Reproducible Network Research With High-Fidelity Emulation

Nikhil Handigol[+], **Brandon Heller**[+], Bob Lantz[*], Vimal Jeyakumar[+], Nick McKeown[+]

[+]Stanford University, Palo Alto, USA

[*]Open Networking Laboratory, Palo Alto, USA

The scientific method says:

experiments are only valid if they can be *reproduced.*

The norm in physics, medicine, etc..

But what about the computational sciences?

D.L. Donoho, 1995:

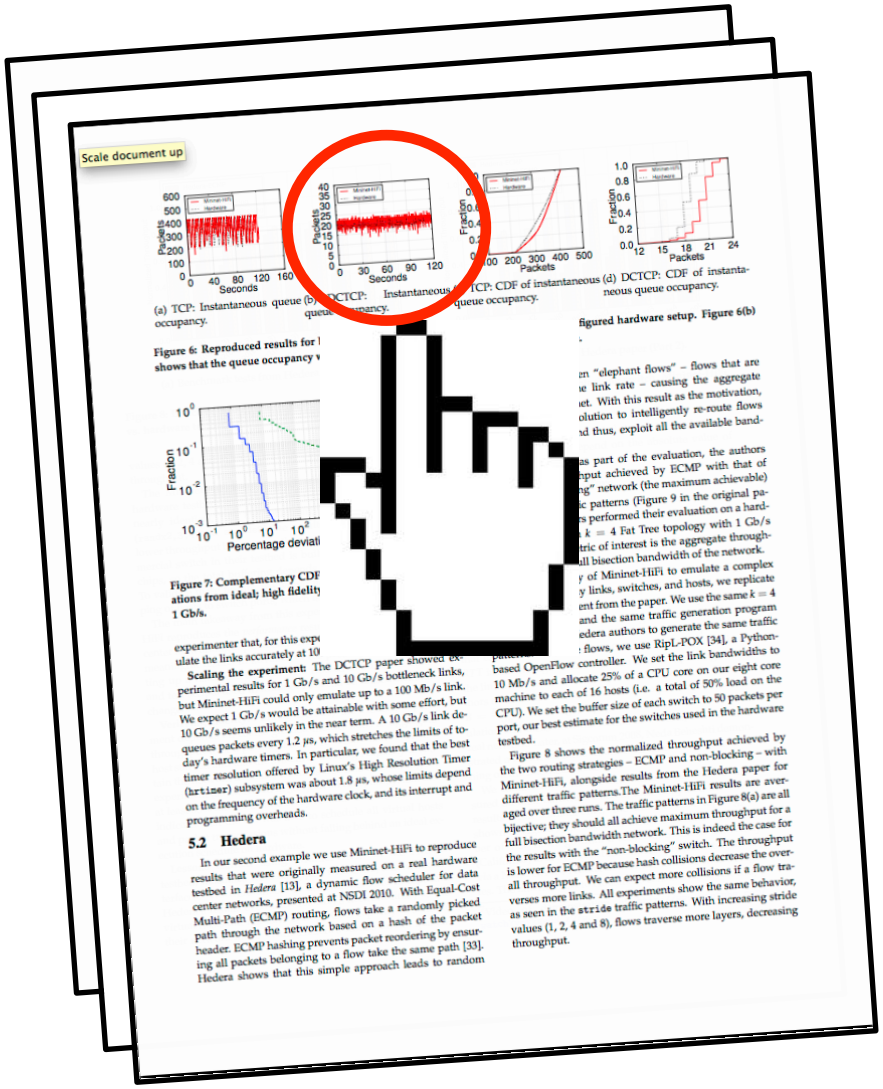
"An article about computational science is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures."

This is what network research should be.

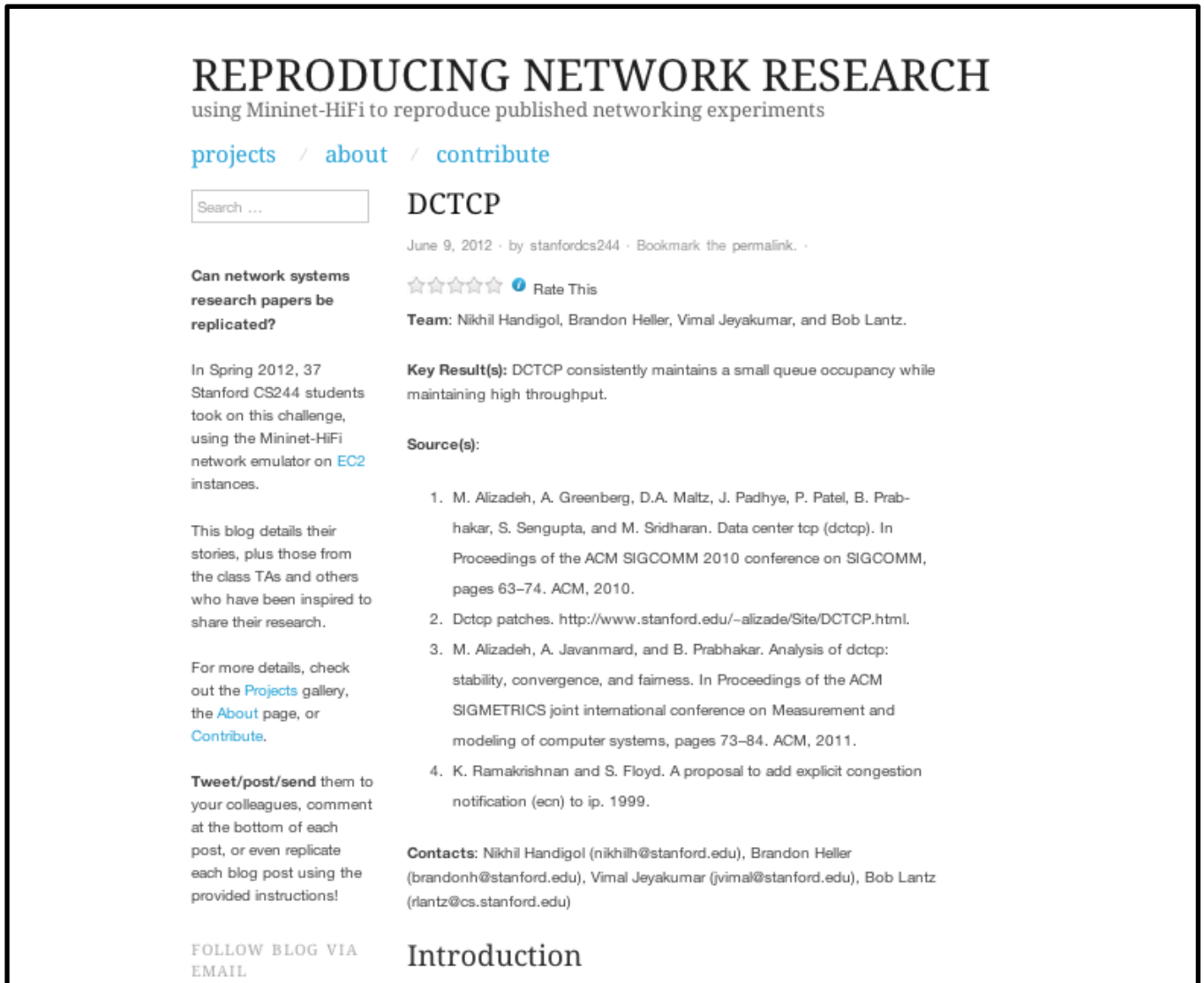
It should be reproducible.

Papers should be **runnable.**

Click on a figure



Brings up a webpage of instructions



REPRODUCING NETWORK RESEARCH

using Mininet-HiFi to reproduce published networking experiments

projects / about / contribute

Search ...

Can network systems research papers be replicated?

In Spring 2012, 37 Stanford CS244 students took on this challenge, using the Mininet-HiFi network emulator on EC2 instances.

This blog details their stories, plus those from the class TAs and others who have been inspired to share their research.

For more details, check out the Projects gallery, the About page, or Contribute.

Tweet/post/send them to your colleagues, comment at the bottom of each post, or even replicate each blog post using the provided instructions!

FOLLOW BLOG VIA EMAIL

DCTCP

June 9, 2012 · by stanfordcs244 · Bookmark the permalink. ·

Rate This

Team: Nikhil Handigol, Brandon Heller, Vimal Jeyakumar, and Bob Lantz.

Key Result(s): DCTCP consistently maintains a small queue occupancy while maintaining high throughput.

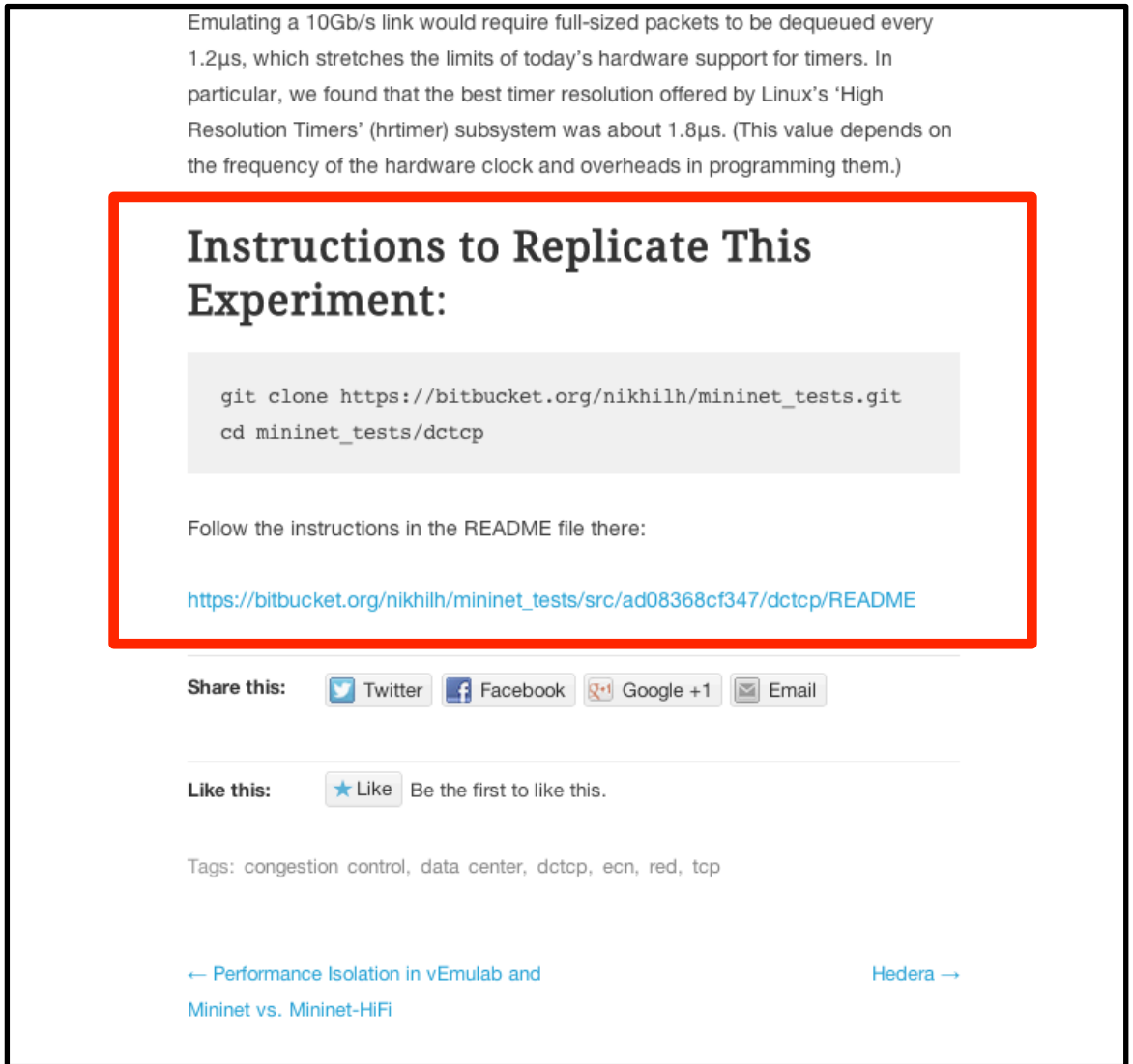
Source(s):

1. M. Alizadeh, A. Greenberg, D.A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. Data center tcp (dctcp). In Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM, pages 63–74. ACM, 2010.
2. Dctcp patches. http://www.stanford.edu/~alizade/Site/DCTCP.html.
3. M. Alizadeh, A. Javanmard, and B. Prabhakar. Analysis of dctcp: stability, convergence, and fairness. In Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems, pages 73–84. ACM, 2011.
4. K. Ramakrishnan and S. Floyd. A proposal to add explicit congestion notification (ecn) to ip. 1999.

Contacts: Nikhil Handigol (nikhilh@stanford.edu), Brandon Heller (brandonh@stanford.edu), Vimal Jeyakumar (jvimal@stanford.edu), Bob Lantz (rlantz@cs.stanford.edu)

Introduction

Scroll to the bottom…



Emulating a 10Gb/s link would require full-sized packets to be dequeued every 1.2µs, which stretches the limits of today's hardware support for timers. In particular, we found that the best timer resolution offered by Linux's 'High Resolution Timers' (hrtimer) subsystem was about 1.8µs. (This value depends on the frequency of the hardware clock and overheads in programming them.)

Instructions to Replicate This Experiment:

```
git clone https://bitbucket.org/nikhilh/mininet_tests.git
cd mininet_tests/dctcp
```

Follow the instructions in the README file there:

https://bitbucket.org/nikhilh/mininet_tests/src/ad08368cf347/dctcp/README

Share this: Twitter Facebook Google +1 Email

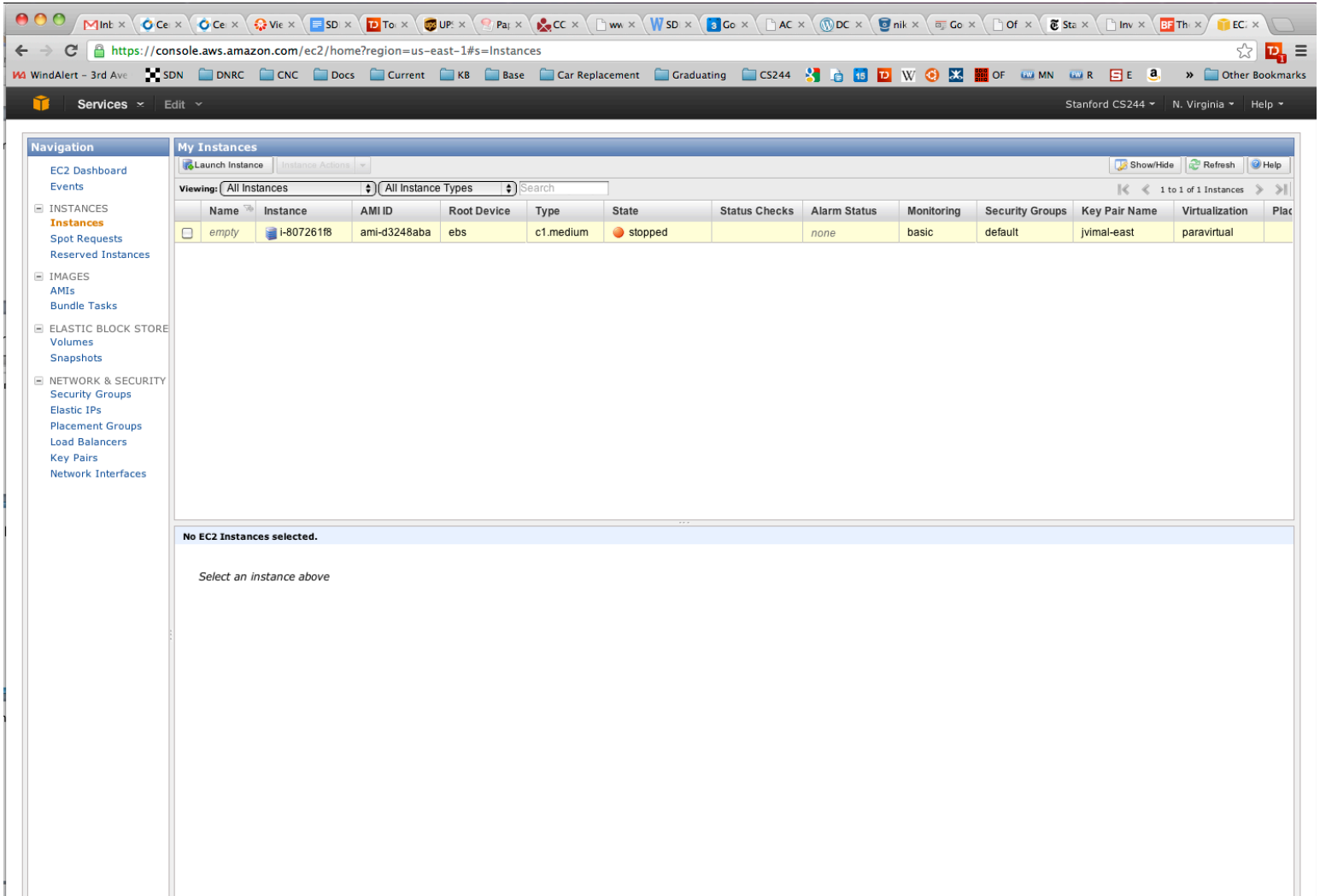
Like this: ★ Like Be the first to like this.

Tags: congestion control, data center, dctcp, ecn, red, tcp

← Performance Isolation in vEmulab and Mininet vs. Mininet-HiFi

Hedera →

Launch an EC2 instance

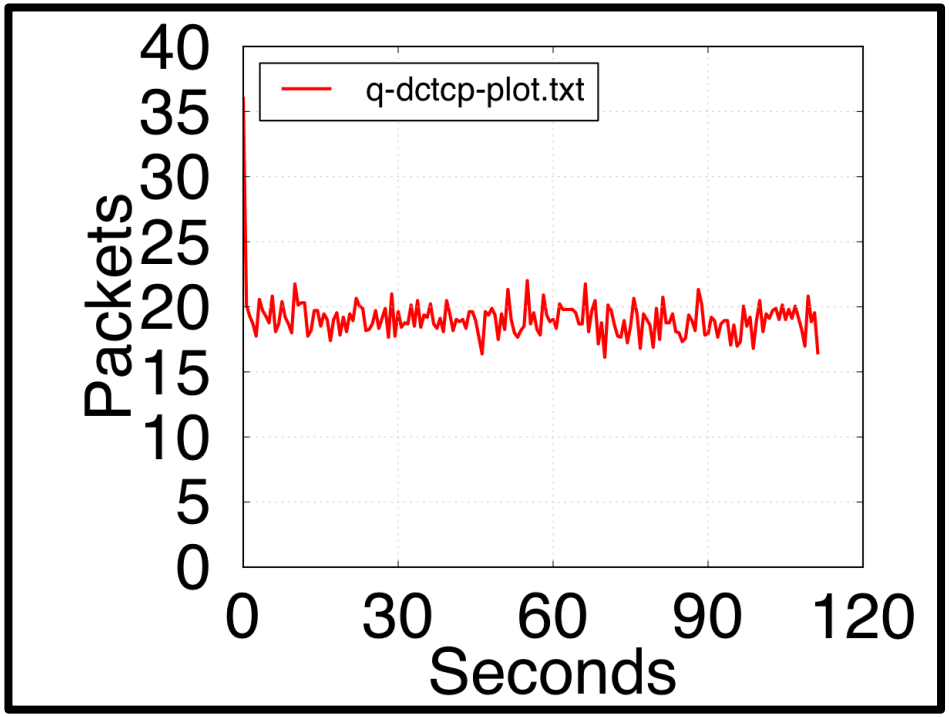


Run a command in the terminal to generate results

```
> ~/mininet-tests/dctcp$ ./run-dctcp.sh
```

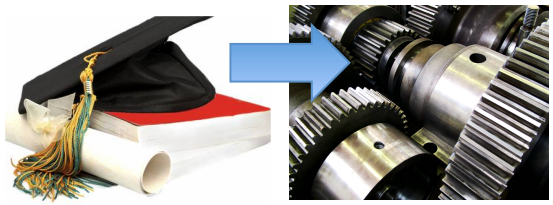
8 minutes & 8 cents after the click: a reproduced result

```
> ~/mininet-tests/dctcp$ ./run-dctcp.sh
> ~/mininet-tests/dctcp/results$ xpdf dctcp.pdf
```

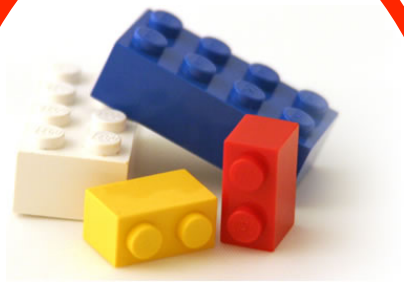


If papers were runnable (easily reproducible):

easier to understand
and evaluate papers



easier to transfer
new ideas to industry

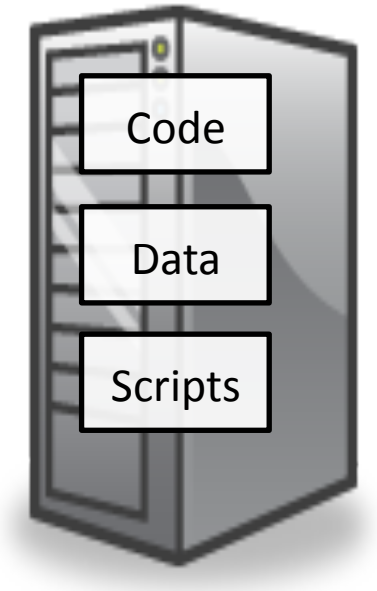


easier to build upon
the work of others

Why aren’t all networking research papers like this?

Much of CS

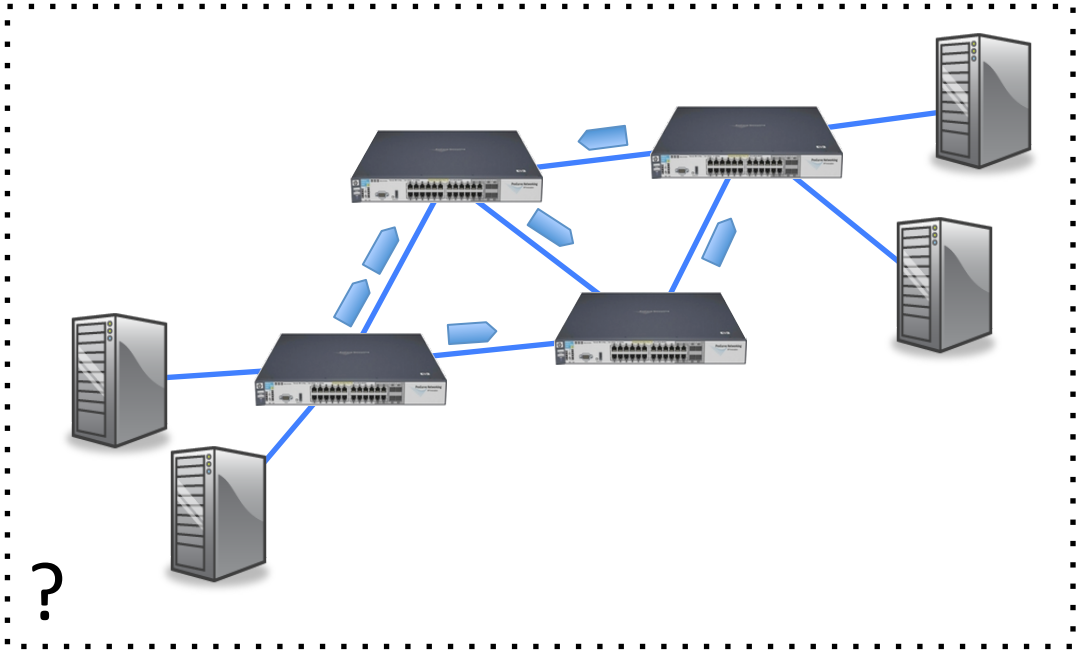
ML, PL, Security, …



Use any commodity PC
(or VM.)

Network Systems

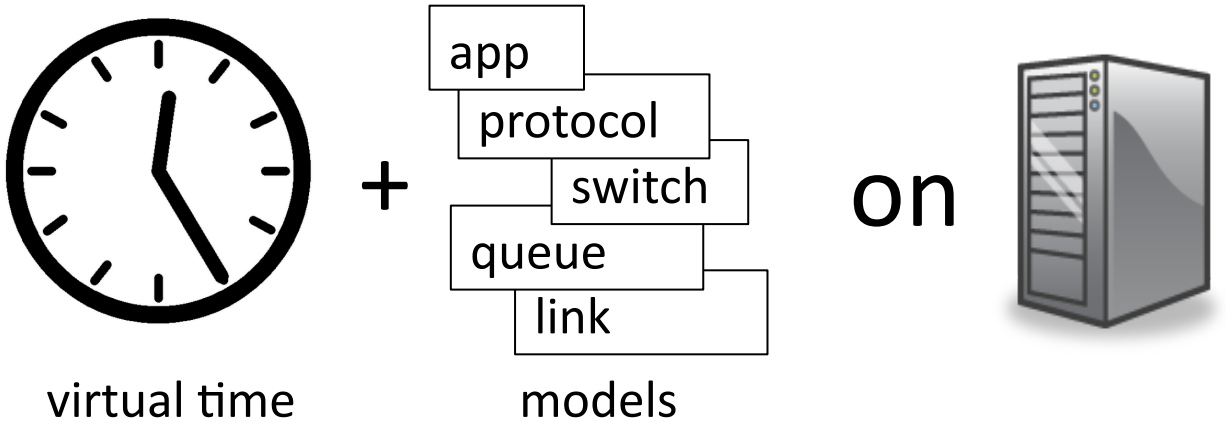
Congestion control, Routing, Topology, …



Must implement **multiple**
servers, network elements, links –
all running in **parallel,**
all with **accurate timing**

Discrete-Event Simulation

ns-2, ns-3, OPNET, …

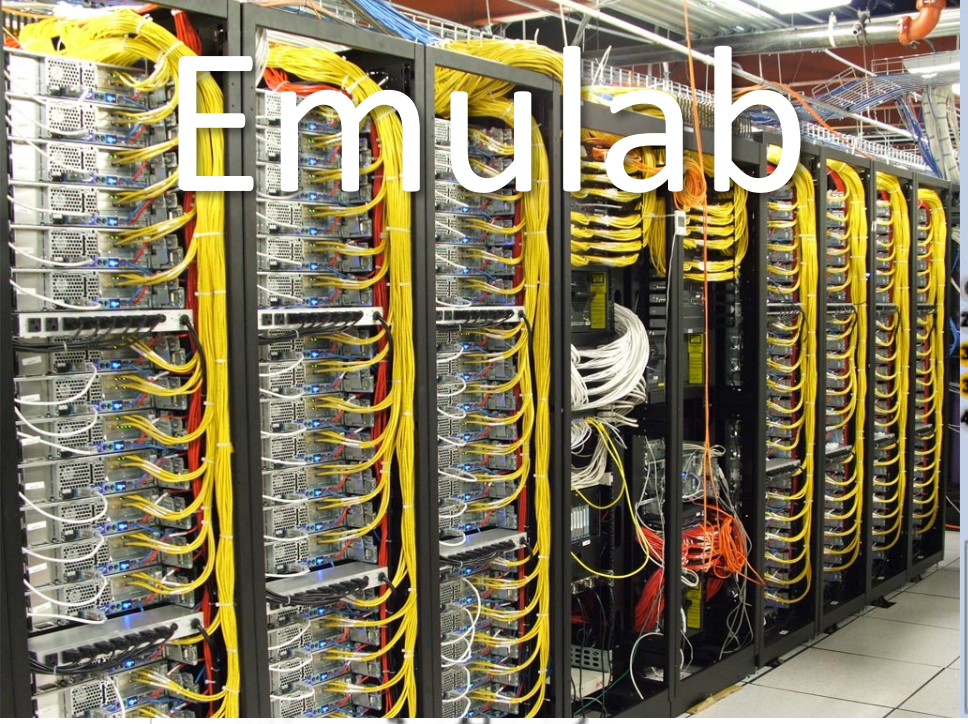


But… we don't trust simulation.

Not *believable* unless validated:

- Modeled links == real links
- Modeled switches == real switches
- Modeled protocols == real protocols
- Modeled applications == real applications

→ **Realism concerns.**



Shared Testbeds

Testbed results can be hard to (re)produce.

Flexibility

- Topology restrictions
- May not be able to change firmware

Resource availability

- before a deadline?
- one year later?

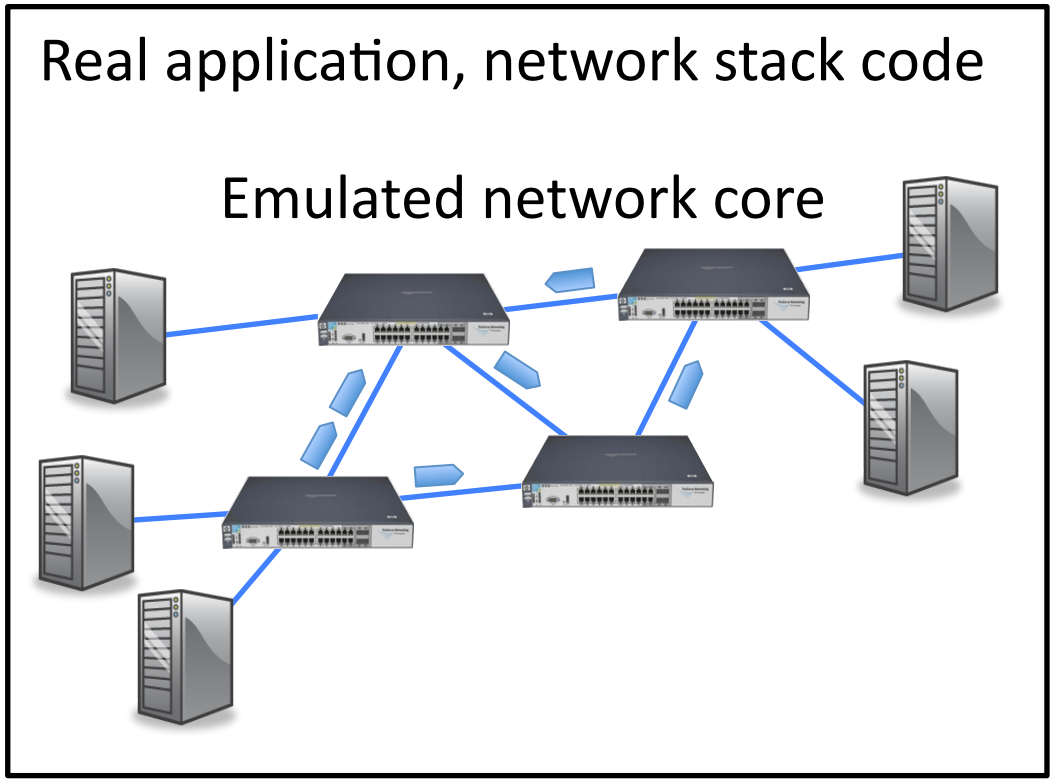
Problem

Network research tends not to be both *easily reproducible* and *realistic*.

Solution

Build an emulator whose results you can trust as well as verify.

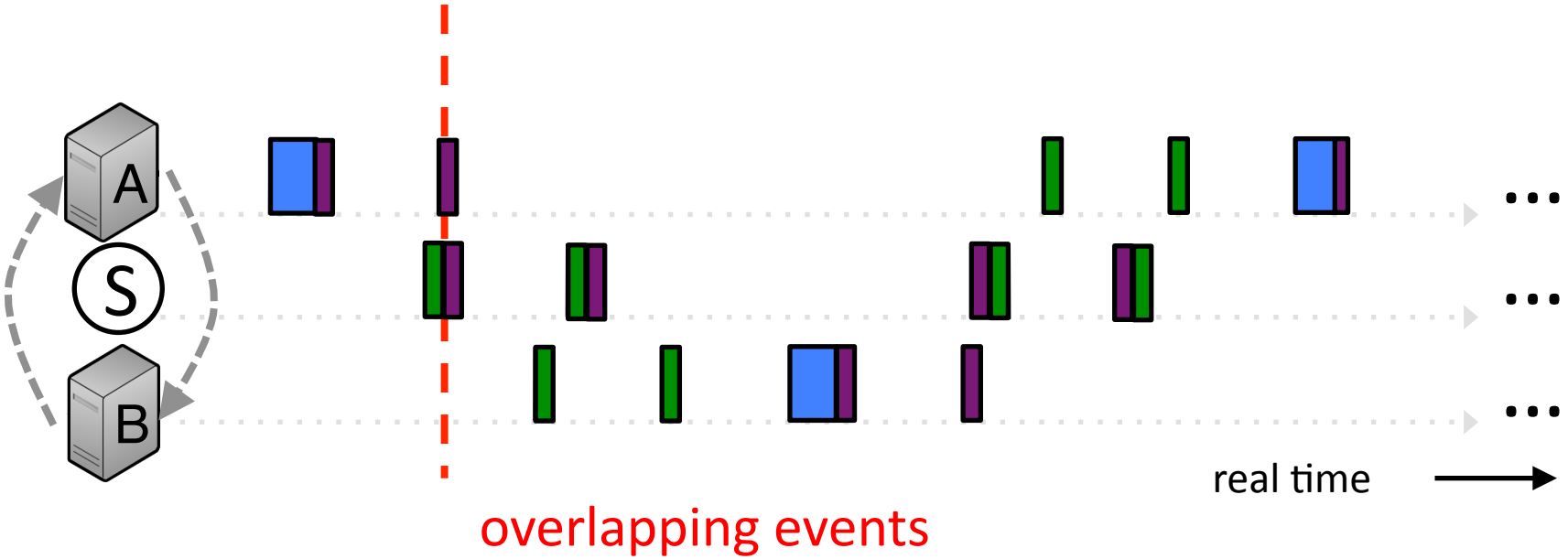
What is a network emulator?



Matching the behavior of hardware:
High Fidelity

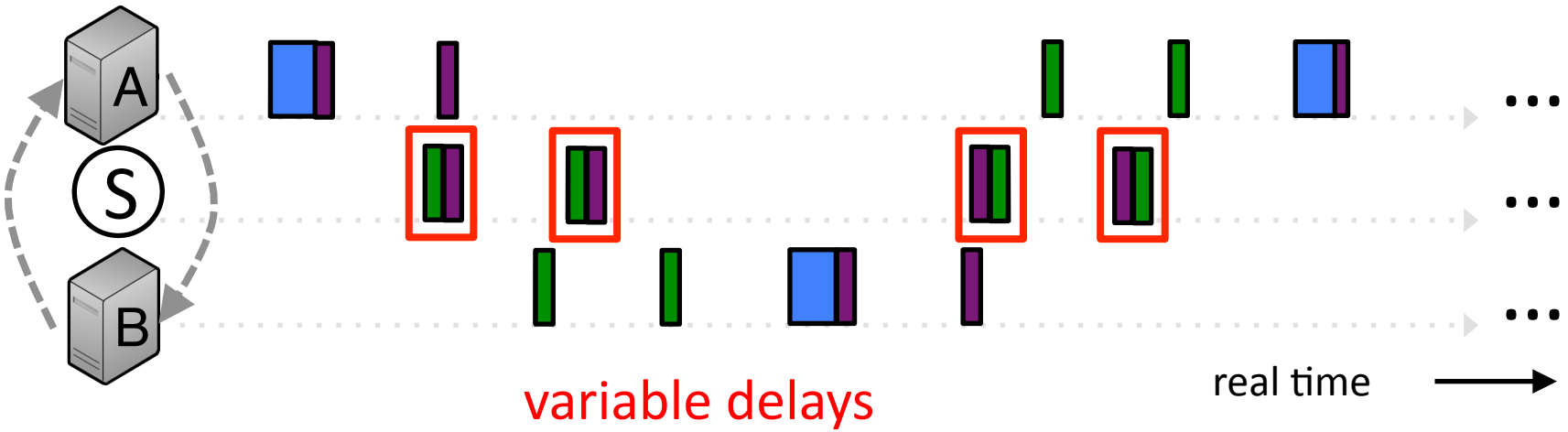
Sources of Emulator Infidelity

Event Overlap

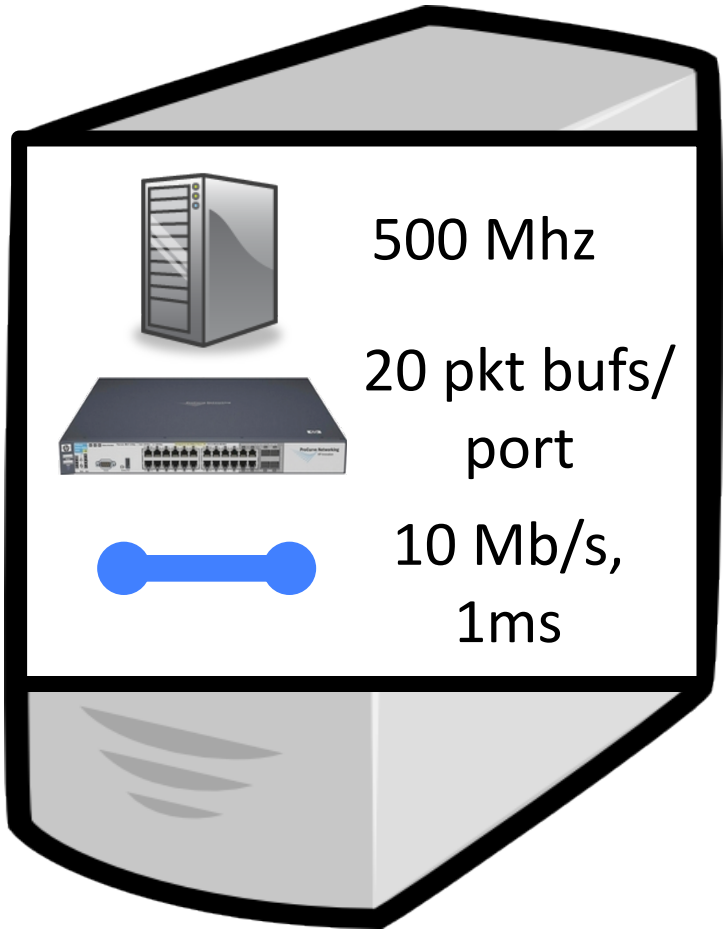


Sources of Emulator Infidelity

Software Forwarding



Our Approach



+

Resource-Isolating Emulator (Mininet-HiFi)

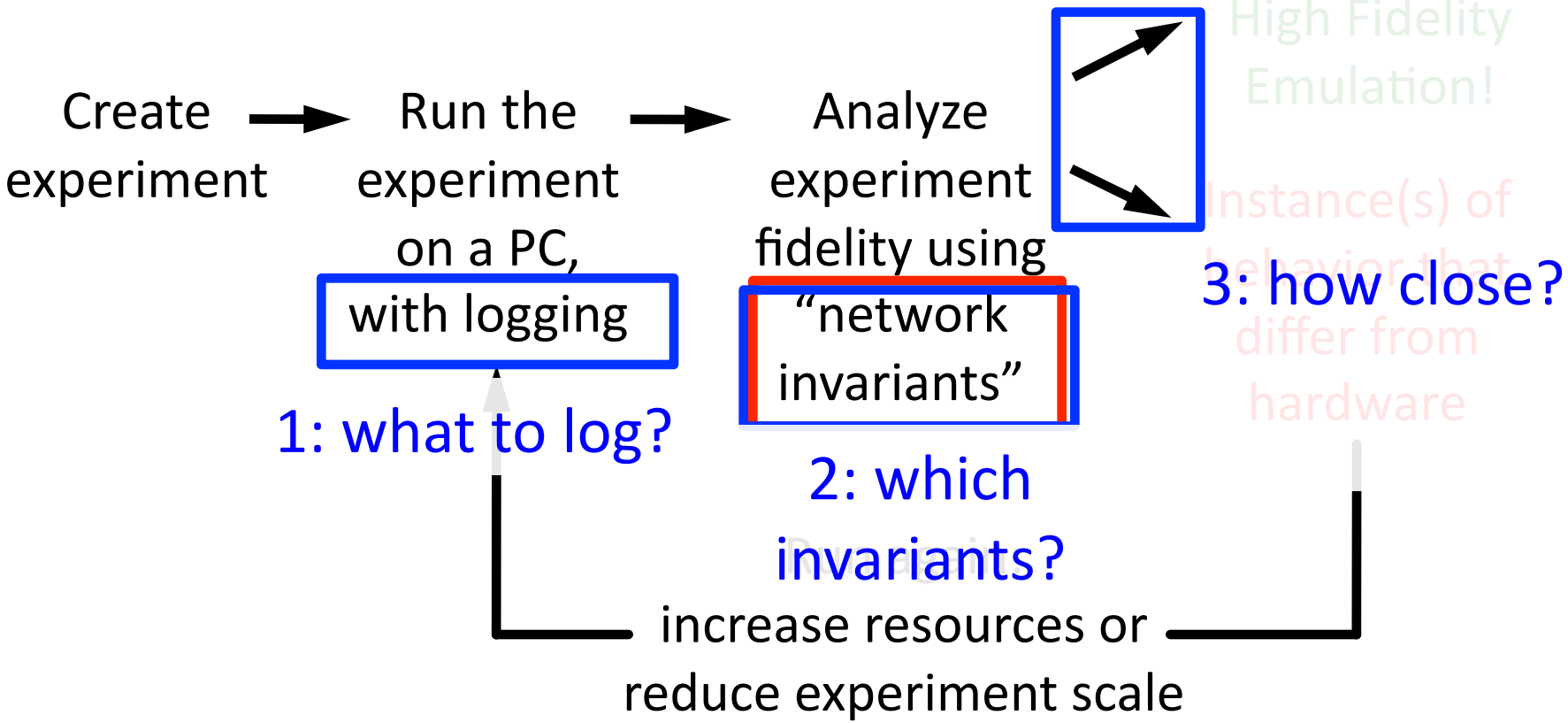
Fidelity Monitor

Talk Outline

- ~~Motivation~~
- 1. Emulator Fidelity
- 2. Mininet-HiFi Architecture
- 3. Reproducing Research
- Related Work
- Progress Report

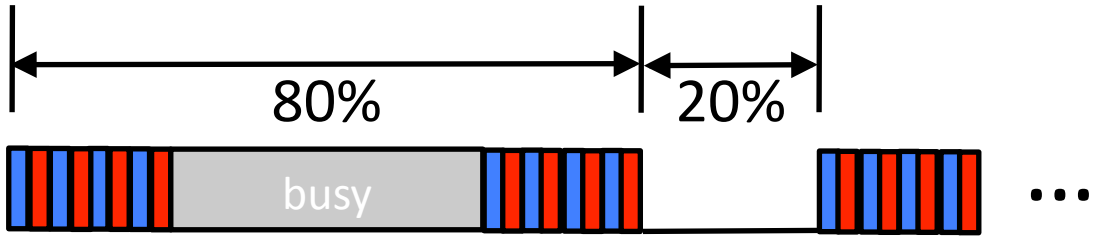
1.
Emulator Fidelity

A Workflow for High Fidelity Emulation



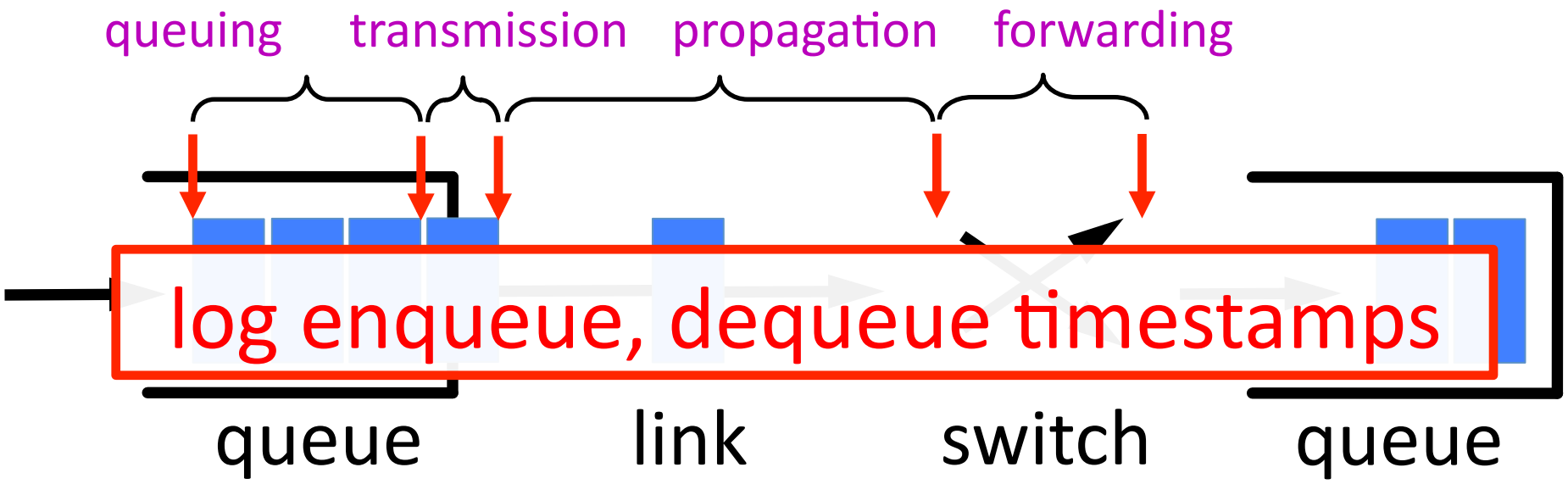
What to log?

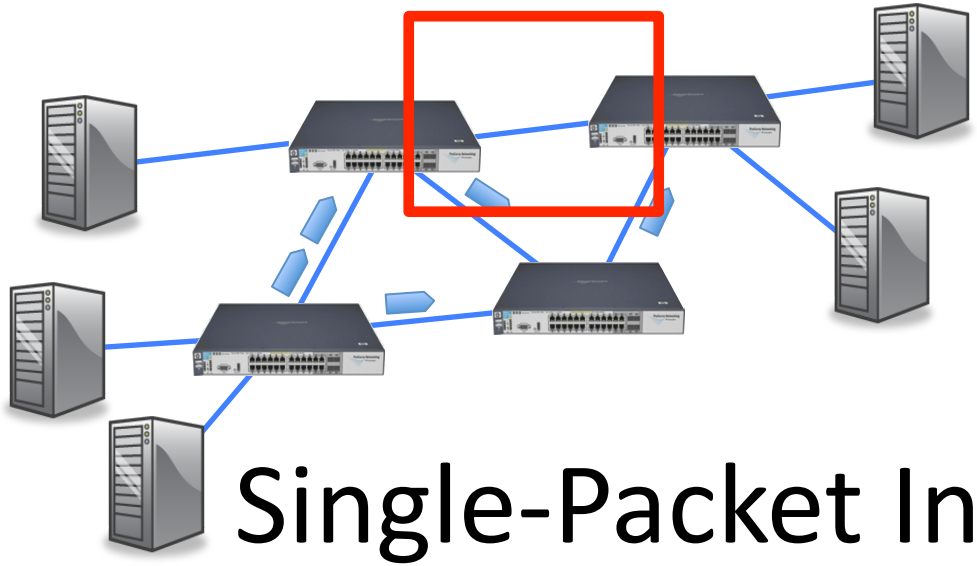
Consider logging utilization of the emulator CPU.
100% is bad.
But is X% (say, 80%) necessarily good?



Can't get back "lost time" in an emulator.

→ CPU utilization is insufficient.
Need to consider **fine-grained event timings.**





Single-Packet Invariants

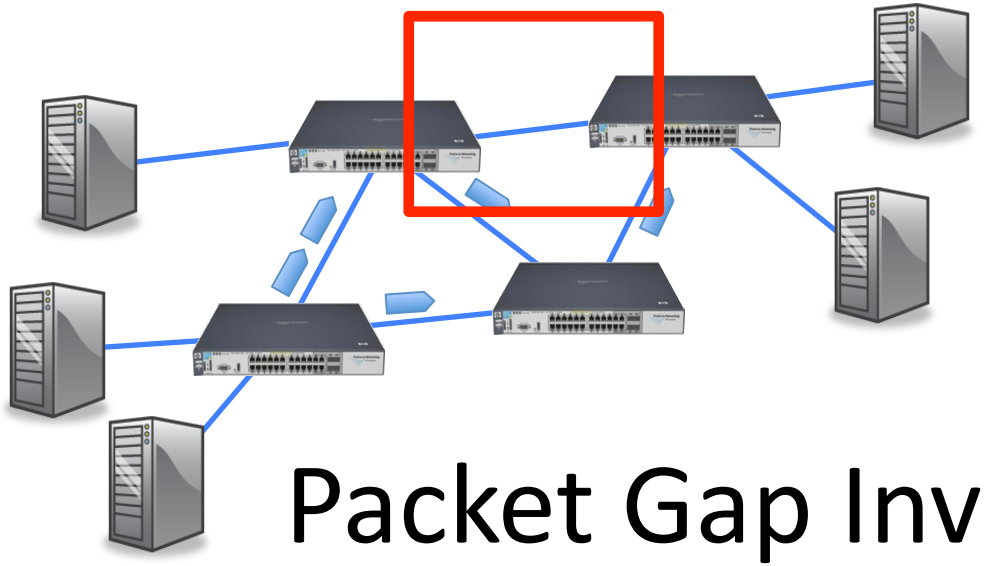
(when queue occupied)

packet spacing

link capacity $R_{measured} \leq R_{configured}$?

Explored in the paper

queue link switch queue



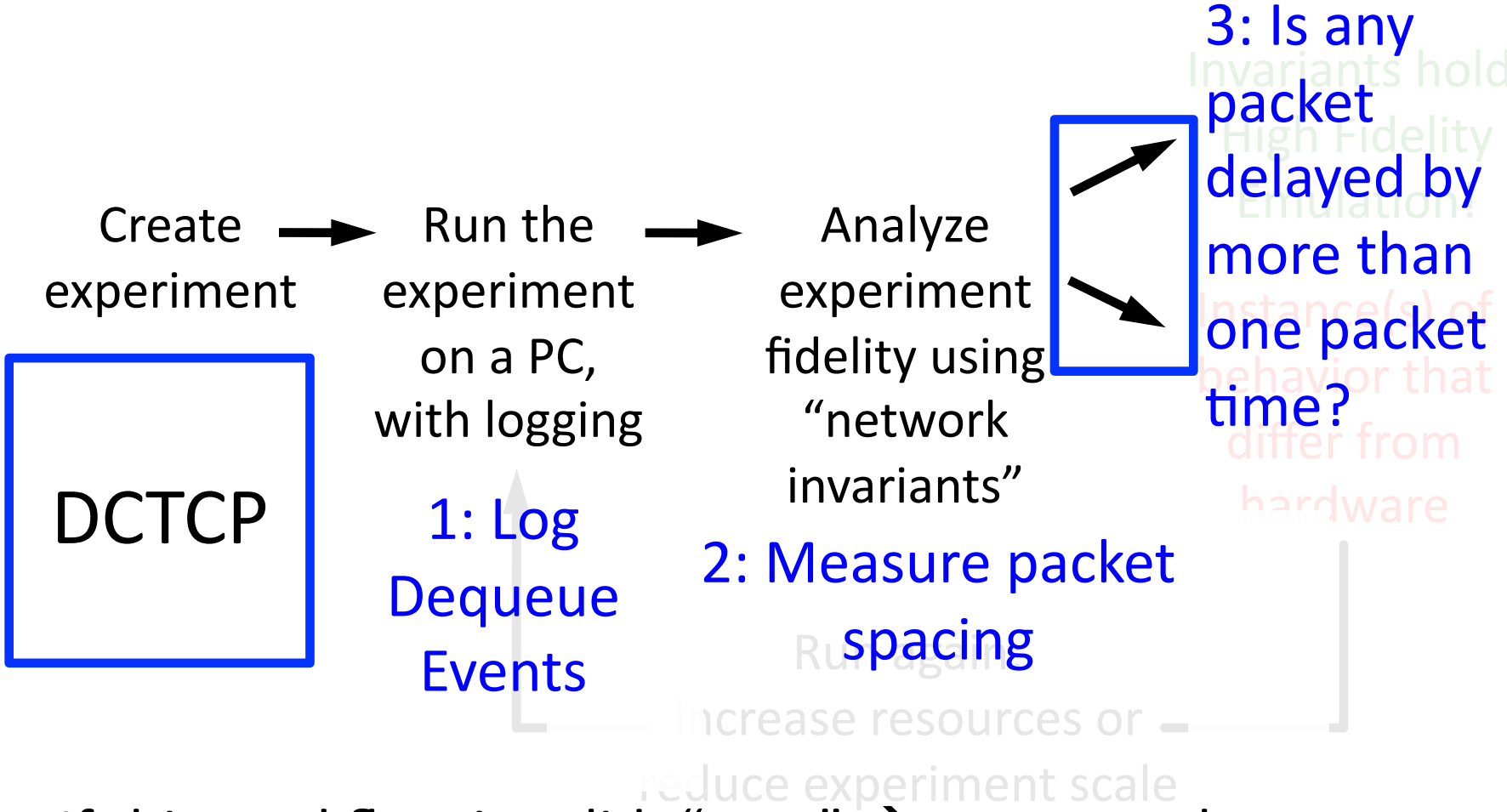
Packet Gap Invariants

How close?

High Fidelity: match hardware variability.

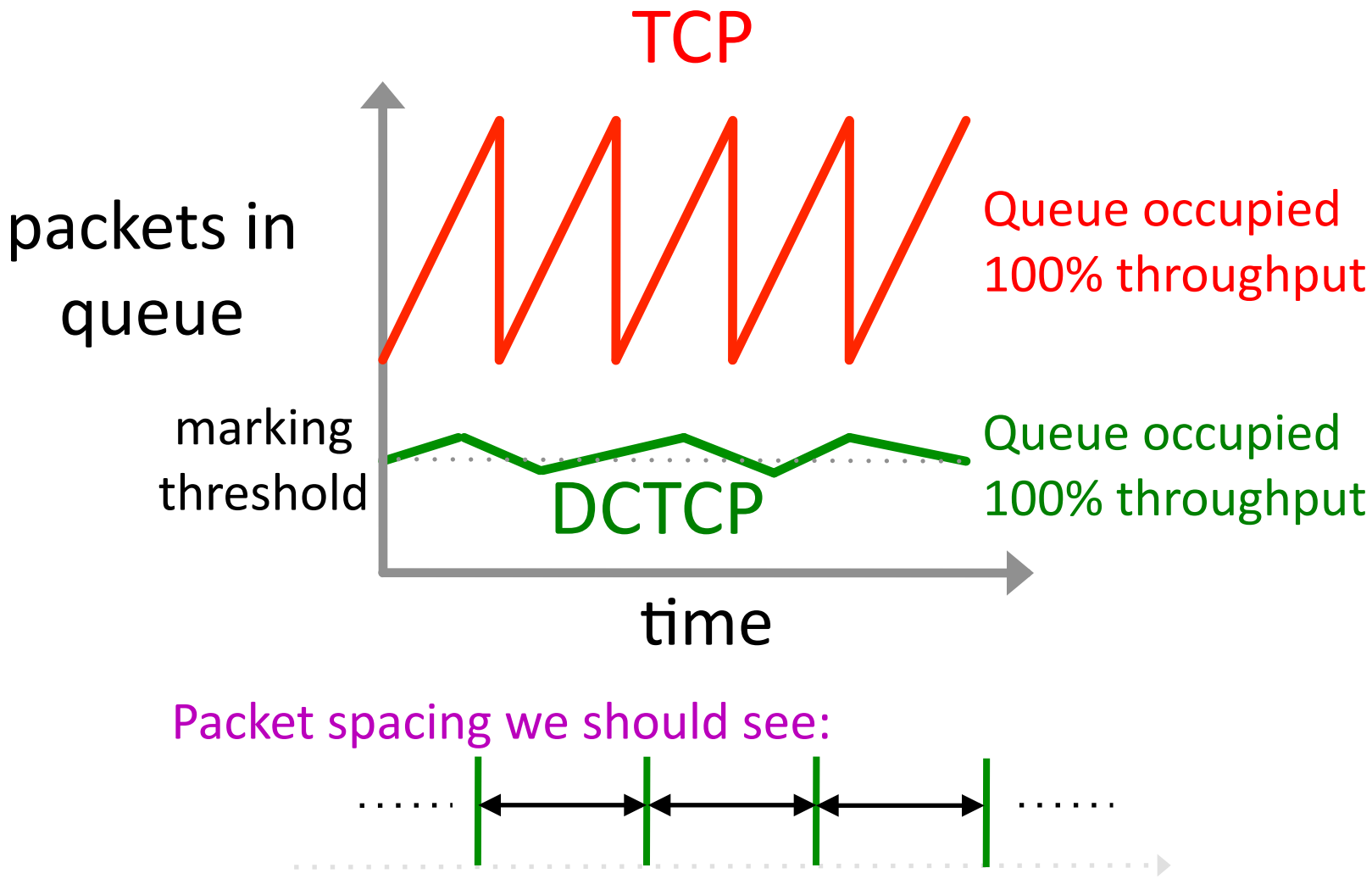
- Clock drift (== **one packet**)
- NIC to memory processing (~= 25 packets)
- Scheduler non-determinism (~= milliseconds)

Example Workflow for One Invariant



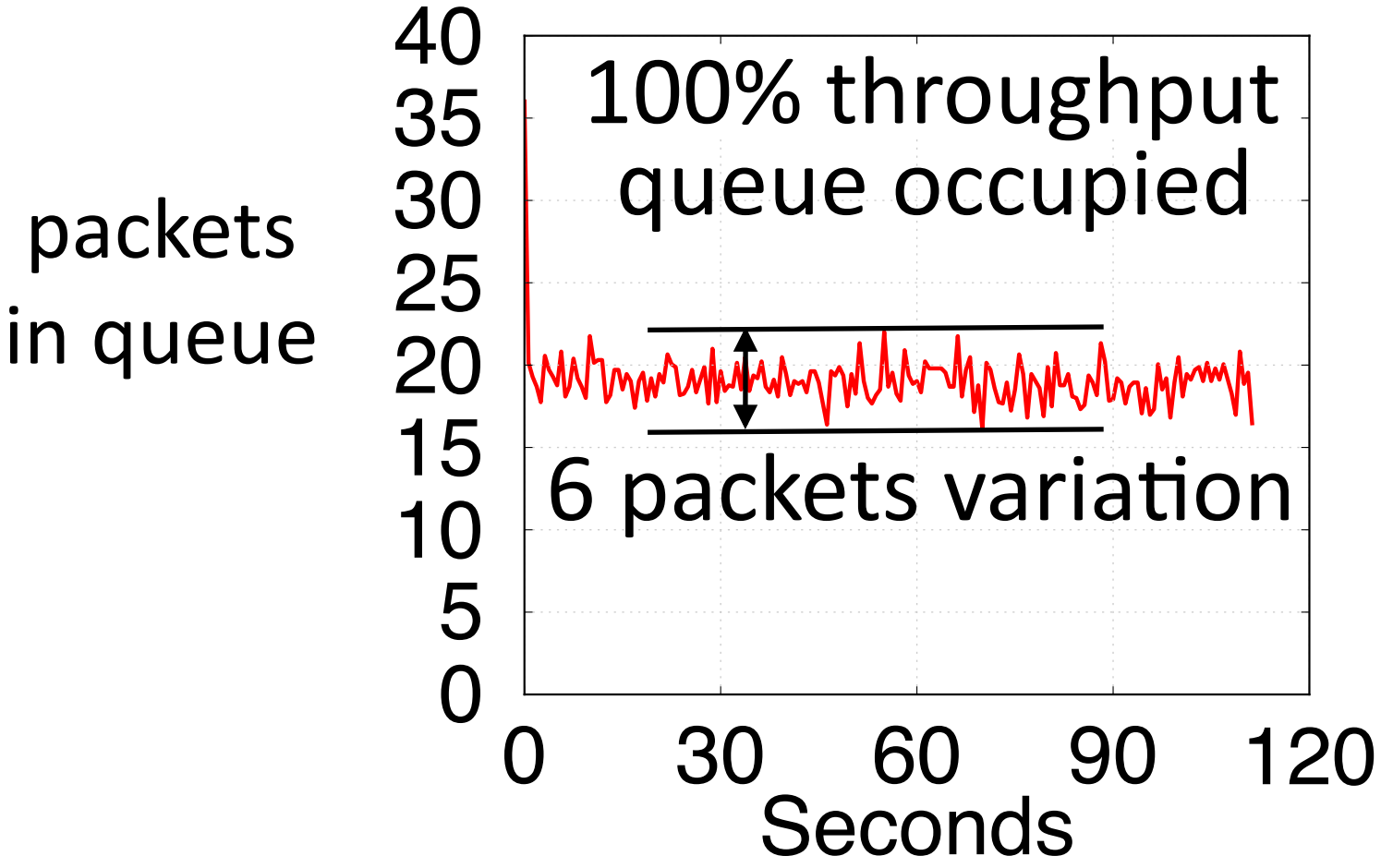
If this workflow is valid, "pass" → same result as hardware.

Data Center TCP (DCTCP)

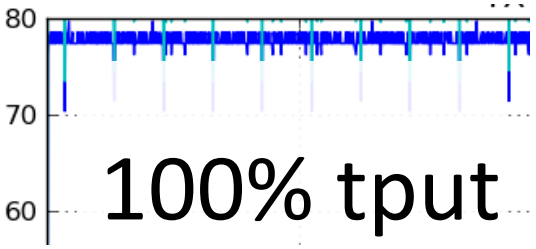
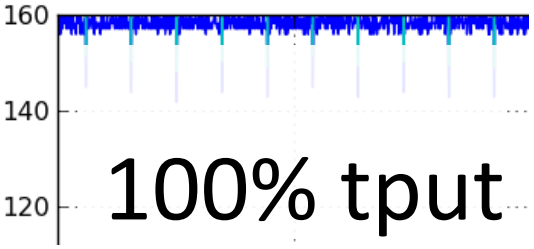
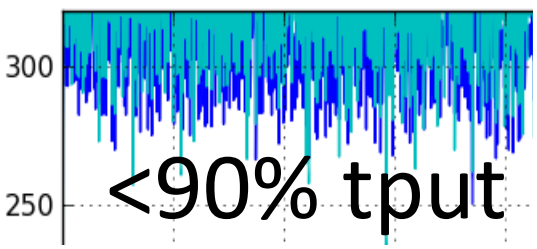
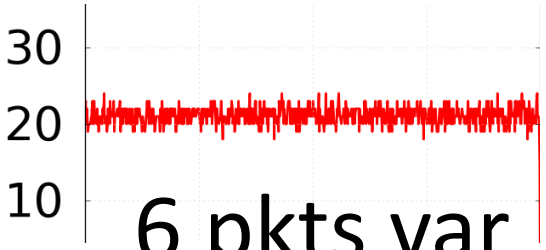
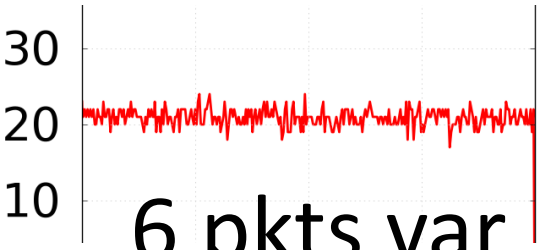
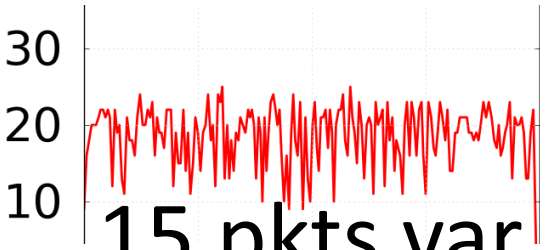


[Sigcomm10, Alizadeh et al.]

Hardware Results, 100 Mb/s

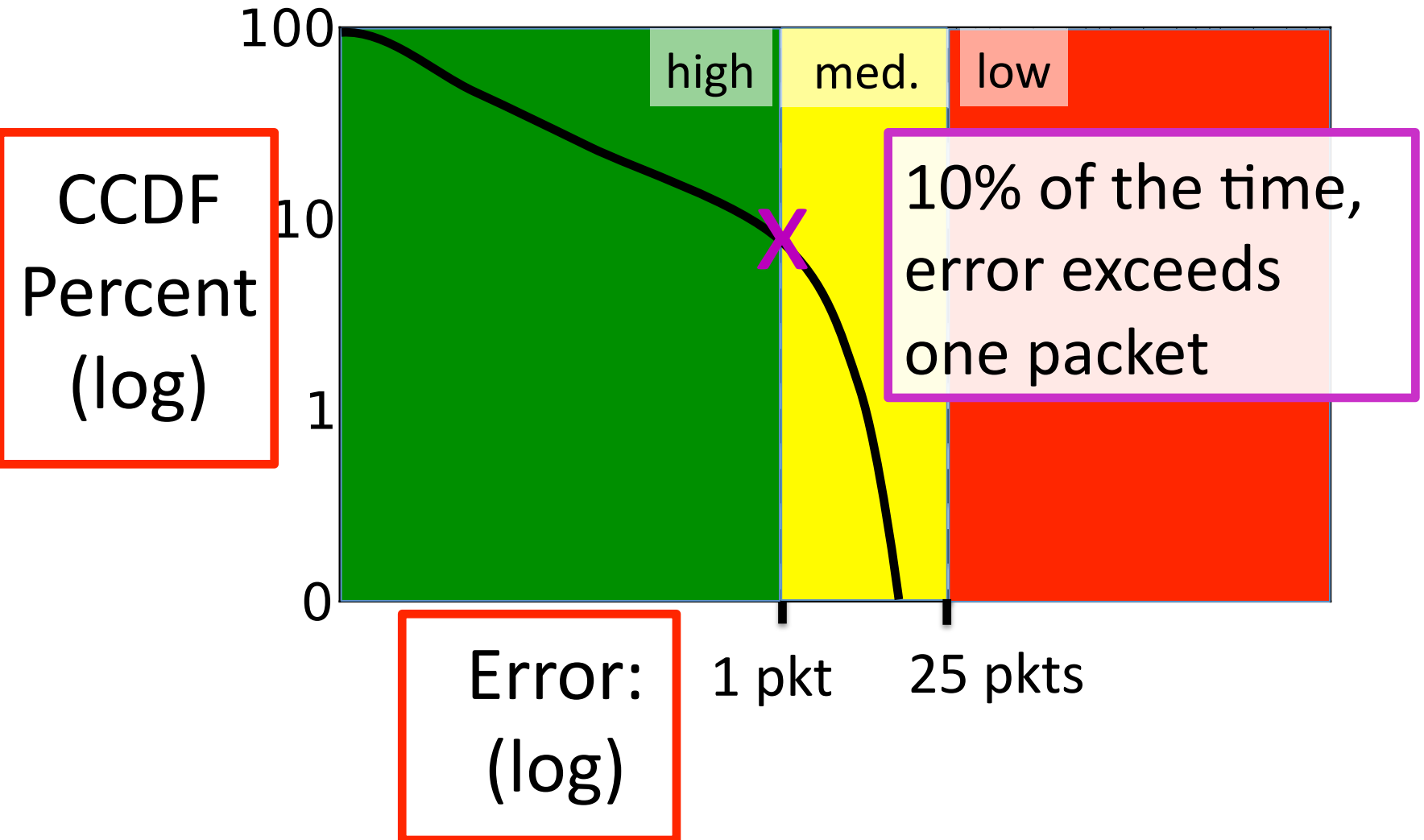


Emulator Results

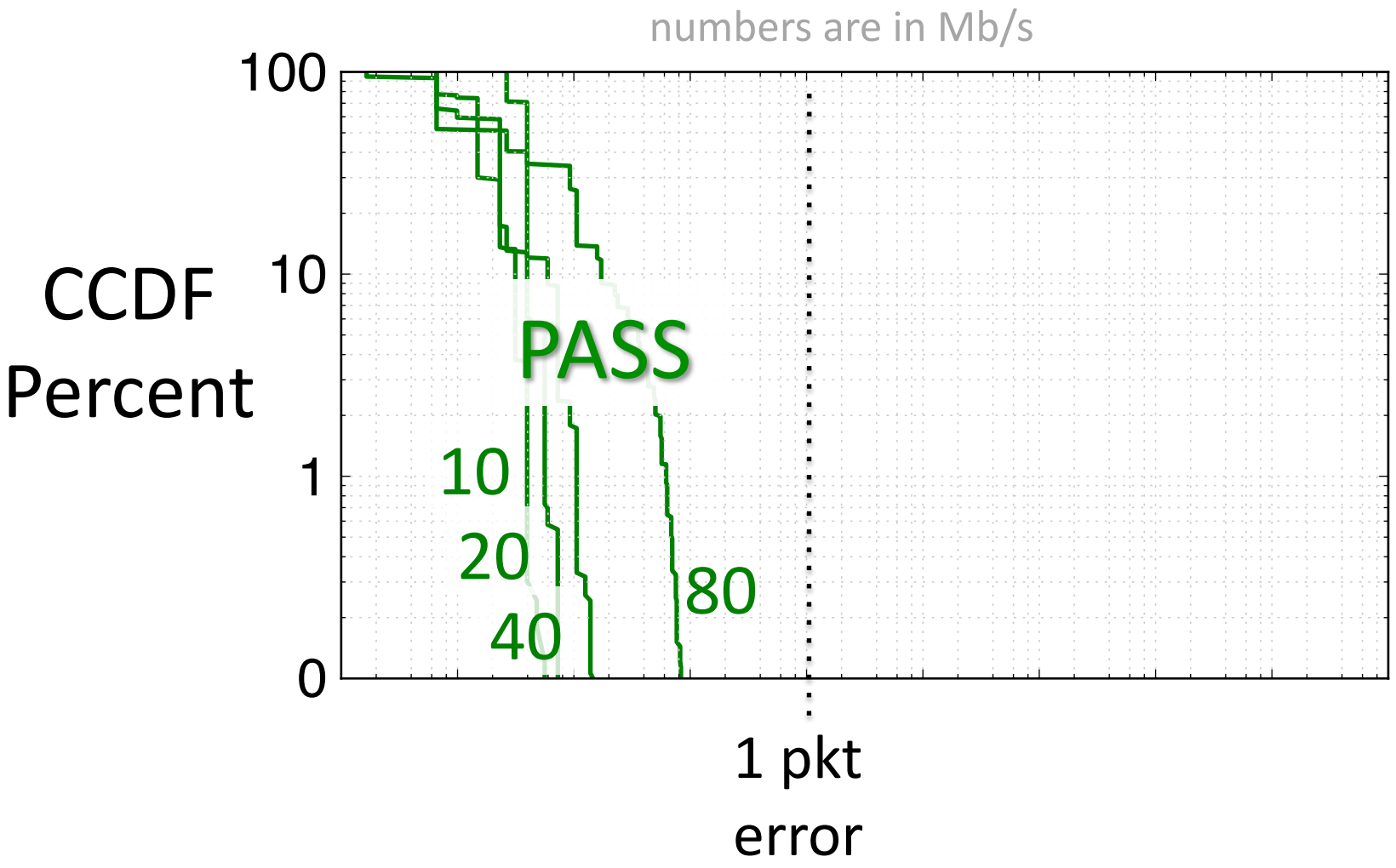
80 Mb/s	160 Mb/s	320 Mb/s
		
100% tput	100% tput	<90% tput
		
6 pkts var	6 pkts var	15 pkts var
same result	same result	wrong; limits exceeded

Does checking an invariant (packet spacing) identify wrong results?

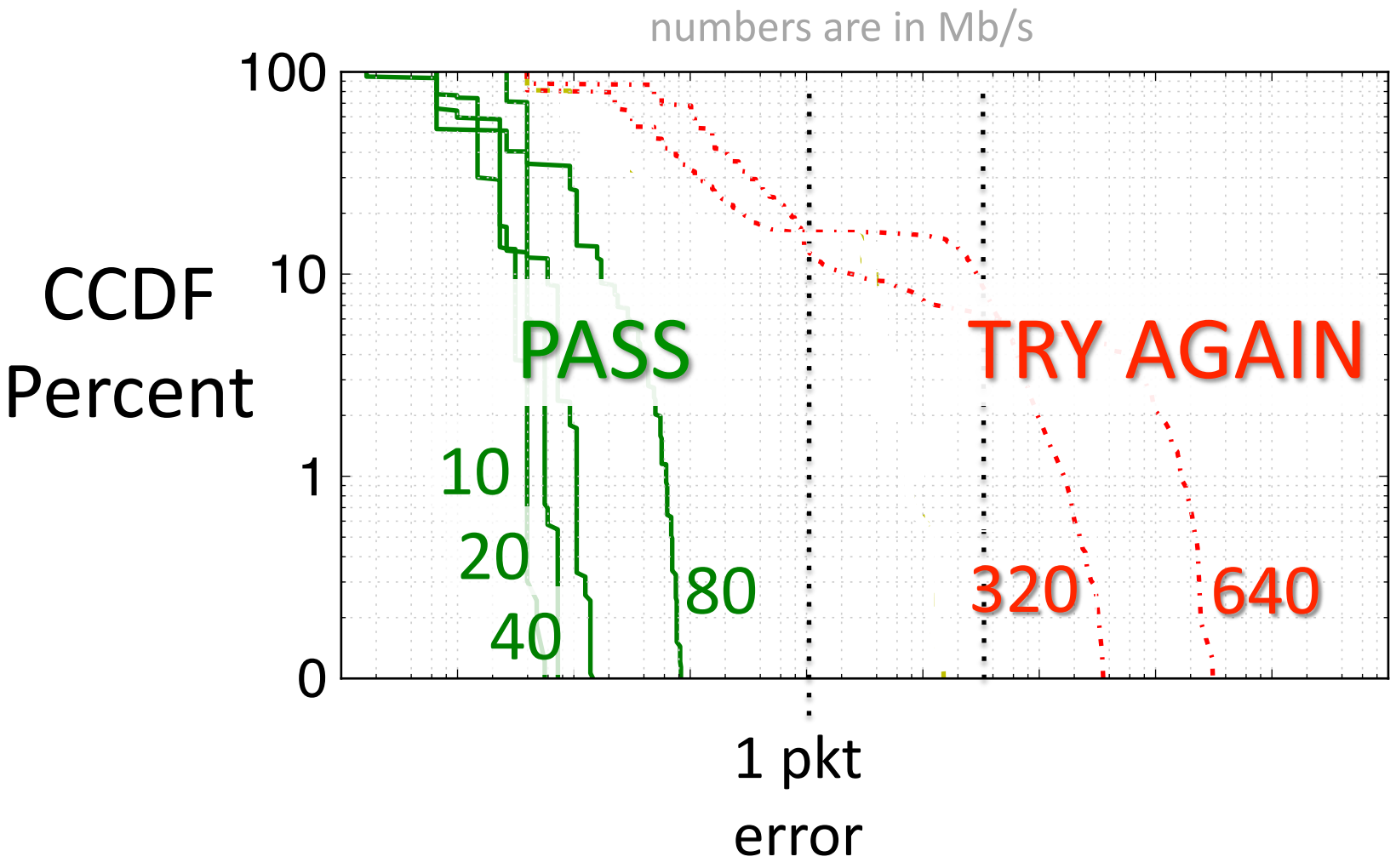
Packet Spacing Invariant w/DCTCP



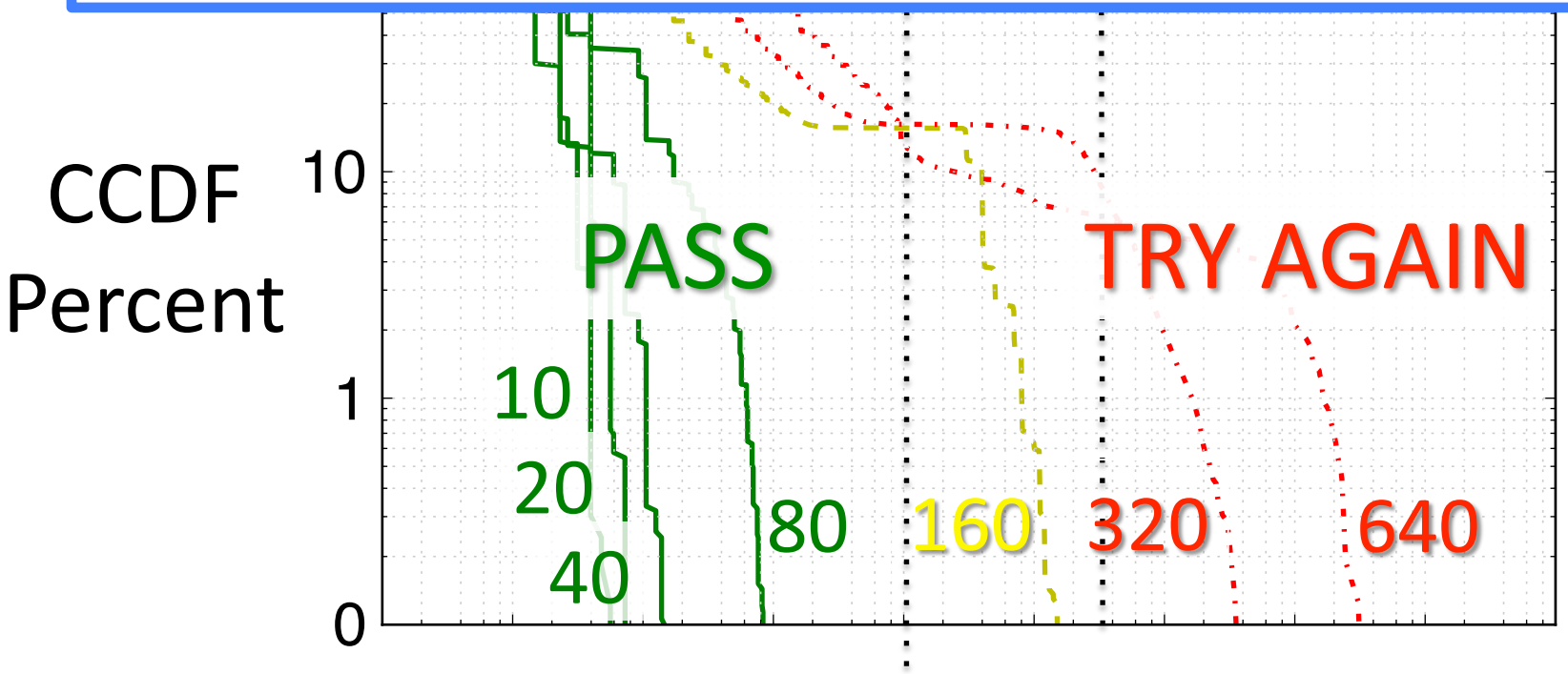
Packet Spacing Invariant w/DCTCP



Packet Spacing Invariant w/DCTCP



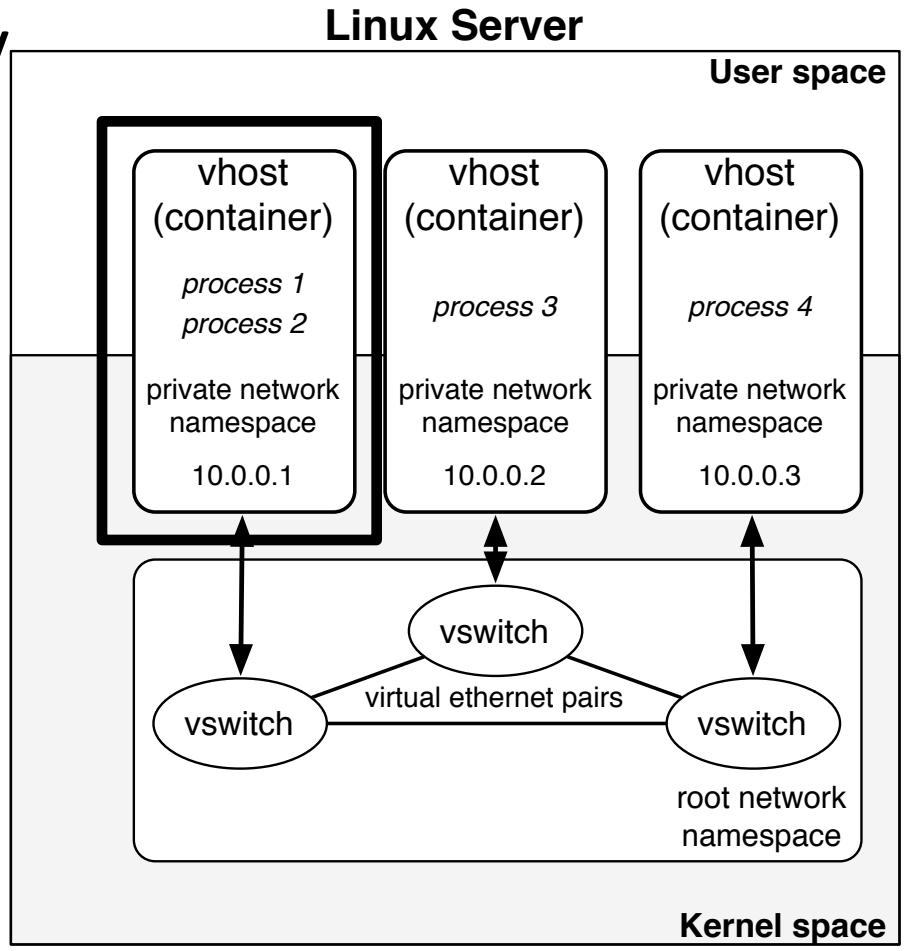
160 Mb/s: failed emulation?
Beauty of networks invariants is that it catches and quantifies the error in this run.



2: Mininet-HiFi Architecture

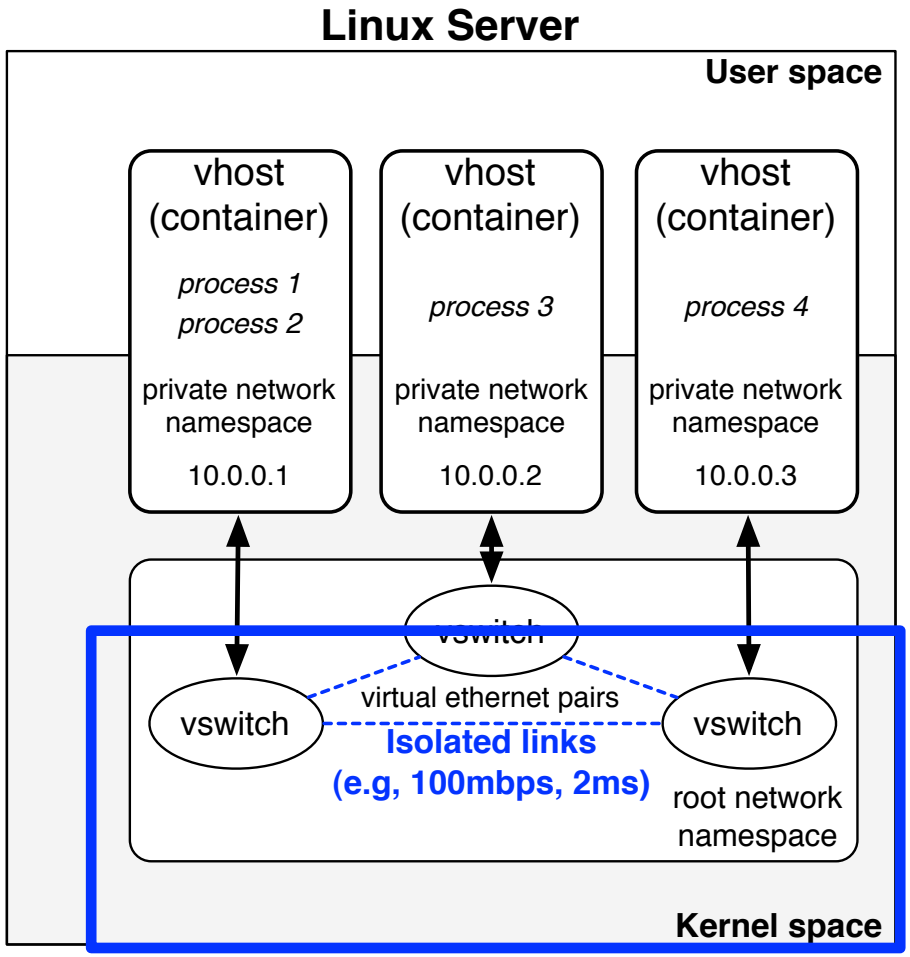
Original Mininet

Containers w/ Network Namespaces



Emulator

Emulator + Performance Isolation

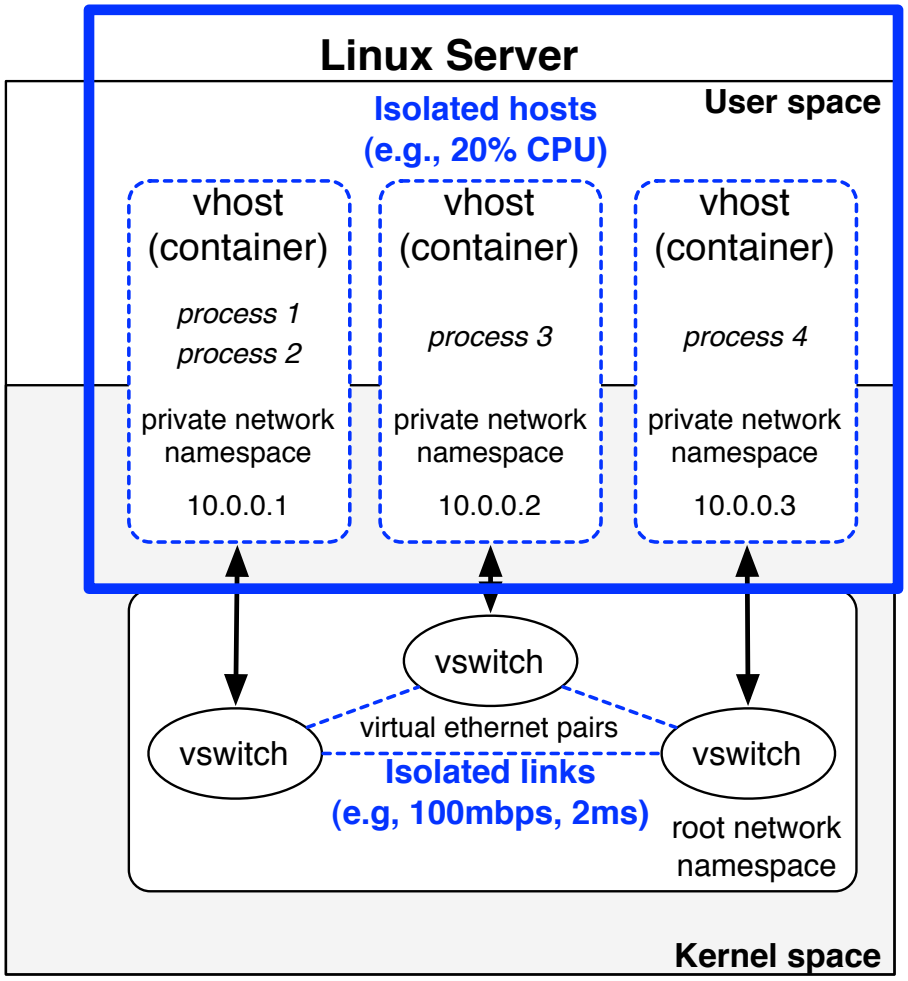


Linux packet schedulers

HTB, HFSC

Emulator + Performance Isolation

Emulator + Performance Isolation



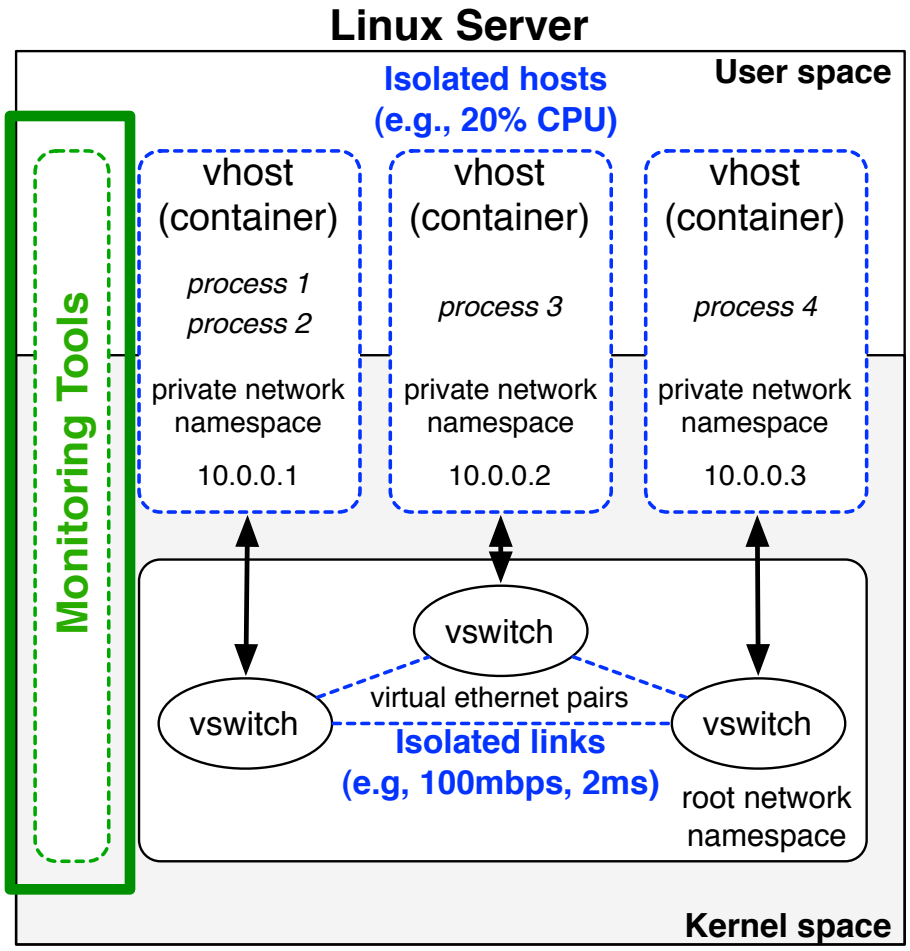
Linux process schedulers

CFS+BWlimits, RT

Emulator + Performance Isolation + Invariant Monitoring

Linux Kernel Tracing

enqueue, dequeue, etc.



3: Reproducing Research

Examples in the paper

- DCTCP [Alizadeh, SIGCOMM 2010]
- Router Buffer Sizing [Appenzeller, SIGCOMM 2004]
- Hedera ECMP [Al-Fares, NSDI 2010]

Able to replicate key results from 3 testbeds …

using an emulator.

How do you know it *really* works?

Test it. On ~~guinea pigs~~ students.

Stanford CS244 Spring ’12: Advanced Topics in Networking

→ Pick a paper.

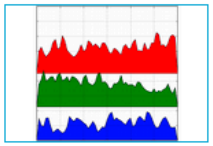
→ Reproduce a key result, or challenge it (with data).

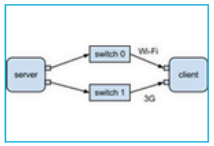
→ You have:

$100 EC2 credit,

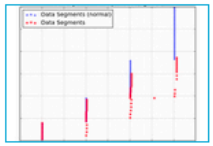
3 weeks, and

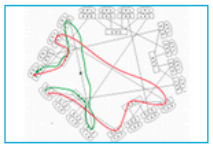
must use Mininet-HiFi.

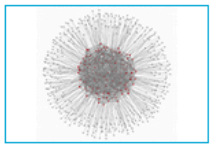

Exploring Outcast

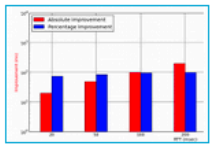


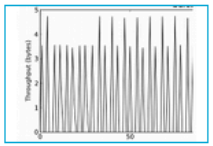
Multipath TCP over WiFi and 3G links

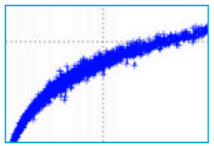

TCP Daytona: Congestion Control with a Misbehaving Receiver

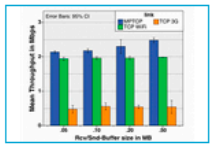

DCell: A Scalable and Fault-Tolerant Network Structure for Data Centers

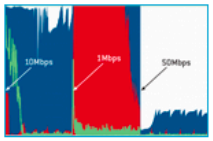

Jellyfish vs. Fat Tree


Choosing the Default Initial Congestion Window

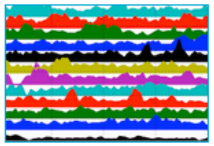

Seeing RED

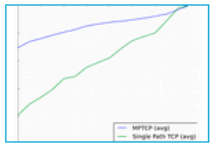

Why Flow-Completion Time is the Right Metric for Congestion Control

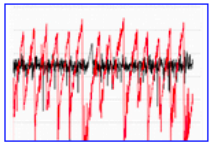

MPTCP Wireless Performance

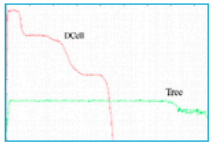


Solving Bufferbloat – The CoDel Way

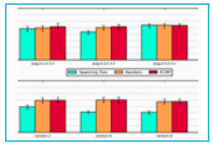

Life's not fair, neither is TCP (... under the following conditions)

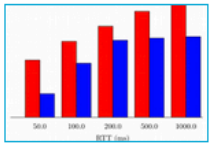

Fairness of Jellyfish vs. Fat-Tree

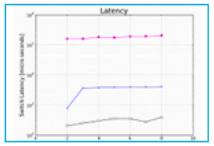

DCTCP and Queues



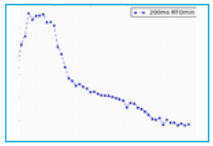
DCell : A Scalable and Fault-Tolerant Network Structure for Data Centers


Hedera


Increasing TCP's Initial Congestion Window

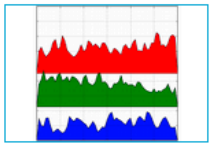


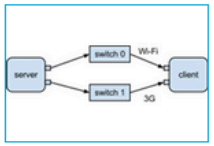
HULL: High Bandwidth, Ultra Low Latency


TCP Incast Collapse

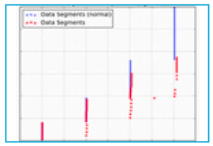
CoDel
HULL
MPTCP
Outcast
Jellyfish
DCTCP
Incast
Flow Completion Time
Hedera
DCell
TCP Initial Congestion Window
Misbehaving TCP Receivers
RED

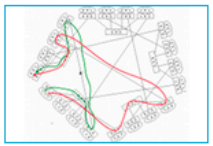
Project Topics: Transport, Data Center, Queuing

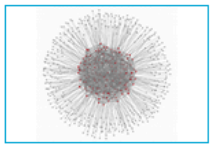

Exploring Outcast

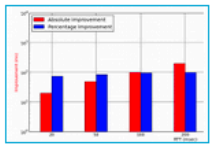


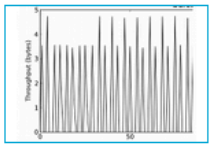
Multipath TCP over WiFi and 3G links

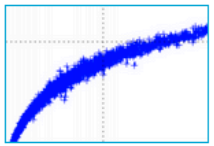

TCP Daytona: Congestion Control with a Misbehaving Receiver

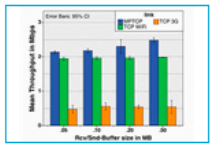

DCell: A Scalable and Fault-Tolerant Network Structure for Data Centers

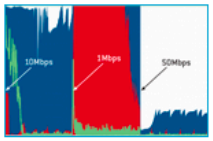

Jellyfish vs. Fat Tree


Choosing the Default Initial Congestion Window

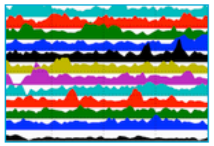

Seeing RED

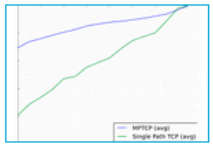

Why Flow-Completion Time is the Right Metric for Congestion Control

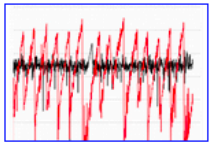

MPTCP Wireless Performance

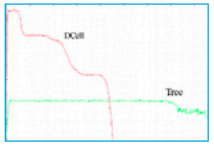


Solving Bufferbloat – The CoDel Way

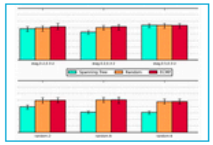

Life's not fair, neither is TCP (... under the following conditions)

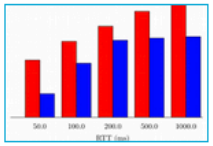

Fairness of Jellyfish vs. Fat-Tree

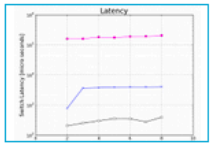

DCTCP and Queues



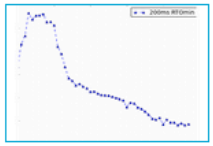
DCell : A Scalable and Fault-Tolerant Network Structure for Data Centers


Hedera


Increasing TCP's Initial Congestion Window

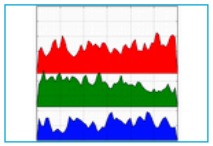


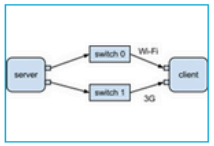
HULL: High Bandwidth, Ultra Low Latency


TCP Incast Collapse

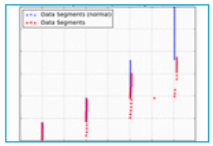
CoDel
HULL
MPTCP
Outcast
Jellyfish
DCTCP
Incast
Flow Completion Time
Hedera
DCell
TCP Initial Congestion Window
Misbehaving TCP Receivers
RED

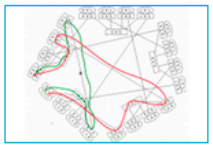
37 students
18 projects
16 replicated

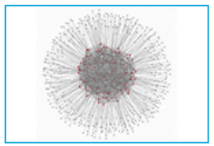

Exploring Outcast

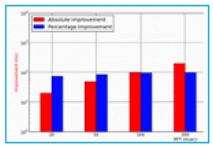


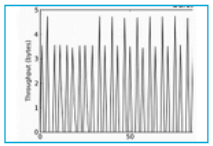
Multipath TCP over WiFi and 3G links

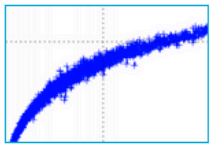

TCP Daytona: Congestion Control with a Misbehaving Receiver

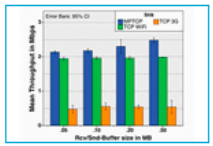

DCell: A Scalable and Fault-Tolerant Network Structure for Data Centers

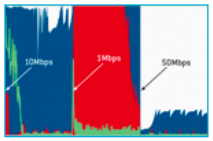

Jellyfish vs. Fat Tree


Choosing the Default Initial Congestion Window

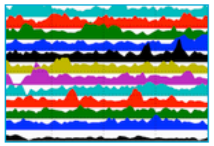

Seeing RED

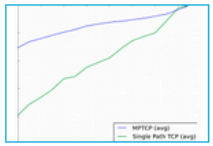

Why Flow-Completion Time is the Right Metric for Congestion Control

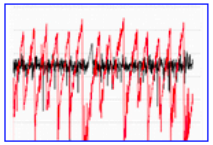

MPTCP Wireless Performance

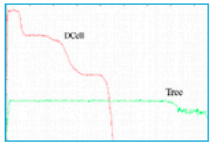


Solving Bufferbloat – The CoDel Way

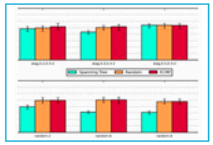

Life's not fair, neither is TCP (... under the following conditions)

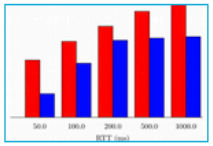

Fairness of Jellyfish vs. Fat-Tree

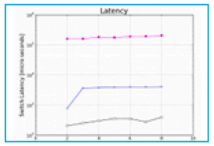

DCTCP and Queues



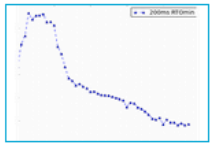
DCell : A Scalable and Fault-Tolerant Network Structure for Data Centers


Hedera


Increasing TCP's Initial Congestion Window

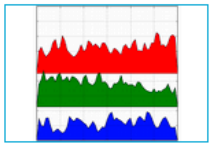


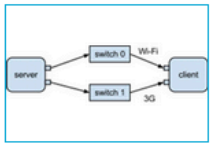
HULL: High Bandwidth, Ultra Low Latency


TCP Incast Collapse

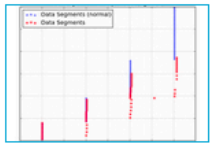
CoDel
HULL
MPTCP
Outcast
Jellyfish
DCTCP
Incast
Flow Completion Time
Hedera
DCell
TCP Initial Congestion Window
Misbehaving TCP Receivers
RED

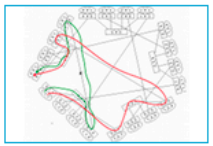
37 students
18 projects
16 replicated
4 beyond

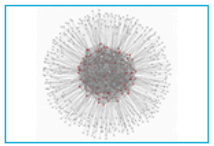

Exploring Outcast

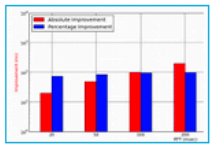


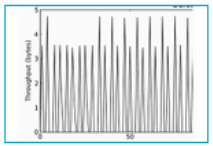
Multipath TCP over WiFi and 3G links

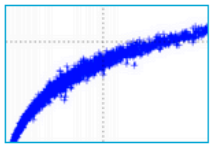

TCP Daytona: Congestion Control with a Misbehaving Receiver

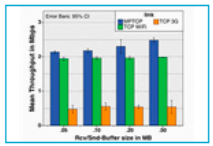

DCell: A Scalable and Fault-Tolerant Network Structure for Data Centers

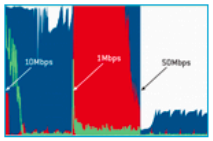

Jellyfish vs. Fat Tree


Choosing the Default Initial Congestion Window

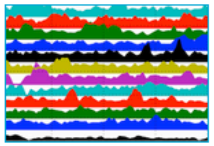

Seeing RED

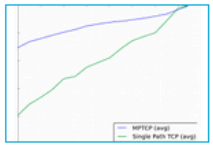

Why Flow-Completion Time is the Right Metric for Congestion Control

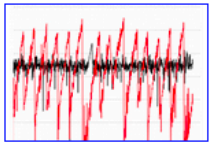

MPTCP Wireless Performance

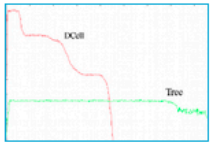


Solving Bufferbloat – The CoDel Way

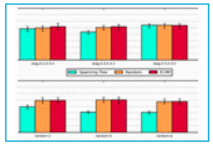

Life's not fair, neither is TCP (... under the following conditions)

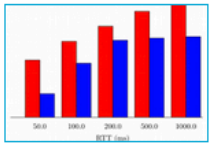

Fairness of Jellyfish vs. Fat-Tree

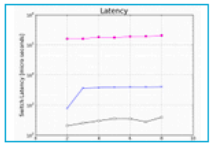

DCTCP and Queues



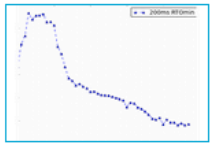
DCell : A Scalable and Fault-Tolerant Network Structure for Data Centers


Hedera


Increasing TCP's Initial Congestion Window



HULL: High Bandwidth, Ultra Low Latency


TCP Incast Collapse

CoDel
HULL
MPTCP
Outcast
Jellyfish
DCTCP
Incast
Flow Completion Time
Hedera
DCell
TCP Initial Congestion Window
Misbehaving TCP Receivers
RED

37 students
18 projects
16 replicated
4 beyond
2 not replicated

CoNEXT ‘12 runnable papers? 15/31 seem like candidates:

- MPTCP is not Pareto-optimal: Performance issues and a possible solution
- Architecting for Edge Diversity: Supporting Rich Services over an Unbundled Transport
- Tuning ECN for Data Center Networks
- Datacast: A Scalable and Efficient Reliable Group Data Delivery Service for Data Centers
- PAST: Scalable Ethernet for Data Centers
- Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming with FESTIVE
- Towards Agile and Smooth Video Adaption in Dynamic HTTP Streaming
- Application-aware Request Splitting for Interactive Cloud Applications
- Automatic Test Packet Generation
- FindAll: A Local Search Engine for Mobile Phones
- A SOFT Way for OpenFlow Switch Interoperability Testing
- Defending against large-scale crawls in online social networks
- BotFinder: Finding Bots in Network Traffic Without Deep Packet Inspection
- Cachet: A Decentralized Architecture for Privacy Preserving Social Networking with Caching
- New Opportunities for Load Balancing in Network-Wide Intrusion Detection Systems

NOT: Wireless, Modeling, Hardware, Social Networking, Security

Related Work

Related Work

- vEmulab: scale-out emulation [ATC08]
- DieCast: time dilation [NSDI07]
- SliceTime: synchronized time slices [NSDI11]

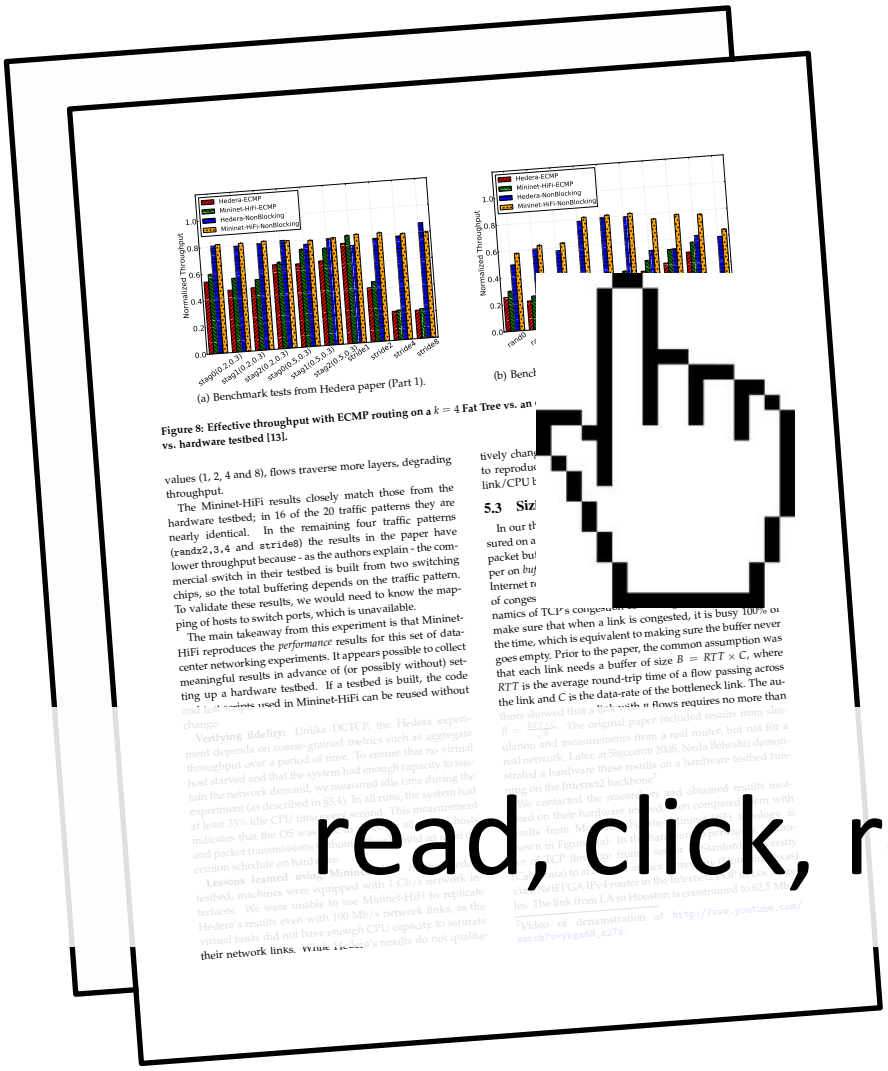
All are complementary techniques that could be added to Mininet-HiFi.

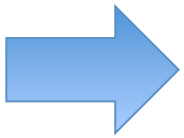
None measure event fidelity (S1)
Last two use full-system virtualization (S2)
None evaluate reproducibility at scale (S3)

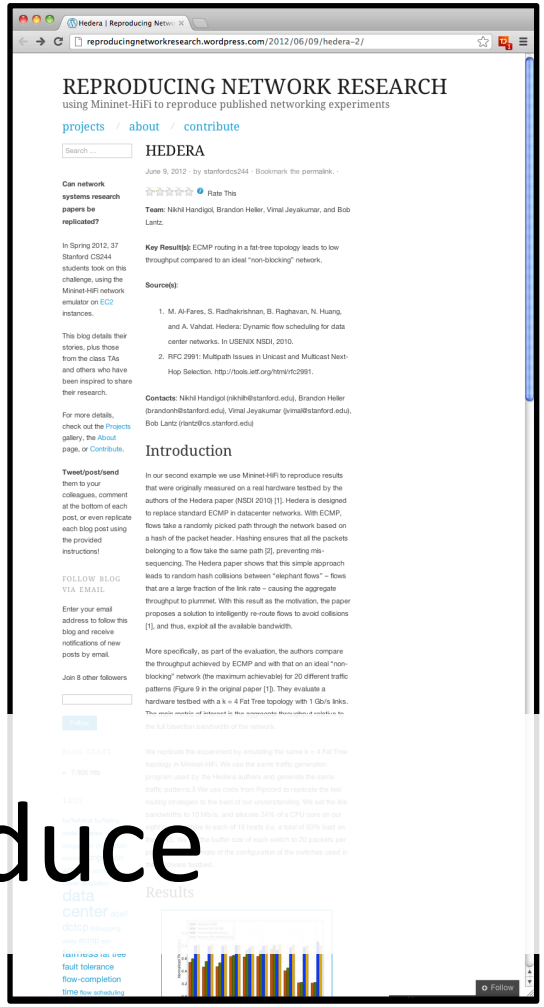
Progress Report:

Making runnable the network-paper default.

Runnable Paper Existence Proof





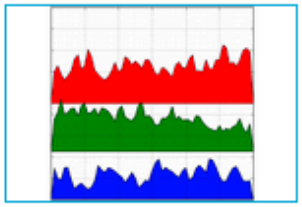


read, click, reproduce

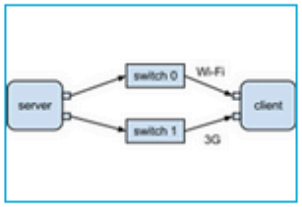
Reproduced Research Examples

reproducingnetworkresearch.wordpress.com

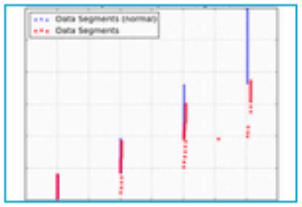
(or Google “reproducing network research”)



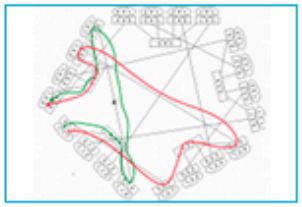
Exploring Outcast



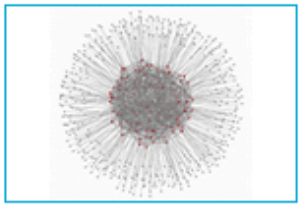
Multipath TCP over WiFi and 3G links



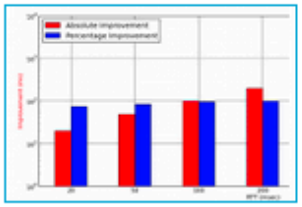
TCP Daytona: Congestion Control with a Misbehaving Receiver



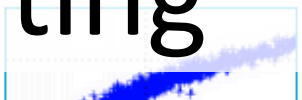
DCell: A Scalable and Fault-Tolerant Network Structure for Data Centers



Jellyfish vs. Fat Tree



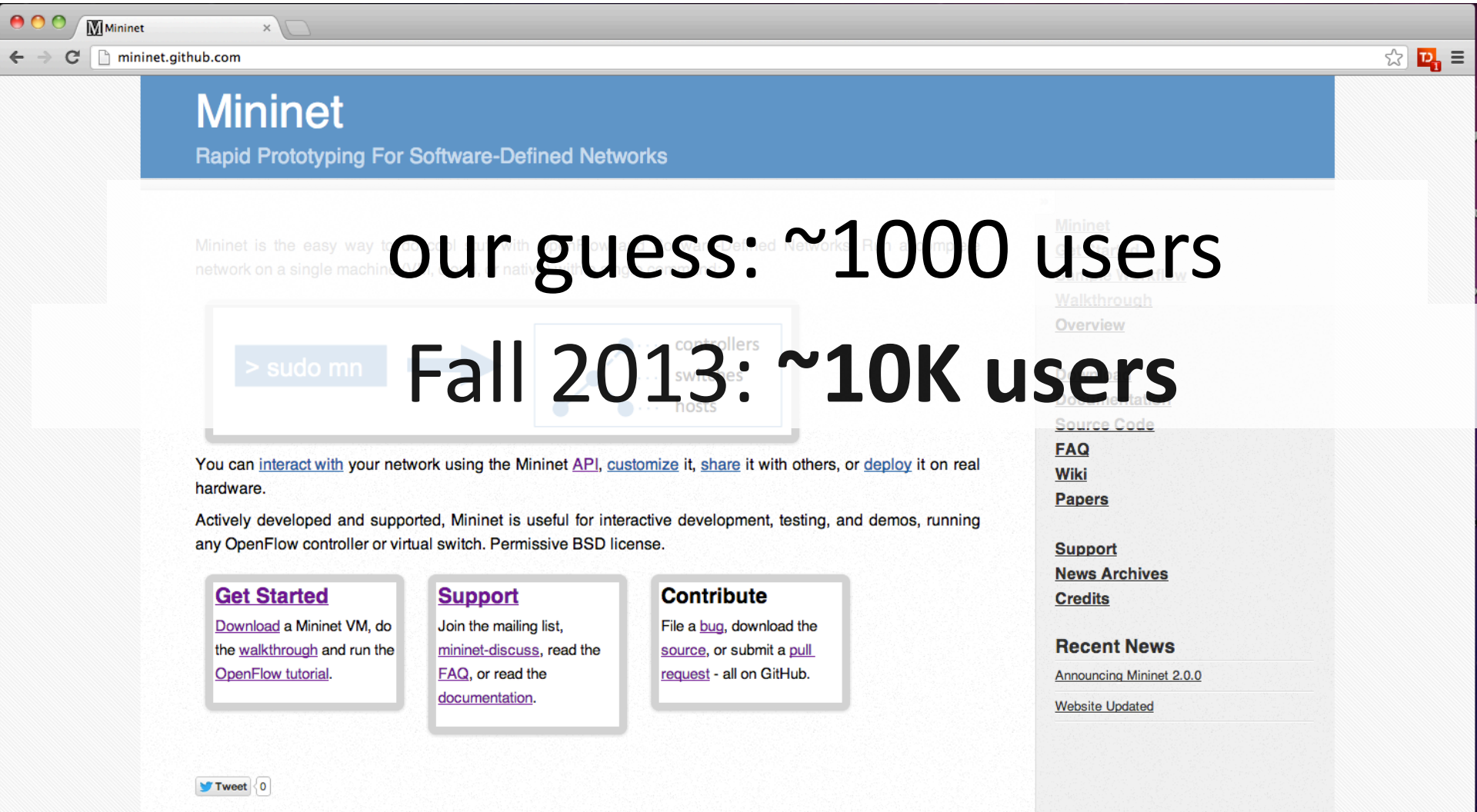
Choosing the Default Initial Congestion Window



20 and counting

Open-Source System w/Active User Community

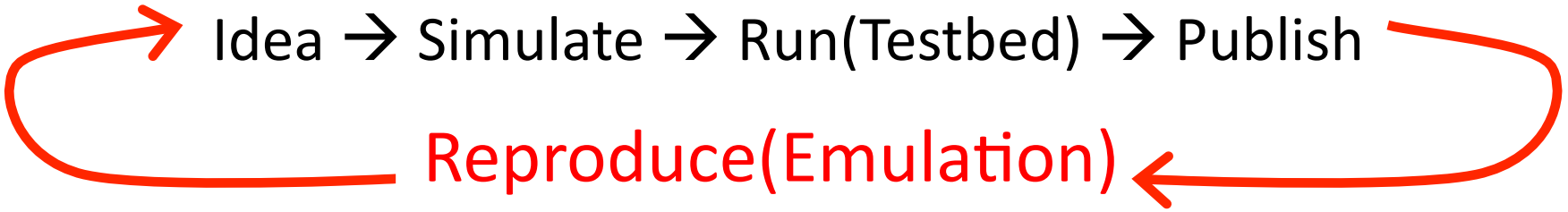
mininet.github.com



our guess: ~1000 users

Fall 2013: **~10K users**

New Workflows



Idea → Simulate → Run(Testbed) → Publish

Idea → Run(Emulation) → Publish

(aside) Why God doesn't have a Ph.D.

1) He had only one major publication.
2) It was in Hebrew.
3) It had no references.
4) It wasn't published in a referreed journal.
5) Some even doubt he wrote it by himself.
6) It may be true that he created the world, but what has he done since then?
7) His cooperative efforts have been quite limited.
8) **The scientific community has had a hard time replicating his results.**
9) He never applied to the ethics board for permission to use human subjects.
10) When one experiment went awry he tried to cover it up by drowning his subjects.
11) When subjects didn't behave as predicted, he deleted them from the sample.
12) Some say he had his son teach the class.
13) He expelled his first two students for learning.
14) He rarely came to class, and he just told students to read the book.
15) Although there were only 10 requirements, most of his students failed his tests.
16) His office hours were infrequent and usually held on a mountaintop.

Source: http://www-psych.stanford.edu/~pinto/god.html

Look for the shirt. Questions?

mininet.github.com

reproducingnetworkresearch.wordpress.com

Backup Slides

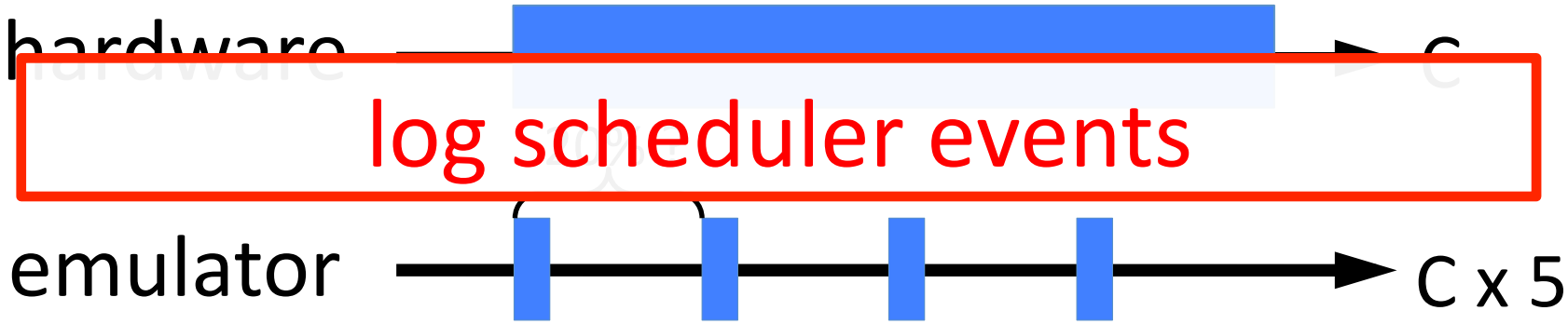
Doesn’t a simulator guarantee these invariants?

- Yes, exactly! A good one will.
- We’re trying to get the network fidelity of an emulator to match a simulator with virtual time.

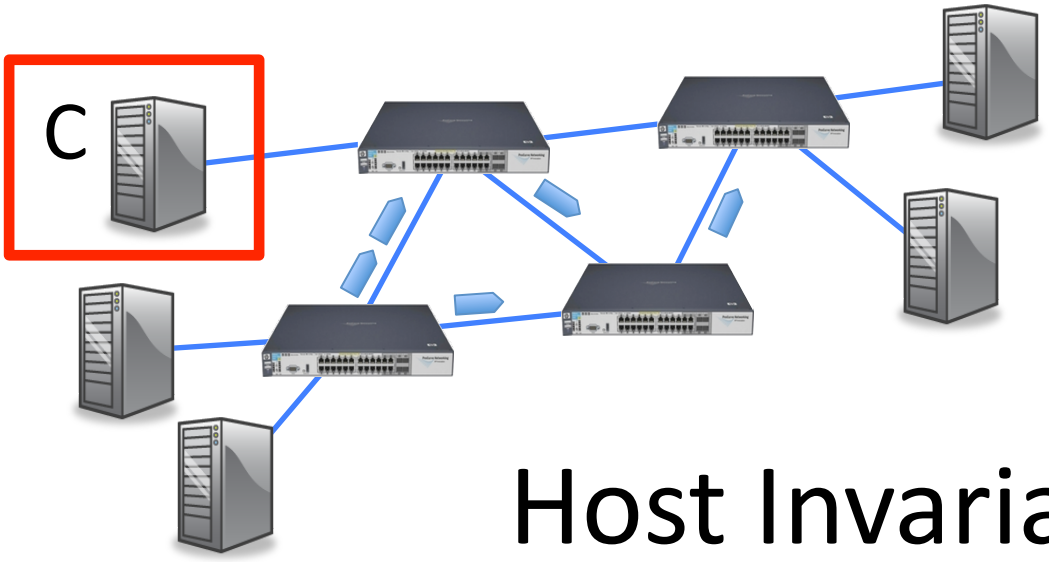
What about an RTOS?

- Every process must be bounded-time.
- Requires kernel mods.
- Conservative provisioning make the resulting system too resource-limited to be useful.
- May needlessly limit resources when they could be used.

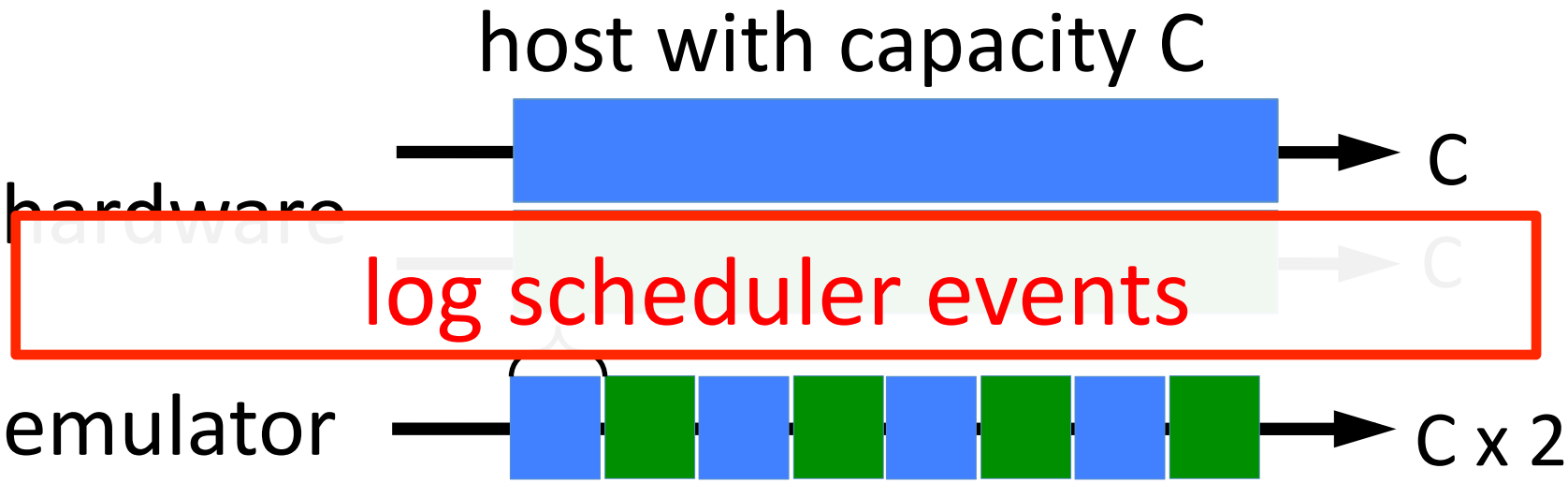
host with capacity C

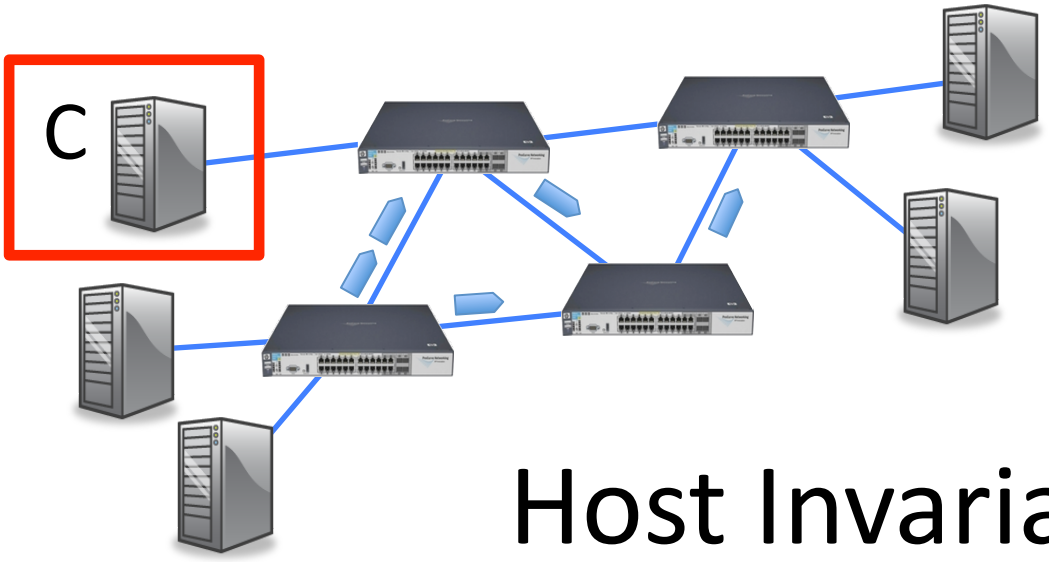


host capacity: $C_{measured} \leq C_{configured}$?



Host Invariants





Host Invariants