Caliper: Precise and Responsive Traffic Generator

Yashar Ganjali
Department of Computer Science
University of Toronto

HotI 2012 – Santa Clara, CA

Joint work with:
Monia Ghobadi, Geoff Salmon, Martin Labrecque, J. Gregory Steffan

yganjali@cs.toronto.edu
http://www.cs.toronto.edu/~yganjali
Motivation

• Changing network components/protocols requires extensive and accurate experiments.

• Real network experiments are very difficult.
• Operators do not like changing their networks …
  • … before exhaustive tests in realistic settings.
  • Intrinsic risks associated with changing a complex network

• Testbed experiments are usually the only option.
Challenges

• **Question.** How can we generate real/realistic traffic for testbed experiments?
  • So that results are applicable in practice

• Three key challenges
  1. **Modeling:** what does real traffic look like?
  2. **Precision:** how can we accurately inject packets to the network?
  3. **Responsiveness:** how can we ensure the generated traffic changes according to network conditions?
Challenges

- **Question.** How can we generate real/realistic traffic for testbed experiments?
  - So that results are applicable in practice

- Three key challenges
  1. **Modeling:** what does real traffic look like?
  2. **Precision:** how can we accurately inject packets to the network?
  3. **Responsiveness:** how can we ensure the generated traffic changes according to network conditions?
Example

- Internet router buffer sizing experiments
  - Tiny buffers: 20-50 packets

- **Accurate packet injections** are extremely critical
  - Tiny errors in injection times can have a severe impact on experiment results

- Cannot ignore TCP **feedback loop**
  - Packet drops can impact future traffic patterns and packet injection times

- **Other examples**: new congestion control schemes, denial of service attacks, …
Problems with Existing Solutions

- Replaying pre-recorded traces
  - Example: Stanford Packet Generator (SPG)
- Feedback loop is broken
- Cannot be adapted based on higher level models
Problems with Existing Solutions

- Commercial traffic generators
  - Proprietary, inflexible, and expensive
    - Limited control on multiplexing, topology, …
  - Limited precision, focus on macro-level metrics
    - Not suitable for time-sensitive experiments

How is the traffic different?
Problems with Existing Solutions

- Commodity hardware + software traffic generation
  - Accuracy bounded system timer resolution
  - Unpredictable behavior due to differences in parameters
    - And, hardware
Caliper

- Precise and responsive traffic generator
  - Based on the NetFPGA platform

- Highly-accurate packet injection times
  - Explicit injection times

- Dynamic based on network state
  - Feedback loop, not a simple replay, …

- Integratable with software-based traffic generation tools
  - Iperf, ns-2, …
Components of Caliper

- Built on **NetThreads**, a platform for developing packet processing applications on FPGA based devices.

[Diagram showing the components of Caliper: 1. User space process or kernel module, 2. nf2 Linux Driver, 3. NetThreads Application, connecting to NetFPGA.]

Packet Travel Direction: Host Computer → NetLink → Linux Kernel → PCI → GigE → NetFPGA
Packet Creation

- Inter-transmission times and payload sizes can be
  - fixed, read from a file, or come from an application

- A user space process or kernel module creates a sequence of packets

- Descriptions of packets and transmission times are sent to the driver

- Communicates with driver using a NetLink socket.

  - Easily replaced by other user space or kernel code
nf2 Linux Driver

- Modified version of nf2 driver.
- Main jobs:
  - Receive packet descriptions from NetLink socket.
  - Build command packets containing multiple descriptions.
  - Send command packets to the NetThreads app using DMA over the PCI bus.
PCI Bus is a Bottleneck

- Cannot copy all packets across PCI bus.

- **Idea**: do not copy packet payloads
  - Payloads are zeroed when sent from NetFPGA.
  - Experiment often only look at packet headers anyway.
  - Or, chosen from predefined payloads

- To minimize PCI transaction overheads
  - Driver gathers multiple packet descriptions into a single command packet.
  - Sent to the NetThreads Application
NetThreads Application on NetFPGA

- Eight threads of execution in NetThreads
  - 7 threads receive command packets and prepare packets to transmit in output memory.
  - 1 thread sends packets from the output memory at correct times.
Integration with Existing Tools

- Caliper acts as a **NIC device** in the kernel.

- Transmits packets generated within the kernel with any software packet generator
  - ping, Iperf, or high level simulation tools (like *ns-2*)

- Caliper can transmit live TCP connections and closed-loop sessions.
  - Thus, the generated traffic becomes “responsive”.
  - Need careful **synchronization** with software packet generator
Integrating Caliper with ns-2

- Define arbitrary topology in _ns-2_
- Create a sequence of packets
- Feed to Caliper
  - And vice versa

```
1. #Caliper's interval in seconds:
2. set caliper_interval 0.001
3. #define the nodes n0, n1, n2, and n3
4. #define the links (n0, n2), (n2, n3), and (n3, n1)
5. #obtain the queue of the specific caliper queue:
6. set caliper_queue_ [$ns simplex-link-op $n2 $n3 queue]
7. #call use-caliper function:
8. $cliper_queue_ use-caliper
9. #set the physical IP/MAC addresses mapping table:
10. $ns insert_nat IP_N2 IP_N3 PORT_N2 PORT_N3 MAC_N2 MAC_N3
11. #Create a UDP agent and attach it to node n0
12. #Create a CBR traffic source and attach it to udp0
13. #set the rate of the CBR source:
14. $cbr0 set interval_ $caliper_interval
```
Evaluation

- **Focus**: accuracy and flexibility features.
- The most important metric is accuracy of packet transmission times.

Inter-arrival times are measured in NetFPGA
- Thus, highly accurate
Transmission Time Precision

- UDP, Fixed Inter-arrivals

It takes 8ns to send 1 byte on GigE.
Transmission Time Precision

- UDP, Pareto Inter-arrivals

NIC/Caliper/SPG with pareto interarrival

NetFPGA router measures inter-arrival times
Transmission Time Precision

- UDP, variable packet sizes

NIC/Caliper/SP G with variable packet size

NetFPGA router measures inter-arrival times

Cumulative Distribution Function

Inter-arrival Time (μs)

Caliper: Precise and Responsive Traffic Generator -- HotI 2012

Yashar Ganjali
Transmission Time Precision

- Closed-loop TCP traffic
  - SPG fails here
  - Three orders of magnitude improvement in error
[In]accuracy of Software Emulators

- Experiment to measure accuracy using NIST Net
- Schedule packet transmissions with fixed-rate timers/interrupts.

---

Caliper

NIST Net host
- Delays packets by 100ms

NetFPGA router
- measuring arrival times

Measure inter-arrivals here

Measure inter-arrivals here
Inaccuracy of Software Emulators

- Ideally, inter-arrival times should not change
Conclusion

- Generating realistic traffic in network testbeds is challenging yet crucial.
- Caliper dynamically and accurately controls the transmission times of a stream of packets.
- Can be integrated with existing software traffic generators and network emulators.
- Extremely small error in injection times (~8 ns)
  - NetFPGA’s clock cycle time.
Thank You!

Questions?
Back up slides
Modified Hardware Designs

- NetThreads design:
  - Removed Output Queues to increase accuracy of transmission times.

- NetFPGA reference router design used in measurements:
  - Removed Input Arbiter and Output Port Lookup.
Inaccuracy of NISTNet

- Software network emulators schedule packet transmissions with fixed-rate timers/interrupts.
- Ran experiment to measure inaccuracy using NISTNet.

![Diagram showing the process of a PTG sending packets through a NISTNet host, which delays packets by 100ms, and a NetFPGA router measuring arrival times.](image)
Packets Sent Every 70 μs

Caliper: Precise and Responsive Traffic Generator -- HotI
2012
Yashar Ganjali
Packets Sent Every 640 μs

Empirical CDF

- PTG
- NIST Net

Observed Inter-arrival Time (microseconds)
Packets Sent Every 700 μs

Empirical CDF

- PTG
- NIST Net

Observed Inter-arrival Time (microseconds)
Motivation: Network Testbed Experiments

- Testing systems which are sensitive to packet arrival times (i.e., packet buffering and scheduling in routers) requires realistic test traffic.
  - Need to *explore* a wide range of traffic
- With a small testbed, this is difficult or impossible.
- With a large enough testbed, this is difficult and expensive.
Throughput: Smaller Packets

Caliper: Precise and Responsive Traffic Generator -- HotI
2012

Yashar Ganjali
Throughput: Smaller Packets cont.

- Errors appear between 6000-7000ns.
- See the same problem for different packet sizes.
- The mean inter-transmission times are correct,
  - not a PCI bottleneck problem.
- Most likely: the sending thread is doing too much work between packet transmissions.
  - In 6400ns each thread has
    \[
    6400 \text{ ns} / 8 \text{ ns per cycle} / 4 \text{ threads} = 200 \text{ clock cycles}
    \]
Throughput: MTU-Size Packets

MTU-sized Packet = 1526 bytes
Transmits in $1526 \times 8 = 12208$ ns
Add Inter Frame Gap of 96ns.

12304ns is back to back!