eCounter

Inline Per-IP Network Monitoring at Millisecond Resolution via

Authors:

Xinxin Mei
Jie Chen
Amitoj Singh
Ilya Baldin
David Lawrence

INDIS

In cooperation with

TCHPC

IEEE COMPUTER SOCIETY

Jefferson Lab

Prometheus Node-Exporter

Prometheus

+

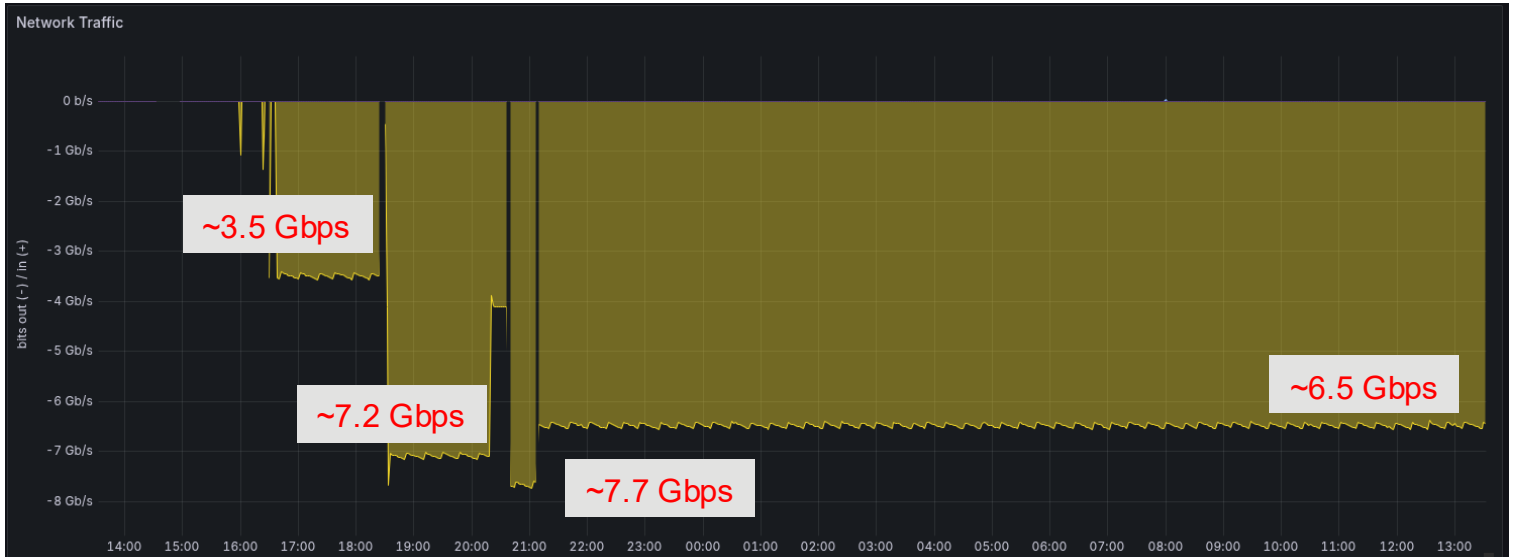
Grafana

```
[xmei@ejfat-5 ~]$ cat /proc/net/dev | column -t
Inter-|         Receive        |          Transmit
face            |bytes         packets    errs      drop    fifo  frame  compressed  multicast|bytes  packets      errs      drop  fifo  colls  carrier  compressed
lo:             3373861614     615741     0         0       0     0      0           0                  3373861614   615741    0     0     0      0        0           0
lowspeed-1:     6717090072     41024376   0         0       0     0      0           1792133            96569961274  73062941  0     0     0      0        0           0
eno2:           0              0          0         0       0     0      0           0                  0            0         0     0     0      0        0           0
enp193s0f0np0:  0              0          0         0       0     0      0           0                  0            0         0     0     0      0        0           0
highspeed-2:    5351621679022  594655570  0         112264  0     0      0           175006             60820781919  40036176  0     0     0      0        0           0
ibp65s0f0:      0              0          0         0       0     0      0           0                  0            0         0     0     0      0        0           0
ibp65s0f1:      0              0          0         0       0     0      0           0                  0            0         0     0     0      0        0           0
```

Sampling frequency matters!

```
xmei@nvidarm:~$ cat cron-iperf3.bash
*/2 * * * * /usr/local/bin/iperf3 -B  src_IP  -c  dst_IP  -P8 -u -b1G -t 60
```

Every 2 minutes, I activated an *iperf3* process sending data at **8 Gbps** for 60 seconds.



MilliSampler, Meta's Success Story

A microscopic view of bursts, buffer contention, and loss in data centers

Authors: Ehab Ghabashneh, Yimeng Zhao, Cristian Lumezanu, Neil Spring, Srikanth Sundaresan, Sanjay Rao | Authors Info & Claims

IMC '22: Proceedings of the 22nd ACM Internet Measurement Conference • Pages 567 - 580
https://doi.org/10.1145/3517745.3561430

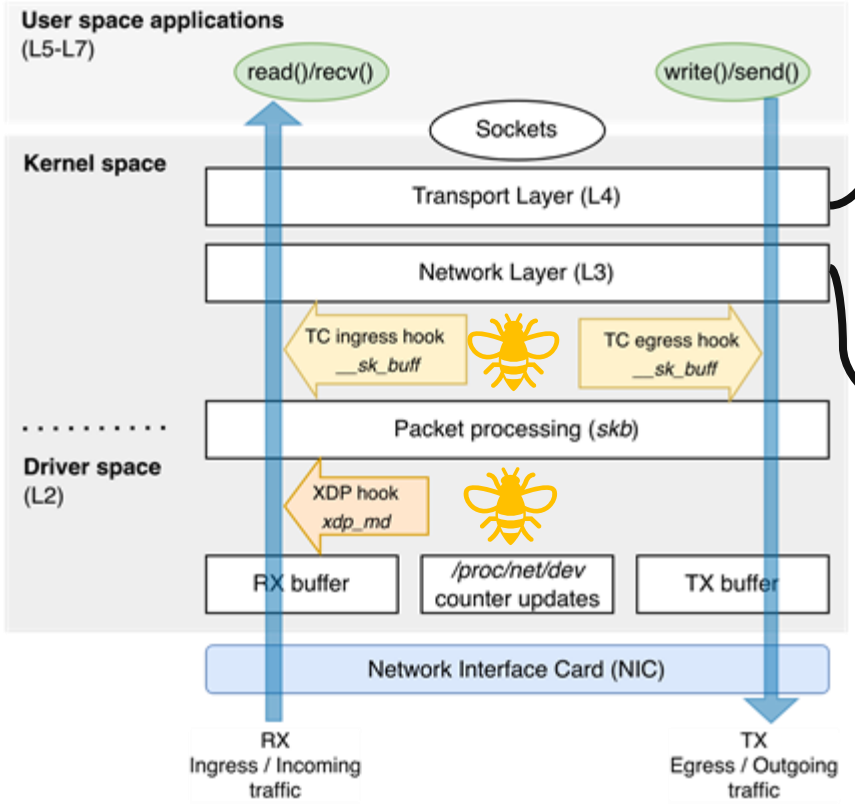
Meta's eBPF-based ***tc*** (traffic classifier) filter that runs in the Linux kernel to collect fine-grained networking data.

- eBPF sampling interval: 0.1, 1, 10 **milliseconds**.
- Metrics are collected per CPU core.
- Batch processing: each run contains 2000 eBPF samples; 9000 racks; *1 run per hour; run for 4 days.*

- https://dl.acm.org/doi/10.1145/3517745.3561430
- https://engineering.fb.com/2023/04/17/networking-traffic/millisampler-network-traffic-analysis/
- https://www.facebook.com/watch/?v=1135811427018906

Why can't we run MilliSampler continuously in a data center?

Linux Networking Stack



L4 Transport Layer

TCP Segment Header Format

Bit #	0 – 7	8 – 15	16 – 23	24 – 31
	Source Port		Destination Port	
32	Sequence Number			
64	Acknowledgment Number			
96	Data Offset	Res	Flags	Window Size
128	Header and Data Checksum		Urgent Pointer	
160...	Options			

UDP Datagram Header Format

Bit #	0 – 7	8 – 15	16 – 23	24 – 31
0	Source Port		Destination Port	
32	Length		Header and Data Checksum	

L3 Network Layer

IPv4 Packet Header Format

Bit #	0 – 7	8 – 15	16 – 23	24 – 31
0	Version	IHL	DSCP / ECN	Total Length
32	Identification		Flags	Fragment Offset
64	Time to Live	Protocol	Header Checksum	
96	Source IP Address			
128	Destination IP Address			
160	Options (if IHL > 5)			

L2 Link Layer

Ethernet (802.3) Frame Format

7 byte	1 byte	6 bytes	6 bytes	2 bytes	42 to 1500 bytes	4 bytes	12 bytes
Preamble	Start of Frame Delimiter	**Destination MAC Address**	**Source MAC Address**	Type	Data (payload)	CRC	Inter-frame gap

Image source: https://networkengineering.stackexchange.com/questions/25563/what-is-the-reason-for-the-different-order-of-the-source-and-destination-in-a-l2

eCounter Design

Agent	Layer	Prototype	#Metrics	eBPF map type	Resolution	Continuous monitoring
MilliSampler	L4	HTTP (TCP)	6	PERCPU_ARRAY	Sub-millisecond - millisecond	✗
Node-exporter	L2	N/A	2	N/A	Second - Minute	✓
eCounter	**L3**	**TCP/UDP**	**4**	**LRU_HASH**	**Millisecond**	✓

Inline (per-packet) L3 (per-IP address) millisecond-resolution network monitoring agent
Designed for scientific data acquisition (SciDAQ) system

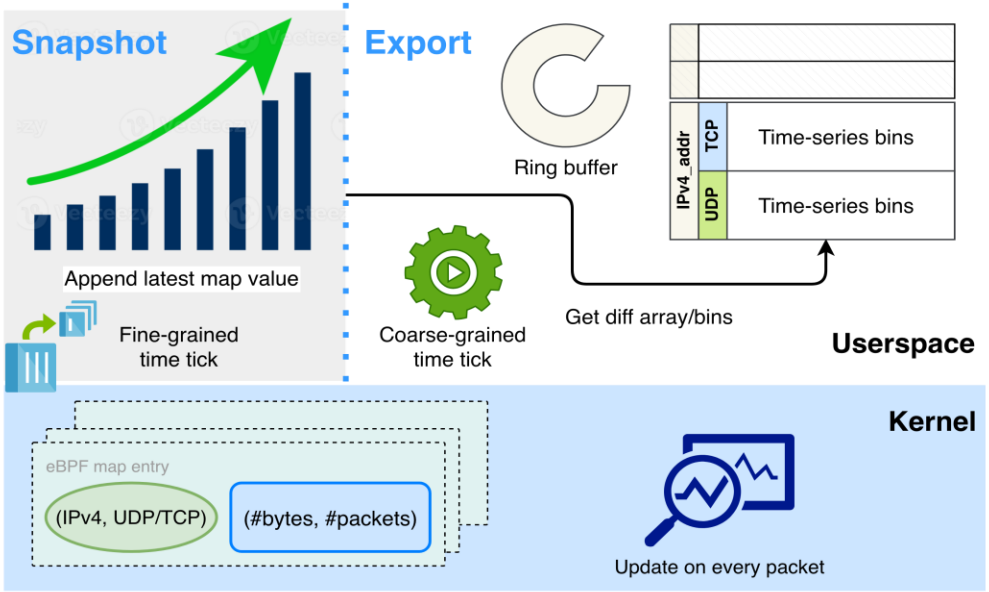
eCounter Architecture

```c
struct traffic_key_t {
    __u32 ip;  // IPv4 addresses denoted as integer
    __u8 proto;  // IPPROTO_TCP, IPPROTO_UDP, etc
    __u8 pad[3];  // pad to 4 bytes
};  // user-defined map key, 8 bytes

struct traffic_val_t {
    __u64 packets;   // packet number
    __u64 bytes;     // packet length
};  // user-defined map value, 16 bytes

// Define an eBPF LRU hash map named as "map_out_tc"
struct {
    __uint(type, BPF_MAP_TYPE_LRU_HASH);
    __uint(max_entries, 2048);  // entries allowed
    __type(key, struct traffic_key_t);
    __type(value, struct traffic_val_t);
} map_out_tc SEC(".maps");
```

Listing 1: Definition of our eBPF maps.



eBPF map size: 48KiB
Memlock: 185216 B
CPU L1 data cache: 4 - 5 MiB

Fine-grained timescale: 1/4000 - 1/20 seconds
Coarse-grained timescale: 1 second
Ring-buffer: 60 seconds

Validation

- Kernel counter validation: compared to "nc", "/proc/net/device" statistics
- User-space validation:
  - Fine-grained time bins within 1 second: 20, 196, 1784, 3257
  - iperf3 UDP streaming, 1400-byte payload, capped at 8 Gbps
  - 8.16 Gbps reported by our agent
  - 8.16 / 8 = 1428 (payload + L3&L4 headers) / 1400

Operating system
- Ubuntu
- Alma

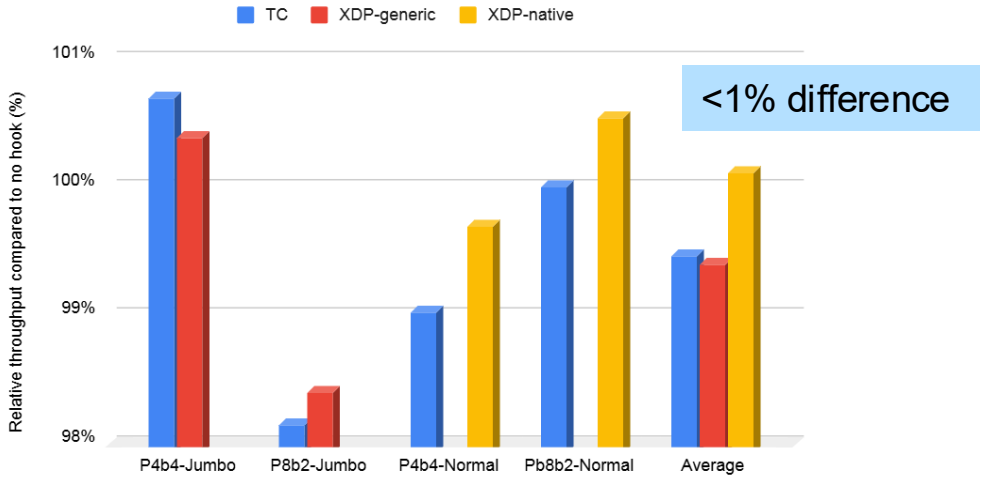
Server
- Arm
- x86

NIC
- BlueField-2 DPU
- ConnectX-6 Dx

MTU size
- 9 KB
- 1.5 KB

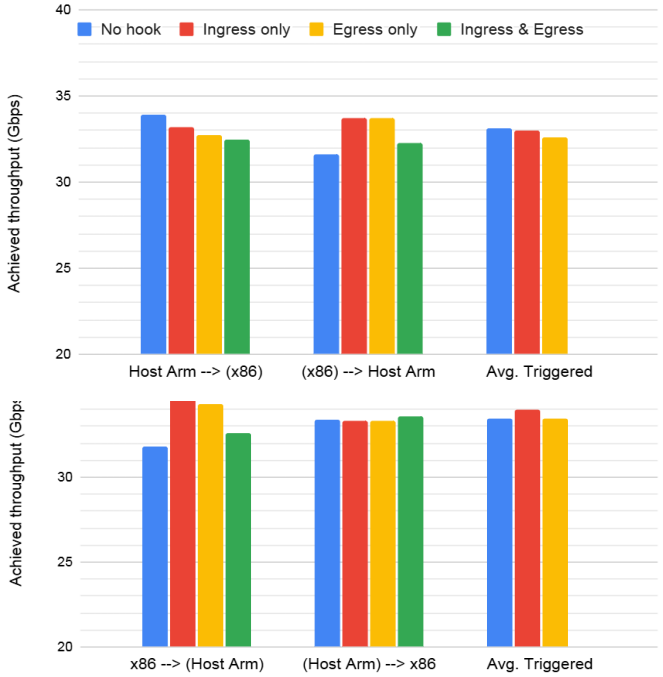
Minimal Throughput Impact

- Ingress eBPF hook types: XDP-offload, XDP-native (MTU <= 3498 B), XDP-generic, TC
- Egress eBPF hook type: TC



<1% difference

UDP unlimited rate, ingress hook only



TCP bi-directional, MTU=9 KB, same hook type on both ends

Low Resource Utilization

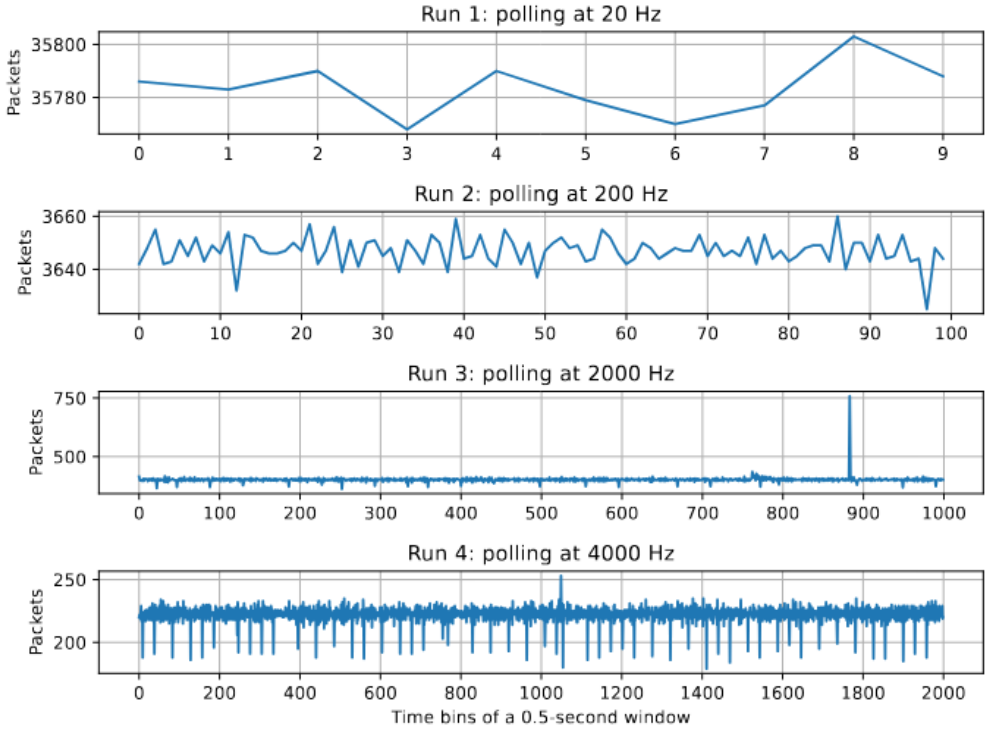


Table 4: Resource Utilization of User-Space Applications

Target polling frequency (Hz)	20	200	2000	4000
Max. CPU utilization (%)	1	2	3	4
Involved CPU cores	2	2	2	3
Per-second JSON report (KiB)	0.35	2.57	19.19	34.81
Max. disk write (KiB/s)	4	8	24	40
Max. cache miss (*minflt/s*)	6	57	72	179

iperf3 client "pidstat" monitoring

Ingress	Lower	Lower
Egress	Higher	Higher
Hook	Throughput	Kernel CPU %

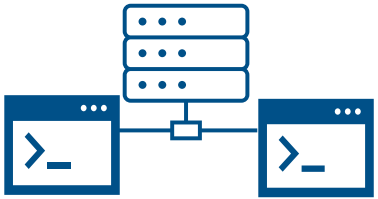
Stress Testing

UDP

```
[SUM]   0.00-10.00  sec   1   es   98.6 Gbits/sec   0.000 ms   0/13781620 (0%)   sender
[SUM]   0.00-10.00  sec   1   es   96.8 Gbits/sec   0.012 ms   257031/13781620 (1.9%)   receiver
```

TCP

```
[SUM]   0.00-10.00  sec   1   s   99.0 Gbits/sec   900             sender
[SUM]   0.00-10.00  sec   1   s   98.9 Gbits/sec                   receiver
```



- 100 Gbps NICs, single link
- Sender: egress TC hook
- Receiver: ingress TC hook
- iperf3 option –P16 –b0
- 5-run average

Fine-grained time tick (milliseconds)	100	10	1
TCP (Gbps)	99.0	99.0	98.9
UDP (Gbps) - Sender	96.0	96.1	97.0
UDP (Gbps) - Receiver	94.2	94.0	95.0

Scalability Analysis

- r: packet rate (PPS), limited by NIC
- l: number of active connections in a coarse-grained window
- f: number of fine-grained bins in a coarse-grained window

Operation		Time	Space
Kernel	Updating eBPF map	$O(r)$	$O(1)$
User-space	Polling eBPF map	$O(fl)$	$O(fl)$
	Exporting statistics	$O(fl)$	$O(fl)$

For l=1, we pushed to fl ~3,200

SciDAQ (e.g., EJFAT) system:
- High data rate from a single source
- Relatively consistent topology
- l should be within tens

Considering cache miss for larger l, we will target at fl ~ 1,000

For distributed deployment
- ✓ Use smaller coarse-grained window size, e.g., 1 second ==> 0.1 second
- ✓ Number of fine-grained bins: 20 - 50

Summary

- Lightweight millisecond-resolution L3 network monitoring;
- Nearly no influence on the throughput;
- Minimum influence on CPU & memory utilization;
- Independent of SciDAQ streaming applications;
- Works on all modern (kernel 5.x) Linux platforms;
- ...

- First bin spike because of the coarse-grained window switch;
- Does not work for kernel-bypassing traffic or InfiniBand => DPU
- "sudo" permission required
- Harder to deploy than Prometheus exporters
- ...

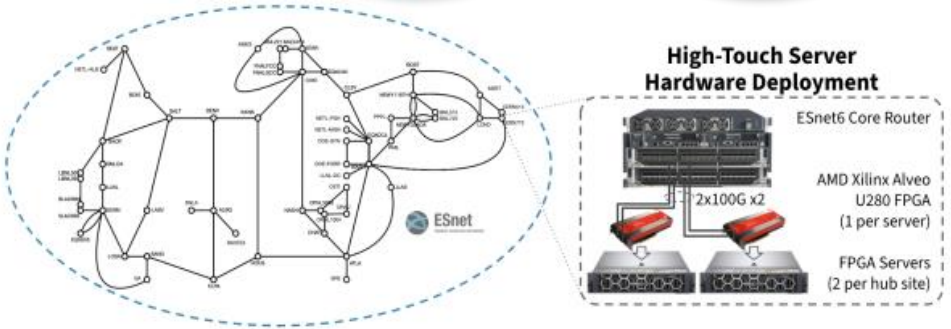
Envision

High-Touch+
- Last-mile visibility

eCenter
- A networking digital twin

AI+
- Anomaly detection
- ...

SciDAQ
- Autonomous experiment



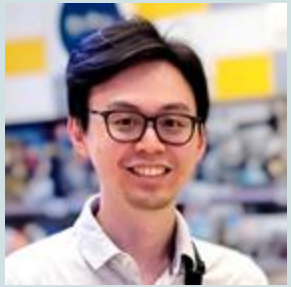
High-Touch: https://www.es.net/network-r-and-d/esnets-high-touch-project/

Acknowledgement

The research described in this presentation was conducted under the Laboratory Directed Research and Development Program at Thomas Jefferson National Accelerator Facility for the U.S. Department of Energy.

FY25 / first-year contributors

Dr. Jie Chen



Dr. Jeng-Yuan Tsai

Raiqa Rasool

Forthcoming DOE booth demonstration

https://scdoe.info/demonstrations/

1:00 p.m. CST

WEDNESDAY, NOV. 19

Station 1

Team:
Xinxin (Cissie) Mei

Catch Every Packet: Millisecond Network Monitoring on Linux

Associated Organizations: Jefferson Lab

» Abstract

Experience millisecond-resolution network monitoring in action! This demonstration shows how Linux systems can capture and report fine-grained telemetry in real time, enabling HPC and AI workloads to achieve unprecedented visibility into network performance.