Distributed Simulation and Emulation of Mobile Nodes in Dynamic Networks

Mobile Infrastructure'n'Network Integrated World (MiniWorld)

Nils Tobias Schmidt

February 26, 2017

Department of Mathematics & Computer Science
University of Marburg

Introduction: Outline

Introduction: Network Emulation

- Network is a critical part of software
- Software works well under good network conditions
- How to simulate bad network conditions ?
- Network emulation: "Reproduction of the function or action of a different computer, software system, etc."[1]
  - Compromise between *simulation* and *real-world experiments*
  - FreeBSD TCP stack evaluated with simulation, bug only showed up in network emulation[2]

[1] https://en.oxforddictionaries.com/definition/emulation. Last viewed on 07.12.2016

[2] Emmanuel Conchon, Tanguy Perennou, Johan Garcia, and Michel Diaz. W-nine: a two-stage emulation platform for mobile and wireless systems.

Introduction: Requirements

Network Emulation:

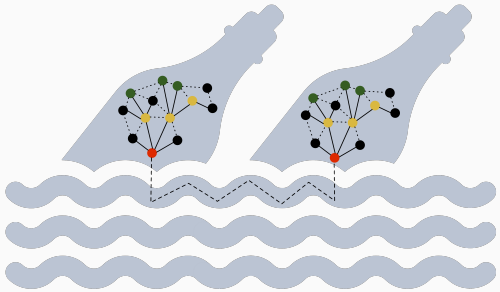
- Link impairment such as bandwidth, delay, packet loss, duplicate packets, etc.
- Node virtualization
- Virtual Network
- Reuse of layers 3 and above while simulating layers 1 and 2[3]

[3] Emmanuel Lochin, Tanguy Perennou, and Laurant Dairaine. When should i use network emulation? Annuals of telecommunications-anuales des télécommuncations.

Introduction: MiniWorld

MiniWorld aka Mobile Infrastructure'n'Network Integrated World

Logo:



- Python 2 based distributed network emulator
- Helps testing distributed applications, routing algorithms, etc.
- Node virtualization via full-system virtualization
  - Not limited to certain OS, kernel, hardware
- Leverages Linux kernel networking
- Distributed mode, since full-system virtualization poses high demands on resources

Design: Outline

Design: Architecture

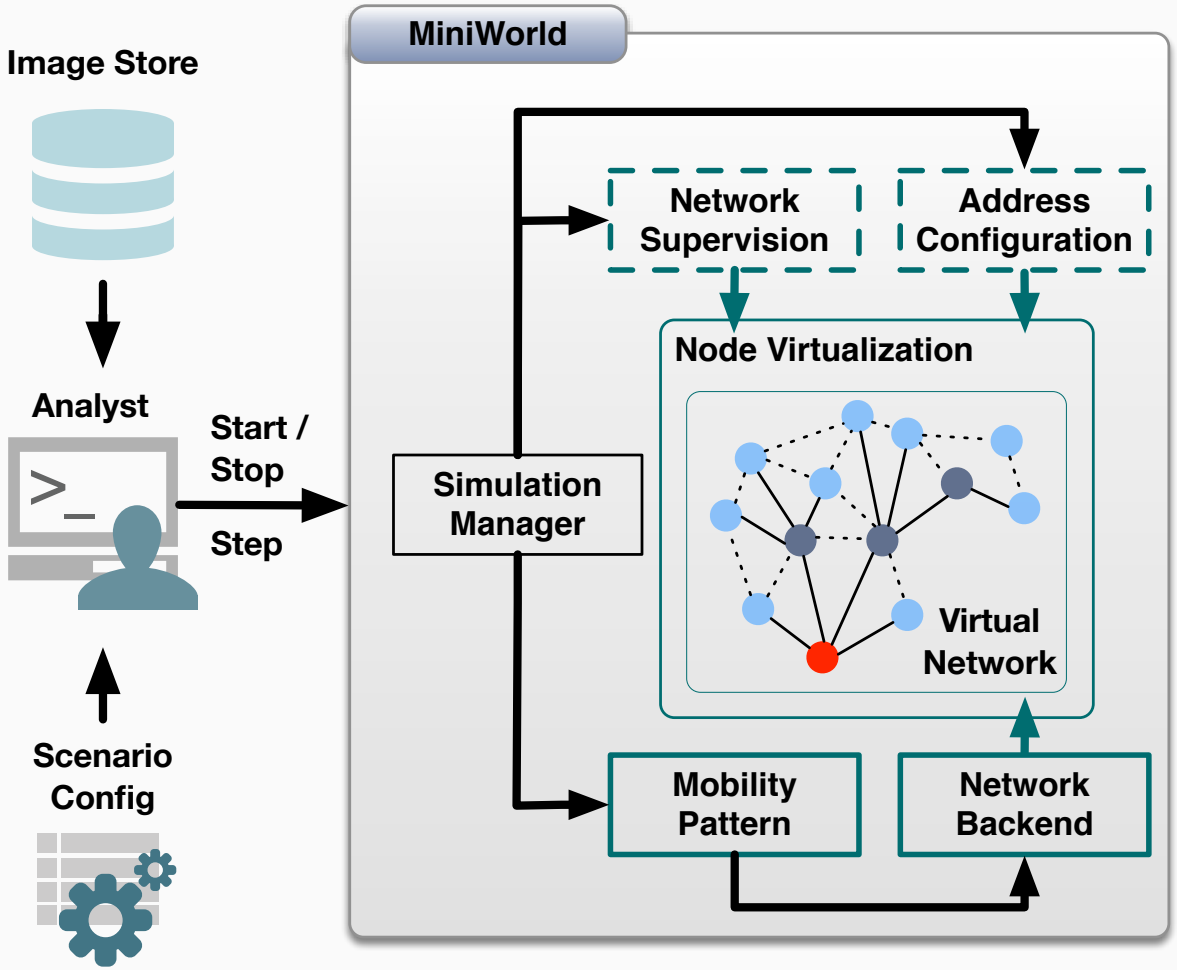


Figure 1: MiniWorld Architecture Implementation

Design: Workflow

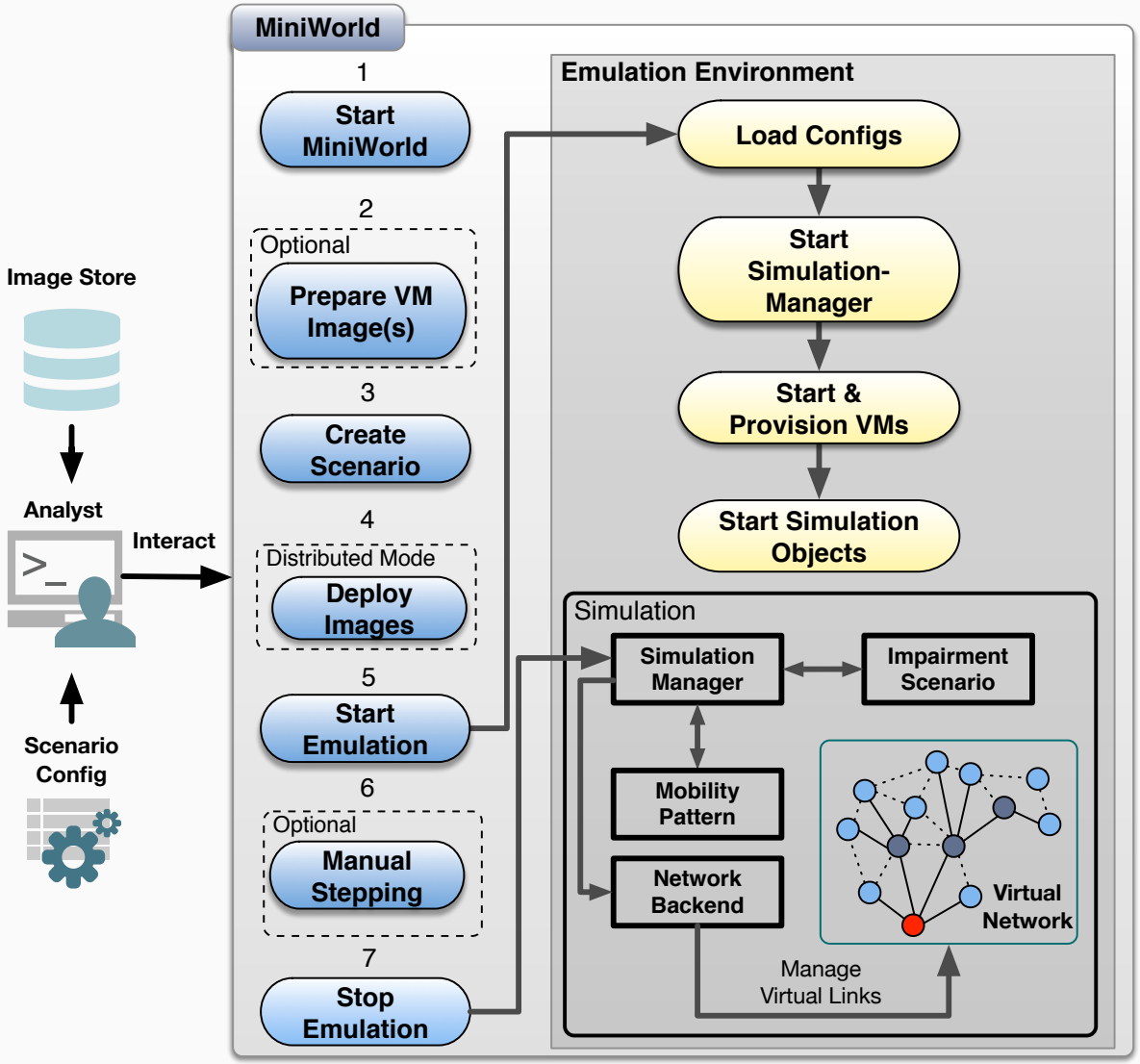


Figure 2: Workflow

Design: Network Backends

Requirements:

- Interconnects VMs $\Rightarrow$ Manages Virtual network
- Controls link impairment
- Influences switching times (should be $< 1s$ for live emulation)
- Distributed mode

Implemented:

- Bridged LAN
- Bridged WiFi
- VDE
- WiFi

To be continued ...

Design: Mobility Pattern

Batteries included ...

Node mobility:

- Distance-based (d)
  - Based on OSM maps
  - Realistic movement of mobile nodes
  - Link impairment could be influenced by obstacles between nodes
- Coordination-based (c)
  - Only distance matters for link impairment

Implemented:

- RandomWalk (c)
- MoveOnBigStreets (c)
- Arma3 (c)
- CORE Mobility
  - LAN (d)
  - WiFi (c)

Design: Interfaces

Interface:

- Represents NIC in VM
- Different interface types
- Multiple instances of interface type possible
- InterfaceFilter
- Special Interfaces

Interface types:

- Mesh
- Hub
- Management
- Ap
- Ad-Hoc
- Bluetooth

Design: Link Quality Model

- Controls link-impairment
- Tightly coupled with network backend, since network backends differ strongly (user-space vs kernel-space etc.)

- Bridged network backends: Linux TC
- VDE network backend: Wirefilter
- WiFi network backend: -

Design: Link Quality Model

- Fixed-Range Model
  - Distance $<$ 30m
  - Static Link Impairment only

Design: Link Quality Model

- Fixed-Range Model
- WiFi Simple Linear

- Varies bandwidth and delay with respect to the distance.
- Requires a maximum bandwidth to be defined.
- Bandwidth decreases linear, delay increases linear with the distance.
- Static & Dynamic Link Impairment

Design: Link Quality Model

- Fixed-Range Model
- WiFi Simple Linear
- WiFi Simple Exponential

- Halves bandwidth and doubles delay every four meters.
- Static & Dynamic Link Impairment

Design: Outline

Design: Network Backend VDE

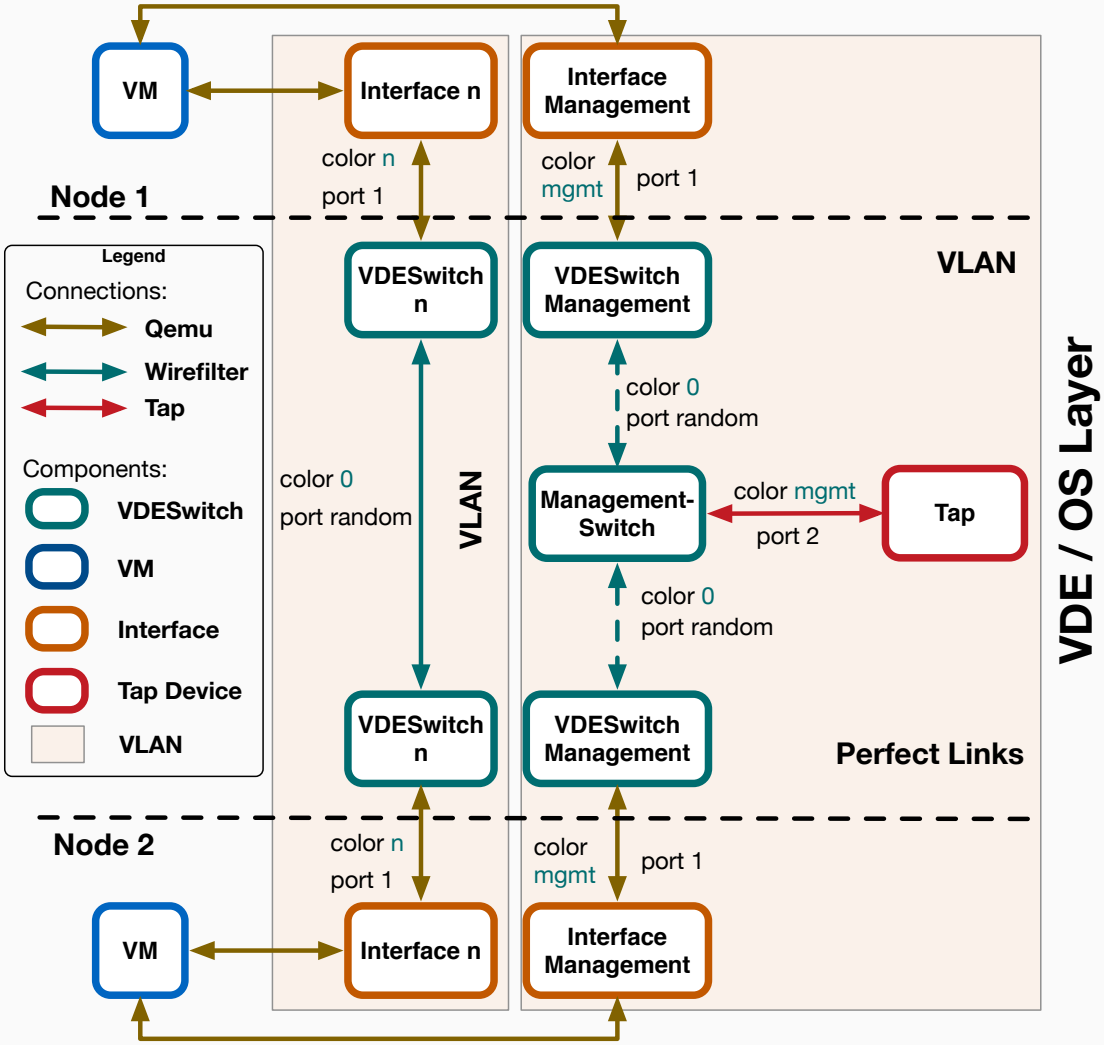


Figure 3: Network Backend VDE

Design: Bridged Network Backends

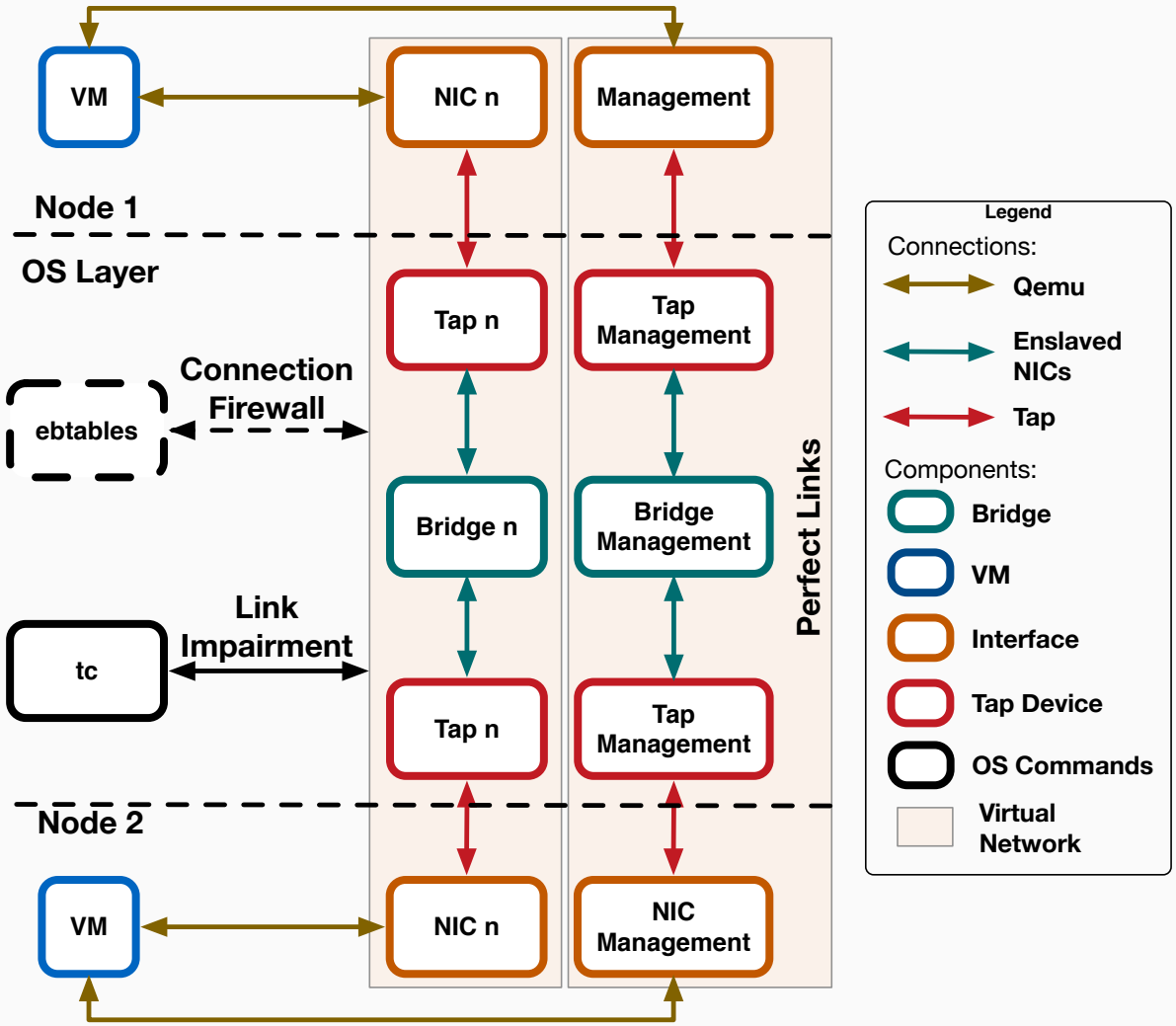


Figure 4: Bridged Network Backends

Design: WiFi Network Backend

- Linux-only
- Leverages mac80211_hwsim kernel module
- Simulates arbitrary number of WiFi NICs
- Offers nl80211 interface to user-space
- Network backend runs as user-space application inside VM

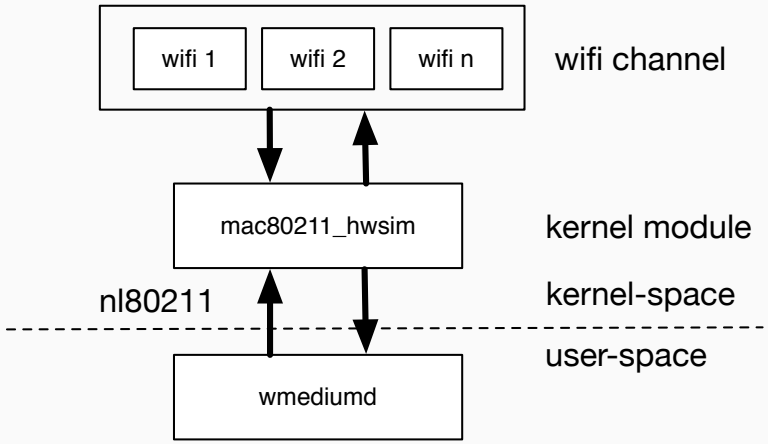


Figure 5: Wmediumd architecture

Design: WiFi Network Backend

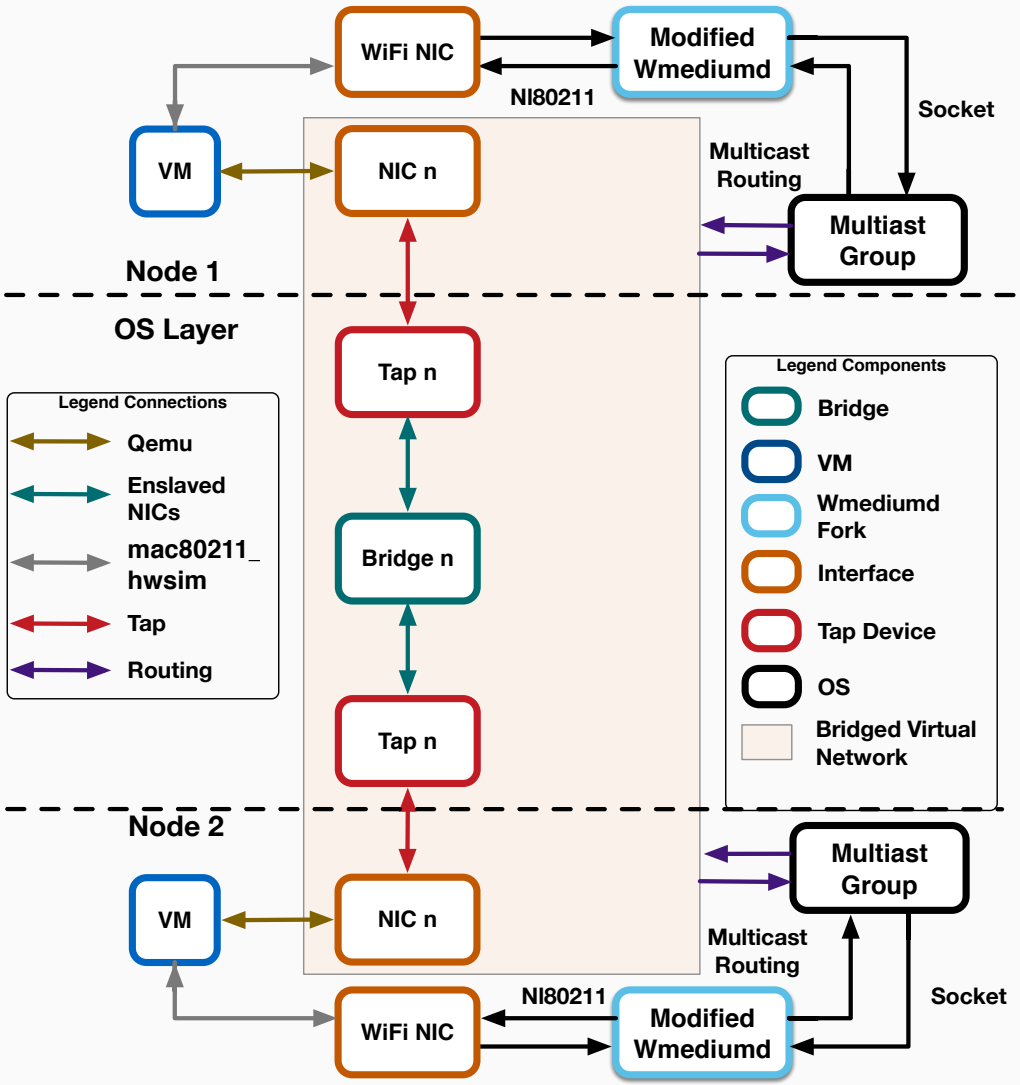


Figure 6: WiFi Network Backend

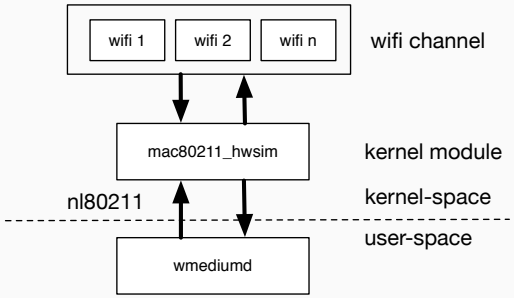


Figure 7: Wmediumd architecture

Design: Scenario Config

- Defines the #nodes, link quality model, network backend, shell commands to be executed etc.
- JSON-based configuration file for a scenario
- More in demo ...

Design: Distributed Architecture

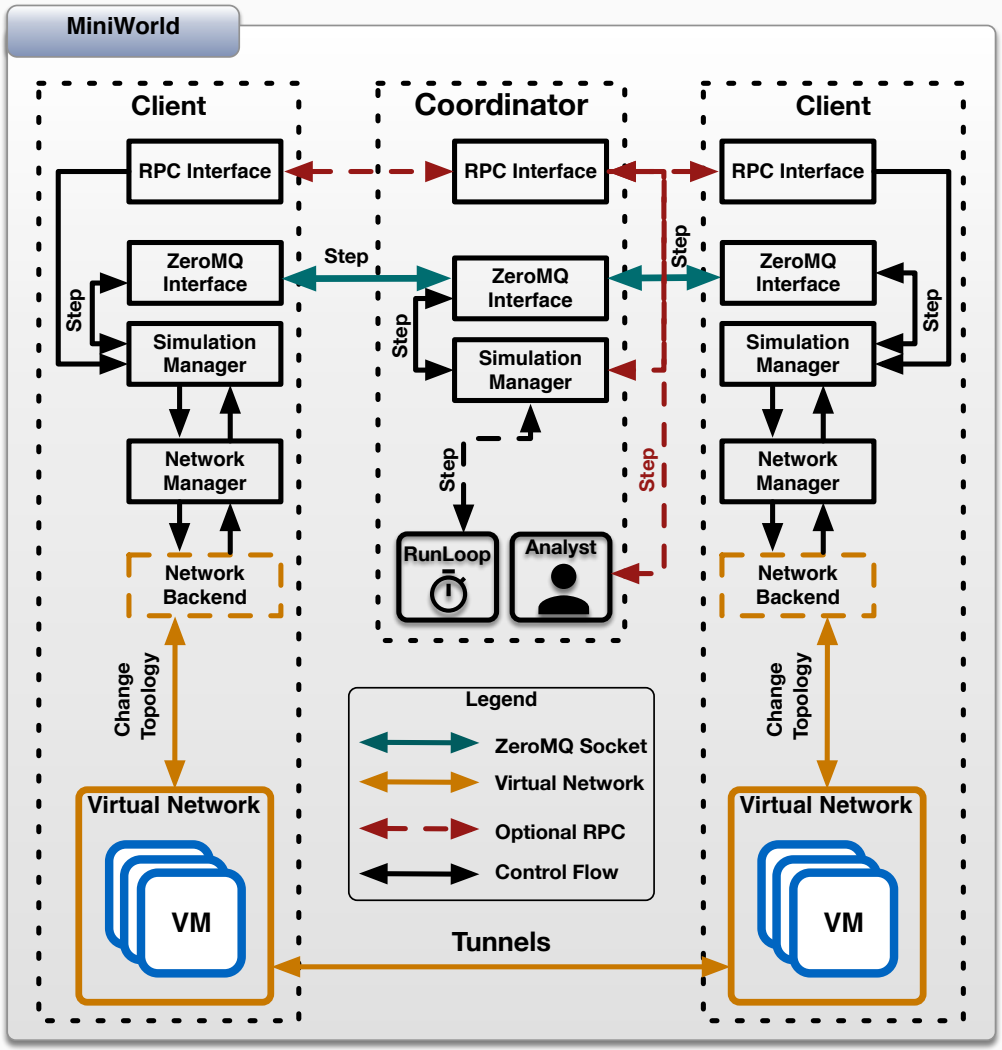


Figure 8: Distributed Architecture

Design: Distributed Communication

ZeroMQ:

- "sockets on steroids" [4]
- Distributed message passing without central message queue
- Message queue included in each program
- Very fast

Implementation:

- DFA states:
  1. Register
  2. Exchange
  3. Start nodes
  4. Loop: Distance Matrix
- Request-Reply vs. Publish-Subscribe pattern
- JSON vs. MessagePack serialization

[4] http://zguide.zeromq.org/page:all. Last viewed at 26.0.2.17.

Design: Scheduling

- Scheduler Equal
- Schedule nodes equally among all Emulation Servers

Design: Scheduling

- Scheduler Equal
- Scheduler Score

- Schedules based on CPU and RAM score
- CPU
  - Bogomips
  - Nodes distributed according to CPU score
- RAM
  - $(free -m) for free RAM
  - RAM only used to prevent memory overcommitment

Demo Time: Outline

Demo Time: MiniWorld in Practice

Evaluation: Outline

Evaluation: Test Environment

Resource	Value
Host OS	Ubuntu 14.04.4 LTS
Docker OS	Ubuntu 14.04.4 LTS
Kernel	4.2.0-42-generic
RAM	32x DIMM DDR3 Synchronous 1600 MHz 8GiB
HDD	1862GiB, 15000rpm
CPU	AMD Opteron(tm) Processor 6376 @2.3GHz, Turbo 3.2GHz
CPU Cores Physical/Virtual	16/64
Network	NetXtreme II BCM5709 Gigabit Ethernet 1Gbit/s
Iproute2	Git release v4.2.0
Pyroute	Git [5]
Qemu	Git release v2.6.0

Table 1: Technical Data Test System 1

[5] `https://github.com/svinota/pyroute2`. Commit ID: 13f1adb1ab2d5ca8927f1eb618bbb93170b61c33

Evaluation: Test Images

Image Name	Size (MiB)
OpenWrt Barrier Braker	68
Debian 8	1568

Table 2: Node Images

Evaluation: Outline

Evaluation: Boot Mode Times

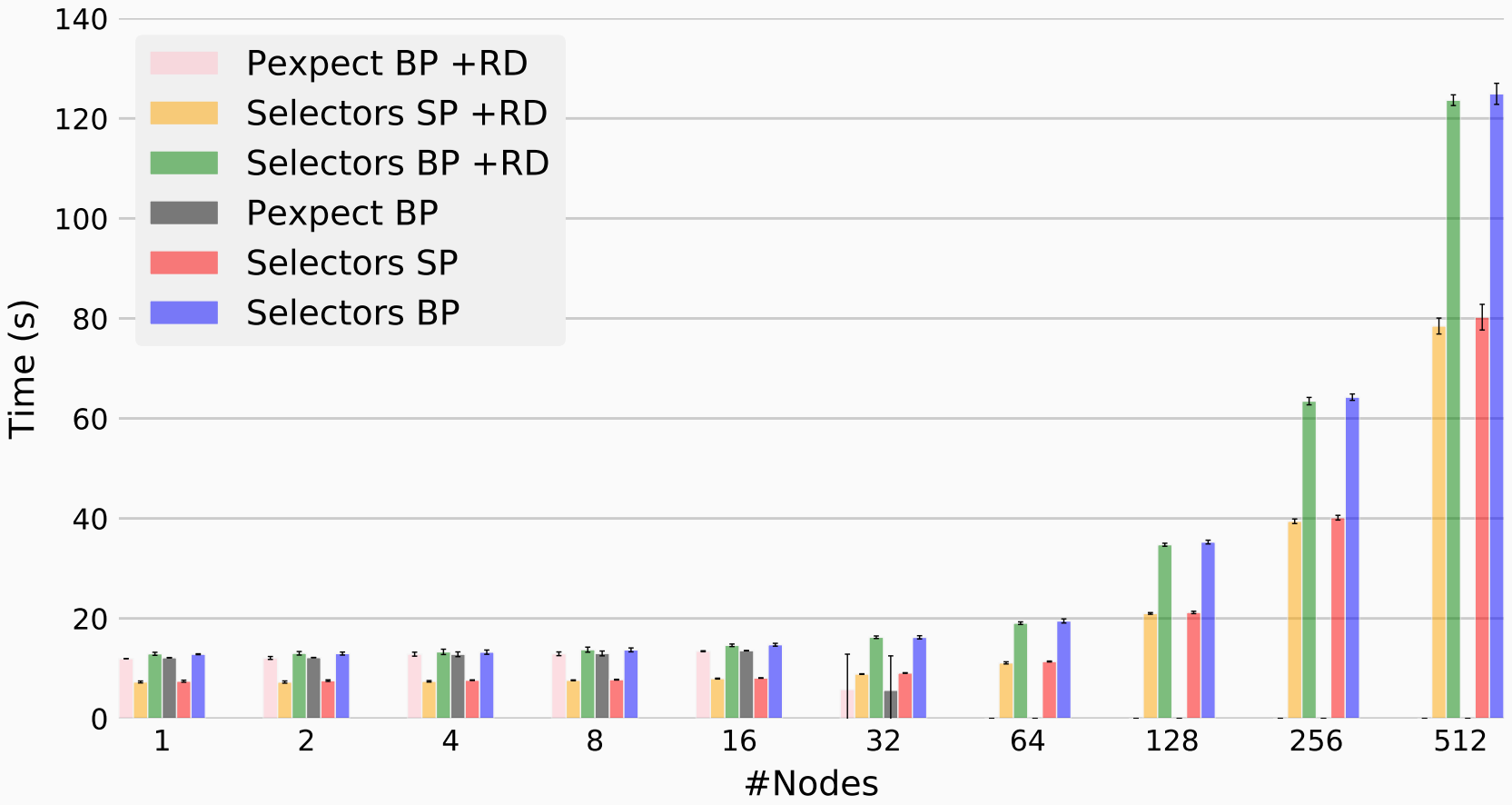


Figure 9: Boot Mode Times OpenWrt Barrier Braker, 32MiB RAM

Evaluation: Boot Mode Times

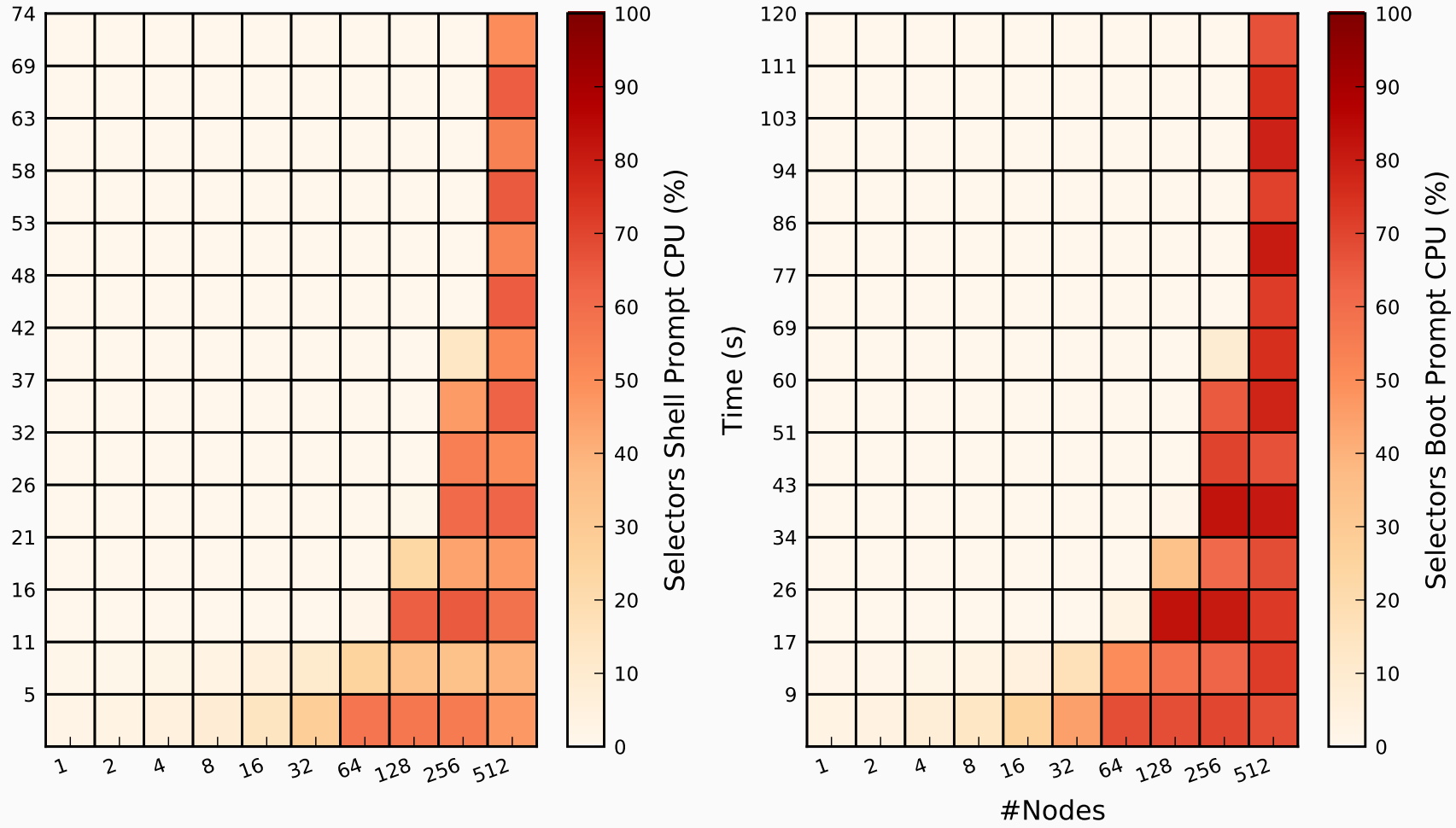


Figure 10: Boot Modes CPU, OpenWrt Barrier Braker, 32MiB RAM

Evaluation: Image Comparison

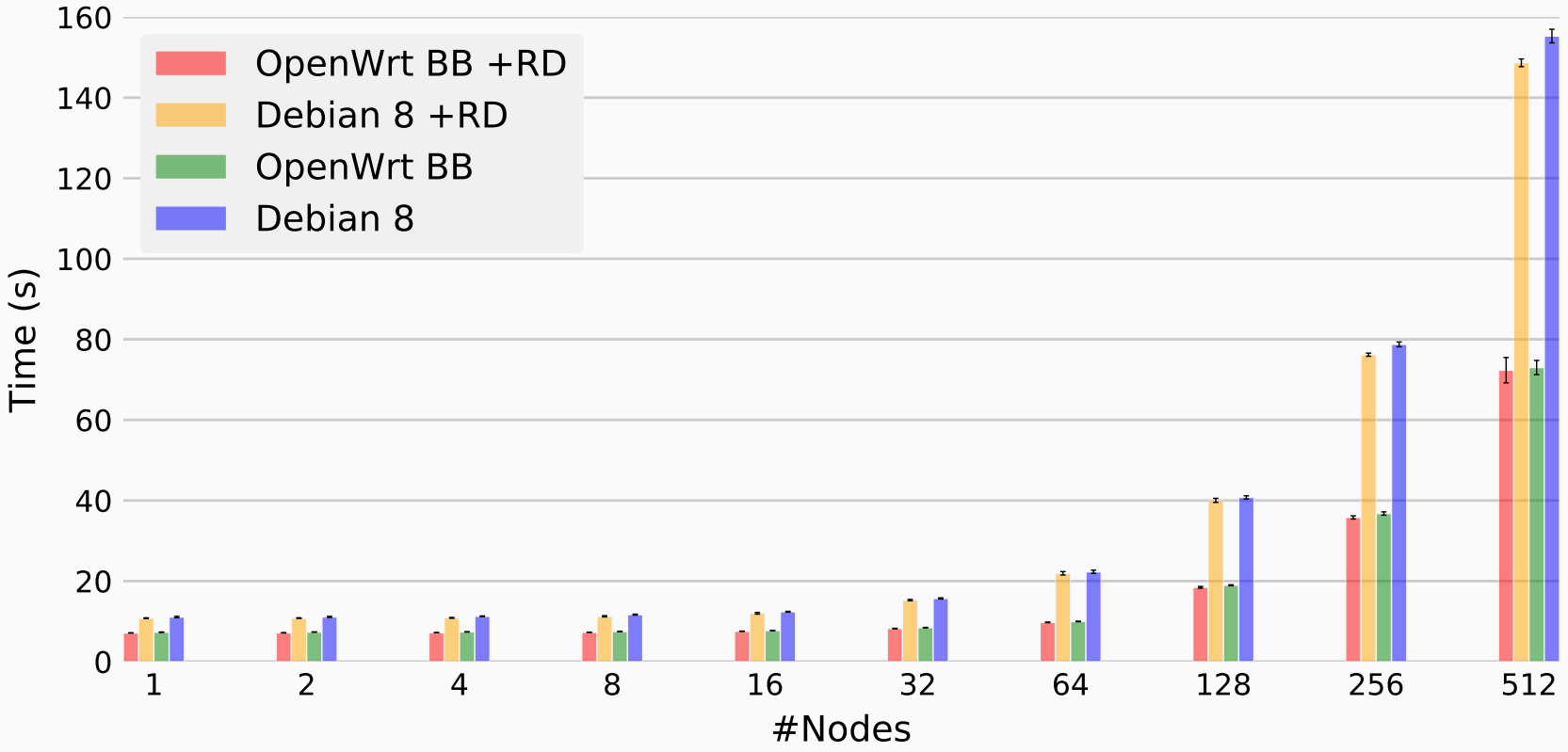


Figure 11: OpenWrt vs. Debian 8 Boot Times. OpenWrt RAM: 32MiB. Debian RAM: 256MiB. Boot Mode: *Selectors Shell Prompt*

Evaluation: Image Comparison

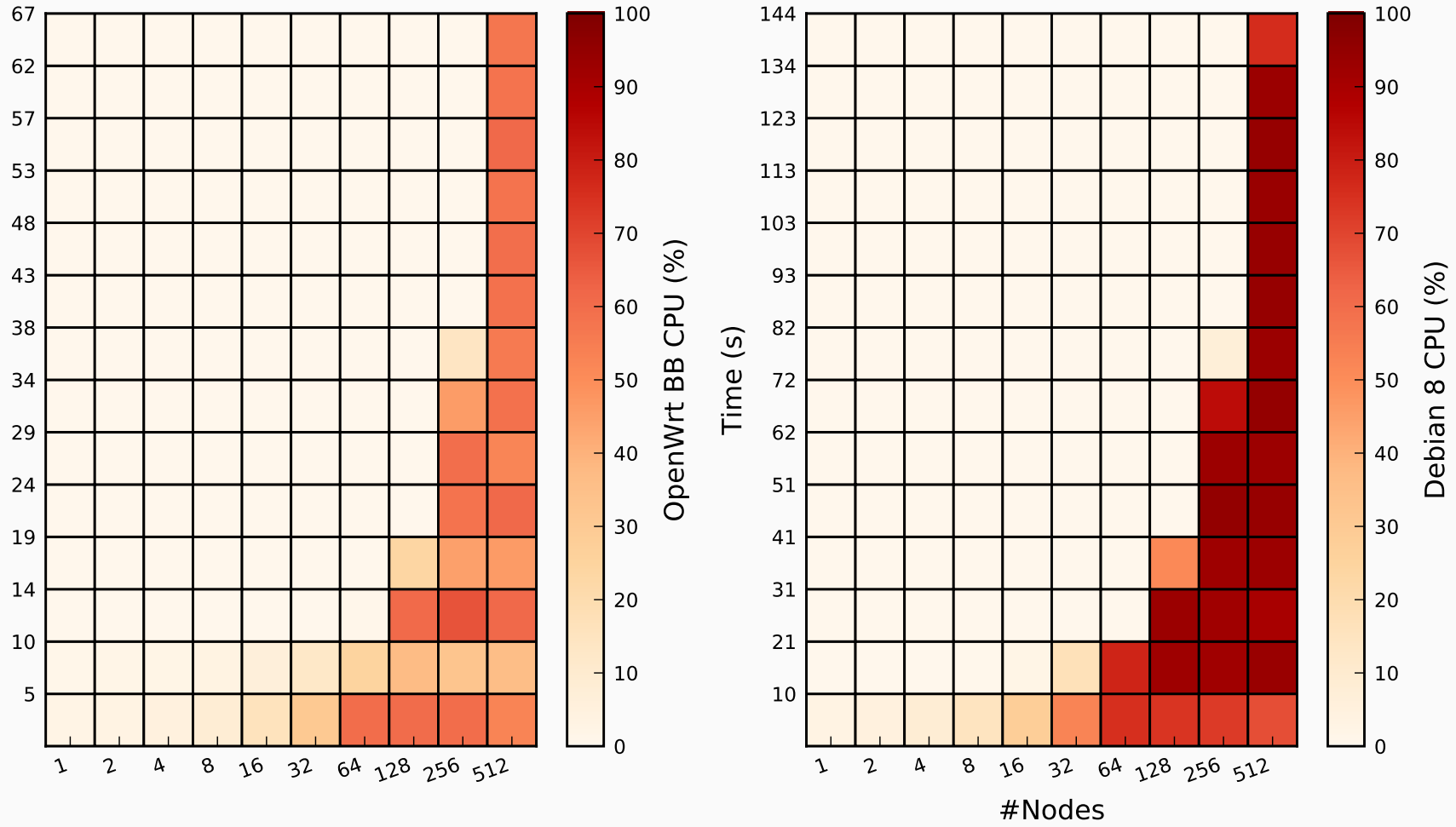


Figure 12: Boot Times CPU Usage (+RD): OpenWrt Barrier Braker vs. Debian 8

Evaluation: Snapshot Boot Mode

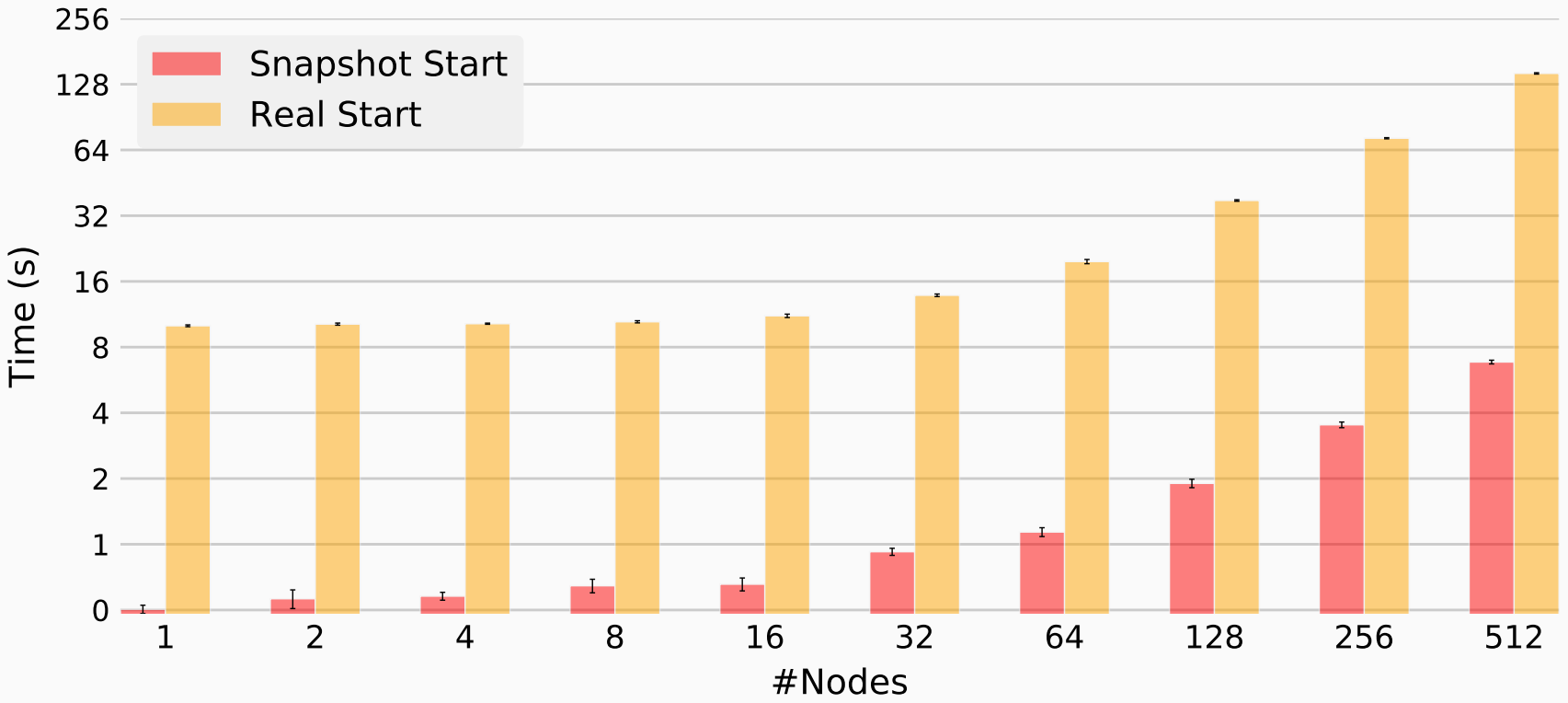


Figure 13: Boot Modes Debian

Evaluation: Snapshot Boot Mode

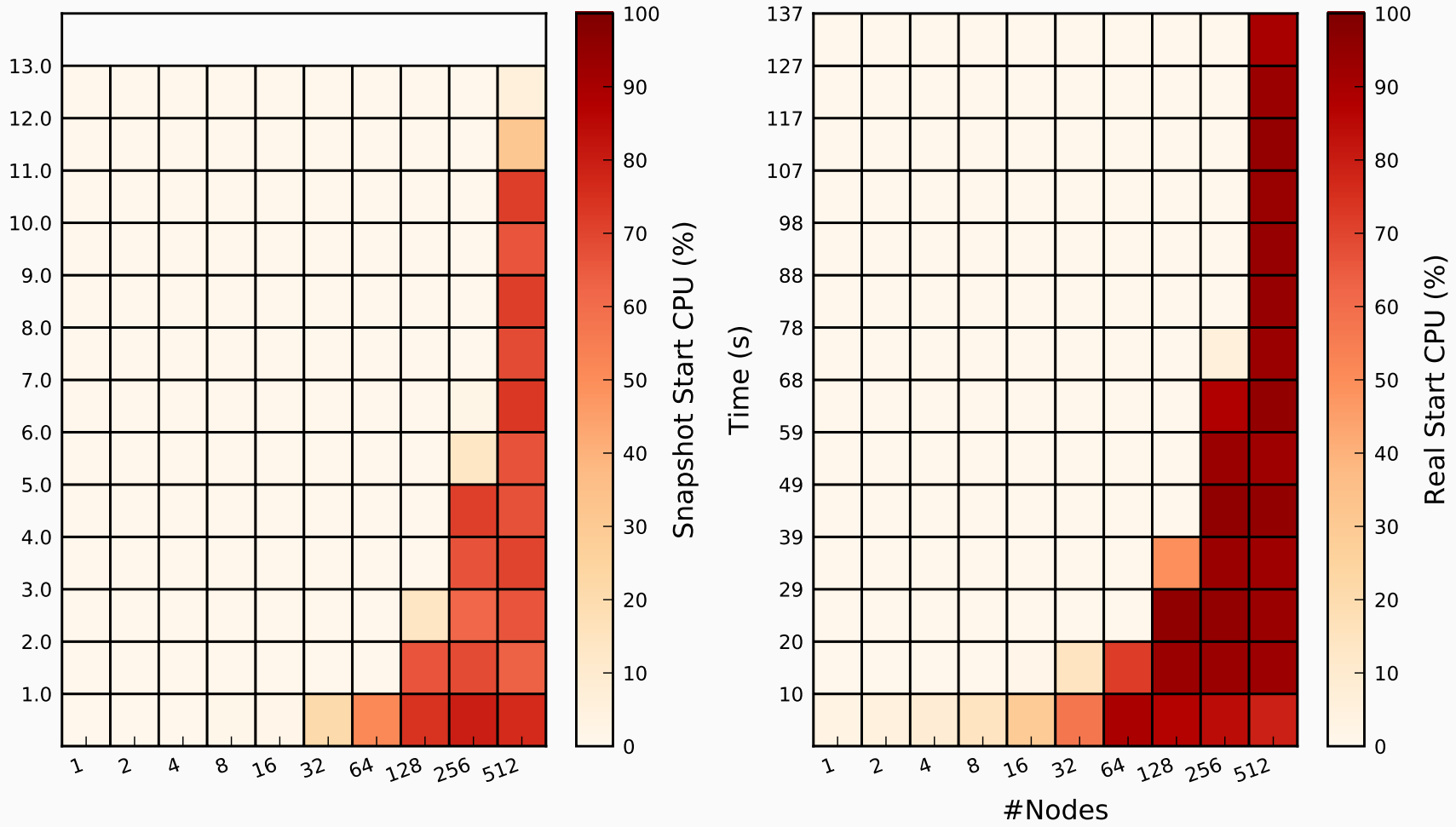


Figure 14: Boot Modes Debian CPU

Evaluation: Qemu NIC Models

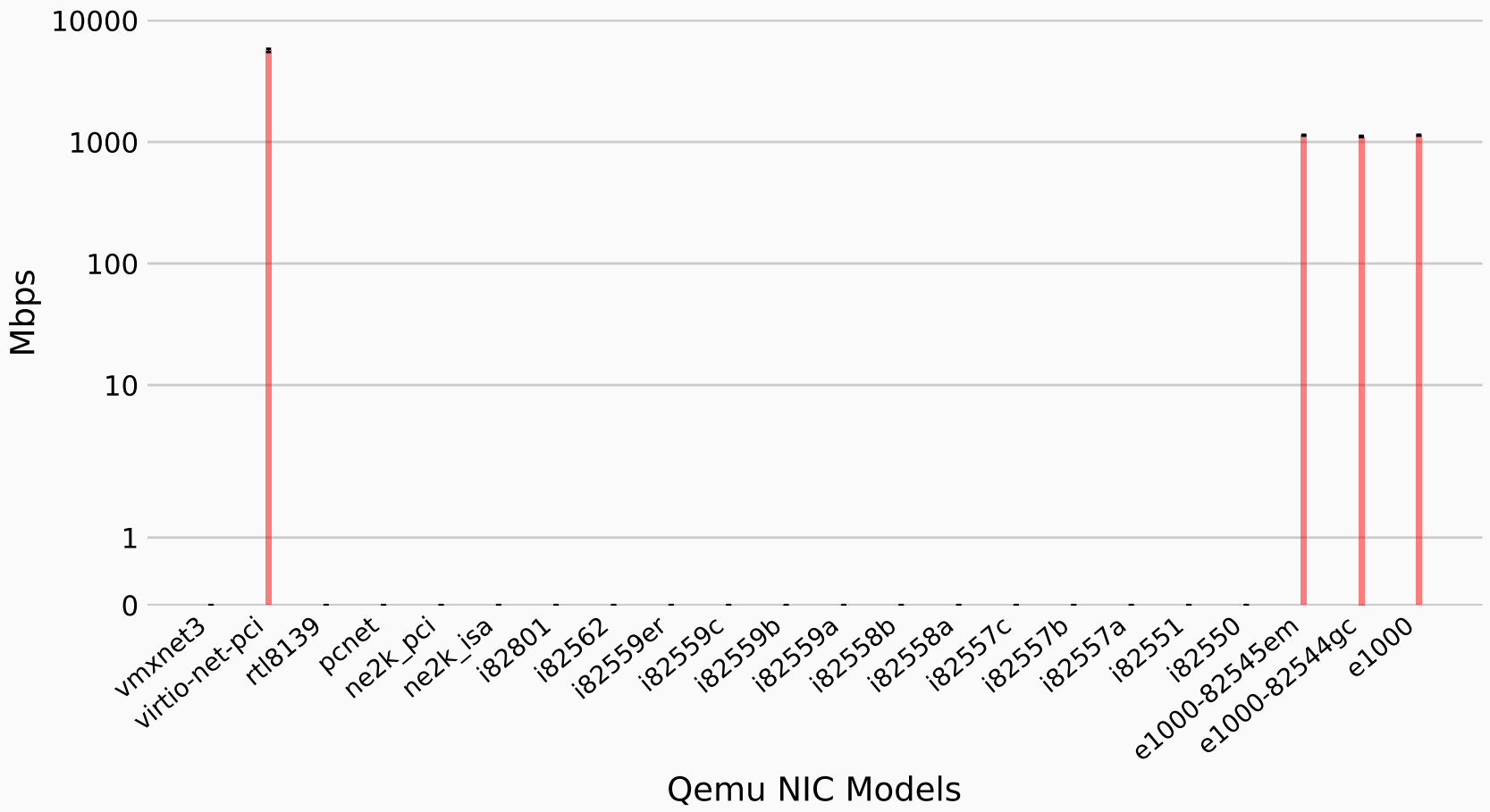


Figure 15: Qemu NIC Models Bandwidth OpenWRT BB

Evaluation: Outline

Evaluation: Network Backends Bandwidth

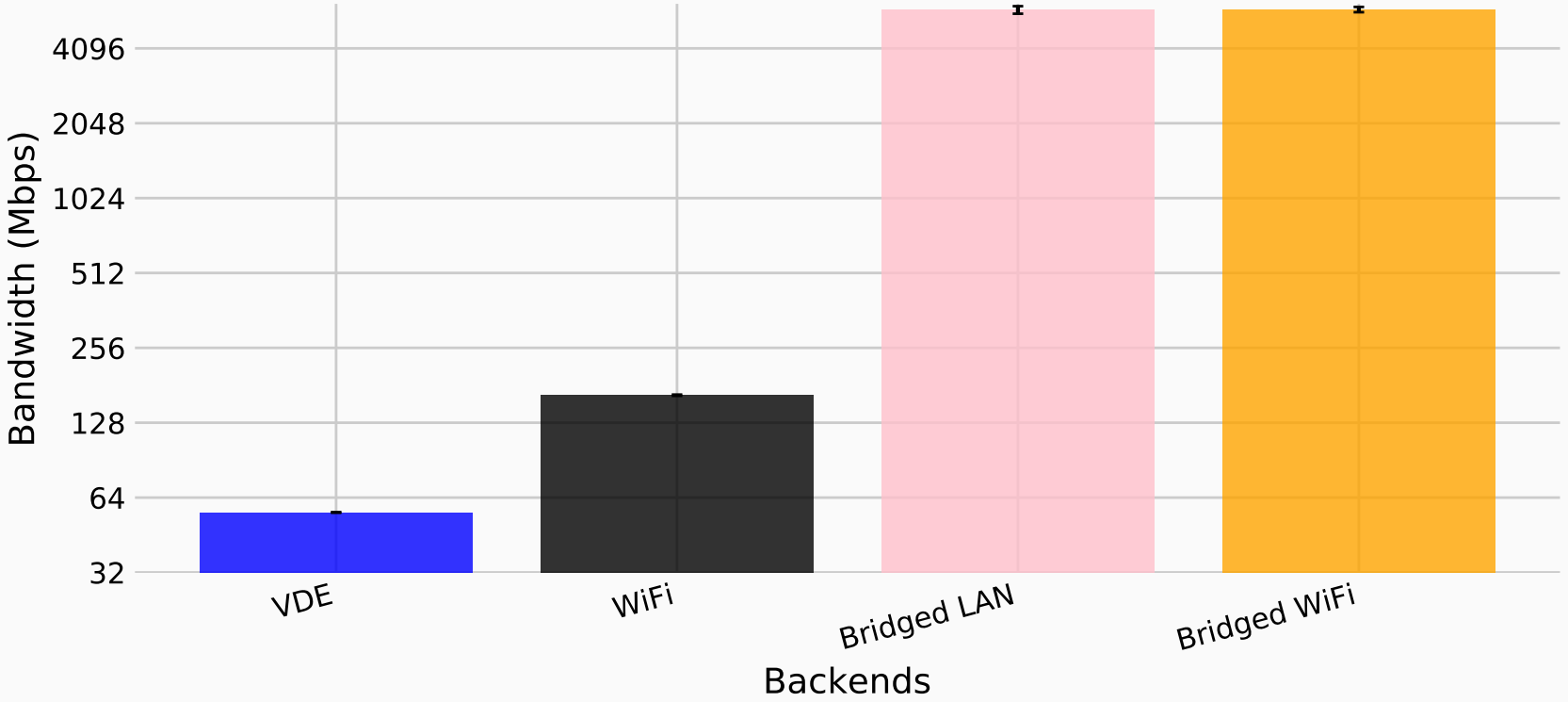


Figure 16: Network Backend Throughput

Evaluation: Network Backends RTT

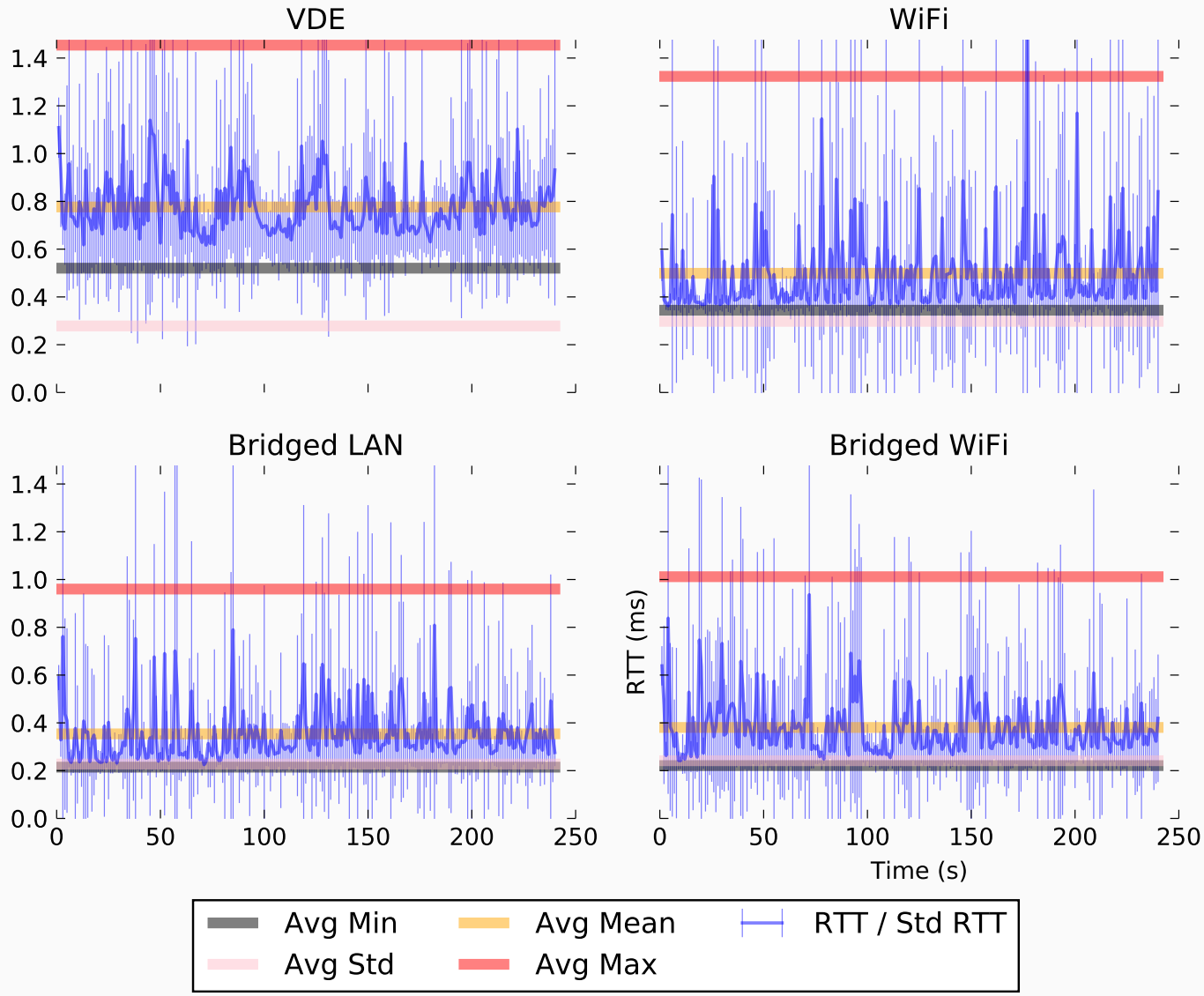


Figure 17: Network Backend Delays

Evaluation: Network Backend WiFi

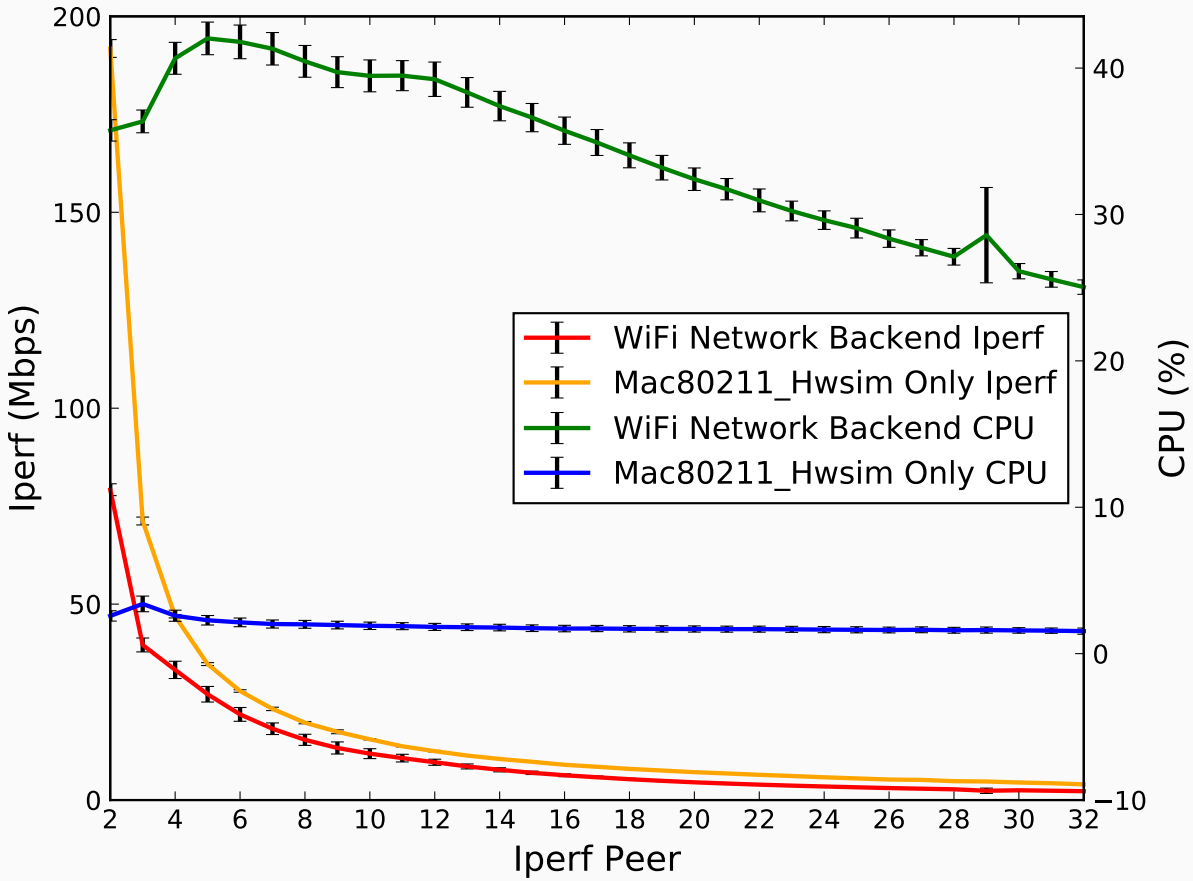
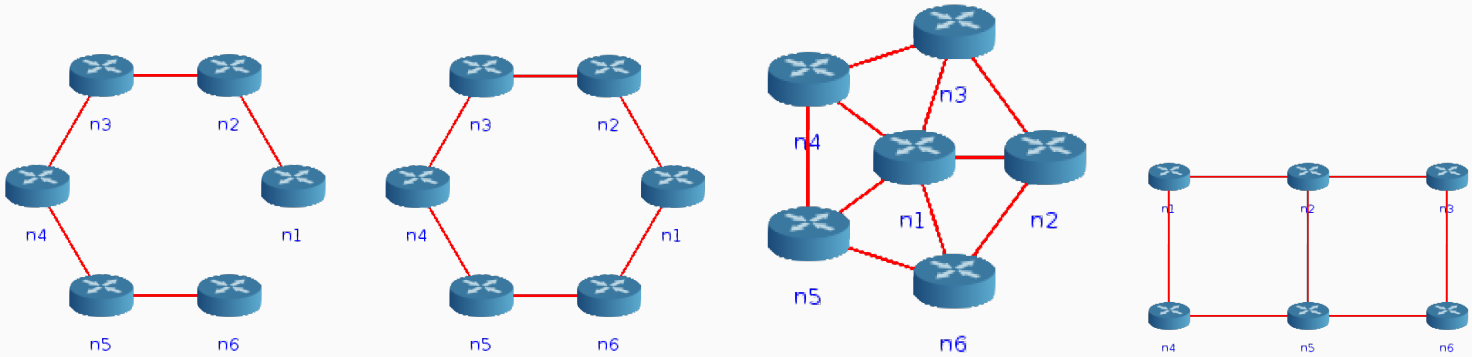


Figure 18: Distributed mac80211_hwsim Showcase

Evaluation: Differential Connection Switching



(a) Chain 6 **(b)** Cycle 6 **(c)** Wheel 6 **(d)** Grid 6

Figure 19: Differential Switching Network Topologies

Evaluation: Differential Connection Switching

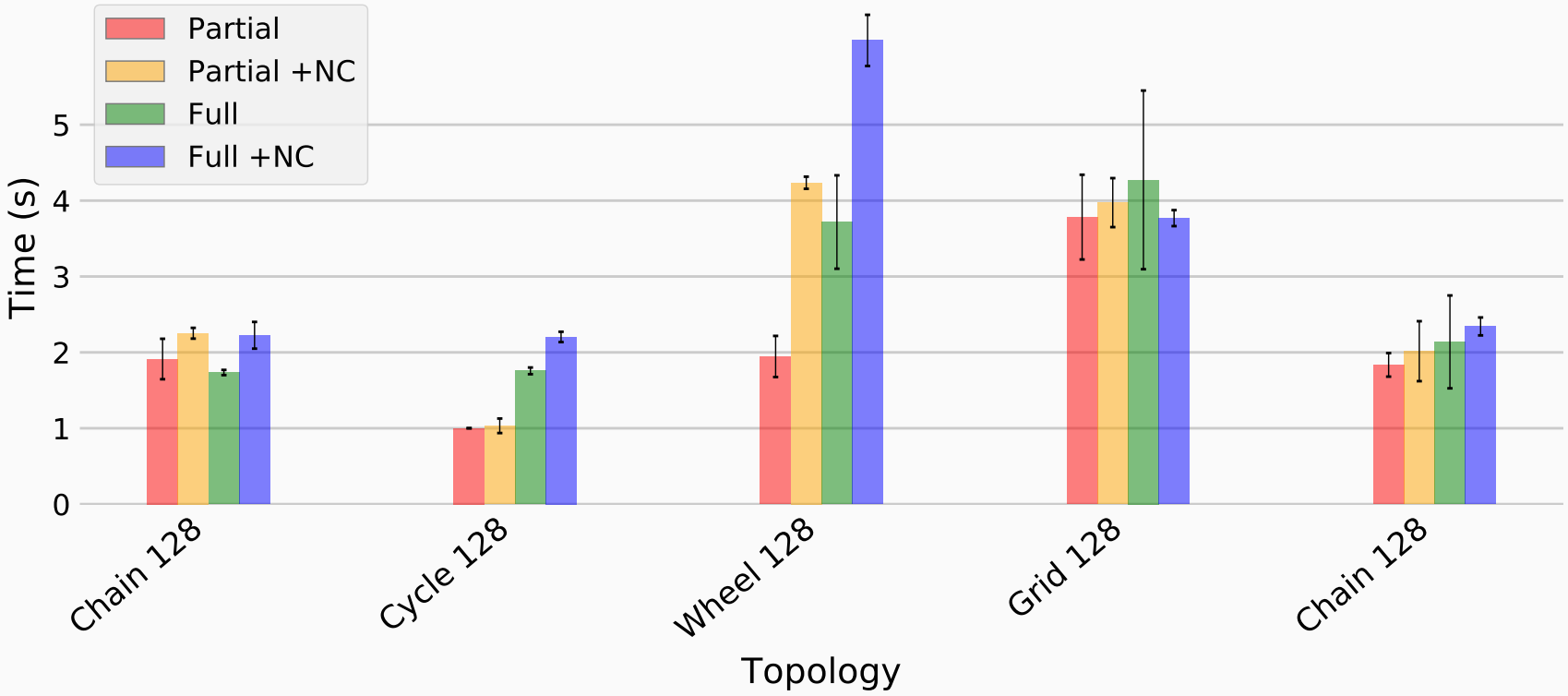


Figure 20: Network Switching Topologies (WiFi Network Backend)

Evaluation: Differential Connection Switching

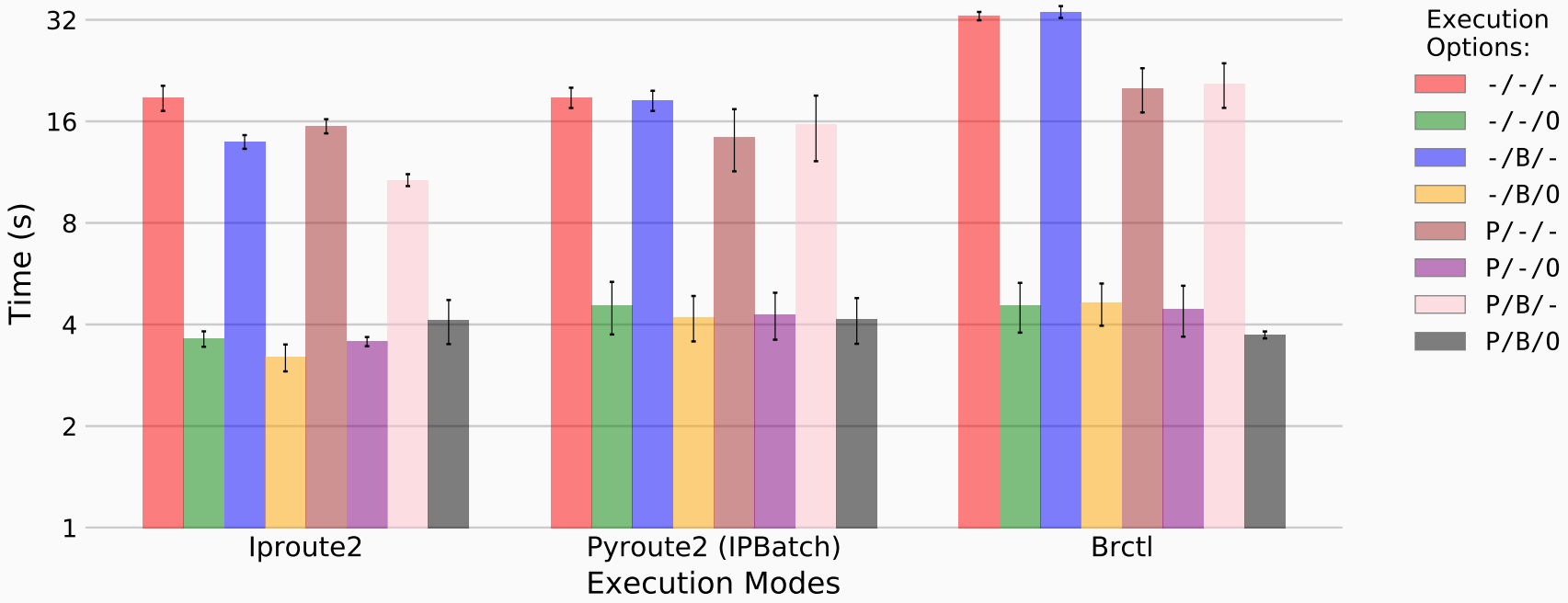


Figure 21: Network Switching Modes (Bridged WiFi Network Backend)

Evaluation: Differential Connection Switching

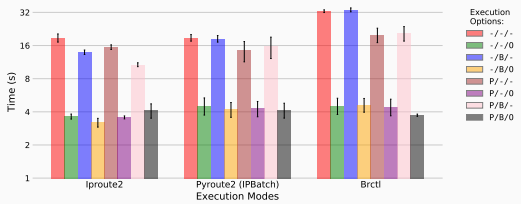


Figure 21: Network Switching Modes (Bridged WiFi Network Backend)

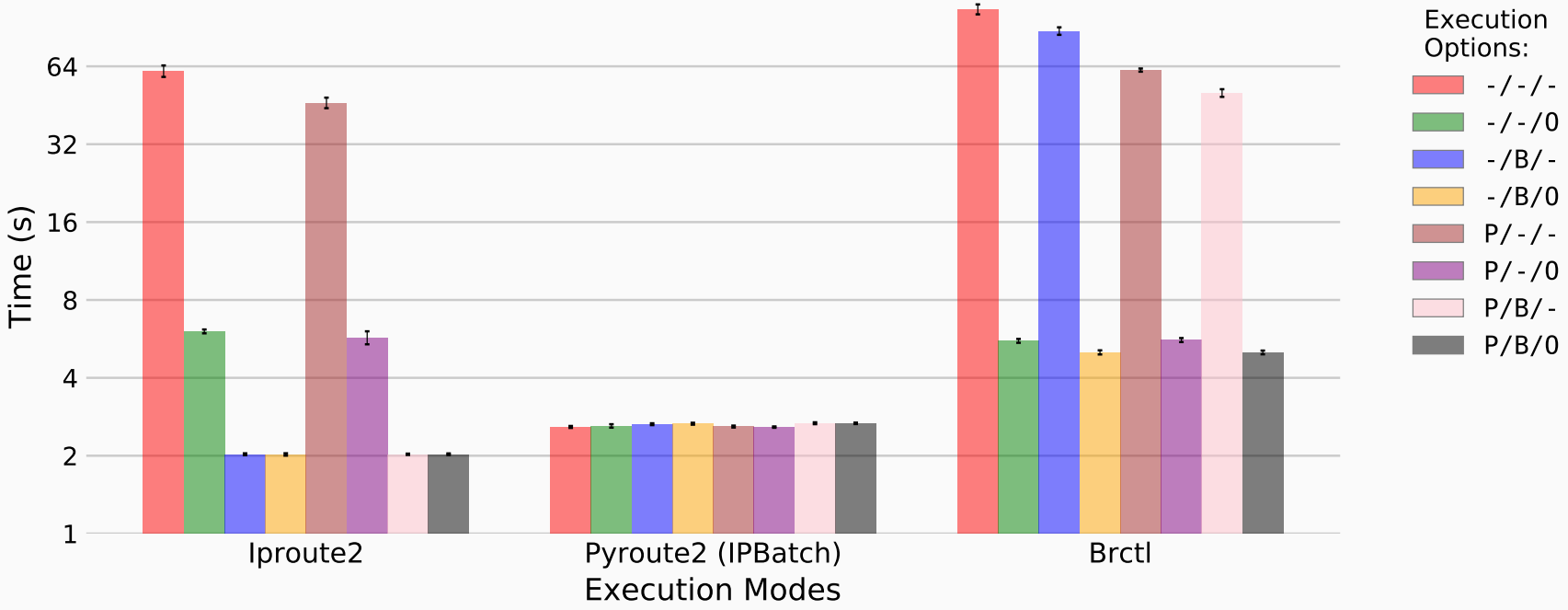


Figure 22: Network Switching Modes (Bridged LAN Network Backend)

Evaluation: Outline

Evaluation: Distributed Emulation

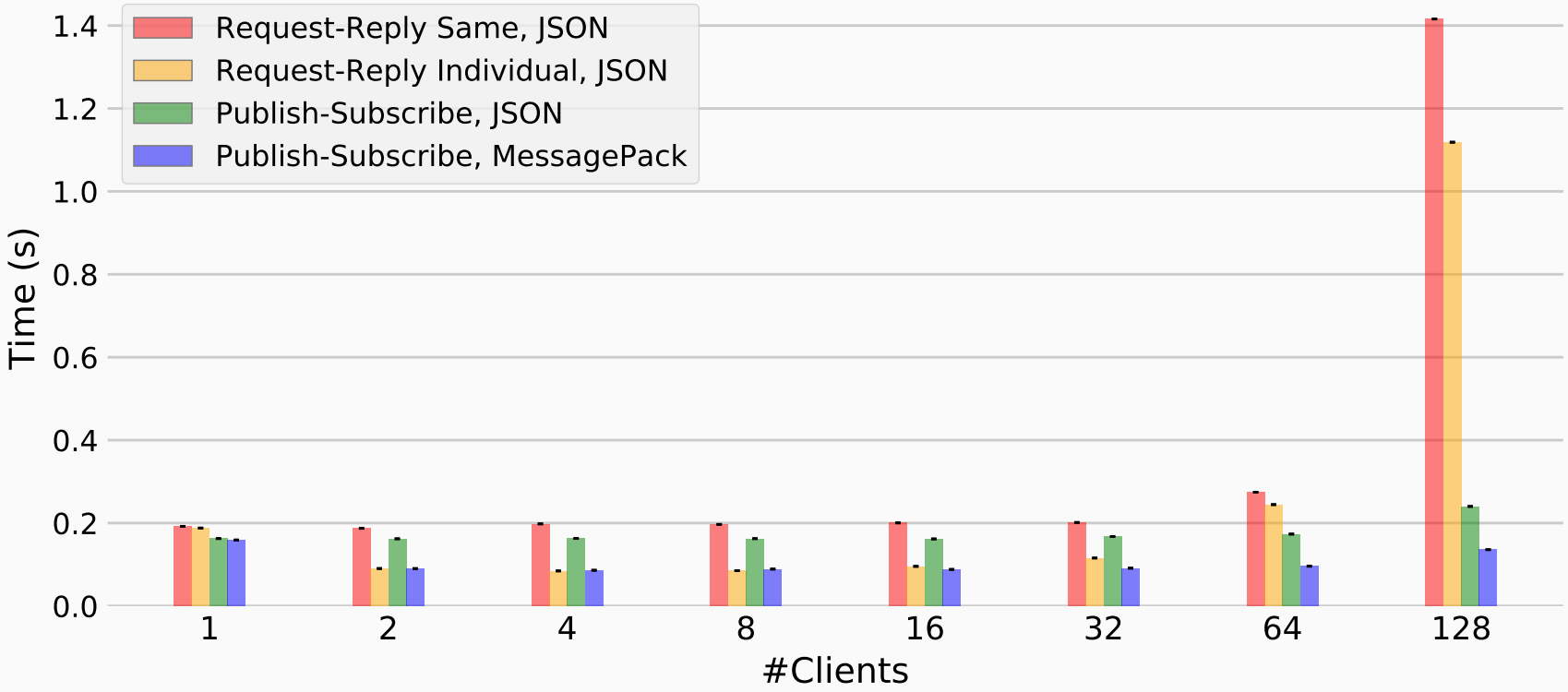


Figure 23: Distributed Coordination Step Times

Evaluation: Distributed Emulation

Resource	Value
Host OS	Ubuntu 16.04.1 LTS
Docker OS	Ubuntu 14.04.5 LTS
Virtualization Technology	KVM
Kernel	4.4.0-38-generic
RAM	2x DIMM DDR3 Synchronous 1600 MHz 8GiB
CPU	Intel(R) Core(TM) i7-4771 CPU @ 3.50GHz
CPU Cores Physical/Virtual	4/8
Network	Intel Corporation Ethernet Connection I217-LM (rev 04)
Iproute2	Git release v4.2.0
Qemu	Git release v2.7.0

Table 3: Technical Data Test System 2

[5] `https://github.com/svinota/pyroute2`. Commit ID: commit 13f1adb1ab2d5ca8927f1eb618bbb93170b61c33

Evaluation: Distributed Emulation

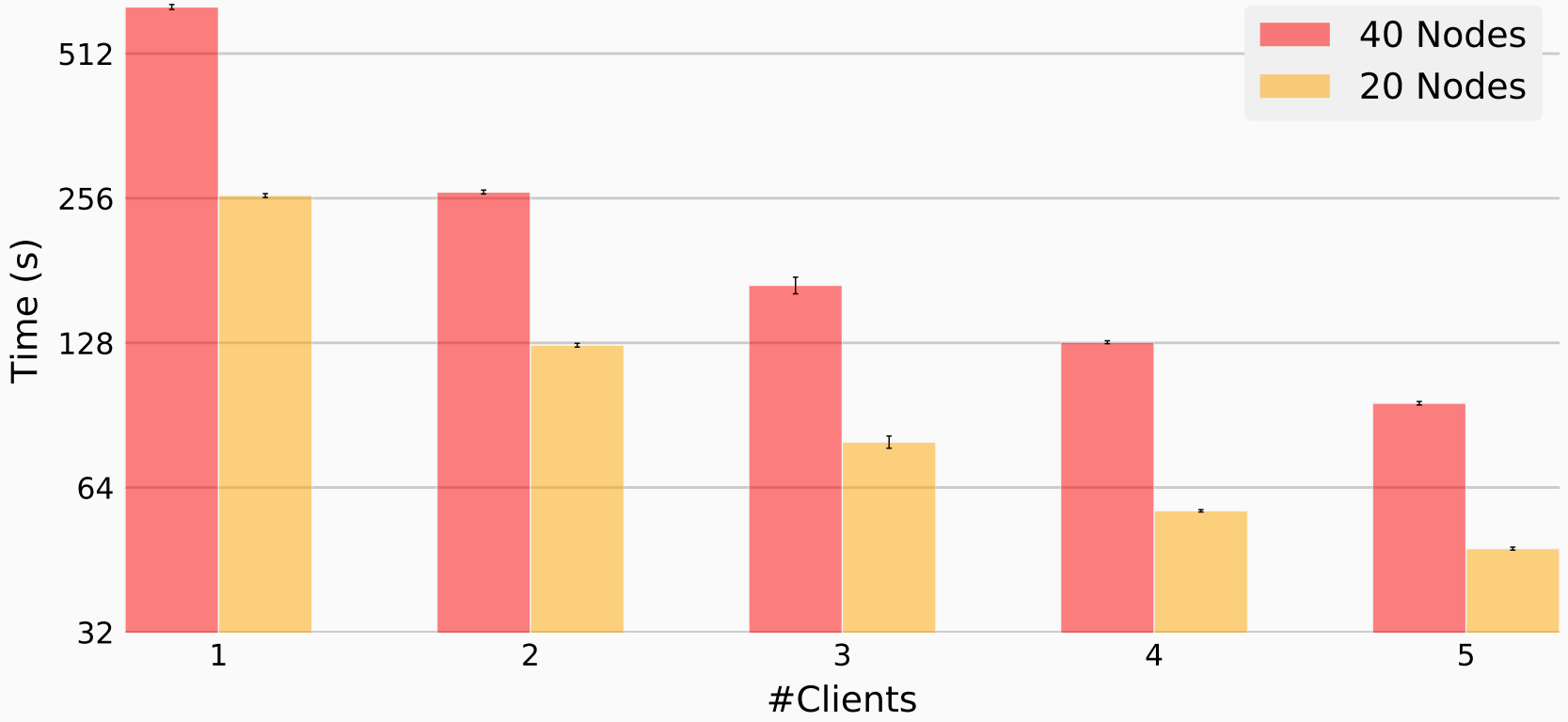


Figure 24: Distributed Boot Times, 40 OpenWrt BB Nodes, RamDisk, 128MB RAM, Selectors Boot Prompt, NodeDistributionEqual, Test System 2

Evaluation: Distributed Emulation

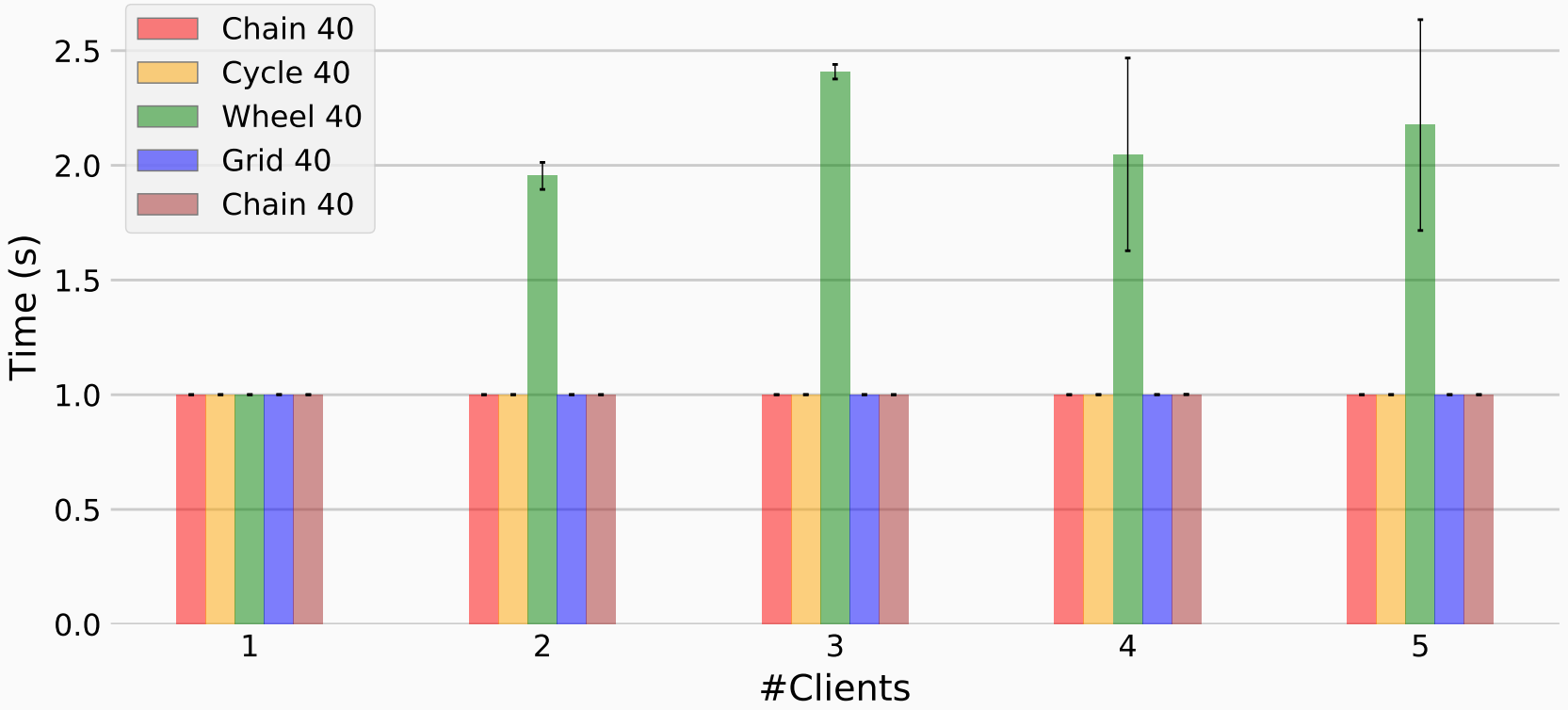


Figure 25: Distributed Differential Network Switching, 40 OpenWRT BB Nodes, Test System 2, Fixed-Range Model, Bridged WiFi Network Backend, No Link Quality Impairment, 2 Runs Only

Evaluation: Distributed Emulation

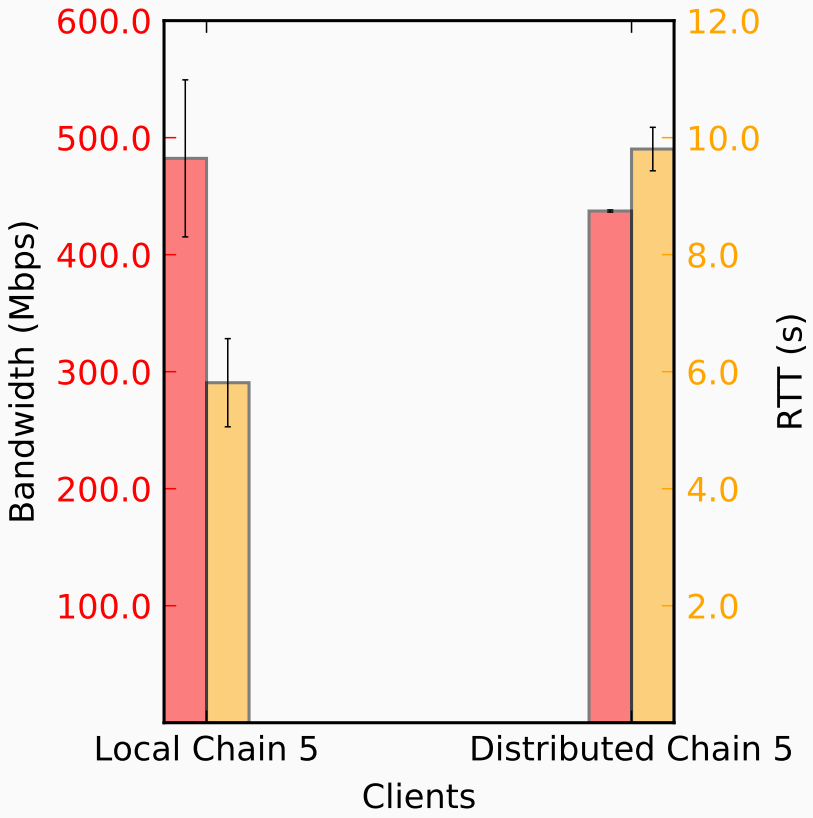


Figure 26: Distributed Mode: Tunnel Overhead, 5 OpenWRT BB Nodes, Test System 2, Fixed-Range Model, Bridged WiFi Network Backend, No Link Quality Impairment

Evaluation: Serval Study

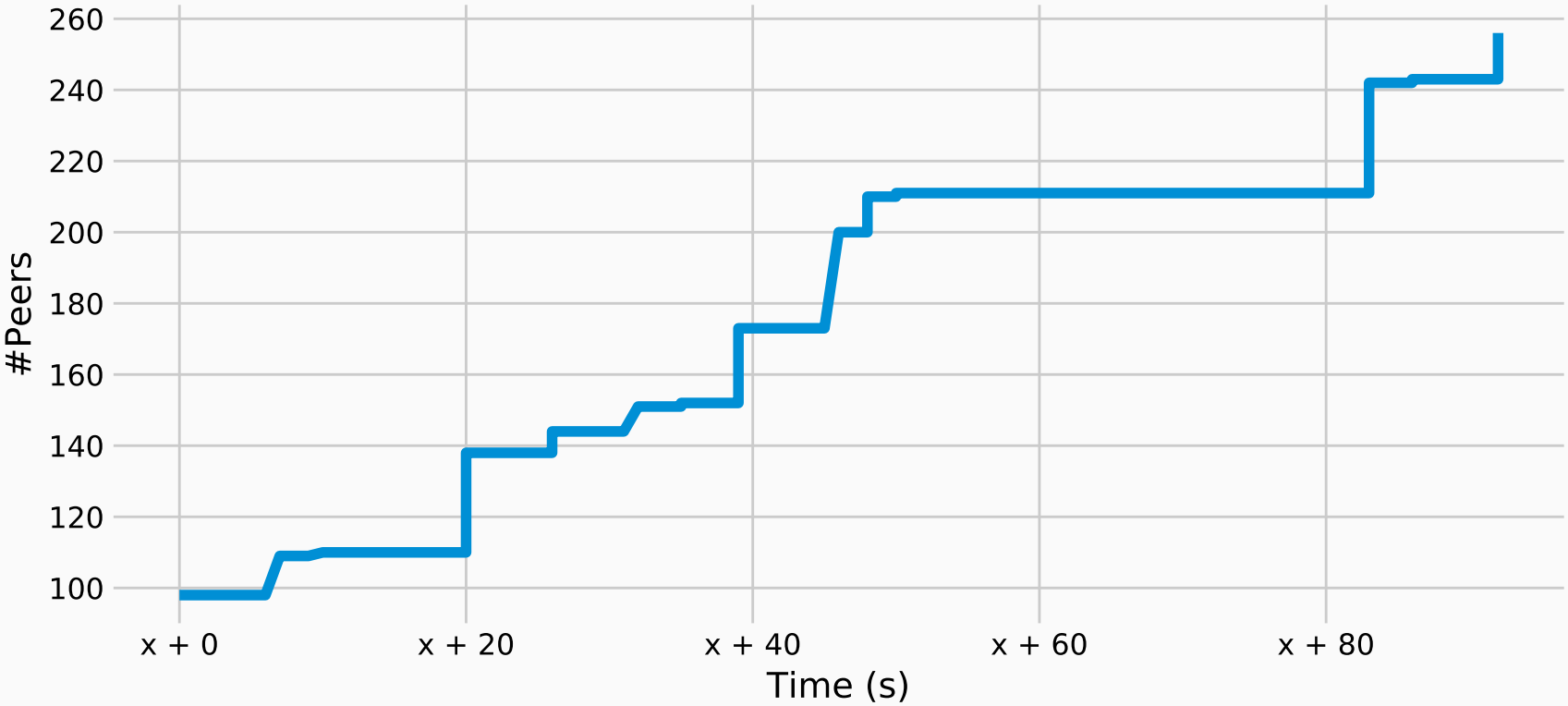


Figure 27: Serval Peer Discovery, Distributed Chain 256, 7 Computers

Evaluation: Serval Study

Client	Hostname	Bogomips	Free RAM (MB)	#Nodes
1	Andro*	55870.56	15117	23
2	BigBox	**294406.4**	**225214**	**120**
3	Rechenschieber	55871.04	29605	23
4	Andro*	55870.56	15051	23
5	Andro*	55870.56	15529	23
6	Andro*	55870.56	15232	23
7	Andro*	55870.56	15135	21

Table 4: Cluster Resources

Evaluation: Serval Study

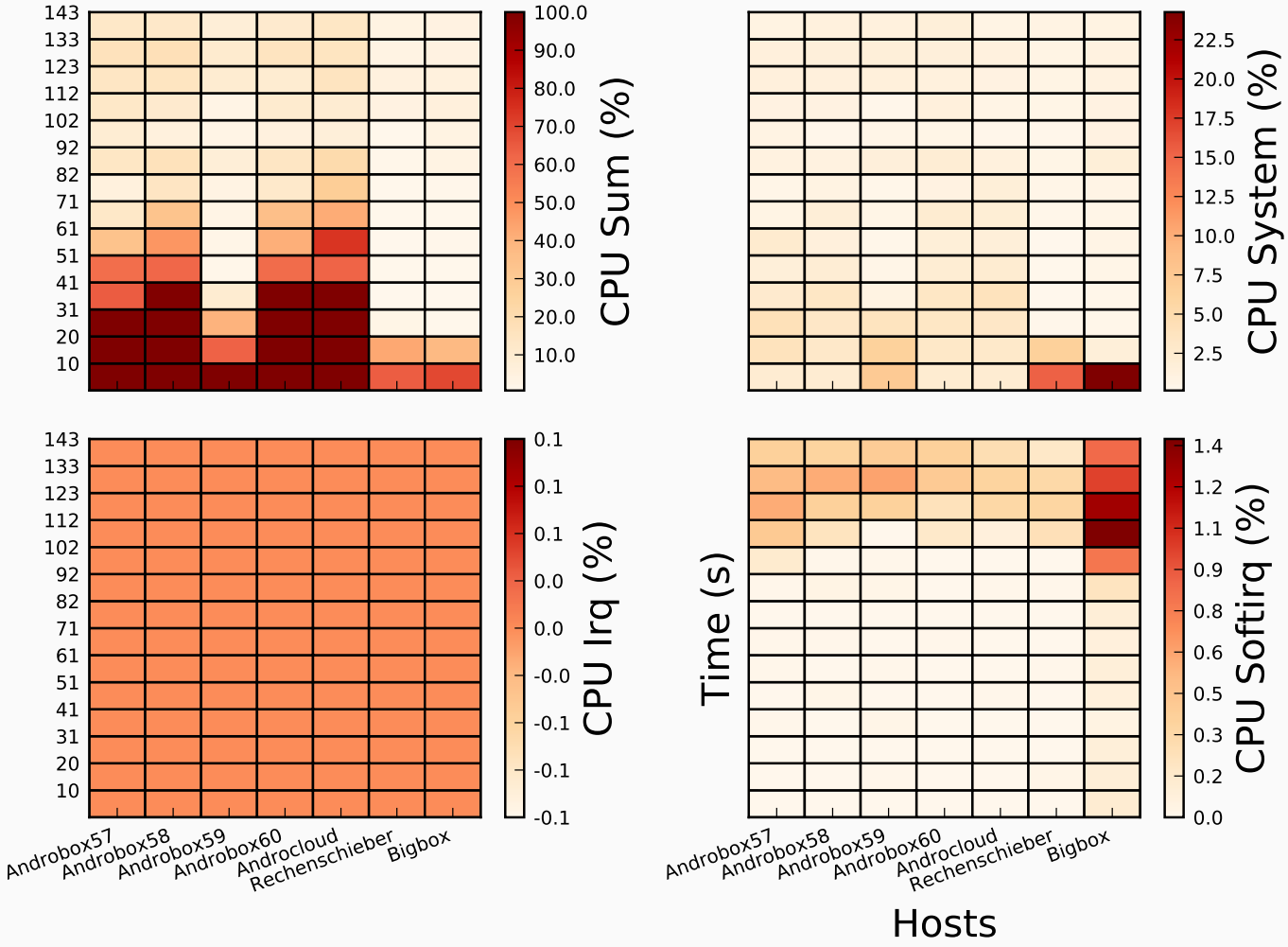


Figure 28: CPU Usage Cluster Serval Chain 256

Evaluation: Serval Study

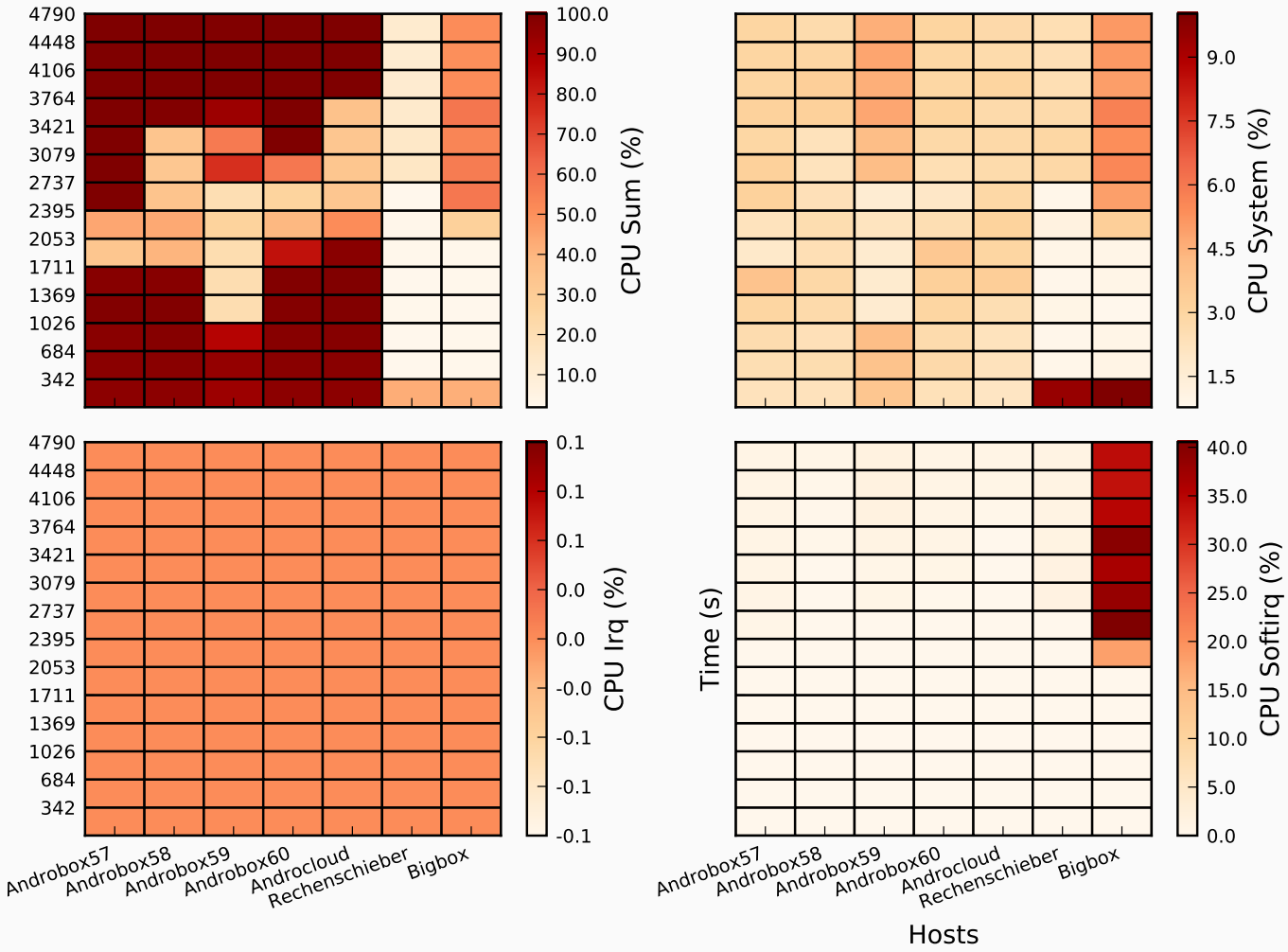


Figure 29: CPU Usage Cluster Serval Chain 512

Conclusion: Outline

Conclusion: MiniWorld

- Distributed network emulation framework
- Both wired- and wireless network emulation
- Static & event-based link-impairment
- Fast differential connection switching
- KVM enables every OS, software to be tested
- Snapshot boot mode for fast scenario-repetitions

Future Work: Outline

Future Work: MiniWorld

- High-fidelity link-emulation (ns-3, EMANE)
- Real mobile Android nodes
- Lightweight virtualization for some cases
- Link-impairment & better bandwidth for WiFi network backend
- Fix VDE network backend bandwidth and delay problems with big number of switch ports

Questions?

BACKUP SLIDES

Outline

Outline

Network Backends Comparison

Property	WiFi	VDE	Bridged LAN	Bridged WiFi
WiFi NIC	y	-	-	
Mobility Pattern	-	All	CORE	All
Link Quality Model*	-	1	2,3	2,3
Broadcast Domain	y	y	-	y
Interfaces: Management/Hub	-/-	y/y	y/y	y/y
Distributed Mode	?	-	Gretap/VLAN/VXLan	Gretap/VLAN/VXLan
Link Impairment	-	Wirefilter	TC (HTB+Netem)	TC (HTB+Netem)
Connection Switching	-	Wirefilter	Iproute2/Pyroute2/Brctl	Iproute2/Pyroute2/Brctl

Table 5: Network Backend Features

[5] The WiFi network backend has not been evaluated in the distribute mode at the time of writing, but multicast routing should enable the distributed communication of nodes.

Index	Link Quality Model
1	Fixed-Range
2	WiFi Simple Linear
3	WiFi Simple Linear

Table 6: Link Quality Models

Outline

Code

```
ebtables="ebtables --concurrent --atomic-file /tmp/MiniWorld/ebtables_atommic"

# clear tables
$ebtables --atomic-init
$ebtables --atomic-commit

# create chains
$ebtables --atomic-save
$ebtables -N wifi1 -P DROP
$ebtables -A FORWARD --logical-in wifi1 -j wifi1
$ebtables --atomic-commit

# layer2 firewall
$ebtables -atomic-save
$ebtables -I wifi1 -i tap_00001_1 -o tap_00002_1 -j mark --set-mark 1 --mark-target ACCEPT
$ebtables -I wifi1 -i tap_00002_1 -o tap_00001_1 -j mark --set-mark 1 --mark-target ACCEPT
$ebtables -I wifi1 -i tap_00002_1 -o tap_00003_1 -j mark --set-mark 2 --mark-target ACCEPT
$ebtables -I wifi1 -i tap_00003_1 -o tap_00002_1 -j mark --set-mark 2 --mark-target ACCEPT
$ebtables --atomic-commit

# change network topology
ip -d -batch - <<<"link add name wifi1 type bridge
link set dev wifi1 group 2
link set dev wifi1 type bridge ageing_time 0
link set dev tap_00001_1 master wifi1
link set dev tap_00001_1 up
link set dev wifi1 up
link set dev tap_00002_1 master wifi1
link set dev tap_00002_1 up
link set dev tap_00003_1 master wifi1
link set dev tap_00003_1 up"
```

Listing 1: Bridged WiFi Network Backend Chain 3 Setup

Link Quality Models

	WiFi Simple Linear Model			WiFi Simple Exponential Model		
Distance	Bandwidth	Delay Const	Delay Var	Bandwidth	Delay Const	Delay Var
0	54000	0	0.0	54000	1	0.1
1	54000	0	0.0	54000	1	0.1
2	36000	1.0	0.1	54000	1	0.1
3	27000	2.0	0.2	54000	1	0.1
4	21600	3.0	0.3	27000	2	0.2
5	18000	4.0	0.4	27000	2	0.2
6	15429	5.0	0.5	27000	2	0.2
7	13500	6.0	0.6	27000	2	0.2
8	12000	7.0	0.7	13500	4	0.4
9	10800	8.0	0.8	13500	4	0.4
10	9818	9.0	0.9	13500	4	0.4
11	9000	10.0	1.0	13500	4	0.4
12	8308	11.0	1.1	6750	8	0.8
13	7714	12.0	1.2	6750	8	0.8
14	7200	13.0	1.3	6750	8	0.8
15	6750	14.0	1.4	6750	8	0.8
16	6353	15.0	1.5	3375	16	1.6
17	6000	16.0	1.6	3375	16	1.6
18	5684	17.0	1.7	3375	16	1.6
19	5400	18.0	1.8	3375	16	1.6
20	5143	19.0	1.9	1688	32	3.2
21	4909	20.0	2.0	1688	32	3.2
22	4696	21.0	2.1	1688	32	3.2
23	4500	22.0	2.2	1688	32	3.2
24	4320	23.0	2.3	844	64	6.4
25	4154	24.0	2.4	844	64	6.4
26	4000	25.0	2.5	844	64	6.4
27	3857	26.0	2.6	844	64	6.4
28	3724	27.0	2.7	422	128	12.8
29	3600	28.0	2.8	422	128	12.8

Table 7: Link Quality Models WiFi: (a) Linear Model (b) Exponential Model

Link Impairment

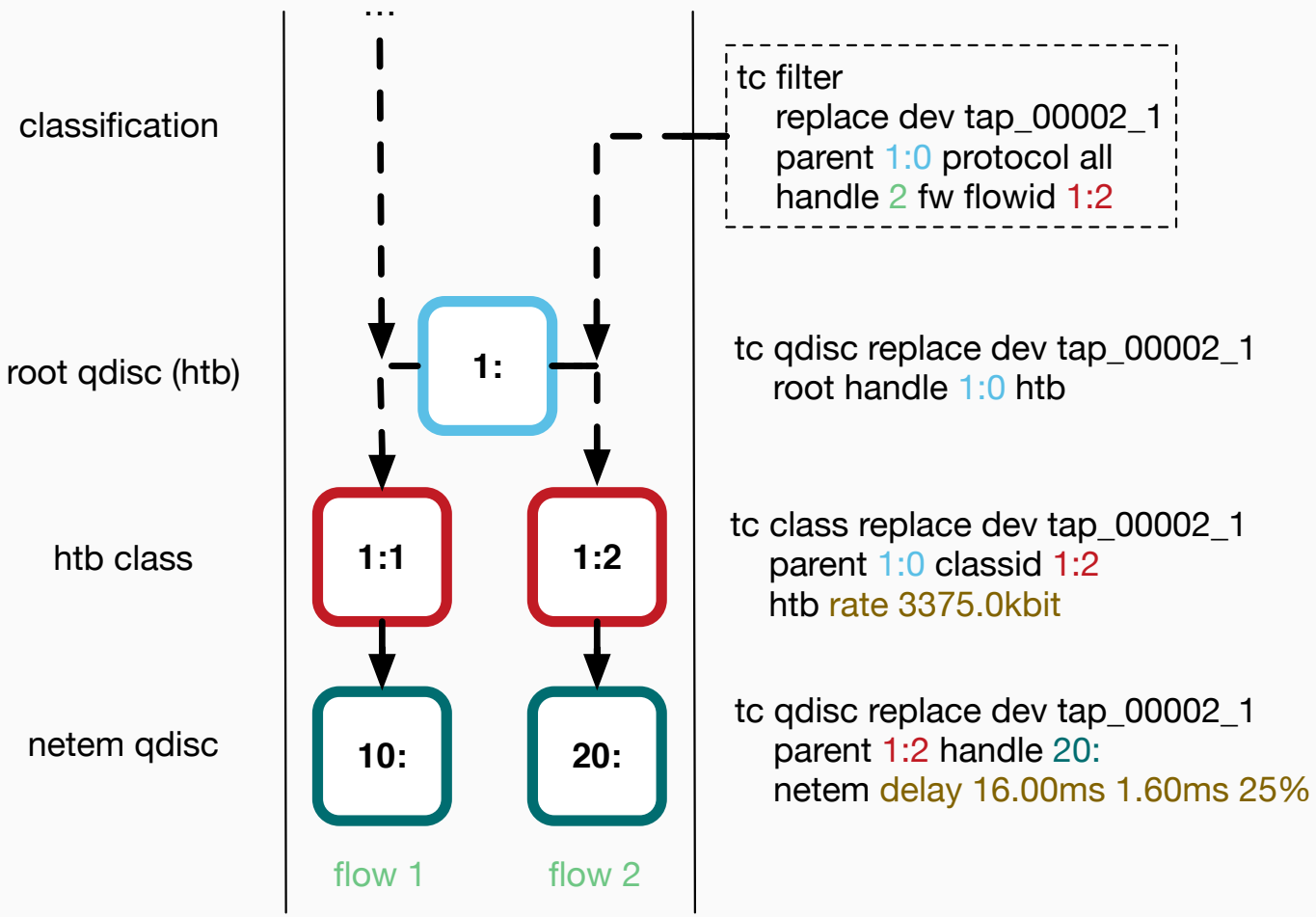


Figure 30: Traffic Shaping

Linux Netfilter Hooks

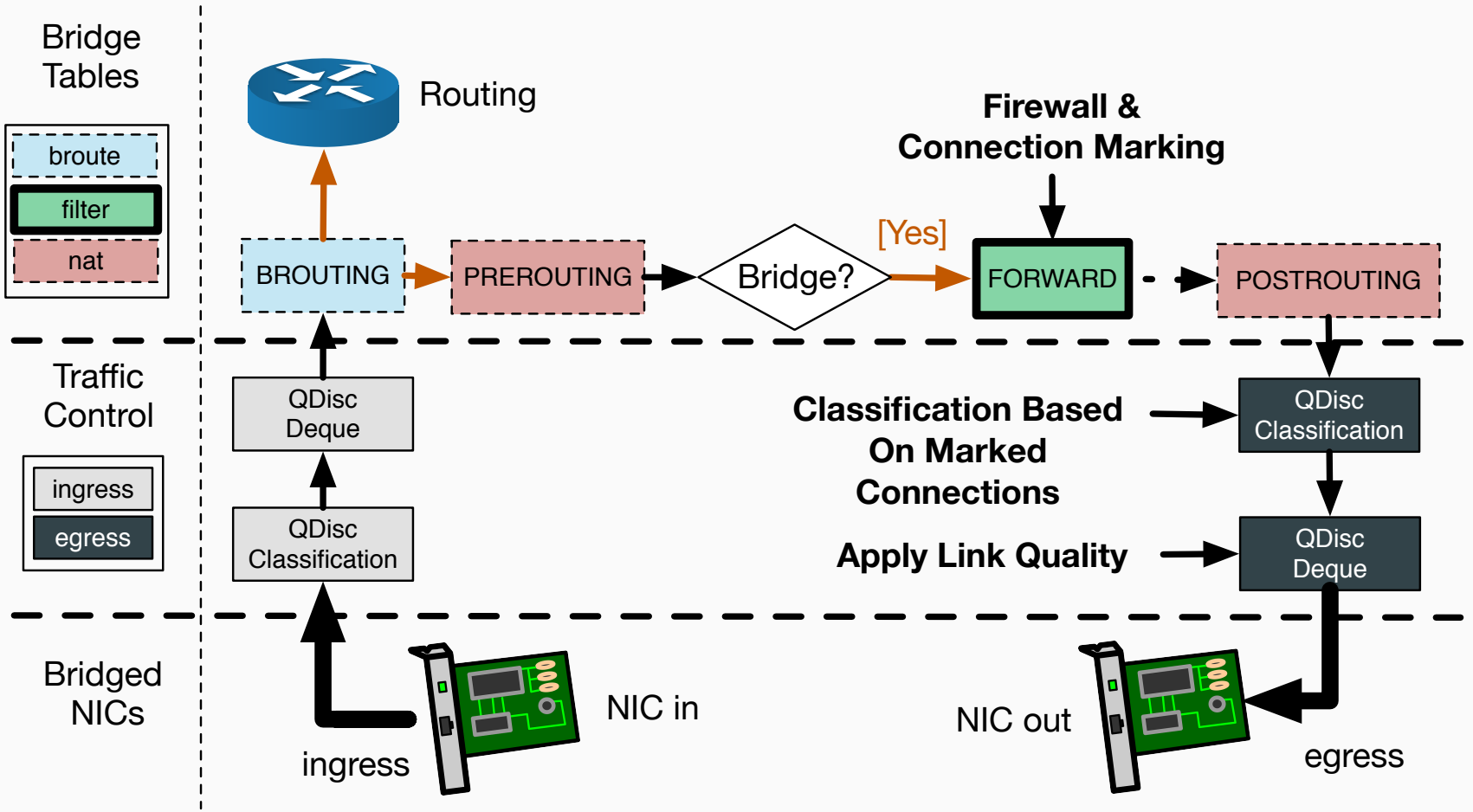


Figure 31: Traffic Shaping

Scenario Config

```
{
  "scenario" : "nbt_bridged_multi",
  "cnt_nodes" : 2,
  "provisioning" : {
    "image": "MiniWorld_Images/serval_paper/openwrt-x86-miniworld_dev_v55.qcow2",
    "regex_shell_prompt": "root@OpenWrt:/#",
    "regex_boot_completed": "procd: - init complete -.*",
    "shell" : {
      "pre_network_start": {
        "shell_cmds": [
          "ifconfig eth0 down",
          "ifconfig br-lan down",
          "brctl delbr br-lan",
          "ifconfig eth0 up",
          "iperf -s &"
        ]
      }
    }
  },
 "qemu" : {
   "ram": "1024M",
   "nic": {
     "model": "virtio-net-pci"
    }
  },
  "network" : {
    "backend" : {
      "name" : "bridged",
```