Performing Time-Sensitive Network Experiments

Neda Beheshti
Stanford University
nbehesht@stanford.edu

Yashar Ganjali
University of Toronto
yganjali@cs.toronto.edu

Monia Ghobadi
University of Toronto
monia@cs.toronto.edu

Nick McKeown
Stanford University
nickm@stanford.edu

Jad Naous
Stanford University
jnaous@stanford.edu

Geoff Salmon
University of Toronto
geoff@cs.toronto.edu

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Internetworking

General Terms

Experimentation, Measurement

1. INTRODUCTION AND MOTIVATION

It is commonly believed that the Internet has deficiencies that need to be fixed. However, making changes to the current Internet infrastructure is not easy, if possible at all. Any new protocol or design to be implemented on a global scale requires extensive experimental testing in sufficiently realistic settings; simulations alone are not enough. On the other hand, performing network experiments is intrinsically difficult for several reasons: *i*) Creating a network with multiple routers and a topology that is representative of a real backbone network requires significant resources, *ii*) Network components have proprietary architectures, which makes it almost impossible to figure out all of their internal details, *iii*) Making changes to network components is not always possible, *iv*) We cannot always use real network traces and generating high volumes of artificial traffic which closely resemble operational traffic is not trivial, and *v*) We need a measurement infrastructure which collects traces and measures various metrics throughout the network. These problems become even more pronounced in the context of time-sensitive network experiments. These are experiments that need very high-precision timings for packet injections into the network, or require packet-level traffic measurements with accurate timing. Experimenting with new congestion control algorithms, buffer sizing in Internet routers, and denial of service attacks which use low-rate packet injections are all examples of time-sensitive experiments, where a subtle variation in packet injection times can change the results significantly. In this work we study the challenges of conducting time-sensitive network experiments in a testbed. We provide a set of guidelines that aim at eliminating sources of inaccuracy in a time-sensitive network experiment. We should note that these guidelines are not meant to be comprehensive. For the sake of space, we only focus on issues that are most likely to be overlooked, and thus unknowingly distort the results of a time-sensitive network experiment.

2. GUIDELINES

1. Identify important traffic characteristics. The network traffic in the testbed should be representative of the traffic that exists in the actual system being tested. The results of time-sensitive network experiments are especially sensitive to subtle traffic properties; thus, these properties must be identified and replicated in the testbed.

2. Be wary of software configuration details and limitations. General purpose operation systems offer many options to tweak and tune their network stacks for a wide range of scenarios. Their defaults are not appropriate for every time-sensitive experiment in every testbed environment.

3. Be wary of hardware configuration details and limitations. As with software solutions, hardware components may have parameters and limitations which have a negligible effect on normal operation but have a large impact on the results of a time-sensitive experiment. In certain experiments, the requirements for precise timings and configuration may actually dictate the choice of hardware.

4. Account for the effects of scaling. Experiments that attempt to emulate a large network in a smaller testbed face many timing issues. Hosts and components that should be logically separate must be conflated together in the testbed. It is necessary to recognize and account for these approximations when scaling down large networks.

3. CHALLENGES

In this section, we will describe the challenges associated with time-sensitive network experiments in the context of buffer sizing experiments. In these experiments, we aim at finding out the buffer size requirements of Internet routers. Due to space limitations, we refer the interested reader to the complete version of this paper [1].

Traffic Generation: Based on the first guideline, the testbed's traffic should approximate the traffic of the actual system being tested. Having traffic that is representative of the real network being studied is crucial for any time-sensitive network experiment. In the buffer-sizing experiment, we need to emulate the traffic arriving to a core router in the Internet. Such traffic is comprised of packets generated in multiple access networks with different bandwidths. The difference between link bandwidths changes the

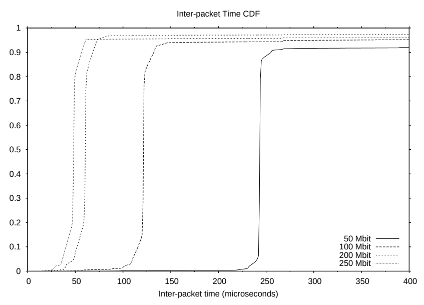

Figure 1: Packet Pacing With PSPacer.

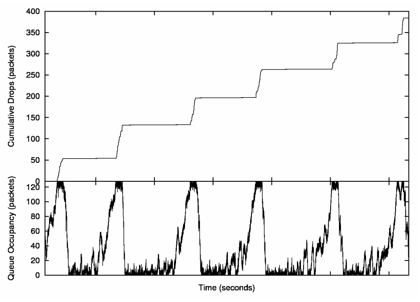

Figure 2: Precise queue occupancy and packet drops monitored with the NetFPGA

packets pacing as the packets go from one link to another. This is an aspect of Guideline 4 as limited resources in the testbed are emulating a large scale system.

In our testbed, we use the open-source traffic generator Harpoon [2]. Harpoon sends its traffic through the normal Linux network stack, which has many tuning options. The behaviour of most of these options is well documented [1]. To emulate the Internet's ratio of fast core links to predominately slower access links we use the PSPacer[3] package. Figure 1 shows the resulting paced packet timings. The MTU of the network is 1500 bytes which can be transmitted in roughly $12\mu s$ at 1Gbps speeds. The machine sending the packets uses PSPacer to simultaneously simulate four different access links with the speeds 50, 100, 200 and 250 Mb/s, which can transmit 1500 bytes in roughly 240, 120, 60 and 48 μs, respectively. The figure shows the CDFs of inter-arrival times for the fours sets of packets received at another computer.

Network Card Options: As Guidelines 2 and 3 describe, some of the problems associated with time-sensitive network experiments are due to software/hardware limitations and configuration. For instance, general purpose computers and operating systems can not provide precise timing guarantees. However, there are some features on network cards that can affect the spacing and burstiness of packet transmissions: TCP Segmentation Offload (TSO), Interrupt Coalescing (IC) and Ethernet Flow Control. With TSO enabled, Linux can pass to the card a packet that is much larger than the maximum transmission unit supported by the medium. This option frees the CPU from the segmentation overhead but can adversely cause bursts of back to back packets on the wire. IC introduces queueing delays and alters the pacing of packets, and Flow Control can cause unnecessary queuing. In our technical report [1], we provide examples of experiments with demonstrably different results by tuning these options.

Choosing a Router: Commercial routers are usually presented as black boxes by their manufacturers. The details of their architectures are closely guarded secrets, and, although they do allow some configuration and monitoring, they are of limited use in time-sensitive network experiments which require precise control and measurement at a per-packet level. In addition to requiring precise configuration, the analysis of a time-sensitive network experiment may depend on timings of the internal state of a router. In particular, understanding the behavior of small buffers requires monitoring the exact queue occupancy, a feature that commercial routers do not provide. As Guideline 2 describes, some of the problems associated with time-sensitive experiments cannot be addressed by configuring software components. Higher levels of timing guarantees require hardware support, and the type of hardware support necessary depends on the experiment. In the context of buffer-sizing experiments, we show how we have been able to address many of the timing issues using NetFPGA board[1], a programmable and configurable network component, as the router in the experiment. It allows output buffer sizes to be set precisely, in either bytes or packets. By modifying the router to record the time of each packet arrival, departure and drop at each output buffer, we can reconstruct the exact buffer-occupancy time-series. Figure 2 shows a sample of the buffer occupancy and packet drop information for a buffer limited to 128 packets. To obtain this detailed information, we modified the NetFPGA router design to support instrumenting the router's output queues. When a packet arrives, departs, or drops at an output queue, the size of the queue and the current clock of the NetFPGA, which has an 8ns granularity, are recorded. Further details about the NetFPGA router and our modifications are explained in our technical report [1].

4. CONCLUSIONS

To make a testbed, with limited resources, approximate closely a real large heterogeneous network, one needs to exactly identify the traffic pattern. Subtle changes in the traffic pattern can make the result of a time-sensitive experiment different. Appropriate software and hardware configuration is then required to generate traffic with the desired characteristics. This work provides guidelines for identifying traffic patterns and configuring software and hardware components to generate traffic in a testbed.

5. REFERENCES

[1] N. Beheshti, Y. Ganjali, M. Ghobadi, N. McKeown, J. Naous, and G. Salmon. Performing time-sensitive network experiments. Technical Report TR08-UT-SNL-09-10-00, University of Toronto, September 2008.

[2] J. Sommers and P. Barford. Self-configuring network traffic generation. pages 68–81, Taormina, Sicily, Italy, 2004. ACM.

[3] R. Takano, T. Kudoh, Y. Kodama, M. Matsuda, H. Tezuka, and Y. Ishikawa. Design and evaluation of Precise Software Pacing mechanisms for fast long-distance networks. *3rd PFLDnet Workshop*, 2005.

[1]http://www.netfpga.org/