Schooling NOOBs with eBPF

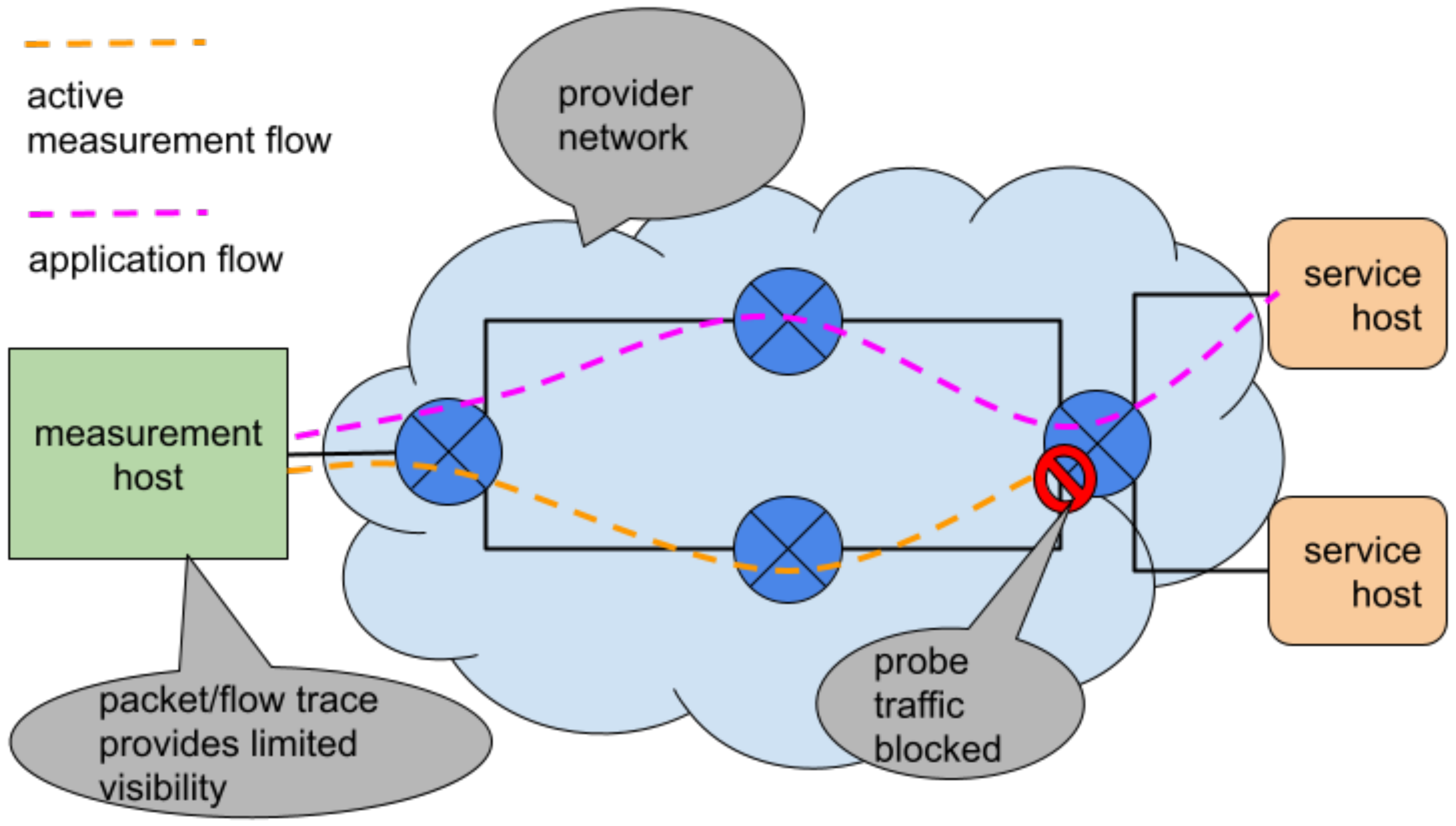
Joel Sommers
Colgate University

Nolan Rudolph
University of Oregon

Ramakrishnan Durairajan
University of Oregon

Motivation

- Typical active and passive measurements can provide significant insight into network performance and traffic behavior
    - Ping, traceroute, packet/flow capture
- But they have many shortcomings
    - Passive measurements have limited visibility
    - Performance observed by typical active measurement can be misleading due to load balancing
    - Typical measurement probes are subject to blocking and rate limiting
- Situation has led to **NOOB** (network oblivious) applications and end hosts



jsommers@colgate.edu

Goal

eBPF

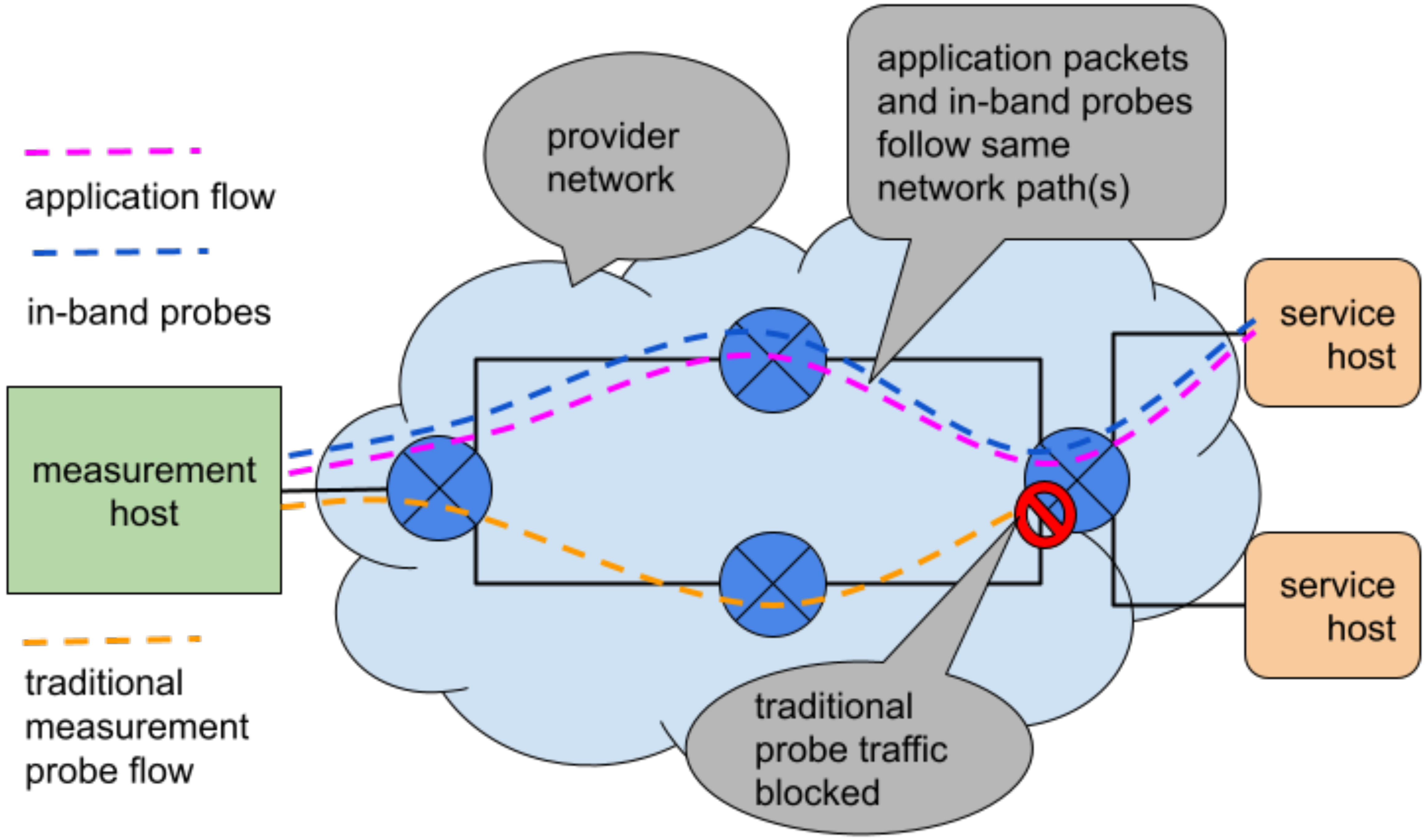
https://ebpf.io

- Explore use of eBPF to provide fine-grained active and passive telemetry to address the NOOB problem
- Why eBPF?
  - Low-overhead and portable in-band active measurement (tc/cls-bpf + XDP)
  - Efficient passive measurement (XDP)
  - Plus all the "usual" benefits of eBPF: Safe in-kernel execution, no kernel/user boundary crossings (*cf.* libpcap), no need to modify applications

jsommers@colgate.edu

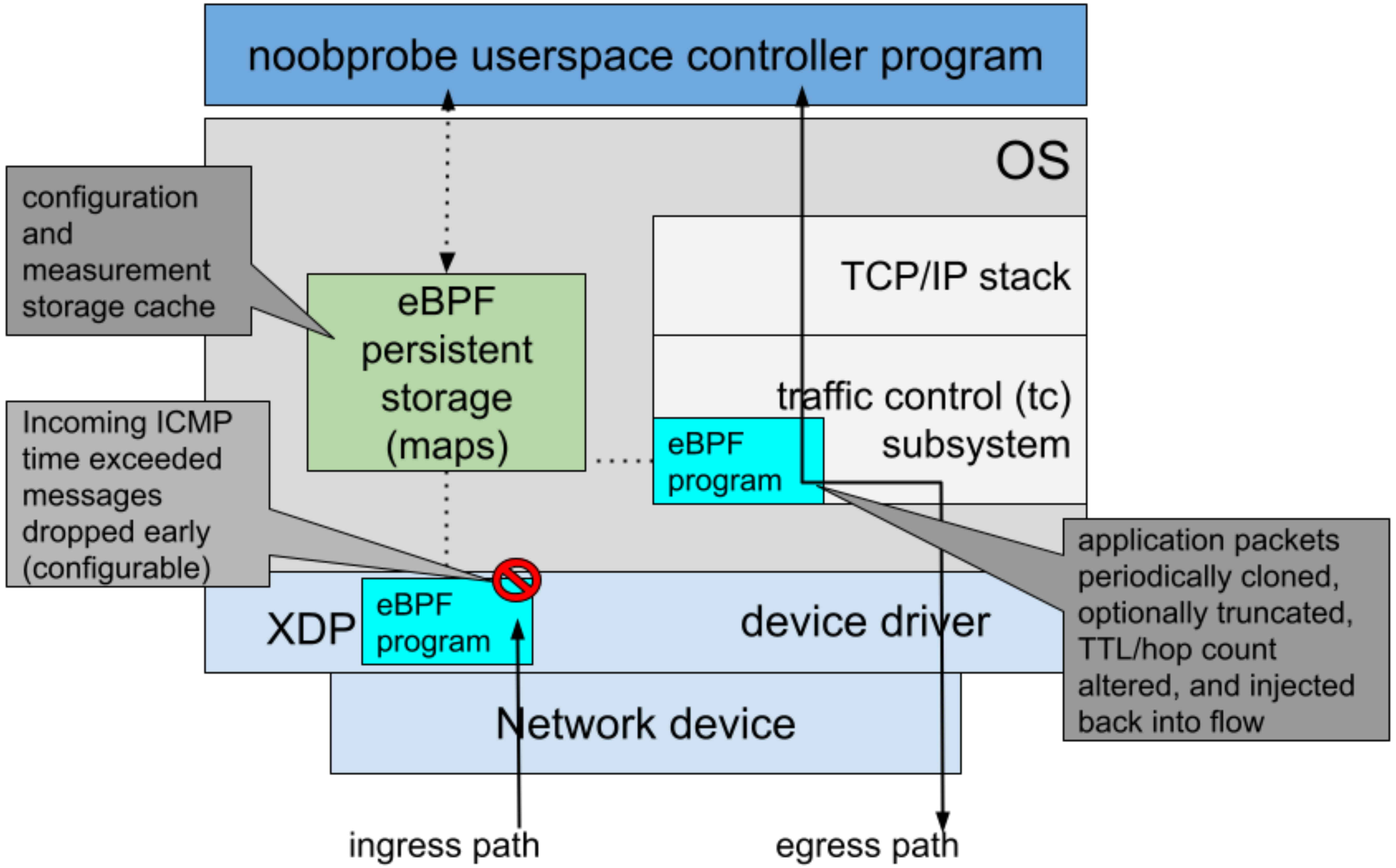
noobprobe: In-band active measurement

- In-band measurement: probes share same IP and transport layer information (e.g., 5-tuple)
  - Hash-based load balancing causes probes to follow same path as application flow
  - In-band probes are subject to same blocking policy as application traffic
  - Use of eBPF offers a significant performance improvement over libpcap (Sommers and Durairajan, TMA 2022)



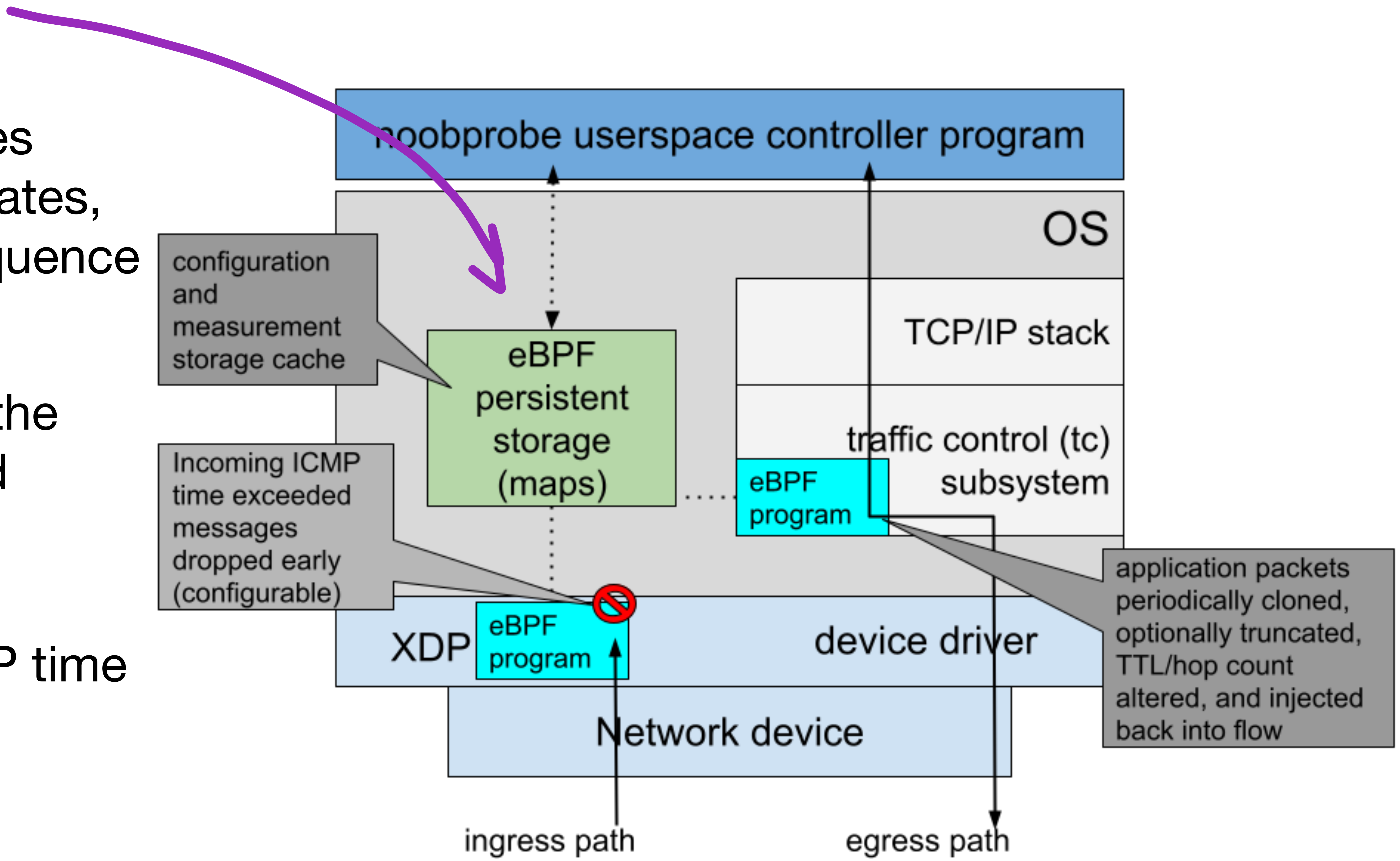
noobprobe overview

- User specifies destinations of interest (or application/process of interest)
- tc/cls-bpf program periodically clones application packets, optionally truncates, reduces TTL/hop count, writes a sequence number, injects probe into app flow
- Probe TTL/hop count expires along the path, triggering ICMP time exceeded message
- Ingress XDP program: inspects ICMP time exceeded message, matches with outgoing probe, and drops prior to entering standard network stack processing



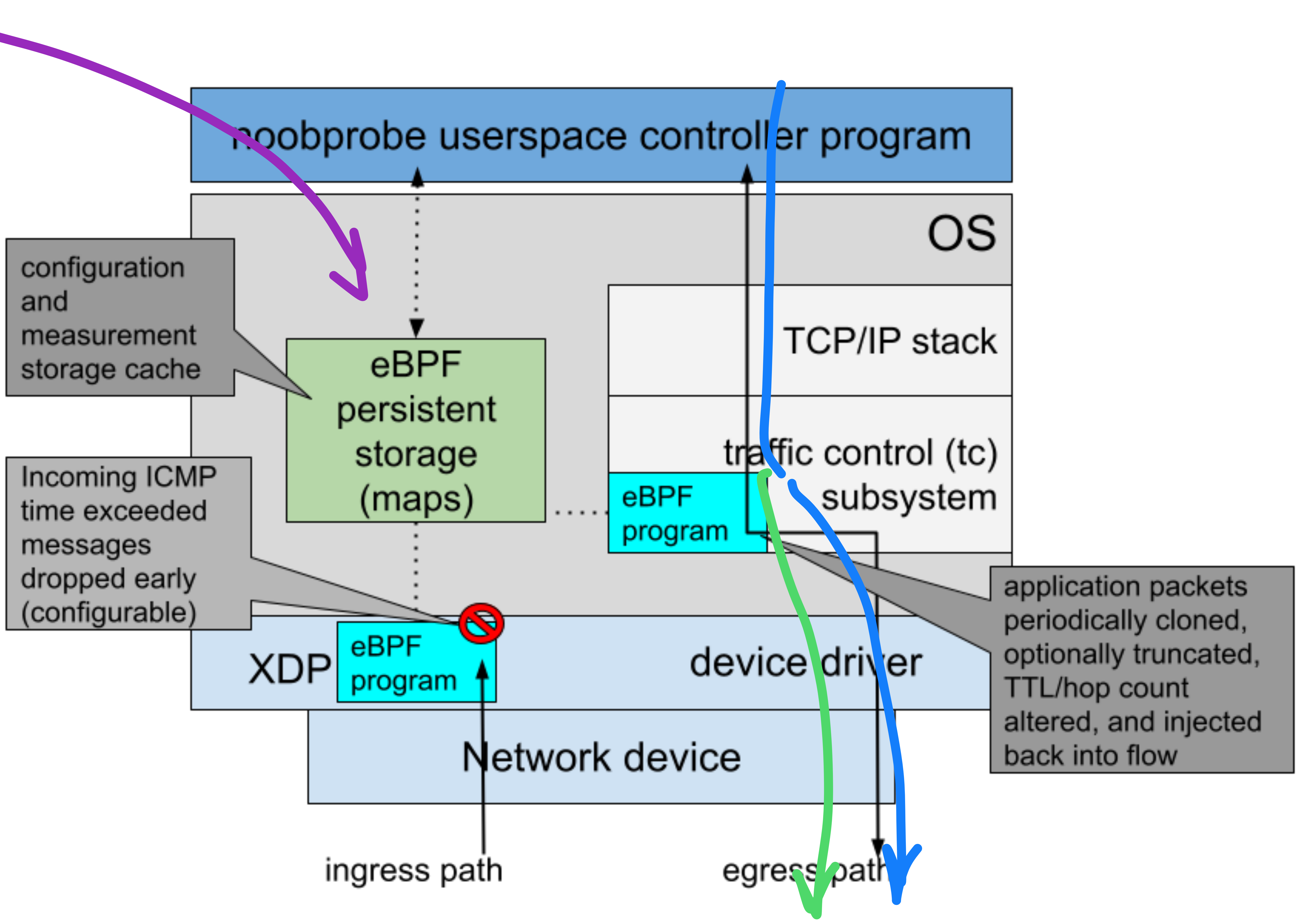
noobprobe overview

- User specifies destinations of interest (or application/process of interest)
- tc/cls-bpf program periodically clones application packets, optionally truncates, reduces TTL/hop count, writes a sequence number, injects probe into app flow
- Probe TTL/hop count expires along the path, triggering ICMP time exceeded message
- Ingress XDP program: inspects ICMP time exceeded message, matches with outgoing probe, and drops prior to entering standard network stack processing



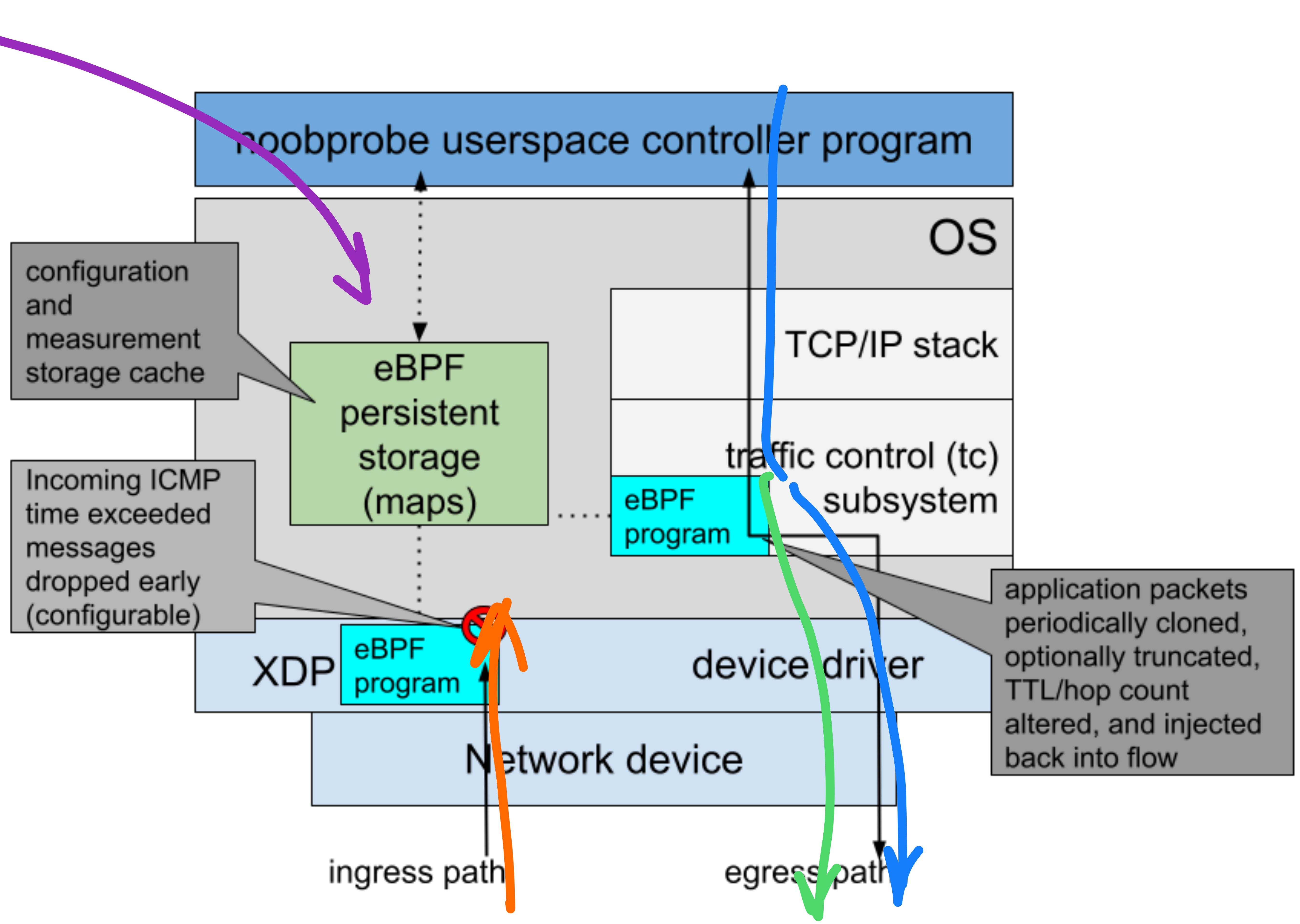
noobprobe overview

- User specifies destinations of interest (or application/process of interest)
- tc/cls-bpf program periodically clones application packets, optionally truncates, reduces TTL/hop count, writes a sequence number, injects probe into app flow
- Probe TTL/hop count expires along the path, triggering ICMP time exceeded message
- Ingress XDP program: inspects ICMP time exceeded message, matches with outgoing probe, and drops prior to entering standard network stack processing



noobprobe overview

- User specifies destinations of interest (or application/process of interest)
- tc/cls-bpf program periodically clones application packets, optionally truncates, reduces TTL/hop count, writes a sequence number, injects probe into app flow
- Probe TTL/hop count expires along the path, triggering ICMP time exceeded message
- Ingress XDP program: inspects ICMP time exceeded message, matches with outgoing probe, and drops prior to entering standard network stack processing



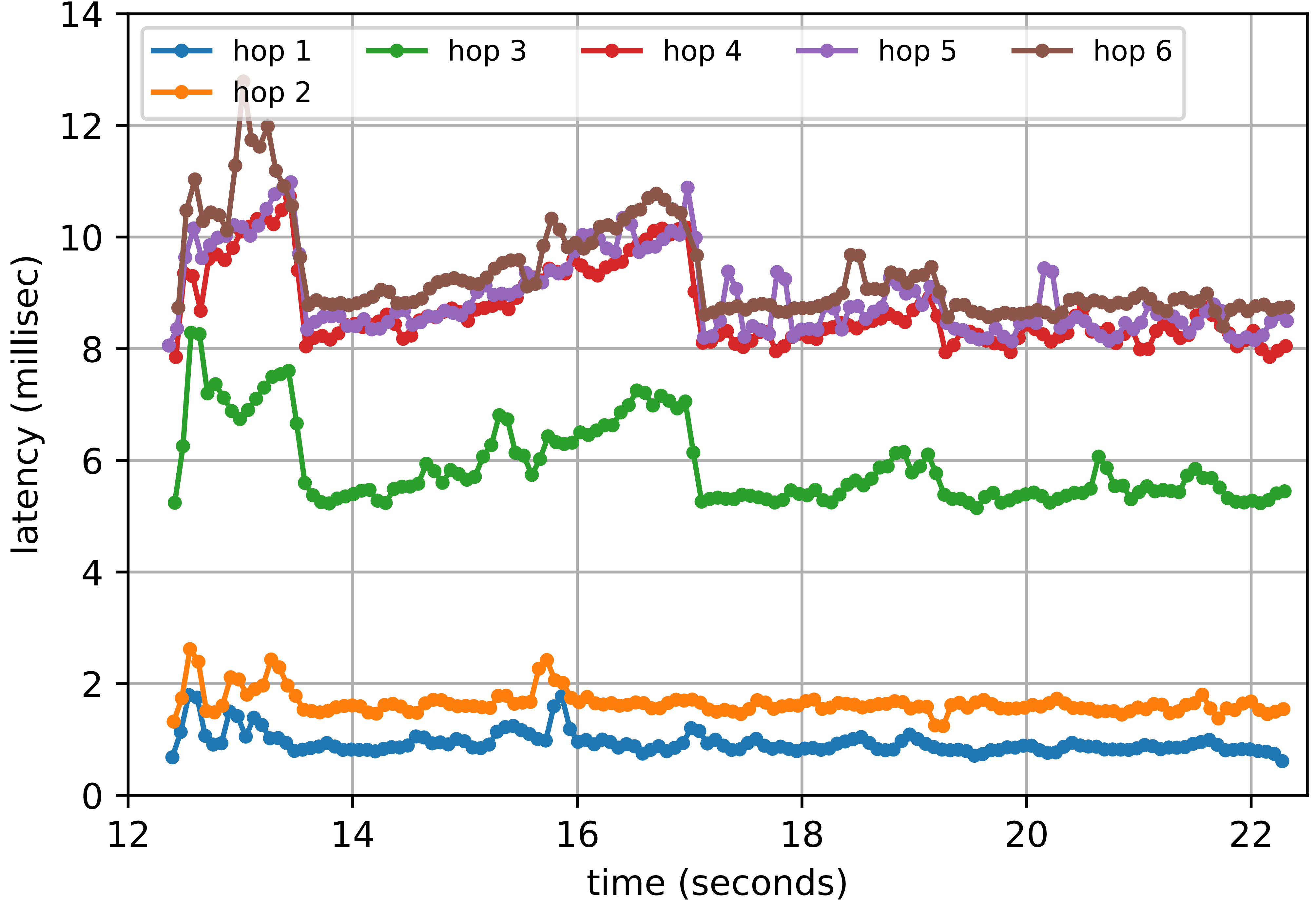
noobprobe implementation details

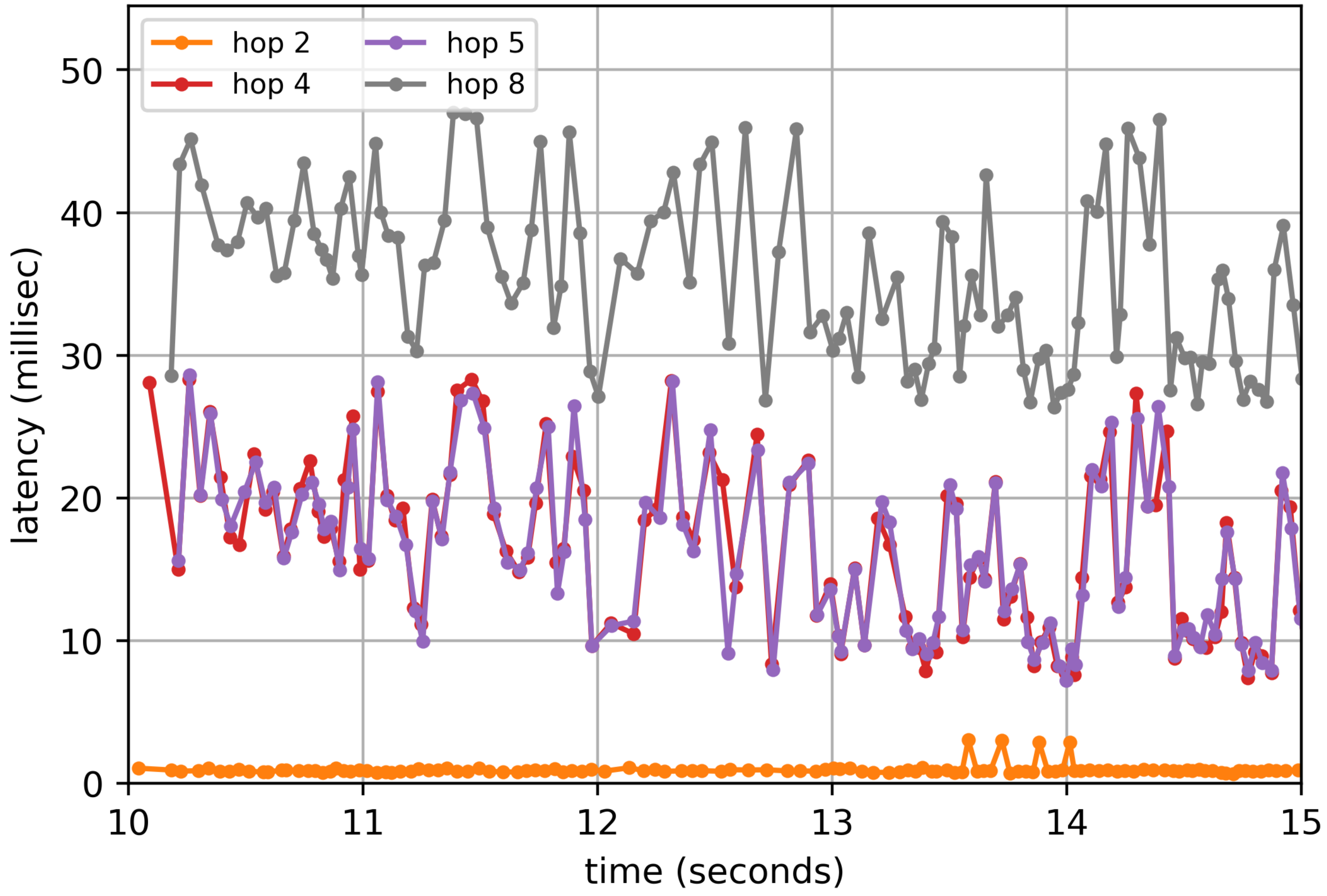
- Implemented using the BPF Compiler Collection (bcc), a library to simplify aspects of eBPF programming
  - eBPF program at Linux tc hook performs probe creation, program at XDP hook for probe reception
  - Code structure is modularized using BPF program jump tables
  - User can write their own code, invoked before probes send and/or after receive
- Python management program runs until stopped
  - Options for maximum probe rate, whether to truncate probes, destinations or app of interest
  - Measurements stored in a CSV file as they are copied from kernel BPF map

https://github.com/iovisor/bcc

Wide-area experiments (1)

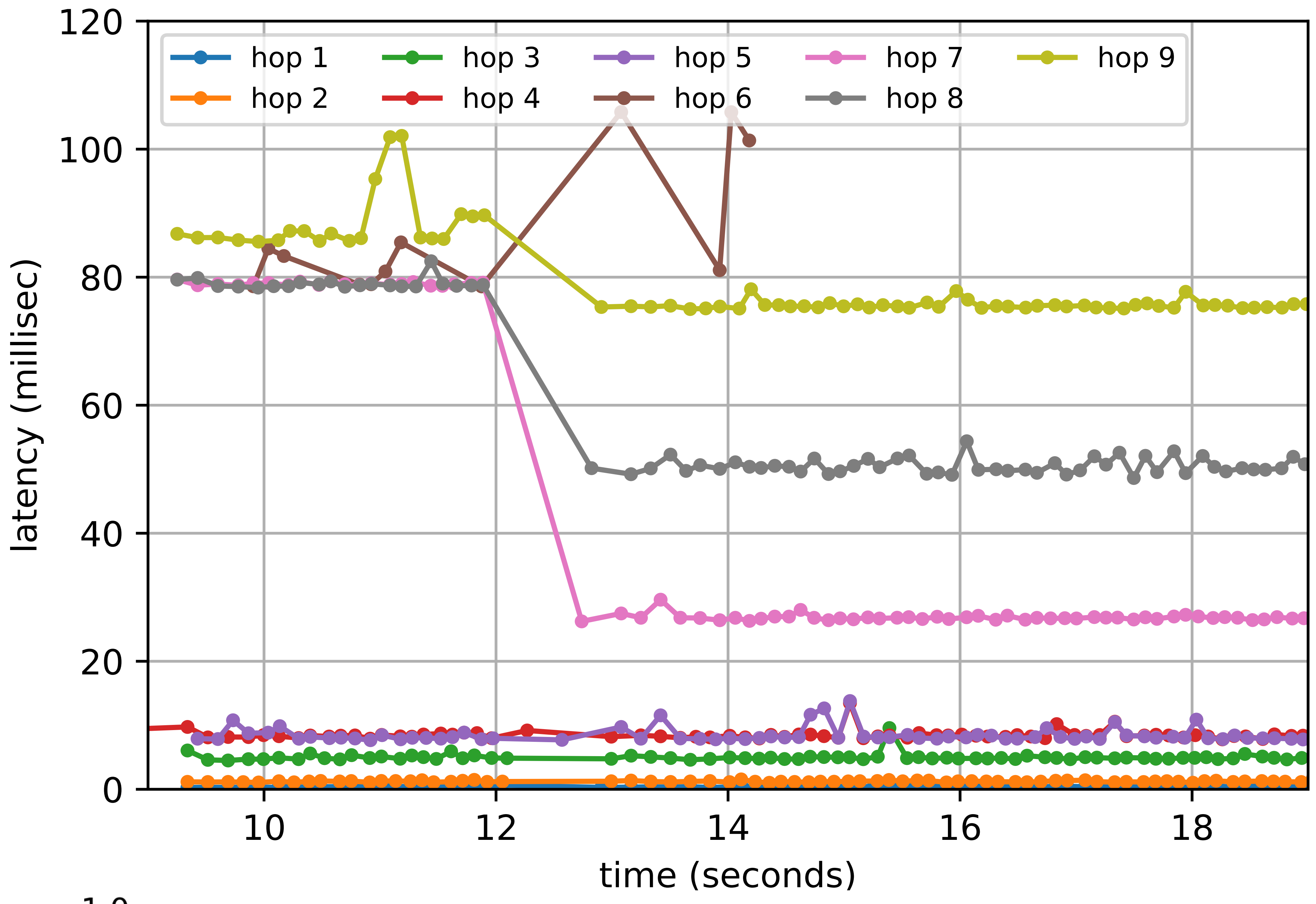
- Instrumented hourly "speedtest-style" flows for one week, from 4 Cloudlab locations and 1 university location
  - NDT throughput tests with 12 M-Lab locations around the world
  - Netflix's fast.com throughput test
- Found that ~90% of all routers respond to in-band hop-limited probes without apparent throttling
  - We used a 100 probes/sec maximum rate
- High-resolution queuing delay plots emerge
  - Top plot is NDT flow between university site and NDT LGA server
  - Bottom plot is fast.com test from the university site

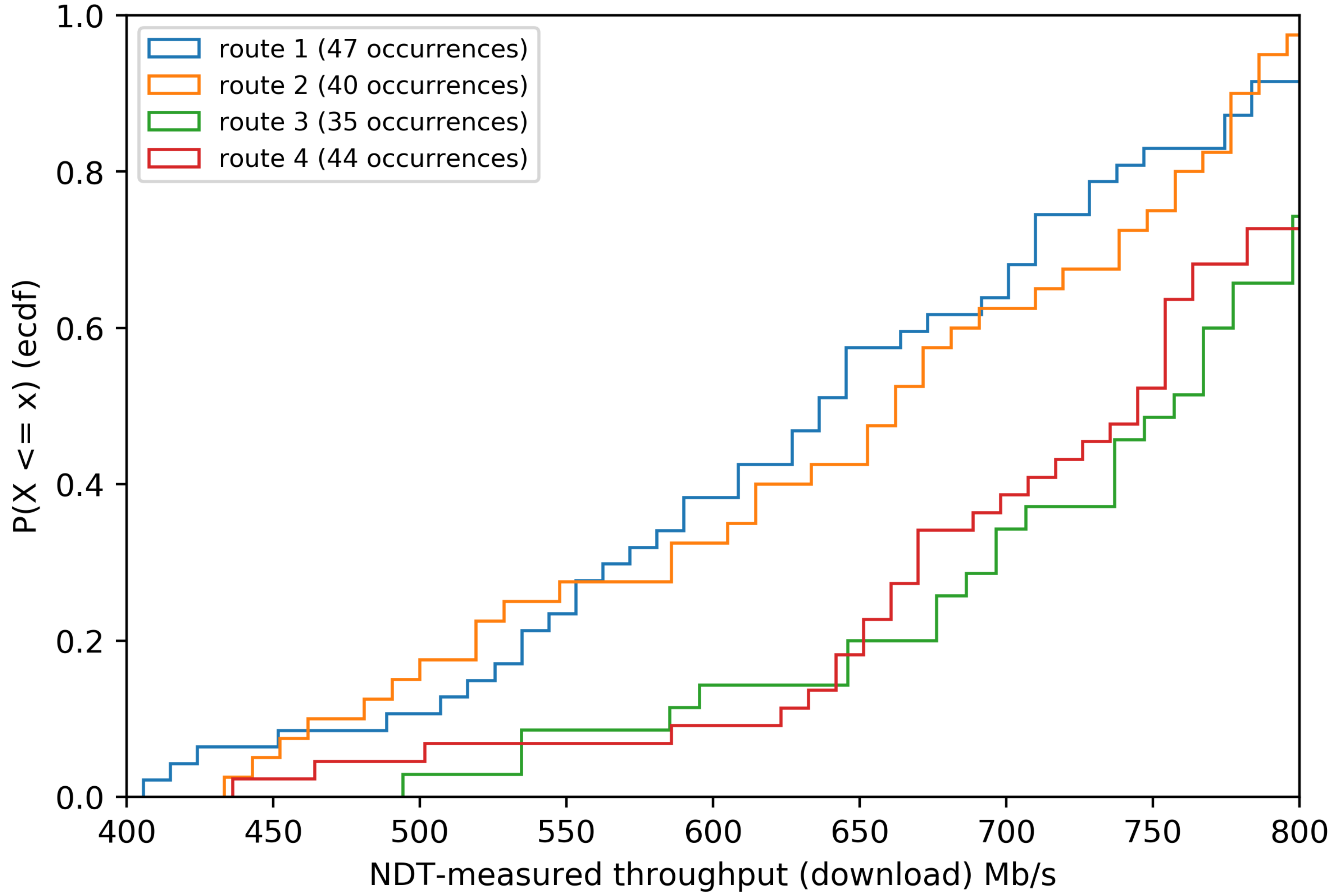




Wide-area experiments (2)

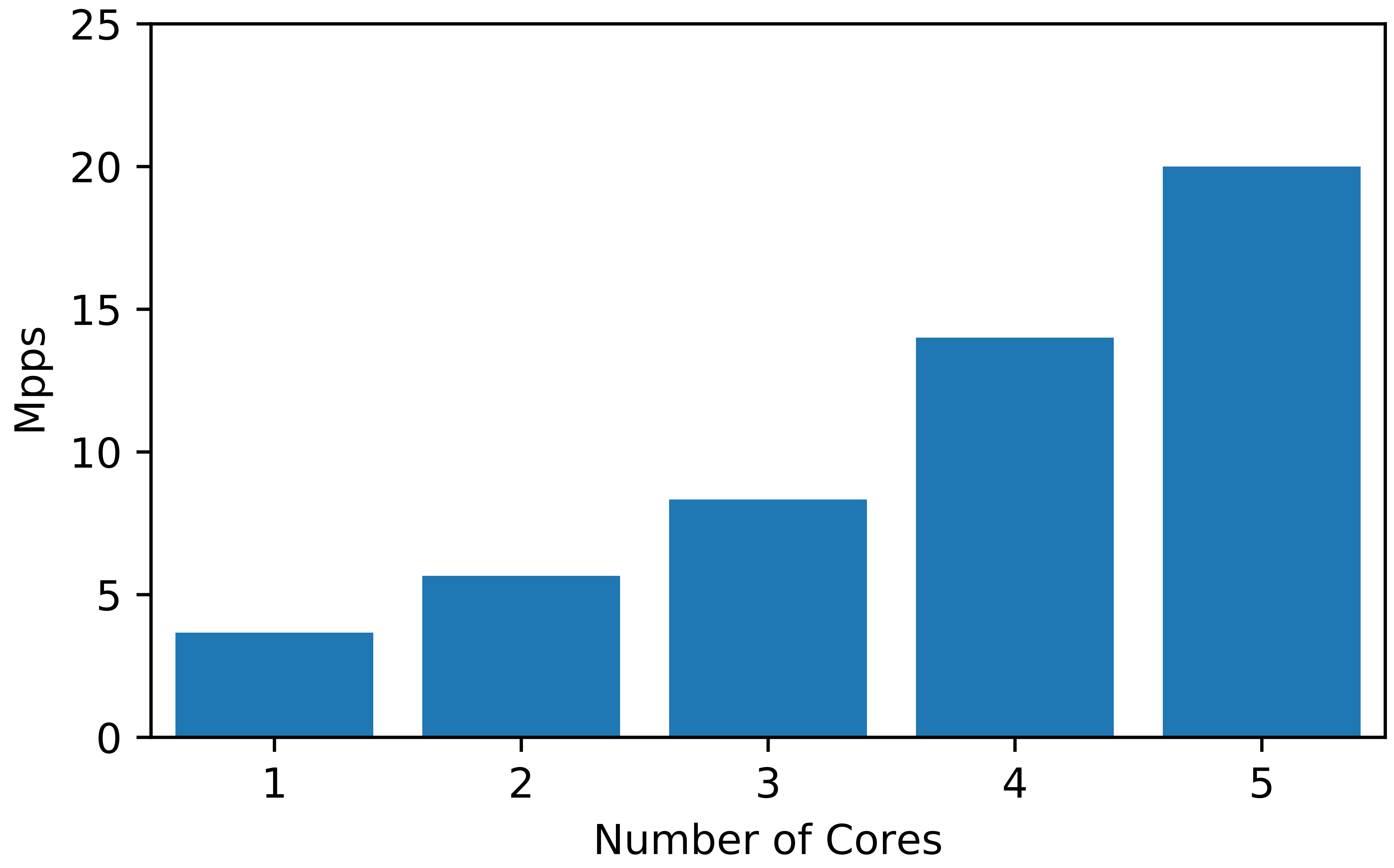
- Route changes and degraded throughput (top plot)
    - NDT client between university site and M-Lab server in Vancouver, Canada
    - 9 interdomain route changes observed in our week-long data collection
- (Significant) unequal throughput from load balanced paths (bottom plot)
    - Example is from data collected between Clemson Cloudlab site and Dallas-Fort Worth M-Lab site
    - Many more examples of statistically significant performance disparity on load-balanced paths





noobflow: passive flow capture

- Passive flow measurements can provide rich, fine-grained detail on network activity
  - Collect at the edge, or in the cloud
- XDP component, written using bcc
  - Two per-CPU maps (double buffering) with atomic swap for lock-free flow collection
- Experiments in CloudLab using hosts with 25 Gb/s interfaces
  - Generate traffic 60 byte UDP packets with pktgen, from 1 Mpps to 20 Mpps
  - Plot shows maximum offered packet rate sustainable without loss



jsommers@colgate.edu

Summary

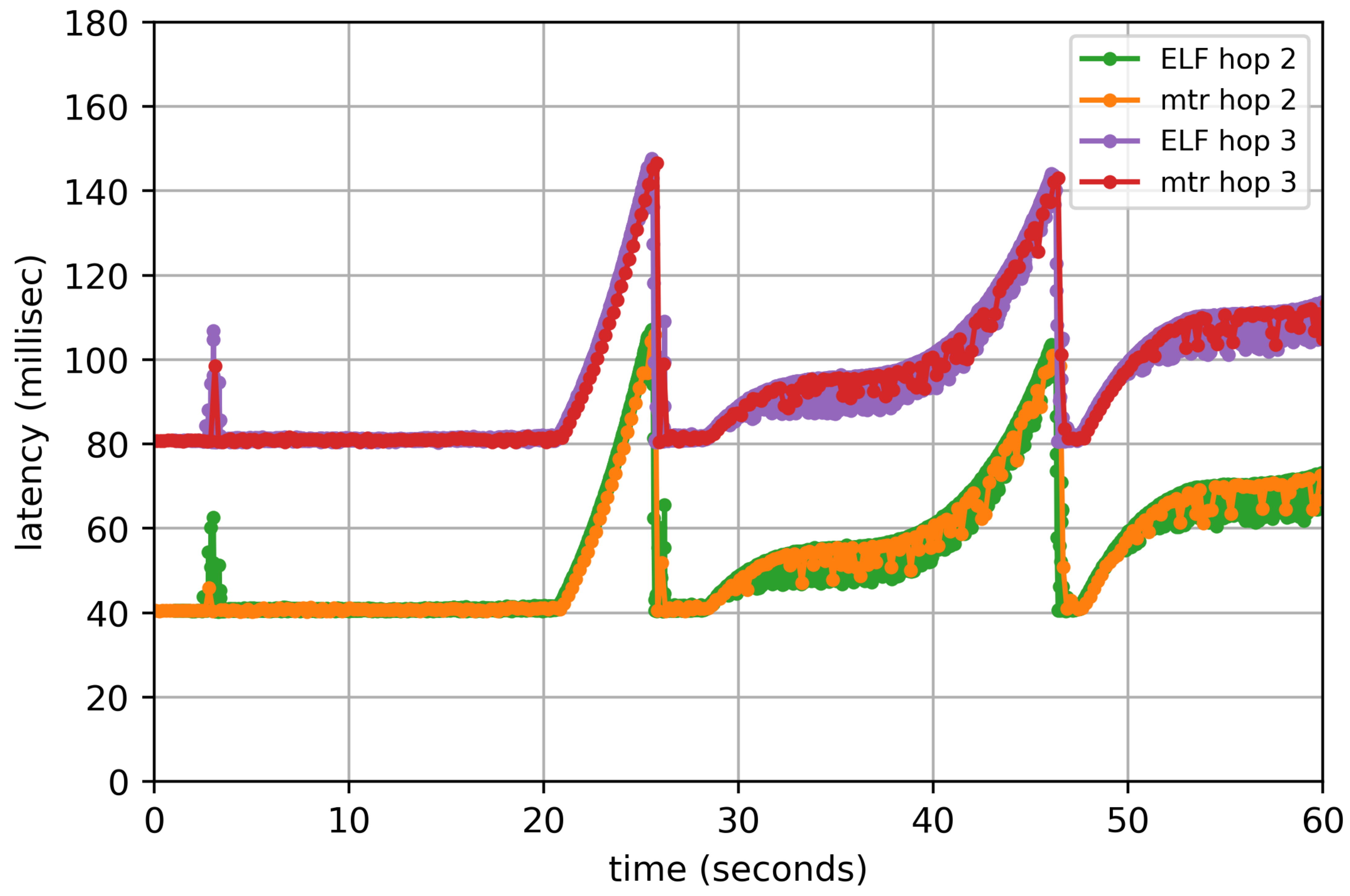
- The NOOB problem is a persistent challenge
  - eBPF offers a compelling implementation platform for network telemetry to address NOOBs which we explored with noobprobe/noobflow
- Future work
  - Investigate perf buffers for delivering telemetry to userspace
    - We used an older version of bcc which only supported fixed-size buffers
  - Investigate bringing better network awareness to applications
  - Better understand the nature of noise in latency measurements derived from ICMP time exceeded responses
- Code is available: https://github.com/jsommers/noob

Lab experiments: libpcap vs eBPF

- Goal: understand performance differences between libpcap- vs. ebpf-based in-band measurement
- Simple linear topology with three Linux hosts (A-B-C)
  - Packets emitted with Linux pktgen at A, 2kpps up to 512kpps offered loads
  - libpcap or ELF at B, cloning every 100th packet
  - Original packet and clone received at C
- At low rate (32 kpps and above), packet loss and high variability for libpcap
  - Negative spacing: some probes arrive before original packet — only with libpcap

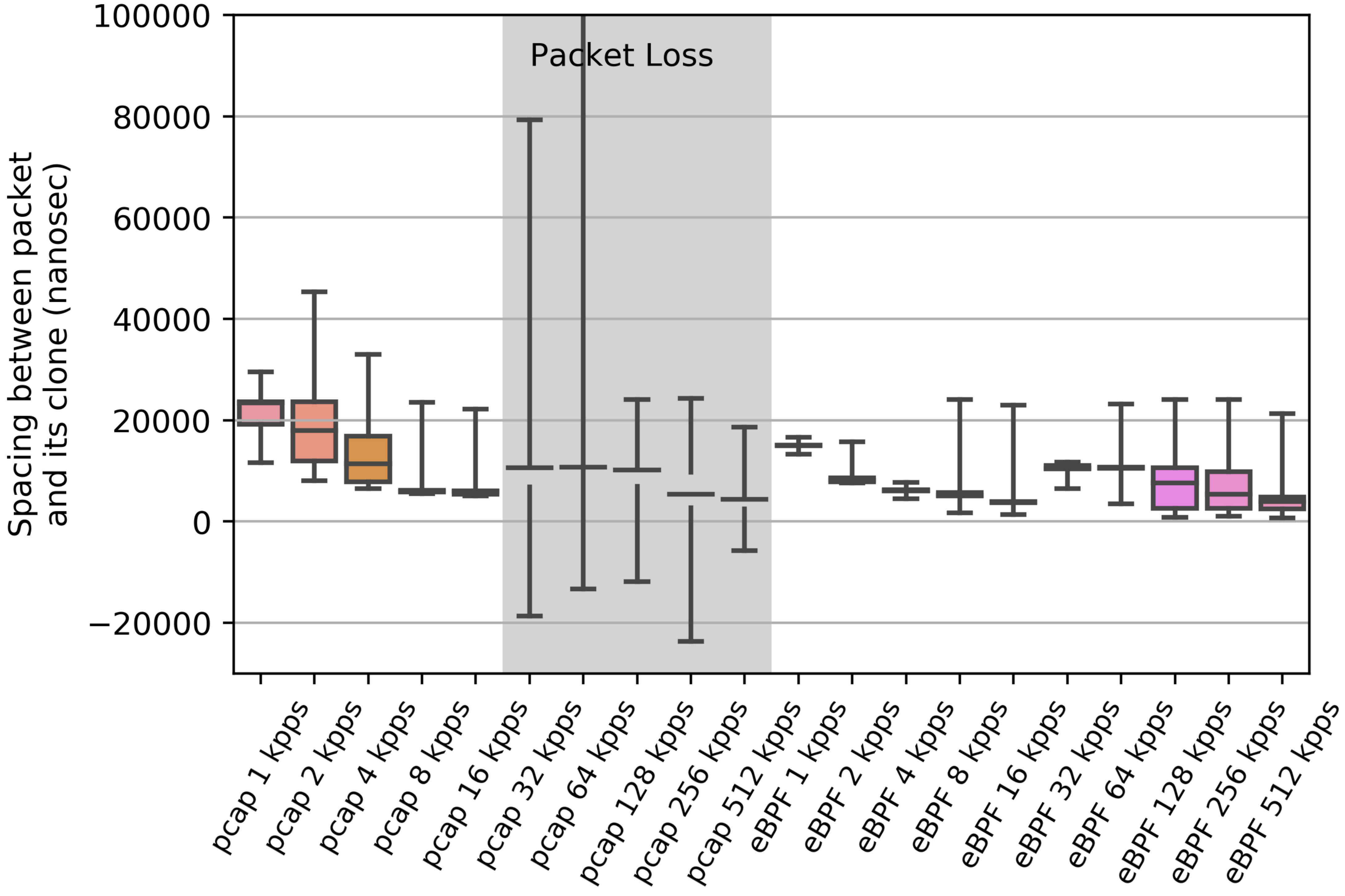
Lab experiments: queuing delays

- Linear topology of 5 Linux hosts
  - TCP traffic generated using iperf3
  - Experiments with cross traffic at different hops
  - 20 millisecond one-way delays imposed at two different hops, using Linux tc
- Figure shows ELF and mtr-measured delays at the 2nd and 3rd hops, no cross traffic
- Probe rate from ELF is a miniscule 32 kbit/sec, yet a detailed profile of queuing delay emerges
- Congestion primarily and clearly occurs at hop 2



Lab experiments: libpcap vs eBPF

- Goal: understand performance differences between libpcap- vs. ebpf-based in-band measurement
- Simple linear topology with three Linux hosts (A-B-C)
  - Packets emitted with Linux pktgen at A
  - libpcap or ELF at B, cloning every 100th packet
  - Original packet and clone received at C
- At low rate, packet loss and high variability for libpcap
  - Negative spacing: some probes arrive *before* original packet — only with libpcap



 jsommers@colgate.edu