Rapid generation of structural model from network measurement

Ph.D. Dissertation Proposal

Kun-Chan Lan
kclan@isi.edu
Department of Computer Science
University of Southern California

April 25, 2002

Guidance Committee

Prof. John Heideman (Chair)
Prof. Ramesh Govindan
Prof. Christos Papadopoulos
Prof. Ashish Goel
Prof. Ahmed Helmy (Outside Member)

Abstract

The utility of simulations and analysis heavily relies on a good model of network traffic. While network traffic constantly changing over time, existing approaches typically take years from collecting trace, analyzing the data to finally generate and implement models.

In this work, we describe approaches and tools that support rapid parameterization of traffic models from live network measurements. Rather than treating measured traffic as a time-series of statistics, we utilize the traces to estimate end-user behavior and network conditions to generate application-level simulation models. We also show multi-scaling analytic techniques are helpful for debugging and validating the model.

To demonstrate our approaches, we develop a structural model of UDP-based RealAudio traffic and evaluate its accuracy via simulation. We then extend the same methodology by automating and integrating the process from collecting trace to ultimately generating structural simulation models for web and FTP traffic.

Finally, to get a complete picture of network-wide view of traffic, topology and path characteristics, we plan to explore approaches to correlate and integrate measurements from multiple points in the network, and develop tools that automatically collect and integrate distributed data into a coherent traffic model.

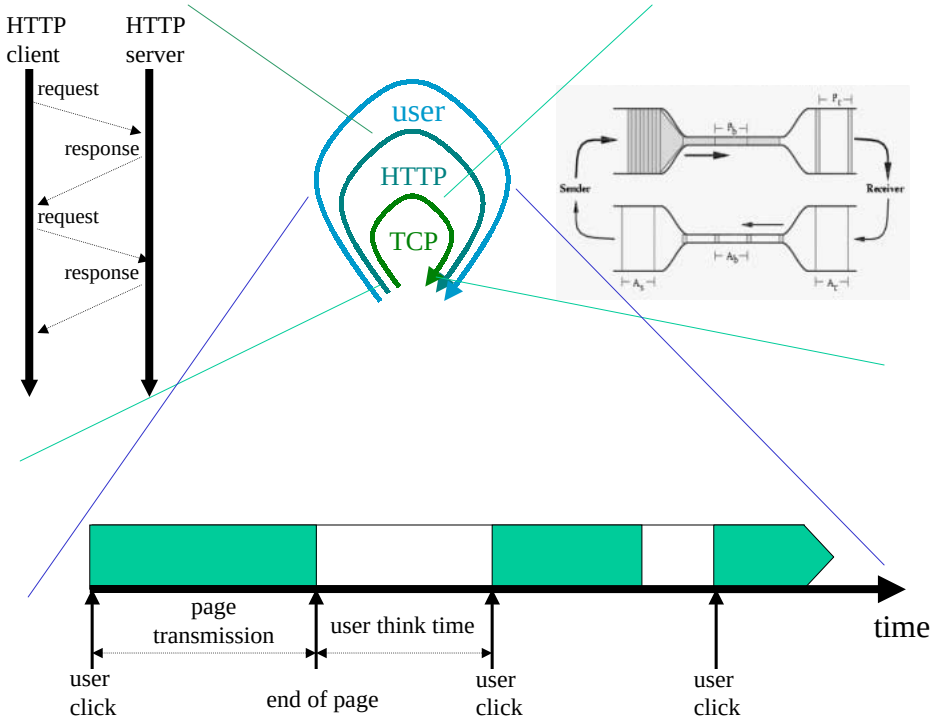


Figure 1: Multiple levels of feedback in web traffic

1 Introduction

Simulations are important for exploring and understanding the complexity of network. However, it is difficult to simulate and model the Internet due to its scale, heterogeneity and dynamics [27]. Internet traffic is constantly changing over time both in volume and statistical properties, even observed at the same location. For example, Kim claffy et al. [55] showed the volume of online game traffic is increasing over the years in the backbone traffic. Recently Campos et al. observed that HTTP requests are becoming larger over time while HTTP responses show a trend toward smaller size when comparing with their previous data collected two years ago. Another recent work [86] showed that, depending on the particular aspect of constancy and the dataset under consideration, the constancy of Internet path properties will start to break at the time scale of hours.

Even we fix our interest to a single point of time, the traffic still look different at different places due to the immense heterogeneity of the Internet: the diversity of topology and link properties, different protocol usage and user populations in different networks. For example, the traffic at different websites might be different due to their content differences. The distribution of flie size in a trace distribution site like Internet Traffic Archive [68] tends to be less heavy-tailed and more like bimodal, where small files account for web pages that describe traces and large files for traces themselves. Cao et al. [13] showed that, due to the lower link utilization and higher degree of multiplexing, the traffic in backbone links tends to have higher nonstationarity than traffic in the access links.

In fact, even when we only look at one particular part of the network at a single point of time, network traffic can still show great variations just in terms of direction of flows. For example, inbound traffic and outbound traffic seen at the ingress or egress points of the network typically differs from each other due to reasons like different

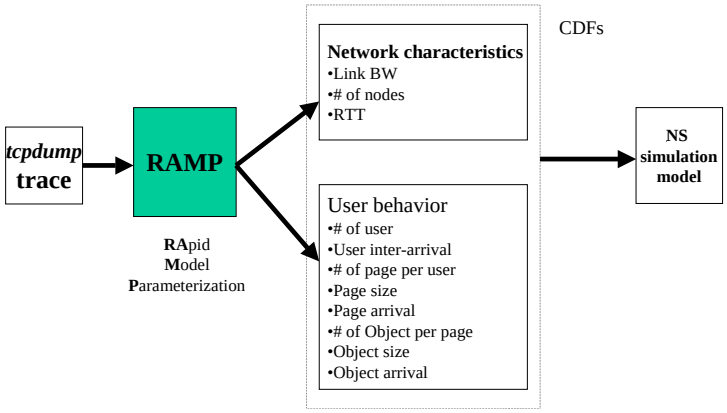


Figure 2: Rapid model parameterization from measurement

user populations, different application usages and different paths properties etc.

Rapid and unpredictable change of traffic will threaten to make the research obsolete before it is finished. Some assumptions about traffic mix, topology or protocols might only be valid for less than a few years. However, take today's most widely-used web models as an example, it still takes years from collecting traces, analyzing the data to finally generating and implementing models [4, 50]. Furthermore, the existing models are all based on a small set of traces collected from one particular part of the network within some particular time period. Considering the Internet's great technical and administrative diversity and immense variations over time regarding how applications are used, it is not obvious that one can model *his* traffic accurately based on the models derived from measurements taken previously from other parts of the network. Hence, there is strong need for approaches that are able to rapidly generate realistic traffic model in a constantly changing distributed network environment like Internet.

To generate realistic traffic model, we promote structural modeling, as first discussed in [23, 82]. Opposed to traditional time series analysis approaches which typically ignore the effect of network dynamics (such as the fact that traffic is frequently shaped by the network's properties), structural modeling approach emphasizes on how to characterize source-level pattern in which the data is sent and exploits the physical mechanisms that exist in the networking context. It is in sharp contrast to the traditional black-box modeling methodology based on packet-level time series analysis that generally ignores underlying physical structure. Take web traffic for example, as shown in Figure 1, we can see there are multiple levels of feedback effect within the hierarchical structure of web traffic, and each level operates at different time scales. For instance, TCP has its own congestion control mechanism which operates at the time scale of seconds, while HTTP has the request-response loop functioning at the time scale of tens of seconds. Hence, it is important to reproduce the structure of application in the model in order to accurately reproduce the traffic. To demonstrate this approach, we examine RealAudio traffic as a case study. We identify the structural properties of RealAudio traffic, develop an application-level model and validate the model using multi-scale analytic techniques. We show the output of resulting simulation model closely matches the original

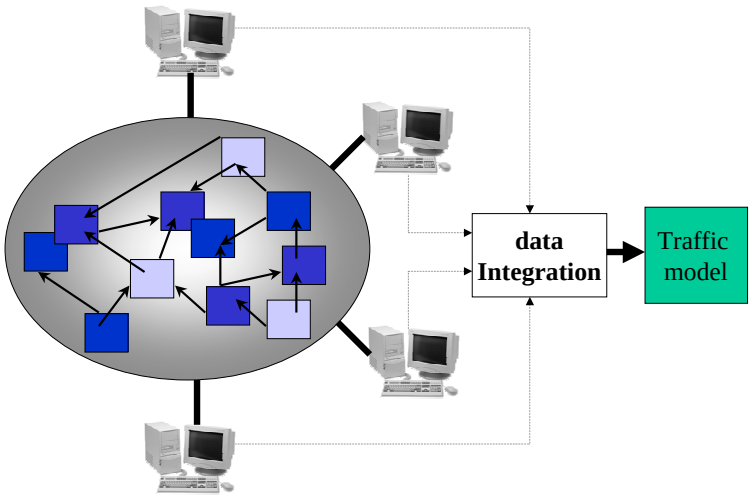


Figure 3: Integration of measurements from multiple points

trace.

To cope with the rapidly evolving nature of network traffic, based on this application-level modeling approach, we further develop tools and methodologies that automate and integrate the processes from trace analysis to model parameterization and validation, as shown in Figure 2. Our initial studies emphasize two types of dominant traffic in today's Internet, namely web and FTP traffic. We are able to generate a high quality simulation model from passive measurements in a *timely* fashion. In contrast to previous work, our approach does not make any implicit assumption of traffic properties (for example, heavy-tailed distribution for file size/transmission time in web traffic) and hence is more applicable in coping with the heterogeneity of Internet traffic.

Finally, Distributed measurements is required to get a network-wide view of traffic while keeping the size of collected data maintainable, as shown in Figure 3. Furthermore, distributed measurements will allow us to explore data-level parallelism. For example, instead of having a special "powerful' hardware placed at the center of network "continuously" monitoring the traffic, one can obtain the same data statistics via random sampling performed by a few off-the-shelf PCs distributed across the network. Accurate characterization of network-wide traffic dynamics will help various applications such as caching placement and content distribution. However, to the best of our knowledge there is no previous work being done in this area. To integrate distributed data together will require approaches for overlap detection and hole filling. In other words, it will demand methodology to merge measurements from multiple points and ways to infer traffic statistics at places where the measurements is not available. To address this problem, we plan to explore new algorithms and tools to merge distributed data into a coherent traffic model.

In summary, we have identified the following specific issues to *rapidly generate realistic traffic model in a constantly changing distributed network environment like Internet.* We will address them in detail in the rest of this paper.

- ***Structural Modeling***: We have demonstrated the importance of using structural modeling to characterize

traffic and developed a structural model for RealAudio traffic, evaluated through simulation and validated via multi-scale analysis.

- ***Rapid Model Parameterization***: We have showed Internet traffic not only changes over time but also exhibits great variations in other dimensions such as locations and flow directions. Based on structural modeling approach, we have developed methodology and tool that rapidly parameterizes traffic models from live measurement. Currently our tool supports web and FTP traffic.

- ***Integration of Distributed Measurement***: We will discuss our plan to explore algorithms and tool to integrate traffic measurements from multiple points.

2 Related Work

Research efforts related to our work fall into the following categories including traffic modeling, workload generation, multi-scale analysis techniques and distributed measurements. We will discuss them in the order below.

2.1 Traffic modeling

Traditional traffic modeling techniques typically treat the measurement as a time series, and try to capture the statistical characteristics (particularly autocorrelation and marginal distribution) of empirical data based on various approaches such as Markov models like MMPP (Markov Modulated Poisson Process) or MMDP (Markov Modulated Deterministic Process) [34, 49, 77, 64, 47] , autoregressive models like ARIMA (Auto Regressive Integrated Moving Average) or FARIMA (Fractal ARIMA) [49, 20, 16, 21, 35, 32, 5, 33], TES (Transform-expand-sample) [59, 58, 57, 54], FBM (Fractional Brownian Motion) [28, 65], FGN (Fractional Gaussian Noise) [66, 67, 43], wavelet-based models [48, 75] and others [19, 76, 2, 38, 45, 62, 40]. Although being able to correctly reproduce similar time series as the actual traffic, this type of curve-fitting approaches provides no or little insight about the observed characteristics of measured traffic and its underlying causes. Furthermore, it does not take into account the fact that traffic is usually shaped by the network dynamics.

Structural modeling approach has been first proposed in [44] and then studied in greater detail in [83, 82], where they showed the cause of self-similarity in aggregated LAN traffic can be explained via a simple application-level ON/OFF model. Subsequently, a number of attempts have been made at providing structural models for WAN traffic, including the same reduction to simple ON/OFF models for individual source-destination pairs in WEB traffic [17]. Furthermore, there are several studies that model traffic workload based on application-level statistics, including TELENT,FTP [18, 71], HTTP [50, 4], NNTP [37] and MPEG [39] etc.

We have developed models for RealAudio, Web and FTP traffic based on this structural modeling approach. The structure we choose to model web traffic is similar to previous work of Mah [50] and Crovella, et al [4, 17]. We also adopt Mah's approach in terms of describing web traffic based on CDF of real data, which has the advantage of being able to represent arbitrary distribution.

2.2 Workload generation

Research on Internet workload generation has typically focused on creating generative models based on packet traces of various applications. Several studies has adopted this approach to develop workload generators for web traffic, including SURGE [4],IPB [51] and work at RPI [85]. Their work focused on fitting statistics derived from a set of traces to some widely-used distributions which are then used to generate synthetic traffic workload. However, first, their approaches from collecting traces, analyzing the data, to finally generate and implement models take too long, (eg. in Crovella's study [4], it requires modification of browser codes in order to capture web-user's browsing

behavior) considering the network conditions are constantly changing. Considering Internet traffic is changing constantly, it is generally not applicable to characterize the current traffic simply based on statistics collected years ago from different parts of network. Second, even we assume the existence of some universal statistical property (eg. heavy-tail distribution of file size), parameterization is still a non-trivial job for the previous models which are fairly static. On the contrary, our approach is capable of parameterizing the traces and generating simulation models in a timely fashion that allows the users to study their *current* traffic. Additionally, except from modeling user/application behavior, our work also attempts to estimate path characteristics (namely, delay and bottleneck bandwidth) which are important parameters to drive simulation.

We have taken similar approaches as the previous studies of Feldmann et al. [22] and Smith et al. [80] in the sense that we also manage to reconstruct application-level statistics (eg. request/response) of web traffic on-the-fly from individual packets captured by the sniffer. However, in Feldmann's work, it requires special hardware and software to be able to extract full HTTP level information. The methodology we adopt to construct web model is closer to Smith's work where they reconstruct the data exchanges in the HTTP connections based on only the TCP/IP header information. Additionally, we also model path characteristics (hence the resulted models can be directly built into the widely-used ns network simulator [7]) and provide more comprehensive validation mechanism including a wavelet-based analysis. Furthermore, we include another dominant traffic, namely FTP traffic, in our study except from web traffic which previous work has focused on, and automate the whole process from trace analysis to finally the implementation and validation of the models.

2.3 Multi-scale analysis technique

The empirical finding of self-similarity in recent studies has motivated the need for analysis tools that are well-suited for identifying structures in network traffic. Among various different scaling analysis techniques, wavelet-based scaling analysis method is especially successful for its ability to summary statistics from scale to scale and probe the local structure of packet traces. Previous studies [1, 24, 74] has focused on the technical and quantitative aspects of these method and yielded some very interesting results. More recently, Gilbert [29] , Feldmann [25] and Huang [36] further exploited the qualitative aspects of this technique to detect "interesting" features or pattern in time and scale in the underlying packet trace and gain the insight of specific network path properties (eg. RTT).

Inspired by previous studies [25, 36], we utilize multi-scale analysis technique to debug and validate our model generation process. By comparing the scaling plot of the model against that of the trace we can evaluate the accuracy of our model by examining if the scaling structure of original traffic is reproduced in the model, which is impossible to achieve via commonly-used first-order statistics analysis by simple comparison of different CDF plots.

2.4 Distributed Measurement

We see our future work in distributed measurements is close in spirit to SCAN [30] project in which multiple measurement clients dynamically divide the network space to efficiently monitor and detect faults. However, their goal is to integrate network element (such as router) performance measurements taken from multiple points for network monitoring in a large-scale heterogeneous network like Internet, while our focus is on fusing data traffic to get a complete picture of network-wide view of traffic. In the context of large-scale wireless sensor network, recent work of Zhao [87] demonstrated an approach to extract the remaining energy distribution of the network by applying in-network aggregation of distributed data.

There are several research efforts that try to characterize internal network behavior based on end-to-end performance measurements, including MINI, IEPM, AMP, RIPE, Surveyor [9] and TReno [53] etc. They utilize either unicast probes (via tools such as *traceroute* and variants of *ping*) or multicast probes and correlate end-to-end traffic measurements collected at different monitoring machines across the Internet to infer statistical properties of the

network such loss, delay and topology. Another form of indirect measurements, also known as network tomography, emphasizes on estimating individual flow characteristics based on aggregated-flow measurements [81, 11, 12] taken at the end hosts. Their goal is to infer traffic matrices (the set of traffic between all pairs of sources and destinations) based on link bytes counts which are readily available through SNMP that is provided by most of the commercial routers. Our work is related these work in the sense we also try to infer internal network status (where the measurement is not available) based on the measurements taken at the edges of the network. We expect our future work can be either built on their results or as a valuable addition to these measurement infrastructures.

3 Structural Modeling

In this section we will first discuss the need of structural modeling approach to understand the physical structure of the traffic. We then demonstrate its application via a structural model of RealAudio traffic. We also introduce the multi-scale analysis for debugging and validating the model.

3.1 Why structural modeling

Traditional black box approaches focus on employing complex time-series analysis to model network traffic. Although being able to reproduce the measured traffic correctly, it completely ignores the underlying network structure and hence provide no or little insight about the observed characteristics of measured traffic and its underlying causes. On the other hand, structural modeling, first discussed by Willinger [82], 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 while still providing a physical explanation for observed phenomena.

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, while trace-replay techniques typically ignore the fact that traffic is frequently *shaped* by the network's current properties,
- Internet protocols present very rich, multi-fractal behavior across a range of time scales. Simple trace-replay 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.

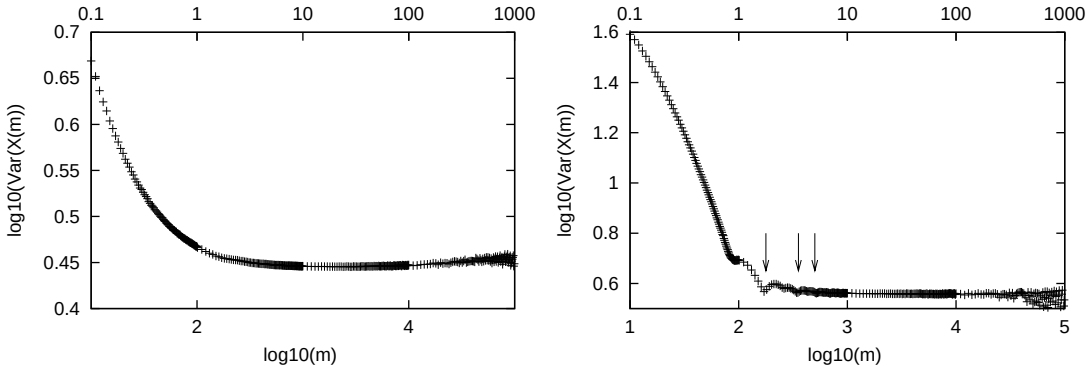
3.2 Multi-scale analysis

Scaling behavior appears to be an ubiquitous and inherent feature of data network traffic [44, 24]. Multi-scale analysis is important to capture the structures of network traffic that span a range of time scale. In our study, we focus on the qualitative use of this technique. In this section, we show two multi-scale analysis techniques that are used throughout our study, and an example that demonstrate how these techniques can detect model errors that might escape simple, first-order statistical analysis.

time-variance plot Let X = (X_t: t=0,1,2,....) be a stationary time series. We define $X^{(m)}$ = ($X_k^{(m)}$: k=1,2,3,...) by averaging the original series X over non-overlapping blocks of size m. That is, for each m=1,2,3,....,

$$X_k^{(m)} = 1/m(X_{km-m+1} + ... + X_{km}), k = 1, 2, 3, ... \quad (1)$$

We obtain the *time-variance plot* by plotting the variance of $X^{(m)}$ against m at a log-log scale. A straight line with a slope($-\beta$) greater than -1 is often indicative of a self-similar process with Hurst parameter $H = 1 - \beta/2$.



(a) Time-variance plot for the unsynchronized mode

(b) Time-variance plot for the trace

Figure 4: Debug model using multi-scale analysis

wavelet-based scaling plot The second method, a wavelet-based scaling plot, uses wavelet transform of a time series to study its global scaling property [1], where the statistics of time series viewed at each resolution level or scale is taken as a function of scale. Consider a time series $X_{0,k}$, *k*=0,1,2,...,at the finest scale 2^{-n}, and iterpret X_0 as a traffic rate process (for example, the number of packets per 1 ms), we coarsen X_0 by averaging over non-overlapping blocks of size two

$$X_{1,k} = \frac{1}{\sqrt{2}}(X_{0,2k} + X_{0,2k+1}) \tag{2}$$

and obtain a new time series X_1, a coarser resolution picture of the original time series X_0. The *difference* between these two pictures is defined as

$$D_{1,k} = \frac{1}{\sqrt{2}}(X_{0,2k} - X_{0,2k+1}) \tag{3}$$

Now we can reconstruct the original time series X_0 from the coarser representation X_1 by simply adding the *difference* D_1; i.e.,$X_0 = \frac{1}{\sqrt{2}}(X_1 + D_1)$. Using Eqns.(2) and (3), we can interate this process for as many scales as are present in the original time series and obtain $X_0 = 2^{-n/2}X_n + 2^{-n/2}D_n + ... + 2^{-1}D_1$. To determine the global scaling property of data, we plot log(E_j), where the *energy*E_j is defined as

$$E_j = \frac{1}{N_j}\sum_k |D_{j,k}|^2, j = 1, 2, ..., n, \tag{4}$$

as a function of scale j. The energy level E_j can be interpreted as the level of irregularity or burstiness of sampled data. The higher E_j is, the more bursty the traffic is on time scale j. A linear relationship between log(E_j) and scale j indicates the existence of self-similar scaling.

In our study of RealAudio traffic, the 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 multi-scaling plots detected several errors in our first models. For example, we initially did not capture the synchronization effect of flows in the model, which will be described later in Section 3.3.2. Compared the time-variance plot of flows without synchronization, as shown in Figure 4(a), to that of the trace in Figure 4(b), 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 model.

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

3.3 Case Study: RealAudio

In this section, we demonstrate the application of structural modeling approach in our study of RealAudio traffic.

3.3.1 Background

The primary audio traces used in our study of RealAudio traffic are the same set of data used in a previous study by Mena [60]. (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 [84], 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 [73] 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.

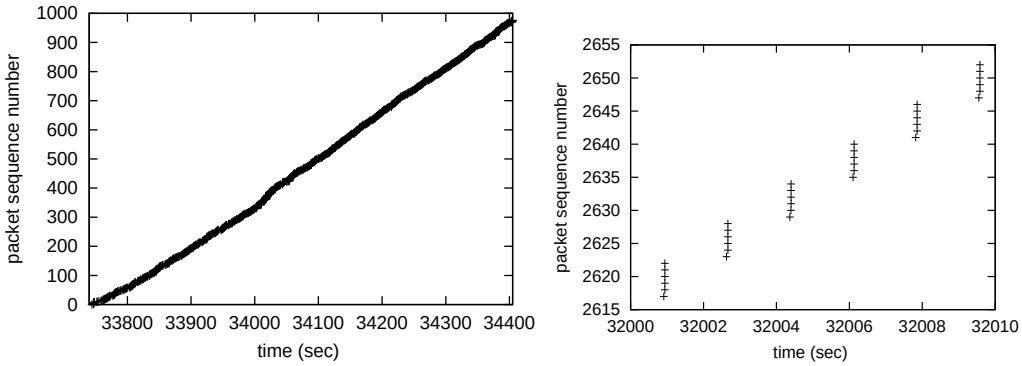
Finally, to supplement our understanding from trace analysis, we perform several experiments, where eight clients are connected to the server via a LAN. The server uses a trial version RealServer G2, which only supports up to eight concurrent audio sessions, and runs on a Pentium III 500MHz Linux box. (To insure that these results are not OS-dependent we also repeated these experiments on a Pentium II 266MHz computer running FreeBSD v3.3.) All clients use RealPlayer 8.0 Basic and utilize Intel Pentium II/III class (266M to 500M Hz) hardware running Linux.

3.3.2 Characteristics of RealAudio traffic

Here we summarize some key characteristics of RealAudio traffic 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 5(a). However, at small time scale (single seconds), it behaves like a bursty on-off source with off period in approximately multiple of 1.8 seconds as shown in Figure 5(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.



(a) medium-timescale view of flow (b) fine-timescale view of flow

Figure 5: Packet Sequence Number Plot for one single flow of the trace

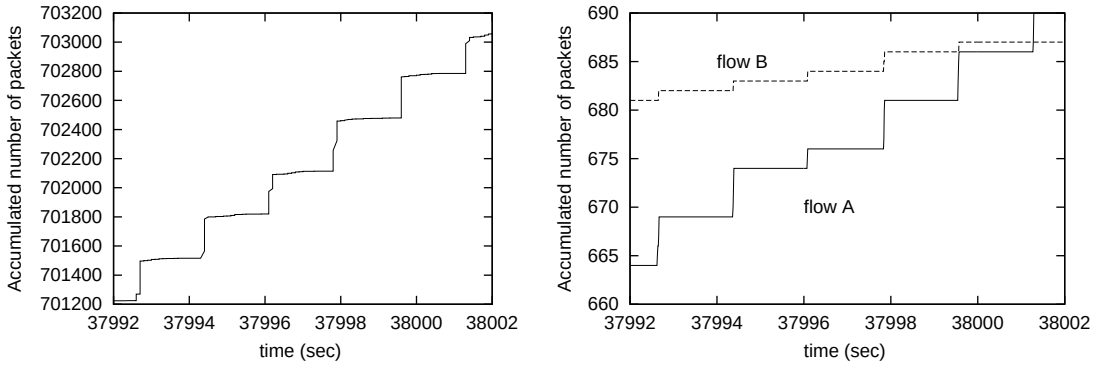
After further examining the trace, we were surprised to find that aggregated RealAudio traffic also shows similar bursty on-off behavior as the individual flow with off period approximately 1.8 seconds, as shown in Figure 6(a). Put in another way, RealAudio flows to different users appear to be synchronized, transmitting together at about the same time. This synchronization can be seen in Figure 6(b) where two randomly chosen flows show bursty transmission in phase with each other. A previous study [26] 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).

Although intuitively the periodic bursty nature of RealAudio traffic, as described previously, 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 [28] showed that self-similarity can be resulted from the dependence of bandwidth variation on scene changes for VBR video traffic. However, after applying the multi-scaling methods described in Section 2.3 to our trace, we conclude RealAudio traffic does not have significant long-range dependence as web traffic does. In the time-variance plot, as shown in Figure 7(a), there is no indication of a straight line with slope greater than -1 in the plot. We also do not see a linear relationship between $\log(E_j)$ and scale j in the wavelet scaling plot, as shown in Figure 7(b).

3.3.3 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 3.3.1.

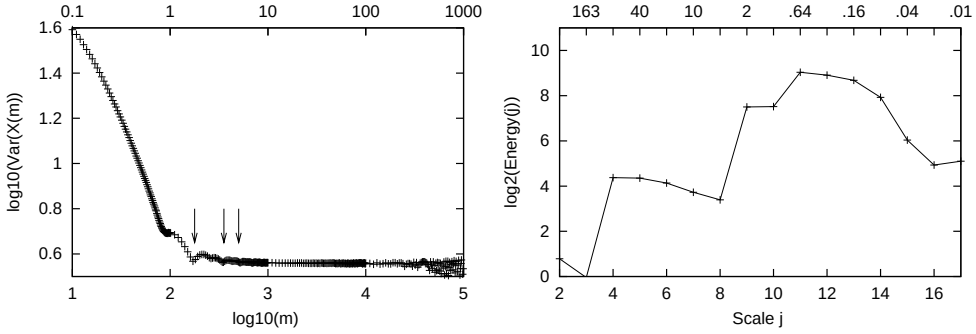
A single RealAudio flow is sent at roughly constant bit rates at *medium* and *large* time scales (Figure 5(a)). We believe this is because that streaming media protocols inherently target steady rate due to encoder and network issues. Some continuous traffic is inherent in streaming media by definition—streaming media sends data incrementally and gradually. Many encoders also target fixed bit rate because some networks such as telephone



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

(b) synchronization of two individual flows

Figure 6: Synchronized RealAudio flows from the trace (y-axes have different scales)



(a) Time-Variance Plot

(b) Global Scaling Plot

Figure 7: Multi-scale analysis of trace

and ISDN only provide small, fixed bandwidth channel. VBR codecs are less desirable for fixed bandwidth links because they are generally more complex.

Figure 5(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*[56] 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 [56]). The cost of a context switch is inherent in supporting multiple processes with separate CPU resources (registers, memory protection, etc.).

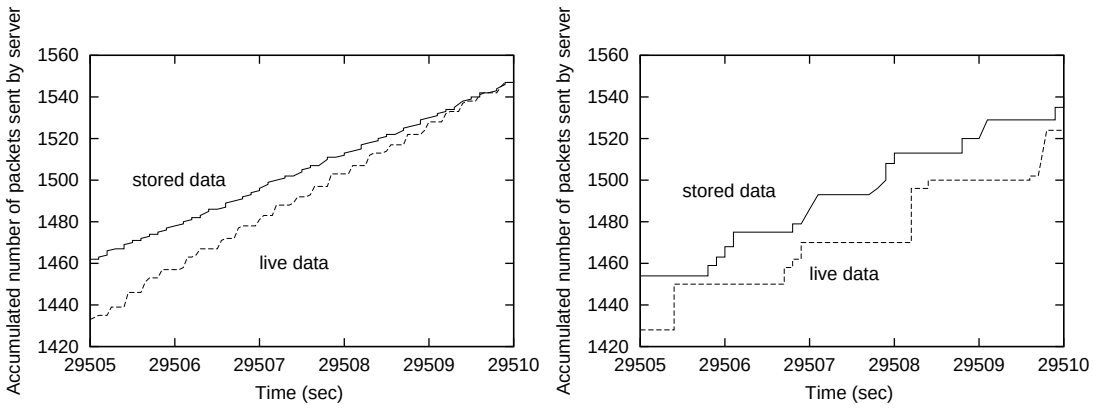
Although there must be a context switch cost, this cost does not need to be incurred on every packet exchange. A simple solution is to send multiple packets each time when the thread is scheduled, possibly dynamically adjusting the number of packets sent to the frequency of context switches. To verify 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[1] programs [78], we see burstiness similar to that observed in Figure 5(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 8(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 8(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.

[1]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



(a) low CPU load

(b) high CPU load

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

3.3.4 Structural model of RealAudio

We have analyzed RealAudio traffic (Section 3.3.2), 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 a simulator model of this traffic.

We chose to model this traffic in ns-2 [7, 31] 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.

Based on the description of RealAudio traffic from Section 3.3.2, we designed a three-level (namely, user-, flow-, and packet-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 [8] and reaction to the congestion. The arrival of 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.

3.3.5 Validation of RealAudio 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 consistent.

Individual flow 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.

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 6(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

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

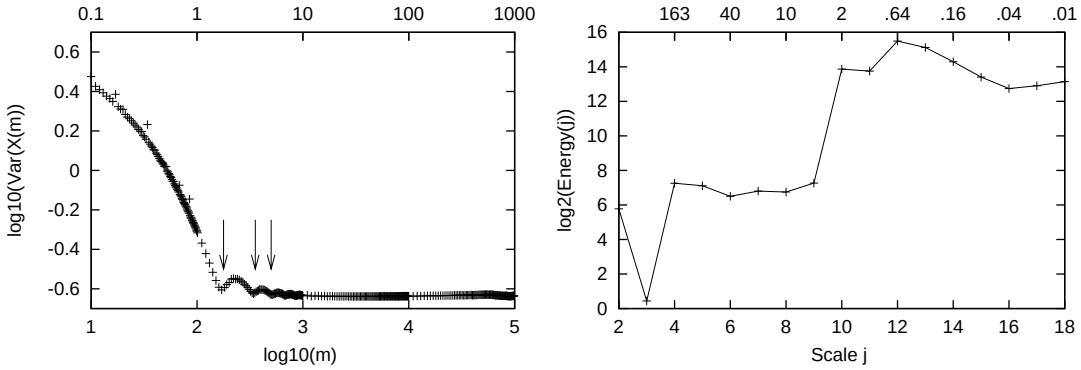
Table 3: Comparison of first-order statistics for trace and model.

Aggregated traffic 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. We then compare two multi-scale plots, as described in Section 2.3, between our model and trace. In all cases they matched closely.

First-order statistics We compare five metrics quantitatively between trace and model as shown in Table 3. 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. 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 [60]. Although not presented here, we found a similar level of consistency with trace data sets 4 and 5 from [60]. We later evaluate the sensitivity of these results to newer version of RealAudio in Section 3.4.

Multi-scaling analysis The two multi-scale plots generated from our model are shown in Figure 9. First we consider the time-variance plot, as shown in Figure 9(a). Comparing Figure 9(a) to Figure 7(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



(a) Time-Variance Plot

(b) Global Scaling Plot

Figure 9: Multi-scale analysis of model

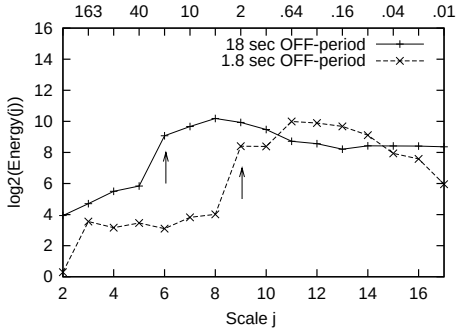


Figure 10: Global scaling plots for models with different OFF-periods

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 9(b) to Figure 7(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 10.

3.4 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 3.3.3. 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 8(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 3.3.3, live data and stored content will behave differently in the aggregated traffic, and eventually determine how we structure the model.

4 Rapid Model Parameterization

Floyd and Paxson [27] provide an excellent overview of the problem, namely the constantly-changing and decentralized nature of the Internet, that results in a poor understanding of traffic characteristics and makes it difficult to define a typical configuration for simulating the Internet. Motivated by the their observations and based on the structural modeling approach, we have developed methodologies and tools that allow users to quickly populate traffic models based on the measurements and generate realistic *contemporary* traffic in their simulations.

Opposed to traditional trace-replay techniques which typically ignore the fact that traffic is frequently *shaped* by the network's current properties, our approaches focus on how to characterize source-level pattern in which the data is sent. We have developed tools and methodologies to support this trace-driven application-level model approach for generating synthetic traffic. Our initial studies emphasize two types of dominant traffic, namely web and FTP traffic, and show that we can accurately generate the simulation model from live data in a *timely* fashion. Potential applications of a rapid model parameterization tool will include on-line simulation, input for QoS parameters setting and input to analysis-based algorithms to detect failure conditions.

The results of our work has two folds. First, we strengthen Floyd and Paxson's arguments by showing that network characteristics not only change over time but also show great variations in other dimensions such as locations and flow directions. Second, we develop methodology that allows users to rapidly generate simulation model that captures their *current* traffic, and tool that automates this process.

In this section, we first shows Internet traffic is different any which way you look. We then describe the tool *RAMP* we have developed that aims to cope with this rapidly evolving nature of network traffic.

4.1 Traffic is different any which way you look

In this section, we show Internet traffic looks differently both in time and space domain after examining the traces we obtained from different locations and at different time. These observations clearly stress the importance of being able to re-populate models with new data to account for evolution of the traffic.

Trace	ITA	ISI
Date	Nov 2001	Nov 2001
Duration (hr)	24	42
Total Packets	2.5M	218M
Bytes	2.4G	187G
TCP Packets	2.5M (100%)	143.9M (66%)
Bytes	2.4G (100%)	122G (65%)
UDP Packets	3 (0%)	69.8M (32%)
Bytes	150 (0%)	65G (35%)
HTTP Packets	0.1M (4%)	50M (23%)
Bytes	50M (2%)	71G (38%)
FTP Packets	2.4M (96%)	39M (18%)
Bytes	2.35G (98%)	64G (34%)

Table 4: Summary of ITA and ISI traces

4.1.1 Traces

The data used in this study are from two sources. One was collected on the web server of Internet Traffic Archive [68]. (this set of trace will be referred to as "ITA" in this paper) The other was recorded at a 100Mbps ethernet link connecting the Information Science Institute to the rest of the Internet. (referred to as "ISI")

ITA trace was collected using publicly available software *tcpdump*. ISI trace was captured via tcpdpriv [61] utility which anonymizes *libpcap*-format (same format used in *tcpdump*) traces. tcpdpriv can collect traces directly or post-process them after collection using a tool like *tcpdump*. Both traces captured all inbound and outbound traffic but only TCP/IP header information was recorded for reasons like privacy and storage overhead. Note that the traffic volume of ITA trace is significantly lower than that of ISI trace and mainly consists of outbound traffic.

The ITA trace was collected during a 24-hour period starting from 15:20 Nov 6, 2001, and shows obviously bimodal distribution of traffic mix consisting primarily of HTTP and FTP traffic. The ISI traces was collected during six one-hour sampling periods each day over a seven-day period starting from Nov 9, 2001. The one-hour sampling periods were chosen somewhat arbitrarily with the intention to capture the variation of traffic between different time of the day.

The typical link utilization during collection period is around 16% to 23% and there is no packet drop in our measurement. For simplicity, in this paper we only show the analysis of two sets of one-hour long ISI data which were collected at different time of the same day. One was recorded starting at 2:00 pm Nov 13 2001 (referred to as ISI-1) and the other was recorded starting at 7:00 pm Nov 13 2001 (referred to as ISI-2). Intuitively, one captures the traffic in a normal business-hour and the other shows traffic in after-hours. The details of traces are given in Table 4.

4.1.2 Metrics used for comparison

We determine if two different sets of data are *different* by comparing them qualitatively and quantitatively. By qualitatively, we visually inspect the CDF plots of first-order statistics at three different levels (i.e. packet-, flow- and user-level statistics) and the wavelet scaling plots between the trace and model to see if they match closely. Here we define a *flow* as an unidirectional series of IP packet traveling between a source and a destination IP/port pair within a certain period of time, and an unique IP address as a *user*. Specifically , the metrics we use for comparison include packet inter-arrival time, packet size, flow duration, flow size, flow inter-arrival, user inter-

arrival, user duration, protocol mix and traffic volume. We only show the CDFs of flow statistics and wavelet scaling plot in this paper for brevity since they are less dependent on the density of traffic.

By quantitatively, We perform Kolmogorov-Smirnov goodness of fit test [52] to formally determine if two sets of traffic data are significantly different from each other, except from visually examining their CDF plots. The Kolmogorov-Smirnov D value is the largest absolute difference between the cumulative distributions of two sets of data. We first compute D value of two data sets and then compare the result to the *critical value* of D. For large number of samples, the critical value at the .05 level significance is approximately $\frac{c}{\sqrt{n}}$, where n = sample size and c is a constant that is distribution-dependent. For example, if the tested data comes from a normal distribution then c=1.36 [79] (c=1.08 for exponential distribution [46] and c=0.874 for Weibull distribution [14]) If the computed D is less than the critical value then we accept the null hypothesis that the distributions of two data sets are not different from each other. Note that since we do not assume our data comes from any theoretical distribution, we use c=0.874 (which is the most restrict as we know) as an approximation to perform the test. In other words, at a 0.05 level significance and for 10000 samples, we will claim two data sets are *different* if the maximum absolute deviation between their cumulative distributions is greater than 0.00874. As reported by previous studies [69, 4], it is difficult to apply goodness-of-fit test for large empirical data set. Therefore we also adopt similar approach as described in previous work [6, 69, 4] by using random sub-samples in our test. The number of samples (which are randomly picked) we use for K-S test are 10000 throughout the paper. (In other words, we compare the computed D value with a critical value of 0.00874 in each test.)

4.1.3 Traffic seen in different direction

First we look at traffic flows in different direction (i.e. inbound traffic versus outbound traffic) during the same period of time. We found inbound traffic and outbound traffic are significantly different in terms of protocol mix and via comparison of first-order statistics and wavelet analysis.

The protocol mixes for inbound and outbound traffic of ISI-1 data are shown in Table 5. The traffic mix is noticeably different in different direction, where the inbound traffic is dominated by News traffic while the outbound traffic mainly consists of web and FTP traffic. Note that NNTP traffic in outbound data mainly consists of ACKs, which is the reason it contributes very little in terms of bytes to the total traffic volume. In terms of the number of flows, the majority of the flows are contributed by DNS traffic in inbound traffic while by web traffic in outbound data.

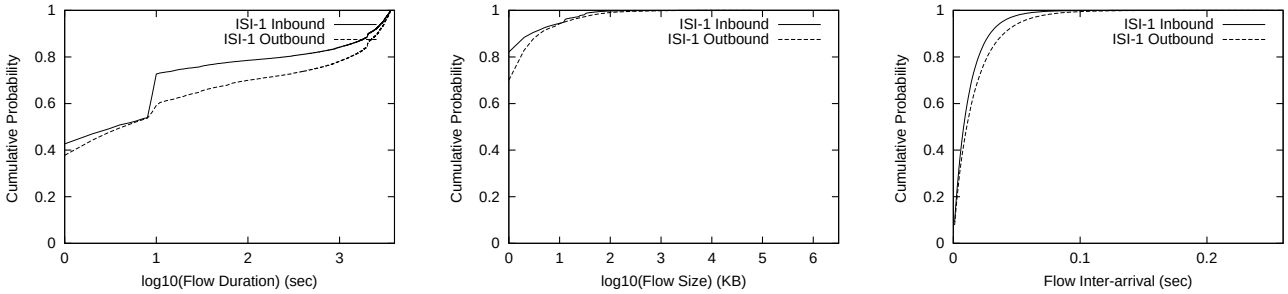
We next look at the first-order statistics. The comparison of flow statistics, including flow duration, flow size and inter-arrival time of inbound and outbound data are shown in Figure 11. Outbound traffic has comparatively longer flow duration and size than inbound traffic, which is possible due to the fact that the majority of the flows are contributed by DNS traffic in inbound traffic while by web and FTP in outbound traffic, as shown in Table 5. Although in Figure 11(b) and Figure 11(c) the CDF plots for outbound and inbound traffic look similar in the tail of the distributions (lower tail in flow size and upper tail in flow inter-arrival), none of them pass the Kolmogorov-Smirnov test. The D values for Figure 11(b) and Figure 11(c) are 0.121 and 0.097 respectively, which are larger than the critical value and hence fail the test. (The number of samples we use are 10000 which corresponds to a critical value of 0.00874.)

The corresponding wavelet scaling plot is shown in Figure 12. We observe there is a pronounced dip on the order of about 128ms, which reflects the underlying periodic component (i.e. RTT) for outbound traffic, while the dominant RTT for inbound traffic is on a relatively smaller time scales (about 40ms).

All the statistics conclude that ISI-1 inbound and outbound traffic are noticeably different from each other.

Protocol	Inbound	Outbound
NNTP (% packets)	39.4%	8%
(% bytes)	**64.4%**	0.02%
(% no. of flows)	0.06%	0.08%
HTTP (% packets)	15.8%	27.6%
(% bytes)	20.0%	**50%**
(% no. of flows)	38.5%	35.8%
DNS (% packets)	29.9%	31.6%
(% bytes)	4.8%	4.8%
(% no. of flows)	51.4%	30.1%
FTP (% packets)	5.5%	20.4%
(% bytes)	4.1%	33.7%
(% no. of flows)	8.7%	26.2%
OTHERS (% packets)	9.4%	20.4%
(% bytes)	6.7%	13.3%
(% no. of flows)	1.5%	7.8%

Table 5: Summary of protocol mix of in ISI-1 traffic



(a) Comparison of flow duration of inbound and outbound traffic

(b) Comparison of flow size of inbound and outbound traffic

(c) Comparison of flow inter-arrival time of inbound and outbound traffic

Figure 11: Comparison of flow statistics in ISI-1 data

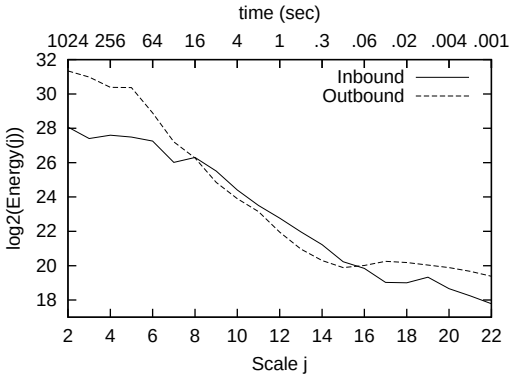


Figure 12: Comparison of wavelet scaling plot of inbound and outbound traffic in I SI-1 data

Protocol	ISI-1	ISI-2
NNTP (% packets)	8%	10%
(% bytes)	0.02%	0.02%
(% no. of flows)	0.08%	0.09%
HTTP (% packets)	27.6%	17.5%
(% bytes)	**50%**	24.0%
(% no. of flows)	35.8%	32.6%
DNS (% packets)	31.6%	41.0%
(% bytes)	4.8%	11.4%
(% no. of flows)	30.1%	34.5%
FTP (% packets)	20.4%	22.1%
(% bytes)	33.7%	**45.7%**
(% no. of flows)	26.2%	31.3%
OTHERS (% packets)	20.4%	9.4%
(% bytes)	13.3%	18.9%
(% no. of flows)	7.8%	7.0%

Table 6: Summary of protocol mix of of ISI outbound traffic at different time

4.1.4 Traffic seen at different time

We next look at two sets of ISI traffic captured at different time (i.e. ISI-1 and ISI-2). Here we concentrate on the comparison of outbound traffic. Since it was recorded during the time when most people have left the office. Intuitively the inbound traffic in ISI-2 data will be different from ISI-1 because of its smaller user population. (For inbound traffic, ISI-1 has 517 users while ISI-2 has only 128 users. For outbound traffic, ISI-1 has 16447 users and ISI-2 has 14259 users.) The following statistical comparisons show that ISI-1 and ISI-2 outbound traffic are different from each other.

First we look at the traffic mix, as shown in Table 6. Although large percentages of traffic in both data sets are made up by web and FTP traffic, but one is dominated by FTP while the other by web traffic.

The distributions of flow statistics including flow duration, flow size and inter-arrival time for ISI-1 and ISI-2 data are shown in Figure 13. The flow duration in ISI-2 data is significantly longer than that in ISI-1, as shown in Figure 13(a), which is probably due to that ISI-1 data is dominated by web traffic while ISI-2 is dominated by FTP flows. In terms of flow size, there are more short flows in ISI-2, which is probably because there is more DNS traffic and short HTTP connections in ISI-2 data, as shown in Table 6.

Again, although the CDF plots between ISI-1 and ISI-2 in Figure 13(b) and Figure 13(c) have similar shapes, they all fail the Kolmogorov-Smirnov test. (the D value are 0.09 and 0.14 respectively, for 10000 samples)

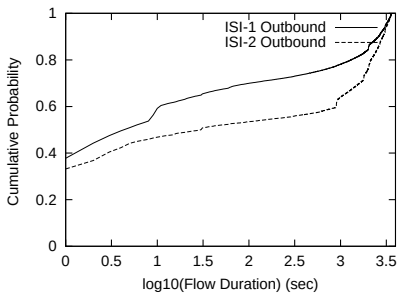
The wavelet scaling plot, as depicted in Figure 14, indicates ISI-2 traffic has smaller and more heterogeneous RTT behavior shown as a dip stretches from 8ms to 128ms while ISI-1 data has a main dip at 128ms.

All the statistical comparisons indicate that ISI-1 outbound traffic is different from ISI-2 outbound traffic.

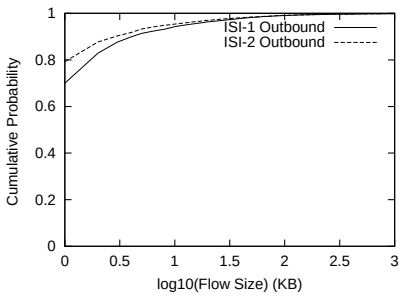
4.1.5 Traffic seen at different location

Finally we look at the comparison between ISI-1 and ITA data and show traffic is different at different locations. Again, we only focus on outbound traffic.

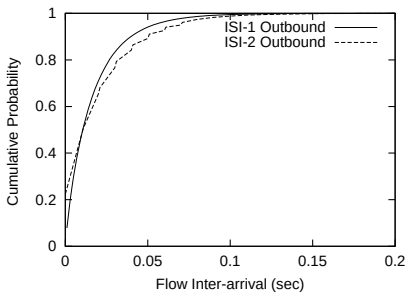
In terms of protocol mix, ITA data only consists of HTTP and FTP traffic, which is obviously different from the protocol mix in ISI-1 traffic. The distributions of flow statistics, including flow duration, flow size and inter-



(a) Comparison of flow duration between ISI-1 and ISI-2



(b) Comparison of flow size of between ISI-1 and ISI-2



(c) Comparison of flow inter-arrival time between ISI-1 and ISI-2

Figure 13: Comparison of flow statistics for ISI outbound traffic at different time

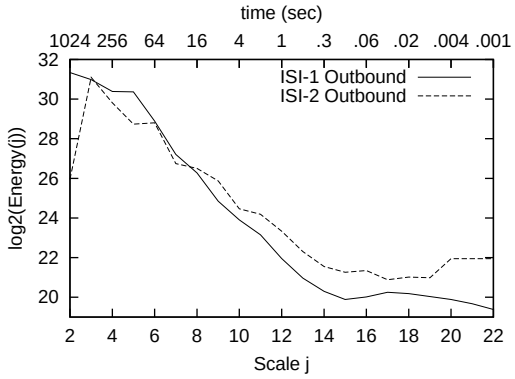
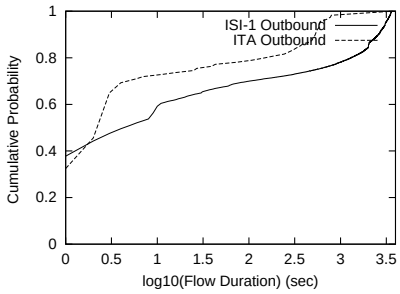
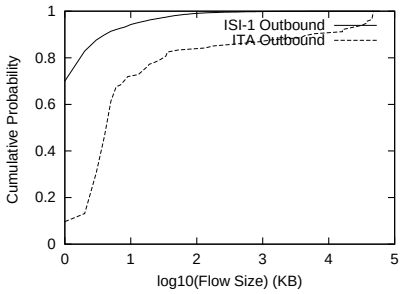


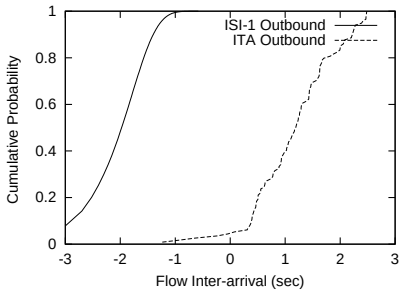
Figure 14: Wavelet scaling plot for ISI-1 and ISI-2 outbound traffic



(a) Comparison of flow duration between ISI-1 and ITA



(b) Comparison of flow size of between ISI-1 and ITA



(c) Comparison of flow inter-arrival time between ISI-1 and ITA

Figure 15: Comparison of flow statistics for ISI and ITA outbound traffic

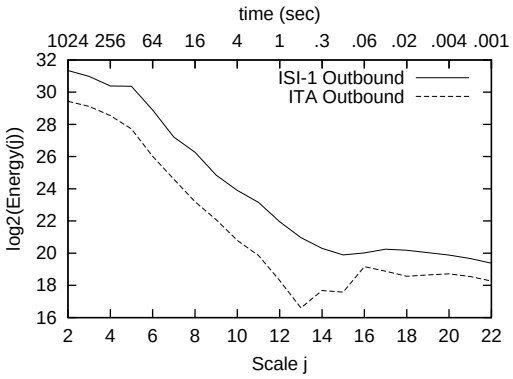


Figure 16: Comparison of wavelet scaling plot between ISI-1 and ITA outbound traff ic

arrival time for ISI-1 and ITA data are shown in Figure 15. We see ISI-1 has longer flow duration but smaller flow size. A close look shows that the long flows in ISI-1 mainly are contributed by DNS, NTP (periodic time synchronization between servers) and NNTP traffic (periodic news exchanges between servers). ITA data has larger flow size because it mainly consists of bulk FTP transfer. It is not surprising that ITA has much larger flow inter-arrival time since its traffic is much more sparse than ISI-1. We did not apply the Kolmogorov-Smirnov Test to ITA and ISI-1 data since their CDF plots are obviously different.

In the wavelet scaling plot, as shown in Figure 16, we observe there is a main dip at time scale of around 500ms for ITA data, which is about 4 times larger than the 128ms in ISI-1 data. A closer look shows ITA traffic is dominated by a couple of FTP transfers between ITA site and some hosts in the US west coast and Europe. All the statistical comparisons here show that traffic can be different at different sites because of the nature of their contents difference.

The above discussion concludes that network traffic not only changes over time but also shows great variations in different directions and different locations. We demonstrate the differences can be due to a variety of reasons such as user behavior, path characteristics and application usage etc., and hence it is difficult to obtain a “general” traffic model.

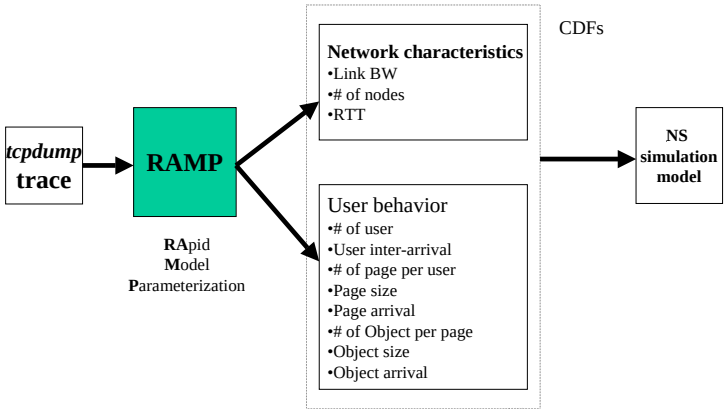


Figure 17: Data flow of RAMP

4.2 RAMP: RApid Model Parameterization

Motivated the previous observation that it is important to quickly re-populate models with new data to account for the diversity of the traffic, we design a tool called *RAMP*. RAMP can convert live measurements into simulation models which then be used to generate realistic synthetic traffic. In this section we first describe the design of RAMP. We then show the validation of RAMP via various statistical techniques. Finally we will discuss the performance of RAMP and some of its limitation.

4.2.1 Design of RAMP

Our approach is to automatically generate statistics that model user behaviors and network path characteristics by analyzing TCP/IP header information captured in the measurements. The resulted model will then be built into the widely-used ns network simulator [7] and validated against the original trace via wavelet-based analysis and first order statistical comparison.

The input of RAMP is a tcpdump-format file, recorded at a single tap point of the network, that contains only TCP/IP header information. The output of RAMP is a set of CDF (Cumulative Distribution Function) files that model the corresponding traffic, as shown in Figure 17. Specifically, the CDF files consist of two types of data. One set of CDF files model user/application level statistics of the traffic, such as user session arrival, page/file size etc. Currently RAMP only supports web and FTP traffic which are among the most dominant types of traffic [55] of the present Internet. The other one models path characteristics of the network. In particular, we focus on characterizing RTT and bottleneck bandwidth of the measured traffic since they are important parameters for driving network simulation. Typically it takes tens of minutes for RAMP to process a trace file with the size of several hundreds Megabytes.

997826350.296819 129.1.7.14.80 > 202.10.162.34.4645: S 2278247361:2278247361(0) ack 132534867 win 8760
997826350.312486 129.1.7.14.80 > 202.10.162.34.4645: . ack 326 win 8760
997826350.313099 129.1.7.14.80 > 202.10.162.34.4645: P 1:1461(1460) ack 326 wi n 8760
997826350.430730 129.1.7.14.80 > 202.10.162.34.4645: P 1461:2458(997) ack 326 win 8760
997826367.549809 129.1.7.14.80 > 202.10.162.34.4645: . 2458:3918(1460) ack 675 win 8760
997826367.549942 129.1.7.14.80 > 202.10.162.34.4645: P 3918:5378(1460) ack 675 win 8760
997826367.550065 129.1.7.14.80 > 202.10.162.34.4645: P 5378:6838(1460) ack 675 win 8760
997826367.565980 129.1.7.14.80 > 202.10.162.34.4645: . 6838:8298(1460) ack 675 win 8760
997826367.566105 129.1.7.14.80 > 202.10.162.34.4645: . 8298:9758(1460) ack 675 win 8760
997826367.566228 129.1.7.14.80 > 202.10.162.34.4645: P 9758:11218(1460) ack 675 win 8760
997826367.581947 129.1.7.14.80 > 202.10.162.34.4645: . 11218:12678(1460) ack 675 win 8760
997826367.582068 129.1.7.14.80 > 202.10.162.34.4645: P 12678:14124(1446) ack 675 win 8760
997826397.549684 129.1.7.14.80 > 202.10.162.34.4645: F 14124:14124(0) ack 675 win 8760

Table 7: *tcpdump* trace that shows two request/response exchanges in a persistent HTTP connection

User and application behavior characterization First we describe the techniques we employ to characterize the source-level behaviors based on the TCP/IP header information captured in the trace. We focus on the analysis of web and FTP traffic which are among the most dominant types of traffic in the present Internet.

Web traffic Here we present the methodology used to characterize the important components of web traffic based on only the information in the TCP/IP headers and knowledge of the TCP and HTTP protocol.

To reconstruct the data exchanges in the HTTP connections based on only the information in TCP/IP header, we adopt a similar approach and heuristics from previous work [80]. One observation in their study is that when the server receives a HTTP request it will send TCP acknowledgement (ACKs) indicating the in-order byte sequence it has received, and all of the request message will be ACKed before the corresponding HTTP response data is sent. (Note that here we assume there is no pipelining in use.) Hence we can infer the size of request by the amount of ACK value advances and the size of response by the amount of data sequence number advances. As the example shown in Table 7, the ACK-only segment from the server following the SYN+ACK segment indicates the first request was 325 bytes in size. In the following segments, the data sequence numbers advance to 2458 (the size of first response) with no further changes in the ACK values. In the next segment, the advance of ACK number indicates the size of the second request was 349 bytes (675 - 326). In the following segments, the data sequence numbers advance with no further changes in the ACK values. The size of second response is 11756 byes (14124 - 2458).

Adopting similar heuristics as those developed originally by Mah [50] and Barford and Crovella [4], we assume a new page is requested after some period of idle time (or “think” time) at the client. We identify idle periods in which either the client has no established TCP connection or no established connection has an active request/response exchange in progress.

Our web traffic model is similar to those developed originally by Mah [50] and Barford and Crovella [4]. However, we found it is important, but not captured by the previous studies, to model the TCP window size and the usage of persistent connection.

It is important to model TCP window size in order to accurately characterize sending rate of the servers. For example, as shown in Figure 18, more than 80% of clients in the ISI1 inbound traffic use window size less than 16K. Using small window size will limit the servers from fully utilizing increasingly-popular broadband networks such as DSL and cable modem. Note that we did not observe any connection that uses TCP window scale option in our traces.

Motivated by the increasingly important role of persistent connection in web traffic, as reported by previous

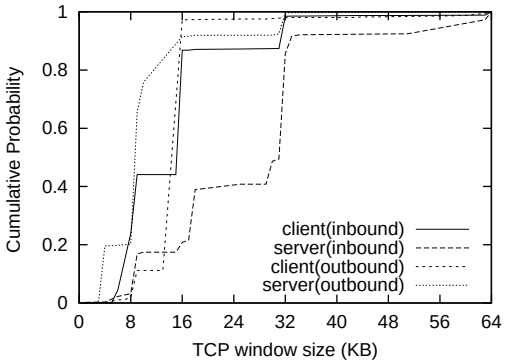


Figure 18: Comparison of the usage of TCP window size in inbound and outbound traffic of ISI-1 data

Protocol	**Inbound**	**Outbound**
Number of connections	26426	4425
objects	44399	7187
bytes	318.7 MB	424 MB
Persistent connections	4756 (18%)	708 (16%)
Objects on Persistent	22841 (51%)	3506 (49%)
Bytes on Persistent	121.5 MB (38%)	85.4 MB (20%)

Table 8: Summary for the usage of HTTP persistent connections in ISI-1 traffic

study [80], we also model the persistent connection used in HTTP/1.1, As shown in Table 8, although only less than 20% of connections are persistent, they account for about 50% of all objects transferred and more than 20% of all bytes transferred. This clearly shows persistent connection plays an important role in the dynamics of TCP connections for the Web. In our datasets, over 50% of persistent connections are used for three or more request/response exchanges and 10% of them carry more than nine (the graphs are not shown here). Our result for the usage of persistent connections shows strong agreement with recent studies [80]. Note that although we have observed in our datasets that some browsers still use multiple concurrent connections to transmit one single page as reported in Balakrishnan's study [3], we did not model that since it accounts for only less than two percents of total number of pages in our traces.

FTP traffic Here we show that, unlike commonly believed, it is non-trivial to extract FTP flows in the traces. (In particular, it is not sufficient that one only looks at the flows that origin from or destine to port 20 or 21.)

For FTP traffic, we assume an unique IP address represents a single human user and a new TCP connection is used for each file transmission. This heuristics allows us to identify the points when client starts a new file. The FTP protocol [72] specifies that the client first connects from a random unprivileged port ($N > 1024$) to the FTP server's command port, port 21. The client then starts listening to port N+1 and sends the FTP command "PORT N+1" to the FTP server. The server will then connect back to the client's specified data port from its local data port, which is port 20. This is also known as Active-mode FTP.

However, from our datasets we observed that there are significant number of clients are using Passive-mode FTP, in which the client initiates both control and data connections to the server. When opening an FTP connection, the client opens two random unprivileged ports locally ($N > 1024$ and N+1). The first port contacts the server on

port 21, but instead of then issuing a PORT command and allowing the server to connect back to its data port, the client will issue the PASV command. The result of this is that the server then opens a random unprivileged port ($P > 1024$) and sends the PORT P command back to the client. The client then initiates the connection from port N+1 to port P on the server to transfer data. To identify FTP traffic, we first locate FTP clients by looking at those connected to server port 20 and find out what are the control ports (N) they use. We then look for the connections originating from the neighboring port (N+1) of client's control port and classify them as FTP data connections.

Characterization of network path properties Next we describe how do we estimate the topology information from the measurement. Particularly we focus on characterizing the round trip delay and bottleneck bandwidth since both of them are important for driving the simulation.

RTT We determine the RTT of each TCP connection in our traces by computing the difference of timestamp between data packet and the first ACK packet which has the same sequence number. However, this approach is not applicable for packets captured at the data receivers end, where the timestamp difference between data and ACK doesn't reflect the path delay. For situation where the clients are near the measurement point while servers are at the remote end (eg. the inbound traffic), we rely on the three-way handshake at the start of each TCP connection to calculate the delay of the path. In other words, we compute the RTT by taking the timestamp difference between the SYN packet and its corresponding ACK. For each connection we take the minimum of RTT samples as an approximation of propagation delay of the path (after dividing the RTT by 2) and consider the deviations from the minimum RTT as variances caused by queuing delay and transmission delay. We use this approximation to drive our simulation.

Bottleneck bandwidth Our traces contains both outbound and inbound traffic. For outbound traffic, we use Sender Based Packet Pair (SBPP) [70] to compute the bottleneck bandwidth between the local servers and the remote clients. That is, we estimate the spacing between a pair of back-to-back TCP packets after passing the bottleneck link by examining the arrival times of their corresponding ACKs (for delayed-ACK packets, we estimate the spacing between the second and the forth packets of a group of 4 back-to-back packets). For inbound traffic, we rely on Receiver Only Packet Pair (ROPP) [41], which uses the arrival times of two consecutive full-size packets at the receiver to estimate the bottleneck bandwidth between remote servers and the local clients. We also apply similar techniques to filter noise such as density estimation as described in [42].

4.2.2 Structural simulation model

We design a three-level simulation model to characterize web traffic and two-level model to characterize FTP traffic as shown in Table 9 and Table 10. Note that we only model the data connections of FTP traffic for simplicity since the bandwidth usage of control channel is negligible (typically less than one percent of total traffic in our datasets). Our web traffic model is similar to those developed originally by Barford and Crovella [4]. Additionally, we model TCP window size and the usage of persistent connection. We also model HTTP request size motivated by the trend in using large requests due to the increasing popularity of "web email" [80].

4.2.3 Validation of RAMP

To validate if RAMP accurately reproduce the traffic under study, we incorporate its output into ns-2 simulator and compare the result of simulation against the original traces. To understand if RAMP can perform as well as existing work in terms of generating realistic synthetic workload, we also compare RAMP against SURGE [4], a popular web traffic workload generator.

User behavior

1. User arrival is modeled as a Poisson process with certain rate.
2. The number of pages per user session is randomly picked from the CDF(Cumulative Distribution Function) of trace.
3. the sources of page are chosen from a CDF that matches the popularity of servers
4. Each page is sequentially requested by the users as described below.

Page

1. Page size is chosen from a CDF
2. The inter-arrival time of page is picked from a CDF
3. The number of objects within one page is picked from a CDF
4. The size of request to a page is picked from a CDF
5. User decides a TCP connection is used for multiple request/response exchanges or a single request/response exchange based on the probability of persistent connection (HTTP1.1) versus non-persistent connection (HTTP1.0) computed from the trace. In persistent connection mode, all objects within the same page are sent via the same TCP connection.

Object

1. The inter-arrival time of object is picked from a CDF
2. The size of object is picked from a CDF
3. The TCP window size for both servers and clients are also randomly chosen from a CDF

Table 9: Structural model of web traffic

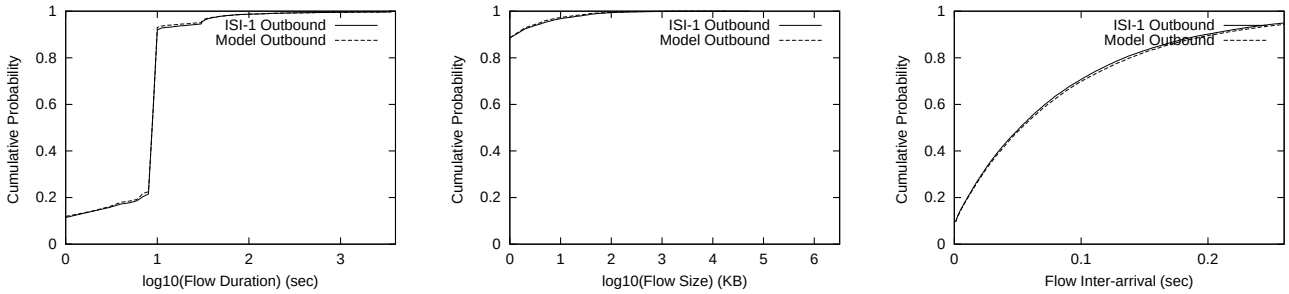
User behavior

1. User arrival is modeled as a Poisson process with certain rate.
2. The number of file transmitted per user session is randomly picked from the CDF(Cumulative Distribution Function) of trace.
3. the sources of file are chosen from a CDF that matches the popularity of servers
4. User starts a new TCP connection for each new file which is sequentially transmitted as described below.

File

1. file size is chosen from a CDF
2. The inter-arrival time of file is picked from a CDF
3. The TCP window size for both servers and clients are also randomly chosen from a CDF

Table 10: Structural model of FTP traffic



(a) Comparison of flow duration between model and ISI-1 trace

(b) Comparison of flow size of between model and ISI-1 trace

(c) Comparison of flow inter-arrival time between model and ISI-trace

Figure 19: Comparison of flow statistics for model and ISI-1 outbound traffic

Comparison with original traces In this section we use ISI-1 data to evaluate the accuracy of RAMP. The result shows the output of simulation match the original traces closely. Note that because currently our tool only supports web and FTP traffic, we first filter the traces so that they only contain web and FTP data before being compared against the simulation result. (Together web and FTP traffic account for 83.7% of the total traffic in term of the number of bytes in ISI-1 trace.)

The statistics here we use for validation including the distributions of flow arrival, flow size, flow duration, packet inter-arrival time, wavelet scaling plot and the application-specific parameters, such as page size, page arrival, object size (for web traffic), file size, file arrival (for FTP traffic), user arrival and user duration. Again, here we only show outbound traffic and only CDF plots of flow statistics for simplicity. (Although the graphs of inbound traffic are not shown here, they are consistent with the results of outbound traffic.)

The CDF plots of flow statistics for ISI-1 model are depicted in Figure 19, which shows the model matches the trace closely. The Kolmogorov-Smirnov test D values for Figure 19(a), Figure 19(b) and Figure 19(c) are 0.0019, 0.0013, 0.0018 respectively. They all pass the K-S test given a critical value of 0.00874.

The corresponding wavelet scaling plot for ISI-1 model, as depicted in Figure 20, also shows large degree of resemblance between trace and model, such as similar energy value (the model has slightly lower energy though) and a dip around 128ms (which reflects the RTT of the underlying traffic).

The CDF plots of model parameters such as page/file size, user arrival etc. also match closely (which are not shown here), which are not surprising though since the model is directly driven by those parameters.

All the statistical comparisons show RAMP is able to accurately reproduce the original traffic.

Comparison with SURGE In order to understand if RAMP can generate representative workload, we compare RAMP against an existing traffic generator, namely SURGE [4]. We demonstrate that our model generation tool is capable of achieving the same functionality of SURGE (i.e. generating similar traffic workload like SURGE) without suffering its limitation due to some of its implicit assumptions.

SURGE contains a set of programs that pre-compute several datasets and a multithreaded program that makes web requests using those datasets. Both are written in C. The datasets consist of the distribution models of number of requests, file sizes, popularity of files, embedded objects, file temporal locality and OFF time.

To validate RAMP against SURGE, we performed a lab experiment by running SURGE for 30 minutes and recording the traffic via *tcpdump*. We then fed the SURGE trace into RAMP and inspected if the output of ns-2 simulation model from RAMP agrees with SURGE trace. The environment of experiment consists of five PCs connected by an 10MBps ethernet switch. Four of these boxes are used as SURGE clients which are Pentium

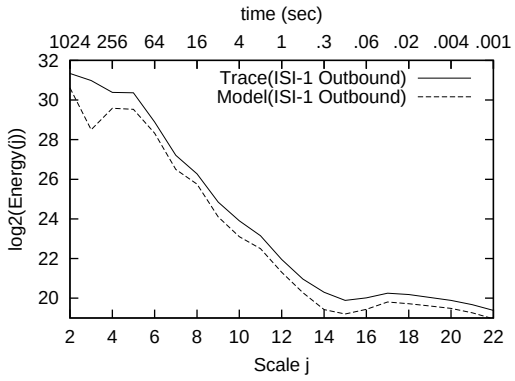


Figure 20: Comparison of wavelet scaling plots between model and trace for ISI-1 outbound traffic

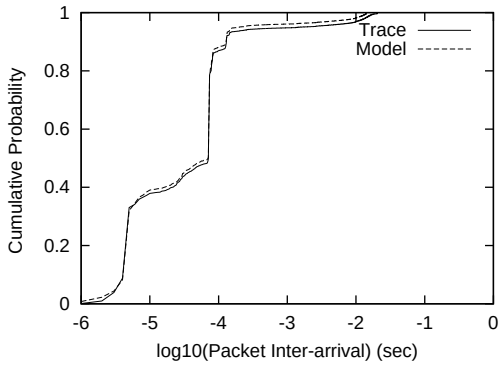


Figure 21: Comparison of packet inter-arrival time between SURGE and RAMP

II/III class (266MHz and above) Linux boxes. We use a Pentium IV Linux box (1.7GHz with 750M memory) as SURGE server which ran Apache v.1.3.22. The number of UE (user entity, SURGE's representation of a web user) and CP (client process, which decides how the threads are spawn) are 5 and 50 respectively. We ran SURGE v.1.00a with HTTP 1.0.

We look at the packet inter-arrival time and wavelet scaling plot of the outputs of SURGE and our model respectively. All the statistics match closely, as shown in Figure 21 and Figure 22.

One limitation of SURGE is that it attempts to fit the models into some widely-used analytic functions (such as using Pareto to describe the distributions of file sizes and off time). However, it is not universally true that all the web traffic follow these assumptions. For example, these assumptions might break for a trace distribution site like ITA. We have observed the distribution of page size in ITA traffic (which are mainly made up by simple plain html files that describe traces and collection/analysis software) is not heavy-tailed, and hence can not be modeled by SURGE. The presence of heavy tails typically is indicated by an approximately straight line in the tail in the LLCD plot [17], which we do not observe in ITA data, as shown in Figure 23. On the other hand, our tool is based on empirical distributions of traffic and does not have any implicit assumption about the distribution of the traffic, and hence it is more flexible to cope with the diversity of the traffic. As the wavelet plot shown in Figure 24, the ITA model generated by RAMP does capture the important features of ITA traffic (such as a dip at 500ms and similar energy levels).

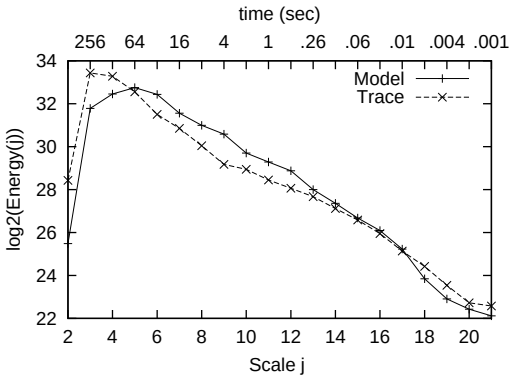


Figure 22: Comparison of wavelet scaling plots between SURGE and RAMP

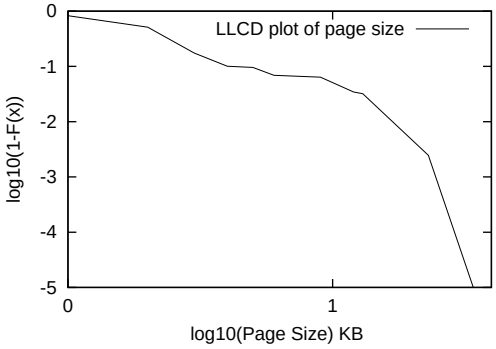


Figure 23: Log-Log Complementary Distribution plot of Page Size in ITA traffic

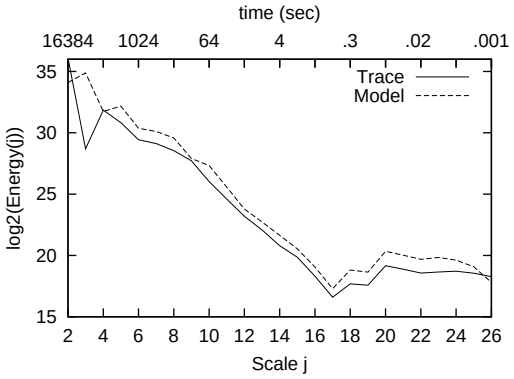


Figure 24: Comparison of wavelet scaling plots between model and trace for ITA outbound traffic

Trace	ISI-1	ISI-2	ITA
file size (MB)	614	561	203
no. of bytes (GB)	1.0	7.3	2.4
no. of packets (M)	9.2	8.4	2.5
no. of flows (K)	506	398	1.3
process time (min)	25	21	8
speed (thousand packets/sec)	6.1	6.3	5.7

Table 11: Process time of RAMP for different traces

4.2.4 Performance of RAMP

Depending on the nature of the traffic (currently RAMP only supports web and FTP traffic) and its actual volume, the time required for the process from analyzing the traces to finally generating the simulation models typically takes tens of minutes for an trace with size of several hundred Megabytes. In this section, we show the speed of RAMP is a function of number of packets in the trace file. Currently we support traces captured in tcpdump format.

To understand what are the factors that will affect the performance of RAMP, we ran RAMP on a 1.7G Hz Pentium IV Linux box with 1G memory for different trace files obtained at different time and different places. As shown in Table 11, we can see the process time of RAMP is approximately proportional to the number of packets in the trace (and hence also proportional to the file size).

4.2.5 Limitation

Our methodology to infer the source behavior of web traffic is based on the limited information available in TCP/IP header for one direction of a TCP connection. There are a number of uncertainties arising from issues such as pipelining, user/browser behavior, caches and TCP segment re-ordering will affect our inference, as described by Smith et al. [80]. However, we expect these cases will typically only appear as very small percentage of total traffic and will not noticeably affect the normal operating condition of our model generation tool.

We find incomplete TCP connections at the beginnings and ends in the data since our traces only cover specific intervals of time (i.e. one hour). We excluded these incomplete connections from our analysis. However, we expect this might have some effect on the results since it will affect some of the model parameters (eg. page size and number of objects per page). However, in this particular study the incomplete connections account for only less than 4% of the total connections in our traces, so we believe it will not significantly distort our results.

There are some known issues with using SBPP and ROPP to measure the bottleneck bandwidth. For example, post-bottleneck queuing will tend to distort the estimation. We make no attempt to claim our estimation of bottleneck bandwidth is very accurate since our inference is only based on passive measurement. However, as shown in Section 4.2.3, our results indicates that these techniques combined with some simple filtering mechanism give us reasonable approximation to estimate bottleneck bandwidth for driving our simulation model.

4.3 Future Work

- **Task 1**: In this study, we use only two set of traces (from ISI and ITA respectively) for the design and validation of RAMP. We plan to collect more traces from other places, particularly those that potentially have very different traffic characteristics (such as at a very high speed link or a very congested site), to further investigate and validate RAMP.

- **Task 2**: (Optional) Currently our tool supports web and FTP traffic which only accounts for a subset of real network traffic. To make the output of RAMP more representative, we would like to incorporate other types of important traffic such as DNS, multimedia traffic (such as Real Audio/Video) and increasingly popular peer-to-peer traffic (such as Morpheus) etc. into our tool.
- **Task 3**: (Optional) To accurately model traffic, it is important to characterize the temporal relationship between different types of traffic. For example, DNS behavior is very likely linked closely to web traffic pattern since most of the web connections are usually preceded by DNS lookups. We plan to study this issue and understand how to orchestrate different traffic classes correctly in the model.

5 Integration measurement from multiple points

Distributed measurements is required to get a network-wide view of the traffic. Furthermore, it is typically infeasible to collect fine-grained data (eg. packet headers information) at the center of the network due to administrative and technical issues. For example, a single direction of OC48 link could produce as much as 100GB of packet headers per hour [63]. For even higher speed link like OC192, such massive volumes of data will place enormous demands on storage and computation resources at the collector and make the centralized approach infeasible if not impossible. To conquer this challenge, one possible way is to infer the internal network traffic by correlating and integrating data collected at the end systems distributed across the network.

5.1 Challenges

However, to correlate and integrate measurements from multiple points of the network will need some techniques for overlap detection and hole filling. In other words, it will demand methodology to merge measurements from multiple points and ways to infer traffic statistics at places where the measurements are not available. We plan to extend the techniques we have developed for RAMP and explore new algorithm and tool to meet the following challenges in a distributed environment: Where should we take the measurement? How to merge traffic? How to infer traffic at places where measurement is not available?

First, where should we take the measurement? Possible candidates may include (as shown in Figure 25):

- Vertex cover: They are the minimum subset of nodes that connect all the other nodes in the network. such as the shaded nodes shown in Figure 25(a). However, such nodes are typically located at the core of network where taking measurement is possibly not feasible for both political and technical reasons.
- All the edges nodes: As an alternative, we can take measurement at all the edge nodes since they are the sources of the traffic (the shaded nodes shown in Figure 25(b)). However, such approach most likely will not scale well for a large network with hundreds of thousands of end hosts.
- A random subset of edge nodes: We can take the measurement at some of the edge nodes, such as the shaded nodes shown in Figure 25(c). However, this approach will require some techniques to infer traffic at places where measurement is not available. We will describe some possible approach to meet this challenge later.

Second, how to merge traffic? how does traffic collectors communicate and coordinate with each other? We propose the following scenario as one of the possible approaches to tackle this problem:

- Assuming the traffic at each node can be categorized into the following: traffic originates from this node, traffic destines to this node and transient traffic.

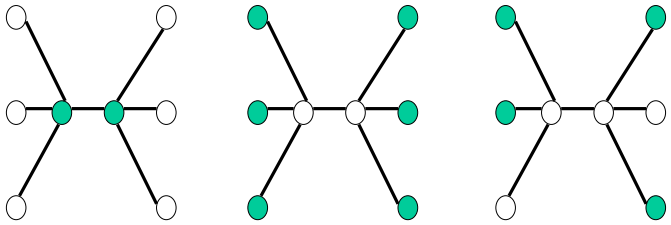


Figure 25: Some possible locations to take measurement

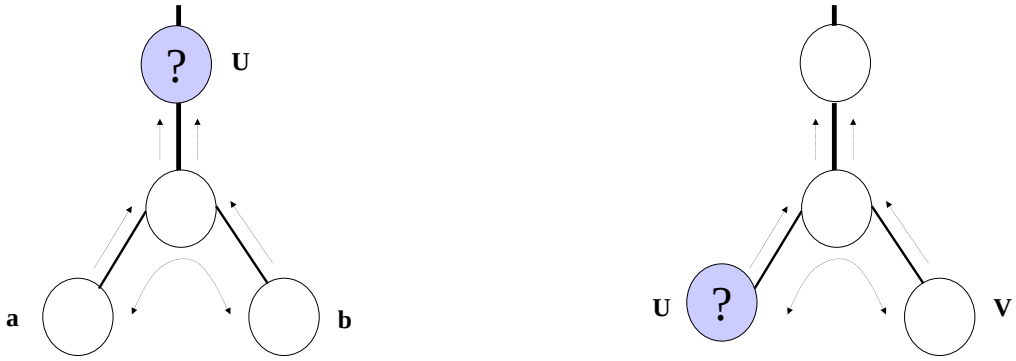
- We first classify traffic into source/destination groups based on the IP prefix.
- Each data collection agent is responsible for a set of groups and forwards the information of the other groups to their responsible collectors, assuming network topology information can be obtained via the help of some existing software[30, 15, 10].
- All the data agents join a well-known multicast session to exchange information (alternatively, we can use unicast or peer-to-peer protocol to exchange information when multicast is not available). There are some open issues regarding this approach. For example, how do we classify traffic into groups and assign to each data collection agent? How often and what information should be exchanged among the data collection agents? We plan to answer these questions as future work.

Finally, how to infer traffic where taking measurement is infeasible if not impossible? In particular, we are interested in how to infer the traffic at the center of network based on measurements at a limited subset of edge nodes, as the case shown in Figure 25(c).

To make the problem tractable we first assume the network can be simplified into a tree topology, as shown in Figure 26, and the node U is the place where measurement can not be taken. Therefore, node U will be either a root/internal node or leaf node. If U is a root node, as shown in Figure 26(a), then an estimation of traffic at U can be obtained by

$$TRAFFIC_U = TRAFFIC_a + TRAFFIC_b - CROSSTRAFFIC_{a,b} \quad (5)$$

The real challenge arises when node U is the leaf node, as shown in Figure 26(b). We then need some ways to infer traffic at U based on the measurement taken at its neighboring node V. As a first step, we will undertake



(a) Node U is an internal node

(b) Node U is a leaf node

Figure 26: A simple tree topology

a feasibility study to understand under what conditions traffic is correlated between nodes that are close to each other (for example, would two nodes have similar traffic distribution if they have similar user population or similar content distribution etc). If it is feasible, we plan to develop algorithm and tool to infer the traffic of a leaf node based on the measurements taken from the other nodes. Otherwise, we intend to explore and understand what kind of support (active probing or passive monitoring) needs to be in place to provide a infrastructure for distributed measurement.

5.2 Validation

To understand if our results correctly reproduce the traffic, we plan to compare our models against measurements and see if they match closely (within some error bounds). We will examine the traffic statistics at three different levels:

- single node: We will look at the inbound and outbound traffic for each single node, and compare our models against traces via similar statistical techniques used in RAMP (i.e. comparison of first-order statistics and multi-scaling analysis)
- cross traffic among nodes: For example, does traffic received by B (from A and C) match the sum of traffic sent by A and traffic sent by C?
- network-wide view of traffic: For example, does the model correctly characterizes the total number of unique flows, traffic mix and other traffic statistics of the whole network?

5.3 Tasks

We plan to explore and understand how to integrate measurements from multiple points by proceeding the following tasks:

- **Task 1**: Collect traces from multiple points of the network (the target network is either ISI or los nettos network) and understand if there is any temporal or spatial correlations between traffic seen at different places.
- **Task 2**: Develop methodology and infrastructure that allow one to infer traffic at places where measurement is not avaialble.
- **Task 3**: Implement data collection agent and design the protocol used by the agents to communicate with each other.
- **Task 4**: Develop methodology and tool that converts collected data into traffic model. In particular, the model will include the following statistics: characterization of traffic sources, how the data sent by each traffic source is distributed into the network and characterization of the network topology.
- **Task 5**: Validate the model by comparing the simulation results against the traces.

6 Thesis Plan

To date, we have

- demonstrated the importance of using structural modeling to model traffic and developed a structural model for RealAudio traffic, evaluated through simulation and validated via multi-scale analysis.
- showed Internet traffic not only changes over time but also exhibits great variations in other dimensions such as locations and flow directions. Based on structural modeling approach, we have developed methodology and tool that rapidly parameterize traffic models from live measurement. Currently our tool supports web and FTP traffic.

As future work, we plan to

- **Task 1**: collect more traces from other places, particularly those that potentially have very different traffic characteristics (for example, an OC3 link trace) to further validate RAMP.
- **Task 2**: collect traffic from multiple points of the network, and understand if there is any correlation in the traffic among different vantage points.
- **Task 3**: Develop methodology and infrastructure that allow one to infer traffic at places where measurement is not available.
- **Task 4**: Implement data collection agent and design the protocol used by the agents to communicate with each other.
- **Task 5**: Develop methodology and tool that converts collected data into traffic model. In particular, the model will include the following statistics: characterization of traffic sources, how the data sent by each traffic source is distributed into the network and characterization of the network topology.
- **Task 6**: Validate the model by comparing the simulation results against the traces.

Phase I (0-4 months)

- Utilize more traces to further validate RAMP
- Collect traces from different parts of the network (the target network is either ISI or los nettos network).
- Analyze the data and investigate if there is any correlations between traffic seen at different points of the network
- Develop methodology and infrastructure that allow one to infer traffic at places where measurement is not availalble.

Phase II (4-8 months)

- Implement data collection agent and design the protcol used by the agents to communiate with each other.
- Develop methodology and tool that converts collected data into traffic model.

Phase III (8-12 months)

- Validate the model via simulation
- thesis write-up.

7 Conclusion

Floyd and Paxson [27] provide an excellent overview of the problem, namely the constantly-changing and decentralized nature of the Internet, that results in a poor understanding of traffic characteristics and makes it difficult to define a typical configuration for simulating the Internet. Motivated by the their observations, we develop tools and methodology that support rapid parameterization of structural traffic models from passive measurements. We plan to extend the techniques we have exploited and develop algorithm that integrate data collected from multiple points into an coherent application-level traffic model.

References

[1] P. Abry and D. Veitch. Wavelet analysis of long-range-dependent traffic. *IEEE Transactions on Information Theory*, 44(1):2–15, 1998.

[2] Heejune Ahn, Jae kyoon Kim, Song Chong, Bara Kim, and Bong Dae Choi. A video traffic model based on the shifting-level process: the effects of SRD and LRD on queueing behavior. In *INFOCOM (2)*, pages 1036–1045, 2000.

[3] Hari Balakrishnan, Venkata N. Padmanabhan, and Randy H. Katz. Tcp behavior of a busy internet server: Analysis and improvements. In *Proceedings of the IEEE Infocom*, San Francisco, CA, USA, March 1998. IEEE.

[4] Paul Barford and Mark Crovella. Generating representative web workloads for network and server performance evaluation. In *Proceedings of the ACM SIGMETRICS*. ACM, 1998.

[5] J. Beran, R. Sherman, M. S. Taqqu, and Walter Willinger. Long-range dependence in variable-bit-rate video traffic. *IEEE Trans. on Comm.*, 43:1566–1579, 1995.

[6] Henry Braun. A simple method for testing goodness of fit in the presence of nuisance parameters. *Journal of the Royal Statistical Society. Series B (Methodological)*, 42(1):53–63, 1980.

[7] 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*, 33(5):59–67, May 2000. Expanded version available as USC TR 99-702b at `http://www.isi.edu/~johnh/PAPERS/Bajaj99a.html`.

[8] R. Cáceres, C. J. Sreenan, and J. E. van der Merwe. mmdump—a tool for monitoring multimedia usage on the internet. submitted for publication, July 1999.

[9] CAIDA. Internet measurement infrastructure. `http://www.caida.org/analysis/performance/measinfra/`.

[10] CAIDA. Skitter. `http://www.caida.org/tools/measurement/skitter/`.

[11] J. Cao, D. Davis, S. Wiel, and B. Yu. Time-varying network tomography : Router link data. *The Journal of American Statistics Association*, 95(452):1063–1075, February 2000.

[12] J. Cao, Scott Vander Wiel, Bin Yu, and Zhengyuan Zhu. A scalable method for estimating network traffic matrices. *Bell Labs Tech. Report*, 2000.

[13] Jin Cao, William S. Cleveland, Dong Lin, and Don X. Sun. On the nonstationarity of internet traffic. In *SIGMETRICS/Performance*, pages 102–112, 2001.

[14] M. Chandra, N. D. Singpurwalla, and M. A. Stephens. Kolmogorov statistics for tests of fit for the extreme-value and weibull distributions. *Journal of the American Statistical Association*, 76(375):729–731, September 1981.

[15] B. Cheswick, H. Burch, and S. Branigan. Mapping and visualizing the internet. In *Usenix 2000 general conference*, June 2000.

[16] D. Cohen and D. Heyman. Performance modeling of video teleconferencing in atm networks. *IEEE Trans. Circuits and Sys. for Video Tech.*, 3:408–420, December 1993.

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

[18] P. Danzig, S. Jamin, R. Caceres, D. Mitzel, and D. Estrin. An empirical workload model for driving wide-area TCP/IP network simulations. *Journal of Internetworking: Research and Experience*, 3(1):1–26, March 1992.

[19] A. Erramilli, R. P. Singh, and P. Pruthi. An application of deterministic chaotic maps to model packet traffic. queueing systems. *Queueing Systems*, 20(3):171–206, 1995.

[20] C. Shim et al. Modeling and call admission control algorithm of variable bit rate video in atm networks. *IEEE Journal on Selected Areas in Communications*, 12, February 1994.

[21] D. Heyman et al. Modeling teleconference traffic from vbr video coders. *Proc. ICC, IEEE*, pages 1744–1748, 1994.

[22] Anja Feldmann. BLT: Bi-layer tracing of HTTP and TCP/IP. *WWW9 / Computer Networks*, 33(1-6):321–335, May 2000.

[23] Anja Feldmann, A.C. Gilbert, Walter Willinger, and T.G. Kurtz. The changing nature of network traffic: Scaling phenomena. *ACM Computer Communication Review*, 28(2):5–29, April 1998.

[24] Anja Feldmann, Anna Gilbert, and Walter Willinger. Data networks as cascades: Explaining the multifractal nature of internet wan traffic. In *Proceedings of the ACM SIGCOMM*, pages 42–55, Vancouver, B.C., Canada, September 1998. ACM.

[25] 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*, pages 301–313, Cambridge, MA, USA, August 1999. ACM.

[26] Sally Floyd and Van Jacobson. The synchronization of periodic routing messages. *IEEE/ACM Transactions on Networking*, 2(2):122–136, 1994.

[27] Sally Floyd and Vern Paxson. Difficulties in simulating the Internet. *ACM/IEEE Transactions on Networking*, xxx(xxx):to appear, February 2001.

[28] M. Garrett and W. Willinger. Analysis, modeling and generation of self-similar VBR video traffic. *Proc. ACM Sigcomm*, pages 269–280, September 1994.

[29] A. C. Gilbert. Multiscale analysis and data networks. *Applied and Computational Harmonic Analysis*, 10(3):185–202, May 2001.

[30] Ramesh Govindan, Cengiz Alaettinoğlu, and Deborah Estrin. Self-configuring active network monitoring (SCAN). White paper, February 1997.

[31] VINT group. UCB/LBNL/VINT network simulator—ns (version 2). `http://mash.cs.berkeley.edu/ns`.

[32] R. Gurenefelder, J. P. Cosmas, S. Manthrope, and A. Odinma-Okafor. Characterization of video codecs as autoregressive moving average processes and related queueing system performance. *IEEE Journal on Selected Areas in Communications*, 9:284–293, 1991.

[33] C. H., M. Devetsikiotis, I. Lambadaris, and A. Kaye. Self-similar modeling of variable bit rate compressed video: A unified approach. In *Proceedings of the ACM SIGCOMM*. ACM, 1995.

[34] H. Heffes and D. M. Lucantoni. A markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance. *IEEE Journal on Selected Areas in Communications*, 4:856–868, 1986.

[35] D. P. Heyman, A. Tabatabai, and T. Lakshman. Statistical analysis and simulation study of video teleconference traffic in atm networks. *IEEE Trans. Circ. and Sys. Video Tech.*, 2:49–59, March 1992.

[36] Polly Huang, Anja Feldmann, and Walter Willinger. A non-intrusive, wavelet-based approach to detecting network performance problems. In *Proceeding of ACM SIGCOMM Internet Measurement Workshop 2001*, San Francisco Bay Area, November 2001.

[37] Thorsten Krogh Iversen and Morten Lauersen. NNTPGen. *TR-00-002, Department of Computer Science, Aalborg University*, January 2000.

[38] A. Karasaridis and D. Hatzinakos. Network heavy traffic modeling using alpha-stable self-similar processes. *IEEE Trans. on Comm.*, 49(7):1203–1214, 2000.

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

[40] Marwan M. Krunz and Armand M. Makowski. Modeling video traffic using m—g—infinity input processes: A compromise between markovian and lrd models. *IEEE Journal on Selected Areas in Communications*, pages 733–748, 1998.

[41] Kevin Lai and Mary Baker. Measuring bandwidth. In *INFOCOM (1)*, pages 235–245, 1999.

[42] Kevin Lai and Mary Baker. Nettimer: A tool for measuring bottleneck link bandwidth. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, March 2001.

[43] S. Ledesma and D. Liu. A fast method for generating self-similar network traffic. In *Proceedings of the 2000 International Conference on Communication Technologies*, pages 54–61, Beijing, China, August 2000.

[44] 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*, 2(1):1–15, February 1994.

[45] Q. Liang and J. Mendel. Mpeg vbr video traffic modeling and classification using fuzzy technique. *IEEE Trans. on Fuzzy System*, 9(1):183–193, 2001.

[46] Hubert W. Lilliefors. On the kolmogorov-smirnov test for the exponential distribution with mean unknown. *Journal of the American Statistical Association*, 64(325):387–389, March 1969.

[47] D. Lucantoni, M. Neuts, and A. Reibman. Methods for performance evaluation of vbr video traffic models. *IEEE/ACM Trans. Networking*, 2:176–180, 1994.

[48] Sheng Ma and Chuanyi Ji. Modeling video traffic in the wavelet domain. In *INFOCOM (1)*, pages 201–208, 1998.

[49] B. Maglaris, D. Anastassiou, G. Karlsson P. Sen, and J. D. Robbins. Performance models of statistical multiplexing in packet video communications. *IEEE Trans. on Comm.*, 36(7):834–844, 1988.

[50] B. Mah. An empirical model of HTTP network traffic. In *Proceedings of the IEEE Infocom*, pages 592–600, Kobe, Japan, April 1997. IEEE.

[51] B. Mah, P. Sholander, L. Martinez, and L. Tolendino. IPB: An internet protocol benchmark using simulated traffic. In *Proceedings of MASCOTS '98*, Montreal, Canada, August 1998.

[52] F. J. Jr. Massey. The kolmogorov-smirnov test of goodness of fit. *Journal of the American Statistical Association, Vol. 46*, 1951.

[53] M. Mathis and J. Mahdavi. Diagnosing internet congestion with a transport layer performance tool. In *Proceedings of INET '96*, Montreal, June 1996.

[54] Ashraf Matrawy, Ioannis Lambadaris, and Changcheng Huang. Mpeg4 traffic modeling using the transform expand sample methodology. *The IEEE 4th International Workshop on Networked Appliances (IWNA4)*, October 2001.

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

[56] Larry McVoy and Carl Staelin. lmbench: Portable tools for performance analysis. In *USENIX Conference Proceedings*. USENIX, January 1996.

[57] B. Melamed and P. Jelenkovic. Automated tes modeling of compressed video. In *Proceedings of the IEEE Infocom*, pages 746–752. IEEE, 1995.

[58] B. Melamed and D. Pendarakis. A tes-based model for compressed star wars video. In *Proceedings of IEEE GLOBCOM '94*, pages 120–126, 1994.

[59] B. Melamed and B. Sengupta. Tes modeling of video traffic, December 1992.

[60] Art Mena and John Heidemann. An empirical study of real audio traffic. In *Proceedings of the IEEE Infocom*, pages 101–110, Tel-Aviv, Israel, March 2000. USC/Information Sciences Institute, IEEE.

[61] G. Minshall. Tcpdpriv, http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html.

[62] S. Molnr and Gy. Terdik. A general fractal model of internet traffic. In *The 26th Annual IEEE Conference on Local Computer Networks (LCN)*, pages 15–16, Tampa, Florida, USA, November 2001.

[63] M. Thorup N.G. Duffield, C. Lund. Charging from sampled network usage. In *Proceeding of ACM SIGCOMM Internet Measurement Workshop 2001*, San Francisco Bay Area, November 2001.

[64] I. Nikolaidis and I. Akyildiz. Source characterization and statistical multiplexing in atm networks. *Tech. Rep. GIT-CC 92-24, Georgia Tech.*, 1992.

[65] Ilkka Norros. On the use of fractional brownian motion in the theory of connectionless networks. *IEEE Journal of Selected Areas in Communications*, 13(6):953–962, 1995.

[66] M. Parulekar and A. M. Makowski. Buffer overflow probabilities for a multiplexer with self-similar traffic. In *Proc. INFOCOM'96*, pages 970–977, San Francisco, CA, 1996.

[67] V. Paxson. Fast, approximate synthesis of fractional gaussian noise for generating self-similar network traffic, October 1997.

[68] Vern Paxson. Internet Traffic Archive, http://www.acm.org/sigcomm/ita/.

[69] Vern Paxson. Empirically derived analytic models of wide-area TCP connections. *IEEE/ACM Transactions on Networking*, 2(4):316–336, 1994.

[70] Vern Paxson. Measurements and analysis of end-to-end internet dynamics. In *Ph.D. thesis, University of California, Berkeley*, April 1997.

[71] Vern Paxson and Sally Floyd. Wide-area traffic: the failure of Poisson modeling. *ACM SIGCOMM 94*, pages 257–268, 1994.

[72] J. Postel and J. Reynolds. Rfc959: File transfer protocol (FTP). http://www.ietf.org/rfc/rfc959.txt, October 1985.

[73] RealNetworks. Realnetworks documentation library. `http://service.real.com/help/library/`.

[74] S. Resnick, G. Samorodnitsky, A. Gilbert, and W. Willinger. Wavelet analysis of conservative cascades. *TR-99-1243, School of Operations Research and Industrial Engineeringi, Cornell University*, June 1999.

[75] Vinay J. Ribeiro, Rudolf H. Riedi, Matthew S. Crouse, and Richard G. Baraniuk. Simulation of nonGaussian long-range-dependent traffic using wavelets. In *Measurement and Modeling of Computer Systems*, pages 1–12, 1999.

[76] B. Ryu and S. Lowen. Point process models for self-similar network traffic. *Stochastic Models*, 1997.

[77] P. Sen, B. Maglaris, N.-E. Rikli, and D. Anastassiou. Models for packet switching of variable-bit-rate video sources. *IEEE Journal on Selected Areas in Communications*, 7(5):865–869, 1989.

[78] SETI@home. Search for extraterrestrial intelligence at home. `http://setiathome.ssl.berkeley.edu/`.

[79] N. Smirnov. Table for estimating the goodness of fit of empirical distributions. *Annals of the Mathematical Statistics*, 19(2):279–281, June 1948.

[80] F. Donelson Smith, Felix Hernandez Campos, Kevin Jeffay, and David Ott. What TCP/IP protocol headers can tell us about the web. In *SIGMETRICS/Performance*, pages 245–256, Cambridge, MA, June 2001.

[81] Y. Vardi. Network tomography : Estimating source-destination traffic intensities from link data. *The Journal of American Statistics Association*, 91(433):365–377, March 1996.

[82] W. Willinger, V. Paxson, and M. Taqqu. Self-similarity and heavy-tails: Structural modeling of network traffic. In *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. Birkh auser, Boston.*, 1998.

[83] Walter Willinger, Murad S. Taqqu, Robert Sherman, and Daniel V. Wilson. Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level. *IEEE/ACM Transactions on Networking*, 5(1):71–86, 1997.

[84] www.broadcast.com. Broadcast.com. `http://www.broadcast.com`.

[85] M. Yuksel, B. Sikdar, K. S. Vastola, and B. Szymanski. Workload generation for ns simulations of wide area networks and the internet. In *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS) part of Western Multi-Conference (WMC)*, pages 93–98, San Diego, CA, 2000.

[86] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the constancy of internet path properties. In *Proceeding of ACM SIGCOMM Internet Measurement Workshop 2001*, San Francisco Bay Area, November 2001.

[87] Yonggang Jerry Zhao, Ramesh Govindan, and Deborah Estrin. Residual energy scans for monitoring wireless sensor networks. In *IEEE Wilress Communications and Networking Conference (WCNC'02)*, Orlando, FL, USA, March 2002.