Multi-scale Validation of Structural Models of Audio Traffic

Kun-chan Lan, John Heidemann*
kclan@isi.edu, johnh@isi.edu

Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina Del Rey, CA 90292

ISI-TR-544a†

April 4, 2002

Abstract

This paper identifies the structural properties of RealAudio traffic, develops an application-level model and validates the model using multi-scale analytic techniques. RealAudio traffic is quite different from web traffic and constant bit rate traffic, models commonly used for Internet and multimedia traffic respectively today. We demonstrate that RealAudio has rich behavior across a range of time scales, with regularity at large time scales, strong periodicities from RTT to a few seconds, and complex behavior at small time scales. A simulation model was developed which captures the main structural characteristics of RealAudio, and demonstrates the importance of multi-scale analysis in validating this model. We also demonstrate via simulation that RealAudio loss rates can be noticeably reduced by slightly changing its characteristics to reduce burstiness. Although our results focus on Real Audio traffic, this methodology may be useful for studies of other types of real-time streaming media.

1 Introduction

The rising popularity of Internet has increased use of audio and video applications, including streaming playback of music, sports and news, as well as real-time voice telephony and conferencing. With increasing deployment of new broadband technology like DSL and cable modem, continuous improvement of PC performance, and better multimedia encoding protocols, we expect this type of traffic will continue to grow. Unlike web traffic, most of the streaming multimedia applications are sensitive to delay and jitter, and transmitted at relatively large and constant bit rate. Multimedia streams often have long duration compared to the request/response nature of traditional HTTP traffic. They typically employ UDP as their underlying transport protocol, and react slowly, if at all, to network congestion. Potentially this class of traffic has a very different traffic profile compared to web traffic and as the amount of such traffic grows, the network performance might be affected significantly. It is important for network engineering to understand the nature of multimedia traffic. Although there have been several research efforts in characterizing audio [1, 2] and video traffic [3, 4, 5, 6, 7] over the years, the study of multimedia traffic is still in a preliminary state.

To evaluate protocols for scalability over large scale topologies under varying network conditions, simulations are necessary when real experiments may be too expensive or impossible. Useful simulations require accurate models that capture important characteristics of the network under study. While a number of models have been proposed to simulate web traffic [8, 9], there is little attention being paid to modeling multimedia traffic like audio and video stream applications. In this paper we emphasize the use of structural models, which is contrast to the traditional modeling methodology from time series analysis that usually ignores specific physical features of the underlying communication network structure. We design a structural model that captures the characteristics of RealAudio traffic at three different levels (namely, user-, flow-, and packet-level), and

*Kun-chan Lan and John Heidemann are with University of Southern California, Information Sciences Institute This material is based upon work supported by DARPA via the Space and Naval Warfare Systems Center San Diego under Contract No. N66001-00-C-8066 ("SAMAN"). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Space and Naval Warfare Systems Center San Diego.

†The first version of this paper was published on Nov 1, 2001

validate it with multi-scale analysis.

Examination of simple first-order statistics such as flow duration and inter-arrival rate is commonly used for validation of simulation model. However, recent studies [10, 11] have shown that wide-area network traffic (WAN) exhibits complex scaling phenomena over a range of time scales, i.e. multi-fractal at small time scales (a few hundreds of milliseconds and below) and self-similar at large time scales (a few hundreds of milliseconds and larger). Comparison of simple statistics will not verify the presence of these more complex phenomena. In this paper we demonstrate the use of multi-scale analysis to debug and validate models of traffic in simulation. This analysis identified synchronization issues in traffic that eluded basic statistical analysis but are crucial to understanding the queuing behavior of streaming traffic.

In our work, we develop an empirical model for RealAudio traffic, which has become a significant audio traffic source, thanks to increasingly widespread commercial use of the Internet. Our approach is based on packet traces of RealAudio traffic collected on audio servers. Through careful traffic analysis, we determine statistics and distributions for high-level parameters such as audio flow duration, flow rate, pattern of burstiness, user behavior and other characteristics. We use these quantities to build a model that can be used by simulation to mimic the RealAudio application. Our ultimate goal is to extend the existing knowledge of proposed web models with a better understanding of how to model streaming media. This will provide better insight into modeling the global Internet, where the volume of multimedia traffic is increasing [12].

Our study makes three contributions to understanding and modeling RealAudio traffic. First, we find RealAudio has richer behavior across a range of time scales, with regularity at large time scales and complex behavior at small time scales. We then provide explanations for these observations. (Section 3) Second, We design a simulation model that captures the main structural characteristics of RealAudio, and demonstrate the importance of using multi-scale analysis to validate the model. (Section 4) Finally, we demonstrate that we can noticeably improve the performance of the RealAudio protocol by slightly changing its characteristics. (Section 5)

2 Background

We use two traces in this paper. The primary audio traces used in our analysis are the same set of data used in a previous study of RealAudio [2]. (We primarily use trace 3 from that work, and check our results against traces 4 and 5. They are summarized in Table 1.) They were captured from the popular Internet audio service at Broadcast.com [13], and obtained using tcpdump running on a separate host. The trace host was connected to the Switched Port Analyzer (SPAN) port of a Cisco 2924 Fast Ethernet Switch. The SPAN port mirrors the traffic from any port on the switch and captures all of the traffic originating from or destined to the audio server. The traces were obtained from different audio servers at the main Broadcast.com site. The servers analyzed use RealServer V5.0, which employs a proprietary protocol called PNA [14] as its streaming transport protocol, from RealNetworks to provide audio streams. We don't know what is the specific OS used in audio servers of Broadcast.com. But they typically utilized Intel Pentium II class hardware running Windows NT or Linux.

Trace	**3**	**4**	**5**
Date	Jun 99	Jun 99	Jun 99
Start time, GMT	16:02	13:32	13:38
Duration (hr)	5.5	10.5	18.2
Packets	5.5M	1.6M	5.9M
Bytes	1.3G	0.4G	1.3G

Table 1: Summary of RealAudio Traces

Besides utilizing the traces from Broadcast.com, we also collected another week-long RealAudio trace from a campus web site, where the server runs RealServer G2 [15]. Although this new trace is not used for our primary conclusion, we use it to validate and generalize our results.

Finally, to supplement our understanding from trace analysis, we perform several experiments, where eight clients are connected to the server via a LAN. The server uses a trial version RealServer G2, which only supports up to eight concurrent audio sessions, and runs on a Pentium III Linux box. (To insure that these results are not OS-dependent we also repeated these experiments on a Pentium II computer running FreeBSD v3.3.) All clients use RealPlayer 8.0 Basic and utilize Intel Pentium II/III class hardware running Linux.

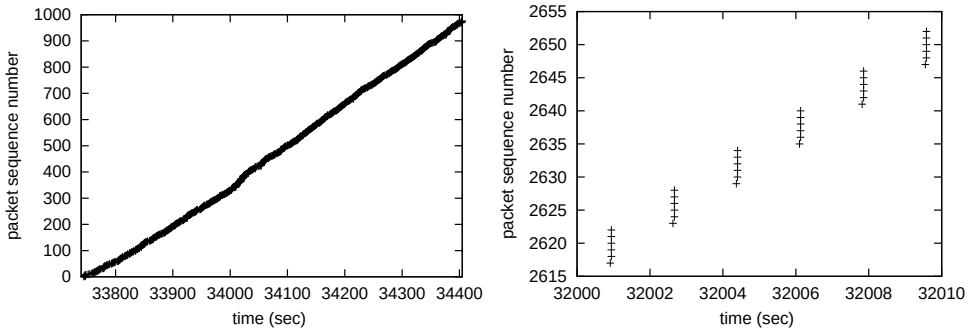
3 Characteristics of RealAudio

Based on a previous study in [2] and the results of our further analysis of RealAudio trace, in this section we present some structural properties of RealAudio traffic.

3.1 Existing work

Here we summarize some key characteristics of RealAudio traffic based on the results from [2], in terms of individual flow and aggregated traffic respectively

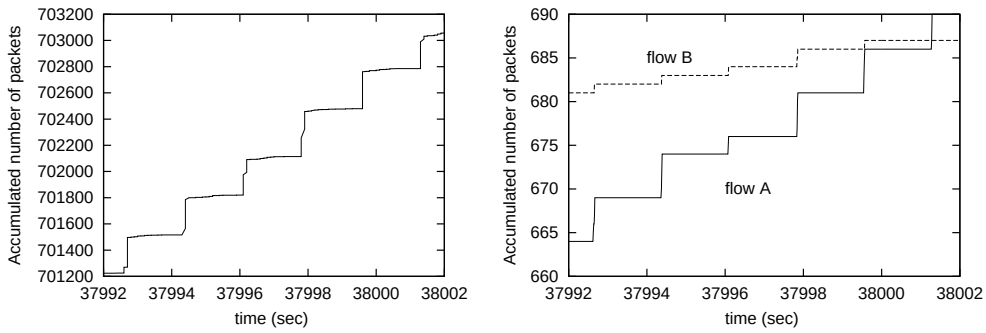
If we look at a *single* RealAudio flow, we can see RealAudio data is sent at constant bit rates at medium time scales (tens of seconds), as shown in Figure 1(a). However, at small time scale (single seconds), it behaves like a bursty on-off source with off period in approximately multiple of



(a) medium-timescale view of flow

(b) fine-timescale view of flow

Figure 1: Packet Sequence Number Plot for one single flow of the trace



(a) Burstiness in aggregated traffic (averaged over 10ms)

(b) synchronization of two individual flows

Figure 2: Synchronized RealAudio flows from the trace (y-axes have different scales)

1.8 seconds as shown in Figure 1(b).

If we look at the *aggregated* RealAudio traffic, the vast majority of audio sessions have only employed one concurrent flow. All the audio sessions use either one or two flows. Those employing two flows use a TCP flow for control and a UDP flow for data. Those employing one flow use TCP alone. Most of the data (60-80%) is sent by UDP, while the rest by TCP. Our analysis focuses on these UDP flows. More than 90% of flows have duration longer than 10 minutes. The packet size of RealAudio was found to concentrate on some particular length (244/254), which might be useful to serve as a tool for identifying RealAudio flows among different types of traffic. Finally, like web traffic, the observed user arrivals of audio sessions also strongly correlate to time of the day or start of events.

3.2 Bursty aggregate traffic

After further examining the trace, we were surprised to find that aggregated RealAudio traffic also shows similar bursty on-off behavior as the individual flow with off period approximately 1.8 seconds, as shown in Figure 2(a). Put in another way, RealAudio flows to different users appear to be synchronized, transmitting together at about the same time. This synchronization can be seen in Figure 2(b) where two randomly chosen flows show bursty transmission in phase with each other. A previous study [16] showed the synchronization phenomenon is also observed in routing traffic. Note that Floyd and Jacobson showed that multiple routers can become synchronized while we will show that multiple streams from a single server can be systematically synchronized due to timeliness issue (playback or stored or live content).

3.3 Is RealAudio Traffic Self-similar?

One interesting issue discussed in recent literature is the statistical self-similarity and long-range dependence behavior in both local and wide area networks. Previous studies [17, 18, 11] showed that self-similarity in data networks can be rooted from higher layer protocols and user-related variability. However, while there have been numerous studies of the cause and characteristics of self-similarity in web traffic [17, 10, 11, 19, 20], there has been relatively little effort in examining this aspect for streaming media traffic in the Internet. Although intuitively the periodic bursty nature of RealAudio traffic, as described in Section 3.2, does not suggest it is self-similar, we still want to see if the user-related variability induced by RealAudio clients will affect the traffic in some unexpected way. For example, Garrett and Willinger [4] showed that self-similarity can be resulted from the dependence of bandwidth variation on scene changes for VBR video traffic. In this paper, we use two methods to test for self-similarity of RealAudio traffic. Details of these methods are described fully in [11, 21]. In addition, we use them to gain insight into the structure of RealAudio itself. Here we present a summary of these techniques.

The degree of self-similarity of a time series can be expressed using a single parameter known as Hurst parameter (H). For self-similar series with long-range dependence, $1/2$ <H< 1. As $H \longrightarrow 1$, the degree of both self-similarity and long-range dependence increases.

The first method, the time-variance plot, relies on the slowly decaying variance of a self-similar series. The variance of $X^{(m)}$ is plotted against m on a log-log plot (where $X^{(m)}$ is the m-aggregated time series by summing the original series X over non-overlapping blocks of size m); a straight line with a slope($-\beta$) greater than -1 is indicative of self-similarity. The Hurst parameter H is given by $H = 1 - \beta/2$.

The second method, the global scaling plot, is a wavelet-based analysis [22] that uses wavelet transform of a time series to study its global scaling property, by which we mean the statistics of the time series viewed at each resolution level or scale, taken as a function of scale. To determine the global scaling property of data, we plot log(E_j), where E_j is the average energy at scale j, as a function of scale j. The energy level E_j is corresponding to the level of irregularity or burstiness of sampled data. The higher E_j is, the more bursty the traffic is on time scale j. By inspecting qualitatively over what range of scales there exists a linear relationship between log(E_j) and scale j, we can determine what range of time scales there exists self-similar scaling. [11] gives a more detailed description of this technique.

By applying the methods described above to our traces, the results indicate that RealAudio traffic does not have significant long-range dependence as web traffic does.

In the time-variance plot, as shown in Figure 3(a), we see some bumps appearing at time scale corresponding to $1s \sim 10s$, which we believe is related to the off-period of RealAudio traffic. At time scale larger than 10s, the curve appears almost as a flat line. Since there is no indication of a straight line with slope greater than -1 in the plot, we conclude RealAudio does not have self-similarity.

In the global scaling plot, [1] as shown in Figure 3(b), we do not see a linear relationship between log(E_j) and scale j. Instead, we observe some fluctuation of E_j at smaller time scales (12-17). We believe the cause of this phenomenon is due to that data sampling rate is smaller than the period of burst (1.8 sec) at small time scales. At larger time scales (4-8) it appears asymptotically as a flat line because, as described in Section 3.1, RealAudio flows behave like constant bit rate streams at larger time scale.

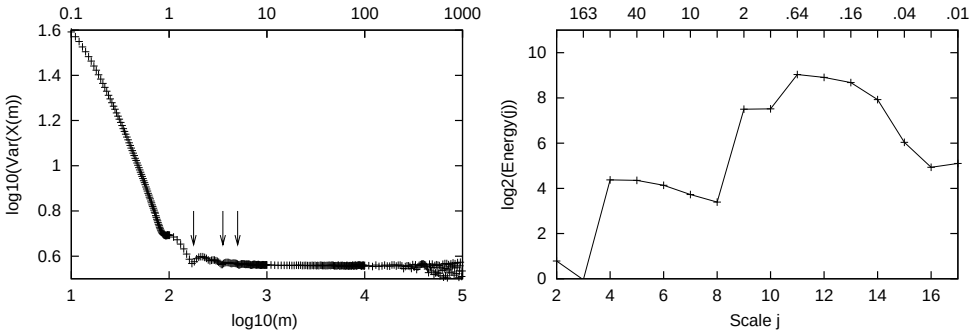
Although the result is not surprising, we will show later the multi-scale analysis we perform here is still very useful for our model debugging and validation.

3.4 Why is RealAudio bursty?

We next present several hypotheses as to why RealAudio exhibits different behavior at different time scales (i.e. it is bursty at small time scales and roughly constant bite rate at medium time scales). Several factors might affect burstiness, including data content, timeliness (stored or live playback), and playback-time systems factors such as CPU load. Our goal is to show which of these factors is correlated with burstiness in individual flows and synchronization of multiple flows in aggregate traffic. We also suggest underlying reasons that may cause these problems. Because the source code to RealAudio server is not available, we verify these hypotheses with experiments on a RealAudio server with eight concurrent clients as described in Section 2.

A single RealAudio flow is sent at roughly constant bit rates at *medium* and *large* time scales (Figure 1(a)). We believe this is because that streaming media protocols inherently target steady rate due to encoder and network issues. Some continuous traffic is inherent in streaming media by definition—streaming media sends data incrementally and gradually. Many encoders also target fixed bit rate because some networks such as telephone and ISDN only provide small, fixed bandwidth channel. VBR codecs are less desirable for fixed bandwidth links because they are generally more complex.

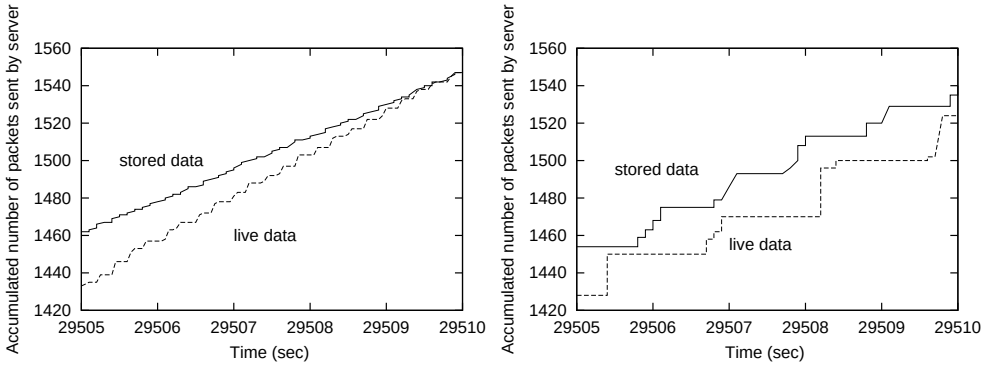
[1] Non-stationarity can invalidate this approach to identifying self-similarity. To avoid this problem this analysis is based on a 3000s portion of the trace where it is stationary. Here we define a time series is stationary if the mean and variance of data remains relatively constant, verified with the run test [23].



(a) Time-Variance Plot

(b) Global Scaling Plot

Figure 3: Multi-scale analysis of trace 3



(a) low CPU load

(b) high CPU load

Figure 4: Aggregate traffic from 8 flows of live and stored data

Figure 1(b) shows that RealAudio is bursty at *small* time scales. This behavior can be plausibly explained by operating systems issues. Operating systems will schedule the process or thread serving a flow periodically. The frequency at which this scheduling required is a function of packet size and target rate. With 200B packets and 15Kb/s flow rates this corresponds to about 10 packets/s. At low CPU loads, this sending rate can easily be achieved since CPU time can be smoothly allocated, but at higher loads or higher rates such frequent context switches will become impossible.

To evaluate the overhead of context switch on real system, we ran *lmbench*[24] on a Pentium II 266MHz Linux box (which is similar to that used by Broadcast.com). For a process with size of 1MB, the context switch time will take about 3ms. In other words, with 200B packets and 15Kb/s flow rates, a server of size 1MB can only support 33 flows concurrently if it does context switch for every packet. This context switch time is not just an artifact of the operating system's process or thread model (in fact, Linux has one of the better context switch times [24]). The cost of a context switch is inherent in supporting multiple processes with separate CPU resources (registers, memory protection, etc.).

Although there must be a context switch cost, this cost does not need to be incurred on every packet exchange. A simple solution is to send multiple packets each time when the thread is scheduled, possibly dynamically adjusting the number of packets sent to the frequency of context switches. To verify this hypothesis, we examined burstiness in a single flow as a function of CPU load. At low CPU loads we do not observe much burstiness (the average burst length 1 packet, the average inter-burst time 0.23 second). Here we define packets in the same burst as those which have inter-arrival time less than 1ms.

When we artificially increase server CPU load by running four concurrent SETIathome[2] programs [25], we see burstiness similar to that observed in Figure 1(b) (the average burst length 5 packets, the average inter-burst time 1.14 second). This burstiness appears independent of content or timeliness.

Flow synchronization magnifies the small-time-scale burstiness of individual flows in *aggregate* traffic. Two factors are correlated with flow synchronization: timeliness (playback of stored or live content) and CPU load. Flows of live content show a large amount of synchronization. We believe this synchronization occurs because multiple clients are listening to the same live event. When the server receives new data, it immediately sends copies of this data to each client. This synchronization is less likely when replaying stored content because each client would be at different places in the data stream. This reasoning is consistent with the live content of the broadcast.com traces. To verify this hypothesis we examined eight concurrent clients listening to stored and live data at low CPU load (Figure 4(a)). The additional burstiness of the live data appears as much larger stair-steps than replay of stored content. Quantitatively, the live flow has the average burst length 9 packets and the average inter-burst time 0.25 second, while the stored data has the average burst length 2 packets and the average inter-burst time 0.12 second.

A secondary factor affecting flow burstiness is CPU load. Repeating this experiment at high CPU load shows burstiness for both stored and live content (Figure 4(b)). Synchronization of live content occurs for the same reasons just described. In addition, stored content is bursty because individual flows are bursty caused by the operating system scheduler. The magnitude of burstiness for live event is seen to be several times higher than that of stored content (the live flow has the average burst length 28 packets and the average inter-burst time 1.05 second, while the stored data has the average burst length 7 packets and the average inter-burst time 0.49 second), which is possibly due to the added synchronization among different streams.

4 Modeling RealAudio

We have analyzed RealAudio traffic (Section 3), but to really understand how this traffic affects the network we would like to be able to reproduce RealAudio-like traffic in a network simulator where it can be part of controlled experiments of RealAudio. This section describes how we created and validated a simulator model of this traffic.

We chose to model this traffic in ns-2 [26, 27] because of our familiarity with it and its support for a wide range of protocols, thus allowing us to evaluate audio traffic in competition with other traffic.

4.1 A structural model of RealAudio

Based on the description of RealAudio traffic from Section 3, we designed a three-level simulation model to characterize RealAudio traffic as shown in Table 2. Two aspects of the model are unusual: first, flows are artificially synchronized by delaying the start of each ON-period until a multiple of 1.8 seconds. Second, the number of packets sent in each ON-period is calculated based on the randomly selected duration of the OFF-period to match the flow rate.

We expected that the particular mechanisms described above differ from a real implementation of RealAudio. For example, in a real server, all these parameters might be dynamically changing because of user selection, feedback control [28] and reaction to the congestion. The arrival of

[2]The reason we choose SETIathome for driving up the load is because it is by nature computation-intensive. To avoid the daemons de-prioritize themselves, we invoke them with the lowest **nice** value

User behavior

1. User arrival is modeled as a Poisson process.
2. The number of flows per user is randomly picked from the CDF(Cumulative Distribution Function) of trace.
3. Each user flow is sequentially generated as described below.

Flow data

1. Flow duration is chosen from a CDF
2. We chose a fixed duration for the ON period T_{on}
3. We chose a fixed rate R_f for each flow based on the CDF of flow rate from the trace
4. We chose a fixed length P_{size} for every packet of the flow
5. To reproduce the periodic burst of aggregated traffic as shown in Figure 2(a), flows are artificially synchronized by delaying the sending of every burst in each individual flow until the time is a multiple of 1.8 seconds.

Packet data

1. Each packet is sent as part of a UDP flow
2. Packets are repeatedly generated until flow stops
3. The duration of OFF period T_{off} is also randomly chosen from a CDF
4. The number of packets being sent during each ON/OFF period for the flow is $R_f \times (T_{on} + T_{off})/P_{size}$

Table 2: Structural model of RealAudio

users and how long the sessions will last might depend on the content that server provides. However, we will show that these parameters can correctly reproduce traffic corresponding to our traces. We also expect that they can match other classes of RealAudio traffic, although the particular parameters used may require change.

4.2 Validating the Model

To validate if our model accurately captures key components of RealAudio protocol, we compared the traffic generated from our model and real trace for both individual flows and aggregated traffic and examined if they are similar to some extents.

To evaluate individual flows we look at time-sequence number plots at two time-scales. Although the graphs are not shown here due to space limitation, we found the flows generated from our model exhibit the characteristics of RealAudio at different time scale (i.e. it appears as constant rate traffic at medium time scales, and behaves like a burst on-off source at small time scales. We also found that the CDFs of packet inter-arrival times for the model and trace are basically identical.

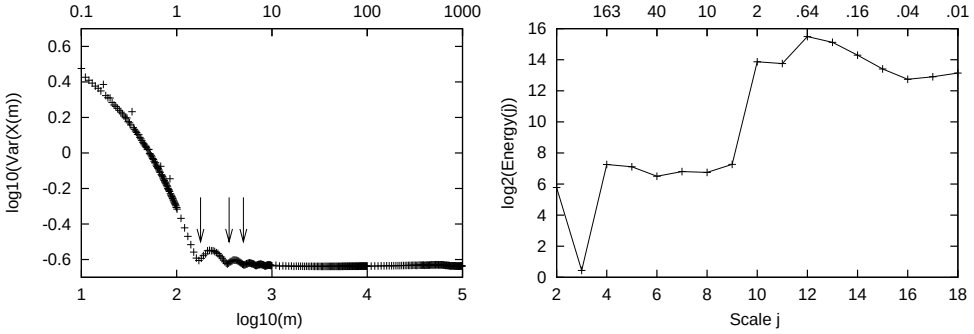
For aggregated traffic, we compare our model with real trace both qualitatively and quantitatively. First, we consider first order statistical comparisons of our model to the trace data. We examined CDF plots of the packet inter-arrival times, flow rates, and user durations. In all cases the model and trace traffic matched within a few percent. This is not surprising given that the models are driven off of these statistics as taken from the trace data. We then compare two multi-scale plots, as described in Section 3.3, between our model and trace. They also matched closely.

The two multi-scale plots generated from our model are shown in Figure 5. First we consider the time-variance plot, as shown in Figure 5(a). Comparing Figure 5(a) to Figure 3(a), we see both exhibit similar bumps at the time scale around 1.8s, 3.6s, corresponding to the duration of flow off-periods. We can see, from time 0.1 to 1s, both show bigger variance since the traffic is more bursty at small time scales. From 10s to 1000s, both show small (and almost constant) variance which is probably because aggregated flows behave as constant rate traffic at large time scales.

Second, we examine the global scaling plot generated from our model. Comparing Figure 5(b) to Figure 3(b), we see that they both have similar overall shape with lower energy at coarser time-scales and higher energy at finer scales. One feature of this plot are particularly relevant in comparison: the dividing point between these areas where energy sharply rises as time scales become finer (at scale 10 in the model and scale 9 in the trace, both at 2s). We believe this sudden increase in energy corresponds to the duration of off-period of the traffic. To validate this hypothesis we experimentally increased this period to 18 second in our model, seeing a corresponding shift in this turning point to coarser time scales, as shown in Figure 6.

Finally, we compare five metrics quantitatively between trace and model as shown in Table 3. Again, it's not surprising that our model matches the trace closely, given the fact that the models are driven off of the statistics taken from the trace data. Nevertheless, this still provides us confidence that our structural model does capture the behavior of RealAudio protocol to some extent.

The comparisons here are between our model and the data from trace 3 from [2]. Although not presented here, we found a similar level of consistency with trace data sets 4 and 5 from [2]. We later evaluate the sensitivity of these results to newer version of RealAudio in Section 6.



(a) Time-Variance Plot

(b) Global Scaling Plot

Figure 5: Multi-scale analysis of model

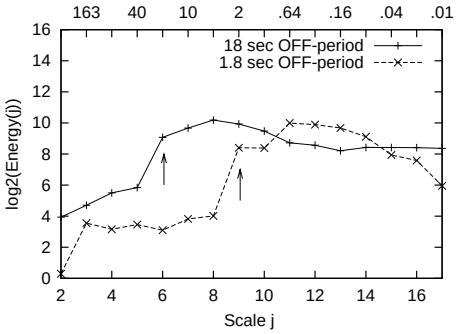


Figure 6: Global scaling plots for models with different OFF-periods

metrics	trace	model
Mean user duration (min)	63.9	64.4
Mean user inter-arrival (sec)	5.67	5.45
Mean flow rate (Kbps)	4.2	3.8
Mean flow duration (min)	58.1	59.0
Mean packet inter-arrival time (sec)	0.0047	0.0051

Table 3: Comparison of first-order statistics for trace and model.

4.3 Observation about the model and validation

The model we just described was developed through a process of trial and error. Although much of the process is typical of interpreting new data, we next highlight two aspects of this process that we found to be very helpful and perhaps deserve to be more widely used. First we consider the use of structural modeling, second, multi-scale analysis.

Structural modeling, as discussed in [10, 29], proposes that we should implicitly take into account the complex hierarchical structure of application and intertwined networking mechanisms in order to accurately reproduce the traffic. Inspired by their work, we structured our model with a clear user-, flow-, and packet-level components, as opposed to trace-replay. There are several advantages for this approach:

- Some protocols must be modeled as end-to-end entities in order to capture the feedback effect such as TCP congestion control
- Internet protocols present very rich, multi-fractal behavior across a range of time scales. Network-level approach will fail to capture this richness.
- By capturing the details of data transfer in an algorithm we can reproduce that traffic with much less storage requirements than trace-replay.

Without referencing RealServer source code, this application-level approach also gives us an opportunity to develop and apply multi-scale modeling techniques, since we can not *cheat* to obtain an accurate model. On the other hand, more insights in the server code might give us deeper understanding of protocol behavior.

Second, we stress the importance of utilizing multi-scale analysis in model validation. Our early models quickly reproduced similar first-order statistics such as flow rates, inter-arrival distributions, and other per-user and per-flow statistics. But it is the interpretation and comparison of the time-variance and global scaling plots detected several errors in our first models. For example, we initially did not synchronize different flows as described in Section 3.2. The time-variance plot of flows without this synchronization is shown in Figure 7. Comparing this figure to Figure 5(a), although basic shapes look similar, the details (1.8s bumps) are all missing. This demonstrates that multi-scale analysis is useful not only to identify structural properties of traffic, but also to serve as a debugging tool when constructing the

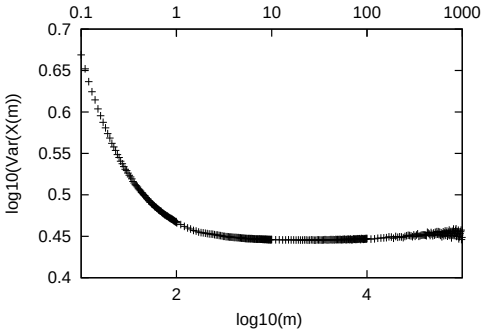


Figure 7: Time-variance plot for the unsynchronized model.

model.

5 Protocol improvements

In this section we will demonstrate how can we improve performance of RealAudio by changing some key components of how data is sent. The performance metric we use here is the loss rate of traffic. We have observed that RealAudio synchronizes separate flows of live content, and that stored content becomes bursty at high CPU load. This behavior increases the burstiness of traffic, stressing router buffering capabilities. We suggest three approaches to reduce this loss rate:

1. We change the model by removing the explicit synchronization of flows as described in section 4 and modeling flow arrival as a purely Poisson distribution. In an implementation of RealAudio, this change would correspond to adjusting the operating system scheduler and server code, or employing faster CPUs, to ensure that CPU time is smoothly allocated at high load.

2. Instead of completely de-synchronizing the flows, we add a small gap between the start time of ON-period in different flows to reduce the synchronization of aggregated traffic. In the RealAudio implementation this would correspond to adding a small explicit delay to the flow servicing procedures.

3. As an alternative of making changes to the server, one could evaluate the buffering requirements of an unmodified server and size buffering in nearby routers appropriately.

To evaluate the effects of these variations on loss rates and buffer requirements, we simulate them using a simple network topology, as shown in Figure 8, where a number of clients listening the same live event broadcast from one single server via the same router. The bottleneck lies between the intermediate routers which implement RED as their queuing discipline. Each router has a buffer size of 10 packets. Through the simulation, the average network utilization is about 54%.

Implementation	Buffer requirement Mean(KB)	Max(KB)
Unmodified RealAudio (0ms gap)	6.98	74.17
Poisson arrival (Variation 1)	1.04	9.95
5ms gap (Variation 2)	1.91	13.21

Table 4: Comparison of queuing requirement for unmodified RealAudio and the variants

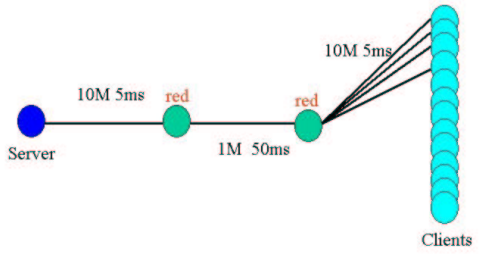


Figure 8: A simple network topology

Table 4 shows mean and peak queuing requirements of server's first-hop router for an unmodified server and our two variants, for the simple topology (with 100 clients) described above. The results show the router will need a much bigger buffer for an unmodified server than our variants.

As shown by comparing the line *5ms gap* against *0ms gap* (unmodified RealAudio) in Figure 9, we are able to reduce the loss rate by half by adding a small 5ms gap between individual flows. We also show the constant bit rate traffic as a lower bound for comparison. (Although CBR provides much lower loss rates, the CPU overhead of very frequent interrupts makes it impractical for actual implementation.) If we completely remove the synchronization factor in our model, the performance of aggregated traffic would be almost as good as constant bit rate traffic.

Ideally the artificial gap introduced in variation 2 should be sized to allow the queue to drain one burst worth of packets. We can analytically compute the required gap to drain a single flow as

$$gap = (n * p)/b \quad (1)$$

where n is the average number of packets sent in the ON-period, p is the packet size and b is the minimum link band-

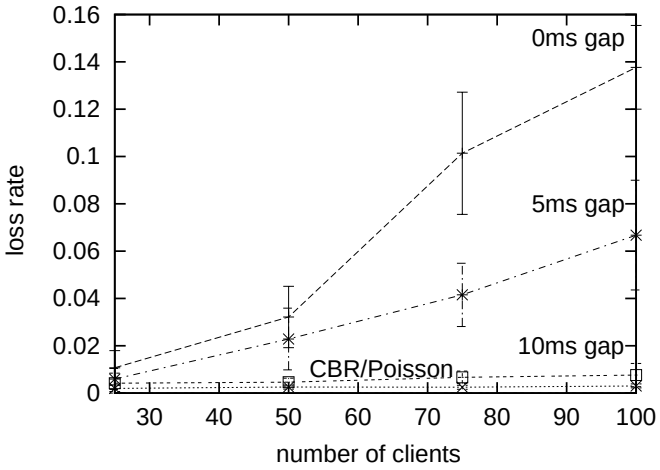


Figure 9: Different approaches to improve performance of RealAudio protocol

width of first few hops from the server.[3] Using the above formula, the ideal gap is 8ms for our sample network. Figure 9 shows the effects of varying this gap. With a 10ms gap RealAudio performs as well as CBR. The cost of this approach is that clients might suffer some latency delay. However, we expect a small delay like 5ms will not affect end-user latency significantly since the latency of streamed audio is already high due to the delay introduced by the buffering at the client side.

Another solution to reduce the burstiness of traffic is through the use of multicast. For live flows multicast should eliminate synchronization by replacing transmissions of the same data to multiple clients with a single transmission to the multicast group. Techniques such as stream merging [30] and caching [1] may be helpful for stored content. Although promising and standardized, multicast is not yet widely available in commercial ISPs today despite the fact it was already supported in RealAudio software since 1997.

Finally, we observe that this type of protocol study is very easy with simulation model, while experimentally it would be much more difficult.

6 Sensitivity of results

We have analyzed RealAudio traces and demonstrated that our model can correctly reproduce them. Since these results are based on a particular set of traces, we next examine how they generalize to other versions of real audio and other types of traffic.

Our traces are based on a currently old version of the RealAudio protocol, PNA, and from a heavily loaded server. Changing either of these characteristics will change the traffic. We took a week long trace from USC ICT's RealAudio G2 server. In that trace we did not observe bursty traffic, although we did observe fixed packet size, long flow durations, and constant bit rates at medium-time scales. This server was very lightly loaded (274 flows over the week) and had no concurrent flows. Current server software is not very bursty at low CPU loads we expect that we would see burstiness in a busier server.

To understand if these differences were because of server load or protocol version we set up a real audio server and artificially increased server CPU as described in Section 3.4. We found the existence of burstiness, but with a period about 1.1 seconds. We therefore conclude that our observation that burstiness is caused by CPU load holds for both old and current RealAudio server versions, although with different periodicities. Another factor that contributes to the periodic nature of traffic is the synchronization of multiple flows listening to the same live event. As shown in Figure 4(a), the traffic is bursty (at a relatively smaller scale) even at low CPU load.

We also have begun examining to what extent our conclusions are affected by traffic *content*. We strongly suspect that gross characteristics such as flow duration are affected heavily by content. For example, a server supplying 30-second music clips would have a very different mean flow duration from a server providing full classical music symphonies or from one providing 24-hour content from a live radio station. As described in Section 3.4, live data and stored content will behave differently in the aggregated traffic, and eventually determine how we structure the model.

7 Related Work

Some of the key parameters we use to construct the model are directly derived from the results of a previous study of RealAudio [2]. However, one of our goals in this study is to understand the aggregate behavior of streaming multimedia traffic across a range of time scales. Unlike previous studies for Ethernet [21] and web traffic [17], which have been found to be self-similar at large time scales and to have a much richer behavior at small time scales, our analysis suggests that RealAudio streams exhibit regular behavior across different time scales, which is more like constant bit rate traffic in general. Moreover, we are surprised to discover the burstiness of RealAudio not only exists in individual flow, but also appears in aggregated traffic. Additionally, we discuss several solutions to reduce this burstiness.

[3] More precisely we should identify the bottleneck link that will have the most queuing due to synchronized traffic from the server. Distant links are unlikely to be queuing bottlenecks (even if they are very low bandwidth) if the clients are dispersed throughout the Internet. For the common case of server connected to a LAN and then over a bandwidth-limited to the ISP, that link is likely the bottleneck.

Chesire et al. [1] analyzed streaming-media traffic using traces collected from the border routers serving the University of Washington. The focus of their work was to characterize a client-based streaming-media workload and compare it to well-studied web workloads in terms of bandwidth utilization, server and object popularity, session statistics, and sharing pattern. They also studied the effectiveness of caching and multicast for reducing streaming media bandwidth. In contrast to our analysis, most of their streaming sessions were short-lived. We attribute this difference to the fact that they studied client-based trace of both audio and video streams to a large number of Internet servers, while we study a server-based audio traces from a single site with live content. Also, our protocol-level optimizations are designed to reduce loss rates while theirs are focused on reducing bandwidth requirements.

A large body of Internet traffic capture and analysis software has been developed over the years. Among them, mmdump [28] was designed for examination of multimedia traffic growth and characteristics. Multimedia applications typically use dynamically assigned UDP ports for exchanging media data. These ports are negotiated using a control protocol such as RTSP [31]. Mmdump extends the popular tcpdump utility and employs protocol-specific parsers that allow it to determine the dynamic port number that are selected for media transport by streaming media and other session control protocols. Currently it supports RTSP and H.323. Although their work focused on building the multimedia monitoring tool, they also reported preliminary results from traces collected from AT&T WorldNet IP network. Their observations such as packet length distribution and the time of day usage pattern are consistent with ours. Their techniques for examining and tracking individual flows are more sophisticated than ours, but they do not apply multi-scale analysis to their analysis of aggregate data.

There are several similar studies for modeling web traffic based on empirical measurement. SURGE [8] described web workload based on analytical models of web use, which has the advantage of being compact and perhaps easy to manipulate. While in Mah's work [9], he simulated traffic based on CDFs of real data rather than mathematical model, which has the advantage of being able to represent arbitrary distribution. Based on the same reason, we follow Mah's approach to construct our model.

The work presented in this paper complements earlier work in modeling web traffic and is the first attempt at developing a structural model for multimedia traffic, which is a significant component of Internet traffic these days.

8 Future Work

How to identify streaming traffic such as RealAudio among different types of traffic is an interesting issue, particularly if we want to detect non-TCP friendly flows. [2] suggests that regularity in packet size and bit-rate of RealAudio might be able to partially serve this purpose. Another alternative which might be also useful to detect different types of flows is to examine and compare their aggregated behavior across a range of time scales.

In Figure 1(a), we can see some evidence of congestion/feedback control. (For example, the data rate seems to slightly change between 34000 and 34100.) Our current model does not include protocol feedback mechanism. Recent version realserver provides a sophisticated client/server mechanism called SureStream [32] for dynamically adjusting the bit-rate based on changing network condition. A detailed study of this mechanism is important to complete our model and to understand the behavior of aggregate traffic. Additionally, it would help us to understand if RealAudio congestion control is compliant to TCP-friendly [33], and how it compares to other proposed congestion control mechanisms such as CM [34] and RAP [35].

Feldmann et al. [11] introduced another wavelet-based technique called local scaling analysis in order to gather information about the local features (e.g. burst of packets) of the traffic. We would like to apply this technique to the traces[4] and models in the future to understand if RealAudio shows multi-fractal behavior at small time scales.

Frequently the best available traffic models are years old because current approaches to collect traces, analyze the data, and implement model take far too long. We plan to develop tools and approaches supporting rapid model generation, by combining parameterized model and live network measurements. Challenges we are beginning to study include which parameters to control and how to integrate network measurements from multiple sources.

Finally, we would like to understand how to apply the methodology developed in this paper to other types of traffic (such as Microsoft Media player and IP telephony etc.) to generate compact application-level models that are accurate across a wide range of time scales.

9 Conclusions

This paper has presented a systematic way of modeling RealAudio traffic via careful analysis of real world traces. We showed some key characteristics of RealAudio in terms of individual flow and aggregated traffic, and demonstrated how we use them to construct an structural model for simulation. We used multi-scaling analysis to show that Real-

[4] To be able to study multi-fractal behavior of RealAudio, it will require us to further collect new set of traces which contain higher bit-rate traffic

Audio is not self-similar, but instead exhibits more complex structure. These tools were crucial to reproduce this structure in our simulation model. Finally, we showed via simulation that the loss rate of traffic can be significantly reduced by slightly changing the behavior of RealAudio server.

Acknowledgements

We would like to acknowledge the many contributions of Walter Willinger and Polly Huang to this work. Their contributions include software for producing global scaling plots and more importantly several detailed technical conversations about early stages of this work and interpreting scaling analysis. For collection of RealAudio traces we would like to thank Henry Heflich, Gary Nelson, and Ed Luczycki of broadcast.com, and Jim Pepin, Michael Vincenc of USC. A preliminary version of this work was presented at the UC Berkeley Multimedia Seminar in October 2000; we would like to thank Larry Rowe and his class for helpful discussions, particularly concerning flow synchronization.

References

[1] Maureen Chesire, Alec Wolman, Geoffrey M. Voelker, , and Henry M. Levy, "Measurement and analysis of a streaming-media workload," in *Proceedings of the USENIX Symposium on Internet Technologies and System, p. to appear.*, San Francisco, CA, USA, Mar. 2001, USENIX.

[2] Art Mena and John Heidemann, "An empirical study of real audio traffic," in *Proceedings of the IEEE Infocom*, Tel-Aviv, Israel, Mar. 2000, USC/Information Sciences Institute, pp. 101–110, IEEE.

[3] Rahul Garg, "Characterization of video traffic," *TR-95-007, International Computer Science Institute, Berkeley, CA*, Jan. 1995.

[4] M. Garrett and W. Willinger, "Analysis, modeling and generation of self-similar VBR video traffic," *Proc. ACM Sigcomm*, pp. 269–280, Sept. 1994.

[5] Marwan Krunz and Armand Makowski, "Source model for VBR video traffic based on m/g/infinity input processes," *in Proceedings of the IEEE INFOCOM '98 Conference*, pp. 1441–1449, Apr. 1998.

[6] Marwan Krunz and Satish K. Tripathi, "On the characterization of VBR MPEG streams," *Proceedings of ACM SIGMETRICS'97 Conference, Vol. 25, No. 1*, pp. 192–202, 1997.

[7] D. Wrege and J. Liebeherr, "Video traffic characterization for multimedia networks with a deterministic service," *In Proceedings of INFOCOM*, Mar. 1996.

[8] Paul Barford and Mark Crovella, "Generating representative web workloads for network and server performance evaluation," in *Proceedings of the ACM SIGMETRICS*, Madison, WI, USA, June 1998, pp. 151–160, ACM.

[9] B. Mah, "An empirical model of HTTP network traffic," in *Proceedings of the IEEE Infocom*, Kobe, Japan, Apr. 1997, pp. 592–600, IEEE.

[10] Anja Feldmann, A.C. Gilbert, Walter Willinger, and T.G. Kurtz, "The changing nature of network traffic: Scaling phenomena," *ACM Computer Communication Review*, vol. 28, no. 2, pp. 5–29, Apr. 1998.

[11] Anja Feldmann, Anna C. Gilbert, Polly Huang, and Walter Willinger, "Dynamics of IP traffic: A study of the role of variability and the impact of control," in *Proceedings of the ACM SIGCOMM*, Cambridge, MA, USA, Aug. 1999, pp. 301–313, ACM.

[12] Sean McCreary and K. Claffy, "Trends in wide area ip traffic patterns: A view from ames internet exchange," *13th ITC Specialist Seminar*, pp. 1–11, Sept. 2000.

[13] Broadcast.com, ," `http://www.broadcast.com`.

[14] RealNetworks., "Realnetworks documentation library," `http://service.real.com/help/library/`.

[15] RealNetworks., "Realnetworks realserver documentation," `http://service.real.com/help/library/servers.html/`.

[16] Sally Floyd and Van Jacobson, "The synchronization of periodic routing messages," *IEEE/ACM Transactions on Networking*, vol. 2, no. 2, pp. 122–136, 1994.

[17] Mark E. Crovella and Azer Bestavros, "Self-similarity in world wide web traffic: evidence and possible causes," in *Proceedings of the ACM SIGMETRICS*, Philadelphia, Pennsylvania, May 1996, pp. 160–169, ACM.

[18] Andras Veres and Miklos Boda, "The chaotic nature of tcp congestion control," in *Proceedings of the IEEE Infocom*, 2000, pp. 1715–1723.

[19] K. Park, G. Kim, and M. Crovella, "On the relation between file sizes, transport protocols, and self-similar network traffic," *Proc. IEEE Int'l. Conf. Network Protocols*, pp. 171–180, Oct. 1996.

[20] Jon M. Peha, "Retransmission mechanisms and self-similar traffic models," *IEEE/ACM/SCS Communications Networks and Distributed Systems Modeling and Simulation Conference*, pp. 47–52, 1997.

[21] Leland, W.E.; Taqqu, M.S.; Willinger, and D.V. W.; Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *ACM/IEEE Transactions on Networking*, vol. 2, no. 1, pp. 1–15, Feb. 1994.

[22] P. Abry and D. Veitch, "Wavelet analysis of long-range-dependent traffic," *IEEE Transactions on Information Theory*, vol. 44, no. 1, pp. 2–15, 1998.

[23] J. S. Bendat and A. G. Piersol, "Random data: Analysis and measurement procedures," 1986.

[24] Larry McVoy and Carl Staelin, "lmbench: Portable tools for performance analysis," in *USENIX Conference Proceedings*. Jan. 1996, USENIX.

[25] SETI@home, ," `http://setiathome.ssl.berkeley.edu/`.

[26] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haobo Yu, "Advances in network simulation," *IEEE Computer*, vol. 33, no. 5, pp. 59–67, May 2000, Expanded version available as USC TR 99-702b at `http://www.isi.edu/~johnh/PAPERS/Bajaj99a.html`.

[27] VINT group, "UCB/LBNL/VINT network simulator—ns (version 2)," `http://www.isi.edu/nsnam/ns`.

[28] R. Cáceres, C. J. Sreenan, and J. E. van der Merwe, "mmdump—a tool for monitoring multimedia usage on the internet," *ACM Computer Communication Review*, Oct. 2000.

[29] W. Willinger, V. Paxson, and M. Taqqu, "Self-similarity and heavy-tails: Structural modeling of network traffic," *In A Practical Guide To Heavy Tails: Statistical Techniques and Applications, ISBN 0-8176-3951-9*, 1998.

[30] Derek Eager, Mary Vernon, and John Zahorjan, "Minimizing bandwidth requirements for on-demand data delivery," *Proc. 5th Int'l. Workshop on Multimedia Information Systems (MIS '99)*, pp. 21–23, Oct. 1999.

[31] R. Lanphier H. Schulzrinne, A. Rao, "Rfc 2326: Real time streaming protocol (rtsp)," April 1998.

[32] RealNetworks., "Realnetworks documentation: White papers," `http://http://www.realnetworks.com/devzone/documentation/wp_surestream.%html`.

[33] Sally Floyd and Kevin Fall, "Promoting the use of end-to-end congestion control in the internet," *ACM/IEEE Transactions on Networking*, vol. 7, no. 4, pp. 458–473, Aug. 1999.

[34] H. Balakrishnan, H. S. Rahul, and S. Seshan, "An integrated congestion management architecture for I nternet hosts," in *Proceedings of the ACM SIGCOMM*, Cambridge, MA, USA, Sept. 1999, pp. 175–188, ACM.

[35] Reza Rejaie, Mark Handley, and Deborah Estrin, "Rap: An end-to-end rate-based congestion control mechnaism for realtime streams in the internet," in *Proceedings of the IEEE Infocom*, New York, NY, USA, March 1999, IEEE.