Network Emulation

Corso di Tecnologie di Infrastrutture di Reti

Martin Klapez

Department of Engineering *Enzo Ferrari*
University of Modena and Reggio Emilia

Modena, 27 May 2022

Overview

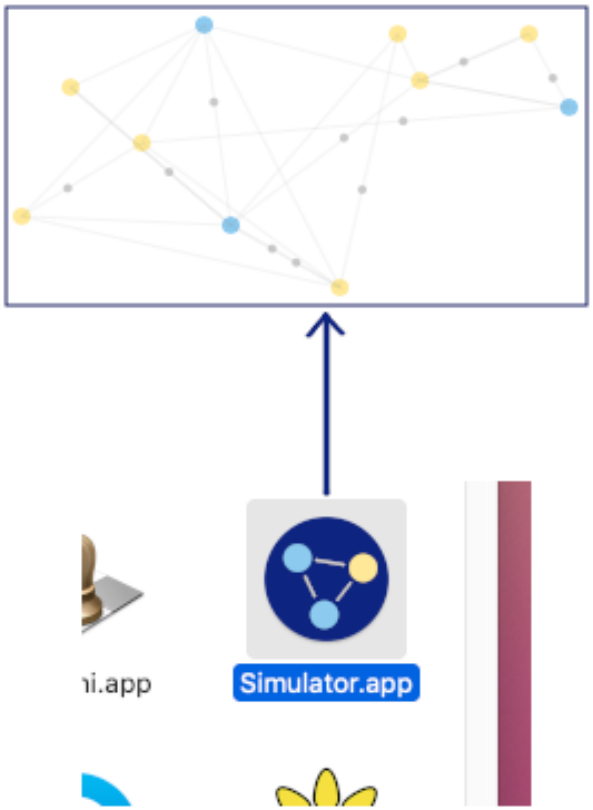
Slides at: netlab.unimore.it/pdf/emu.pdf

- Network Simulation vs Network Emulation vs The Real Thing
- Introduction to Mininet
- Experience Bufferbloat with Mininet
- Assignment: create a Mininet network with a custom topology

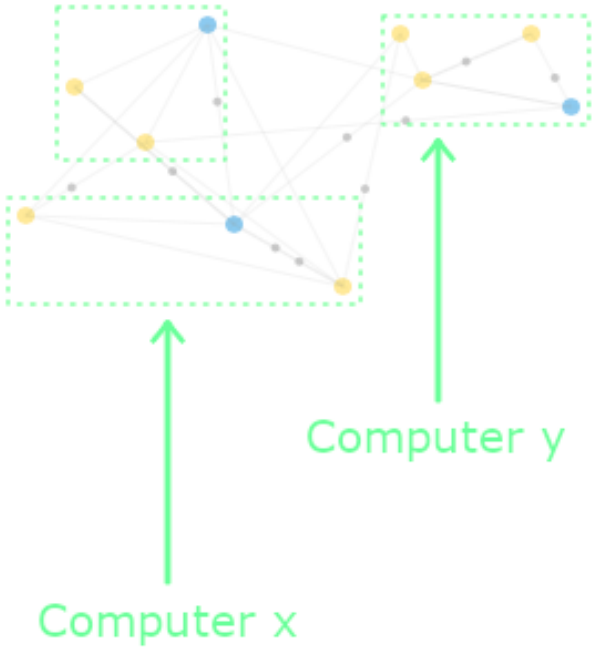
Networks: Simulation vs Emulation vs The Real Thing

A graphical representation

Simulation



Emulation



'Real'

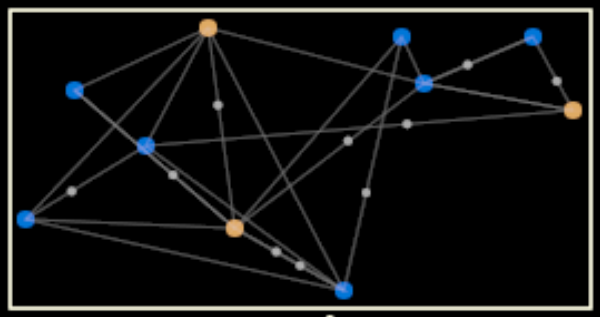


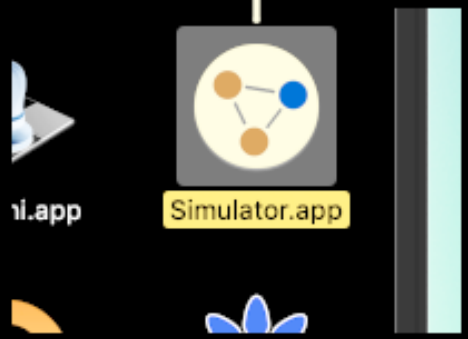


Networks: Simulation vs Emulation vs The Real Thing

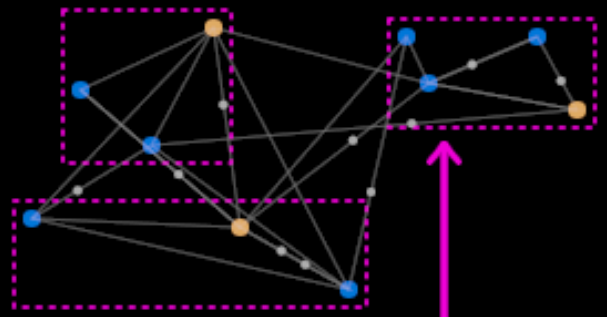
A graphical representation

Simulation

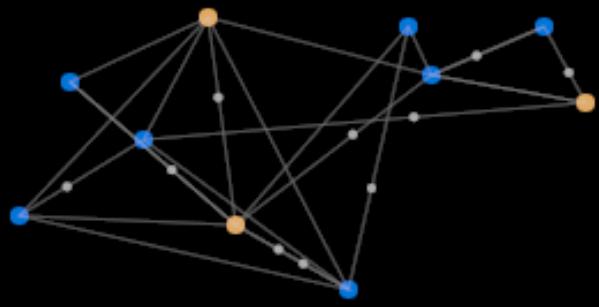




Emulation



'Real'





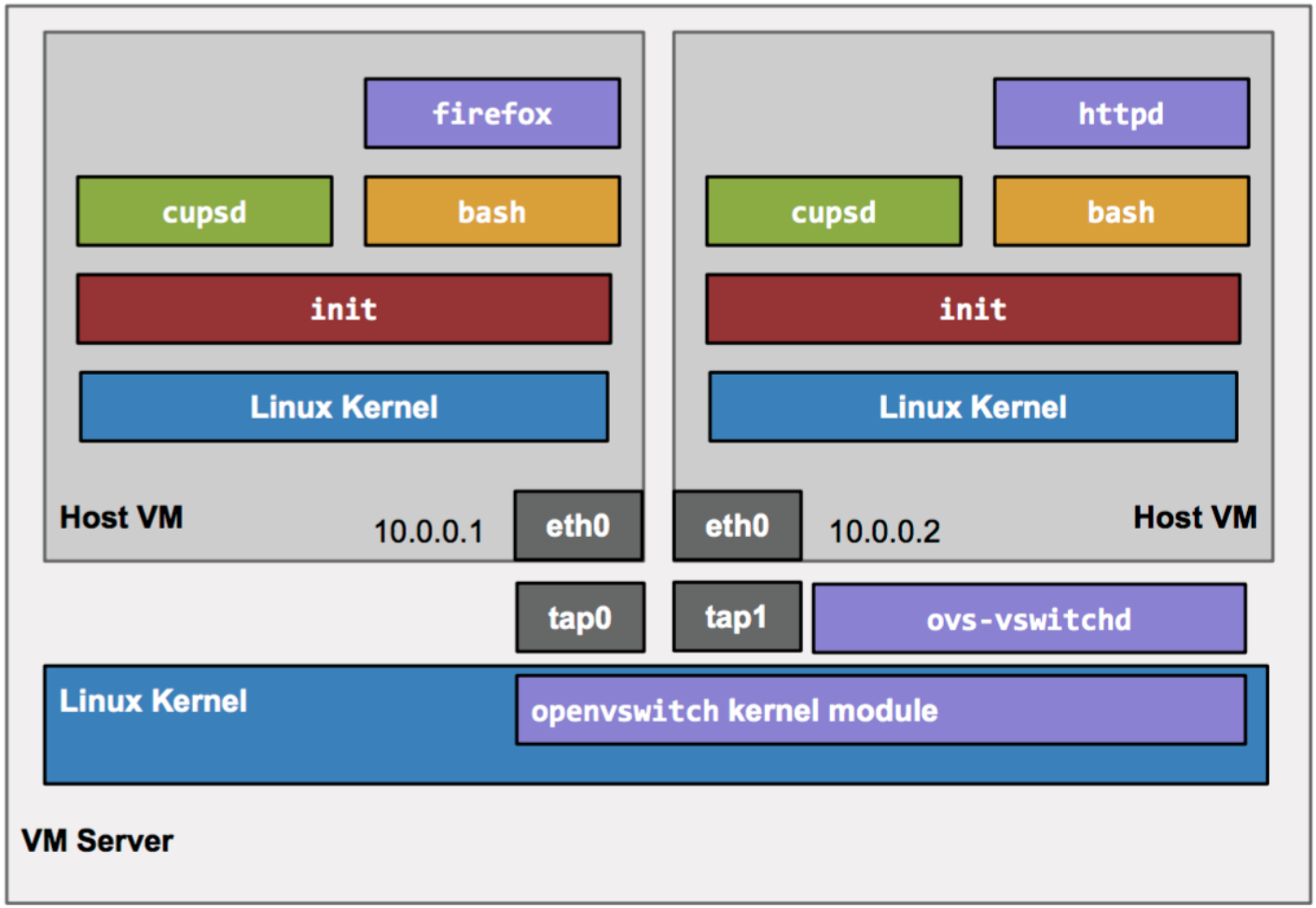
Networks: Simulation vs Emulation vs The Real Thing

A comparison

	Simulation	Emulation	Real
Code	y	x	x
Accuracy	if you're lucky	reasonably accurate	.
Cost	€€	€	$€^{€}$

Introduction to Mininet

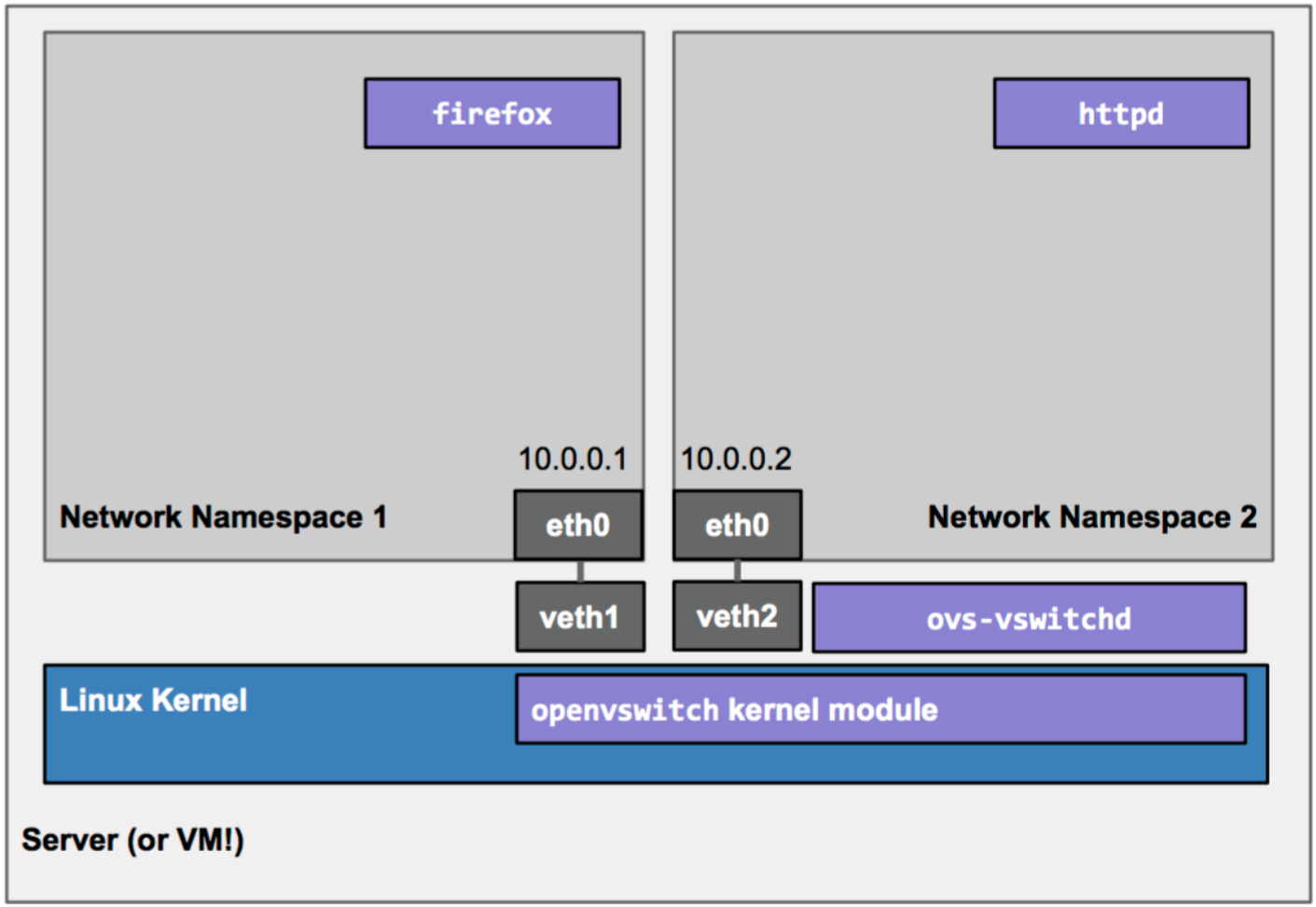
Network Emulator Architecture: Full System Virtualization



OK, but VMs are heavyweight: the memory overhead for each VM limits the scale to just a handful of switches and hosts.

Introduction to Mininet

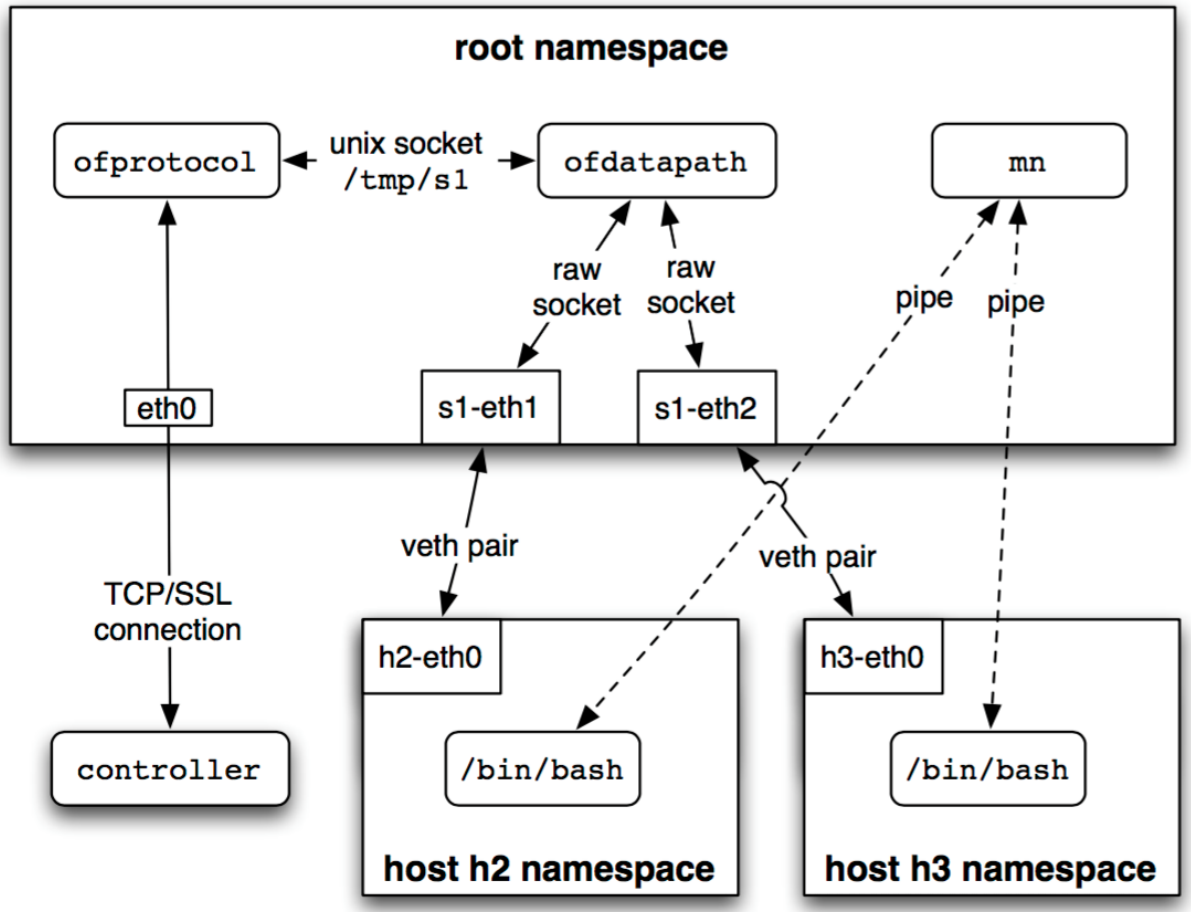
Network Emulator Architecture: Lightweight Virtualization (Mininet)



Mininet leverages Linux features (processes and virtual Ethernet pairs in network namespaces).

Introduction to Mininet

Network Emulator Architecture: Namespaces (Mininet)



Mininet creates a virtual network by placing host processes in network namespaces and connecting them with virtual Ethernet (veth) pairs. In this example, they connect to a user-space OpenFlow switch.

Introduction to Mininet

Mininet: the Main Components

- **Hosts**. *Network namespaces are containers for network state*. They provide processes (and groups of processes) with exclusive ownership of interfaces, ports, and routing tables (such as ARP and IP).
- **Links**. A virtual Ethernet pair, or veth pair, acts *like a wire* connecting two virtual interfaces; *packets sent through one interface are delivered to the other*, and each interface appears as a fully functional Ethernet port to all system and application software.
- **Switches**. The same packet delivery semantics that would be provided by a hardware switch.
- **Controllers**. An SDN controller could run inside a VM, natively on a host machine, or in the cloud.

Introduction to Mininet

Interaction with the network

- **Run commands on hosts**.
- Verify switch operations.
- Induce failures.
- Adjust link connectivity.
- Change the state of the network.
- Modify OpenFlow tables.
- ...

Introduction to Mininet

Limitations

Lack of performance fidelity, *especially at high loads*.

- ⋆ Time-multiplexing of CPU resources. There is no guarantee that a ready-to-send-a-packet host will be scheduled promptly, or that all switches will forward at the same rate.
- ⋆ Software table search in RAM $\rightarrow O(n)$. Lookup in TCAM $\rightarrow O(1)$. TCAM can be seen as the opposite of RAM. With the latter you need to provide the memory address of the data to retrieve, while data on TCAM can be retrieved by performing a query on its entire contents that returns the data address in a single clock cycle.

Experience Bufferbloat with Mininet

Prepare the ground

Turn on the provided virtual machine (Mininet-VM).
user: mininet
pass: mininet

Configure your keyboard layout:

```
> sudo dpkg-reconfigure keyboard-configuration
```

Ensure you have Internet access from the VM:

```
> ping 8.8.8.8
```

If you already have the folder below, delete it:

```
> sudo rm -r bufferbloat/
```

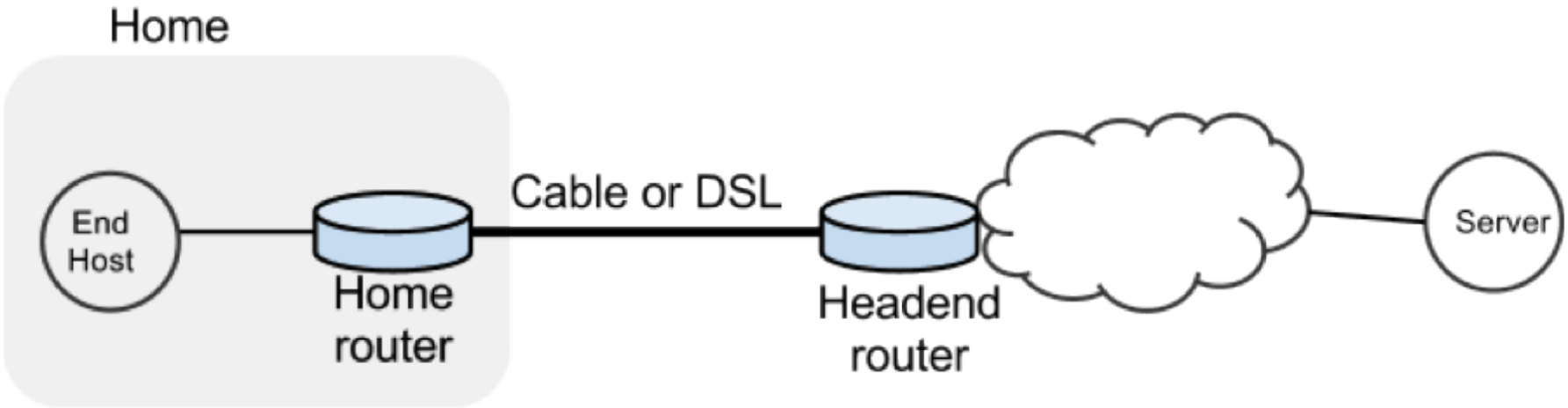
Experience Bufferbloat with Mininet

Introduction to the problem

Big buffers: 👍 or 👎?

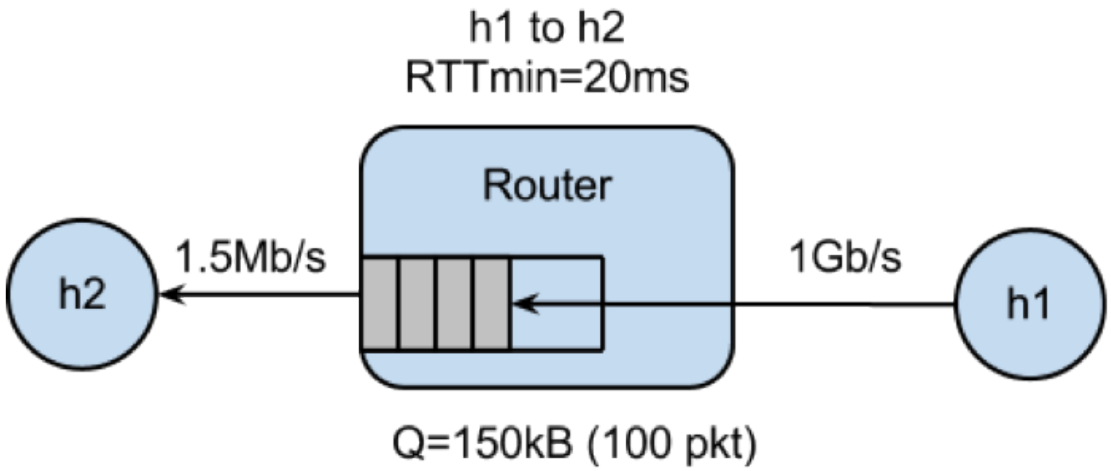
Big buffers deployed in the Internet may counterintuitively cause higher latencies, higher jitter, and lower throughput.

You lose less packets, but some of them may fall behind others for a very long time before being dispatched.



Experience Bufferbloat with Mininet

Get the topology



Get and run the Mininet network that emulates the topology:

```
> git clone https://gitlab.com/really_real/bufferbloat.git
> cd bufferbloat
> sudo ./run.sh
```

Experience Bufferbloat with Mininet

Commands

You can run the command *help* in the mininet console to see the list of available commands and their function, e.g.,

```
mininet > help
mininet > help net
mininet > net
```

Measure the delay between the two hosts:

```
mininet > h1 ping -c10 h2
```

Measure how long it takes to download a web page from h1:

```
mininet > h2 wget http://10.0.0.1
```

Experience Bufferbloat with Mininet

Video flow and Short flow: router buffer of 100 packets

Simulate a video streaming flow:

```
mininet > h1 ./iperf.sh
```

- ⋆ Start a simulation of a video streaming flow using the provided iPerf script
- ⋆ While iPerf is running, see how the long-lived iPerf flow affects the web page download
- ⋆ Observe how the delay between the hosts evolve over time

Experience Bufferbloat with Mininet

Video flow and Short flow: router buffer of 100 packets

Simulate a video streaming flow:

```
mininet > h1 ./iperf.sh
```

- ⋆ Start a simulation of a video streaming flow using the provided iPerf script
- ⋆ While iPerf is running, see how the long-lived iPerf flow affects the web page download
- ⋆ Observe how the delay between the hosts evolve over time

Why it happens what it happens?

- ⋆ Download: *cwnd*
- ⋆ Delay: *buffers*

Experience Bufferbloat with Mininet

Video flow and Short flow: router buffer of 20 packets

Restart Mininet with the smaller buffer:

```
mininet > exit
> sudo ./run-minq.sh
```

- Start a simulation of a video streaming flow using the provided iPerf script
- While iPerf is running, see how the long-lived iPerf flow affects the web page download
- Observe how the delay between the hosts evolve over time

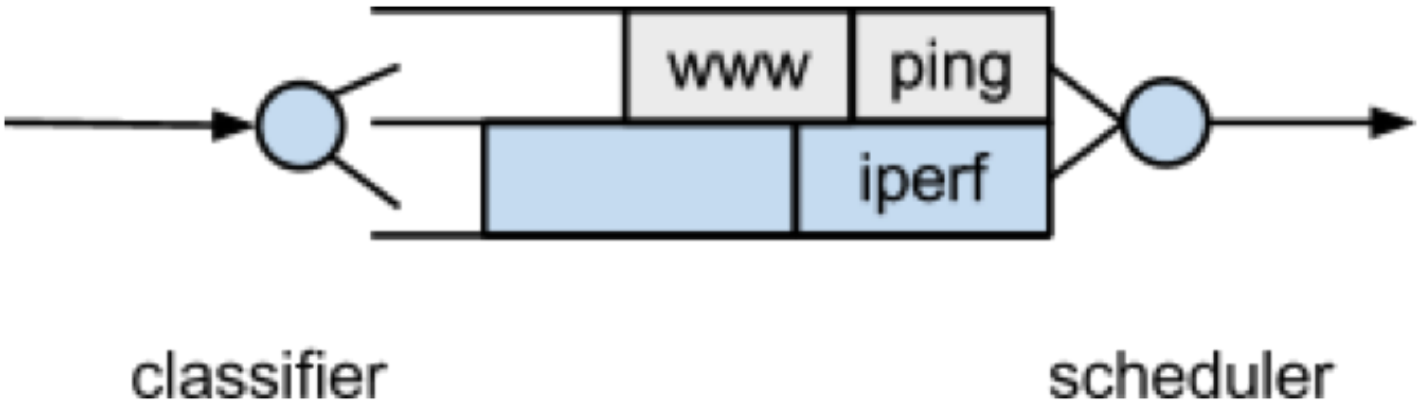
Worse? Better? Why?

Experience Bufferbloat with Mininet

Video flow and Short flow: different queues

The problem seems to be that packets from the short flow are stuck behind a lot of packets from the long flow.

What if we maintain a separate queue for each flow and then put iPerf and wget traffic into different queues?



In the experiment, the scheduler implements fair queueing so that when both queues are busy, each flow receives half of the bottleneck link rate.

Experience Bufferbloat with Mininet

Video flow and Short flow: different queues

Restart Mininet with the two queues on the Headend router:

```
mininet > exit
> sudo ./run-diff.sh
```

- ⋆ Start a simulation of a video streaming flow using the provided iPerf script
- ⋆ While iPerf is running, see how the long-lived iPerf flow affects the web page download
- ⋆ Observe how the delay between the hosts evolve over time

Worse? Better? Why?

Create Emulated Networks with Mininet

Assignment: Preparation

Mininet provides as templates a number of network topologies, create them with the commands below and explore the topologies.

1. > sudo mn --topo single
2. > sudo mn --topo linear
3. > sudo mn --topo linear,3
4. > sudo mn --topo tree
5. > sudo mn --topo tree,2
6. > sudo mn --topo tree,depth=2,fanout=3

(inside mininet, use *help* if you don't remember the commands)

```
mininet > help
```

Create Emulated Networks with Mininet

Assignment: custom topology

You should create with Mininet the custom topology that follows, getting inspiration from the example Mininet scripts, which you can find either:

- in the virtual machine at the path: /home/mininet/mininet/examples
- in the official Mininet repo at the URL: https://github.com/mininet/mininet/tree/master/examples

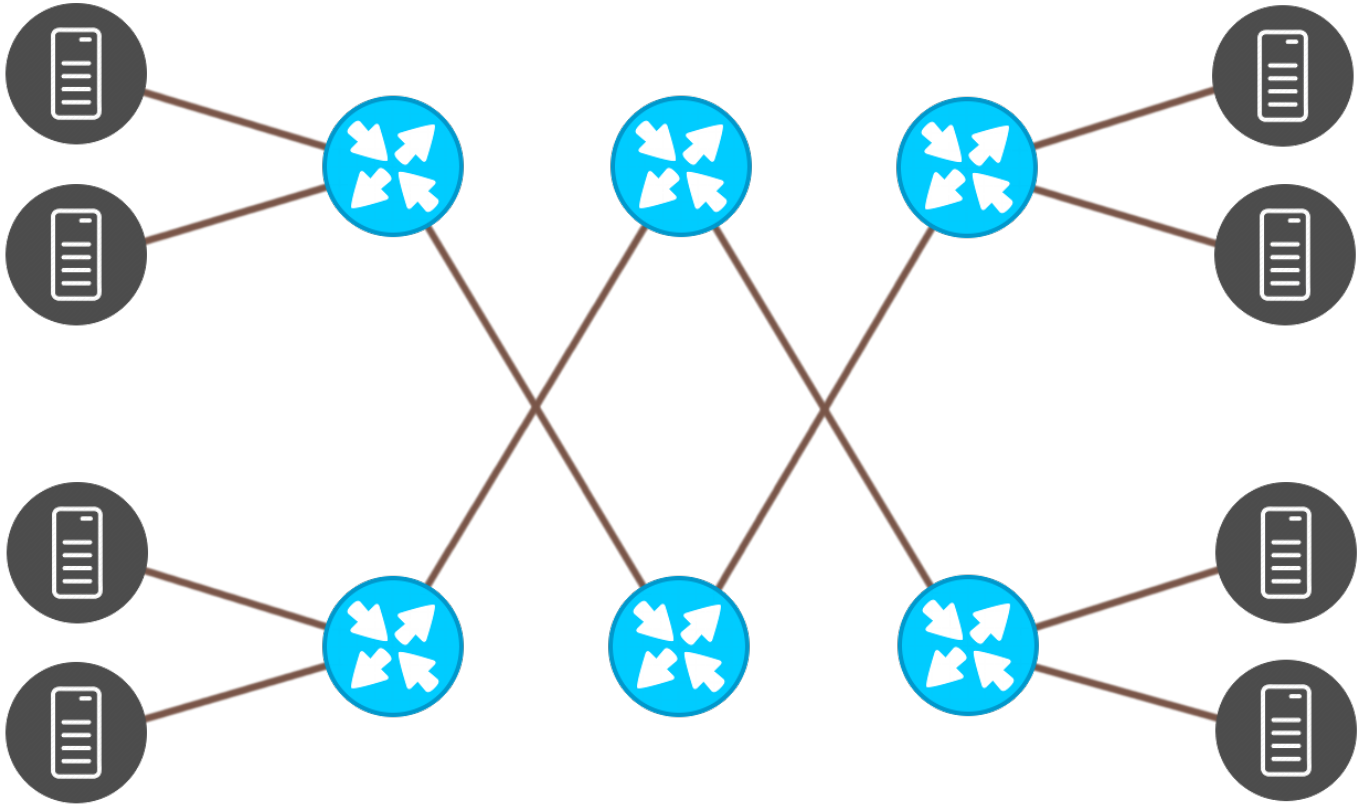
Tools:

```
> less file.ext          to show the content of the file (q to quit)
> nano myFile.ext        to edit the file (ctrl-o to save, ctrl-x to exit)
> cp file.ext fileCopy.ext   to copy file.ext into filecopy.ext
```

hint: emptynet.py is a good starting point

Create Emulated Networks with Mininet

Assignment: A custom topology



To execute your script:

```
> sudo python yourScript.py
```

Network Emulation

Sources

- ⋆ Bob Lantz, Brandon Heller, Nick McKeown, "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks", ACM, Hotnets '10, October 20-21, 2010, Monterey, CA, USA.
- ⋆ Te-Yuan Huang, Vimalkumar Jeyakumar Bob Lantz, Brian O'Connor, Nick Feamster, Keith Winstein, Anirudh Sivar, "Teaching Computer Networking with Mininet", ACM, SIGCOMM 2014 Tutorial, August 18, 2014, Chicago, IL, USA.
- ⋆ Stanford CS144, "An Introduction to Computer Networks, Bufferbloat Exercise", https://github.com/mininet/mininet/wiki/Bufferbloat.