Can We Containerize Internet Measurements?

Chris Misa (University of Oregon)

Sudarsun Kannan (Rutgers University)

Ramakrishnan Durairajan (University of Oregon)

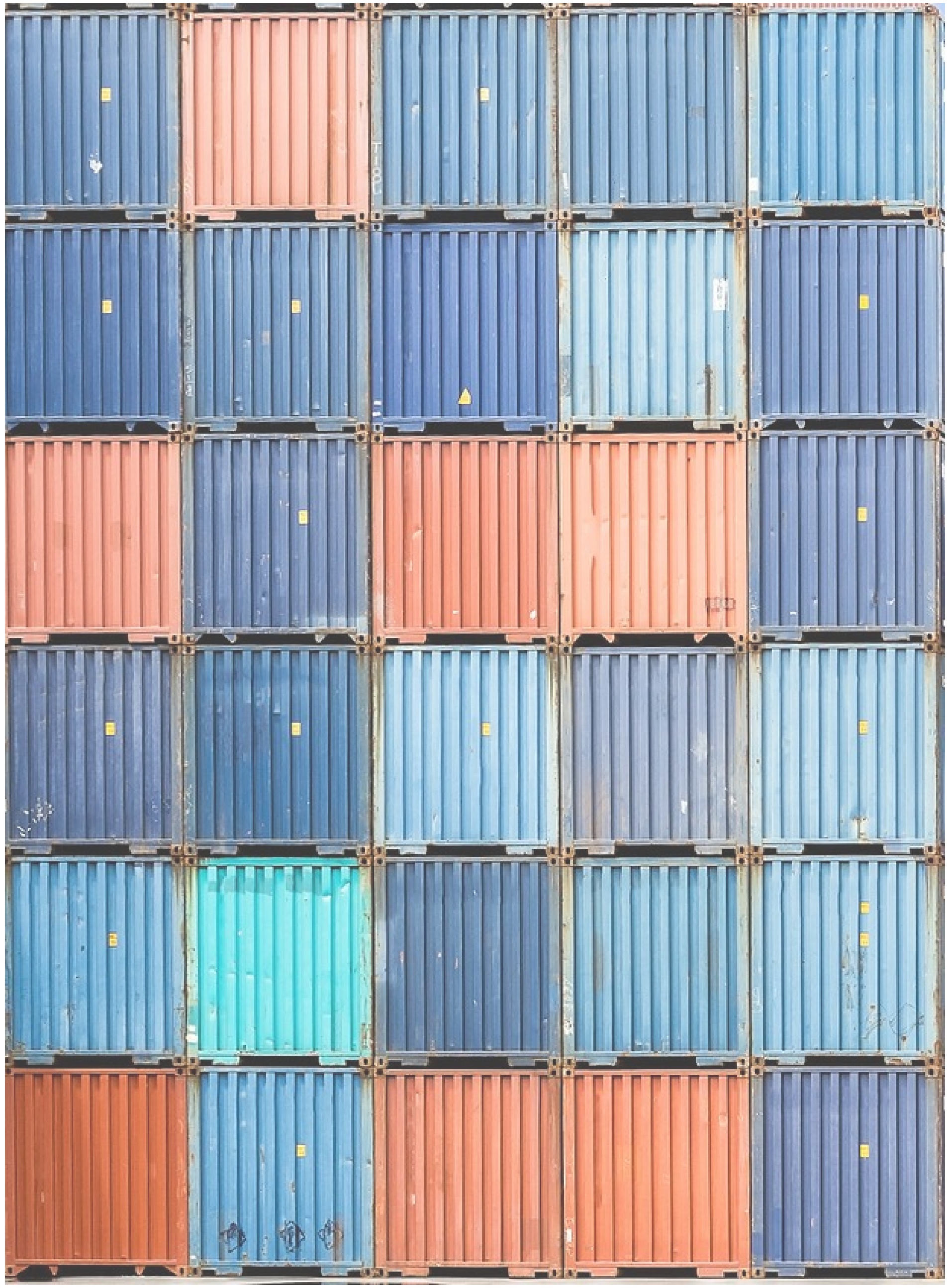
cmisa@cs.uoregon.edu

Outline

- **Containerized measurement issues**
- Proposed solution: MACE
- Evaluation of MACE

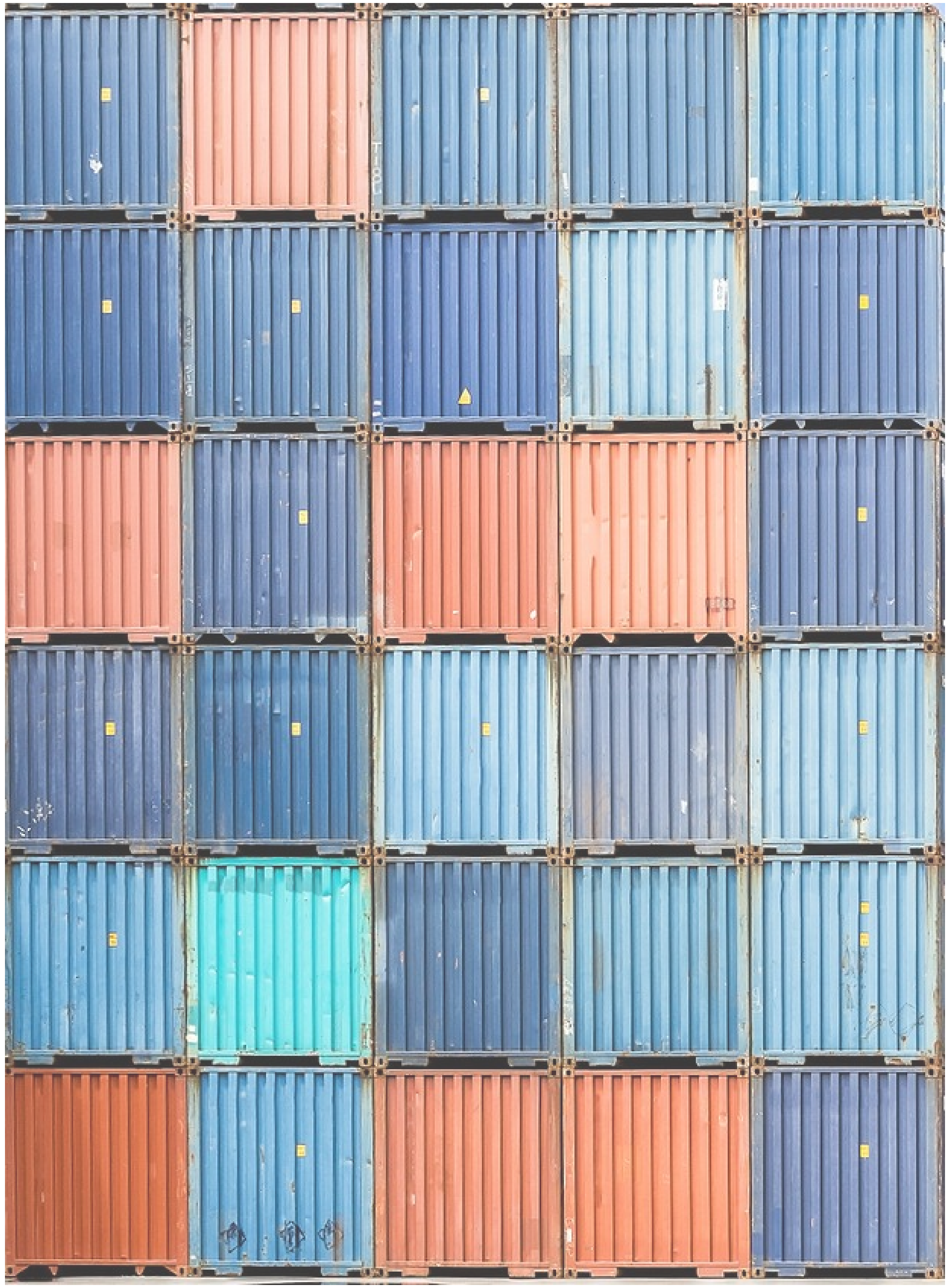
Containers

- Lightweight virtualization mechanism
  - Package, deploy, isolate



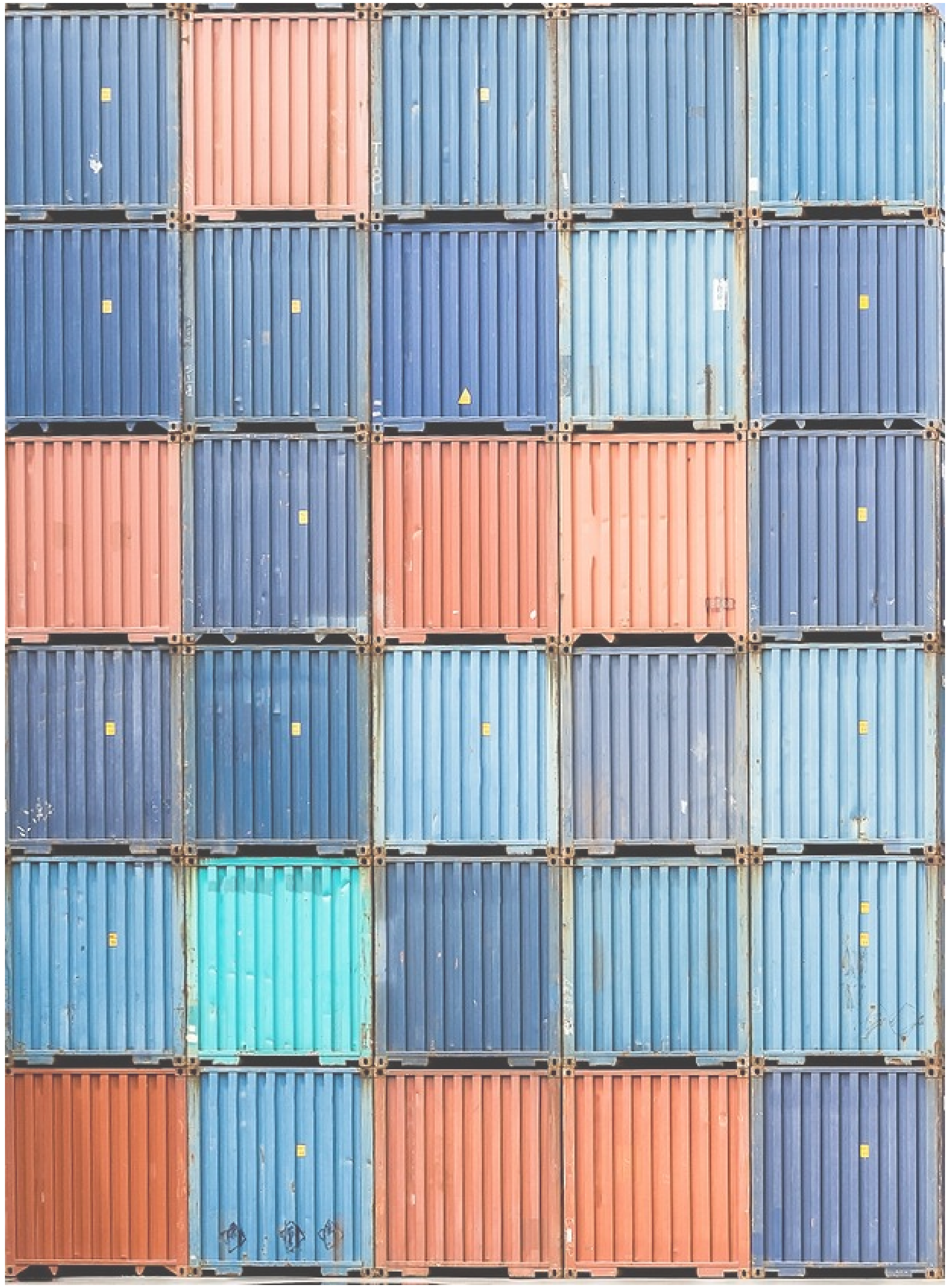
Containers

- Lightweight virtualization mechanism
  - Package, deploy, isolate
- Based on recent developments in Linux
  - Namespaces, cgroups



Containers

- Lightweight virtualization mechanism
  - Package, deploy, isolate
- Based on recent developments in Linux
  - Namespaces, cgroups
- Rapidly replacing VMs
  - Smaller, faster



Motivation

- Streamline experiments
  - Package scripts, tools, libraries
  - Consistent interface

Motivation

- Streamline experiments
  - Package scripts, tools, libraries
  - Consistent interface
- Expose new, cloud-native vantage points
  - Azure
  - AWS
  - GCP
  - etc.

Motivation

- Streamline experiments
  - Package scripts, tools, libraries
  - Consistent interface
- Expose new, cloud-native vantage points
  - Azure
  - AWS
  - GCP
  - etc.
- Less CPU and memory overheads than VMs [1]

PlanetLab since 2012 [0]

cmisa@cs.uoregon.edu

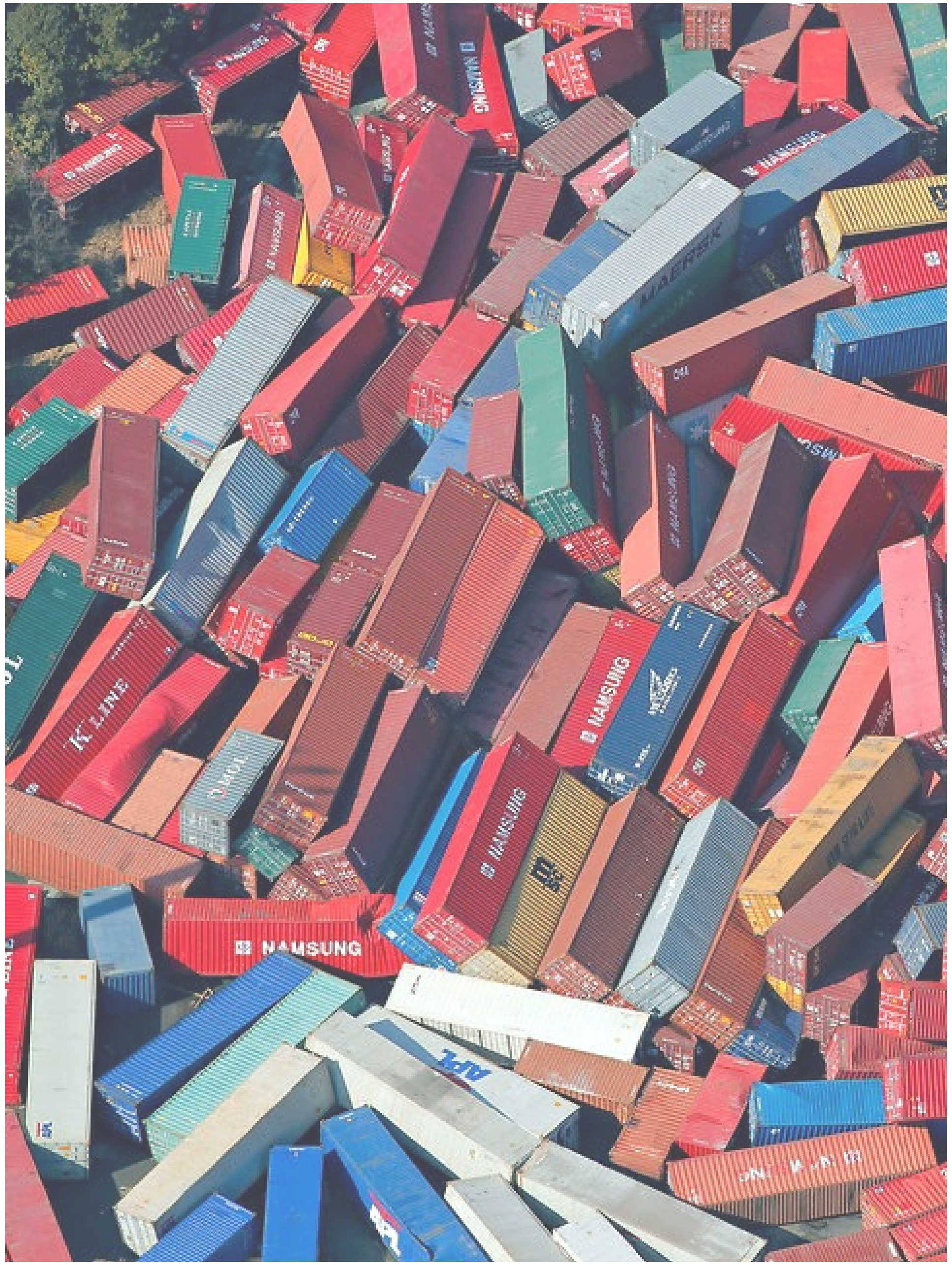
Sure we can!

Sure we can!

Why not?

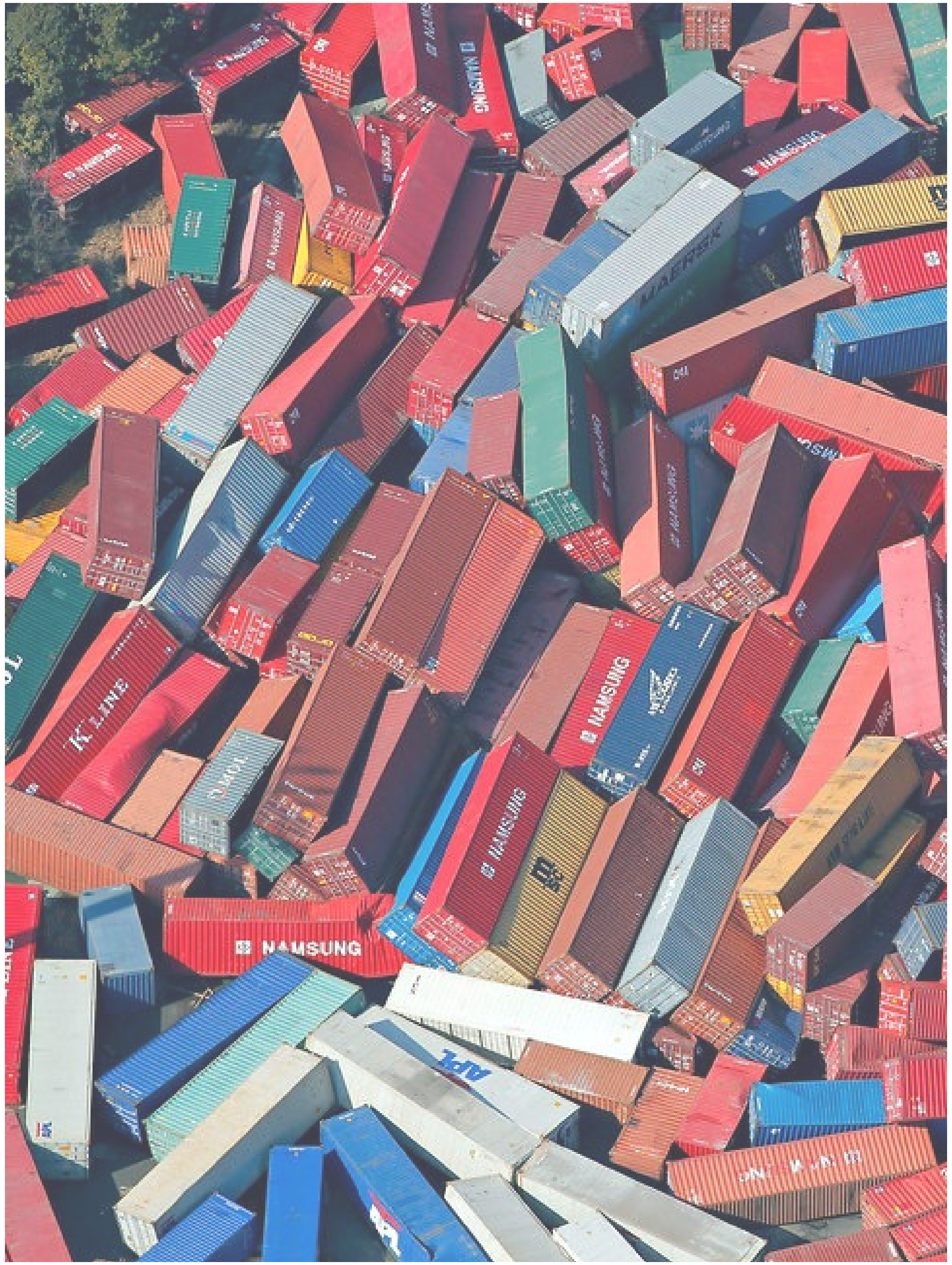
Network Isolation

- Extra latency [2]
  - ~50µs in resting system



Network Isolation

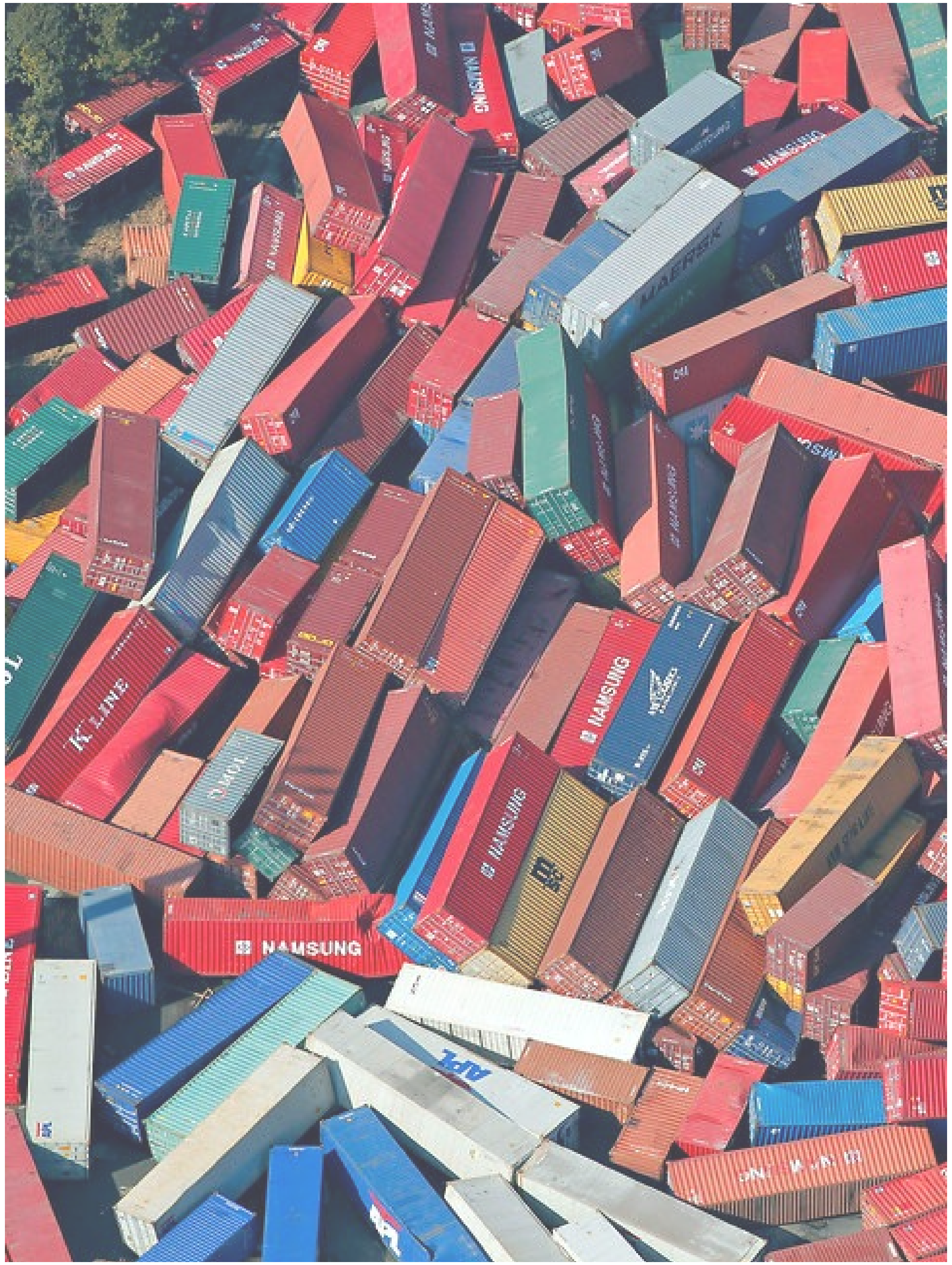
- Extra latency [2]
  - ~50µs in resting system
- Co-located containers
  - Up to 300µs depending on traffic



cmisa@cs.uoregon.edu

Network Isolation

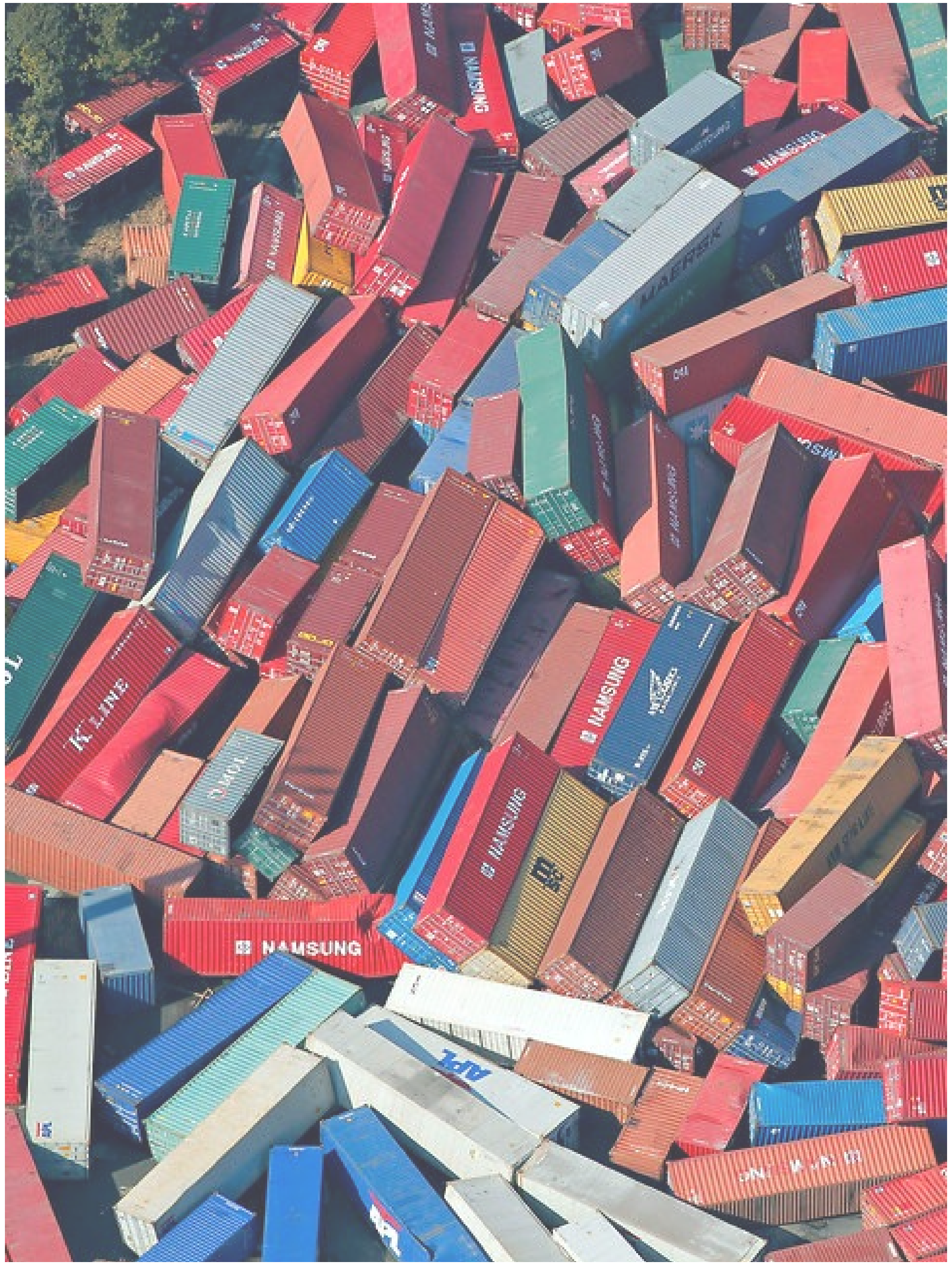
- Extra latency [2]
  - ~50µs in resting system
- Co-located containers
  - Up to 300µs depending on traffic
- Biased measurement results
  - Non-constant latency overheads



cmisa@cs.uoregon.edu

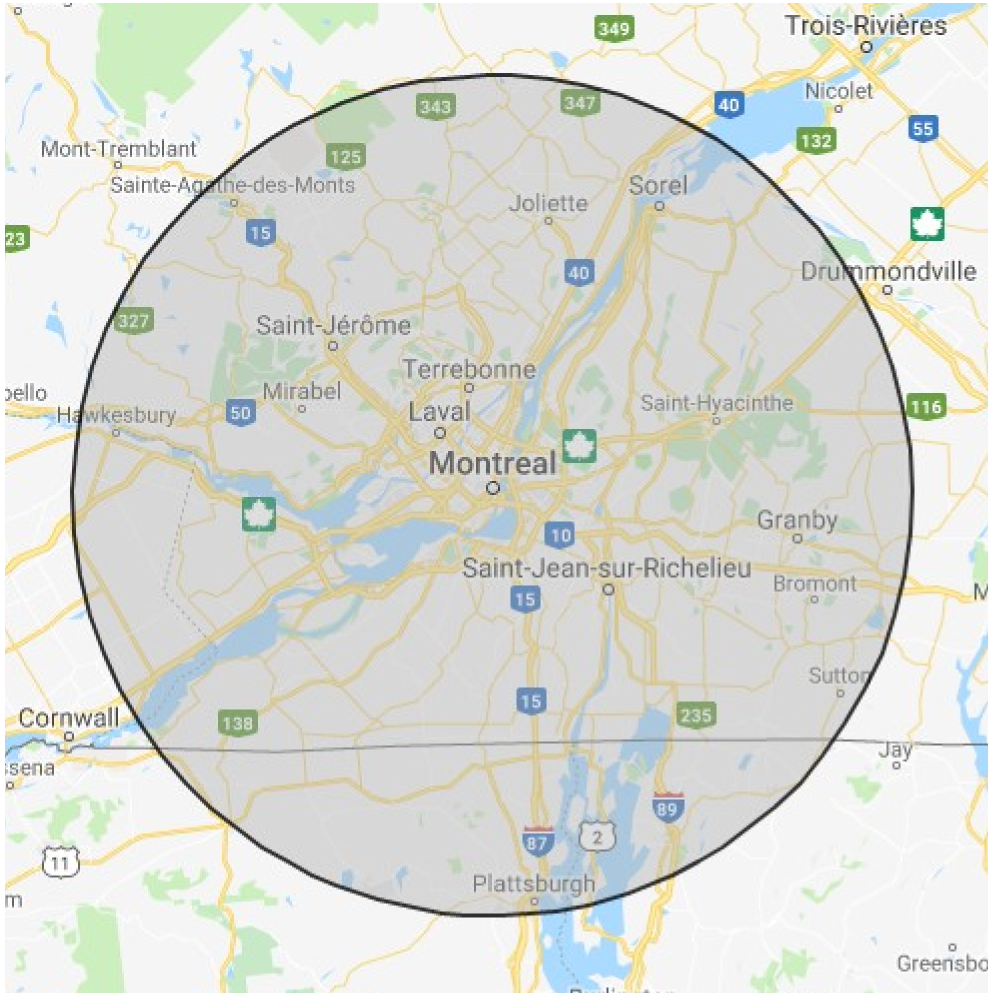
Network Isolation

- Extra latency [2]
  - ~50μs in resting system
- Co-located containers
  - Up to 300μs depending on traffic
- Biased measurement results
  - Non-constant latency overheads
- Slim [3], FreeFlow [4] don't help
  - Flow-based, RDMA



Importance of Latency

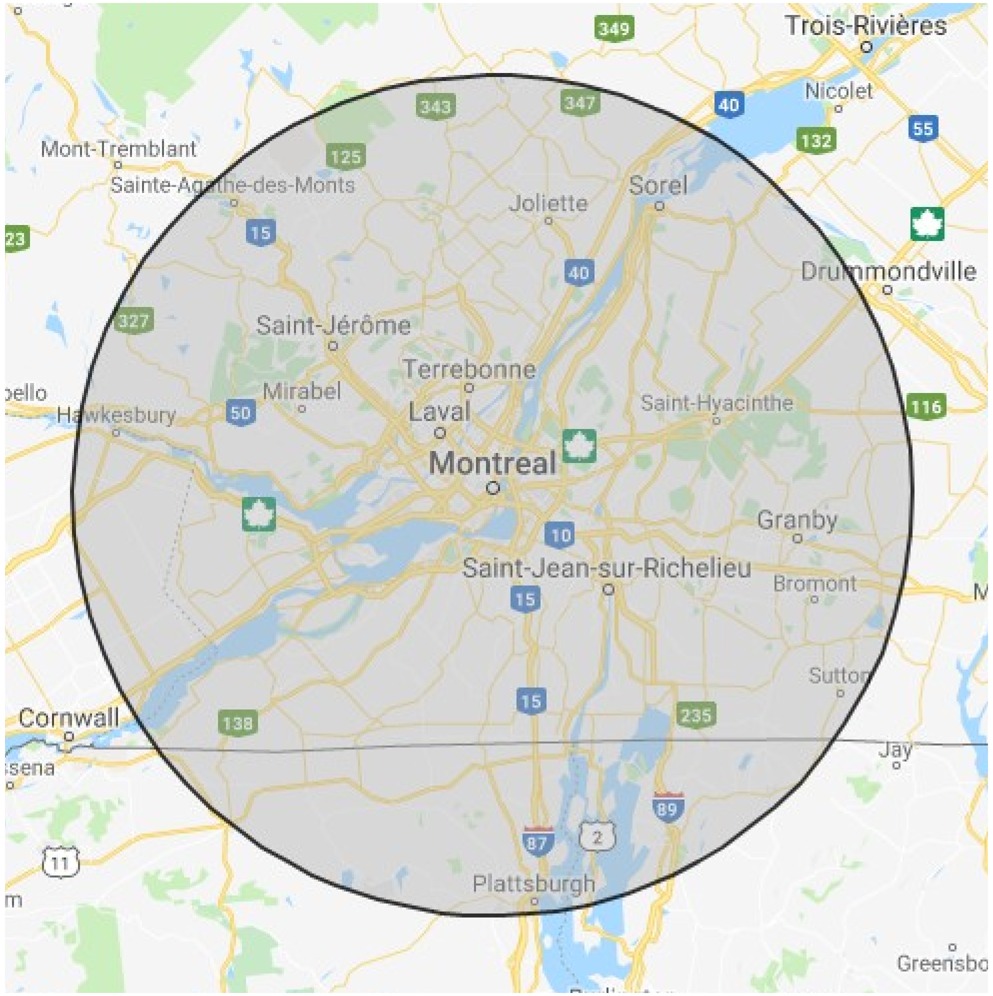
- An error of 300μs translates to
  - 90km at the speed of light [6, 7]
  - $1.2 million for online trading [5]



Source: Google

Importance of Latency

- An error of 300μs translates to
  - 90km at the speed of light [6, 7]
  - $1.2 million for online trading [5]
- Hard to isolate latencies
  - OS, virtualization, physical



Source: Google

How to account for latency in a running container system?

How to account for latency in a running container system?

MACE:

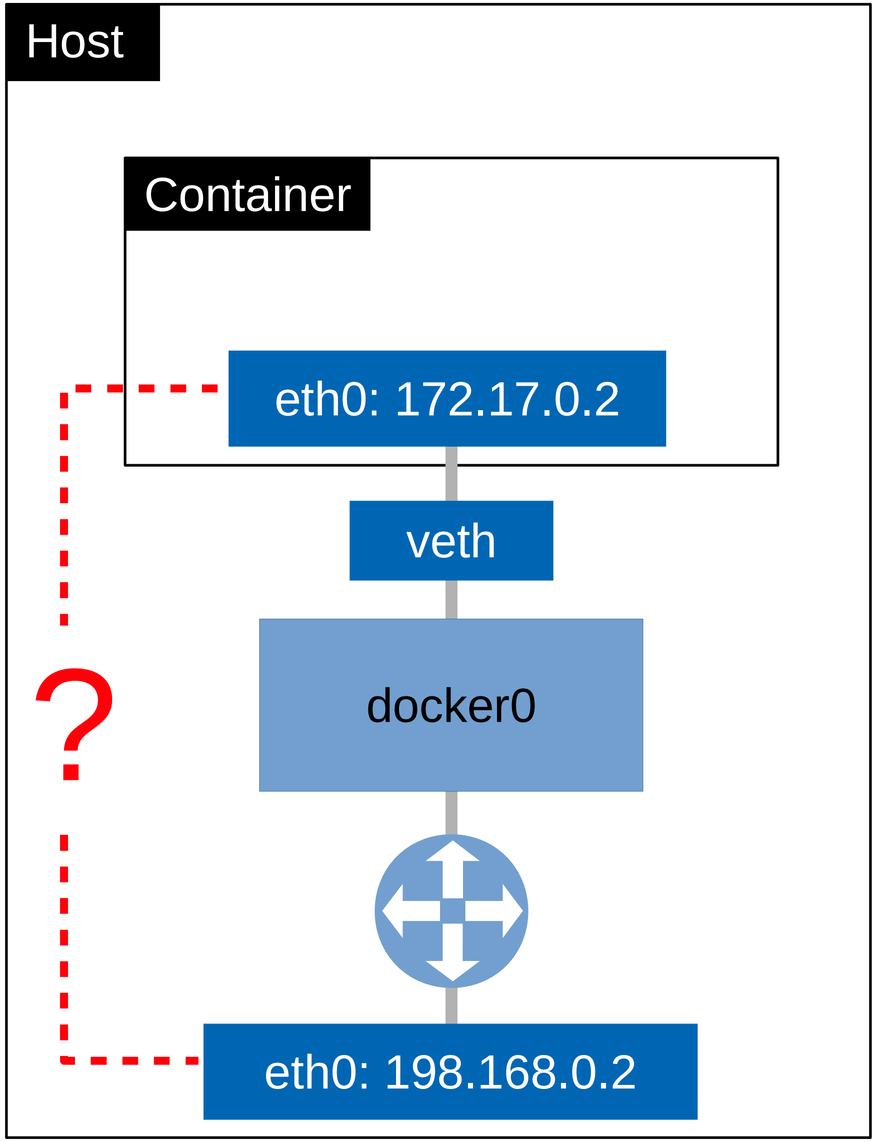
Measure the Added Container Expense

Outline

- Containerized measurement issues
- **Proposed solution: MACE**
- Evaluation of MACE

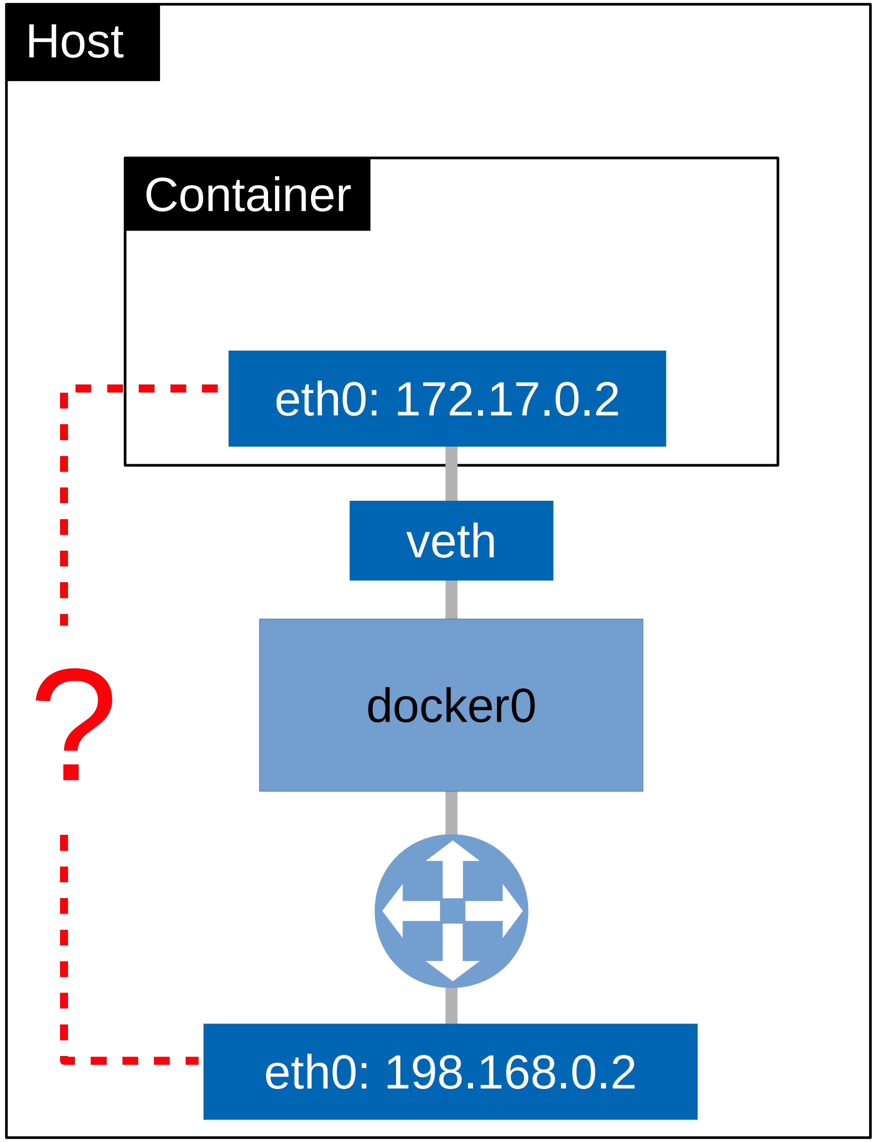
MACE: Goals

- Packet-level latencies
  - Ingress and egress
  - High accuracy



MACE: Goals

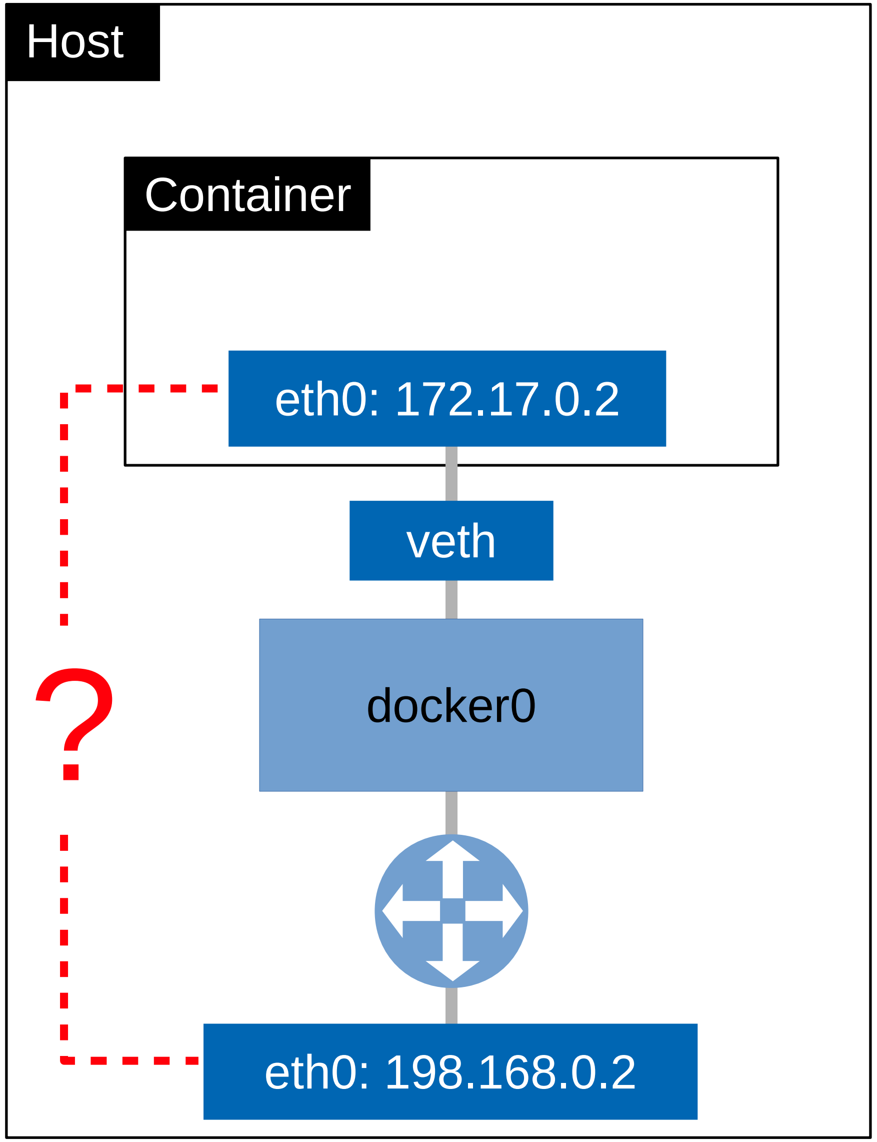
- Packet-level latencies
  - Ingress and egress
  - High accuracy
- Minimal impact on network performance



cmisa@cs.uoregon.edu

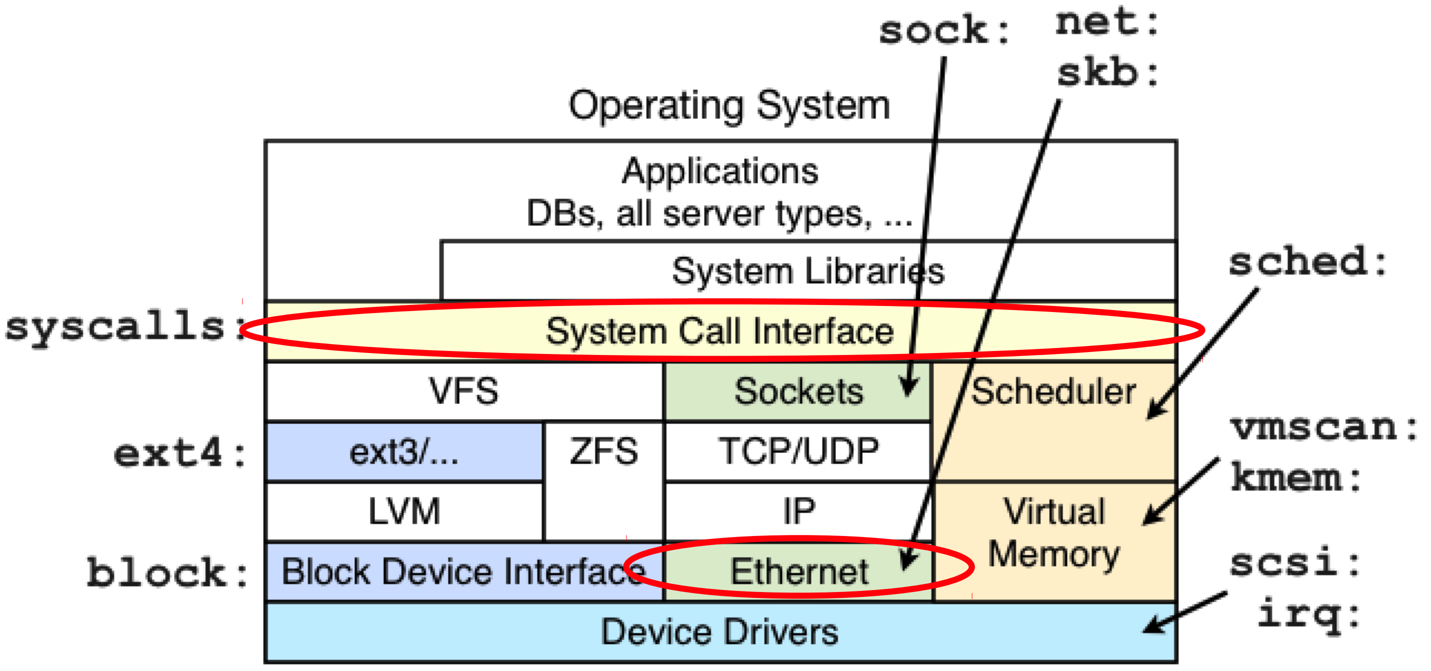
MACE: Goals

- Packet-level latencies
  - Ingress and egress
  - High accuracy
- Minimal impact on network performance
- Consistent, container-friendly interface



MACE: How?

- Linux Kernel Tracepoints [9]
  - Hooks into kernel
  - Net device and system call subsytems

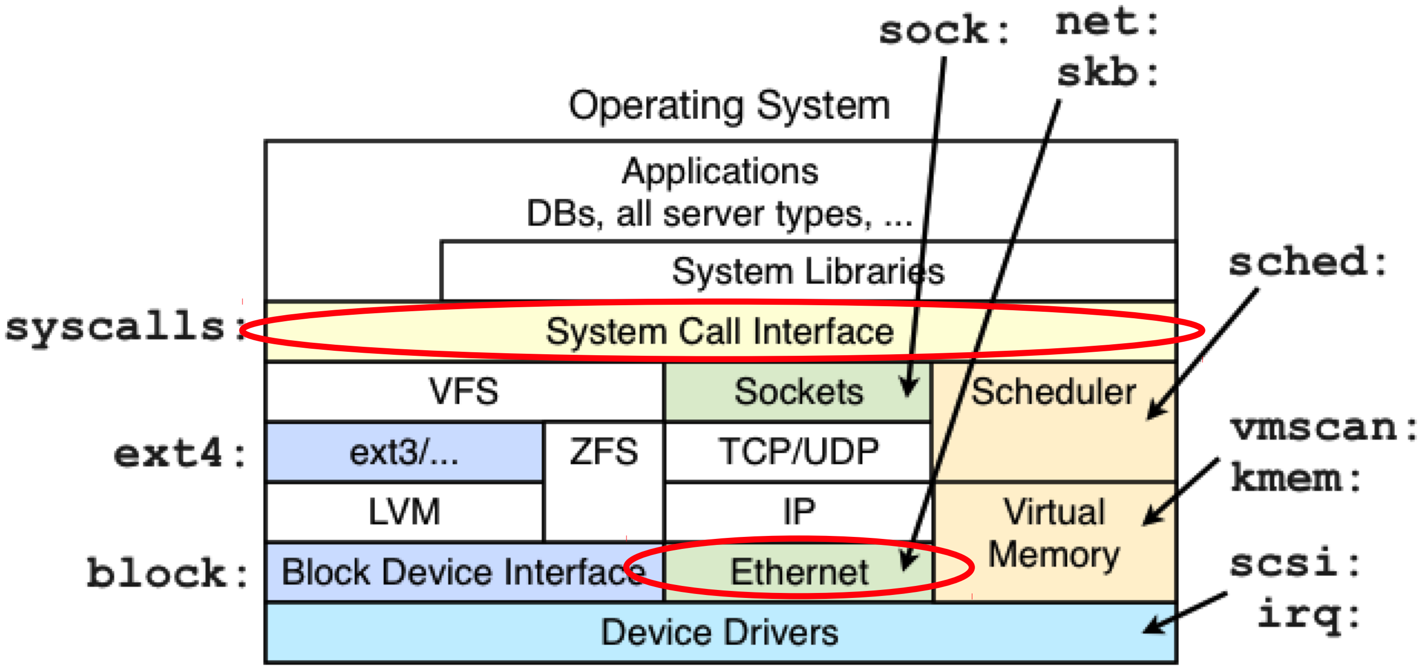


Static Tracepoints

Source: http://www.brendangregg.com

MACE: How?

- Linux Kernel Tracepoints [9]
  - Hooks into kernel
  - Net device and system call subsytems
- Existing tracers
  - Large perturbation

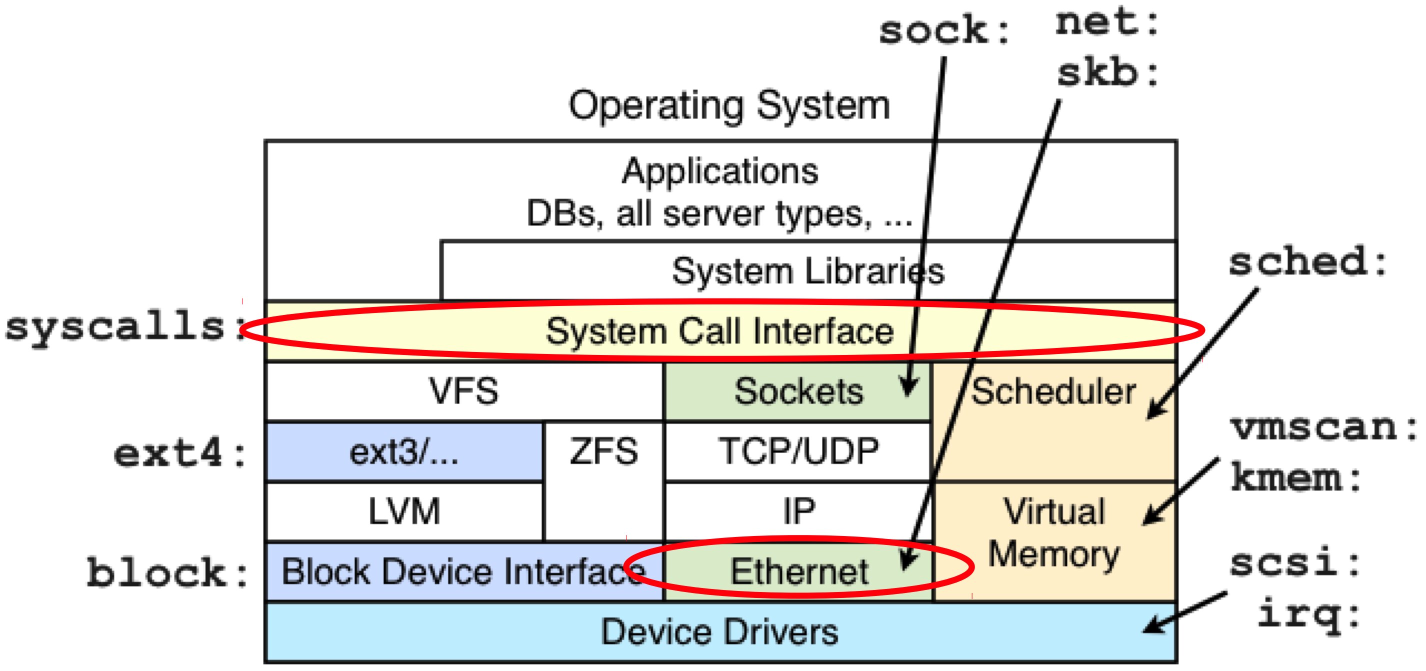


Static Tracepoints

Source: http://www.brendangregg.com

MACE: How?

- Linux Kernel Tracepoints [9]
  - Hooks into kernel
  - Net device and system call subsytems
- Existing tracers
  - Large perturbation
- Kernel module
  - For container hosts
  - Report to containers

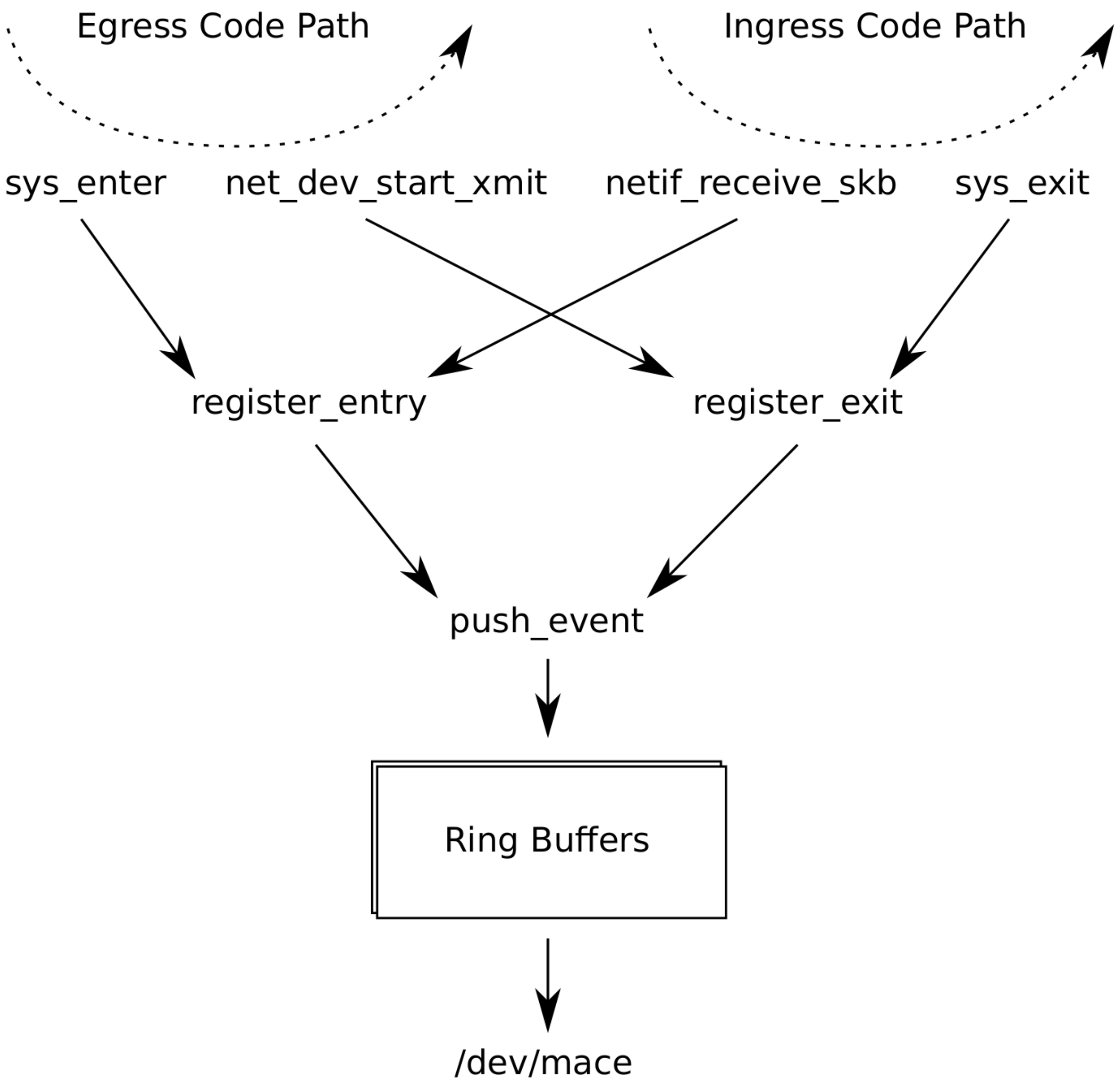


Static Tracepoints

Source: http://www.brendangregg.com

MACE: Design

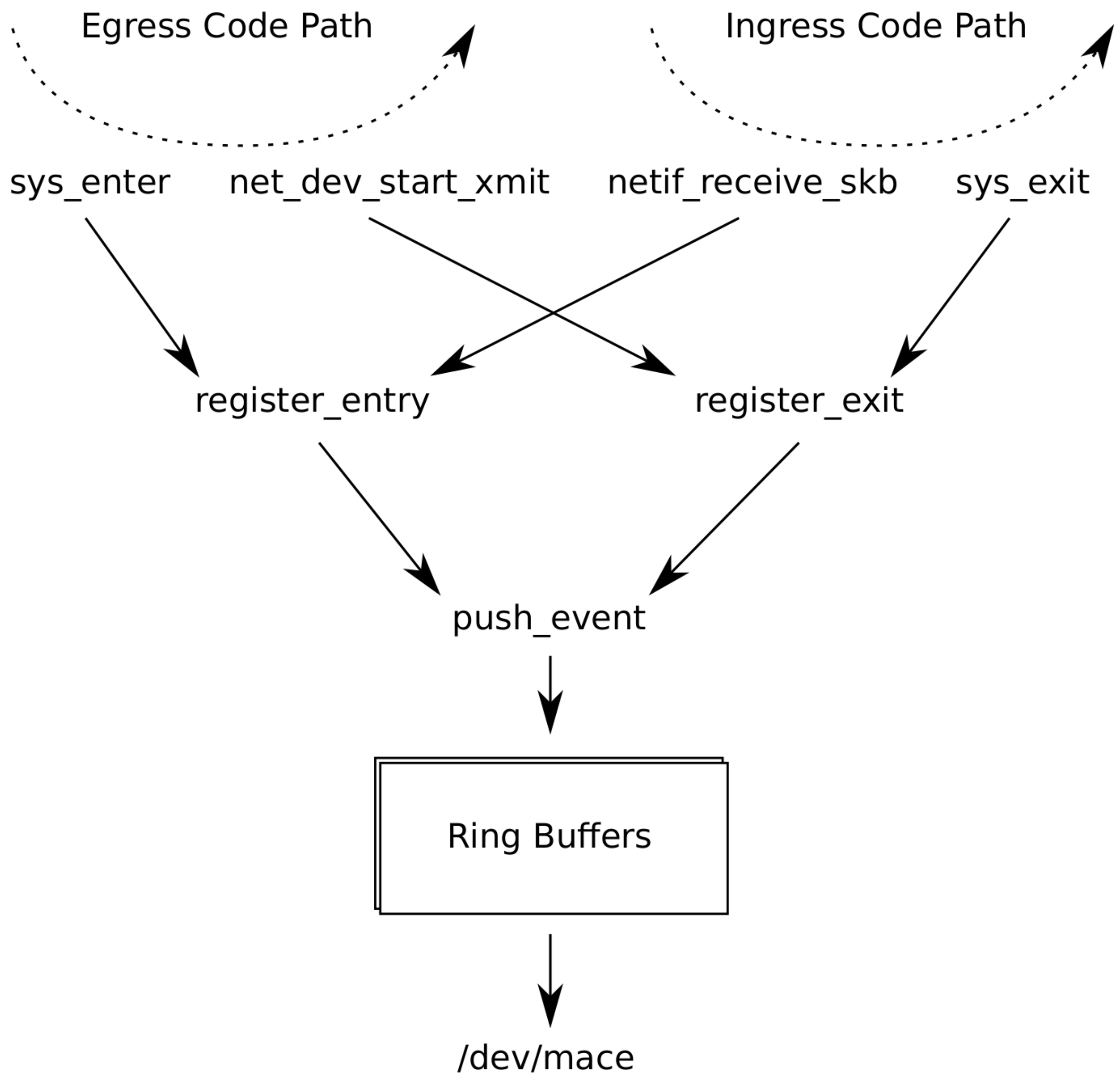
- Filter trace events
  - Interface
  - Namespace



cmisa@cs.uoregon.edu

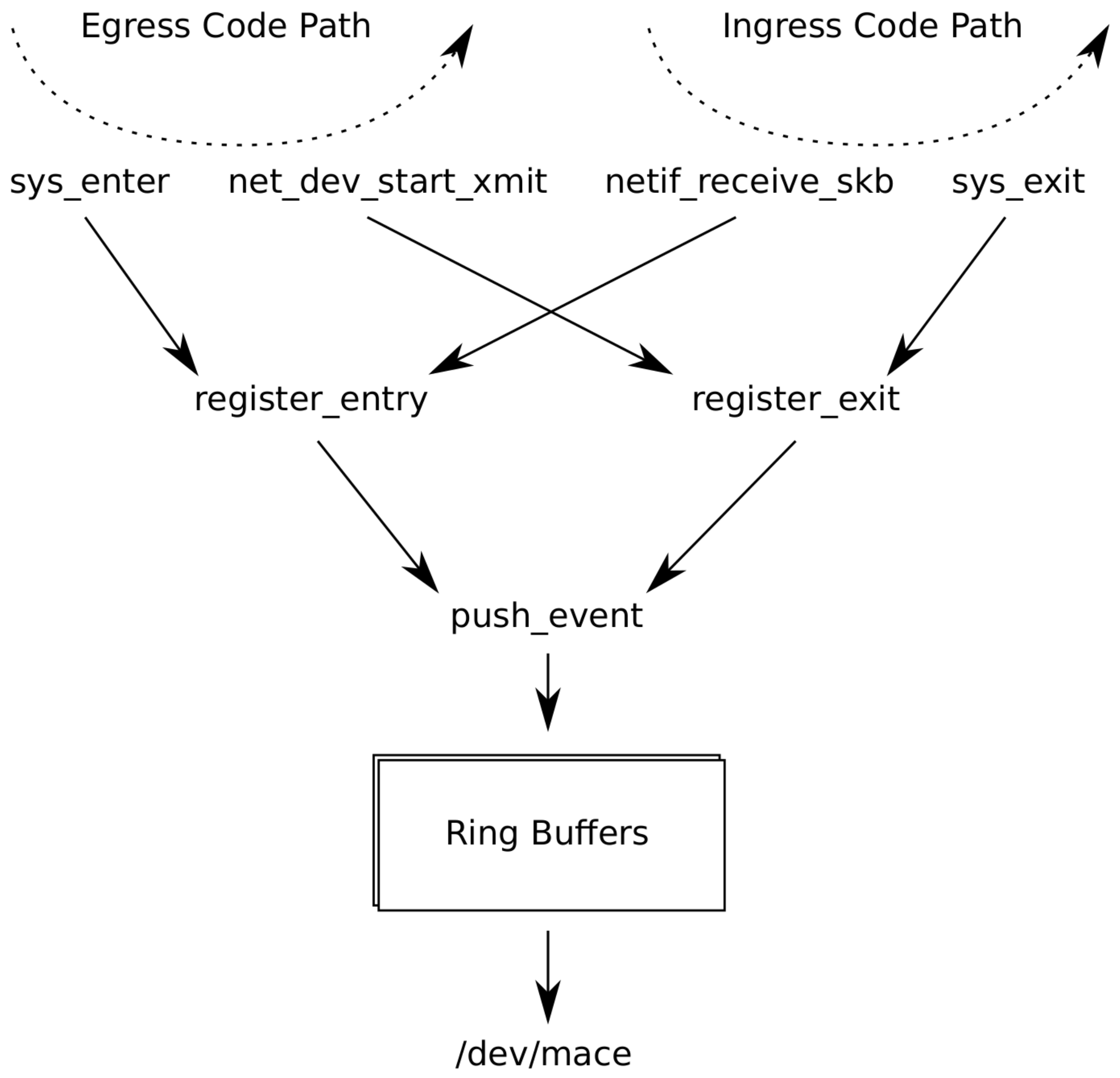
MACE: Design

- Filter trace events
  - Interface
  - Namespace
- Correlate events in hash tables
  - Ingress
  - Egress



MACE: Design

- Filter trace events
  - Interface
  - Namespace
- Correlate events in hash tables
  - Ingress
  - Egress
- Maintain list of latencies
  - Report via device file



MACE: Implementation

- High accuracy
  - Read tsc for timing

Open source at:
github.com/chris-misa/mace

MACE: Implementation

- High accuracy
  - Read tsc for timing
- Low perturbation
  - Only lock hash buckets
  - Atomic types for ring buffer

Open source at:
github.com/chris-misa/mace

cmisa@cs.uoregon.edu

MACE: Implementation

- High accuracy
  - Read tsc for timing
- Low perturbation
  - Only lock hash buckets
  - Atomic types for ring buffer
- Consistent API
  - Interface is namespace-aware
  - Allow and enable per container

Open source at:
github.com/chris-misa/mace

MACE: Interface

- Select the container’s namespace:

```
# echo 1 > sys/class/mace/on
```

MACE: Interface

- Select the container’s namespace:

```
# echo 1 > sys/class/mace/on
```

- Execute measurement:

```
# ping –c 10 google.com
```

MACE: Interface

- Select the container's namespace:

```
# echo 1 > sys/class/mace/on
```

- Execute measurement:

```
# ping -c 10 google.com
```

- Collect latencies:

```
# cat dev/mace
[1552589043.315681] (1) egress: 80932
[1552589043.315937] (1) ingress: 46208
[1552589043.316012] (2) egress: 13699
...
```

How do we know those numbers are correct?

Outline

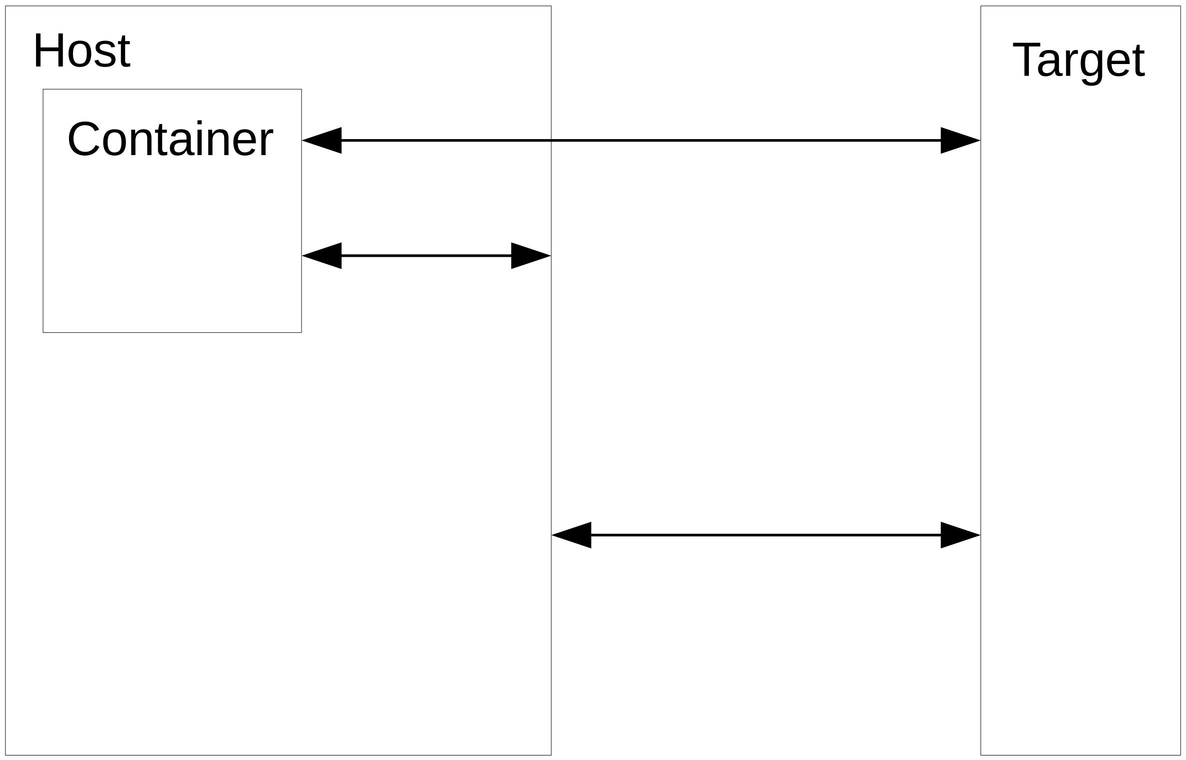
- Containerized measurement issues
- Proposed solution: MACE
- **Evaluation of MACE**

Evaluation: Methodology

- No direct method

Evaluation: Methodology

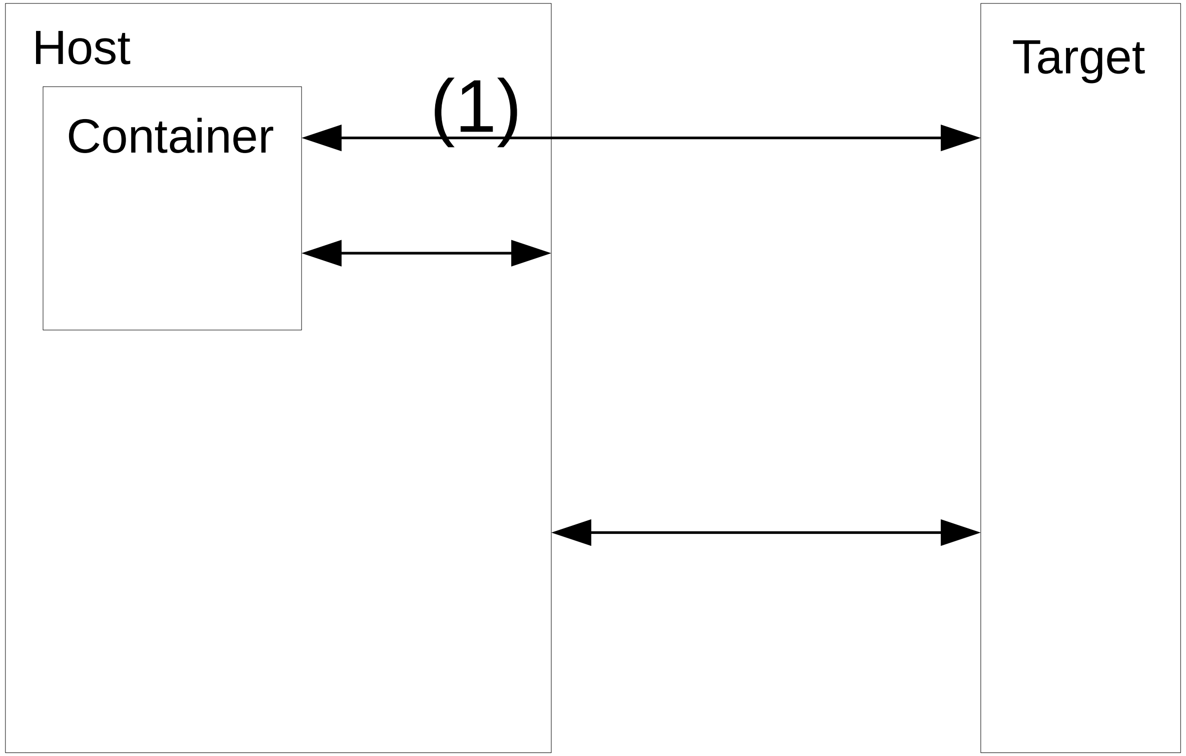
- No direct method
- Use difference in RTT



Evaluation: Methodology

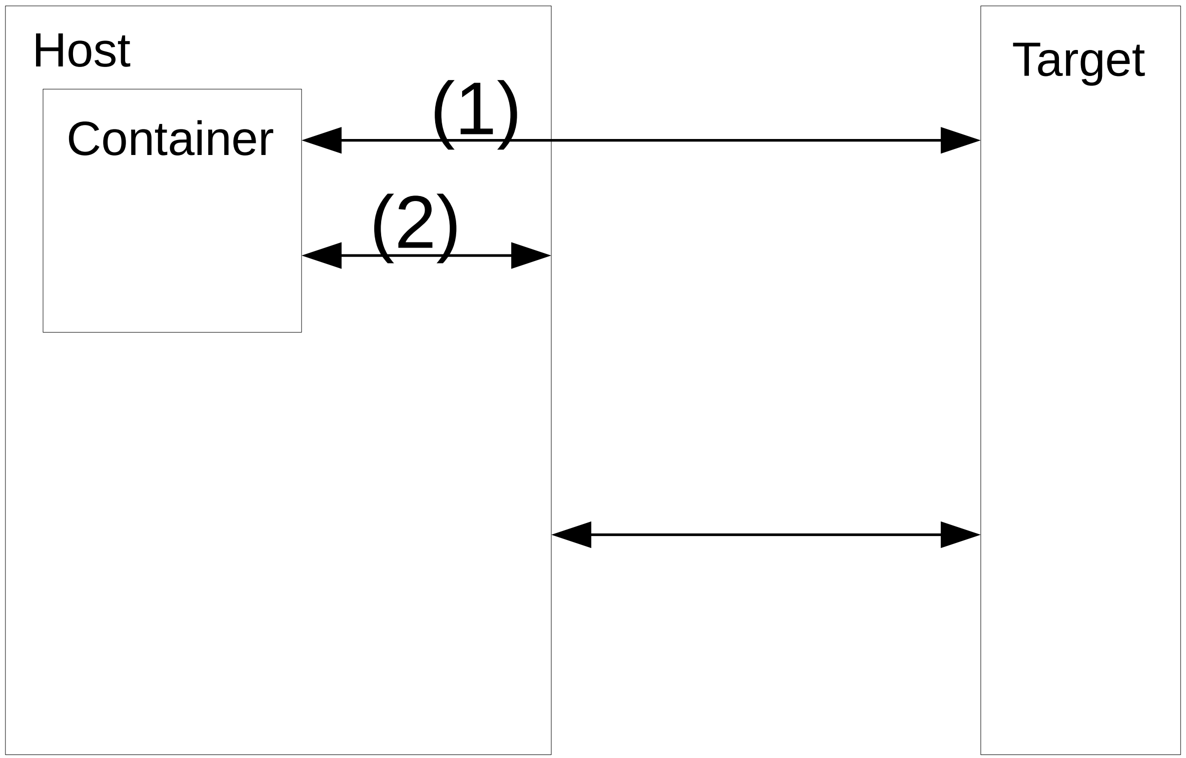
- No direct method
- Use difference in RTT

    (1) RTT from container



Evaluation: Methodology

- No direct method
- Use difference in RTT
  - (1) RTT from container
  - (2) Latency overheads from MACE



Evaluation: Methodology

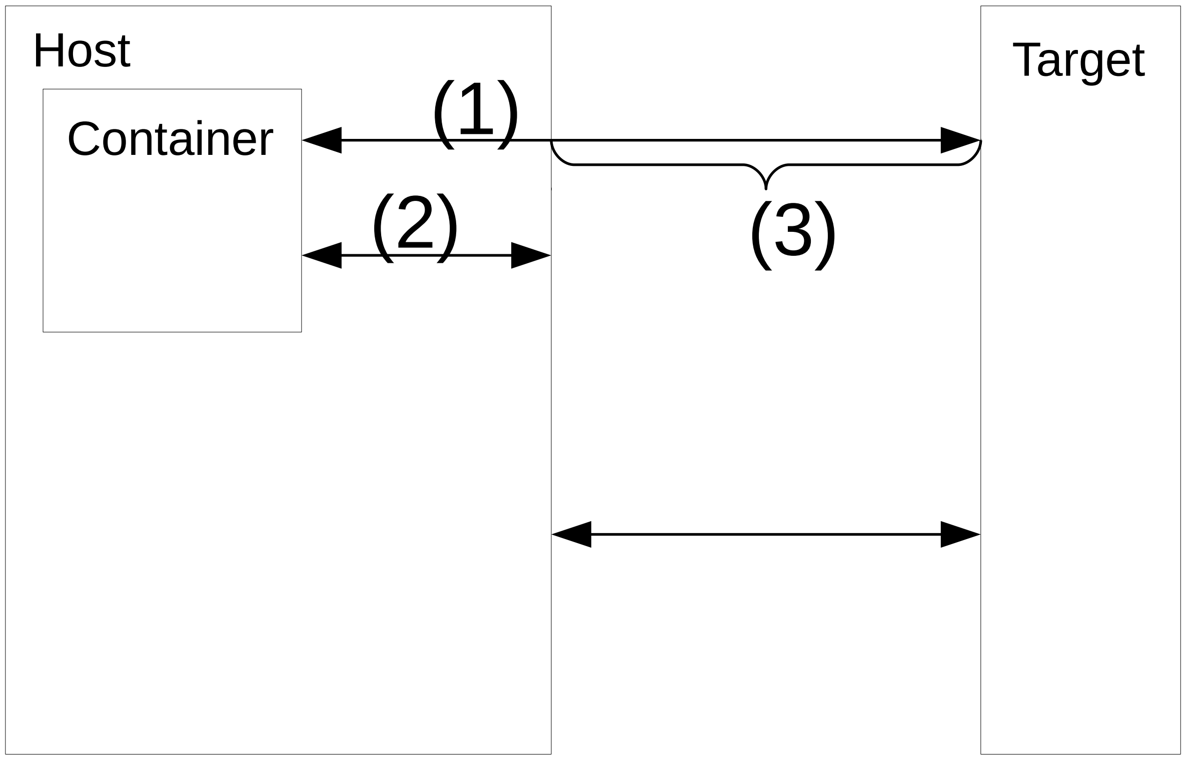
- No direct method
- Use difference in RTT

  (1) RTT from container

  (2) Latency overheads from MACE

  (3) 'corrected' RTT

  = (1) minus (2)



Evaluation: Methodology

- No direct method
- Use difference in RTT

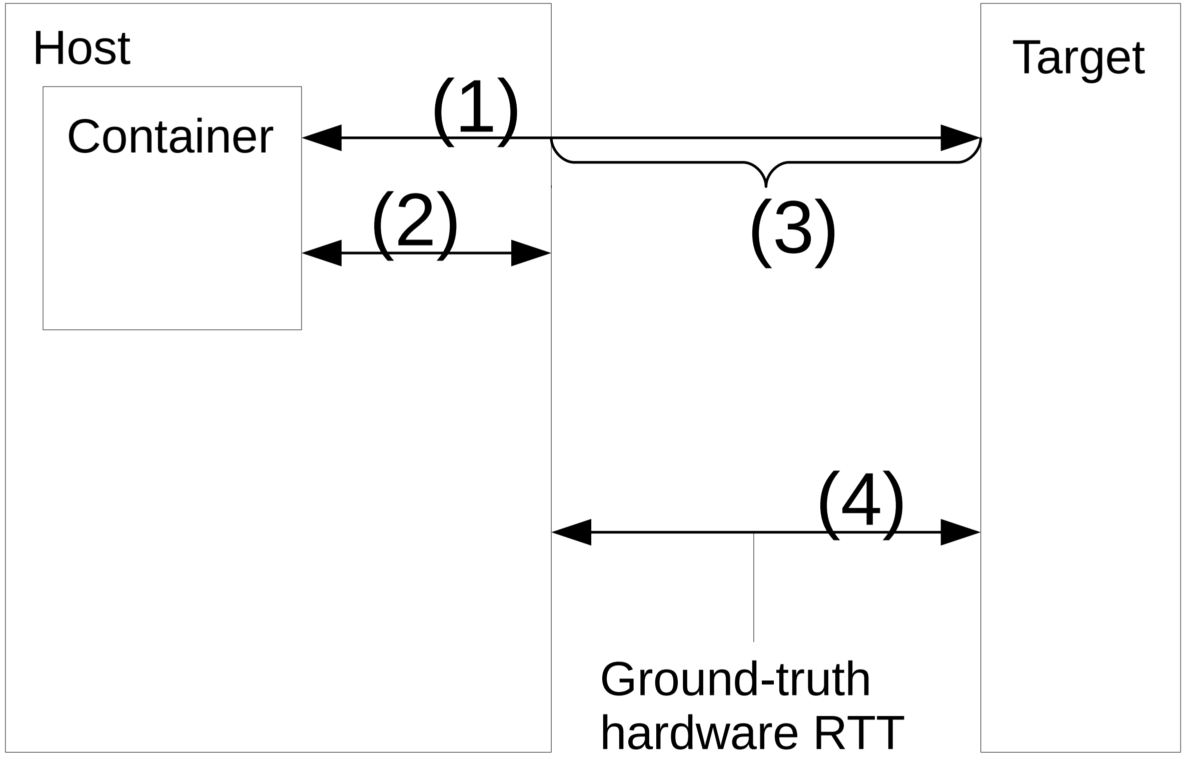
  (1) RTT from container

  (2) Latency overheads from MACE

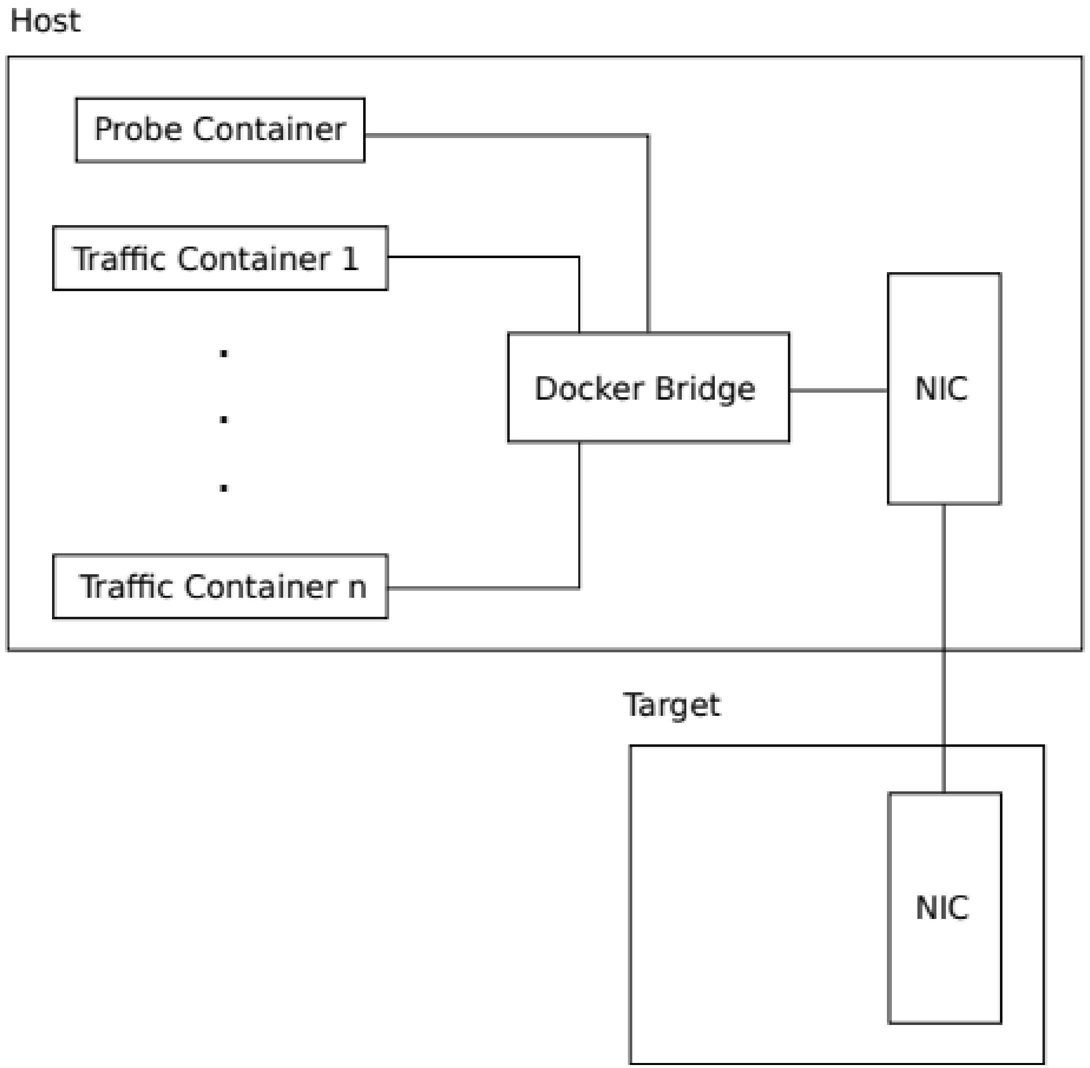
  (3) ‘corrected’ RTT

  = (1) minus (2)

  (4) Compare with RTT measured from hardware

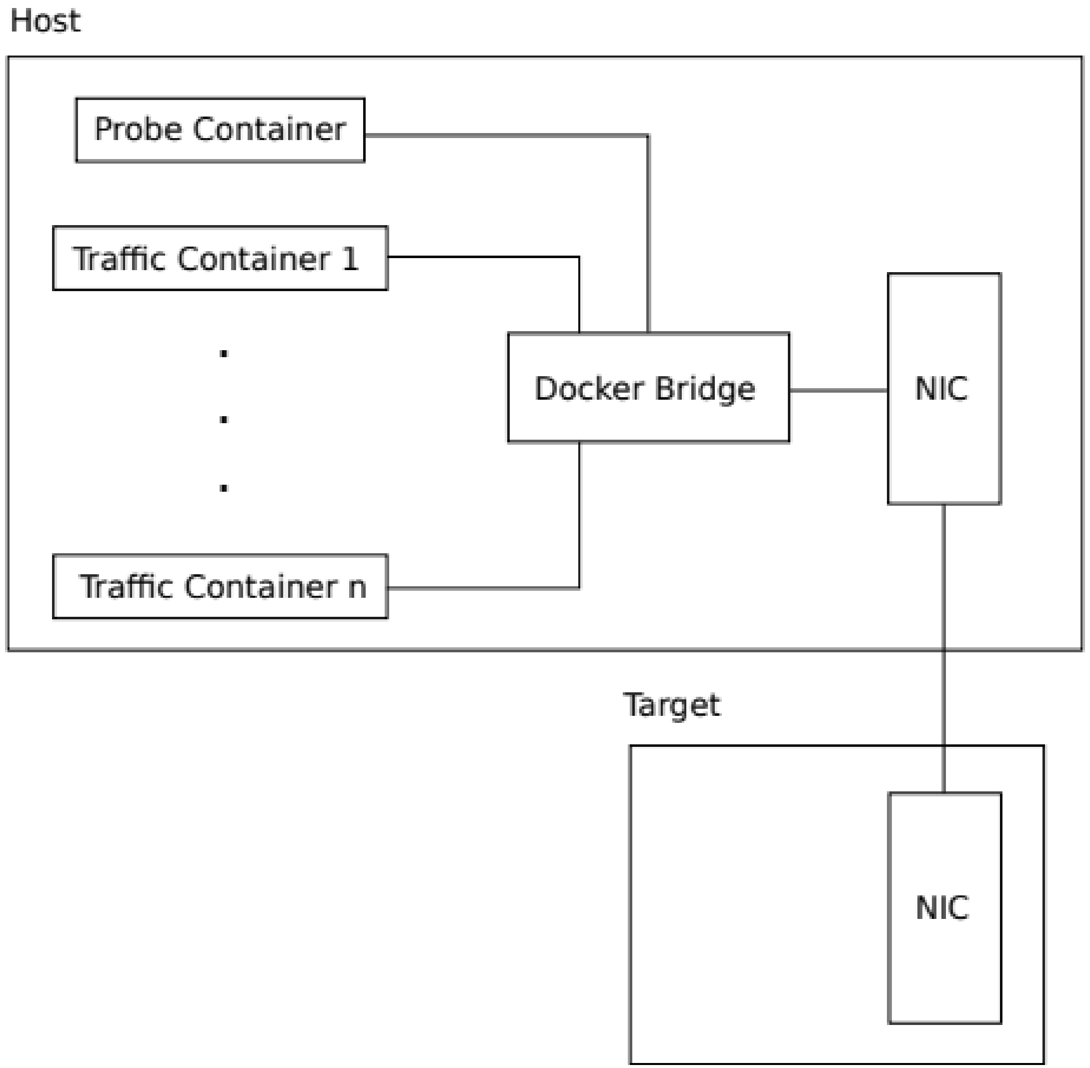


Evaluation: Setting



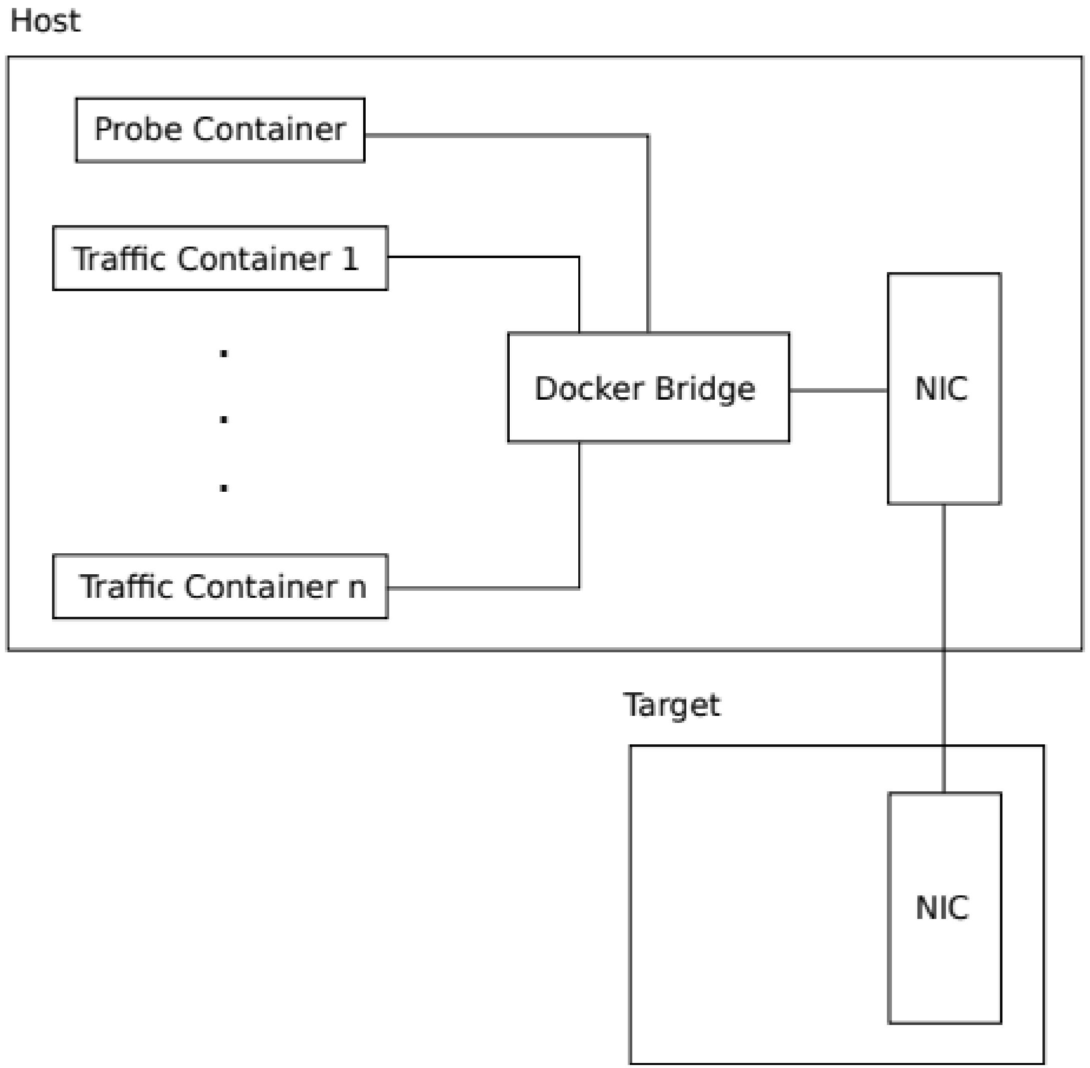
- Ping across single physical link
  - Minimize network latency

Evaluation: Setting



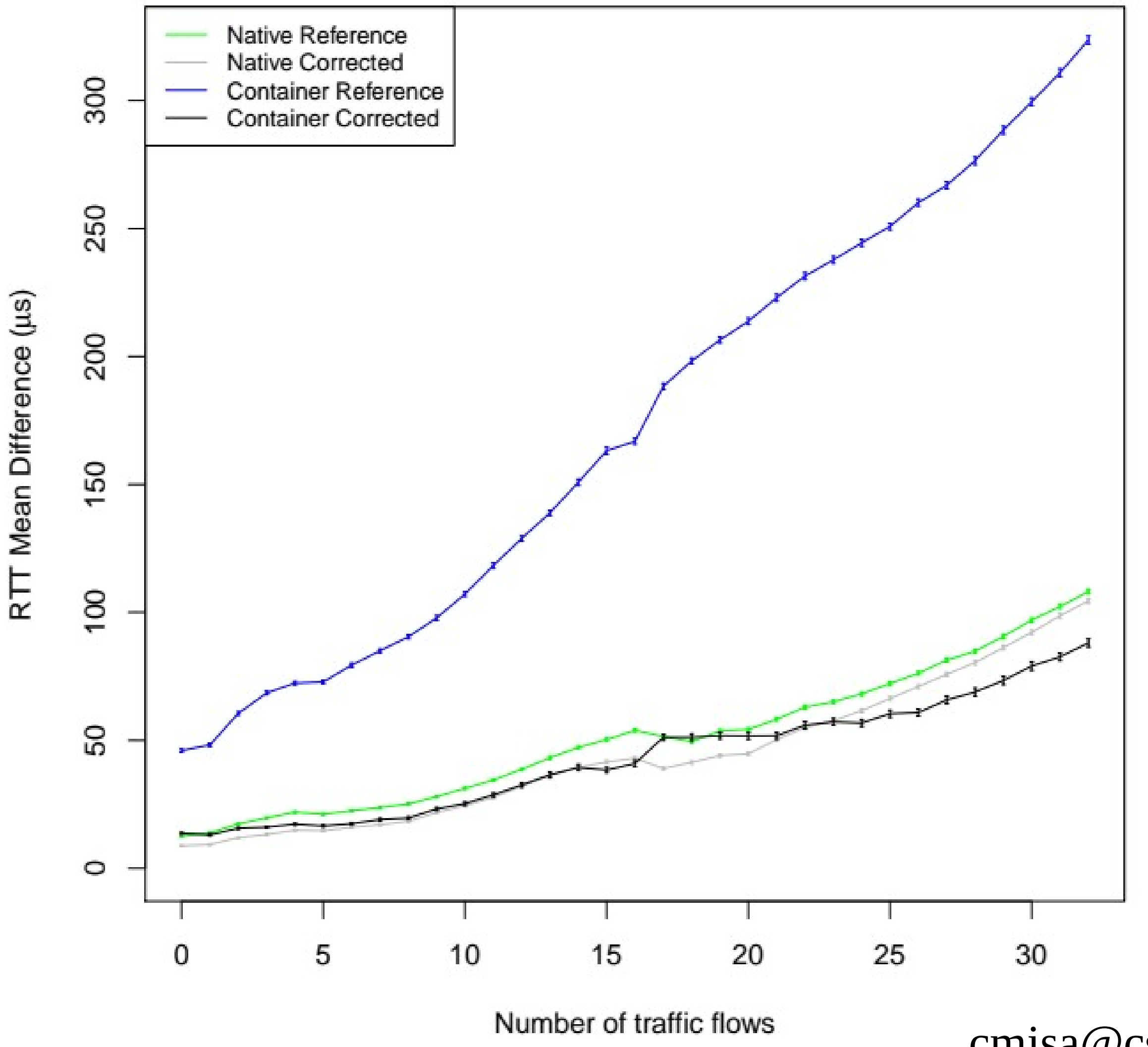
- Ping across single physical link
  - Minimize network latency
- Add co-located containers
  - Flood ping
  - Worst-case traffic setting

Evaluation: Setting



- Ping across single physical link
  - Minimize network latency
- Add co-located containers
  - Flood ping
  - Worst-case traffic setting
- Run on Cloudlab [10]
  - Some RTT noise from experiment network

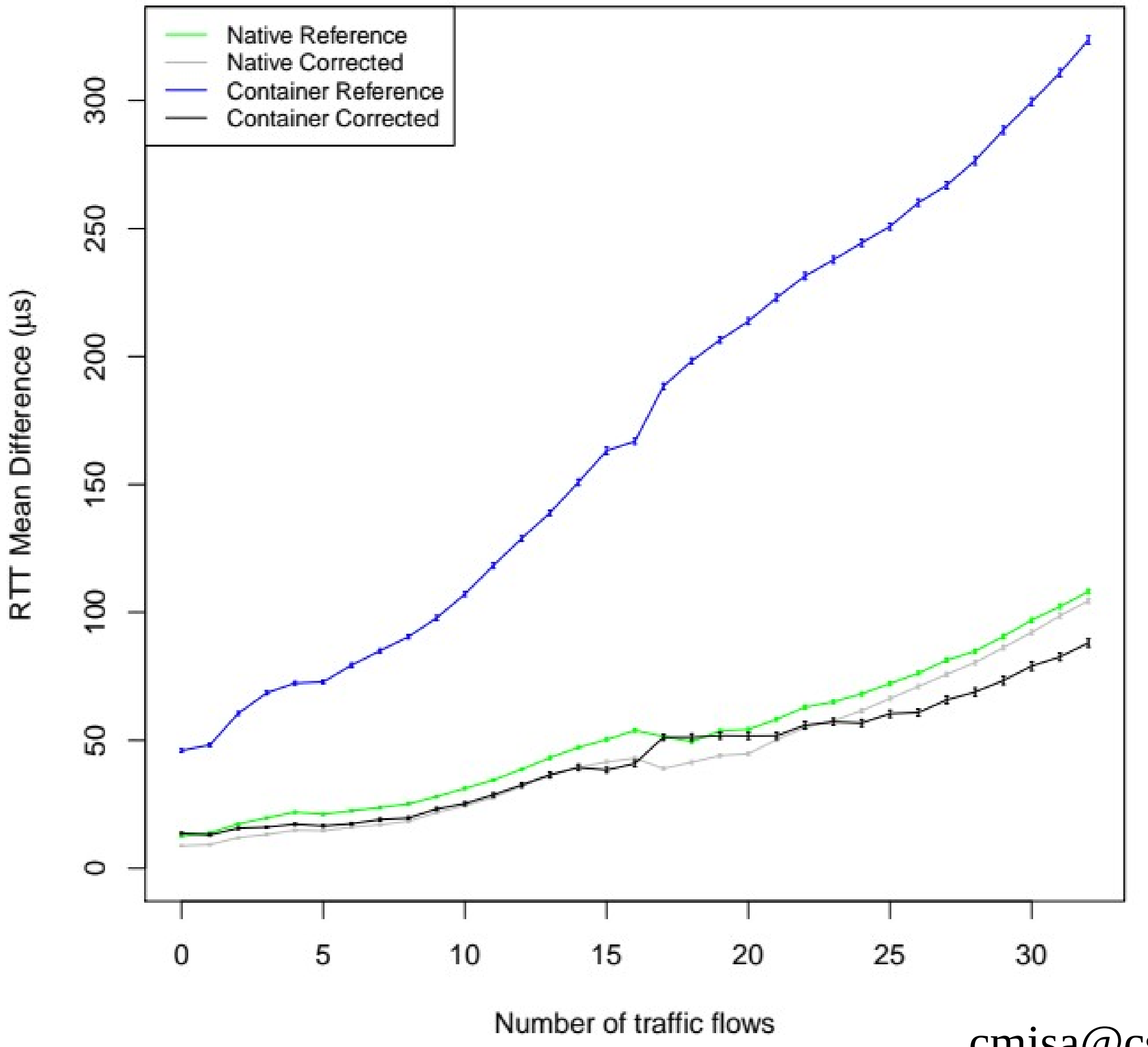
Results: RTT Bias



- Reported RTT - actual RTT
  - ‘raw’ container (blue)
  - ‘corrected’ container (black)

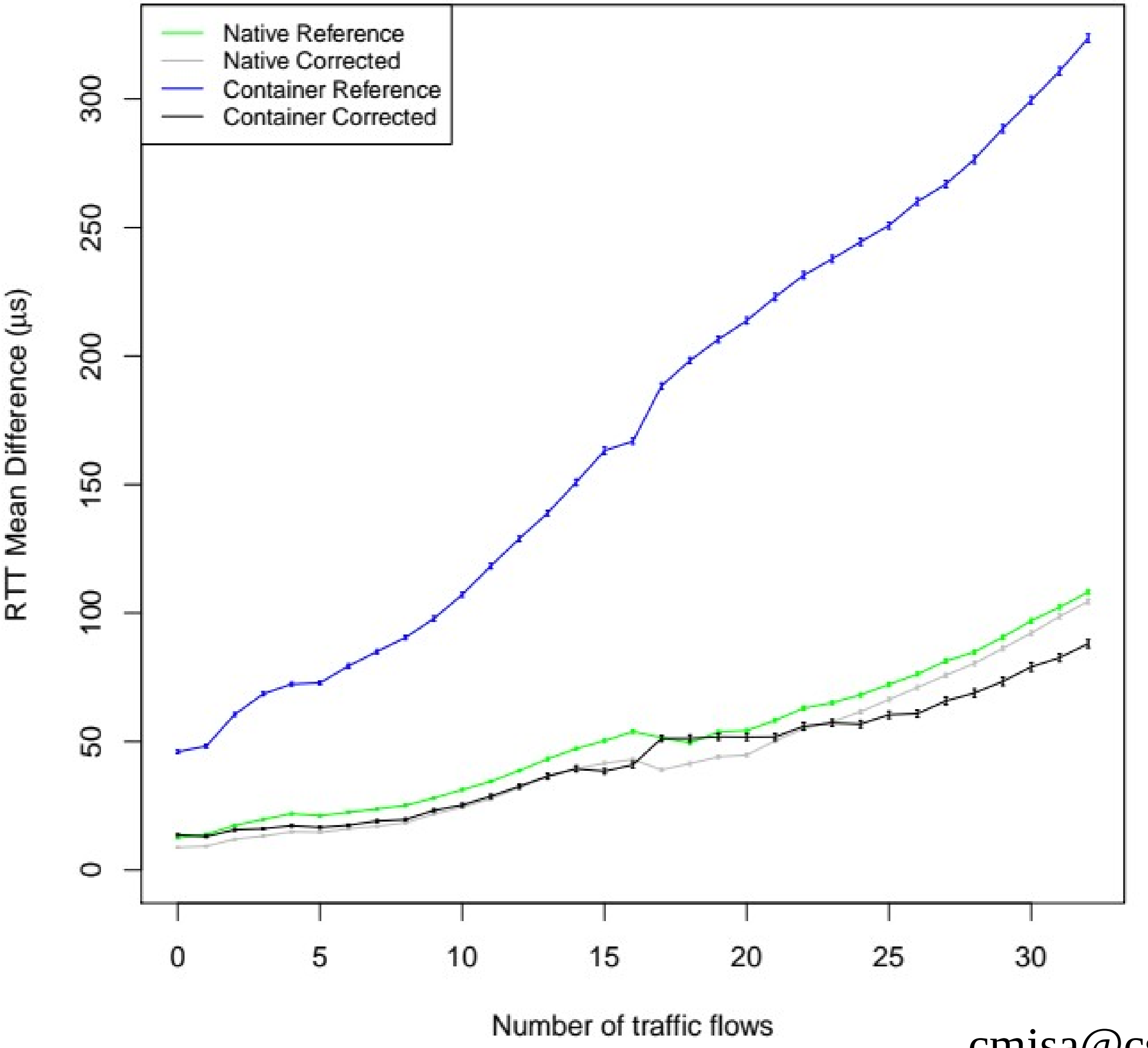
cmisa@cs.uoregon.edu

Results: RTT Bias



- Reported RTT - actual RTT
  - ‘raw’ container (blue)
  - ‘corrected’ container (black)
- MACE-corrected RTT is **within 20µs** in worst case

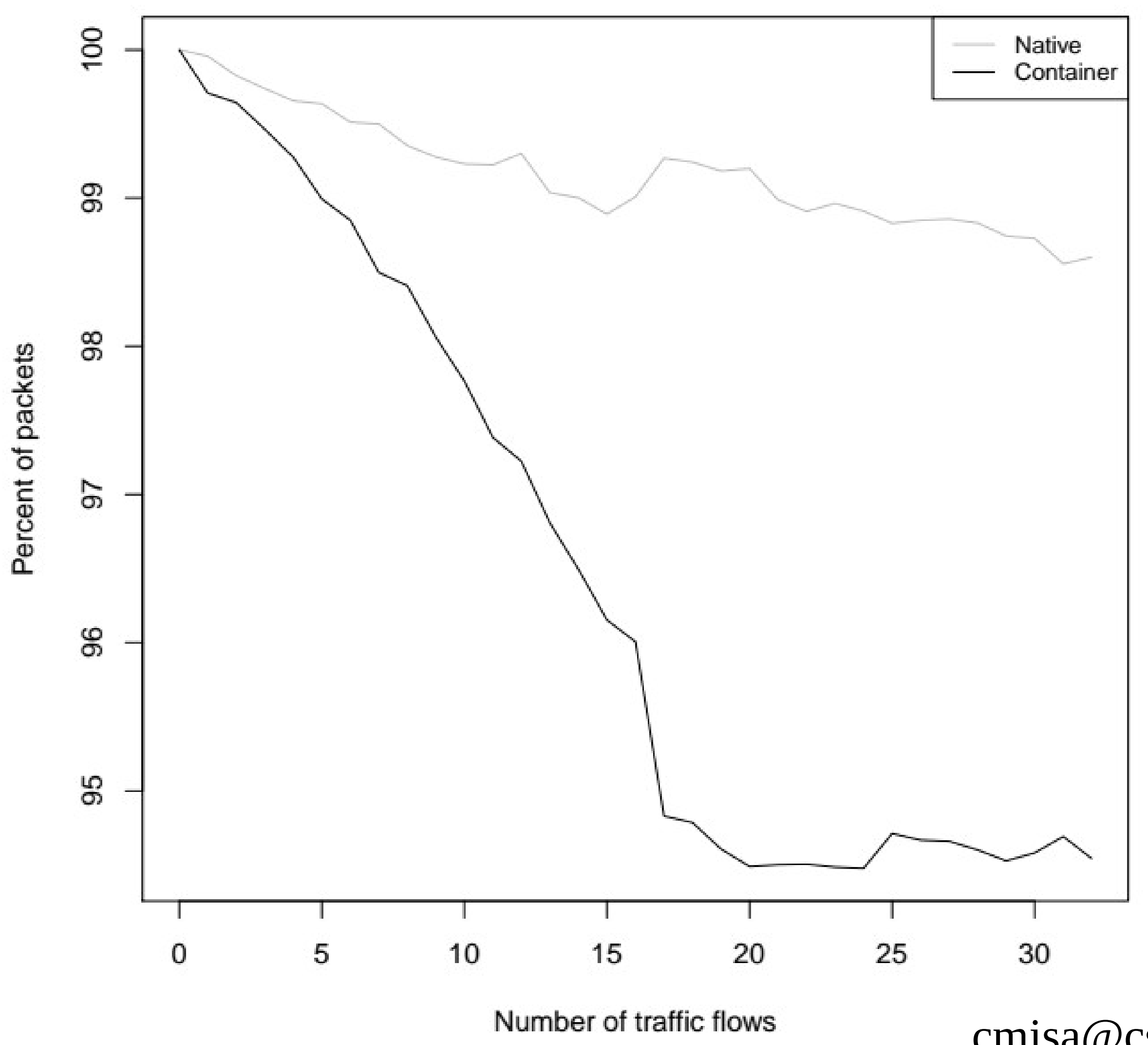
Results: RTT Bias



- Reported RTT - actual RTT
  - ‘raw’ container (blue)
  - ‘corrected’ container (black)
- MACE-corrected RTT is **within 20µs** in worst case
- Traffic impacts all software RTTs
  - Up to **100 µs**

cmisa@cs.uoregon.edu

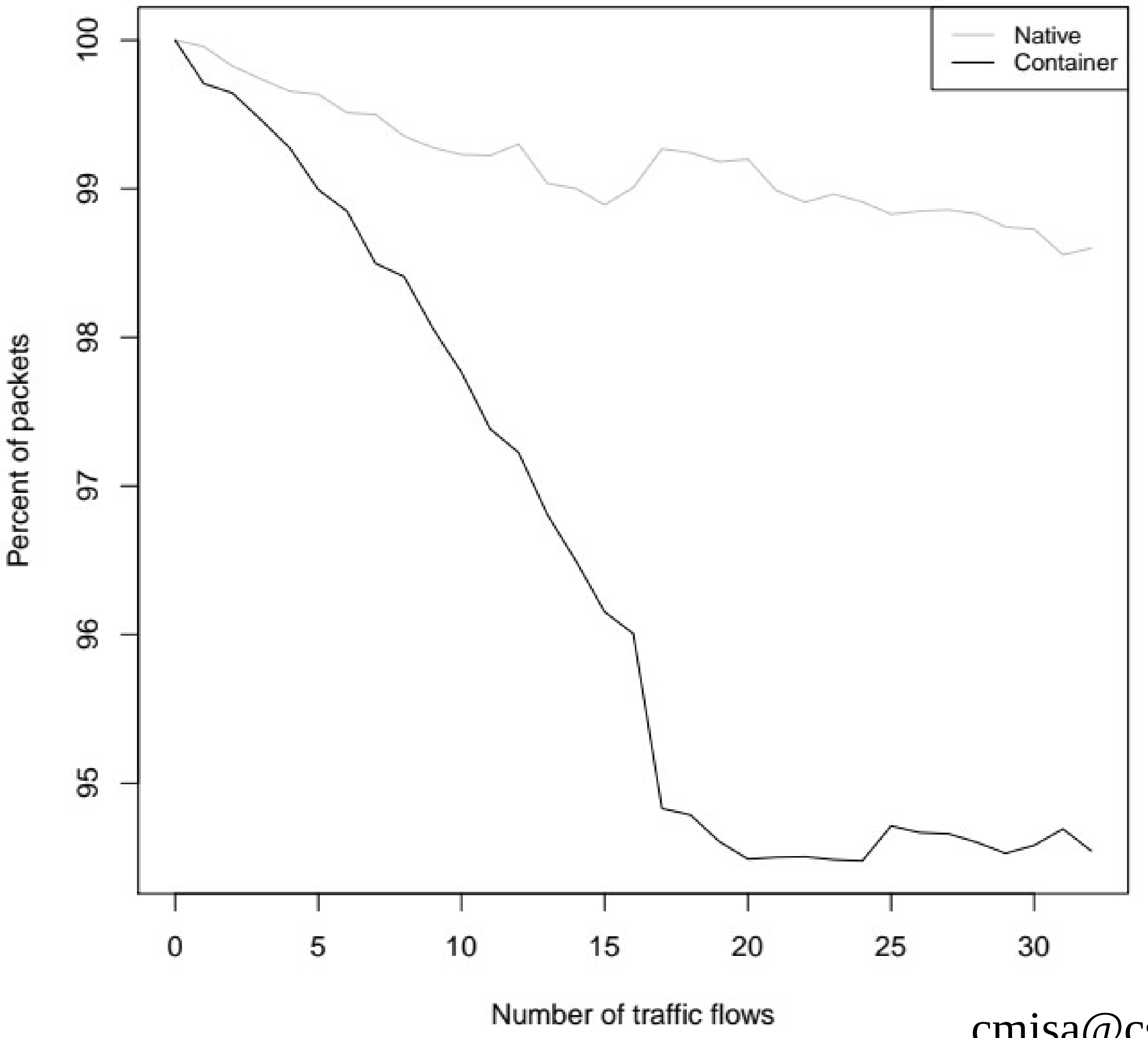
Results: Coverage



- Latency reports / packets (%)

cmisa@cs.uoregon.edu

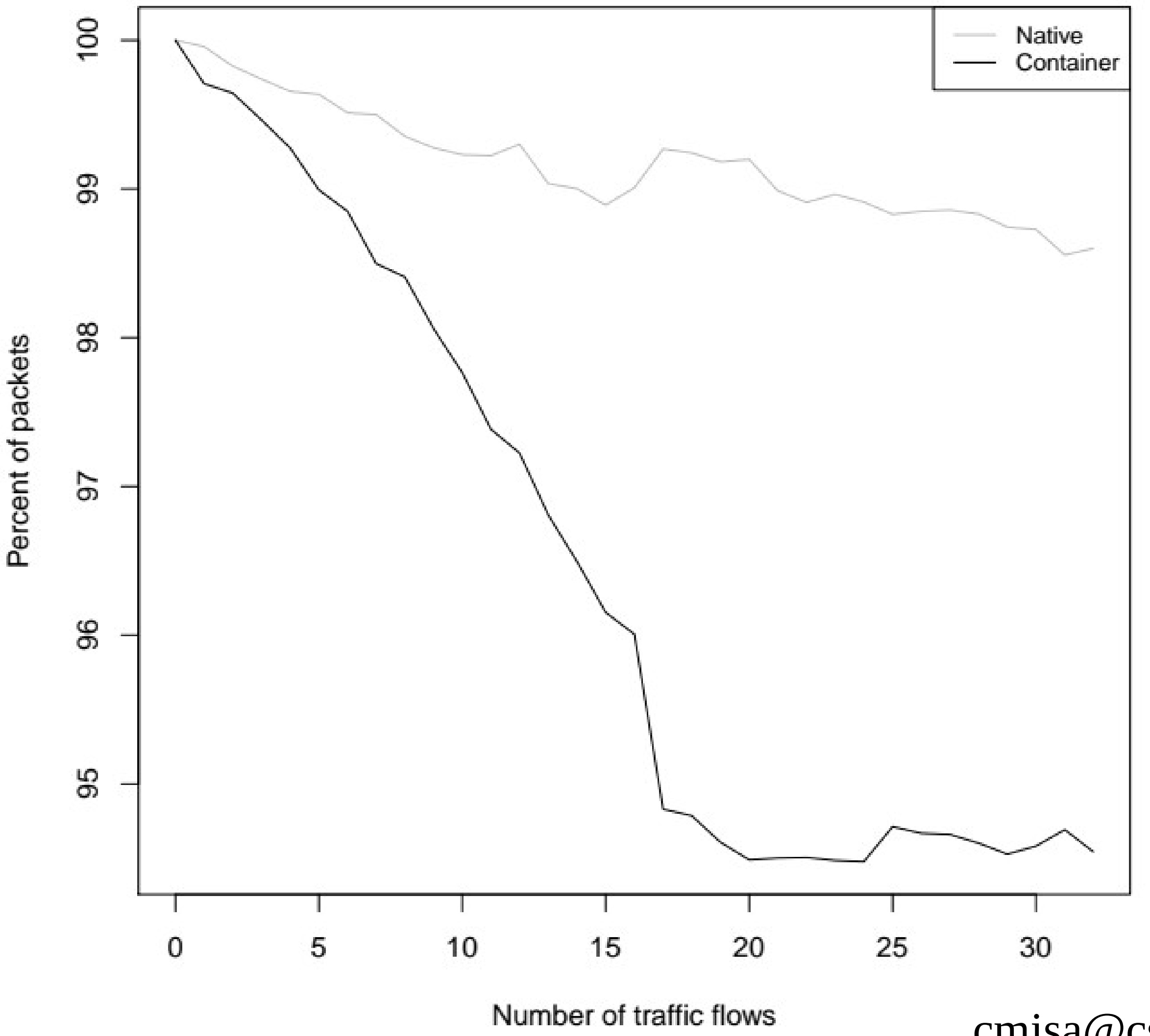
Results: Coverage



- Latency reports / packets (%)
- Decrease due to collisions in hash tables

cmisa@cs.uoregon.edu

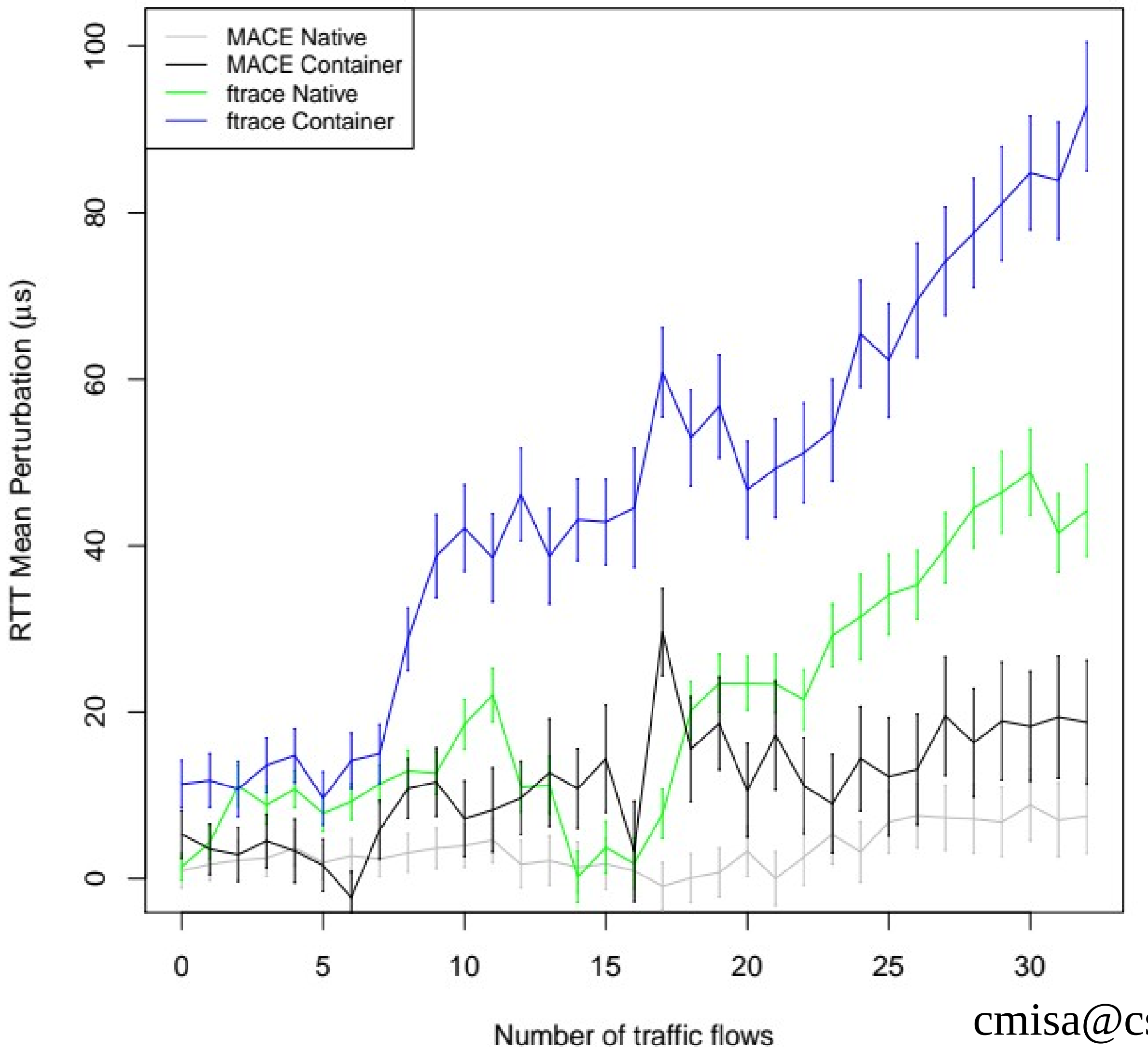
Results: Coverage



- Latency reports / packets (%)
- Decrease due to collisions in hash tables
- Increased table size can improve coverage to 100%

cmisa@cs.uoregon.edu

Results: Perturbation



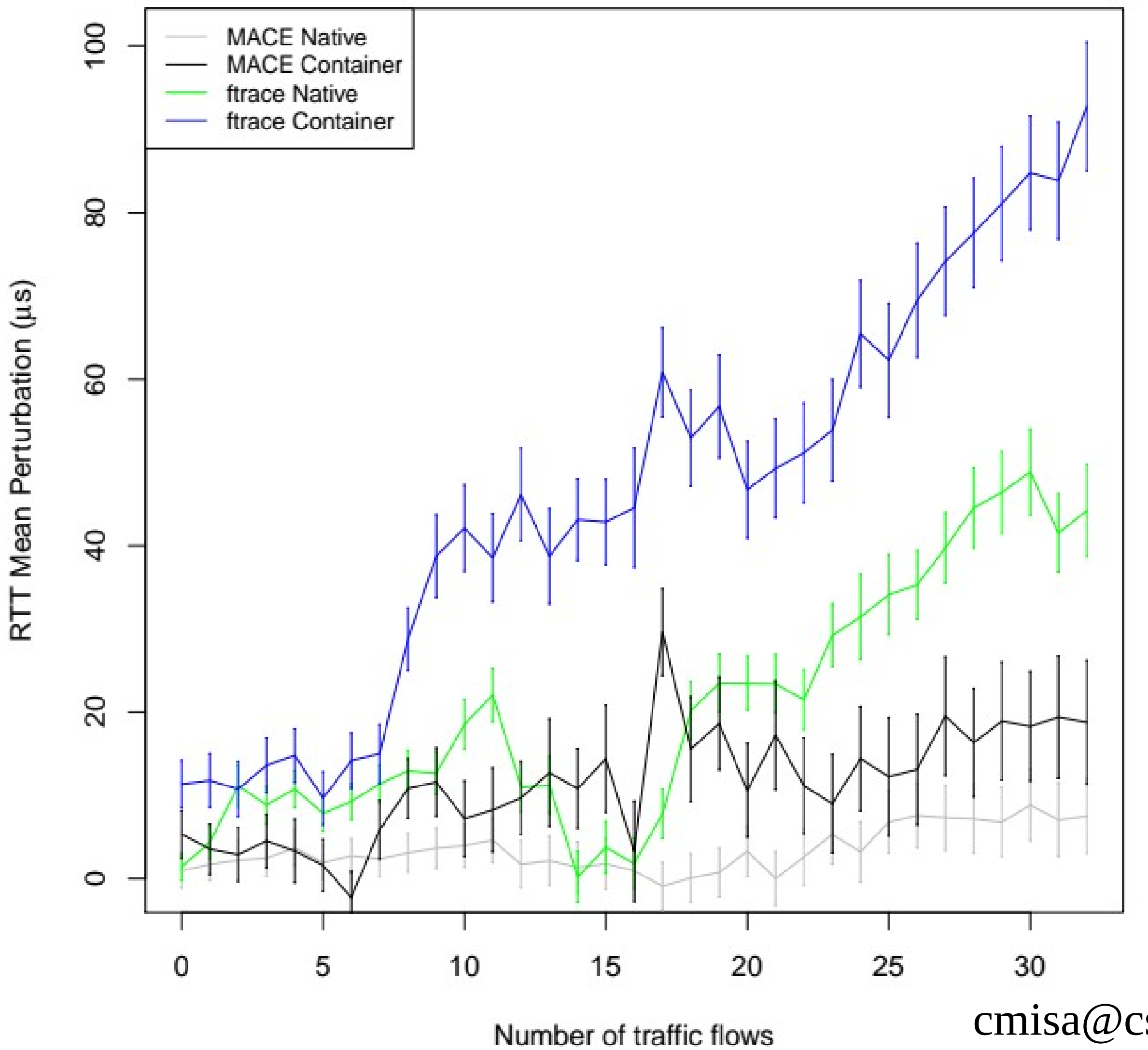
- Instrumented RTT

  minus non-instrumented RTT

  – MACE (black)
  – Ftrace (blue)

cmisa@cs.uoregon.edu

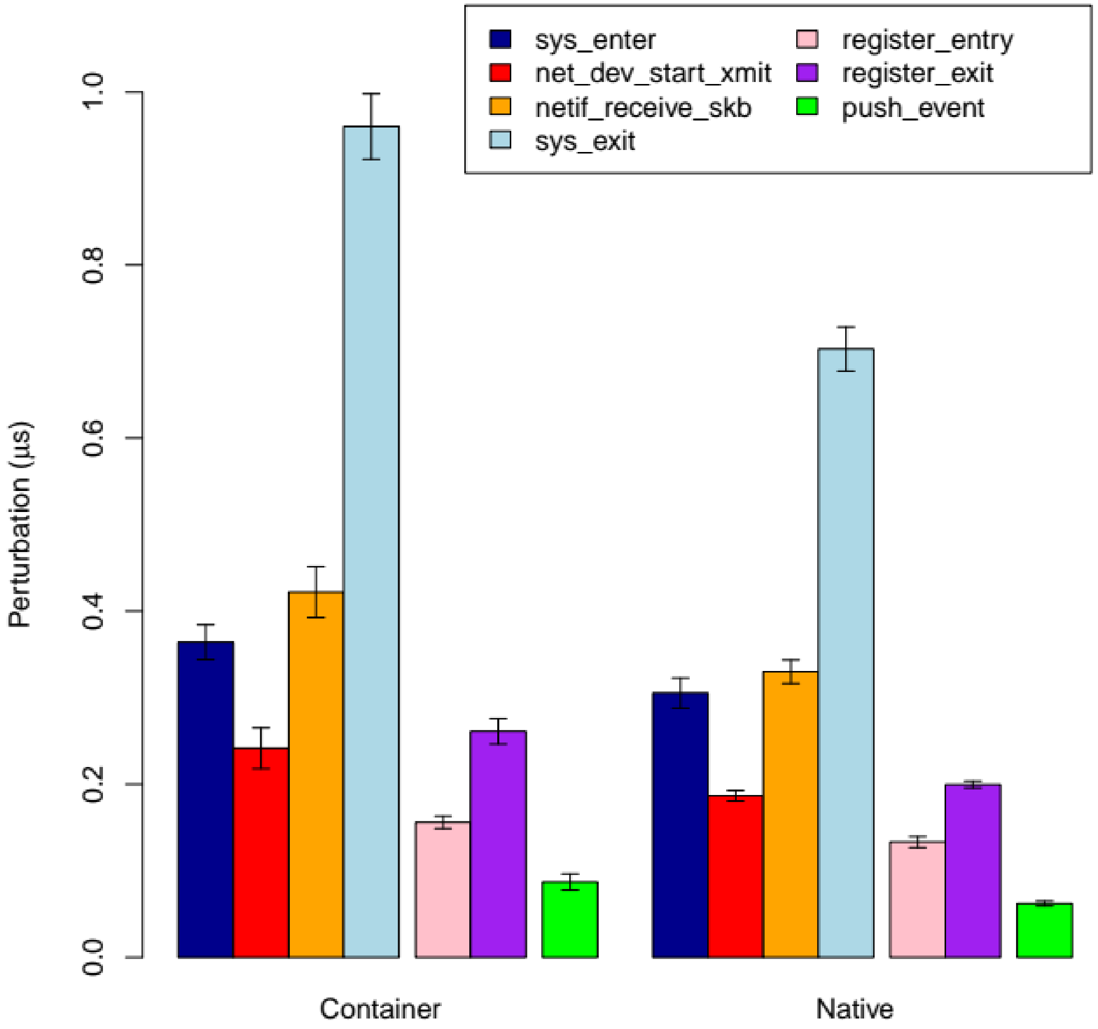
Results: Perturbation



- Instrumented RTT

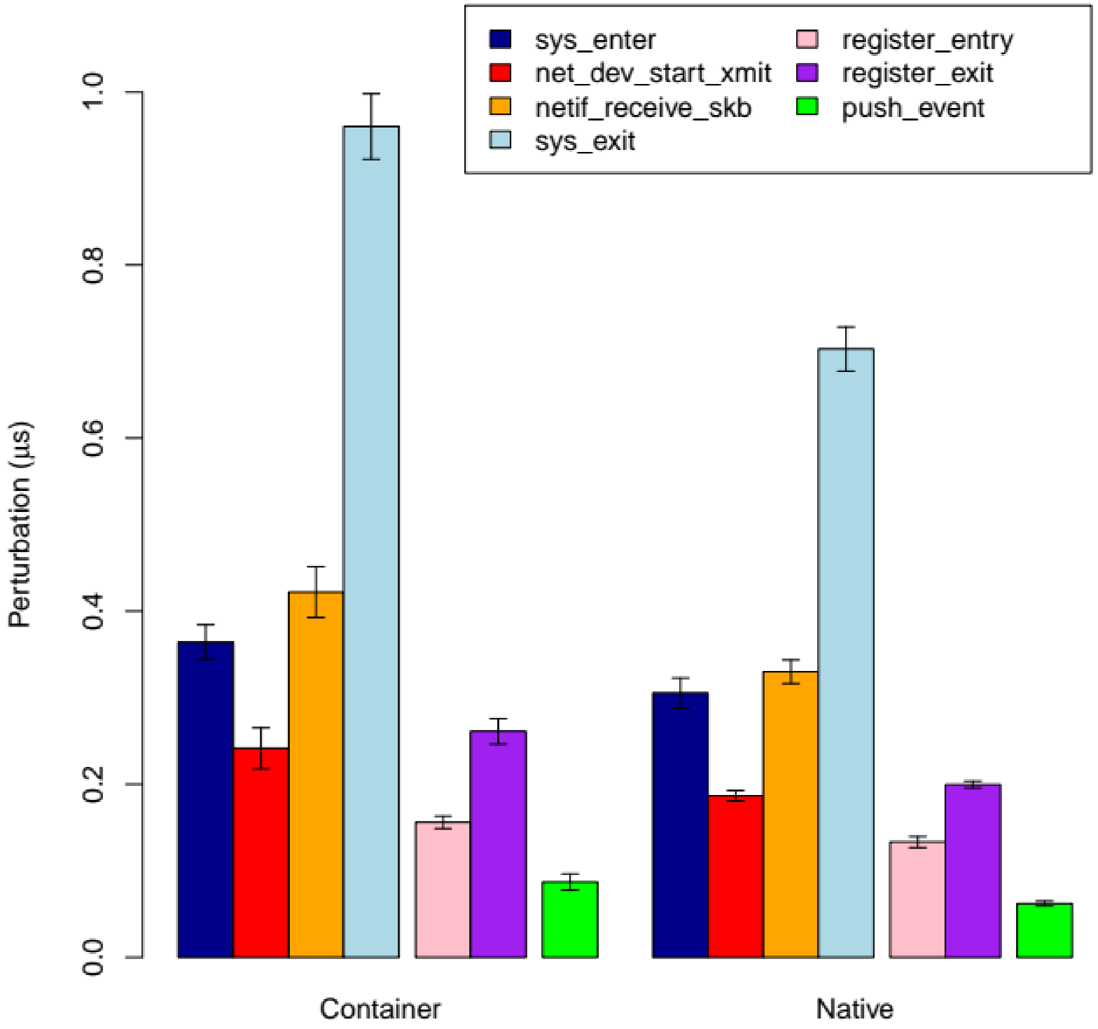
  minus non-instrumented RTT
  - MACE (black)
  - Ftrace (blue)
- MACE **scales well** as traffic increases

Results: MACE Functions



- Execution time of MACE functions
  - Tracepoint probes
  - Hash table management
  - Latency list management

Results: MACE Functions



- Execution time of MACE functions
  - Tracepoint probes
  - Hash table management
  - Latency list management
- System call tracepoints are slow
  - Accessing data in userspace
  - Needed for correlation

Future Goals

- Improving MACE
  - Add TCP, UDP support
  - Hardware timestamps
  - Better in-flight correlation
  - Ease of application-level correlation

Future Goals

- Improving MACE
  - Add TCP, UDP support
  - Hardware timestamps
  - Better in-flight correlation
  - Ease of application-level correlation
- Applying MACE
  - Improving measurement accuracy (e.g. geolocation)
  - Virtual network telemetry

Summary

- Containerized measurement issues
- Proposed solution: MACE
- Evaluation of MACE

Thank You!

- UO VPRI* and NSF
- Anonymous reviewers
- CloudLab team

* This work is supported by a fellowship from the University of Oregon Office of the Vice President for Research and Innovation.

cmisa@cs.uoregon.edu

Questions?

Citations

- [0] https://planet-lab.org/node/263, Sept. 2012 (accessed Feb. 2019)
- [1] W. Felter et al., “An updated performance comparison of virtual machines and linux containers.” *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software*, 2015.
- [2] Y. Zhao et al., “Performance of container networking technologies.” *Proceedings of HotConNet* 2017.
- [3] D. Zhuo et al., “Slim: OS Kernel Support for a Low-Overhead Container Overlay Network.” *NSDI* 2019.
- [4] D. Kim et al., “Freeflow: Software-based Virtual RDMA Networking for Containerized Clouds.” *NSDI* 2019.
- [5] N. Shalom and Y. Einav, “Amazon Found Every 100ms of Latency Cost Them 1% in Sales.” http://goo.gl/BUJgV (accessed Feb. 2019).
- [6] https://www.britannica.com/science/speed-of-light (accessed Feb. 2019).
- [7] A. Singla et al., “The Internet at the speed of light.” *Proceedings of HotNets*, October 2014.
- [8] M. Mathis and M. Allman, “A Framework for Defining Empirical Bulk Transfer Capacity Metrics.” RFC 3148, July 2001.
- [9] M. Desnoyers, “Using the Linux Kernel Tracepoints.” https://www.kernel.org/doc/Documentation/trace/tracepoints.txt (accessed Feb. 2019)
- [10] R. Ricci et al.,“Introducing CloudLab: Scientific infrastructure for advancing cloud architectures and applications.” *;login:*, 2014.