Swing: Generating Representative High Speed Packet Traces

[Extended Abstract] [*]

Kashi Vishwanath and Amin Vahdat
University of California, San Diego
9500 Gilman Drive
La Jolla, CA 92093
{kvishwanath,vahdat}@cs.ucsd.edu .

ABSTRACT

The goal of this work is to generate packet traces for traffic arriving at a particular link in the network (e.g., see Figure 1) representative of a wide range of both current and future scenarios. Such traces would be valuable in a variety of settings including: capacity planning, high-speed router design, queue management studies , bandwidth measurement tools, network emulation and simulation.

1. INTRODUCTION

We identify two primary goals for any packet trace generation tool, *realism* and *responsiveness.* To be realistic, a trace must reflect the essential characteristics of the packets arriving at a target network link, including: i) packet inter-arrival rate across a range of time scales, ii) packet size distribution, iii) destination IP address distribution, and iv) flow characteristics including arrival rate and length distributions. To be responsive, a trace generation tool must flexibly and accurately adjust trace characteristics based on a range of settings, such as: i) bandwidth capacity of the target link; ii) round trip time distributions of the flows traversing the link; iii) mix of applications sharing a link, e.g., if P2P traffic were to grow to 40% of the traffic on a link and iv) changes in application characteristics, e.g., if average P2P file transfer size grew to 100 MB (to reflect video versus audio distribution).

2. METHODOLOGY

We focus on determining the bi-directional characteristics of packets traversing a single network link, a *target.* We assume that these characteristics are fully specified by the behavior of: i) the users and programs initiating communication across the link, ii) the protocols employed by these programs, iii) the computers hosting the communication endpoints, and iv) the large space of other network links responsible for carrying the packet traffic between source and destination. Figure 1 gives an abstract view of the inter-play of these various system components. We argue that without accounting for *all* four of these system components, it is not possible to perform realistic and responsive packet trace generation. Without modeling users, it is impossible to study the effects of architectural changes on end users or to capture the effects of changing user behavior (e.g., if user patience for retrieving web content is reduced). Similarly, without an understanding of a wide mix of applications and protocols, it is difficult to understand the effects of evolving application popularity on capacity planning, traffic burstiness, etc. The end hosts running the applications may introduce delays in generating responses that affect packet dynamics. Finally, the bandwidth, latency, and loss rate of the links upstream and downstream will, at minimum, affect packet inter-arrival characteristics and TCP transmission behavior of the end hosts. We take the approach of developing a structural model of these four system components consisting of a simple set of parameters. These parameters together constitute a generic structural model of applications and can be divided into the following five categories:

i) *Users:* Client IP
ii) *Sessions:* Number of request-response exchanges (RREs), interRRE times (synonymous with user think times in case of web requests)
iii) *RRE:* # of Connections
iv) *Connection:* Server IP, Port number, protocol to use (e.g., TCP), # ReqResp pairs
v) *ReqResp pair:* Request Size, server think time to generate response, Size of response and MTU (or duration and bitrate in case of streaming media)

A set of values for these parameters constitute an application's *signature.* For instance HTTP, P2P and SMTP will all have different signatures.

While there are a range of possible techniques for realistically populating our model, we use traces of per-packet communication characteristics of a given link. We have developed a suite of programs to extract appropriate parameter values tarting from an arbitrary packet header net-

[*]More information about Swing is available at `www.cs.ucsd.edu/kvishwanath/~kvishwanath/Swing`

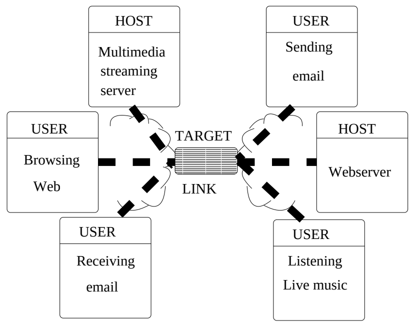

Figure 1: Different users running different applications talk to different hosts while sharing a common network.

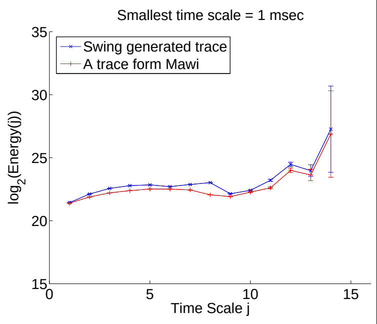

Figure 2: Wavelet Scaling plots.

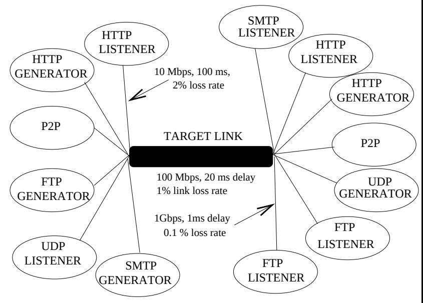

Figure 3: Target network link modeled as a dumb-bell.

work trace. These tools also extract delay, loss rate and capacity values for flows traversing the target link. While any tractable model cannot fully capture the behavior of such a complex system, we have observed that our parameters and their extracted values/distributions are sufficient to accurately capture the essential characteristics of the applications, hosts, and users communicating across a number of measured links. An important aspect of our work is that we build structural models on a *per-application* basis. This allows us to understand the individual behavior of, e.g., HTTP versus peer-to-peer versus streaming multimedia traffic. Further, it allows us to experiment with varying mixes of application traffic and varying the characteristics of individual applications (e.g., if the bitrate for streaming audio were to double). Currently we separate application classes based on the port numbers used and plan to incorporate recent advances in using machine learning techniques to classify traffic based on communication patterns [2] in the future.

Starting with these models, our goal is to generate a family of traces with similar statistical properties to the original trace. Swing consists of generators and listeners that communicate across a cluster of workstations according to extracted structural models (CDFs) of users, applications, and hosts. At a high level, generators initiate communication to listeners using appropriate protocols (e.g., TCP or UDP) with request/response interactions between generators and listeners specified by our model. Swing also assigns IP addresses to listeners and generators according to an underlying distribution. *Critically, listeners and generators run on stock operating systems and employ unmodified TCP/UDP/IP protocol stacks to manage their communication* (as shown in Figure 3).

3. GENERATING TRACES

Trace generation proceeds in two logical phases. First, we create a topology that reflects the wide-area characteristics (extracted using our suite of tools) of the underlying trace. A single link within the topology is designated as our *target*. We then place listeners and generators on either side of the dumbbell with a per-application IP address distribution matching the underlying trace and configure them with the information necessary to initiate communication with a pattern matching our derived models. This is shown in Figure 3.

In the second step, we map the topology along with the associated listeners and generators to a set of machines responsible for executing our code and generating live communication streams (using the ModelNet emulation environment). In ModelNet, unmodified hosts running stock operating systems execute the listeners and generators; it configures the host to route all packet communication through one or more *core* nodes responsible for emulating the characteristics of the specified wide-area topology.

We generate our packet trace by simply recording in the core (in tcpdump format) all packets that cross the link in both directions. Early results, for instance, from reproducing a trans Pacific line from the Wide [1] working group with an average bandwidth of 18Mbps (using 15 commodity PCs running 650 instances of listeners/generators) are encouraging (Figure 2).

4. CONCLUSIONS

Swing is a packet trace generation tool that improves upon the realism of existing tools. It is the first such tool to be responsive to a range of user-specified deployment scenarios reflecting realistic, evolving application and user characteristics. We have validated our tool against a variety of available packet traces and statistical measures.

As a validation of our claims, to the best of our knowledge we have built the first tool that runs real TCP stacks on end hosts to exchange real network packets in an emulation framework and yet closely matches a variety of characteristics of the original network link. Our validation consists of both first and second order statistics and wavelet scaling plots (capturing burstiness) at time scales ranging from milliseconds to minutes.

5. REFERENCES

[1] Mawi working group traffic archive. http://tracer.csl.sony.co.jp/mawi/.

[2] A. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In *ACM Sigmetrics*, 2005.