TCPivo
A High-Performance Packet Replay Engine

Wu-chang Feng      Ashvin Goel      Abdelmajid Bezzaz
Wu-chi Feng           Jonathan Walpole

OGI SCHOOL OF SCIENCE & ENGINEERING
OREGON HEALTH & SCIENCE UNIVERSITY
Motivation

- Many methods for evaluating network devices
  - Simulation
    - Device simulated, traffic simulated
    - \texttt{ns-2}, IXP network processor simulator
  - Model-based emulation
    - Actual device, traffic synthetically generated from models
    - IXIA traffic generator
  - Trace-driven emulation
    - Actual device, actual traffic trace
    - Particularly good for evaluating functions that rely on actual address
      mixes and packet interarrival/size distributions
Goal of work

- Packet replay tool for trace-driven evaluation
  - Accurate
  - High-performance
  - Low-cost
    - Commodity hardware
    - Open-source software

- Solution: TCPiV0
  - Accurate replay above OC-3 rates
    - Pentium 4 Xeon 1.8GHz
    - Custom Linux 2.4.20 kernel with ext3
    - Intel 82544 1000Mbs
    - ~$2,000
Challenges

- Trace management
  - Getting packets from disk
- Timer management
  - Time-triggering packet transmission
- Scheduling and pre-emption
  - Getting control of the OS
- Efficient sending loop
  - Sending the packet
Trace management problem

- Getting packets from disk
  - Requires intelligent pre-fetching
  - Most OSes support transparent pre-fetch via `fread()`
- Default Linux `fread()` latency reading trace
Trace management in TCPivo

- Double-buffered pre-fetching
- `mmap()`/`madvise()` with sequential access hint
Timer management problem

- Must accurately interrupt OS to send packets

- Approaches
  - Polling loop
    - Spin calling `gettimeofday()` until time to send
    - High overhead, accurate
  - `usleep()`
    - Register timer interrupt
    - Low overhead, potentially inaccurate

- Examine each approach using fixed workloads
  - 1 million packet trace
  - Constant-interarrival times $\delta=70 \, \mu\text{sec}, \delta=2500 \, \mu\text{sec}$
Timer management problem

- Polling loop

\[ \delta = 70 \ \mu\text{sec} \]

78% User-space CPU utilization

\[ \delta = 2500 \ \mu\text{sec} \]

99% User-space CPU utilization
Timer management problem

- `usleep()`

**δ=70 μsec**

40% User-space CPU utilization

**δ=2500 μsec**

4% User-space CPU utilization
Timer management in TCPivo

• “Firm timers”
  – Combination of periodic and one-shot timers in x86
    • PIT (programmable interval timer)
    • APIC (advanced programmable interrupt controller)
    • Use PIT to get close, use APIC to get the rest of the way
  – Timer reprogramming and interrupt overhead reduced via soft timers approach
  – Transparently used via changes to `usleep()`
Timer management in TCPivo

- Firm timers

\[
\delta = 70 \ \mu\text{sec} \\
19\% \text{ User-space CPU utilization}
\]

\[
\delta = 2500 \ \mu\text{sec} \\
1\% \text{ User-space CPU utilization}
\]
Scheduling and pre-emption problem

- Getting control of the OS when necessary
- Low-latency, pre-emptive kernel patches
  - Reduce length of critical sections
- Examine performance under stress
  - I/O workload
    - File system stress test
    - Continuously open/read/write/close an 8MB file
  - Memory workload (see paper)
Scheduling and pre-emption problem

- Firm timer kernel without low-latency and pre-emptive patches
- I/O Workload, $\delta=70\mu$sec
Scheduling and pre-emption in TCPivo

- Firm timer kernel with low-latency and pre-emptive patches
- I/O Workload, $\delta=70\mu$sec
Efficient sending loop in TCPivo

- Zeroed payload
- Optional pre-calculation of packet checksums

<table>
<thead>
<tr>
<th>Task</th>
<th>Average time spent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace read</td>
<td>1.30 µsec</td>
</tr>
<tr>
<td>Data padding</td>
<td>1.45 µsec</td>
</tr>
<tr>
<td>Checksum calculation</td>
<td>1.27 µsec</td>
</tr>
<tr>
<td>sendto()</td>
<td>5.16 µsec</td>
</tr>
<tr>
<td><strong>Main loop</strong></td>
<td><strong>9.38 µsec</strong></td>
</tr>
</tbody>
</table>
Putting it all together

- On the wire accuracy
  - $\delta=70\mu$sec workload at the sender
  - Point-to-point Gigabit Ethernet link
  - Measured inter-arrival times of packets at receiver
Software availability

- **TCPivo**
  - [http://www.cse.ogi.edu/sysl/projects/tcpivo](http://www.cse.ogi.edu/sysl/projects/tcpivo)
  - Formerly known as NetVCR before an existing product of the same name forced a change to a less catchier name.

- **Linux 2.4**
  - Firm timers
    - [http://www.cse.ogi.edu/sysl/projects/TSL](http://www.cse.ogi.edu/sysl/projects/TSL)
  - Andrew Morton's low-latency patch
  - Robert Love's pre-emptive patch
    - [http://kpreempt.sourceforge.net](http://kpreempt.sourceforge.net)

- **Linux 2.5**
  - Low-latency, pre-emptive patches included
  - High-resolution timers via 1ms PIT (No firm timer support)
Open issues

- Multi-gigabit replay
  - Zero-copy
  - TOE
  - SMP
- Accurate, but not realistic for evaluating everything
  - Open-loop (not good for AQM)
  - Netbed/PlanetLab?
    - Requires on-the-fly address rewriting
Questions?