

T I C K E T

the Traffic Information Collection Kernel
with Exact Timing

Eric Weigle -- ehw@lanl.gov

Los Alamos National Laboratory



02: Overview

- Introduction
 - What TICKET Is and Does
- Motivation -- Why TICKET?
 - Trends, Other approaches, Pros & Cons
- Implementation Overview
 - Infrastructure, Methodology, Configurations
- Grungy Implementation Details -- API
 - Algorithms, data collected
- Performance evaluation
 - Experiments, configuration, hardware
- Results
 - Comparison with TCPdump/libpcap
- Closing
 - Questions?

03: What is TICKET?

A minimal kernel-level API designed for high-resolution networking tasks

- Implemented using Linux 2.2 and 2.4 series kernels
 - Removed superfluous code
 - Wrote new packet handlers
- Completely kernel-level; no user <-> kernel context switches
- Soft Real-Time with high-resolution timers
 - Will discuss in "Grungy details"

04: What can TICKET do?

■ Traffic Collection

- Nanosecond granularity
- The focus of this paper and talk
- In practice, tested up to 1.6 Gigabits/second.

■ WAN Emulation

- Works for bandwidth*delay up to memory in machine
- In practice, tested up to 200ms delay on 1 gigabit link.

■ Network Flooding (DoS simulation)

- Easily brings our firewall to its knees.
-

■ Limitations resolve to

- PCI bus (Even 64-bit 66-Mhz)
- Card capabilities
 - Lack of memory, processor, 802.3x 'flow' control.

05: Other approaches

Why not use...

- libpcap
- raw sockets
- tc
- iptables/ipchains
- netlogger
- custom hardware
- ...

Different goals means
different design decisions!

- Expense (\$)?
- Kernel-User
 - Packet copies?
 - Context switches?
- Latency
 - Timers?
 - Buffering?
- Competition
 - Users?
 - Processes?
- Scalability?

What do you optimize for?

06: Trends

■ Network

- (Rsrch) 6.4Tbps = 6400.0 Gbps
- (Cmdty) 1.0Gbps = 1.0 Gbps

■ Processor

- (Rsrch) 64-bit/4.0GHz = 256.0 Gbps
- (Cmdty) 32-bit/2.0GHz = 64.0 Gbps

■ Memory

- (Rsrch) 64-bit/333MHz = 21.3 Gbps
- (Cmdty) 64-bit/133MHz = 8.5 Gbps

■ Accessory

- (Rsrch) 64-bit/133MHz = 8.5 Gbps
- (Cmdty) 32-bit/33MHz = 1.1 Gbps

■ Disk

- (Rsrch) 250MBps = 2.0 Gbps
- (Cmdty) 20MBps = 0.2 Gbps

■ Specific numbers obviously open to debate.

- Please save any debate till the end, please :-)

■ The point:

- As all-optical networks become commodity, they gain significantly in performance.
- Current commodity electronics do not scale as quickly.

07: Pros and Cons

Other Approaches

- Not scalable
- Not very efficient
- + Very easy to use
- O(us) resolution timers
- Brute-force capture
- + Some are portable
- Some not real-time
- Some are unreliable
- Some are expensive

TICKET

- + Scalable
- + Very efficient
- More difficult to use
- + O(ns) resolution timers
- + Parsing capture
- Not portable
- + Can perform in real-time
- + Highly reliable
- + Free to inexpensive

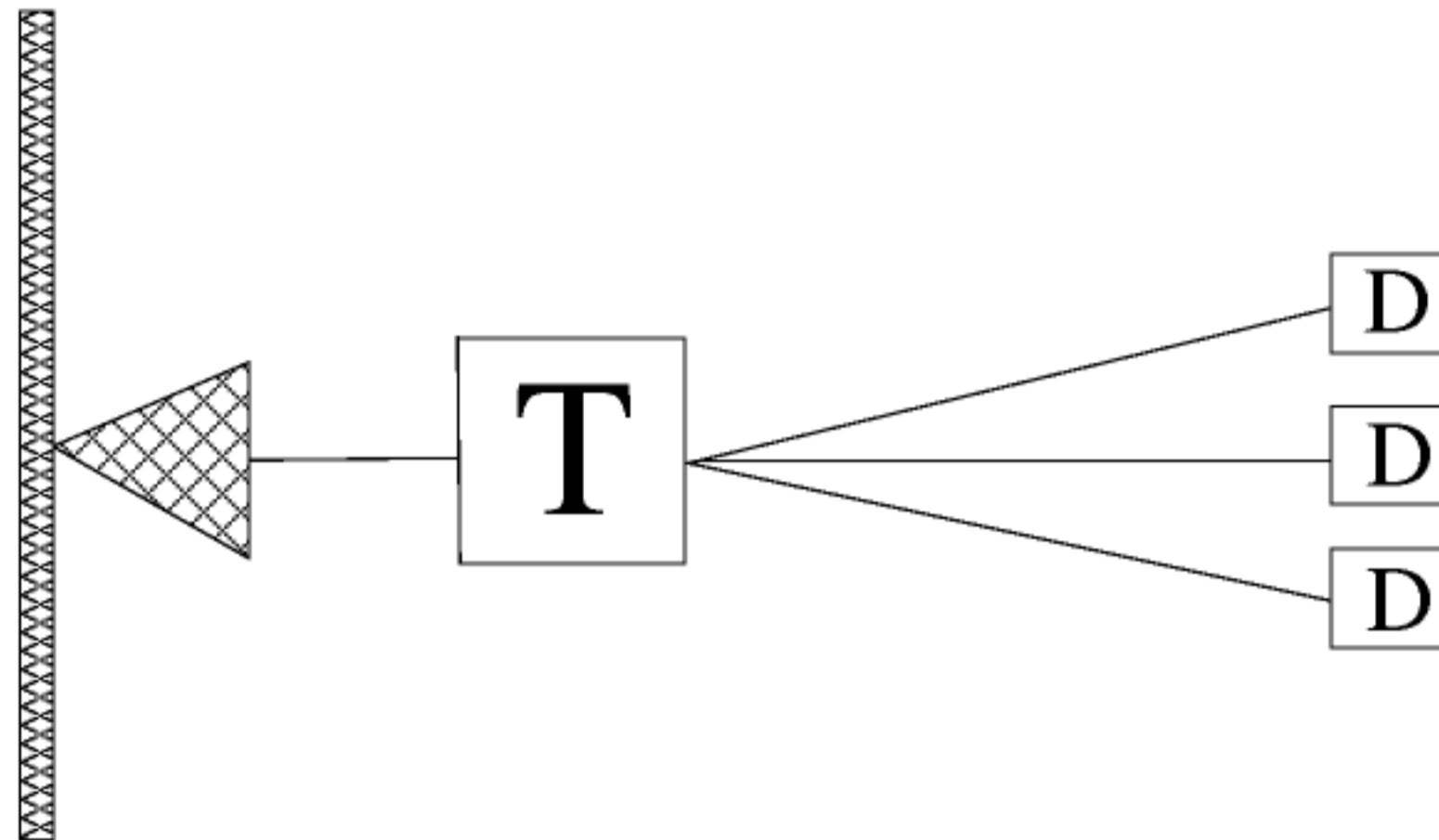
08: Infrastructure

We require something upon which there is (a copy of) traffic to be observed.

- Network Stack --- copy memory-to-memory
 - like libpcap, only local traffic without...
- Switch Port --- copy via port mirroring
 - can kill performance of switch
- Optical Tap --- copy via physical signal split
 - good for line-rate backbone traffic, may have full-duplex issues.
- Promiscuous NICs --- "copy" for wireless
 - Required for non-local traffic

09: Methodology

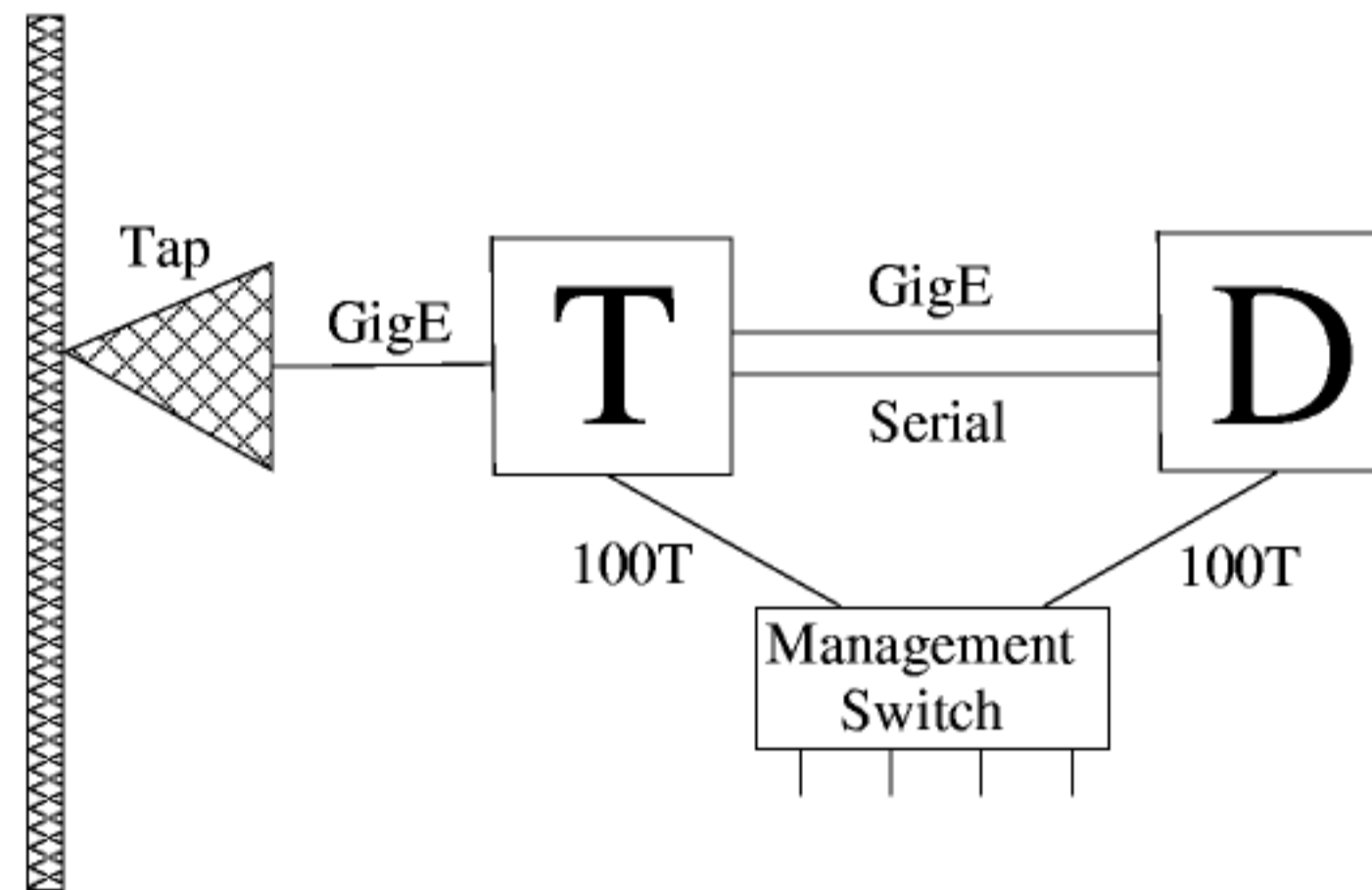
TICKET runs on two or more cooperating hosts:



- One (T) runs dedicated OS
 - initial collection (off the wire)
 - striping data...
 - maximizes efficiency, performance, security.
- One or more (D) running user level tools
 - process (heavy-duty analysis)
 - save (RAID systems)
 - display (multiple consoles).
 - maximizes usability, expressive power.

10: First (simplest) configuration.

Sufficed for our uses.



- One TICKET box (T), one Destination box (D).
- Destination box is a 1.2TB RAID system.
- Management interface, serial cable for debugging.

11: Kernel & Host Algorithm

Kernel

- Probe hardware
- Parse the kernel command line for options.
- Configure network interfaces
- Call 'mode' function (traffic collection)
- Loop:
 - Wait on mutex activated by interrupts-- no scheduling!
 - Receive, timestamp packet
 - Collect information
 - If output buffer is full, select interface to send and enqueue.
 - Send pending output.

Host

- Get a packet
- Call handler (Save).

12: Information Collected

- 64-bit timestamp

- Length ``off-the-wire''

- Ethernet Information:
 - Addresses of Source and Destination
 - Type of service or 802.3 Length

- IP information
 - Addresses of Source and Destination
 - Lengths of Packet and Header
 - Protocol Number

- TCP/UDP information
 - Ports of Source and Destination
 - Length of UDP packet or TCP header

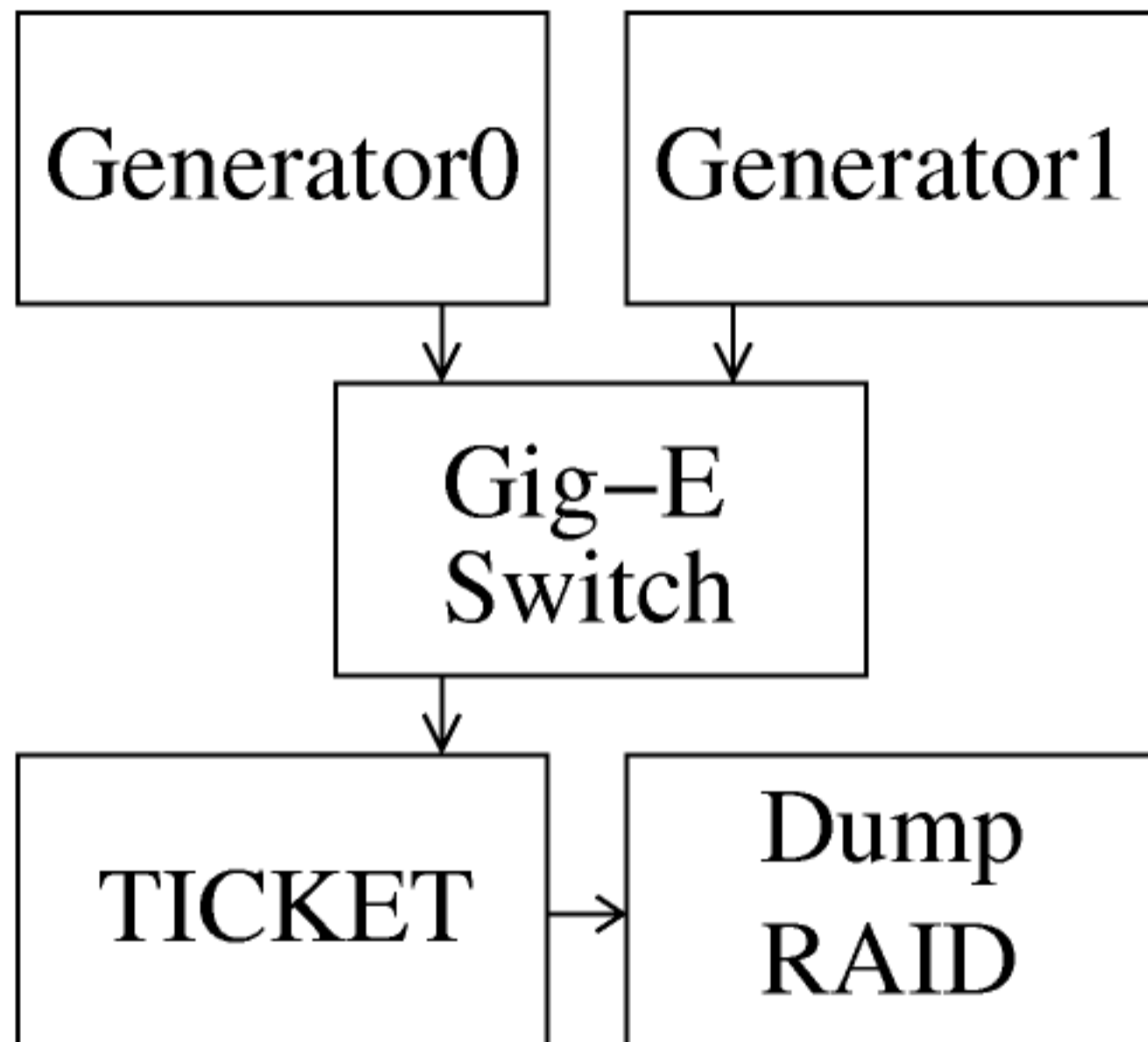
- Capable of 'snaplen' style capture
 - One output mode will be compatible with
`tcpdump -w outfile`

13: Experimental dimensions tested

Interested in:

- Can we 'keep up' at line rates?
 - "Flooding" tests
- Accuracy (timestamps, data collection)
 - Flooding, real-world backbone data collection
- Precision (timestamps)
 - Is higher precision of `rdtsc()` really there, or just noise?
(PCI bus, NIC interrupt coalescing, different code paths, etc)
- What's our control?
 - Tcpdump/libpcap.

14: Setup, topology, versions.



'Generators' send 500Mbps of traffic in 1KB UDP packets (1066B with headers).

Traffic aggregates to 1Gbps at switch, forwarded to TICKET machine for collection.

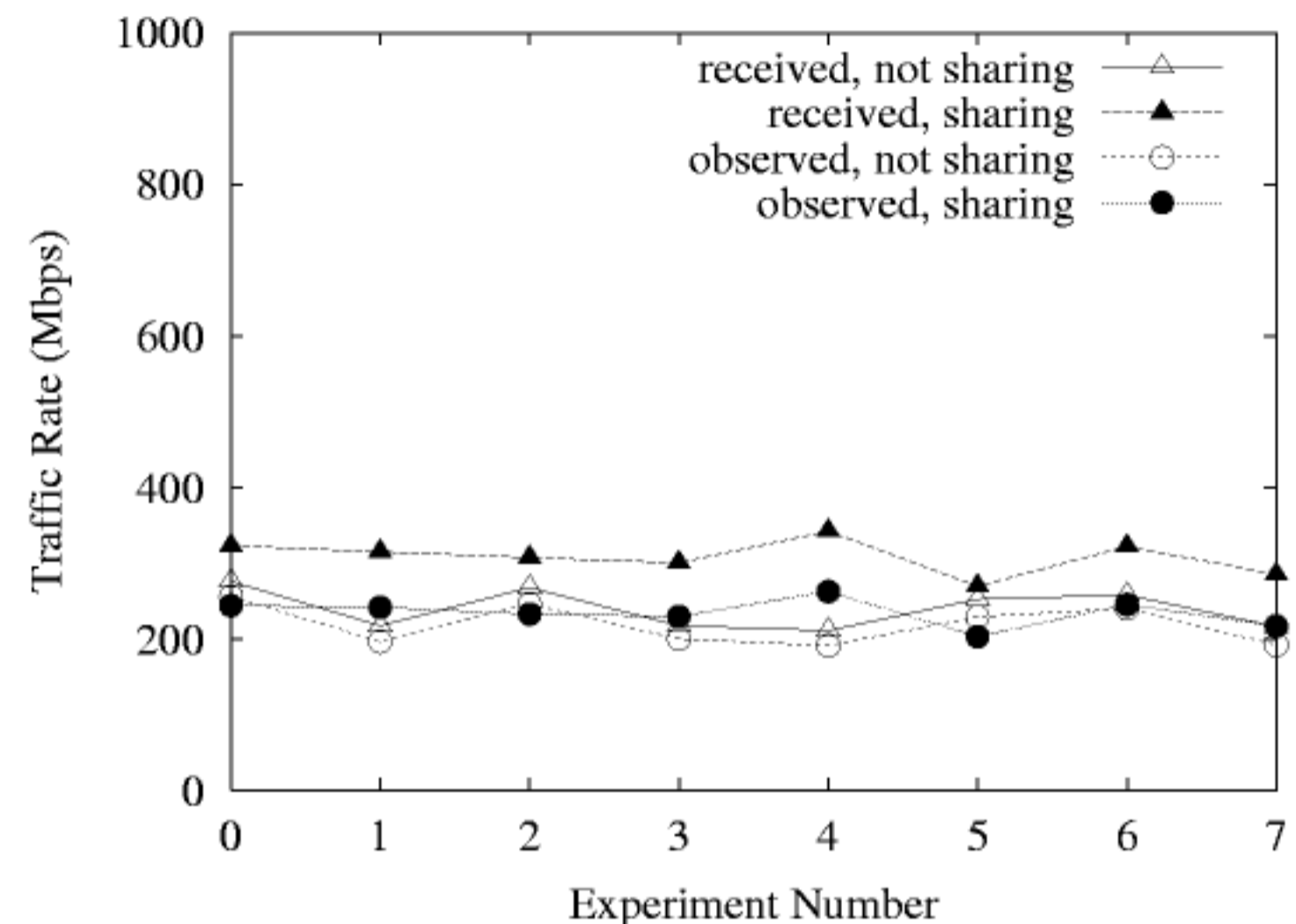
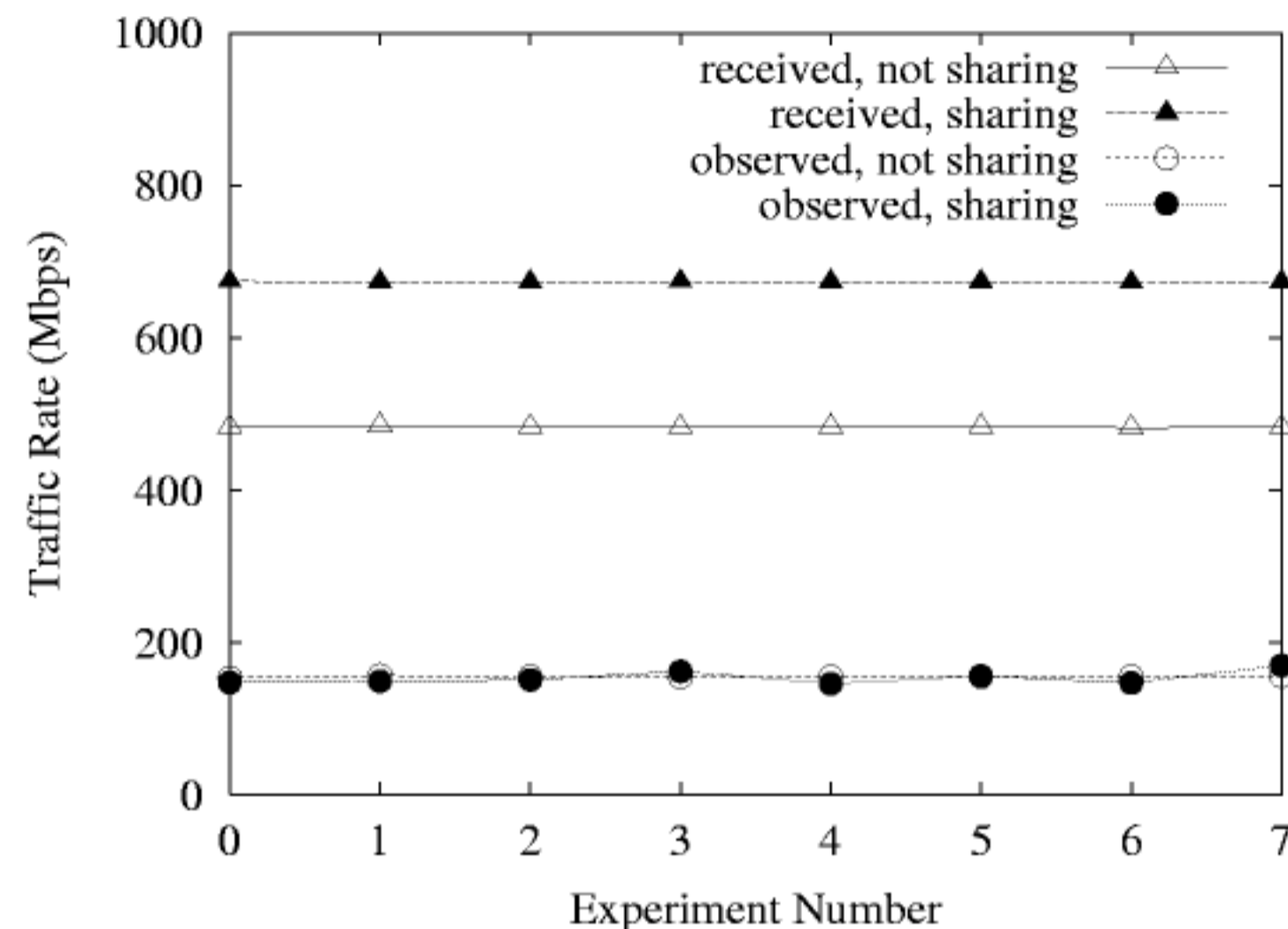
Tests

- Longest code path
- "Average" packet size

Ignores

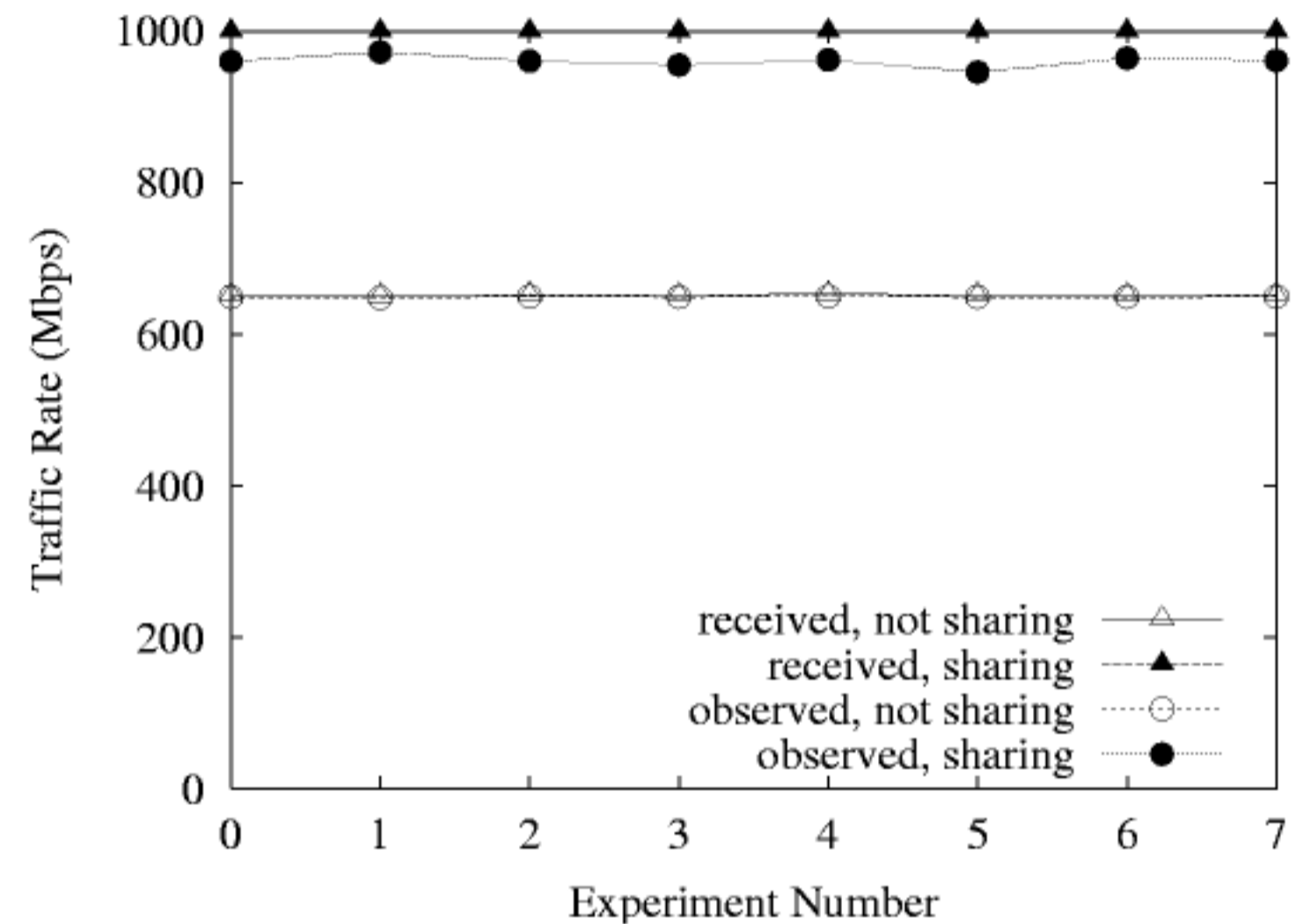
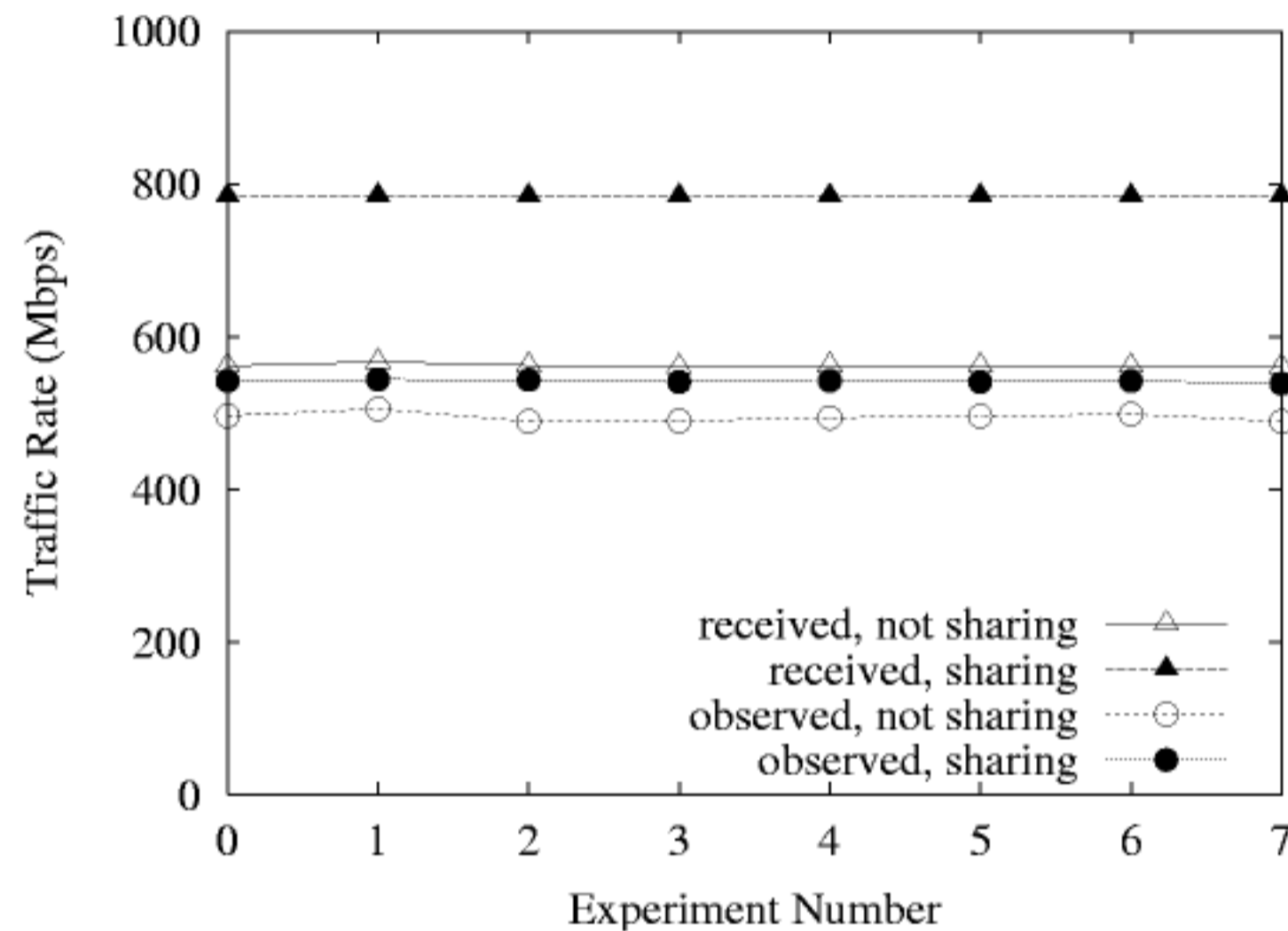
- Short-term high PPS bursts w/ small packets

15: Tcpdump, Saving Data



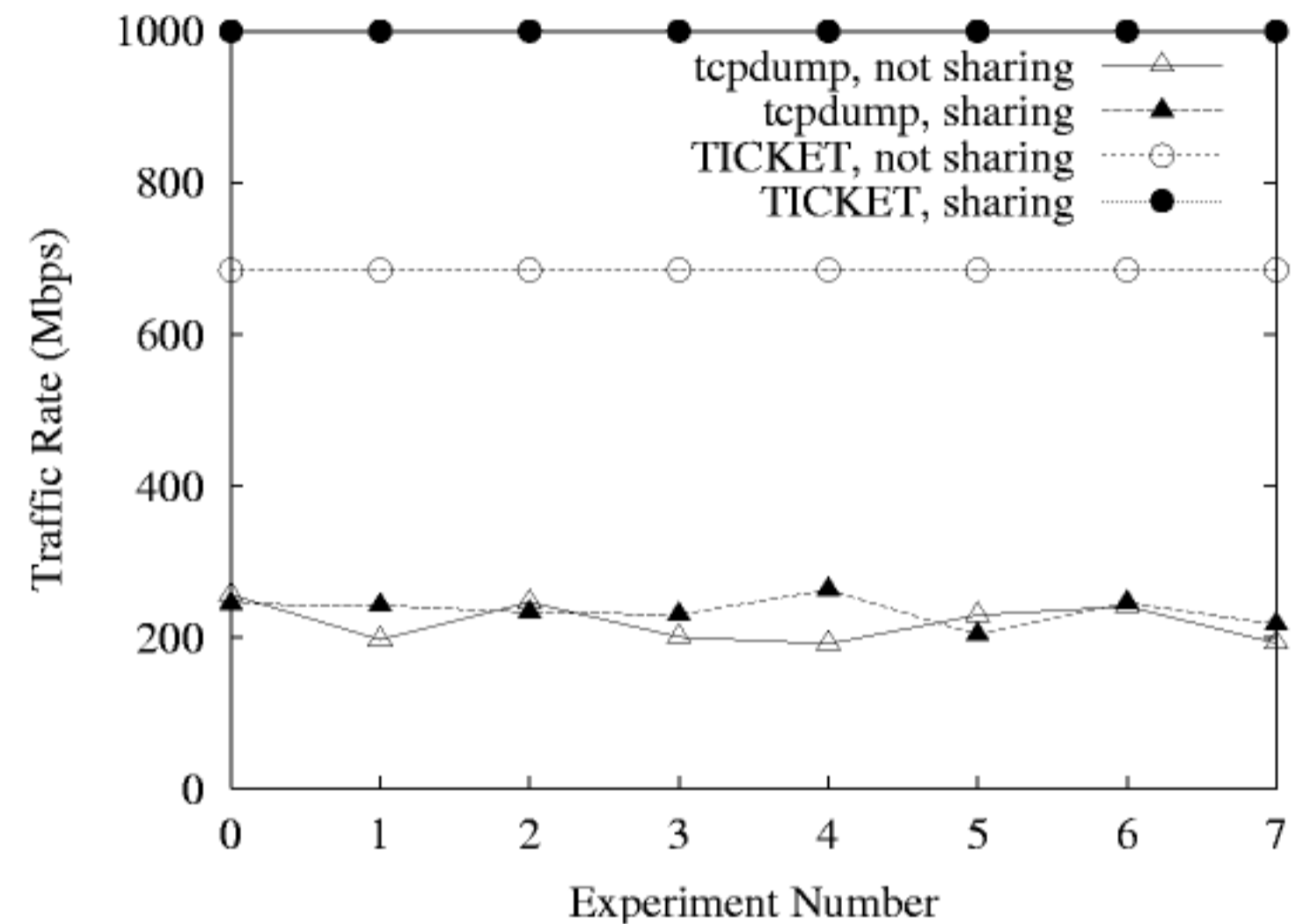
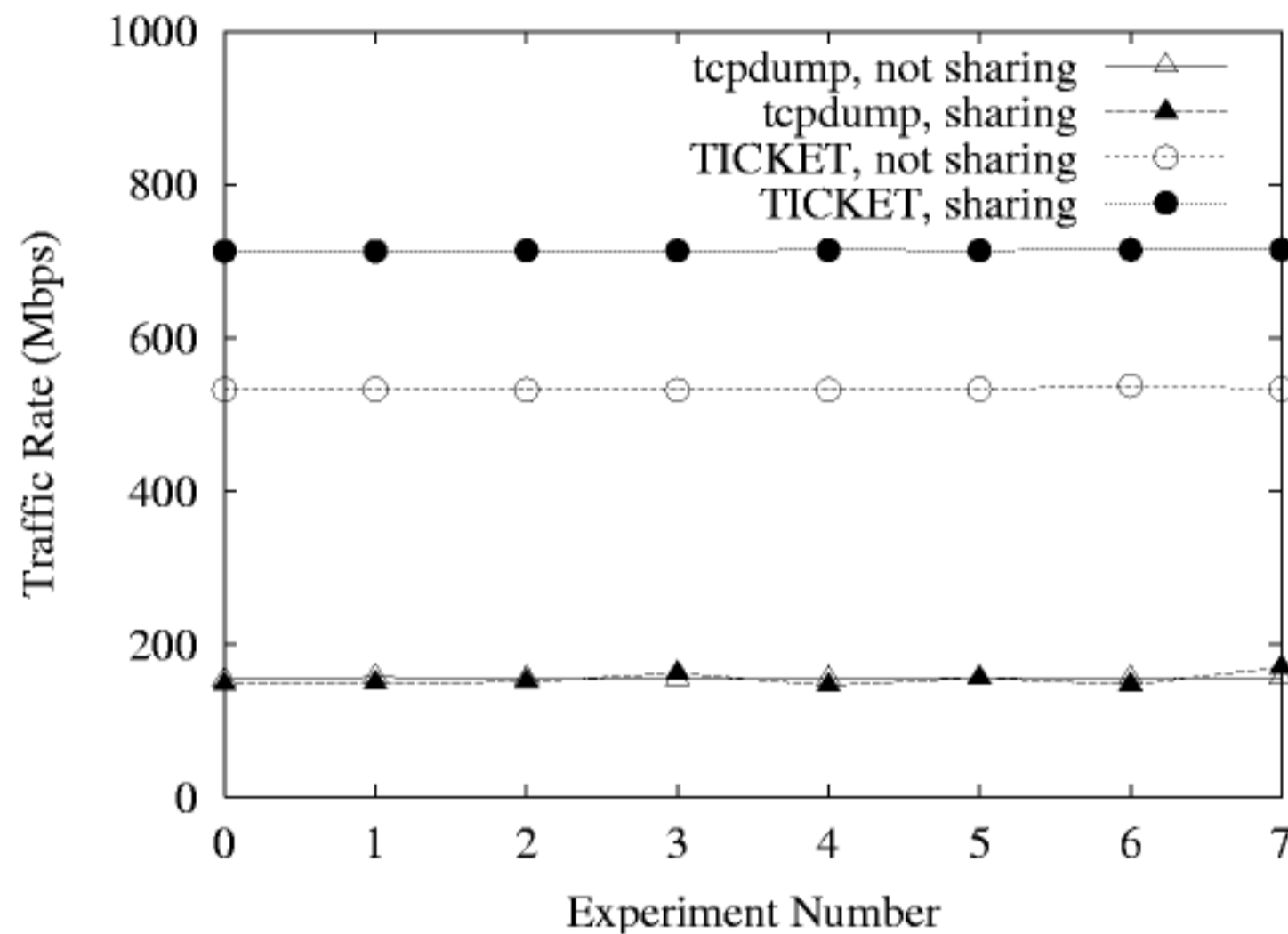
- Left image is 'slow' machine, right is 'fast' machine.
 - 2x400MHz Pentium II with 32bit,33MHz PCI bus
 - 1x933MHz Pentium III with 64bit,66MHz PCI bus*
- Slower machine saves below 20% of the data!
- Faster machine little more; disk limited.

16: Tcpdump, Not Saving Data



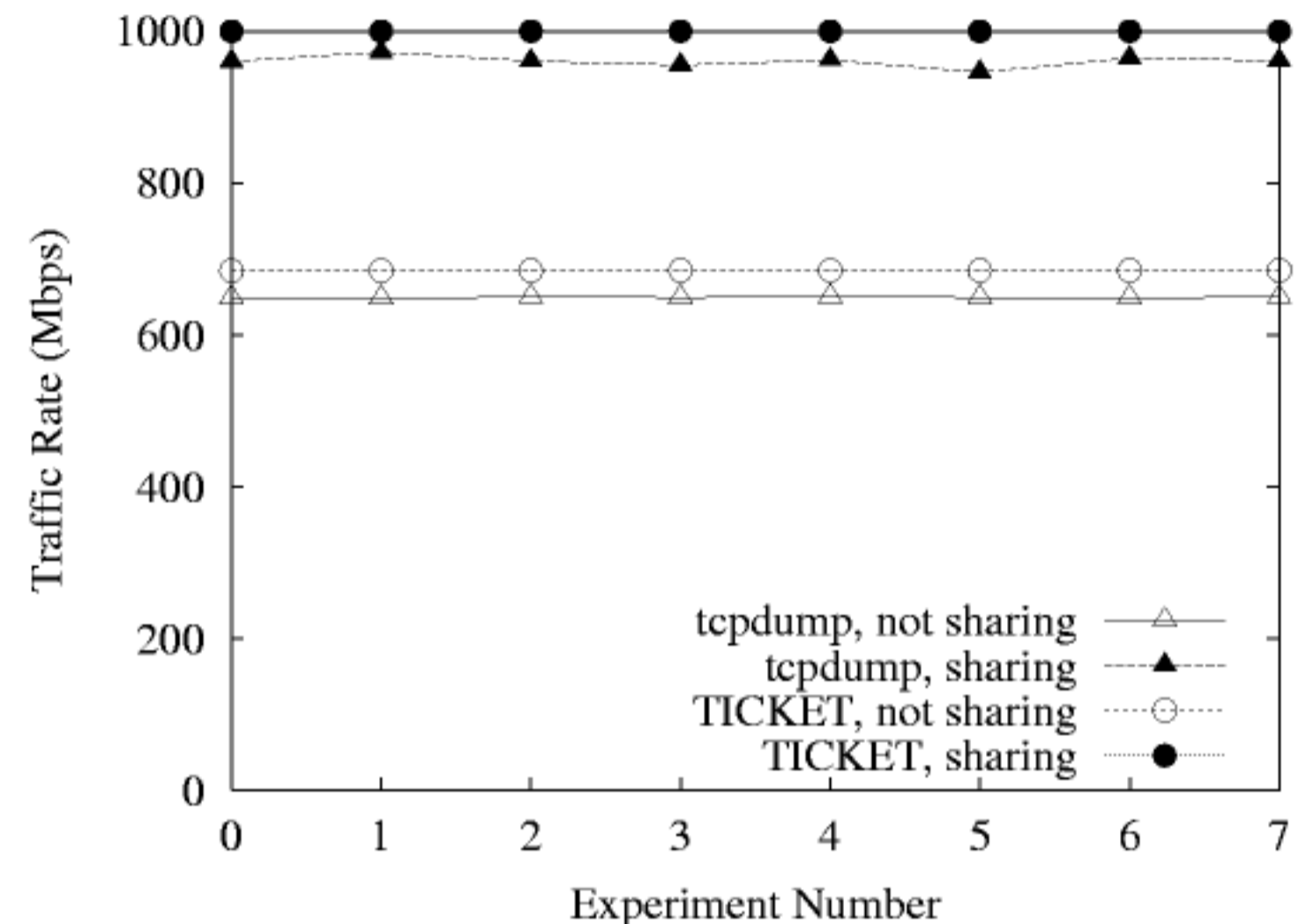
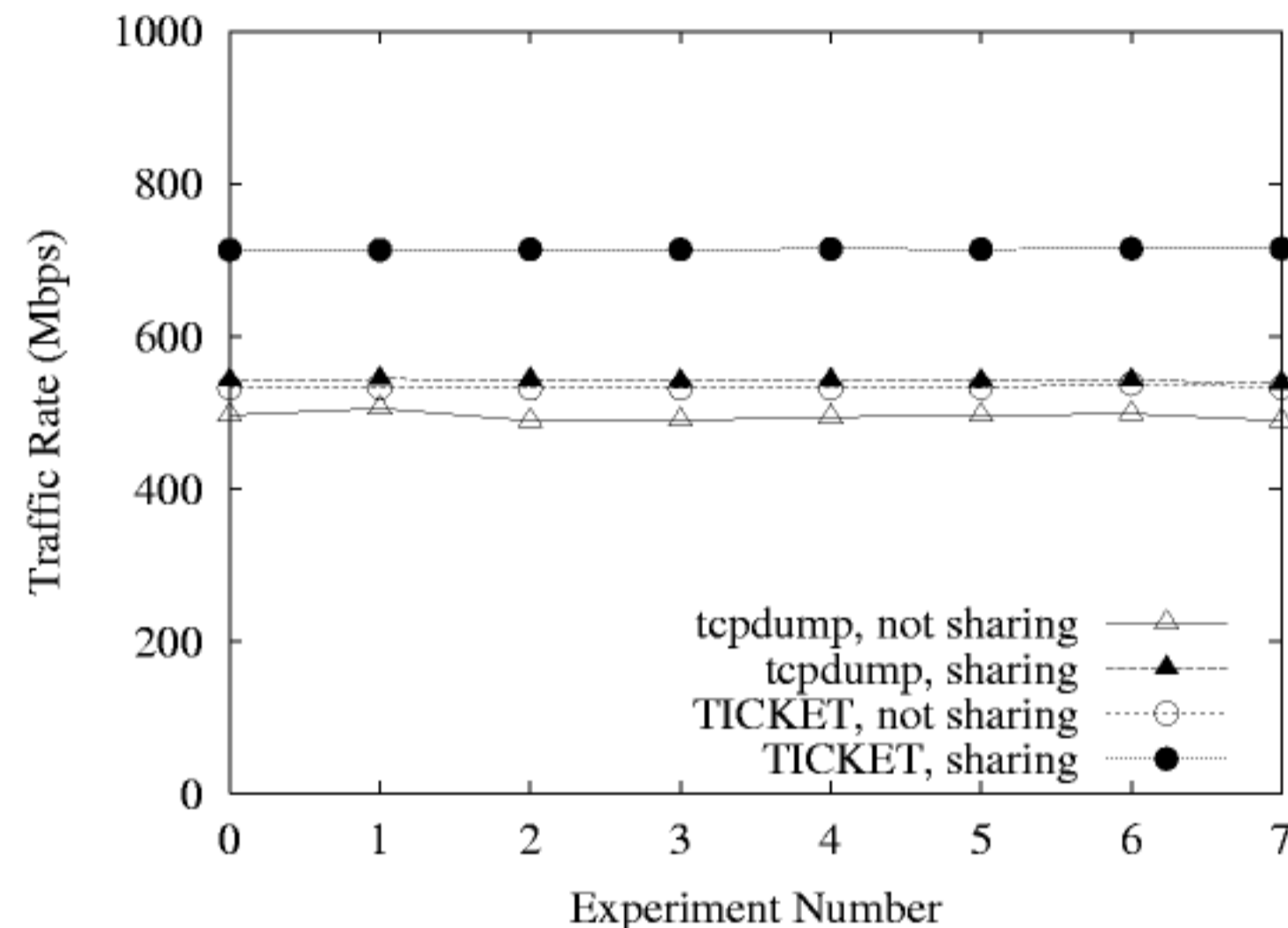
- We can use libpcap to collect data, then send it elsewhere (/dev/null, here) to be saved.
- Analogous to part of TICKET's functionality, but still has user-level issues (when to timestamp, resource contention).

17: TICKET & Tcpdump, Saving Data



- TICKET is the circles, and with load-sharing:
 - Observes 3x what tcpdump on the slow machine!
 - Observes 100% on the faster machine!
- Difference due to save on remote machine.

18: TICKET & Tcpdump Not Saving



- TICKET still outperforms tcpdump.
- Difference due to kernel-user context switch/copy.
- Performance results dominated by performance of NICs and PCI bus.

The End -- Contacts -- Questions?

For more information, visit

<http://www.lanl.gov/radiant/>

Or email me at

ehw@lanl.gov

Physics & Cross-Sectional Bandwidth

Don't think trends will continue?

Predicting the future is impossible, but what about:

- **Move to smaller process (sub 0.13 micron)**
 - Faster, more dense, a while before we hit quantum limits.
- **New doping, materials**
 - More layers per chip, better heat properties...
- **Enhanced parallelism, SMP machines**
 - Becoming commodity.
- **Organic, quantum computing**
 - Solving massively parallel problems.
 - Multiple bit encoding on states of single atom, entanglement
- **Optical networking/computing**
 - DWDM, streaming supercomputers
- **Cross-sectional Bandwidth**
 - Super-exponential growth--with constant Bandwidth/link
- **Better software, compilers**
 - Growth isn't just in hardware