Generation of High Bandwidth Network Traffic Traces

Purushotham Kamath, Kun-chan Lan, John Heidemann, Joe Bannister and Joe Touch
University of Southern California
Information Sciences Institute
Los Angeles, California, U.S.A.
{pkamath, kclan, johnh, joseph, touch}@isi.edu

Abstract

High bandwidth network traffic traces are needed to understand the behavior of high speed networks (such as the Internet backbone). However, the implementation of a mechanism to collect such traces is difficult in practice. In the absence of real traces, tools to generate high bandwidth traces would aid the study of high speed network behavior. We describe three methods of generating high bandwidth network traces: scaling low bandwidth network traffic traces, merging multiple low bandwidth traces and generating traces through simulation by scaling a structural model of real world traces. We evaluate the generated traces and discuss the advantages and disadvantages of each method. We also discuss some of the issues involved in generating traces by the structural model method.

1. Introduction

The behavior of a network depends to a large extent on the nature of the traffic generated by its users. Network protocols and switching mechanisms behave differently under different traffic patterns. Network protocols or switching mechanisms themselves may be the cause of different types of traffic patterns. Understanding network traffic patterns and their causes is central to network research on the Internet. This has lead to several efforts to collect traffic traces on the Internet [17] and to analyze them.

Most real world network traces that are publicly available [17,18] are low bandwidth traces from OC3c, OC12c links or FDDI or Ethernet traces. In contrast to the wide availability of low bandwidth traces, traces from high bandwidth links (OC48, OC192 links, such as in the core of the Internet) are not widely available. The difficulty of implementing a mechanism for collecting traces at high speeds [12] is one factor that contributes towards the lack of public availability of such traces. As link speeds increase in the future, the difficulties involved in obtaining high bandwidth traces will increase.

In the absence of real high bandwidth traces, one option is to attempt to generate traces that are likely to resemble real traces. Applications of such traces include studies of the behavior of routers, switches and network protocols on high speed links.

Prior studies [22] have suggested that traffic characteristics differ widely depending on where and when the data was recorded. Given such differences in traffic characteristics, there is reason to believe that traffic seen in the core of the Internet (high bandwidth traces) may differ from traffic seen at the edges or on a local area network (low bandwidth traces). Earlier studies have indicated that traffic on an Ethernet network is self similar [14] in nature. Recent studies [3] indicate that Internet traffic tends to

Purushotham Kamath is supported, and Joe Bannister and Joe Touch are partially supported as part of the NGI Multicast Applications and Architecture (NMAA-ADDON) project funded by the Defense Advanced Research Projects Agency Information Technology Office, DARPA Order No. H645, Program Code No. 9A20, issued by DARPA/CMD under Contract No. MDA972-99-C-0022.
Kun-chan Lan and John Heidemann are partially supported in this work as part of the SAMAN project, funded by DARPA and the Space and Naval Warfare Systems Center San Diego (SPAWAR) under Contract No. N66001-00-C-8066.
John Heidemann is also partially supported as part of the CONSER project, funded by NSF as grant number ANI-9986208.
Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA or SPAWAR.

Poisson and independent as the load increases, as it does in the core of the Internet. Thus at this point of time, the characteristics of high bandwidth traffic are still under investigation.

In this paper we describe three methods of generating a high bandwidth trace and evaluate the generated traces. As implied above, the major hurdle that we face in validating this effort is the lack of a real world high bandwidth trace to compare our generated traces with. Nevertheless, it is worthwhile to compare real world low bandwidth traces with the high bandwidth traces generated by the three methods to determine how different high bandwidth traffic could be from low bandwidth traffic. The main contribution of this paper is the comparison of the network traffic traces generated by the three methods and a discussion of the issues involved in the generation by the structural model method.

In Section 2 we discuss the challenges involved in generating accurate network traffic traces in general and high bandwidth traces in particular. Section 3 discusses some of the parameters of interest that can be measured in a network trace file and the metrics for doing so. We justify limiting our choice to a few metrics. We describe the three methods of trace generation in Section 4 and discuss which of the parameters each method could be expected to replicate accurately. We also discuss possible reasons why each method should or should not generate accurate traces. Section 5 compares the traces generated by the three methods. In Section 6 we discuss some issues involved in generating traces by the structural model method. These include the difficulties in running an accurate high bandwidth simulation. Section 7 describes some related work and Section 8 summarizes the conclusions of our work.

2. Challenges in generating high bandwidth traces

The trace files that we generate consist of the following fields for each packet:

- Timestamp (time when the packet was received at the point where it is being traced)
- Source IP address, destination IP address and source and destination TCP/UDP port numbers
- Packet size

Generating each of the above fields (for low bandwidth or high bandwidth traces) presents several challenges which we discuss below:

2.1 Timestamp

The timestamp of a packet indicates how busy the link is. The timestamps can indicate the bursty nature of traffic on the link. There are several factors that affect the timestamp of a packet. The factors listed below affect the timestamp of a packet whether it traverses low or high bandwith links or a combination of them. The challenge in generating timestamps for a high bandwidth trace is in deciding if these factors will affect the traces and cause them to differ from traces on a low bandwidth link. Another challenge is how to incorporate these effects in the generation method to ensure that the generated traces are accurate.

- Link characteristics: The bandwidth of a link determines the transmission time of the packet. Both the transmission time and the propagation delay of the link contribute to the round trip time (RTT) of the connection. With a transport protocol such as TCP, the RTT determines when the next packet can be sent and hence its timestamp.

- Switch/router characteristics: The queueing delay at the network switches also contributes to the RTT of the connection. The drop policy at the switch also influences packet transmissions. With a transport protocol such as TCP, the congestion control algorithm reacts to packet drops and hence is dependent on the switch drop policy.

- Transport protocol characteristics: The transport protocol used (TCP) influences packet transmission times through the congestion control algorithm.[1]

- Application protocol characteristics: HTTP influences the packet transmission times through the number of web pages requested, the number of objects per page and the size of each object. The use of persistent and pipelined connections [11] also affects the timestamp of each packet.

- User characteristics: The user arrival rate and the user think times also affect the timestamps of the packets by controlling how often a user sends requests.

Figure 1 shows the factors can impact the timestamp of a packet.

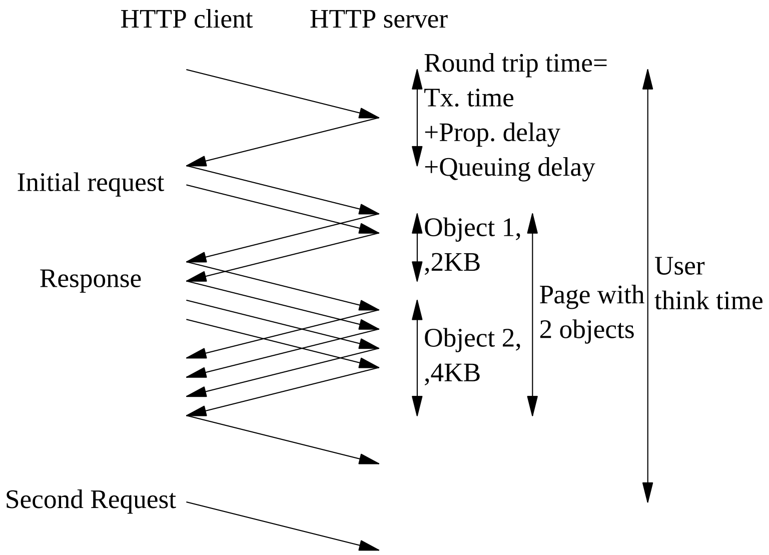

Figure 1: Factors affecting the packet timestamps

2.2 Addresses

The source address and destination address of a packet depend on the number of clients and servers, the user arrival rate at each client and how busy a server is. A high bandwidth link may see more addresses than a low bandwidth link but the increase in addresses seen may not be proportional to the increase in bandwidth. E.g. 90% of the traffic may originate from 10% of the clients or may be destined for 10% of the servers (it is known that a small percentage of end host flows contribute to a large percentage of the traffic [9]).

2.3 Packet size

The packet size distribution depends on the number of requests and the file sizes requested. With web traffic the packet sizes usually vary from 40 bytes (connection setup packets) to 1500 bytes (the maximum transmission unit (MTU) is the minimum transmission unit on the path and Ethernet is usually on at least one segement of most paths. Hence most data packets tend to be 1500 bytes long). A high bandwidth link will see a larger number of packets per unit time than a low bandwidth link, but the packet size distribution should remain the same.

3. Evaluating the quality of the traces

Network traffic characteristics may be described by several metrics. The characteristics that are measured in this study may be divided into two categories.

Packet characteristics:

- Packet interarrival time distribution
- Packet sizes distribution
- Distribution of the number of unique destination addresses seen in a time interval

Connection characteristics:

- TCP connection duration distribution
- Distribution of the number of bytes and packets in a TCP connection

These five metrics reasonably model IP traffic traces from the perspective of at least one application - that of studying network switch/router behavior. The packet interarrival time gives an indication of how fast a router needs to process packets. The packet size affects the buffer space needed as well as the packet transmission time at the output port. The destination address determines where the packet needs to be sent and the distribution of the number of unique addresses seen in an interval affect mechanisms such as routing table caches [10] used in a switch or router. The connection characteristics can affect how often a particular routing table entry is used. The TCP connection duration and the number of bytes and packets sent determine the temporal locality of addresses seen.

In this study, no analysis of the application characteristics (E.g. percentage of web, ftp, telnet traffic, user interarrival times etc.) has been done. Studies by Danzig et al. [7] describe how network and transport level statistics alone are inadequate to drive simulations of networks and how modeling application level characteristics allows the simulation of different traffic models (interactive, bulk etc.). The structural model method (discussed in section 4.3) uses this idea to generate realistic traces. However, we have made no attempt to analyze any application level characteristics in the generated traces. The nature of traffic on high bandwidth links is primarily influenced by multiplexing at switches and protocol characteristics. The mix of application traffic and its characteristics should remain unchanged. Thus, there is no reason to believe that there should be different application characteristics in a higher bandwidth trace.

4. Methods of generating high bandwidth traces

The simplest methods to generate a high bandwidth trace involve processing a low bandwidth trace. This is the methodology used in the first two methods that we describe. Another method of generating high bandwidth traces is to simulate a high speed link and extract traces during the simulation. The third method we describe is based on this methodology. In this section we describe these three methods and discuss how accurate we can expect the generated traces to be.

We have used the Auckland II trace set [18] as the low bandwidth trace set used to generate high bandwidth traces. The Auckland II traces are from a wide area network link with packet peak rate of 2Mbps in each direction. Bit rates (calculated over 100ms intervals) vary from 0.4Mbps to 8Mbps. The duration of the trace is around 11 hours. The trace consist of two trace files for traffic on each direction.

4.1 Scaling

A simple method to generate a high bandwidth trace is to *scale* a real world trace in time by dividing the timestamps in the trace by a constant, the *scaling factor, s.* Most common real world traces are either bidirectional traces such as an Ethernet trace or a pair of unidirectional traces from a wide area link such as the Auckland [18] data set. This method can be applied to either form of trace files.

If the original traffic trace is from a 10 Mbps link and we want to generate a 1Gbps trace file, then the scaling factor

$$s = 1Gbps/10Mbps = 100$$

Each timestamp in the original trace file *Torig* is replaced in the generated trace file by a new timestamp *Tnew* where

$$Tnew = Torig/s$$

All other records in the generated trace file (source and destination addresses, packet sizes) are the same as in the original trace file.

The strongest justification for applying this method is that high bandwidth network traffic is a multiplexing of traffic from several low bandwidth links. If the packet arrivals of *n* streams on a low bandwidth link are independent and the packets interarrival times are uniformly distributed, then the resulting aggregated trace should have an arrival rate equal to approximately *n* times the arrival rate of the low bandwidth links.

There are several flaws in this reasoning. Packets on a TCP connection will not have uniformly distributed interarrival times due to structural effects such as the dependence of packet transmission on RTT and user think times. The presence of a faster link does not imply that the propagation delay would cahnge or that a user would read a web page faster.

In addition, the multiplexing of traffic from several links may have other effects such as

- Queueing delays at switches/routers
- Packet drops at switches/routers (which depend on drop policies such as Drop Tail, RED etc.) which in turn affect the TCP congestion control algorithm (resulting in window size changes, RTT reestimation and RTO recalculation).

It is unlikely that these effects will be accurately modeled by the scaling method.

Another disadvantage of this method is that the low bandwidth trace used for generation needs a long duration trace (at least *s* times the duration of the desired high bandwidth trace). A possible solution to this is to concatenate several low bandwidth traces and then scale. However, this results in abrupt changes in traffic patterns at the points in time where the traces were concatenated.

4.2 Merging

In the *merging* method several real low bandwidth traffic traces are merged. The merging is done by simulating the multiplexing that happens at an output queued switch. The merging is done by a switch simulator. The switch simulator consists of a single switch using a FIFO queue with tail drop policy. The traces files are fed as input to the switch simulator. The timestamps in the input trace files determine the time when each packet is placed in the output queue. The switch simulator serves the queue using a FIFO queueing discipline and send packets out on the output link. Each packet suffers a queueing delay that depends on the number of packets in the queue. Each packet requires a transmission time that depends on the packet size and the bandwidth of the output link. The trace data (timestamp, source and destination addresses, packet size) for the packet is logged in a file before it is sent out. The number of files being merged is called the *merging factor, m.*

This method attempts to correct the deficiency of the *scaling* method by introducing a queueing mechanism which simulates the queueing delays. It simulates the queueing delays at a single switch and not the queuing delay that would be seen in a network of switches. Unlike the *scaling* method, merging does not significantly distort structural effects such as user think times and RTTs.

As with the *scaling* method, this method does not take into account the effect of packet drops or the TCP congestion control algorithm into account. Any effects due to TCP congestion control that were present in the original traces are retained in the generated trace. However, the effect of new packet drops at the switch on the TCP congestion control algorithm are not taken into account.

The method works with undirectional traces only. Since it simulates a switch, bidirectional traces cannot be used as input. To generate bidirectional traffic, each direction will need to be generated separately. Doing this means that the traffic in opposite directions (data and their acks) may not be correlated. Because timestamps of data and acks are changed independently, it is possible that the traffic generated in the two directions may be traffic that could not have been generated in reality.

To generate a trace file of bandwidth *m* times the original trace bandwidth, it is necessary to have *m* low bandwidth trace files. In practice this may not always be feasible. A solution to this is to split a long low bandwidth trace file into *m* shorter trace files and merge them. A disadvantage of this approach is that it may result in the distortion of TCP connections which span the points in time where the file was split - packets that are sent well into the connection may now appear at the start of the connection, or even before the connection was established. A solution to this problem is *renumbering* - destination addresses in the split trace files are renumbered. This alleviates the problem of the distortion of the TCP connection statistics - packets on different input interfaces of the switch simulator all have unique destination addresses and hence belong to different TCP connections. However, this mechanism distorts the destination address statistics.

4.3 Scaling structural model parameters through simulation.

To generate traffic traces through simulation we employ a *structural modeling* approach [23] Traditional black box traffic modeling approaches focus on employing complex time-series analysis to model network traffic. These models ignore 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 proposes that we should explicitly take into account the complex hierarchical structure of application and intertwined networking mechanisms in order to accurately reproduce the traffic. Our structural model [13] attempts to model user and application characteristics [21] and network mechanisms by deriving information from low bandwidth TCP level network traces. Around 50-70% of traffic on the Internet today [6] is web traffic. Based on this observation and on the data obtained from our real world trace files, only user characteristics of web traffic have been modeled.
Our implementation models user characteristics of web traffic only.

The procedure is described below:

1. Cumulative distribution functions (CDFs) for different parameters are derived from a real world trace. The parameters can be divided into three categories: user behavior (user interarrival times), web page characteristics (number of pages per session, number of objects per page), object characteristics (object size, inter object interval).

2. The structural model based on these parameters is described below:

   User behavior:
   • User interarrival times are chosen randomly based on the user interarrival CDF.

   Web page:
   • The number of pages per user session is chosen randomly based on its CDF.
   • The sources of the pages are chosen based on a server popularity CDF.

   Object:
   • The number of objects within one page is chosen based on an object CDF.
   • The size of the objects in a page are chosen based

on an object size CDF.

- A TCP connection is used for multiple request/response exchanges or a single request/response exchange based on the probability of persistent connections (HTTP1.1) versus non-persistent connections (HTTP1.0) as computed from the trace. In persistent connection mode, all objects within the same page are sent via the same TCP connection.[11]
- The TCP window size for both servers and clients are also randomly chosen from a CDF

3. This structural model is used to drive a network simulation from which traces are gathered.

In addition to the structural model of user behavior a suitable network topology must also be chosen. Simulating a backbone network topology is a difficult task. Simulating the entire topology along with traffic sources at each node strains [19] the available computing resources. The topology chosen should be such that it should be easy to increase the amount of traffic on the link on which the traffic trace is recorded. Hence, the backbone topology has been simplified to a dumbbell topology as shown in Figure 2 with clients and servers on either side of the bottleneck link. A packet from a client to a server traverses four router nodes. The traffic traces are collected on the bottleneck link. To assign link latencies, the round trip times are determined from the original traffic trace [13] and a cumulative distribution function is generated. Link latencies are assigned to client links from this distribution.

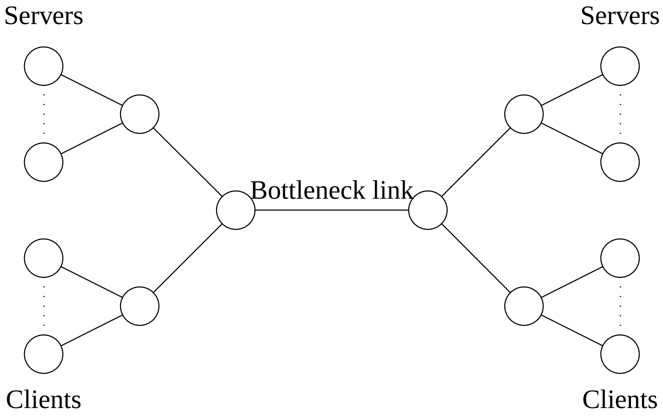

Figure 2: Simulation topology

Studies [4] have shown that the host pairs increases as the square root of the bit rate. Since the scaling factor of the simulation was 100, the number of servers and clients was chosen to be approximately 10 times the number of server and client IP addresses found in the Auck II trace.

To obtain higher bandwidth traces, the link bandwidths were increased linearly by a factor called the *simulation scaling factor* and the user interarrival rates were decreased linearly by the simulation scaling factor.

In contrast to the other two methods, this method will accurately model the queueing delays of a network and mechanisms such as the TCP congestion control behavior.

The main disadvantage of this method is that generating traces of higher bandwidth consumes an increasing amount of resources. In order to generate higher bandwidth traces using a simulator such as NS[2] , the size of the simulation (number of users and traffic) must be increased, which consumes a large amount of resources in the form of memory and processor cycles.

The structural model method represents a challenge in terms of the resources [20] required to run the simulations. Section 6 discusses this issue in greater detail.

5. Evaluation of the generated traces

As described in the introduction, the characteristics of high bandwidth traces are unknown. Therefore evaluation of each method to determine which one generates the most accurate trace is important. The traces generated by each method are evaluated by comparing the cumulative distribution functions (CDFs) of the generated traces with those of the original traces.
Figures 3 through 8 compare the traces generated by the three methods with the original traces. The original (or real) trace file was from the Auck II trace set (as described in Section 4). The duration of the original trace used was around 11 hours.

The three methods *(scaling, merging* and *structural modeling)* attempt to scale the traces by a factor of 100. The original trace was from a 10Mbps link and the goal was to generate traffic on a 1Gbps link. For the *scaling* method the scaling factor was 100. Since the original trace file was around 11 hours long, the generated trace file was around 6 minutes long.

The *merging* method used a merging factor of 100. Since it was not feasible to acquire 100 traces files of 5 minute duration from different sites, a single trace file of 11 hours was divided into 100 trace files of approximately 6

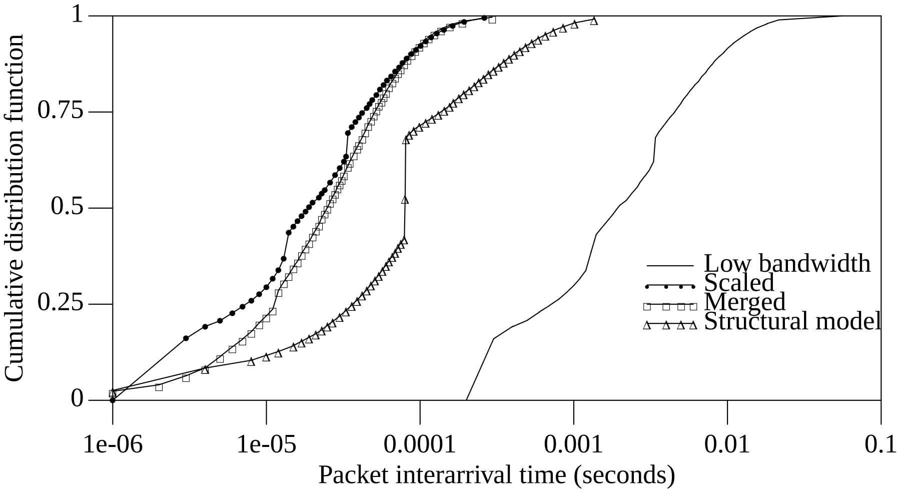

Figure 3: Packet interarrival time CDF

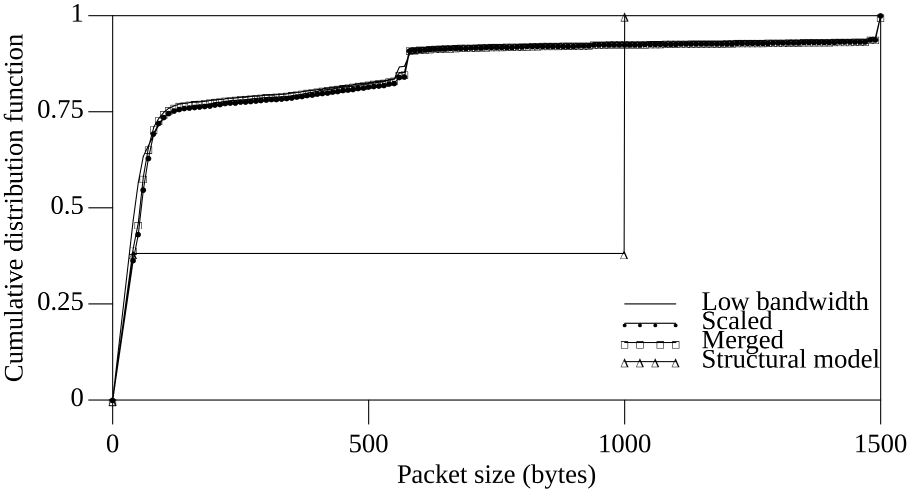

Figure 4: Packet size CDF

minutes each. The timestamps in each trace file were normalized to start at a time of zero seconds. The destination addresses in each trace file were renumbered to prevent distortion of the TCP connection statistics. However while this prevented a distorted in statistics such as connection duration, it may have contributed some distortion in the form of a larger number of destination addresses than would be normally seen.

In the case of the *structural modeling* method, the scaling factor was 100. User characteristics were extracted from 360 seconds of the trace. Around 1000 seconds of time was simulated (to allow the system to reach steady state) and 360 seconds of trace was extracted.

5.1 Packet interarrival time

Figure 3 shows the CDF of the packet interarrival time for the four traces. From the graph it can be observed that the packet interarrival time for the scaled trace has been reduced by the *scaling factor*, shifting the location of the CDF. The shape of the curve is identical to the the real trace. The location of the packet interarrival time for the merged traces has been reduced by the *merging factor.*

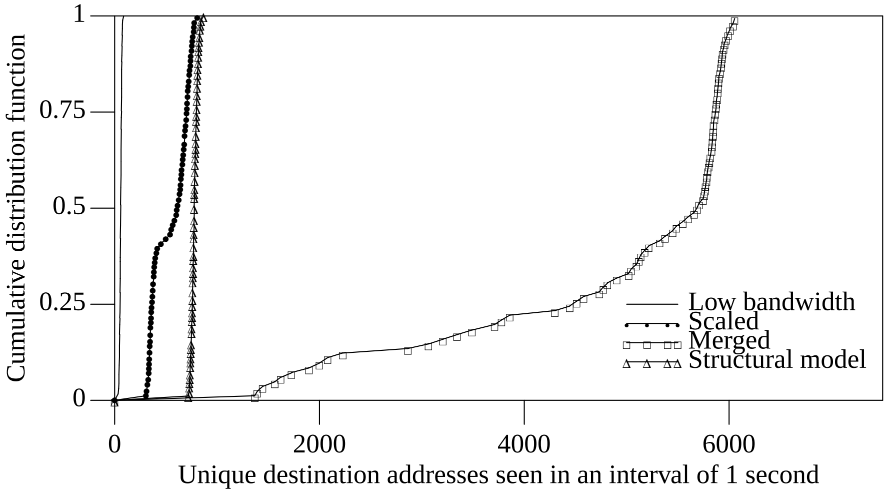

Figure 5: Unique destination addresses in an interval of 1 second (CDF)

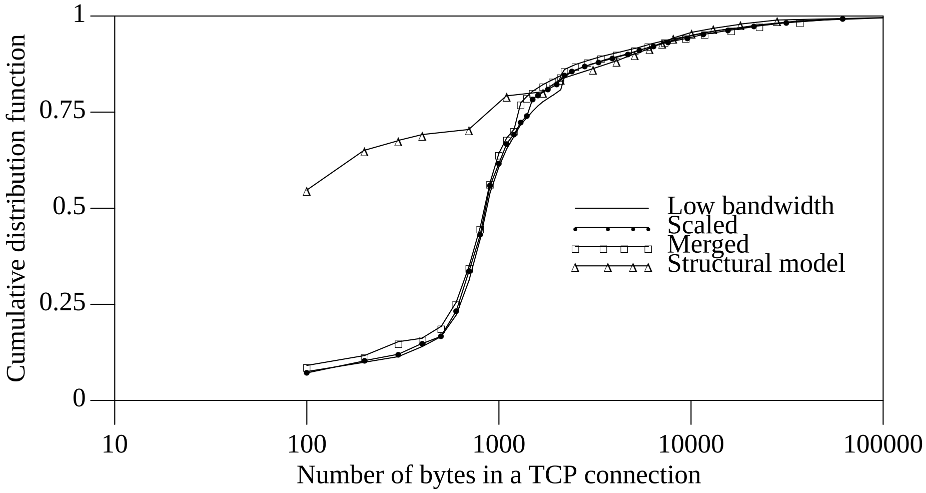

Figure 6: TCP connection bytes CDF

The location of the graph has been shifted to a location very close to the plot for the scaled traces. The shape of the graph is much smoother than the real traces, due to the effect of queueing delays while merging. The shift in the location of the CDF for the *structural model* generated traces is lower than that observed for the scaling and the merging methods which indicates that scaling the user interarrival times has not necessarily scaled the packet interarrival times proportionately. The shape of the graph is closer to the real traces than the traces produced by merging. This indicates that the structual model has represented packet interarrivals more accurately than the merging when both are compared to the original traces.

5.2 Packet size

Figure 4 shows the CDFs of the packet size for the four traces. The real trace and the *scaled* traces have identical CDFs, and are indistinguishable on the graph. The *merging* method gives a packet size distribution which very closely matches that of the real traces. The *structural model* method however has a CDF which is considerably different from the real traces. This occurs for several reasons, some relating to abstractions in the simulation model. First, it is well known that TCP packet size distributions are strongly bimodal, with a large number of ACK packets around 40 bytes and data packets around the MTU

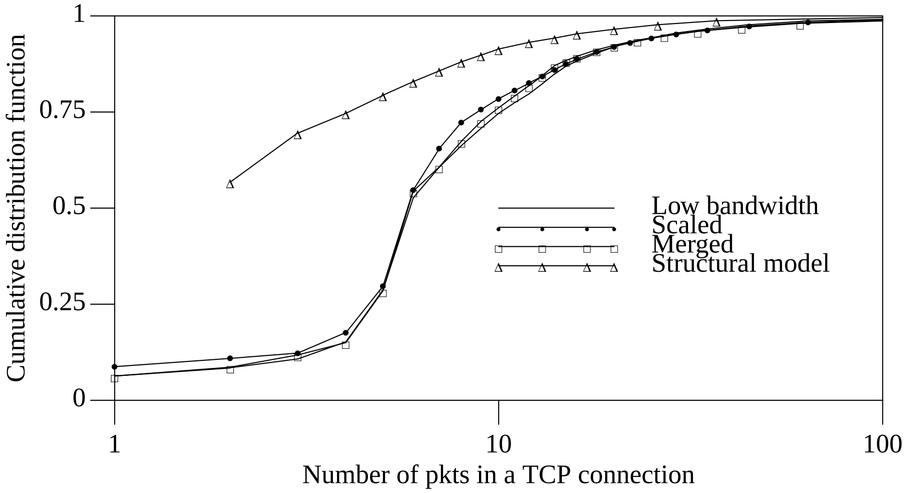

Figure 7: TCP connection packets CDF

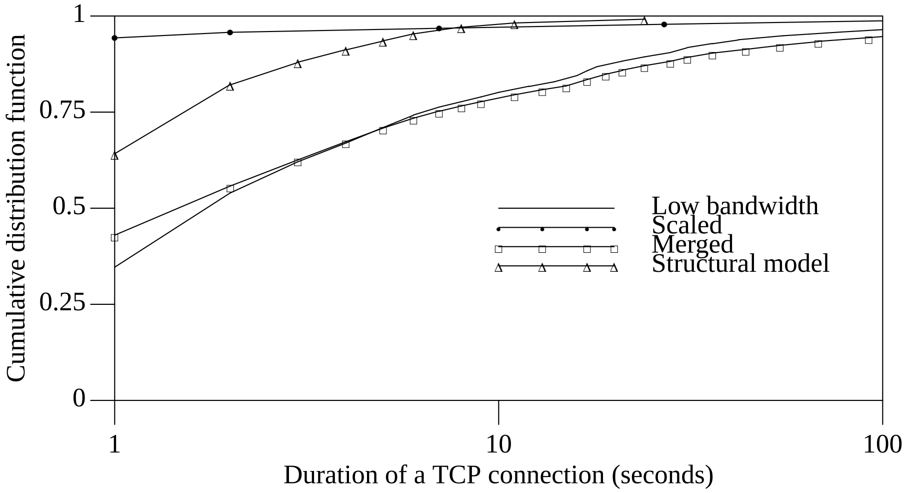

Figure 8: TCP connection duration CDF

size, typically either 540 bytes or 1500 bytes. The simulator's default data packet size is 1000 bytes, half way between the two MTUs observed in the trace. Second, the simulator does not try to model byte-level details of web traffic, but instead rounds transfers to an even number of whole data packets. Thus the traces show some variation in packet length (showing a curve in the 40-500 bytes range and slight slope in the 500-1500 bytes range), while the simulation is completely bimodal (40 or 1000 bytes). Finally, while the simulation model only supports web traffic (which accounts for a large percentage of the total traffic in the trace), the trace contains a variety of protocols with various packet lengths. A large percentage of small packets (those in 40-500 bytes range) are contributed by UDP traffic and protocols such as telnet, ftp etc. Each of these differences could be rectified by a more detailed traffic model in simulation. However, we found that these differences did not have much effect on aggregate traffic patterns

5.3 Destination addresses

Figure 4 shows the CDF for the number of unique destination addresses seen in an interval of 1 second. Studies of traffic locality in the NSFNET backbone [5] indicate that a large percentage of the traffic is destined for a small percentage of hosts. Other studies [4] indicate that the number of host pairs increases as the square root of the bit rate. These effects should be considered when evaluating the generated traces.

The diversity of the destination addresses (number of unique addresses) from the *scaled* traces has remained unchanged but the distribution across time (number of unique addresses seen in an interval) has changed. As seen from the graph the scaling of traces does not increase the number of destination addresses seen in a interval linearly by the scaling factor. The reason for this is because a small percentage of destination addresses account for a large percentage of traffic. Therefore, although more packets are seen in an interval of time, the number of unique destination addresses seen does not increase by the same factor.

In the case of *merging*, the destination address distribution has scaled linearly. This effect is primarily due to how the files were merged. The trace was generated by merging 100 trace files. Since 100 trace files of suitably long duration (360 seconds) were unavailable, the 100 trace files were generated by splitting a long duration trace file (around 11 hours). To avoid distorting the TCP connection statistics, the destination addresses in each of the 100 trace files were renumbered. As a result, the destination address CDF has scaled linearly by a factor of 100.

From the *structural model* plot it can be seen that the destination addresses seen in an interval has been scaled by a factor of around 10. As described in section 4.3, the number of destination hosts in the simulation was scaled by a factor of 10, to accurately model the system. Hence the address distribution has been scaled by about 10.

5.4 TCP connection packets/bytes

Figures 6 and 7 show the CDF for TCP connection bytes and packets. The TCP connection packets is the number of packets sent during a TCP connection (including the connection setup and teardown packets). The TCP bytes is the total number of bytes sent during a TCP connection (including connection setup and teardown packets). The TCP connection packets CDFs are almost identical. For the *scaling* and *merging* methods, the TCP connection bytes also closely approximate the real traces. The *structural model* method however has a much higher percentage of lower byte connections. This is because the structural model is based on user characteristics derived from web traffic only. It has neglected all other types of traffic such as telnet or ftp traffic which may have higher byte connections. In addition the simulator was run in a mode where it did not send connection setup and teardown packets.

5.5 TCP connection duration

Figure 8 shows the TCP connection duration. The TCP connection duration can be influenced by several factors - the bottleneck bandwidth of the connection, the propagation delay, protocol characteristics (as in the case of web session with persistent connections (HTTP 1.1) by the number of objects in a page) or by user characteristics. One of the effects of the *scaling* is that these characteristics will not be retained. E.g. a telnet session may last anywhere from a few minutes to a few hours while a persistent web connection may last a few seconds. Scaling the traffic compresses these periods. As seen in the graph, the connection duration of the traces generated by scaling have been reduced by the scaling factor. The shape of the curve however has been retained. For *merging* the connection duration should be affected slightly by the added queueing delay. However, the graph indicates a close match in the flow duration of the merged trace and the real trace, indicating that the queueing delay was minimal.

The *structural model* method has generated a large number of short duration flows. This is typical of web traffic and is due to extracting characteristics from only web traffic. In addition, the web traffic extracted from the real trace file had a large percentage of non persistent connections (80%) resulting in a large number of short flows.

6. Discussion

There are several issues involved in the three methods that make it difficult to accurately generate traffic on a backbone network. Some of those issues are discussed below:

6.1 Choosing the duration of the input and generated traces

The duration of the input traces and the generated traces plays a large role in determining the accuracy of the generated traces. Input files that are too short and are used in the *scaling* or *merging* methods will not capture long tailed behavior. In particular, for the *structural model* method input files that are too short do not allow the structural model to capture long tailed behavior in the CDFs that are generated. The duration of the simulation also affects the characteristics of the trace generated. Simulations that are too short will not capture long tailed behavior. On the other hand it may be computationally infeasible to simulate a long duration.

Our experience was that if a comparison is to be made between the real and generated traces, then the simulation should be run for a duration at least equal to the duration of the real world trace from which the data to drive the simulation was extracted. E.g. if a generated trace of 100 seconds is required, then data for the structural model should be extracted from a real world trace of duration 100 seconds and then 100 seconds of simulation should be run.

6.2 Increasing the bandwidth and determining steady state of the simulation

In the *strutural model* method high bandwidth traces are obtained by varying two quantities:

(a) User interarrival times
(b) Link bandwidths

We increased both by factors so as to maintain their ratio constant. Thus there is an implicit assumption that the number of users increases proportionately to the link bandwidth. The graph below indicates the bit rate (bps) in the real (10Mbps) and generated traces (1Gbps and 10Gbps) averaged over intervals of 1s for three different scalings v/s time. The graph indicates that increasing the link bandwidth and decreasing the user interarrival time by the *simulation scaling factor s* results in an increase in the traffic by an amount in the order of the *simulation scaling factor.*

The lowest curve is the original trace file (on a 10Mbps link). The middle curve is the simulation run with a simulation scaling factor of 100 (i.e a 1Gbps link) and the top curve has a simulation scaling factor of 1000 (i.e a 10Gbps link). A trace of 1000 seconds of a 1Gbps simulation took about 3 hours and 50 minutes to run on a 1GHz Pentium IV system with 1GB RAM running Red Hat Linux 7.2. The 10Gbps simulation took almost 24 hours. The 10 Gbps simulation was stopped after 500s of simulated time (hence the abruptly terminated curve for the 10Gbps simulation) due to a limit in the environment used to run the simulation. As seen from the graph above it takes approximately 250 seconds of simulation time before the system reaches steady state and traces can be analyzed.

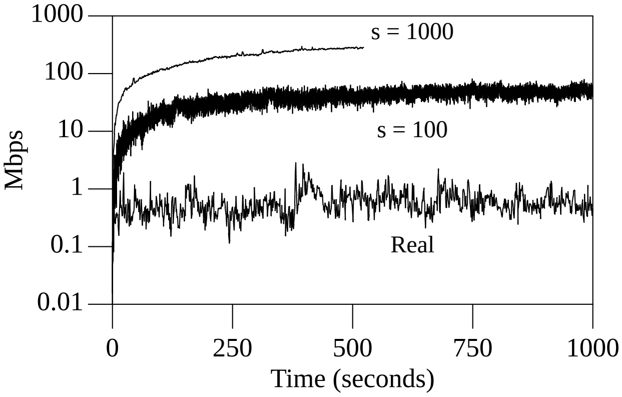

Figure 9: Relationship between the simulation scaling factor and the generated traffic as a function of simulation time.

7. Related work

SynRGen[8] is a file reference generator that is used build test suites for file systems. Like the structural model method it attempts to model the behavior of real users by tracing user behavior. A model of user behavior is built from the trace. Stress testing of the file system is performed by using this user model to simulate a large number of users on the system. This system is in many ways analogous to the structural model which also builds a user model and attempts to increase the number of users to generate higher bandwidth traces.

Lucas et al. [15,16] have characterized the nature of wide area IP traffic and have developed a traffic model called (M, P, S) to generate traffic for simulation. Their method of generation of traffic traces consist of three steps: Generation of aggregate traffic (arrivals per 100 ms intervals) using a self similar traffic model, partitioning the generated traffic by assigning destination addresses according to expected arrival distributions and finally distributing the packet arrivals in arrivals per ms. Unlike the structural modeling method which attempts to build a model of user behavior, their method assumes a particular model for the packet arrivals (self similar). However, it is likely that their model will be able to generate high bandwidth traces much more efficiently than the structural model method.

8. Conclusions

We have compared three different methods of generating high bandwidth traces. Of these methods *scaling* appears to generate the least accurate traces, distorting flow durations and destination address diversity. *Merging with renumbering* appears to be a viable method if the factors of interest are only connection characteristics and packet characteristics such as packet interarrival times and packet sizes. However, because of renumbering, it will not accurately model the destination address characteristics.

The *structural model* method generates accurate traces. However the use of only web traffic as a method of simulating all Internet traffic causes the model to fall short of accuracy. In particular flow durations have not been replicated accurately. An extended model based on different types of Internet traffic would provide more accurate traces. As a method of generating higher bandwidth traces, scaling user interarrival times gives a fairly proportional scaling in the offered load on a link. The simulation should be run for a time equal to the duration of the real world trace used to extract the statistics for the simulation.

Ultimately the limits of processor speed and memory limit the ability of the structural method to generate traces that indicate long tailed behavior. However, this method can generate traces that help understand high bandwidth network behavior for short durations of time (seconds rather than hours).

References

1. Allman, M. and Paxson, V., “TCP congestion control; RFC 2581,” *Internet request for comments* (Apr 1999).
2. Bajaj, S., Breslau, L., Estrin, D., Fall, K., Floyd, S., Haldar, P., Handley, M., Helmy, A., Heidemann, J., Huang, P., Kumar, S., McCanne, S., Rajaie, R., Sharma, P., Varadhan, K., Xu, Y., Yu, H., and Zappala, D., “Improving simulation for network research,” *Technical Report 99-702, University of Southern California,* http://www.isi.edu/nsnam (Mar 1999).
3. Cao, J., Cleveland, W. S., Lin, D., and Sun, D. X., “On the non stationarity of Internet traffic,” *Proceedings of ACM Sigmetrics,* pp. 102-112, Cambridge, MA (Jun 2001).
4. Claffy, K., “Internet measurements: Myths about Internet data,” *Presentation at NANOG 2002,* Miami (Feb 2002).
5. Claffy, K., Braun, H. W., and Polyzosi, G., “Traffic characteristics of the T1 NSFNET backbone,” *Proceedings of IEEE Infocom,* pp. 885-892, San Francisco (Aug 1993).
6. Claffy, K., Miller, G., and Thompson, K., “The nature of the beast: recent traffic measurements from an Internet backbone,” *Proceedings of Inet* (July 1998).
7. Danzig, P.B., Jamin, S., Caceres, R., Mitzel, D., and Estrin, D., “Characteristics of wide area TCP/IP conversations,” *Proceedings of ACM Sigcomm,* pp. 101-112, Zurich (1991).
8. Ebling, M. R. and Satyanarayana, M., “SynRGen: An extensible file reference generator,” *ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems,* pp. 108--117. (May 1994).
9. Fang, W. and Peterson, L., “Inter AS traffic patterns and their implications,” *Proceedings of IEEE Globecom,* Rio de Janeiro, Brazil (Dec 1999).
10. Feldmeier, D., “Improving gateway performance with a routing table cache,” *Proceedings of IEEE Infocom,* pp. 298-307, New Orleans (Mar 1988).
11. Fielding, R., Mogul, J., Gettys, J., Frystyk, H., and Berners-Lee, T., “Hypertext Transfer Protocol -- HTTP/1.1; RFC 2068,” *Internet request for comments* (Jan 1997).
12. Iannaccone, G., Diot, C., and McKeown, N., “Monitoring very high speed links,” *Proceedings of ACM Sigcomm Internet Measurement Workshop,* pp. 267-271, San Francisco (Nov 2001).
13. Lan, K.C. and Heidemann, J., “Rapid model generation from traffic measurement,” *Under submission.*
14. Leland, W. E., Taqqu, M. S., Willinger, W., and Wilson, D. V., “On the self-similar nature of Ethernet traffic,,” *IEEE/ACM Transactions on Networking,* Vol 2, No 1, pp. 1-15 (Feb 1994).
15. Lucas, M. T., Dempsey, B. J., Wrege, D. E., and Weaver, A. C., “Statistical characterization of wide area IP traffic,” *Sixth International Conference on Computer Communications and Networks (IC3N’97),* Las Vegas, NV (Sept 1997.).
16. Lucas, M. T., Dempsey, B. J., Wrege, D. E., and Weaver, A. C., “(M, P, S) - An efficient background

traffic model for wide area network simulation," *1997 IEEE Global Telecommunications Conference,* Vol. 3, pp. 1572-1576 (1997).

17. McGregor, A., Braun, H. W., and Brown, J., "The NLANR Network Analysis Infrastructure," *IEEE Communications Magazine,* Vol. 38, No 5, pp. 122-128, http://pma.nlanr.net/Traces/Traces (May 2000).

18. Micheel, J., Graham, I., and Brownlee, N., "The Auckland data set: an access link observed," *Proceedings of the 14th ITC Specialists Seminar on Access Networks and Systems,* Barcelona, http://pma.nlanr.net/Traces/long (Apr 2001).

19. Paxson, V. and Floyd, S., "Why we don't know how to simulate the Internet," *Proceedings of the 1997 Winter Simulation Conference* (Dec 1997).

20. Riley, G. and Ammar, M., "Simulating large networks - how big is big enough?," *Proceedings of the first international conference on Grand Challenges for Modelling and Simulation* (Jan 2002).

21. Smith, F. D., Campos, F., Jeffay, K., and Ott, D., "What TCP/IP headers can tell us about the Web," *Proceedings of ACM Sigmetrics,* pp. 245-256, Cambridge, MA (Jun 2001).

22. Willinger, W. and Paxson, V., "Where mathematics meets the Internet," *Notices of the American Mathematical Society,* Vol. 45, No. 8, pp. 961-970, b (Aug 1998).

23. Willinger, W., Paxson, V., and Taqqu, M. S., "Self similarity and heavy tails: Structural modeling of network traffic," *A practical guide to heavy tails: Statistical techniques and applications,* ISBN 0-8176-3951-9, Birkhauser, Boston (1998).