Caliper: A Tool to Generate Precise and Closed-loop Traffic

Monia Ghobadi*, Martin Labrecque†, Geoffrey Salmon*, Kaveh Aasaraai†,
Soheil Hassas Yeganeh*, Yashar Ganjali*, J. Gregory Steffan†
*Department of Computer Science, †Department of Electrical and Computer Engineering, University of Toronto
{monia, geoff, soheil, yganjali}@cs.toronto.edu, {martinl, aasaraai, steffan}@eecg.toronto.edu

ABSTRACT

Generating realistic and responsive traffic that reflects different network conditions is a challenging problem associated with performing valid experiments in network testbeds. In this work, we preset Caliper, a highly precise traffic generation tool, built on NetThreads, a flexible platform that we have created for developing packet processing applications on FPGA-based devices and the NetFPGA in particular. We will demonstrate the effect of ad-hoc inter-departure times on a commodity NIC compared to precisely timed inter-departures with Caliper. Both NetThreads and Caliper are available as free software to download [1].

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

General Terms

Design, Experimentation, Measurement

Keywords

NetFPGA, Traffic Generation, Soft Processors

1. INTRODUCTION AND MOTIVATION

There are many challenges associated with performing valid experiments in network testbeds. Generating realistic and responsive traffic that reflects different network conditions and topologies is one of such key challenges. To perform network experiments, researchers often use a collection of commodity Linux machines as traffic generators. However, creating a large number of connections in order to accurately model the traffic shape in networks with thousands of flows is difficult for several reasons. As an alternative, commercial traffic generators are useful for some experiments, but they are usually very expensive and their proprietary nature makes them inflexible for research purposes. Hence, it is intrinsically difficult to perform *time-sensitive* network experiments with confidence on the accuracy of packet injection times. Time-sensitive experiments are those that need very high-precision timings for packet injections into the network. Experimenting with new congestion control algorithms, buffer sizing in Internet routers, and denial of service attacks are all examples of time-sensitive experiments, where a subtle variation in packet injection times can change the results significantly.

This demonstration has two objectives. First, we present Caliper, a precise packet generator that controls the transmission times of packets, created by arbitrary software on the host machine. We demonstrate Caliper's capabilities by visualizing the adverse effect of ad-hoc packet inter-departure times on a commodity NIC compared to the high precision achieved by Caliper. The difference can be perceptible for the audience by comparing two side-by-side video streams. One video feed is transmitted using a commodity NIC and the other using Caliper. As a commodity NIC has an imprecise injection rate of packets onto the network, its corresponding video will suffer from a much higher packet drop rate than the other and consequently deliver a much worse viewing experience. The second part of our demonstration focuses on presenting *NetThreads* [2], the software programmable system on the NetFPGA card that enables Caliper, and its ease-of-use.

2. DESIGN AND IMPLEMENTATION

Caliper allows to specify programmable packet timing requirements using NetThreads, a flexible platform that we have created for developing packet processing applications on FPGA-based devices and the NetFPGA in particular. NetThreads is primarily composed of FPGA-based processors, providing a familiar yet flexible environment for software developers: programs are written in `C`, and existing applications can be ported to the platform. In the rest of this section, we briefly go over the design and implementation of NetThreads and Caliper.

NetThreads: While the Internet infrastructure is dominated by vendors with proprietary technologies, there is a push to democratize the hardware, to allow researchers to revisit network protocols that have not evolved years. This desire to add programmability in the network is formally embraced by large scale projects such as CleanSlate, RouteBricks and GENI, in turn supported by massive infrastructure projects such as Internet2 and CANARIE. NetThreads is a possible solution to fulfill that need as it allows to rapidly develop low-level packet processing applications.

To avoid implementing a networking application in low-level hardware-description language (which is how FPGAs are normally programmed), we instead implement *soft processors* —processors composed of programmable logic on the FPGA. Despite the raw performance drawbacks, a soft

SIGCOMM'10, August 30–September 3, 2010, New Delhi, India.
ACM 978-1-4503-0201-2/10/08.

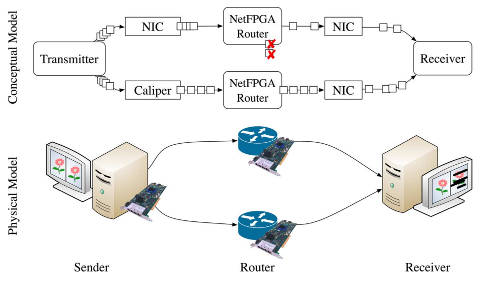

Figure 1: The conceptual (top) and physical (bottom) network configuation for the demonstration.

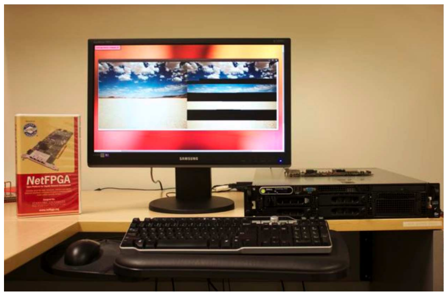

Figure 2: The demonstration in action.

processor has several advantages: it is easier to program (e.g., using `C`), portable to different FPGAs, flexible (i.e., can be customized), and can be used to communicate with other components/accelerators in the design. In this work, our FPGA resides on a NetFPGA board and communicates through DMA on a PCI interface to a host computer [3].

Precise Traffic Generation: Caliper's main objective is to precisely control the transmission times of packets which are created in the host computer, continually streamed to the NetFPGA, and transmitted on the wire. The generated packets are sent out of a single Ethernet port of the NetFPGA, according to any given sequence of requested inter-transmission times. Unlike previous works that replay packets with prerecorded transmission times from a trace file, Caliper generates live packets and supports closed-loop traffic. Therefore, Caliper can easily be coupled with existing traffic generators (such as Iperf, the widely used traffic generation tool in the Linux kernel, or as we demonstrate here, a video streaming application) to improve their accuracy at small time scales. In the following we explain each stage of a packet's journey through Caliper.

First, a user space process or a kernel module on the host computer determines when a packet should be sent. A description of the packet, containing the transmission time and all the information necessary to assemble the packet is sent to the NetFPGA driver. In the driver, multiple packet headers are combined and copied to the NetFPGA card. The last part of Caliper runs as software on the NetThreads platform inside the NetFPGA. The driver sends its messages containing the headers of multiple packets and their corresponding transmission times. Then, Caliper creates these packets and sends them at the appropriate times.

3. DEMONSTRATION

In this section, we describe the two experiments that we will perform, highlighting the key characteristics of Caliper and NetThreads.

Demonstrating Precision and its Importance: The goal of this part of the demonstration is to visually show the adverse effects of imprecise packet injections by commodity NICs while the same experiment works fine using Caliper. For this purpose, we design a time-sensitive experiment, as we explain next.

The continuous growth in network link speeds makes it increasingly difficult to design routers with buffer sizes equal to the standard bandwidth-delay product. As a result, many network providers revert to using routers with small buffers. Those routers have the drawback of losing packets when the traffic is bursty, so they are usually used in conjunction with software packet pacing techniques that reduce traffic burstiness. Software pacing is however not completely effective, due to timing inaccuracies introduced by commodity NICs and their software drivers.

As shown in Figure 1, we use one video transmitter software running on a host equipped with two interfaces: a commodity NIC, and Caliper. Video is transmitted on both interfaces: the NIC traffic is software-paced by the Linux kernel, while Caliper enforces pacing on the other interface. Both interfaces are directly connected to two identical NetFPGA router cards with small buffers, both installed on a single host. Each router forwards packets to separate NICs on the receiving machine. Finally, we display videos side-by-side with a single receiver software. Packet drops result in obvious reduced video quality for the software-paced traffic, showing that pacing with Caliper is substantially more effective. Figure 2 shows two HD videos feeds on the receiving machine: the right hand side of the image shows the stream from the NIC and the left hand side image is from Caliper. As it can be seen, the right hand side picture is suffering from packet drops and exhibits worse quality.

Demonstrating Usability: Caliper relies on some software executing inside the NetFPGA network card. Since our compiler infrastructure for that program is based on a modified `gcc` compiling ordinary `C code`, users need no special introduction to the programming environment. We demonstrate the flexibility of NetThreads by allowing users to change a program that modifies a video stream on-the-fly. In particular they are able to add custom headers to packets including a text which is used as a closed captioning of their choice updated in real-time on the video stream sent by Caliper. This experiment uses the same setup as above: only the NetThreads software program is changed. By getting a first hand experience with NetThreads, attendees can hopefully earn confidence to write gigabit applications on their own by downloading our infrastructure [1].

4. REFERENCES

[1] Caliper wiki. http://netfpga.org/foswiki/bin/view/NetFPGA/OneGig/PreciseTrafGen.

[2] M. Labrecque, J. G. Steffan, G. Salmon, M. Ghobadi, and Y. Ganjali, "NetThreads: Programming NetFPGA with threaded software," in *NetFPGA Developers Workshop*, Palo Alto, California, August 2009.

[3] The NetFPGA project. http://www.netfpga.org/.