Washington University in St.Louis

Engineering

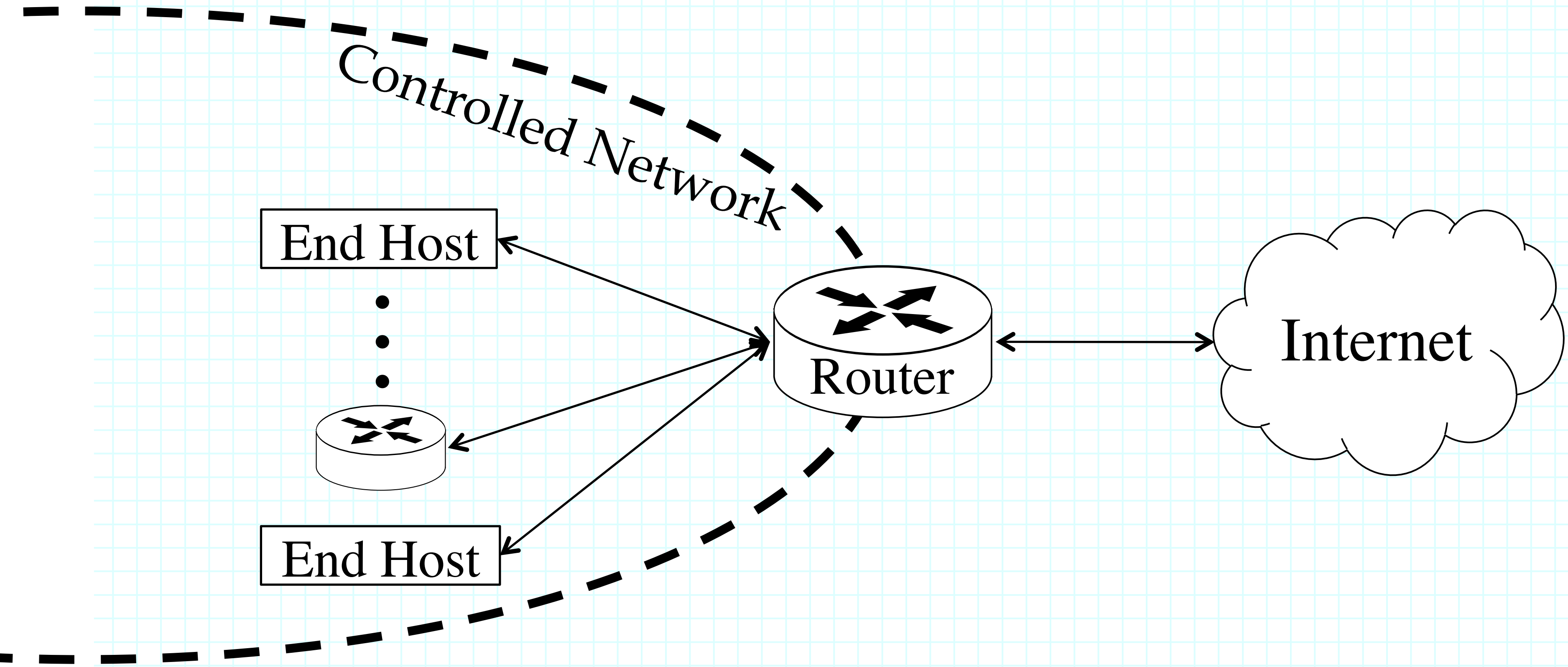
A Passive Network Appliance for Real-Time Network Monitoring

Michael J Schultz, Ben Wun, and Patrick Crowley
Applied Research Laboratory
Washington University in Saint Louis

This material is based upon work supported by the National Science Foundation under Grant No. CNS 096420

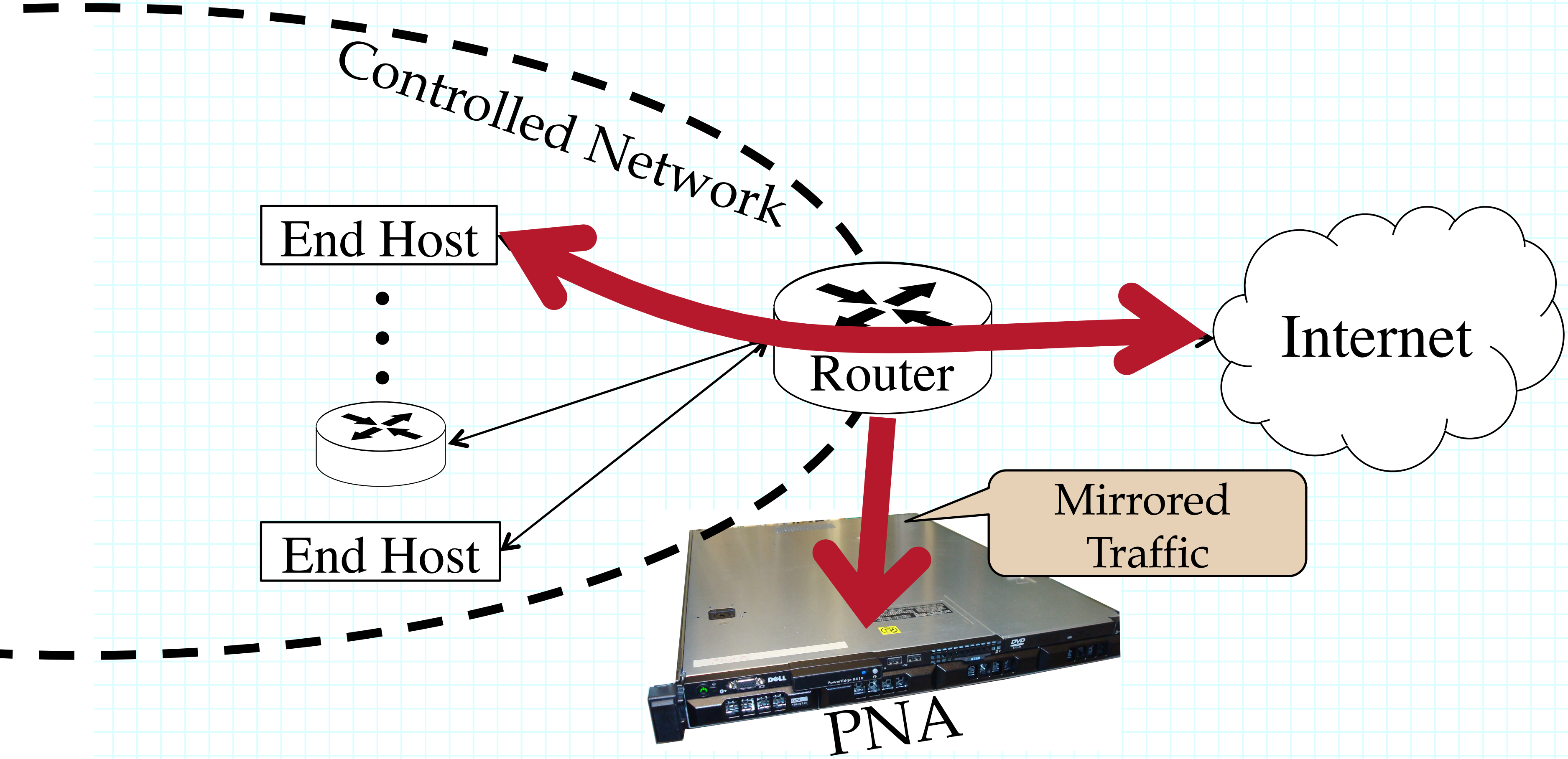
Million Mile View

- Network Operators want to know about their network



An Example

- End host suddenly opens many connections
  - » What happened? Not sure, didn't capture it – something bad?



Getting Data from a Network

- Capture complete packet traces? Nope.

`Disk error: Out of space`

- Sample packets? Nope.

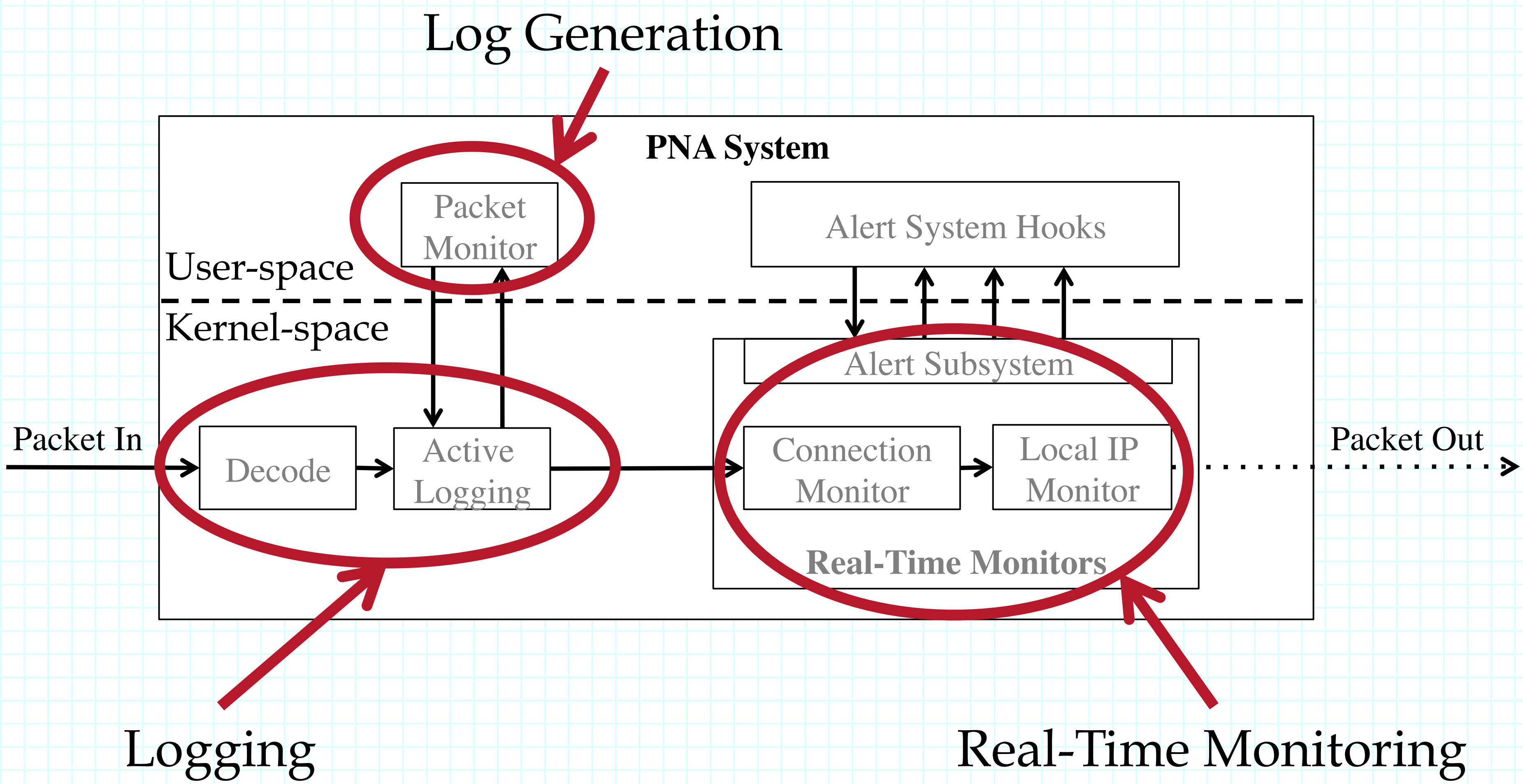
G	O	O	D	E	V	I	L	G	O	O	D	E	V

- Special purpose equipment is costly/hard to maintain

The Passive Network Appliance

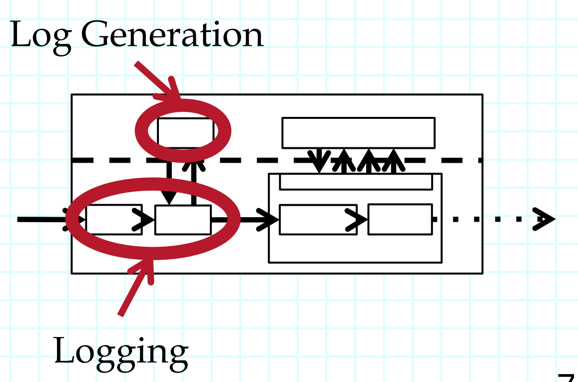
- Re-evaluate what modern commodity hardware can do
- First kernel-space network monitor (that we know of)
- Specifically
  - » Present our kernel-space network monitor
  - » Explain our API that allows monitors to enforce policy at network frame granularity
  - » Quantitative comparison between user-space and kernel-space monitors

PNA Design


Log Generation
PNA System
Packet Monitor
Alert System Hooks
User-space
Kernel-space
Alert Subsystem
Packet In
Decode
Active Logging
Connection Monitor
Local IP Monitor
Packet Out
Real-Time Monitors
Logging
Real-Time Monitoring

Logging and Log Generation

- First and Foremost: Summarize the packet
  - » Where is the packet from?
  - » Gather up summary statistics (bytes, time, etc.)
- Flush records every 10 seconds to capture state of network
- Creates a file that can be aggregated to form continuous view of network

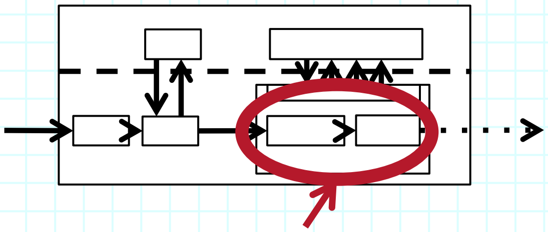


Real-Time Monitoring

- Allows network administrators enforce policy *as network frames arrive*

- Chain arbitrary number of monitors together
  - » Has no direct effect on summary logging
  - » Indirect effect of slowing down the system

- Alerts can be generated *at the moment* malicious activity is detected



Real-Time Monitoring

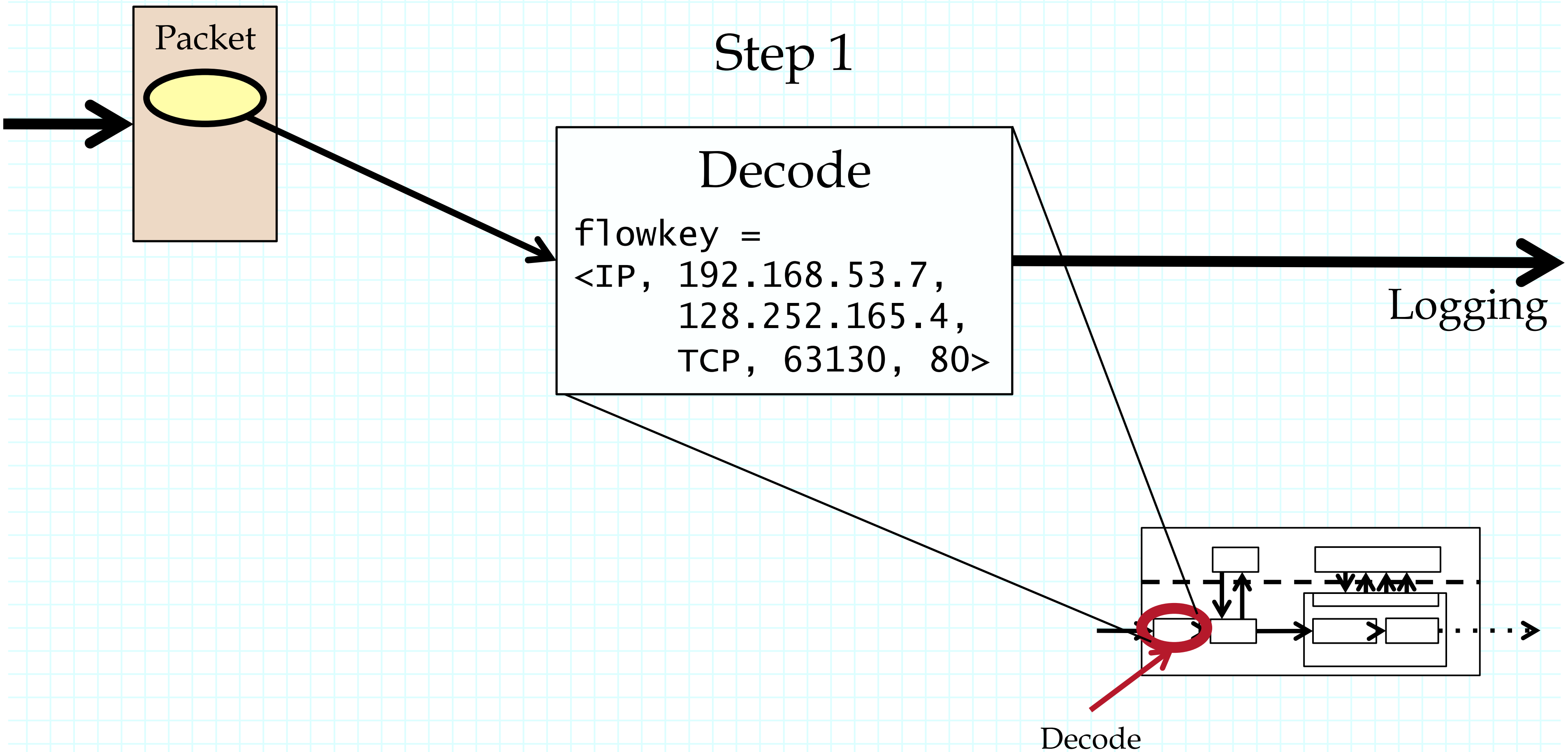
Implementation Details

- Linux Kernel Module
  - » Implies that it will have less overhead than any user-space monitor*
- Runs on commodity hardware
  - » Servers are relatively low-cost (<$3000)
  - » Un-patched Linux Kernel

* We'll get into that a bit later.

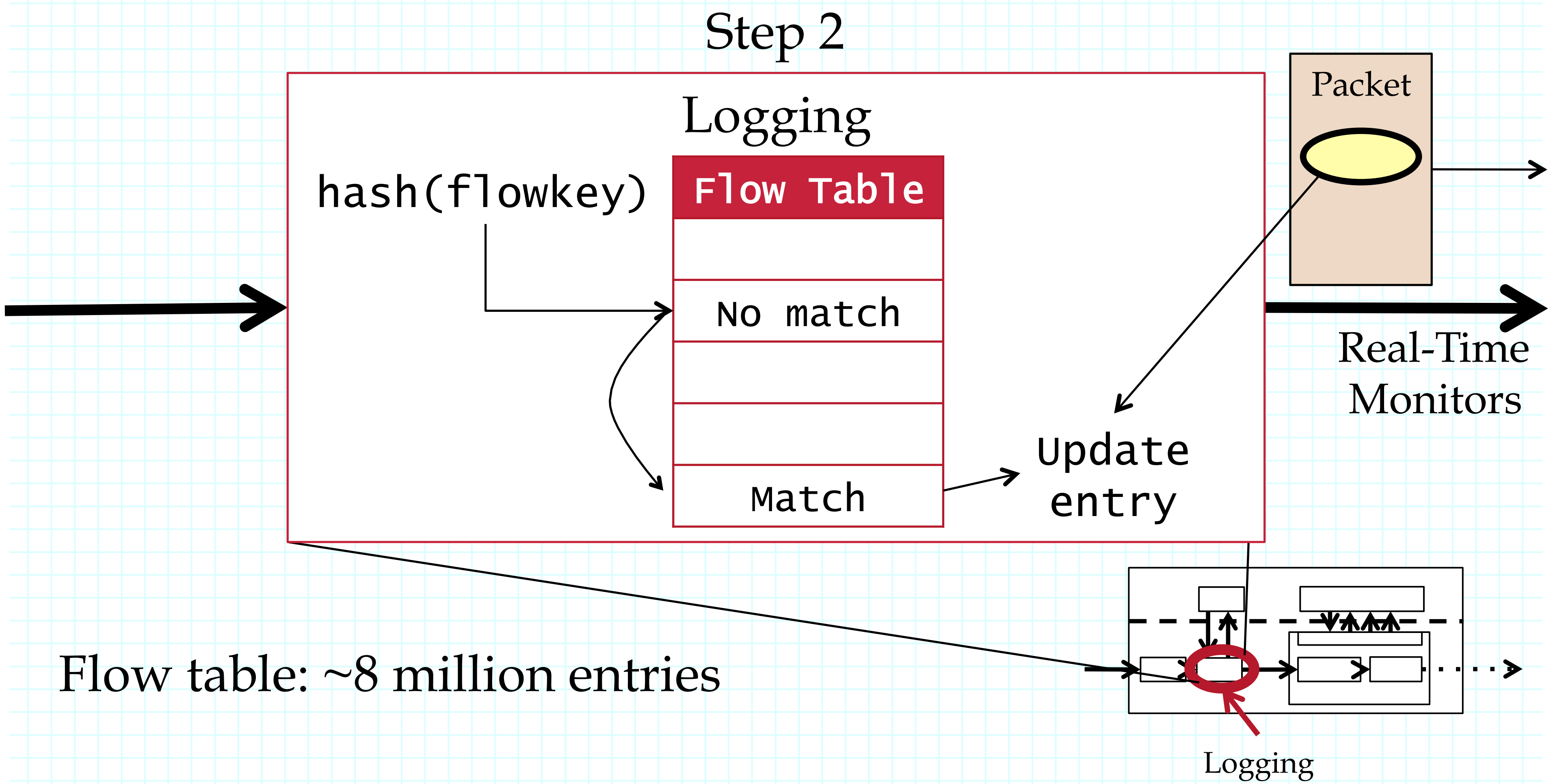
Decode

- Must be quick (every frame is logged)



Logging

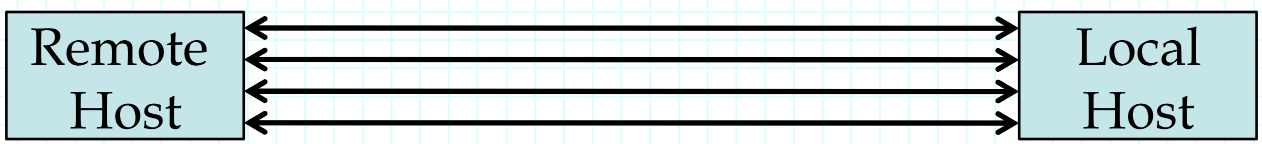
- Must be quick (every frame is logged)



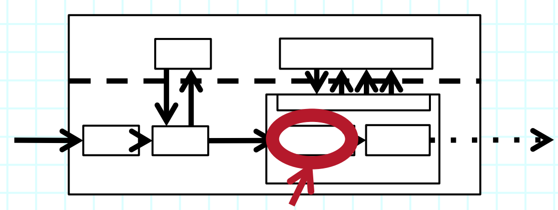
Flow table: ~8 million entries

Real-Time Monitors

- Every frame passes through monitors
- Enforce network policy at *per frame* granularity
- Example: Connection Monitor



×1000 conversations = probably bad

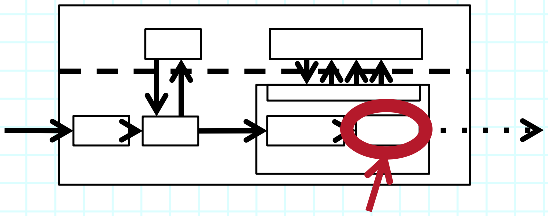


Connection Monitor

Extending the System

- Example: Find all HTTP traffic (on non-standard ports)
  - Write a `hook()` function
  - Look at payload for request method/response status
  - If found use `pna_alert()` to alert network operator

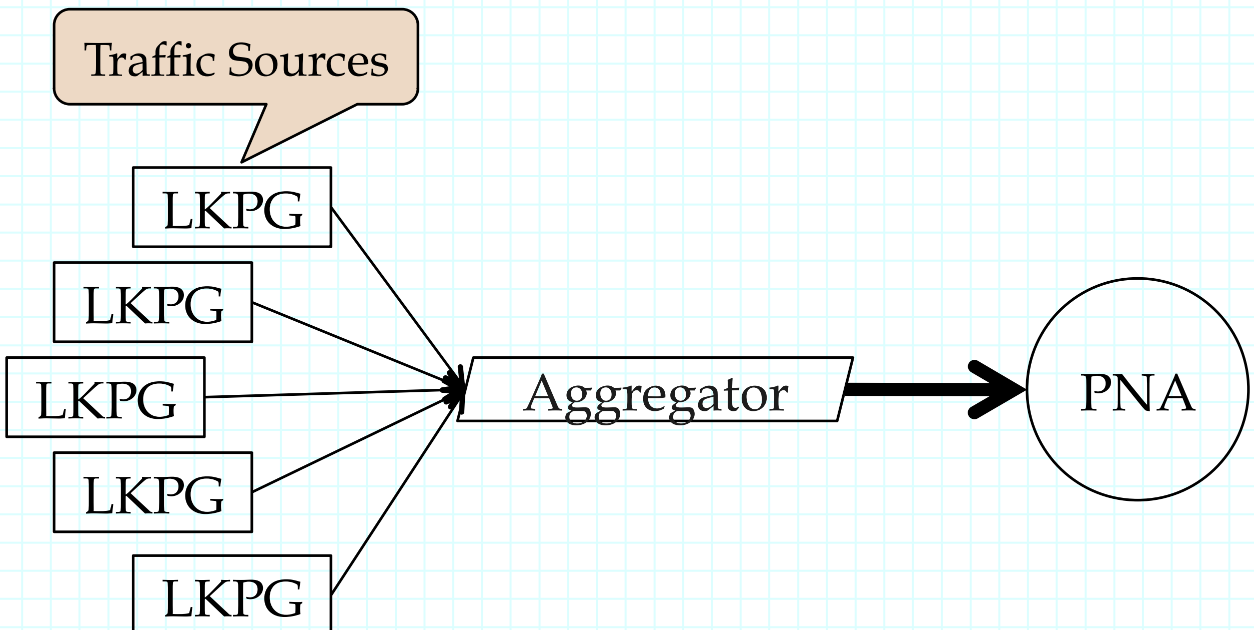
- Other functions
  - `init()` and `release()` prepare/destroy global resources
  - `clean()` runs every 10 seconds and can perform data maintenance



Your Monitor(s)!

Evaluation

- Tested with worst-case and real-world conditions
- PNA System
  - » 2.27 GHz "Nehalem" with 12 GiB memory
  - » Allows about 8 million flow table entries



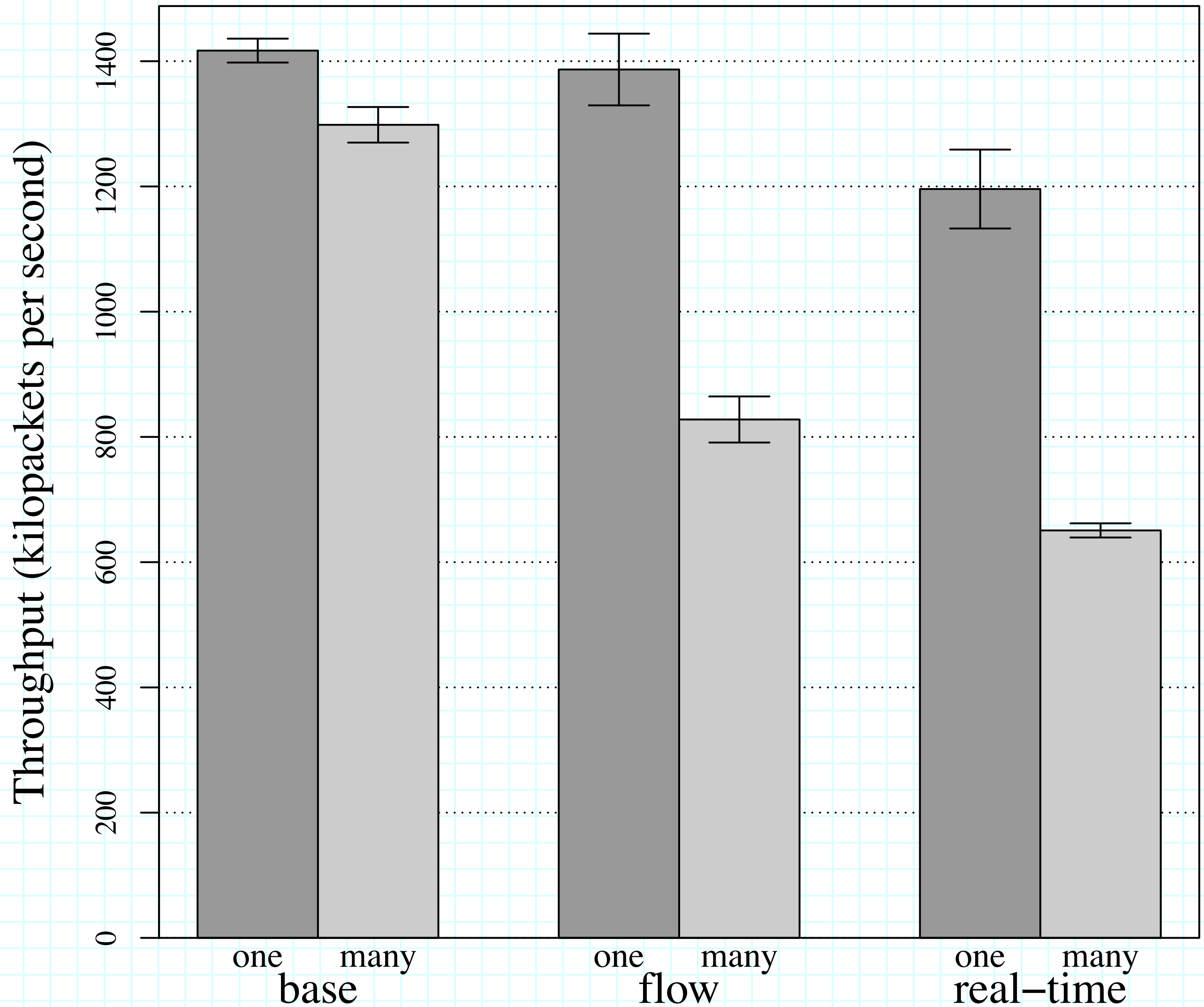
LKPG = Linux Kernel Packet Generator

Laboratory Experiments

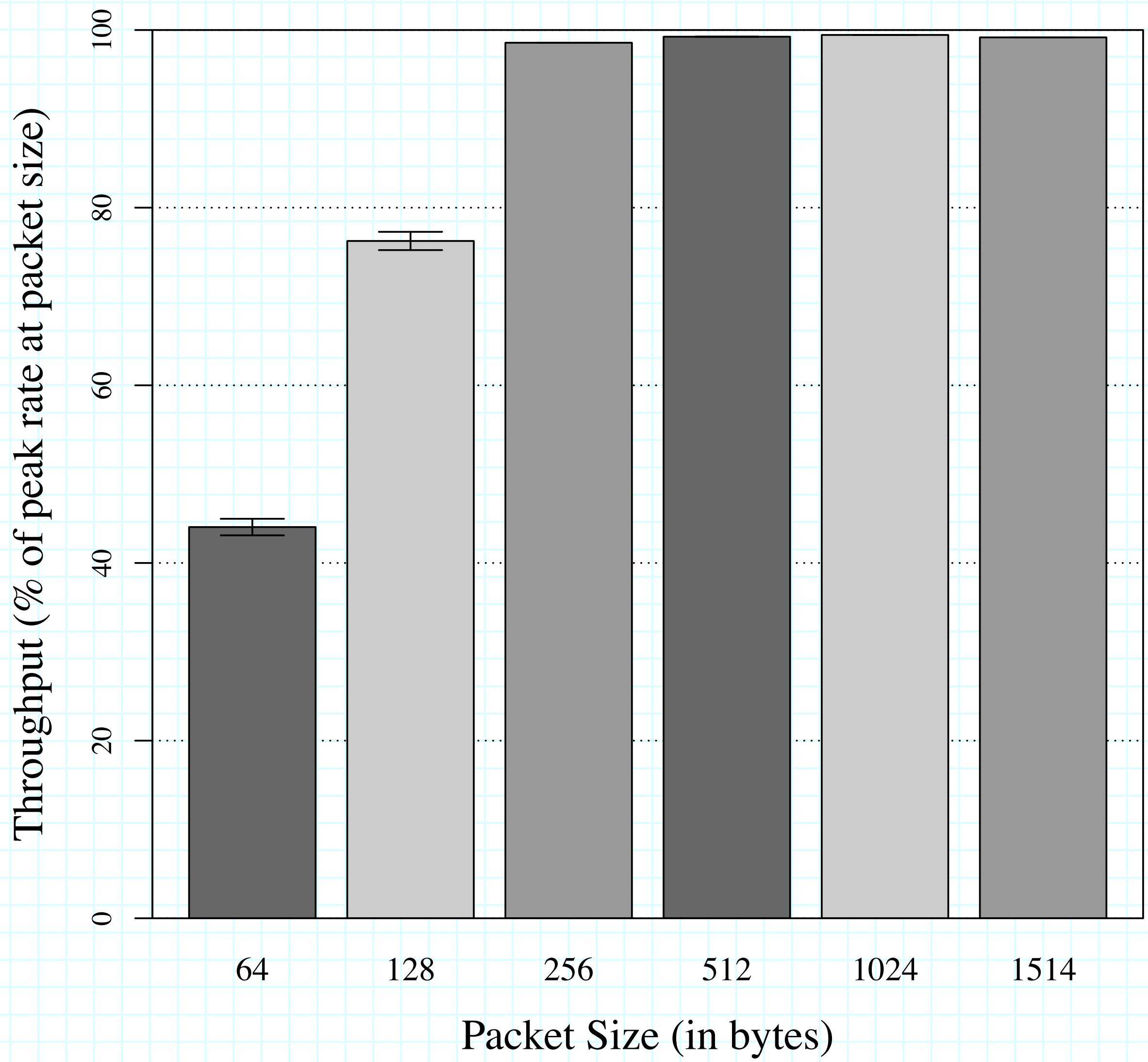
- Ran with "base," "flow,", and "real-time" monitors

	Minimum sized packets	Maximum sized packets
Single flow	**Min** table insertions **Max** packets/second	**Min** table insertions **Min** packets/second
Many flows	**Max** table insertions **Max** packets/second	**Max** table insertions **Min** packets/second

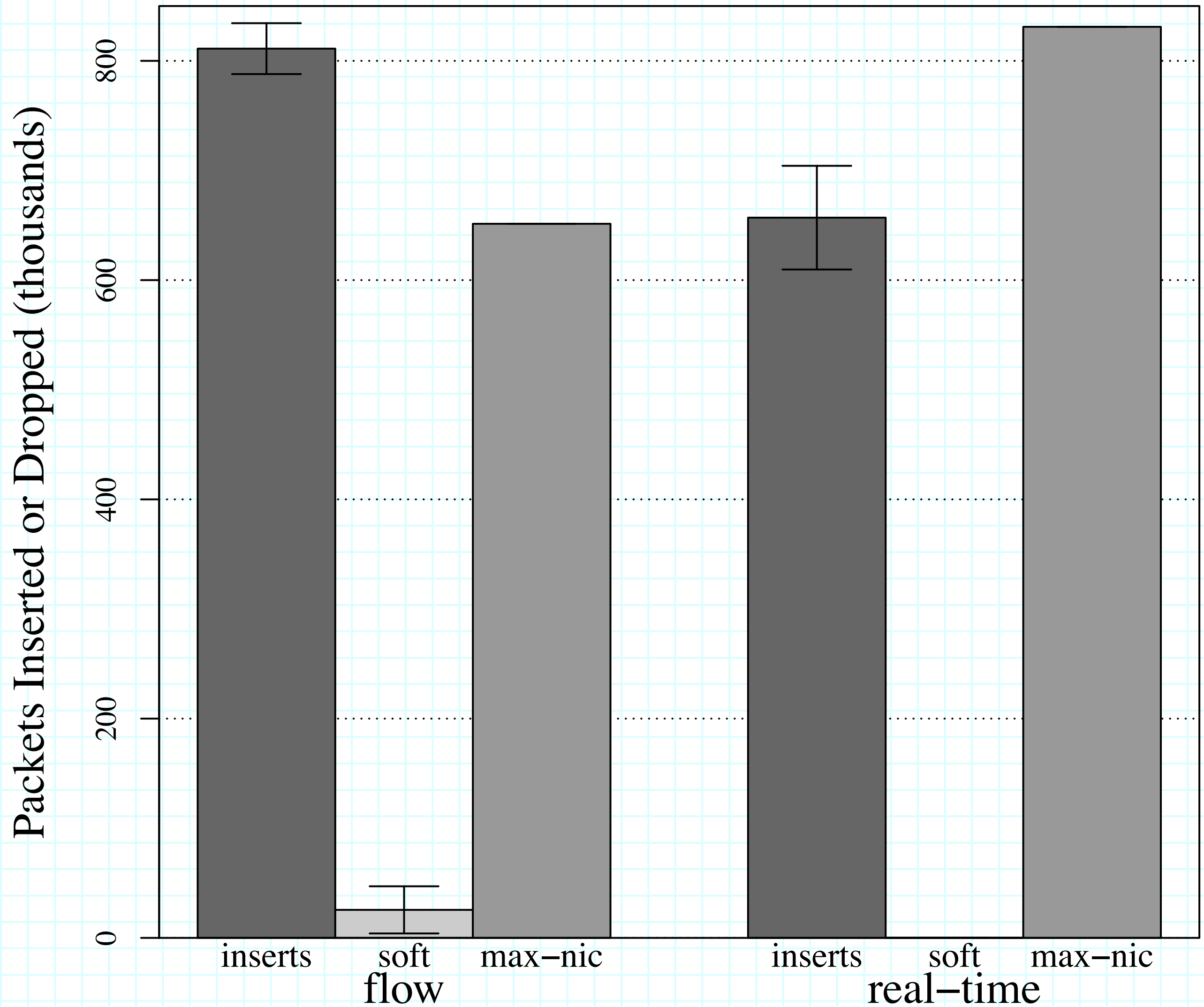
Min-sized Packet Throughput



Throughput at Various Packet Sizes

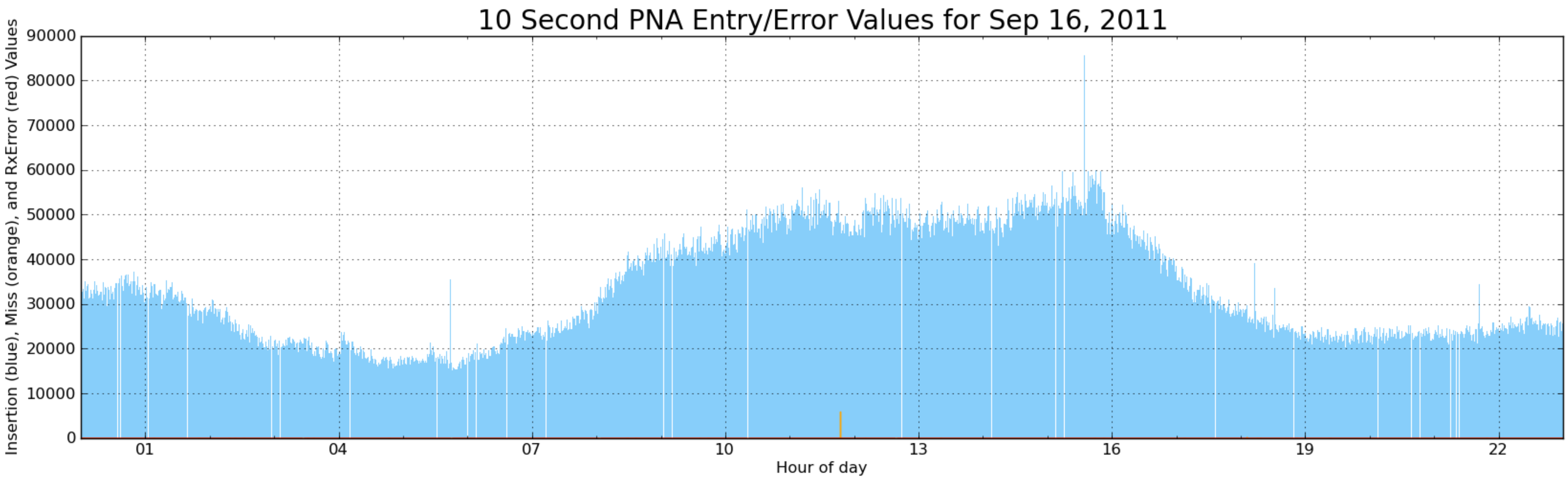


Packet Entries/Drops (per second)



Back in Reality

- Real networks don't see 1.48 Million packets per second
  - » Average packet size PNA sees is about 1000 bytes
- Graph of insertions (blue)/misses (orange)/drops (red)
  - » Per 10 second period



Kernel-space v. User-space

- Known that syscall overheads hurt performance
  - » Prior work minimizes syscall overheads (Deri [7], Braun [5])
  - » What if we *avoid* syscalls altogether?

- Measure single-core performance: capture, count, drop

	Linux Default	PF_RING	Kernel Module
Throughput (Mbps)	**495.89 ± 1.01**	**747.72 ± 7.38**	**951.75 ± 1.23**

Summary

- PNA kernel module gives complete snapshots
- API for real-time monitors to enforce policy *as frames arrive*
- Evaluation under worst-case and real-world conditions
  - » PNA logs *at worst* 43% of traffic
  - » Typically captures all the traffic @ 1 Gbps
- Comparison of Linux default/PF_RING/kernel module

Code available at www.github.com/pcrowley/PNA