Caliper: Precise and Responsive Traffic Generation using NetThreads

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

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. First, it is not always possible to use real network traces as they do not maintain the feedback loop between the network and traffic sources (for example the TCP closed-loop congestion feedback). Second, the complexities of traffic generation increase when trying to capture the heterogeneity of link capacities using only a limited number of physical machines. Finally, commodity hardware does not guarantee the precision of generated traffic, which is bounded by the system timer's resolution.[1] 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 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.

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 will 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 is to present *NetThreads*, the software programmable system on the NetFPGA card that enables Caliper, and its ease-of-use.

[1] A Linux kernel is typically capable of providing resolutions of 1 ms. With links running at 1 $Gbps$, even a large packet of 1500 bytes has a transmission time of less than 12 μs.

2. DESIGN AND IMPLEMENTATION

Caliper allows to specify programmable packet timing requirements because of NetThreads, a flexible platform that we have created for developing packet processing applications on FPGA-based devices and the NetFPGA [1] 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.

2.1 NetThreads

While the Internet infrastructure is dominated by vendors with proprietary technologies, there is a push to democratize the hardware [2], to allow researchers to revisit network protocols that have not evolved in more than a decade. 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 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. This configuration is particularly well-suited for networking: (i) the load of the soft processors is isolated from the load on the host processor, (ii) the soft processors suffer no operating system overheads, (iii) they can receive and process packets in parallel, and (iv) they have access to a high-resolution system clock (much higher than that of the host clock). While a number of other soft processors are available, NetThreads cores provide an increased pipeline efficiency for control-flow intensive programs through the use of multithreading [3].

2.2 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.

Packet Creation to Driver: 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.

Driver to NetThreads: In the driver, multiple packet descriptions are combined and copied to the NetFPGA card. Combining descriptions reduces the number of separate transfers required and is necessary for sending packets at the line rate of 1 *Gbps*.

NetThreads to Wire: This 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 prepares these packets for transmission and sends them at the appropriate times. NetThreads takes advantage of multithreaded processors, which have been shown to outperform single threaded processor on parallel tasks [4].

3. DEMONSTRATION

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

3.1 Demonstrating Precision and its Importance

The goal of this part of the demonstration is to show graphically the adverse effects of imprecise packet injections by commodity NICs. 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. Commodity hardware does not guarantee the timing of outgoing packets, which is determined by the system's timer resolution.[2]

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 the demo in action. Two HD videos are visible in the picture where 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.

3.2 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 plan to demonstrate the flexibility of NetThreads by allowing users to change a program that modifies a

[2] A Linux kernel is typically capable of providing resolutions of 1 ms. In comparison, a packet of 1500 bytes on a 1 *Gbps* link has a transmission time of about 12 μs.

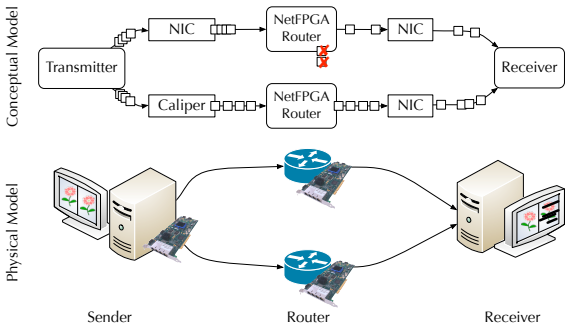

Figure 1: The conceptual (top) and physical (bottom) setup of the demonstration.

Type	Description
power	2500 Watts combined (5 power outlet connections)
cables	4 regular CAT-5 cables
machines	3 PCs with 2 gigabit Ethernet ports each (1 sender, 1 receiver, 1 for routers)
NetFPGA boards	3 NetFPGA (2 Routers, 1 Caliper)
visual equipment	a video projector, and a LCD monitor
space	$4 \times 4\ m^2$
related publications	[3, 5–7]
space needed	standard booth with space to project our demonstration screen (poster size minimum)
setup time required	∼30 minutes to connect the machines, power them and setup the projector.

Table 1: Demo Equipments

video stream on-the-fly. In particular they would be able to insert a closed captioning of their choice updated in real time on the video stream sent by Caliper. This experiment uses the same setup as in Section 3.1: only the NetThreads software program is changed. We hope to give attendees a first hand experience with NetThreads so that they can earn confidence to write gigabit applications on their own.

3.3 Demonstration details

Table 1 summarizes the requirements for our demonstration. This demo is eligible for a travel grant: presenters would be Monia Ghobadi and Martin Labrecque from the University of Toronto.

4. REFERENCES

[1] J. Naous, G. Gibb, S. Bolouki, and N. McKeown, "NetFPGA: reusable router architecture for experimental research," in *PRESTO*, 2008, pp. 1–7.

[2] J. He, M. Suchara, M. Bresler, J. Rexford, and M. Chiang, "Rethinking internet traffic management: from multiple decompositions to a practical protocol," in *CoNEXT '07: Proceedings of the 2007 ACM CoNEXT conference*. New York, NY, USA: ACM, 2007, pp. 1–12.

[3] 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.

[4] M. Labrecque and J. G. Steffan, "Improving pipelined soft processors with multithreading," in *FPL'07*, August 2007.

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

[6] G. S. M. Ghobadi, Y. Ganjali, M. Labrecque, and J. G. Steffan, "NetFPGA-based precise traffic generation," in *NetFPGA Developers Workshop*, Palo Alto, California, August 2009.

[7] M. Ghobadi, G. Salmon, M. Labrecque, Y. Ganjali, and J. G. Steffan, "Caliper: Precise and responsive traffic generation using NetThreads," in *ANCS, submitted*, May 2010.

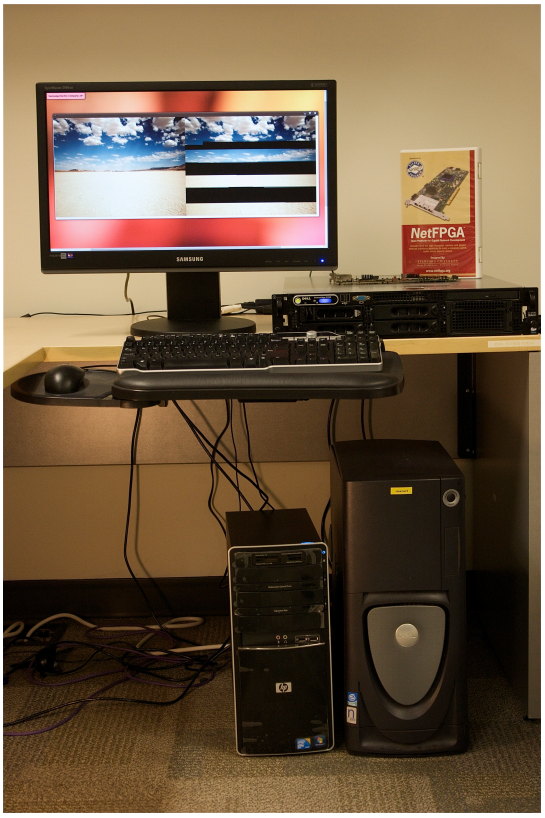

Figure 2: The demonstration in action.