in trace-driven simulations [12]. We combine the measurements of Cunha and Mah to obtain representative values.

Some assumptions were made to expedite the simulation. In steps 2 and 4 all URLs were assumed to fit into six ATM cells (250 bytes). Further step 4 uses exactly three in-lined URLs, a pessimal value consistent with Mah [12]. In steps 3, 4 and 5 a fixed server delay time is used. Although this departs radically from behavior seen today,

there are reasons it is justified for this study. First, companies planning data network services over cable are investing in large caching servers to provide response time acceptable for a commercial service and secondly, server and network delays are beyond the control of the architecture we are considering.

The document size distributions used in steps 3 and 5 were based on information from both Cunha and Mah. Mah found that the distribution of document sizes returned in response to the initial URL had different parameters than that of document sizes returned in response to in-line requests so we use a different document size distribution in step 3 than in step 5. However the largest outliers are clamped since we want our simulations to be test of the network's architecture not a completely faithful representation of web browsing. In step 5, the distribution is used to pick each of the three document sizes which are then converted to a number of maximum sized ethernet packets (1460 data bytes per packet).

Particular TCP implementations vary in the number of ACKs sent and whether ACKs are piggybacked on FINs and SYNs [15]. We assume the client sends an acknowledgment every two packets and combines FINs and ACKs in a single packet.

A difficult problem is picking the average “think time”; that is, the average amount of time a user waits before selecting another page. Although [3] has some measurements of this value, it is unclear how well these apply. It could be extracted from the traces at [11] but these were taken using an unconventional browser, mosaic. A broader question is whether measured values for this timing are useful. Knowing how often users are active is useful in provisioning a system, but it can be misleading. Future developments in browsers might affect these times. Home web-surfers might be found to use the web differently from work and school web-surfers (picture the “couch potato” web surfer doing the equivalent of rapid channel surfing). There may be coincident use patterns so that designing to the average is dangerous underprovisioning. With our goal of exercising architectures in mind, think time was set to be random and uniformly distributed between one and 30 seconds, a fairly conservative (or pessimal) assumption. Altering the think time makes it possible to increase traffic levels while preserving the application behavior.

To speed evaluation, an intermediate approach was taken initially. The web-browsing interaction was used as

the basis for a *local feedback (LF)* source model where backpressure from the modem queues provides feedback to the traffic sources. At each packet send, the LF source sends a message to the cable modem then *waits until the modem signals that its queue is empty* before it adds the server delay and goes on to the next step. Neither the upstream transmission delay of the last cell nor the downstream transmission delay is included in this model. Only part of the transport protocol behavior is modeled and simplifying assumptions were required. 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. Acknowledgments for a window's worth of packets for each TCP connection is sent after each empty queue signal plus server delay. This clearly ignores significant transport protocol interactions, possibly causing the page delay time to be underestimated.

4.4 Two-way world wide web client-server model

Full two-way (FT) client-server interaction necessitated rewriting the TCP model in the ns simulator distribution to set up and take down connections, to handle bidirectional traffic, and to handle variable packet sizes. WWW client models generate URL requests and hand the packets to the TCP connection management which generates the appropriate SYNs, FINs, and ACKs and returns the transferred document size to the client. Packets are segmented into cells and sent through the upstream channel to a server model attached to the headend. The server generates a protocol or data packet which is sent on a downstream channel to the appropriate station. The server delay (again 20 ms) is charged on the trip to the server from the headend and the server selects document sizes to return in the downstream.

Results of these experiments are in figures 6 and 7. The average rate for an LF client in a lightly loaded system is about 920 bps and the average rate for an FT client 980 bps, much higher than in Deng's model. For comparison, a batch poisson model with 925 bps sources (BP-925) is shown. The curves are very similar at light loads, but as the number of stations increases and delays and contention increase, both types of feedback stations cut back their rates as real clients would. Thus median packet delays differ significantly above 750 stations.

FIGURE 6. Local Feedback, Full Two-way, and Batch Poisson

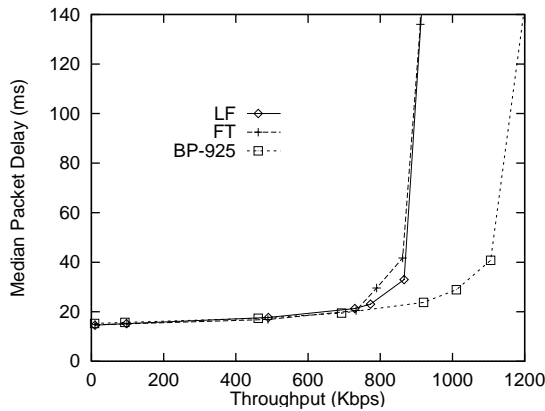
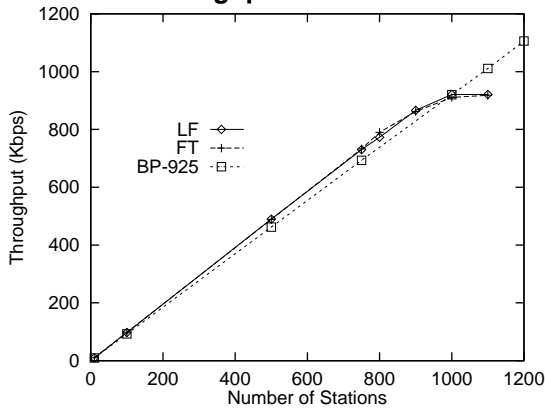


FIGURE 7. Throughput vs. station count



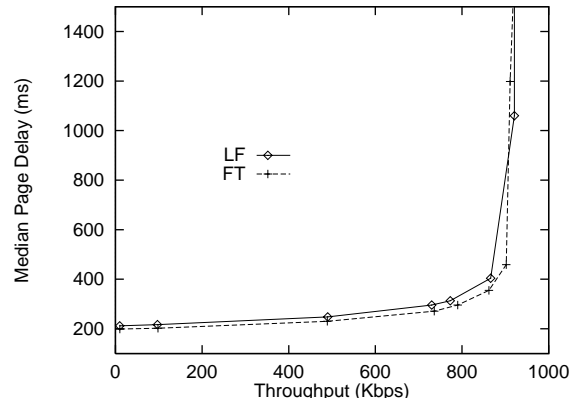
LF and FT models' throughput levels off around 1000 stations because feedback will not allow them to send packets into a saturated network. The open-loop BP model continues to put out the same average data rate per source and shows a sloping curve, approaching upstream bandwidth saturation while the feedback models quickly approach vertical. More differences between open-loop and feedback models will be shown later.

4.5 User-level metrics

A further advantage of the feedback model is that we can measure the user-level metric of page delay (see figure 8). There is a clear knee point after which additional clients result in extreme additional delays, but the exact loca-

tion of this knee is particularly dependent on assumptions about the user think time distribution. The two models produce similar results for this metric, due in part to the fairly detailed LF model used and in part to not modeling the FT path beyond the headend.

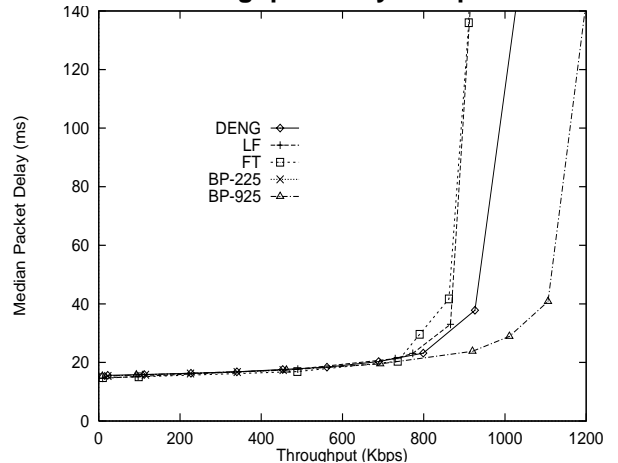
FIGURE 8. Page Delay



4.6 Comparing the models

We further explore similarities and differences across the models. Five models are used: Deng's sources with no OFF state (DENG), the local feedback web browser model (LF), the web browser with a full TCP model (FT) and two batch poisson models (BP-225 and BP-925). We start with throughput-delay curves (figure 9). We simulated DENG sources with up to 4500 stations (unrealistically large) in order to evaluate effects of its heavy-tailed distribution. At

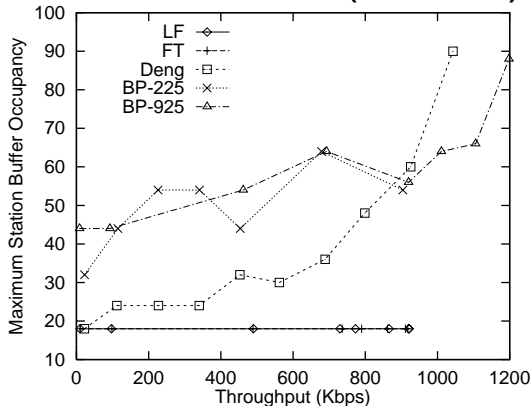
FIGURE 9. Throughput-Delay Comparison



lower throughput all the curves are coincident, showing delay is primarily a function of throughput in that region. As throughput increases the traffic model becomes more important. The BPs are the most optimistic, below and to the right of the other curves. BP-225 is difficult to see as it has a maximum throughput of about 450 Kbps. DENG has a packet delay that lies above the BP curve closer to the feedback curves, but rises to the right of the feedback curves and with a slope that more resembles that of the BP curve.

Next we look at the upstream packet buffers. For each simulation, the maximum per-station buffer occupancy in cells was recorded. In figure 10, this metric is shown as a function of throughput and in figure 11 as a function of the total number of stations. Previously, we saw that piggybacking increases in open-loop models under heavy load. This is confirmed in figure 10 where maximum buffer size increases with throughput for all of the open loop models, but stays flat for the feedback models. These large buffer sizes cause a “false” piggybacking effect because this is not behavior exhibited in a real system. True piggybacking occurs in the feedback WWW models at all traffic intensities resulting both from opening all of the in-line connections at the same time and from getting windows of ACKs on parallel connections. We know the amount of piggybacking cannot be any greater than this due to the TCP/IP dynamics between the client and server. Real-life clients throttle back under load and/or data is dropped.

FIGURE 10. Maximum buffer (in ATM cells)



Contention behavior is critical in HFC architectures. The collision rate was recorded for each second of simulation time after the start-up period of 30 seconds (figure

12). At any value of total throughput, models with lower per-station rates should have more collisions since they require more simultaneously active stations to achieve the total. In the FT and LF models, upstream packets are either 64 byte protocol packets or 250 byte URLs, compared to the much wider BP packet size distribution (section 3.0). Because of the smaller packet sizes, we expect additional contention and channel utilization differences.

FIGURE 11. Maximum per-station buffer (cells)

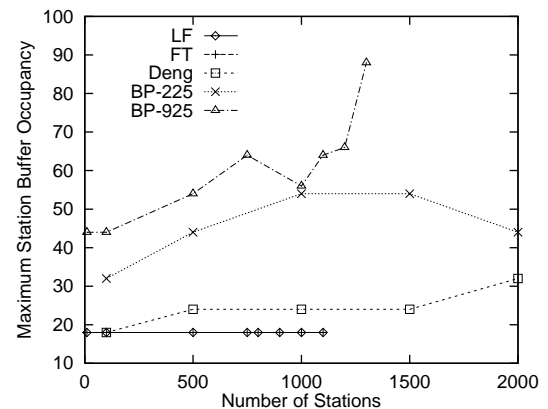
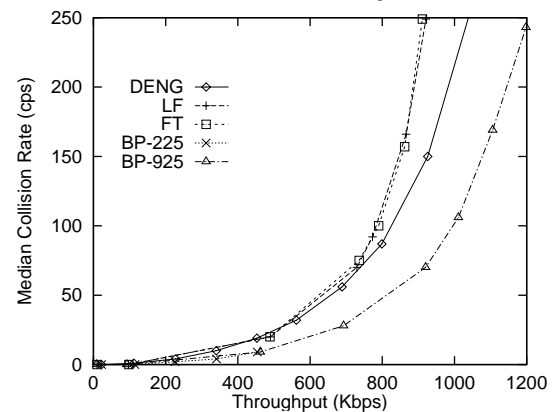
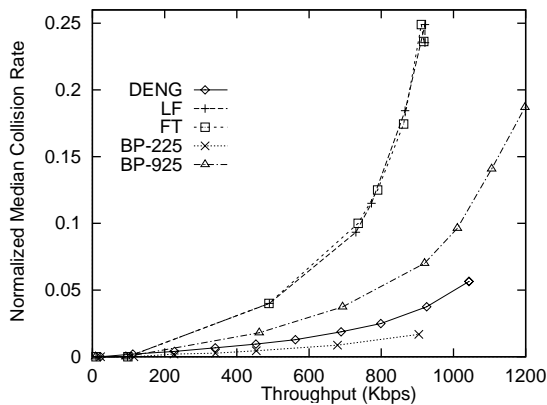


FIGURE 12. Collision rate comparison



To compare collision rates using both the importance of station count and of throughput, the median collision rate was also normalized by the number of stations and plotted against the total average throughput (figure 13). This represents the normalized collision rate the upstream channel sees not the rate at which each station has a collision, since each collision represents two or more stations.

FIGURE 13. Normalized collision rates



The two feedback models are strikingly similar in both graphs. DENG has a total median collision rate curve that is identical to the feedback models at lower throughputs, then diverges. Keep in mind that the DENG values came from many more stations than for the feedback models which becomes apparent in the normalized graph. The feedback models' curves fold back onto themselves at the top of the curve where the total system load caused clients to throttle back sufficiently to decrease the median contention rate per station. The open-loop models instead start to flatten out as the piggybacking rate goes up. Open-loop models produce little contention when the load and number of stations is close to that of feedback models. DENG has no significant collision rate at numbers of stations within the limit of 2000 stations.

4.7 Discussion

Open-loop sources 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 cause large backlogs in an infinite queue system (most models) or be discarded in a finite queue system. Large backlogs are problematic because they will cause packet delays that primarily reflect queuing delays and because piggybacking completely removes these backlogged stations from the contention process and thus mitigates the critical loading on the contention resolution mechanism. There are few real applications that function this way since feedback is a major part of the network sys-

tem dynamics. Real applications enter into a transport protocol level conversation with a remote peer thus will not continue to fill a cable modem's queue beyond the amount proscribed by the transport protocol. Applications based on TCP/IP will not have more than one window of data outstanding (a maximum of 16-64 Kbytes, depending on the implementation).

In most cases, an open-loop model using a heavy-tailed distribution (DENG) produces performance curves that lie between those of feedback and exponential models. The open-loop source model Deng developed by fitting data from network traces to heavy-tailed probability distributions faithfully captures some of the burstiness of the original application but fails to reproduce the behavior as sources adapt to network characteristics through interaction with the architecture and other network flows. Further, the sources cannot be meaningfully scaled to represent heavier usage per source. Deng's model ignores the TCP-based packet traffic, which accounts for most of the upstream traffic during web-browsing and does not permit measurement of user-oriented metrics.

The local feedback model closely tracks the upstream traffic produced by the full two-way TCP model. Though local feedback can be easily added to any model that is presently using open-loop models, they have three major drawbacks. First, the feedback structure needs to be customized to the application. For example, to create a file transfer model, the client model must be rewritten, including its TCP portions. Second, the return traffic path is not modeled so we cannot see its effects on system and application performance. Third, it is not possible to model server contention or resource-sharing problems beyond the upstream channel. A final caveat is that the fidelity of the LF curve fit to a full two-way feedback model will depend on the application modeled and how well it can be represented with an LF model. For example, an earlier version of the LF model had a less faithful TCP model, Though it produced curves that were similar in shape to those of the FT model and much more like FT than open-loop models, they were noticeably more optimistic than the FT curves.

Though metrics based on throughput can be useful, we are particularly concerned with how user perception of performance is affected as the number of users and their network use varies. We expect the number of stations that are active to change with time of day and HFC systems need to adapt accordingly and to avoid the worst perfor-

mance ranges. Feedback models can be used to help understand how installed systems will function.

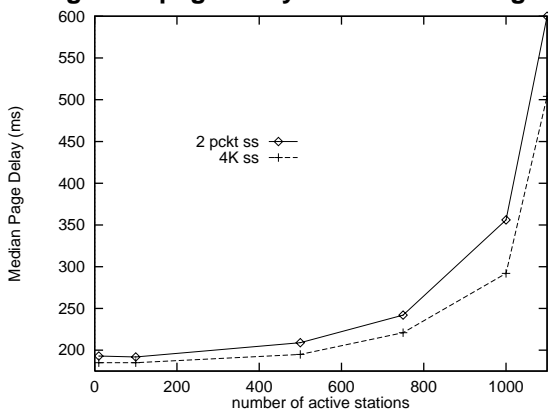
5.0 Results using the Feedback Model

A full, two-way TCP model makes it possible to add new application types and to observe the effects of parameter and architectural changes on them. Focusing on two applications, the web browsers of the last section and an FTP source, we show the effects of an increase in the initial TCP slow start window value on web-browsing sources and the effects of TCP window size changes on ftp throughput. In addition we describe and present the effects of a MAC layer architectural enhancement that uses knowledge about transport protocol behavior to improve performance by predicting when a station will be requesting a grant. This allows us to speed up response times and decrease contention.

5.1 Changing the initial window

The TCP used in the experiments sends at least one ACK for every two packets, uses a 2-packet initial slow-start window, and an 8 Kbyte (6 packet) maximum window. Figure 14 shows the results of increasing the slow-start initial window to 4 Kbytes on median page delays of web-browsing clients, a change that has been proposed for TCP [16]. The improvement in median page delays varies from 4% at the lightest loads to 16% at 1100 stations. The 4K slow start allows three packets to be sent in the first window of data, only one additional packet. The dramatic

FIGURE 14. Effects of slow start window changes on page delays in web browsing



effect as the number of stations increases results from saving about one contention per URL fetched. In addition, the entire response fits in the 4 Kbytes more than 50% of the time.

5.2 Debugging prototypes

It is straightforward to use the full TCP model in an FTP application, a client and server where a file is transferred to the client from the server. A single client-server pair was used to help evaluate measurement results on initial products; simulation and measurement traces were correlated for insights on performance. Increasing the maximum TCP window fills more of the bandwidth-delay pipe and greatly improves throughput (table 1). Note the bandwidth-delay pipe is kept full with a 48 packet window. In addition, at this window size, all the upstream ACKs have their requests piggybacked on earlier ACKs, considerably reducing round-trip delay. The upstream direction saturates before the downstream since about 50 times as much data must be sent in that direction and the bandwidth asymmetry is only a factor of 10. Table 1 shows the upstream and downstream average data throughput for a range of window sizes given in number of packets.

TABLE 1. Channel throughput for different window sizes

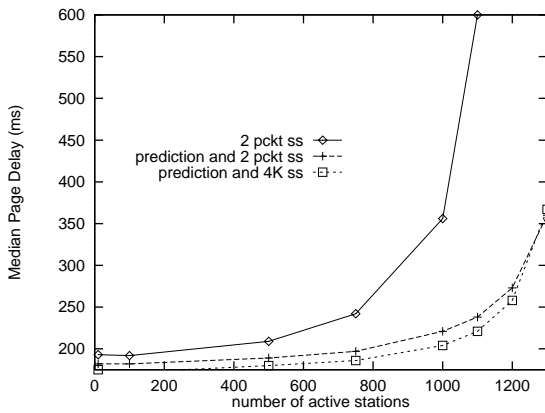
	6	12	24	36	42	48
upstream (Kbps)	55	109	239	388	460	500
dwnstream (Mbps)	2.6	5.2	11.4	18.4	21.8	23.7

5.3 Architectural enhancements

The ability to evaluate architectures and to find opportunities for their enhancements is an important benefit of a realistic simulation. Testbed results were cross checked with simulator results to develop our intuition about traffic behavior for this architecture and to understand how to improve performance. One novel technique emerged as we noted that, in general, a data packet downstream results in an acknowledgment upstream. This observation was used to allow the headend to predict appropriate times to give grants to stations without waiting for them to contend. There are several ways to implement this feature, details are not given here. A conservative version of grant prediction was simulated for both web-browsing and for FTPs.

Figure 15 shows the result of repeating the experiments of figure 14 using predicted grants, including the 2 packet curve from figure 11 for ease of reference. Using predicted grants decreases page delays and supports more stations. In this case, the slow-start window increase results in a smaller range of improvements, 4% to 8%, since contention is already reduced. Thus these values reflect the more architecture-independent effects of increasing the window size. The median page delays are nearly the same at 1300 stations once upstream channel utilization becomes quite large, but this is the beginning of overload effects.

FIGURE 15. Effects of prediction grants on median page delays for web browsing



Predicted grants remove much of the contention from an HFC network. Not only does this reduce delays, it can improve fairness. The latter is best illustrated using FTPs. Average up- and downstream throughputs are listed in table 2 for 1, 5, and 10 active stations, both aggregate and per station values. Table 3 shows the same data when prediction grants are used.

TABLE 2. FTPs throughput without prediction

N	agg up	agg dwn	sta up	sta dwn
1	55 Kbps	2.6 Mbps	55 Kbps	2.6 Mbps
5	263	12.5	53	2.5
10	478	22.7	48	2.3

In this scenario, the channel is shared fairly, both with and without predicted grants (differences between the per station rates are less than 1%). Predicted grants result in about 10% more throughput because they lower the round trip delay. The small number of stations means there are

few collisions, so the gain in performance comes almost solely from decreasing the time to request a grant and not from saving the time to resolve a collision. For 10 active stations, the downstream data payload reaches saturation, but the upstream channel is only about 25% utilized.

TABLE 3. FTP throughput with prediction

N	agg up	agg dwn	sta up	sta dwn
1	60 Kbps	2.9 Mbps	60 Kbps	2.9 Mbps
5	300	14.2	60	2.9
10	500	23.7	50	2.4

To congest the upstream channel, we did a “reverse” FTP (file transfer from station to HE). The MAC layer scheduling ensures all stations with packets to send get a minimum share of the channel but does not upper bound station traffic when there is channel bandwidth available. The reverse FTP uses all the upstream bandwidth it can get, reducing the contention slot allocation to its minimum value of 12% of the upstream channel. Table 4 lists results for a single reverse FTP: alone and with one and two “normal” FTPs. Table 5 adds prediction. Prediction has no effect on the reverse FTP because all of its packets are piggybacked, using no contention grants. The “normal” FTPs suffer without predicted grants because the reduced number of contention slots results in extra delay at each contention, thus extra round trip delay. Prediction removes this delay, increasing the downstream throughput from 1.8 Mbps to 3.0 Mbps per station, a fairer channel use.

TABLE 4. Throughput with a “reverse” FTP

FTP	station to HE		HE to station	
	up	down	up	down
0	1.7 Mbps	34 Kbps	-	-
1	1.6	34	38 Kbps	1.8 Mbps
2	1.6	33	37	1.8

TABLE 5. Throughput with a “reverse” FTP and prediction

FTP	station to HE		HE to station	
	up	down	up	down
0	1.7 Mbps	34 Kbps	-	-
1	1.6	34	64 Kbps	3.0 Mbps
2	1.5	32	59	2.8

It's important to balance the potential gain of sending a predicted grant against the cost of inaccurately predicting a grant and wasting the time slot on the upstream channel. The examples shown here had very few of these misallocated slots, but these applications lend themselves to the use of this feature. The predicted grant implementation was a conservative one. More sophisticated implementations might have traffic "learning" features included.

Bypassing contention is crucial in order to reliably deliver committed service tiers to different stations. This, in addition to its other performance properties, makes prediction an important technique to pursue. Intelligent use of predictive grants in concert with the delivery of different service tiers requires some further study. This feature cannot be meaningfully studied without a realistic closed-loop simulation. In addition to discovering this predicted grant mechanism, these experiments helped us to understand many aspects of the dynamic behavior of the architecture.

6.0 Summary and Pointers to Models

6.1 Summary

When constructing a simulation model, there is a danger of oversimplifying and arriving at inaccurate conclusions. Previous researchers have warned that poisson models do not well represent real traffic though some performance modelers still feel they give valuable system information. When applied to our architecture, we found the poisson models gave us performance numbers that are clearly wrong and dangerously optimistic. Further, we found that using open-loop models in general appears to give optimistic performance and doesn't allow us to measure the kind of quantities we need to evaluate user-level metrics. Our investigations convinced us that we must model MAC layer protocols with sufficient features of the application and transport protocol behavior to design architectures that will function well when deployed in the real world. Open-loop models can be useful in simulation model development, but should be approached with extreme caution for architectural appraisal.

A promising technique intermediate to a full two-way model is to use "local feedback". In our case local feedback is achieved by monitoring the status of the station queues that feed the upstream channel. For several important performance measures, this gives results quite close to the two-way model. It can be much easier to add this type

of feedback to an existing model by modifying the source models than to make a full two-way model. For some simulators, this may also be preferable so that the simulation run-time does not become prohibitively long. Drawbacks are that local feedback must be customized for an architecture and a particular type of source, and that delays and congestion of the return traffic path is not modeled.

The original impetus for this work was twofold: to understand the effects of realistic traffic on our architecture and to make it possible to study more thoroughly the delivery of different tiers of service to different stations, work we initially reported on in [13]. In pursuing models for this purpose, we found several architectural interactions of interest and performance improvement techniques, including the predicted grants described in section 5.

This paper has presented summary statistics, but working with the model showed the dynamic behavior of the models are more divergent. It is difficult to capture and present these results. Future work might look at comparisons of open-loop and feedback models with more emphasis on dynamic behavior differences.

6.2 Pointers to useful simulation models

Although we started with a public domain simulator, it is not possible, or even desirable, for others to pick up the code used here. One problem is that the MAC model used here is proprietary to Com21. The ns version used here was changed in many ways from the original distribution: the TCP model was greatly enhanced, a more efficient event queue was added (the original had a linked list), web and file transfer clients and servers were created, and major architectural changes were made to the simulator to make it easier to add a MAC model. It is still possible for interested readers to perform simulations similar to those reported on here. Since this work was begun, the ns simulator has been superseded by the ns-2 simulator [17], which has a basic architecture more similar to the Com21 custom ns and includes some MAC models (but not, as of this writing, an HFC one). Further, we rewrote one of the TCP models of ns-2 to be an even closer model of reality. This model and web and FTP traffic models were used to study effects of changing the TCP initial window size. This was reported on to the Internet Engineering Task Force's (IETF) TCP Implementors Working Group and written up in an Internet Draft (which is expected to soon become an informational RFC). We've made these models

available at the ns-2 “contributed code” web site [18]. We recommend use of ns-2 with our code modules for those who want a simulation environment close to that described in this paper.

7.0 Acknowledgements

The author was at Com21, Inc. when this work was done and would like to thank Mark Laubach of Com21 for encouragement to persist with traffic-oriented models. In addition, Van Jacobson of Lawrence Berkeley National Laboratory shared much advice, tools for viewing data, and informed many of this paper’s insights.

8.0 References

- [1] W.E. Leland, M.S. Taqqu, W. Willinger, and D.V. Wilson, “On the Self-Similar Nature of Ethernet Traffic,” Proceedings of ACM SIGCOMM '93, pp. 183-193.
- [2] J. Limb, et. al., “Performance Evaluation Process for MAC Protocols”, IEEE 802.14 working group document 96/83R2
- [3] S. Deng, “Empirical Model of WWW Document Arrivals at Access Link”, IEEE ICC '96.
- [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] M. Laubach, “To Foster Residential Area BroadBand Internet Technology”, Connexions, February 1996 Vol 10, no. 2, p 18-30.
- [6] M. Laubach, *The UPSTREAMS Protocol for HFC Networks*, proposed to IEEE 802.14 Working Group, contribution number IEEE802.14-95/152RI, January, 1996.
- [8] See <http://walkingdog.com> for further information about IEEE 802.14.
- [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-95-0 10, July 18, 1995.
- [11] Traces at <ftp://cs-ftp.bu.edu/techreports/95-010-web-client-traces.tar.gz>
- [12] B. Mah, “An Empirical Model of HTTP Network Traffic”, *to appear in* Proceedings of INFOCOM '97, Kobe, Japan, April 7-11, 1997.
- [13] K.M. Nichols and M. Laubach, “On Quality of Service in HFC Networks”, IEEE ATM'96, San Francisco, CA, August, 1996
- [14] Available at <http://www-nrg.ee.lbl.gov/ns>
- [15] V. Paxson, “Measurements and Analysis of End-to-End Internet Dynamics”, Ph.D. dissertation, University of California, Berkeley.
- [16] http://www-nrg.ee.lbl.gov/floyd/tcp_init_win.html
- [17] Available at <http://www-mash.cs.berkeley.edu/ns/>
- [18] Available at <http://www-mash.cs.berkeley.edu/ns/ns-contributed.html>