Hewlett Packard Enterprise

Extensible Flit-Level Simulation of Large-Scale Interconnection Networks

Nic McDonald, Adriana Flores, Al Davis,
Mikhail Isaev, John Kim, Doug Gibson

Why Reinvent the Wheel?

SuperSim: Features and Attributes

– Fast event-driven simulation
  – Only model things that change

– Single threaded (really is this a feature?)
  – Easy to use “run to completion” of each event.
  – Simulations achieve 50k to 5M events per second.

– Source code:
  – ~40,000 lines of code
  – ~400 source/header files
  – 10+ external libraries
  – Supported by many tools

“If a simulator already does what you want it to do, you're most likely asking the wrong questions.”

-Professor Christos Kozyrakis (Stanford CS/EE)

Settings and Configuration

– Extended JSON to configure a simulation

– Command line configuration modifiers

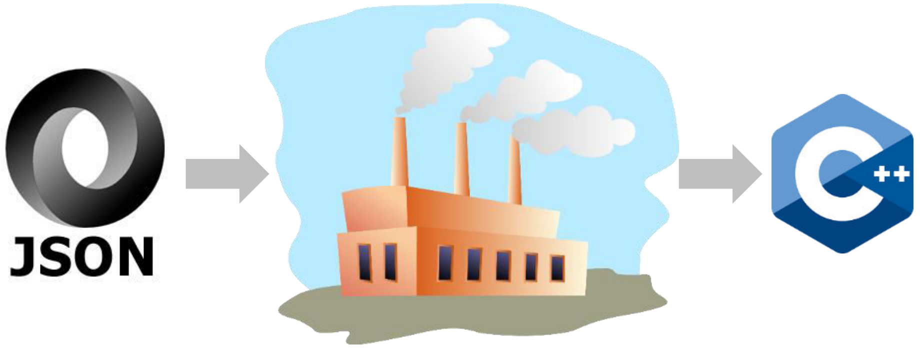
– Hierarchical nature of JSON matches the hierarchical structure of simulation:

```
"network": {
    "topology": "folded_clos",
    "levels": 3,
    "radix": 6,
    "protocol_classes": [{
        "num_vcs": 2,
        "routing": {
          "algorithm": "common_ancestor",
          "latency": 1,  // cycles
          "least_common_ancestor": true,
          "mode": "port",
          "adaptive": false }}],
    "router": {
      "architecture": "input_queued",
      "input_queue_depth": 100,
      "output_queue_depth": 164,
      "crossbar": { "latency": 8  // cycles },
      "vc_scheduler": {
        "allocator": {
          "type": "rc_separable",
          "slip_latch": true,
          "iterations": 2,
          "resource_arbiter": { "type": "lslp" },
          "client_arbiter": { "type": "lslp" }
        }},
    },
```

```
$ supersim myconfig.json \
> network.router.architecture=string=output_queued \
> network.levels=uint=4
```

Smart Object Factories

– Factories with zero-modify module inclusion
  – New model files can just be dropped in.
  – No code changes required to the code base.



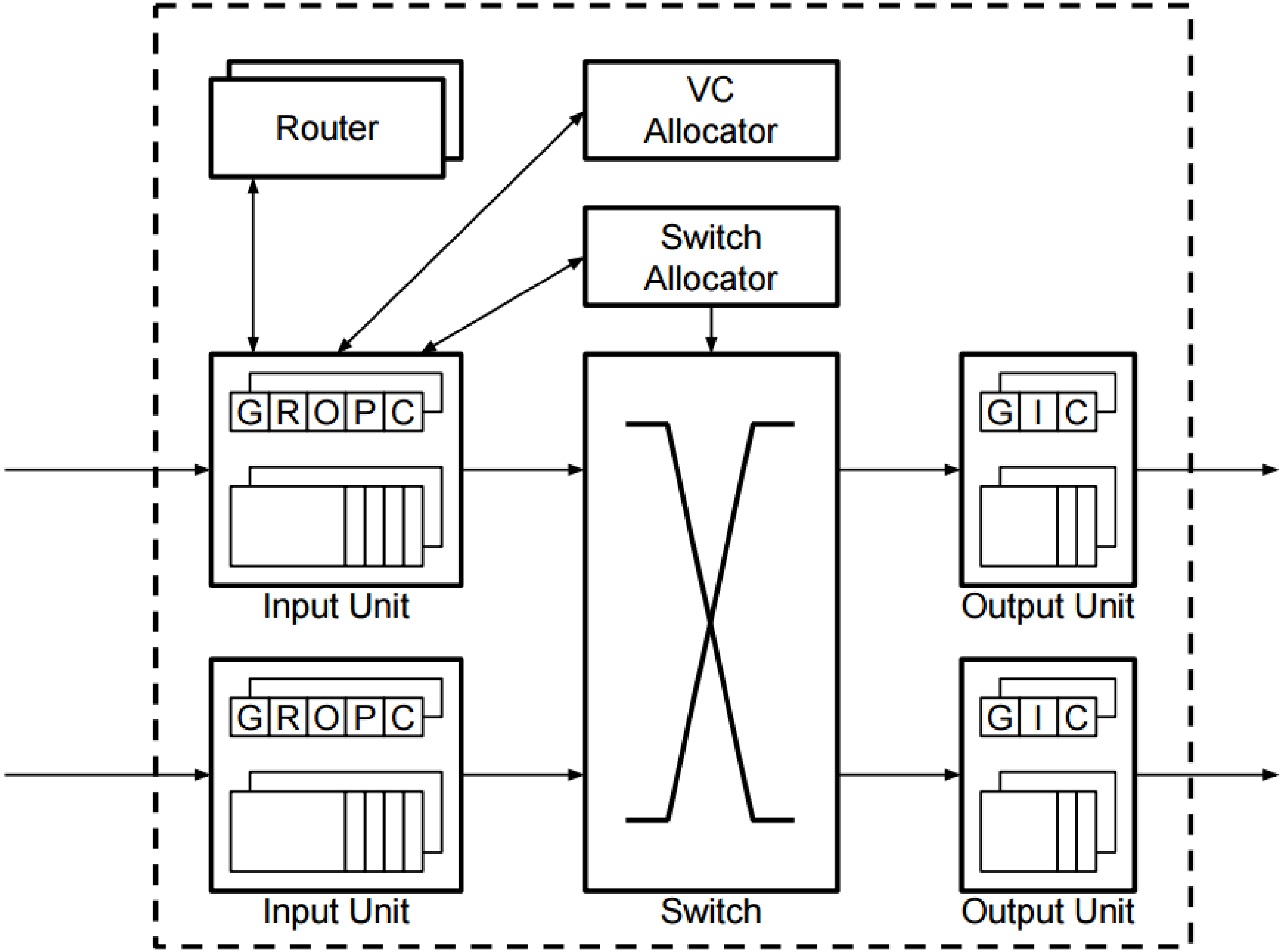
registerWithFactory()

```
#include "traffic/continuous/LoopbackCTP.h"
#include <factory/Factory.h>

LoopbackCTP::LoopbackCTP(
    const std::string& _name, const Component* _parent,
    u32 _numTerminals, u32 _self, Json::Value _settings)
    : ContinuousTrafficPattern(
          _name, _parent, _numTerminals, _self, _settings) {}

LoopbackCTP::~LoopbackCTP() {}

u32 LoopbackCTP::nextDestination() {
  return self_;
}

registerWithFactory(
    "loopback", ContinuousTrafficPattern,
    LoopbackCTP, CONTINUOUSTRAFFICPATTERN_ARGS);
```

Designed for Architectural Exploration and Validation

Use realistic architectural models

- Router pipelines
- Routing algorithms
- Credit management
- Congestion detection

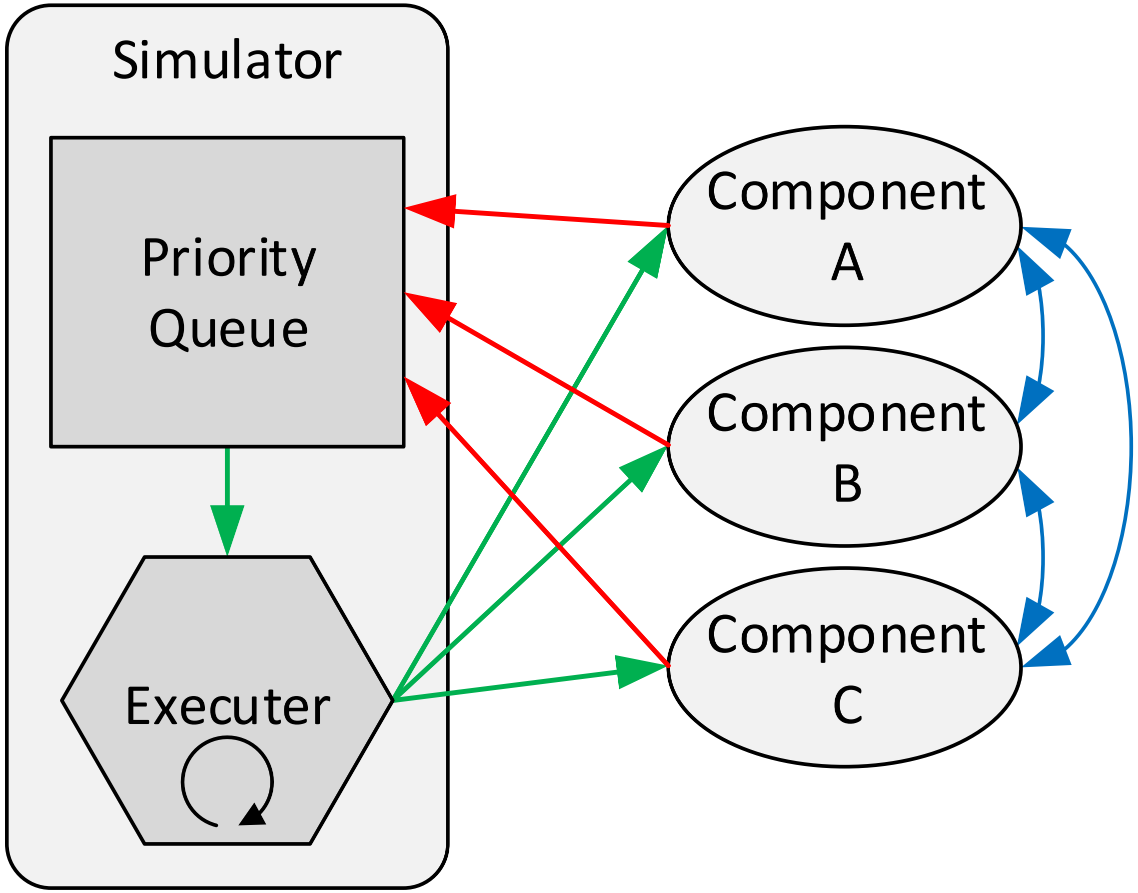


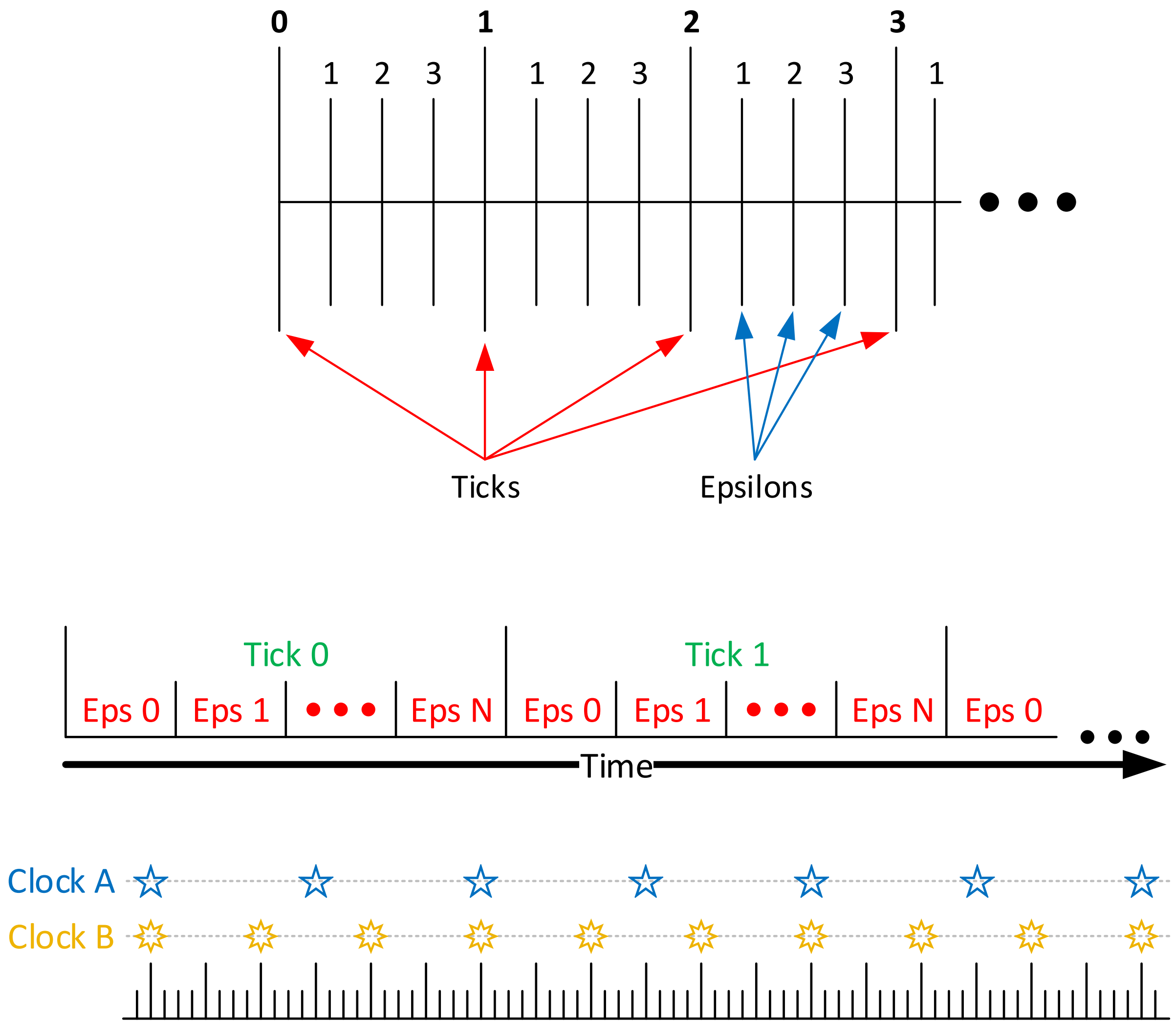
Hewlett Packard Enterprise

SuperSim Structure

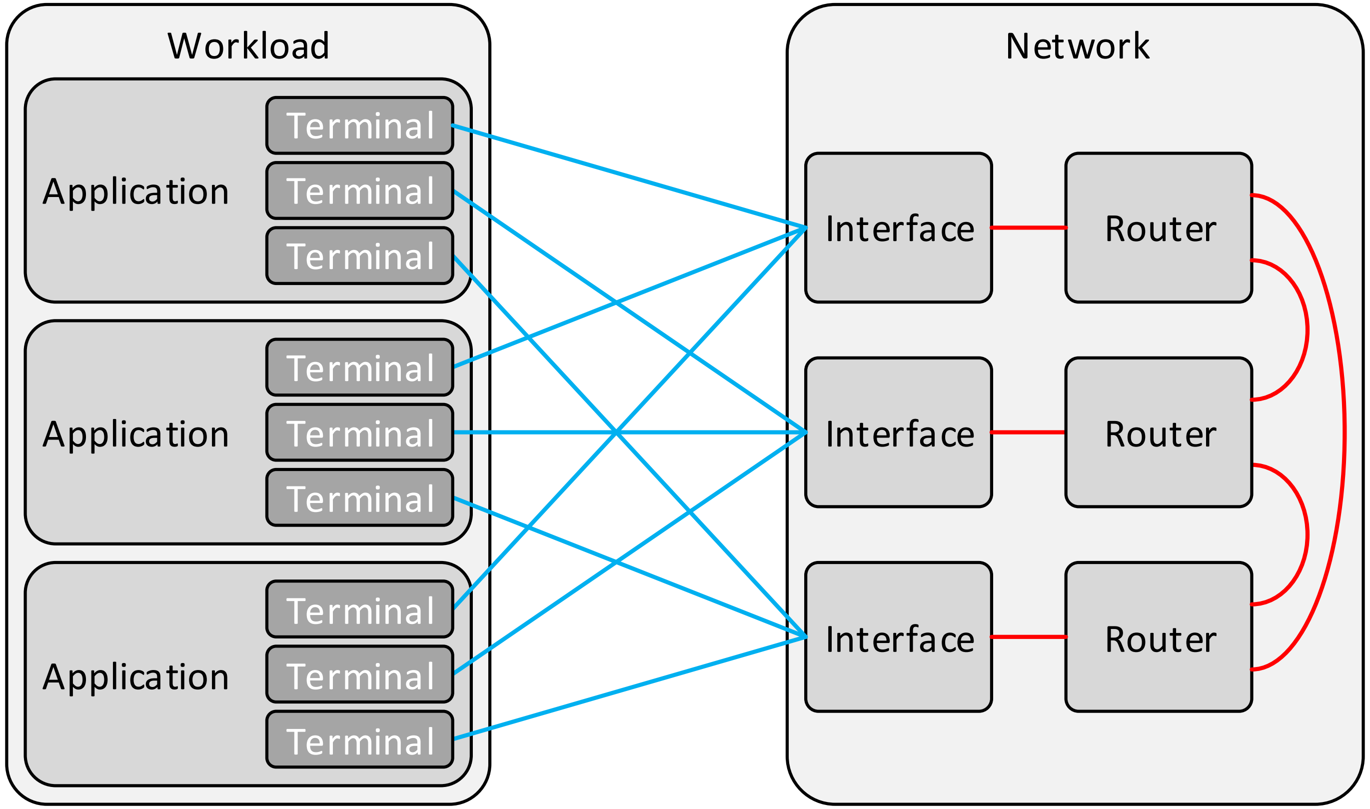
Hewlett Packard Enterprise

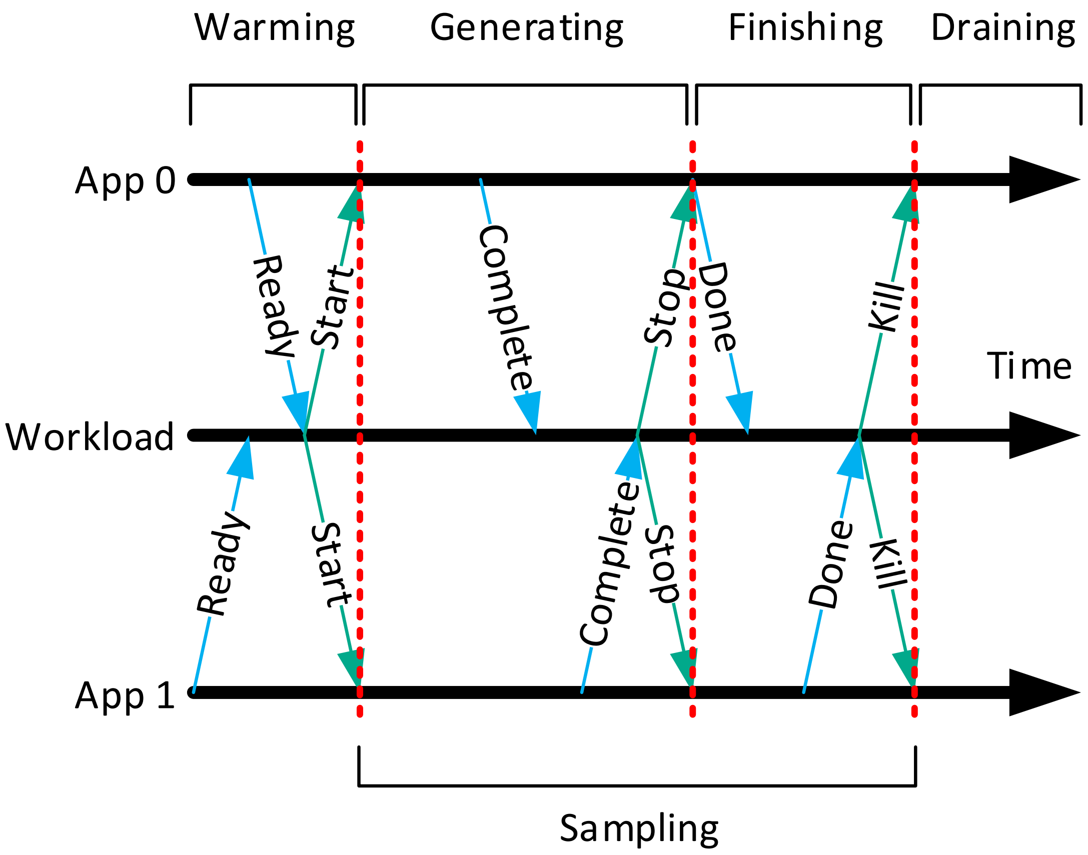
Simulator Core





Simulator Architecture



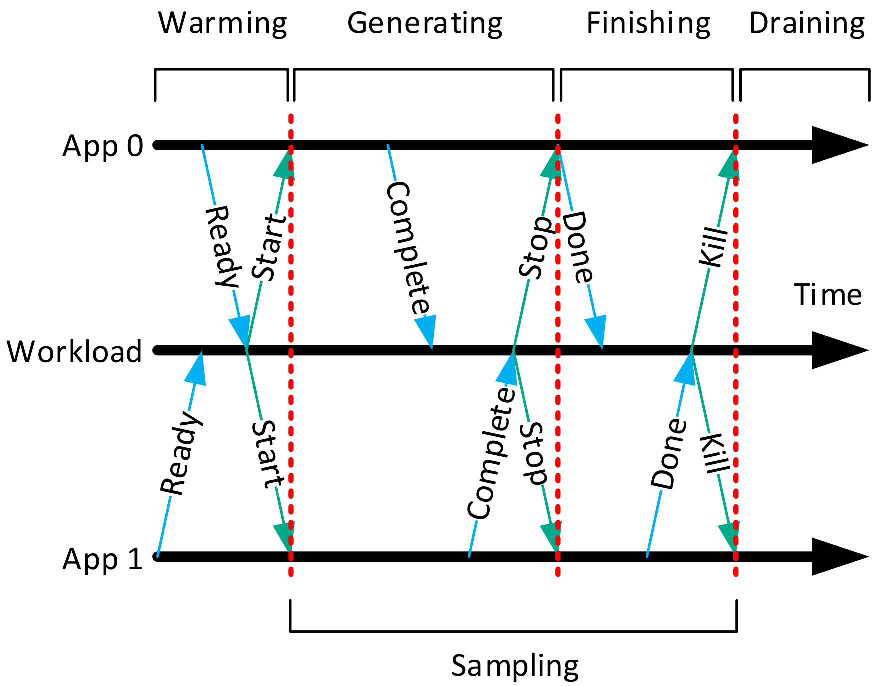


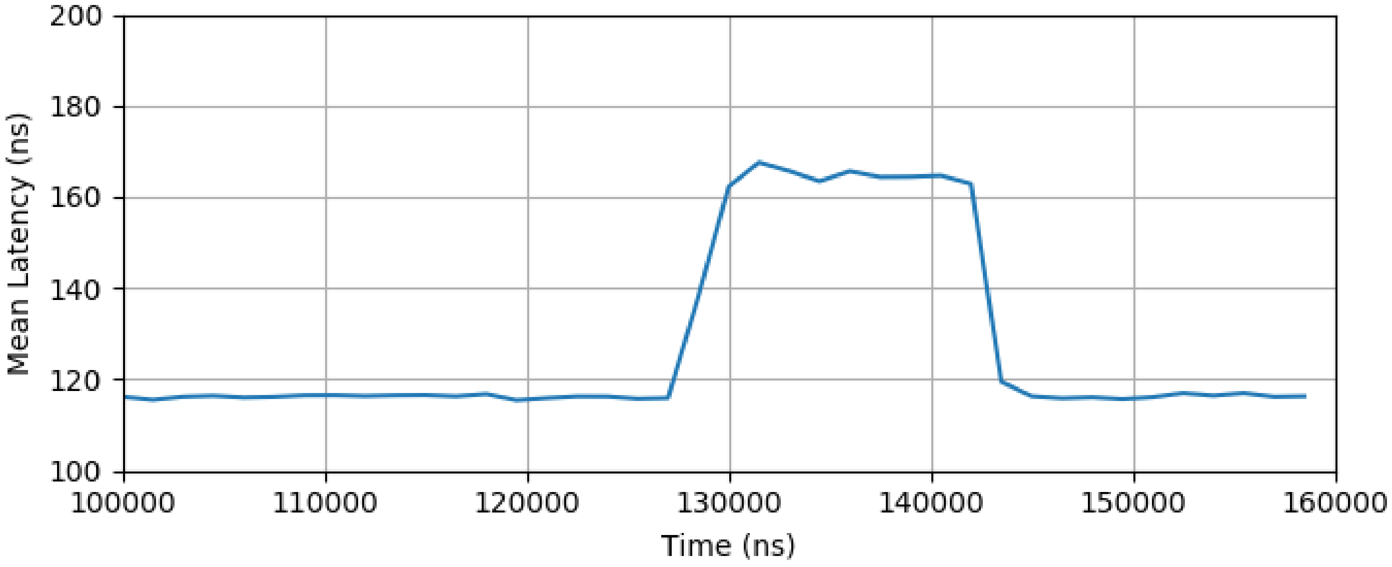
Multi-Application Workload Example

– Explore transient analysis of adaptive routing

– “Blast” application running steady state traffic

– “Pulse” application generates a temporary disturbance with a batch of traffic





Network Topologies (not trying to cover the whole space)

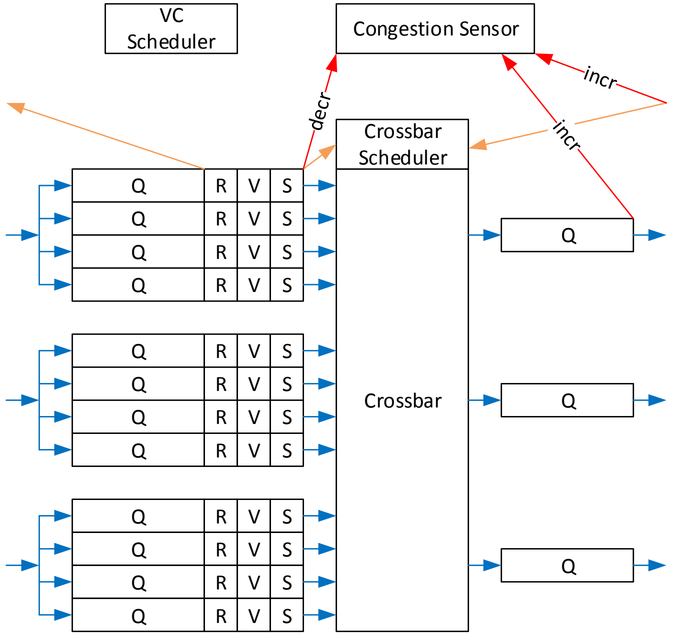
Real Topologies

- Torus
  - Oblivious routing
- Folded-Clos
  - Oblivious and adaptive routing
- HyperX
  - Can generate all HyperCubes and Flattened Butterflies
  - Oblivious and adaptive (to be released soon) routing
- Dragonfly
  - Oblivious and adaptive routing
- SlimFly (to be released soon)
  - Oblivious and adaptive routing

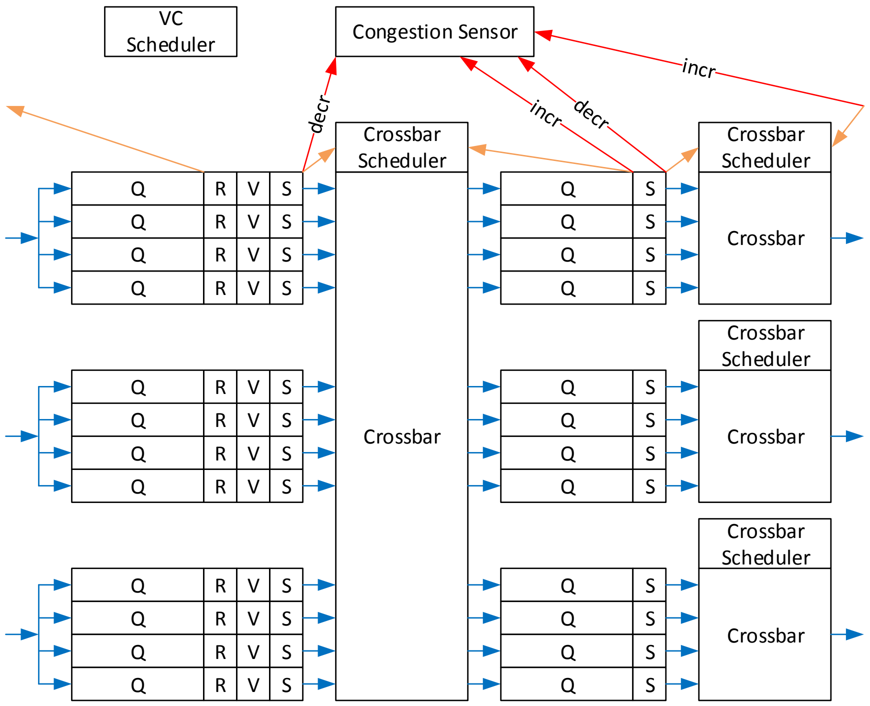
Testing Topologies

- Uno
  - A single router
- ParkingLot
  - A cascade of routers to stress bandwidth fairness

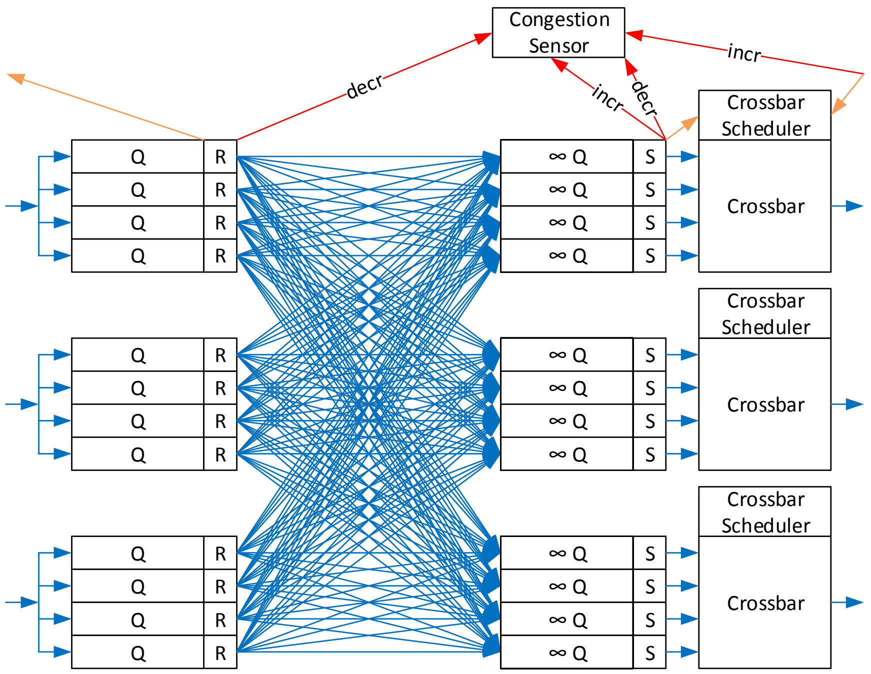
Router Architectures (definitely not covering the whole space)



Input Queued



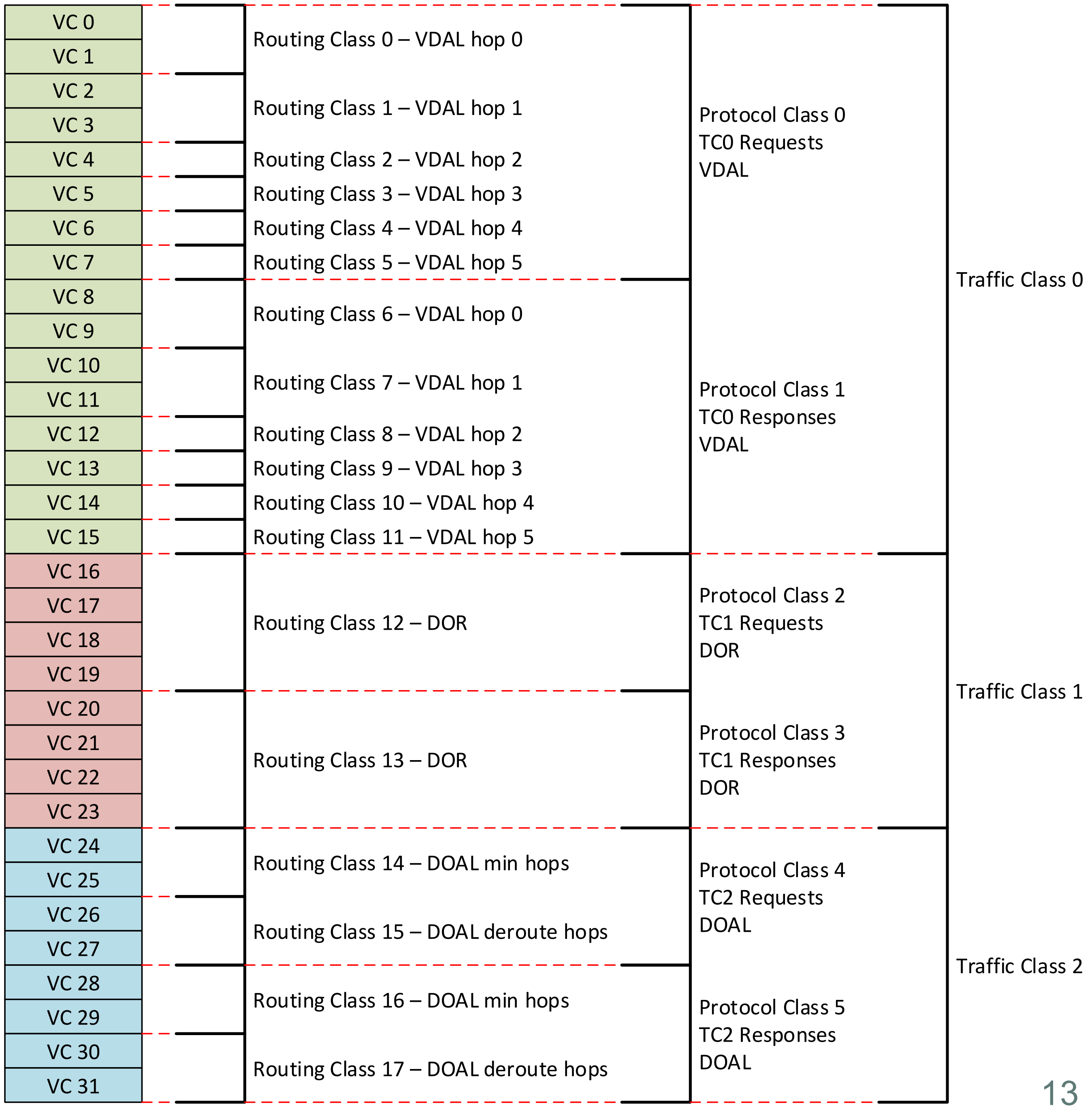
Input/Output Queued



Output Queued

VCs, RCs, PCs, and TCs

- Traffic Classes (TCs)
  - Protocol Classes (PCs)
    - Routing Classes (RCs)
      - Virtual Channels (VCs)



Hewlett Packard Enterprise

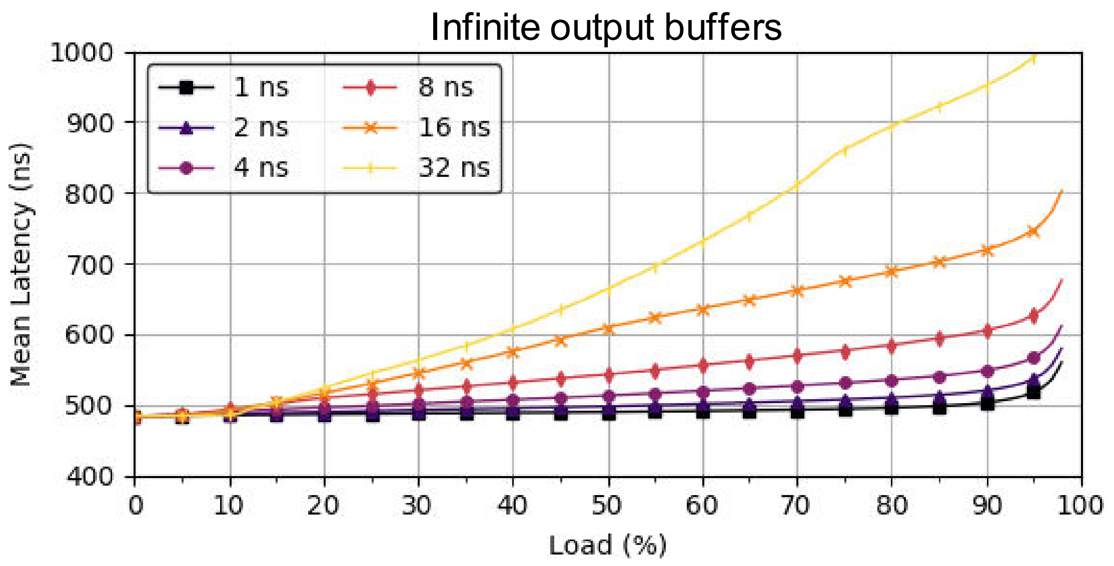
Simulation Experiments

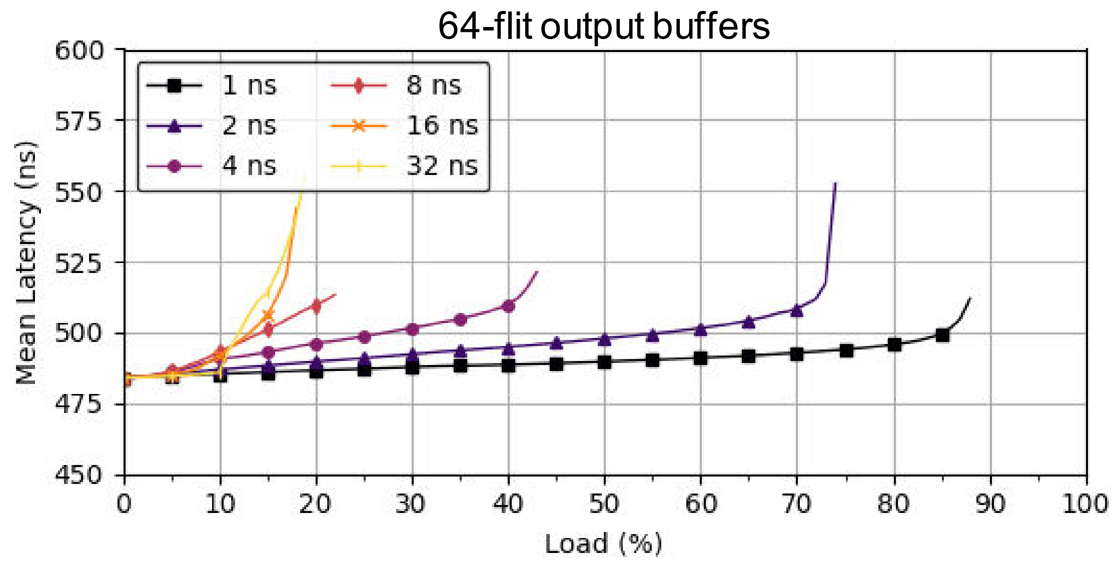
Latent Congestion Sensing

High-radix problem – many input ports bombard a seemly good output port

Congestion latency – the time it takes for the input ports to see the congestion changes on the output ports

Parameter	Value
Network topology	3-level folded-Clos, 4096 terminals
Network channel latency	50 ns (i.e., 10 meter cables)
Routing algorithm	adaptive uprouting
Router radix	32 ports
Router architecture	output-queued (OQ)
Frequency speedup	1x (i.e., none)
Number of VCs	1 VC
Input buffer size	150 flits
Output buffer size	infinite or 64 flits
Router core latency	50 ns queue-to-queue
Message size	1 flit
Traffic pattern	uniform random to root

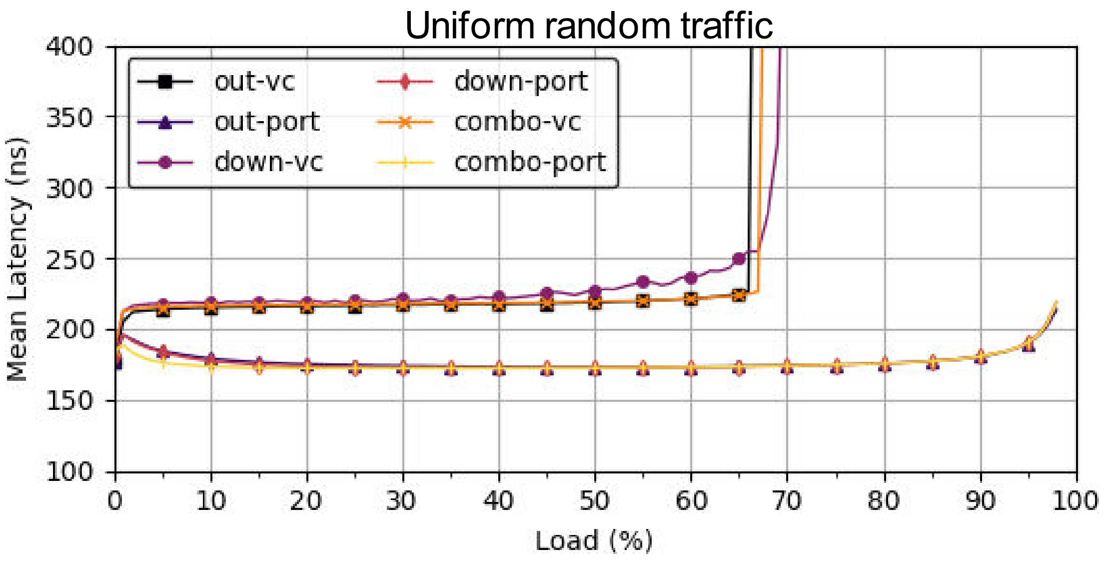


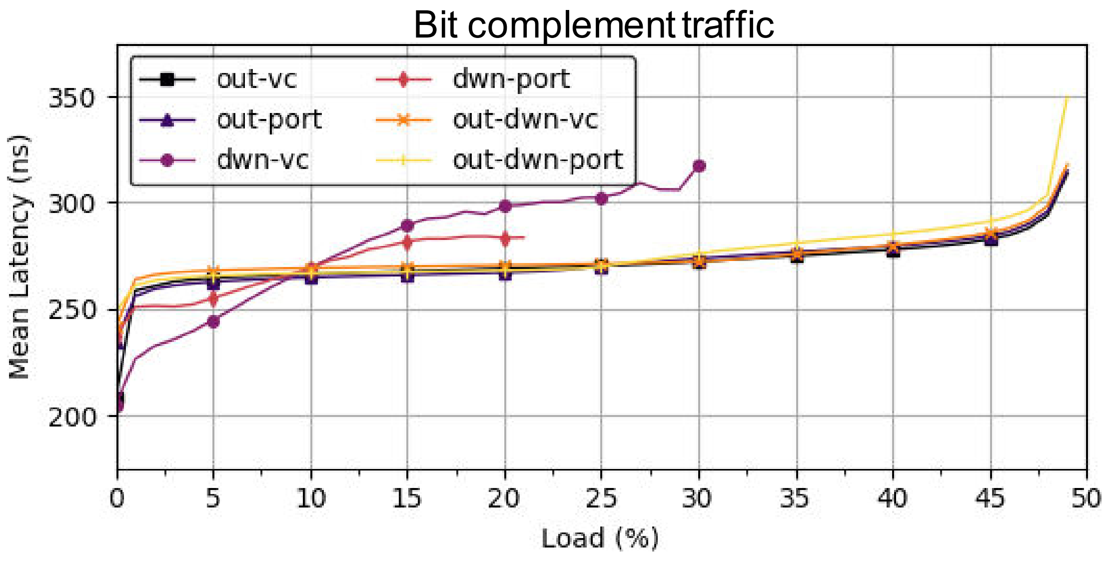


Congestion Credit Accounting

– Use UGAL routing, test different credit account mechanisms:
  – Output, downstream, output-and-downstream
  – VC, port

Parameter	Value
Network topology	1D flattened butterfly, 32 routers, 1024 terminals
Network channel latency	50 ns (i.e., 10 meter cables)
Routing algorithm	UGAL
Router radix	63 ports
Router architecture	input-output-queued (IOQ)
Frequency speedup	2x
Number of VCs	2 VCs
Input buffer size	128 flits
Output buffer size	256 flits
Router core latency	50 ns main crossbar latency
Message size	1 flit
Traffic pattern	uniform random, bit complement





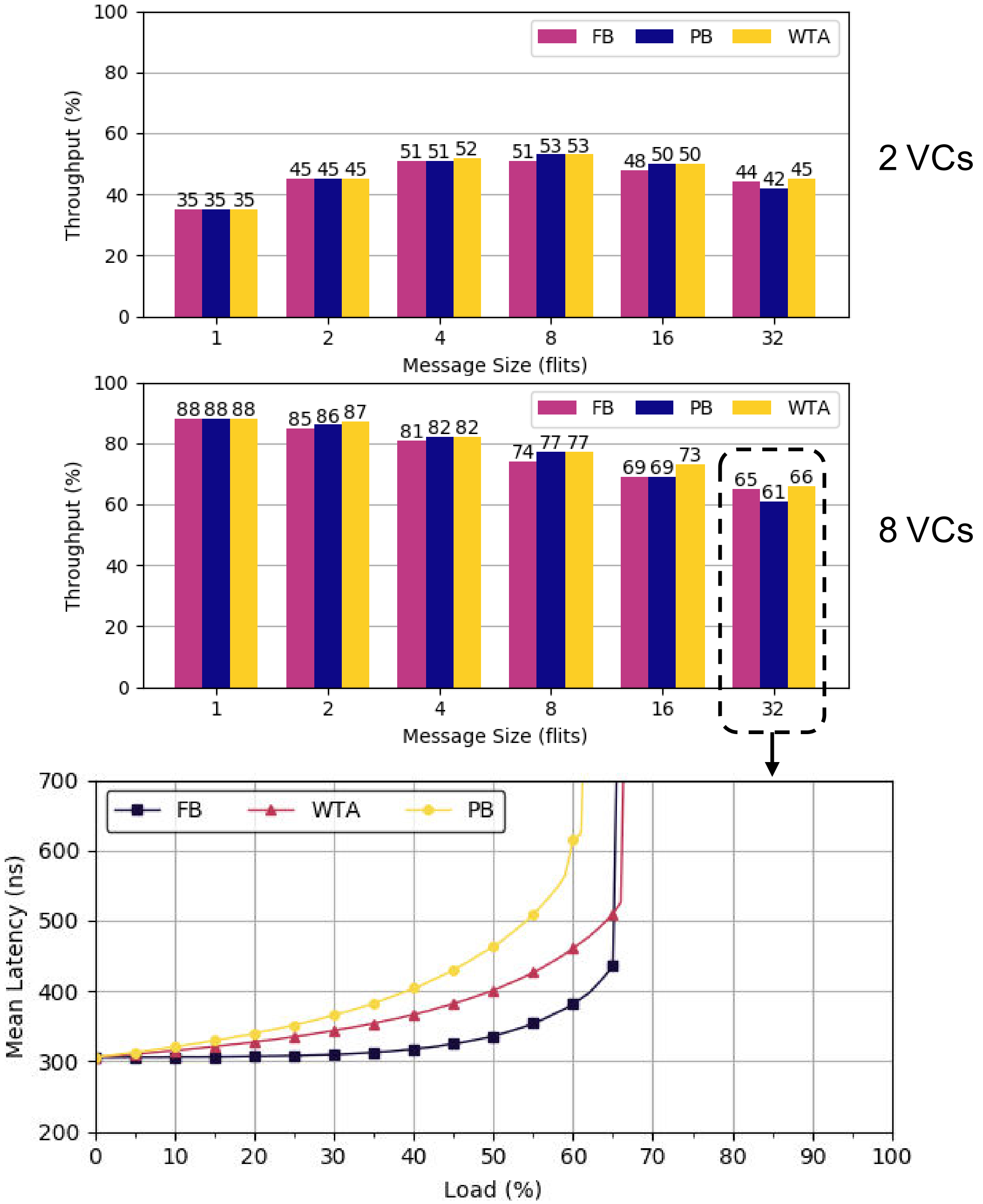
Flow Control Techniques

– Flit-buffer flow control (FB)

– Packet-buffer flow control (PB)

– Winner-take-all flow control (WTA)

Parameter	Value
Network topology	4D torus 8x8x8x8, 4096 terminals
Network channel latency	5 ns (i.e., 1 meter cables)
Routing algorithm	dimension order routing
Router radix	9 ports
Router architecture	input-queued (IQ)
Frequency speedup	1x (i.e., none)
Number of VCs	2,4,8 VCs
Input buffer size	128 flits
Output buffer size	n/a
Router core latency	25 ns main crossbar latency
Message size	1,2,4,8,16,32 flits
Traffic pattern	uniform random

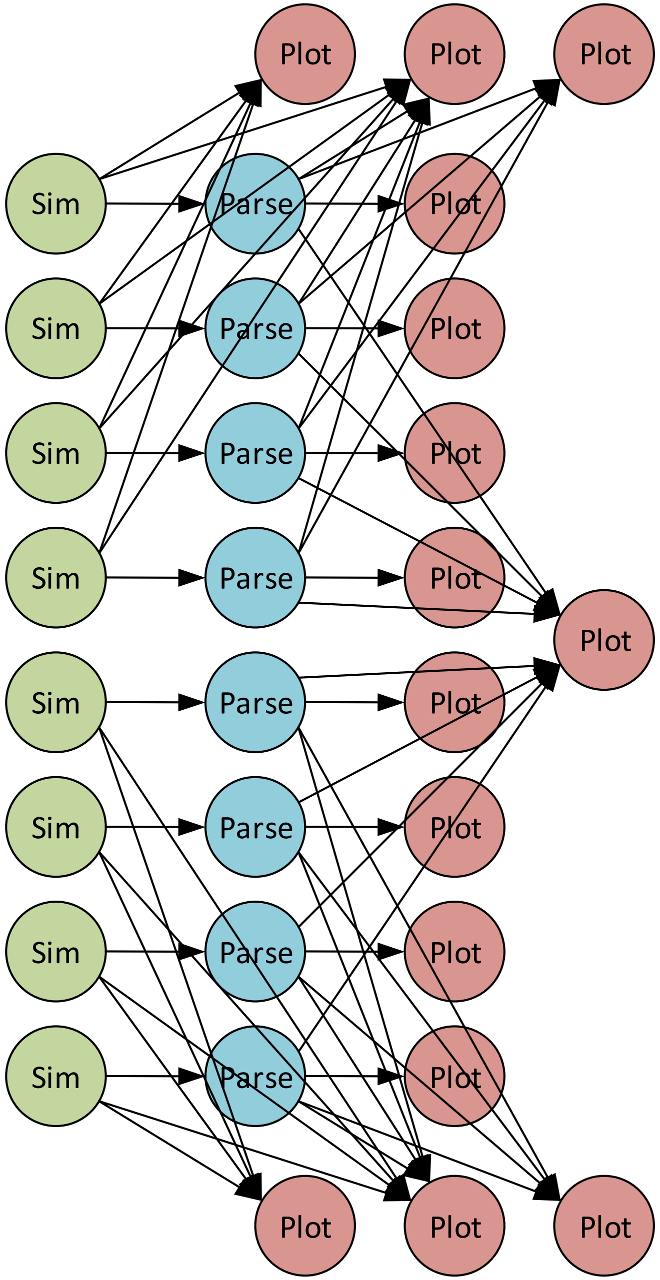


Hewlett Packard Enterprise

Accompanying Tools

Simulation Pipeline

1. Configure
   – Create the simulation configurations needed for the experiment
2. Simulate
   – Run the simulations using the configurations
3. Parse
   – Parse the results of the simulation outputs into the format needed in the remaining steps
4. Analyze
   – Analyze the parsed results from simulation to create desired statistics
5. Plot
   – Generate plots of analysis data
6. View
   – View the analyzed and plotted results



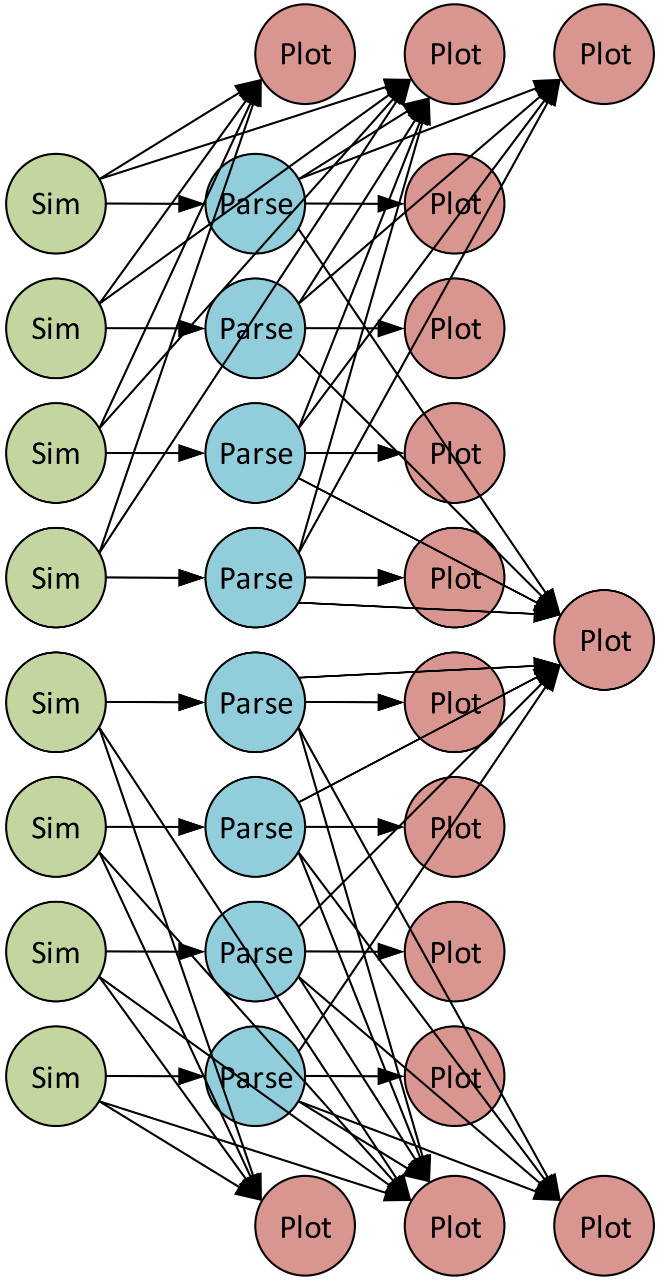
This can easily turn into 10s of 1000s of command line operations!!!

Hewlett Packard Enterprise

Taskrun

– A Python package for declaring tasks and automated execution
– Generic task API supports:
  – Function tasks – executed as a Python function callback
  – Process tasks – locally executed command
  – Cluster tasks – a remotely executed command via a cluster scheduler (e.g., PBS, LSF, Slurm, etc.)
  – Your next big idea…
– Resource management (e.g., memory, CPUs, etc.)
– Dependencies and conditional execution (i.e., like a Makefile)

* Not a tool specific to SuperSim



Hewlett Packard Enterprise

SSParse

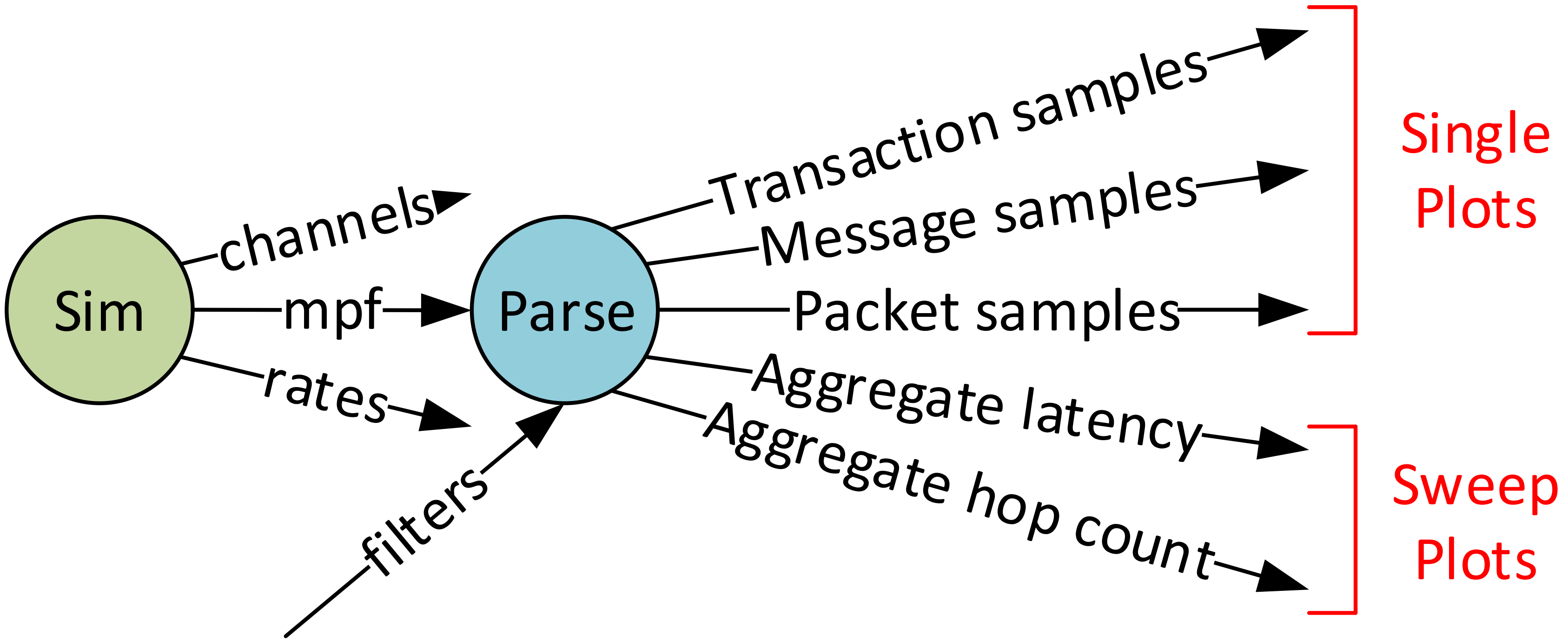
– SuperSim outputs a file containing information for all traffic from the "sampling" window (e.g., *.mpf).

– SSParse parses this file, run analyses, and prepares data sets for plotting

– SSParse exposes a filtering API to only view the information you care about

  – Ex: "+app=1" – only parses data from application 1

  – Ex: "-send=450-890" – parses data not sent between time 450 and 890



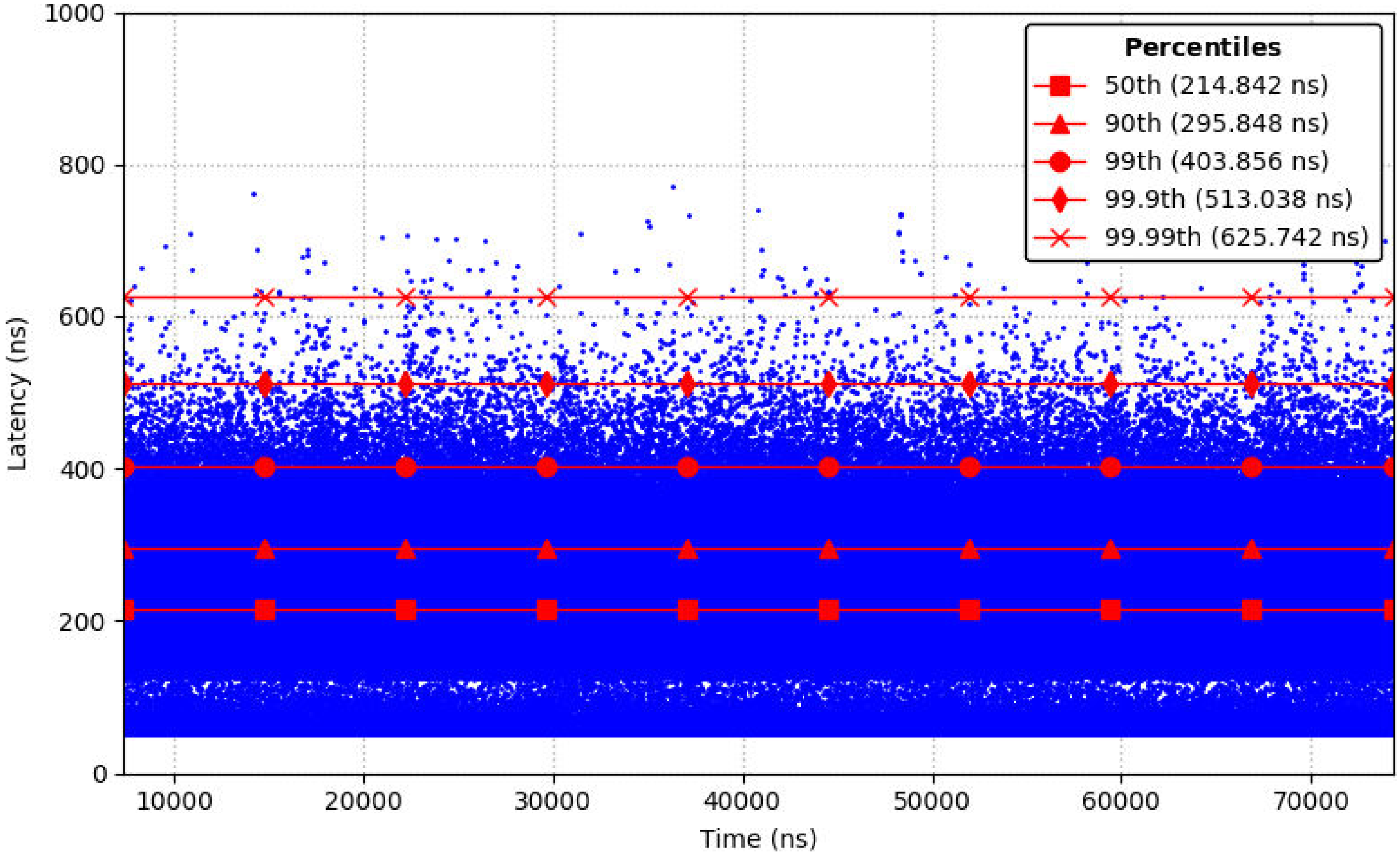
SSPlot

Name	Description	# of Sims
Time-Latency-Scatter	Load vs. latency scatter	1
Latency-PDF	Latency probability density function	1
Latency-CDF	Latency cumulative distribution function	1
Latency-Percentile	Latency percentiles (inverted logarithmic CDF)	1
Time-Latency	Time vs. latency at all distributions	1
Time-Average-Hops	Time vs. average hops	1
Time-Percent-Minimal	Time vs. minimal and non-minimal percentages	1
Load-Latency	Load vs. latency at all distributions	1 sweep
Load-Rate	Offered rate vs. delivered rate (min, mean, max)	1 sweep
Load-Rate-Percent	Offered rate vs. delivered rate (total, minimal, non-minimal)	1 sweep
Load-Average-Hops	Load vs. average hops	1 sweep
Load-Percent-Minimal	Load vs. minimal and non-minimal percentages	1 sweep
Load-Latency-Compare	Load vs. latency across multiple sweeps	N sweeps

SSPlot: Time-Latency-Scatter

Load vs. latency scatter

– This is the result of a single simulation

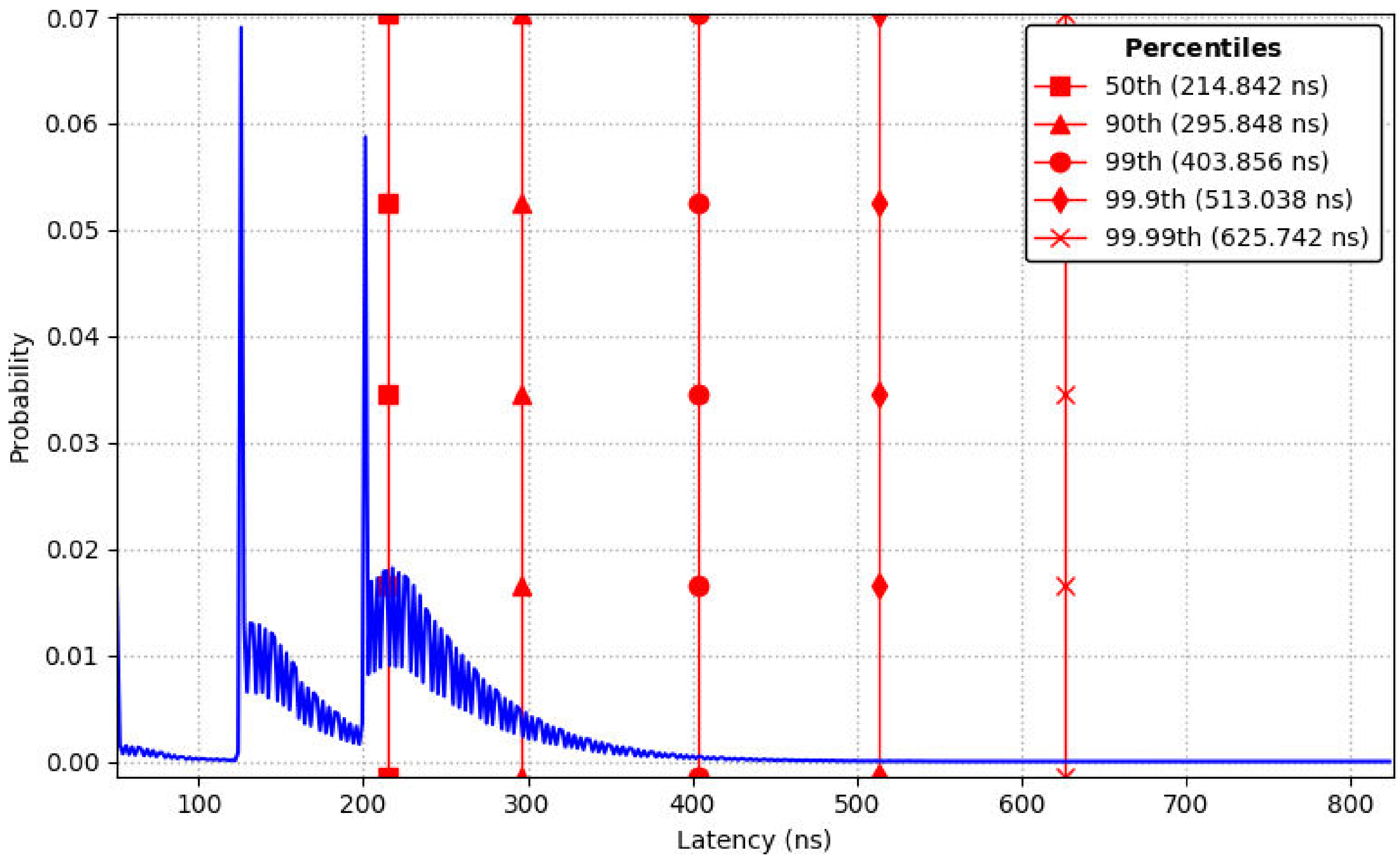


Hewlett Packard Enterprise

SSPlot: Latency-PDF

Latency probability density function

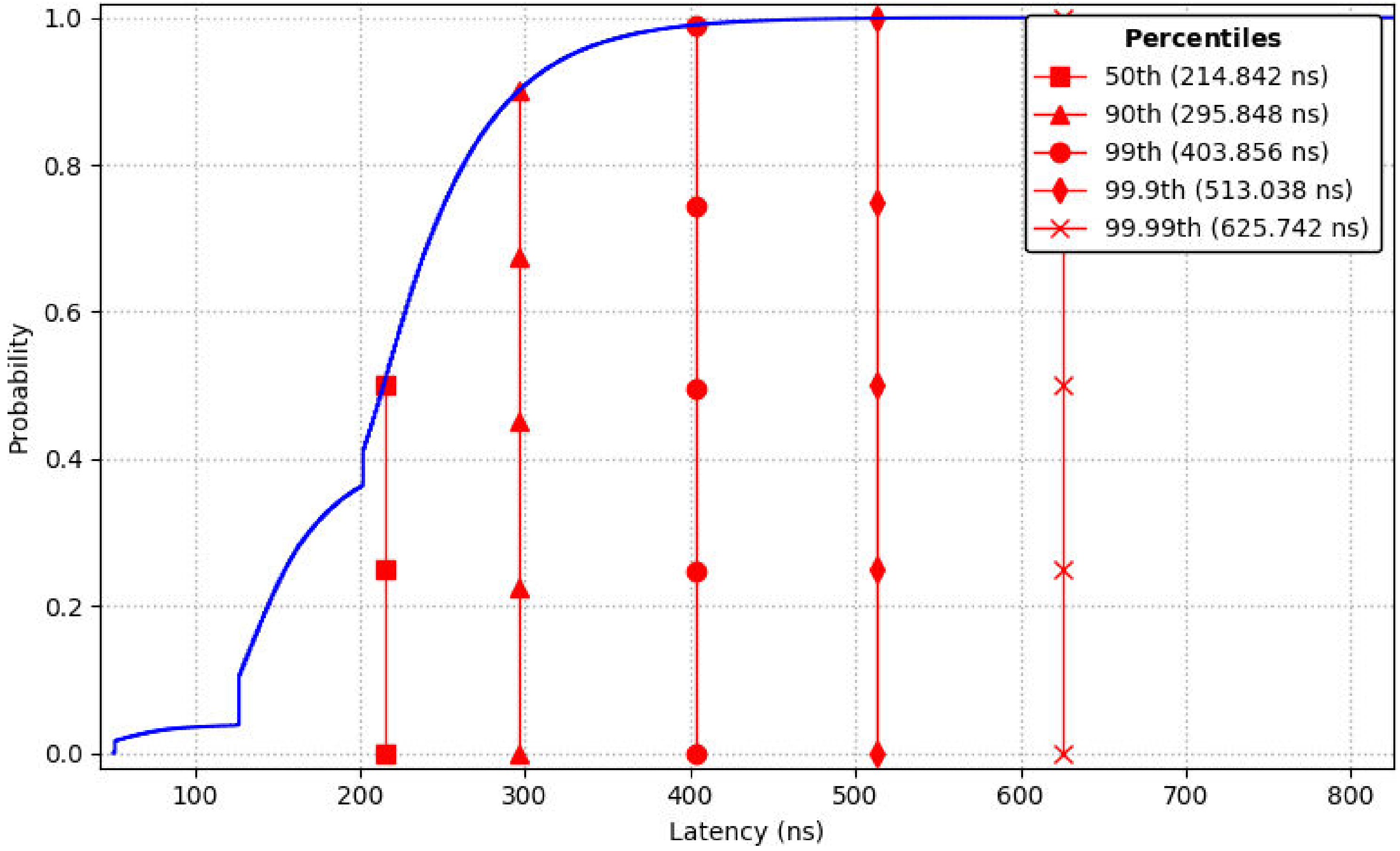
– This is the result of a single simulation



SSPlot: Latency-CDF

Latency cumulative distribution function

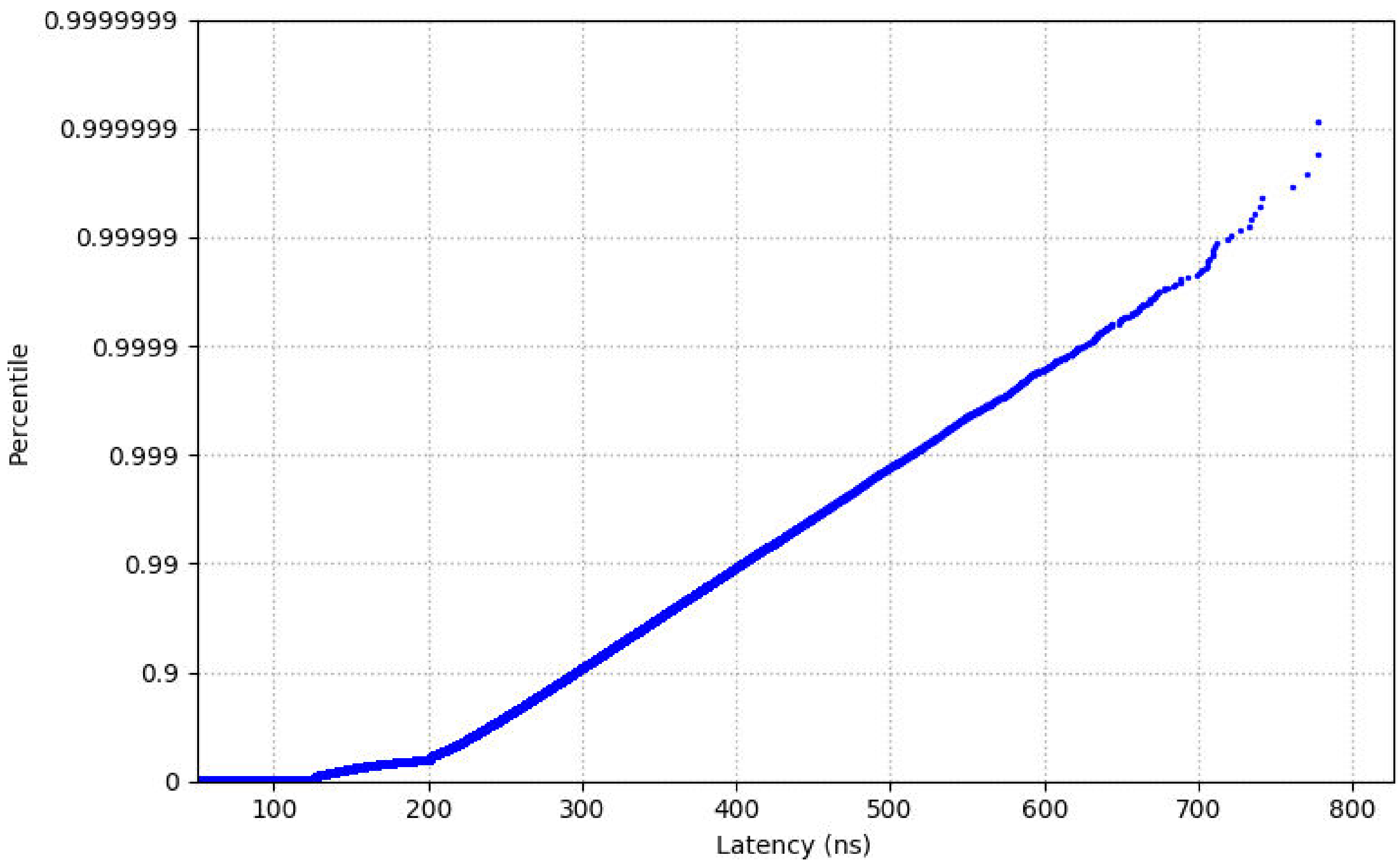
– This is the result of a single simulation



SSPlot: Latency-Percentile

Latency percentiles (inverted logarithmic CDF)

– This is the result of a single simulation



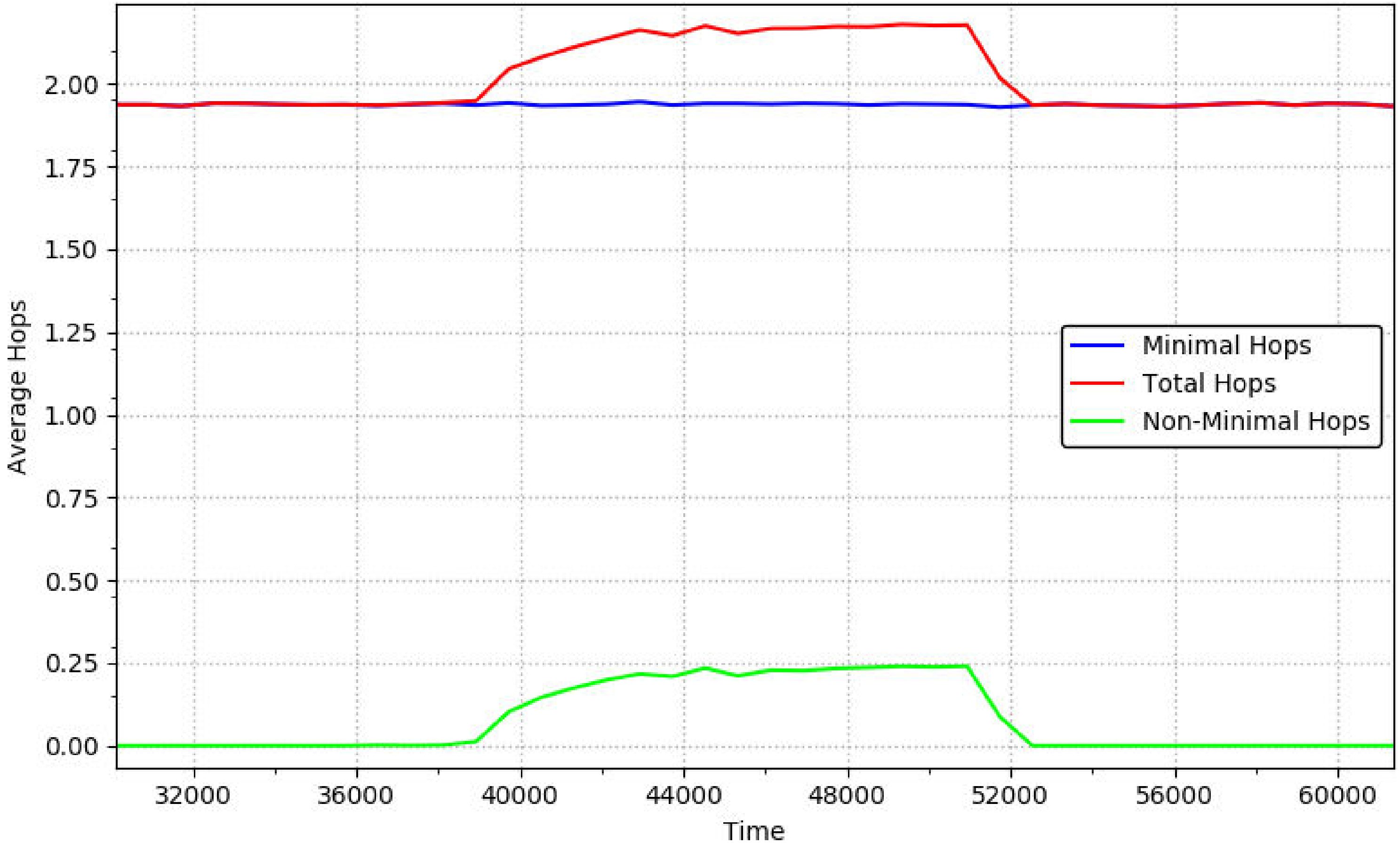
SSPlot: Time-Average-Hops

Time vs. average hops

– This is the result of a single simulation

– In this simulation there are two applications. Application 0 sends uniform random traffic at 10% the whole time. Application 1 sends bit complement traffic (adversarial) in a pulse starting at time ~35,000

– These results show the traffic only for Application 0



Hewlett Packard Enterprise

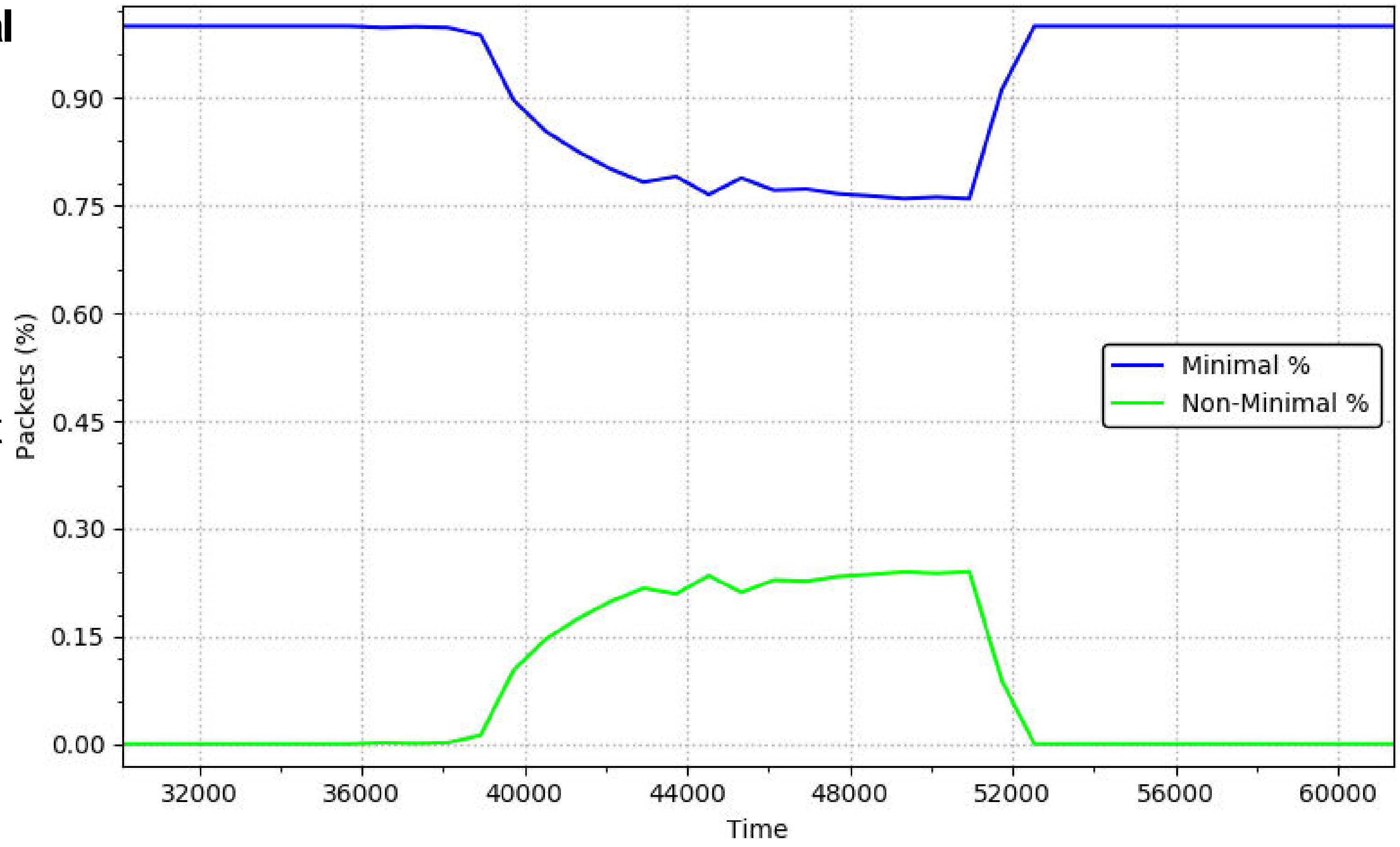
SSPlot: Time-Percent-Minimal

Time vs. minimal and non-minimal percentages

– This is the result of a single simulation

– In this simulation there are two applications. Application 0 sends uniform random traffic at 10% the whole time. Application 1 sends bit complement traffic (adversarial) in a pulse starting at time ~35,000

– These results show the traffic only for Application 0

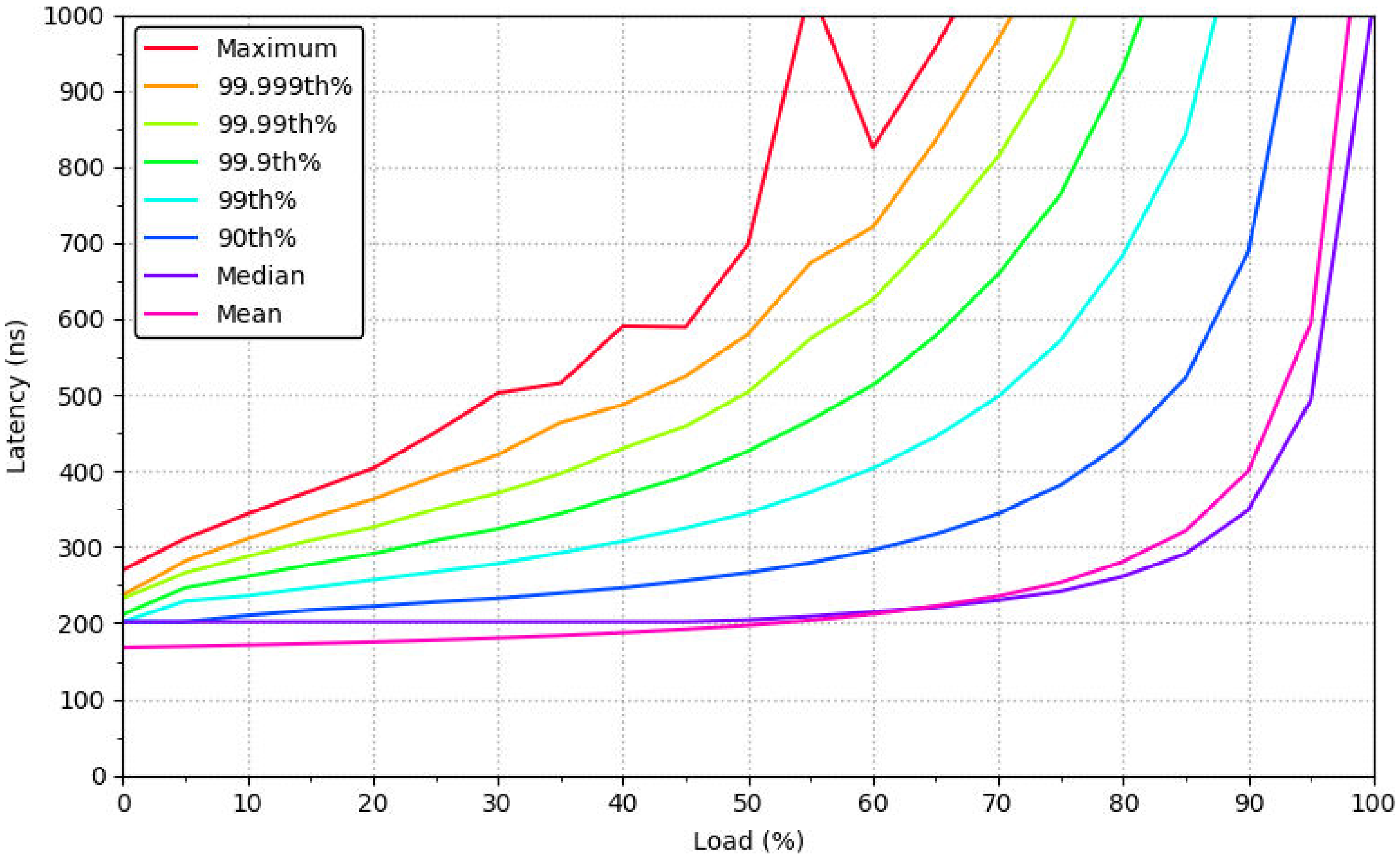


SSPlot: Load-Latency

Load vs. latency at all distributions

– This is the result of a sweep of simulations across injection rate

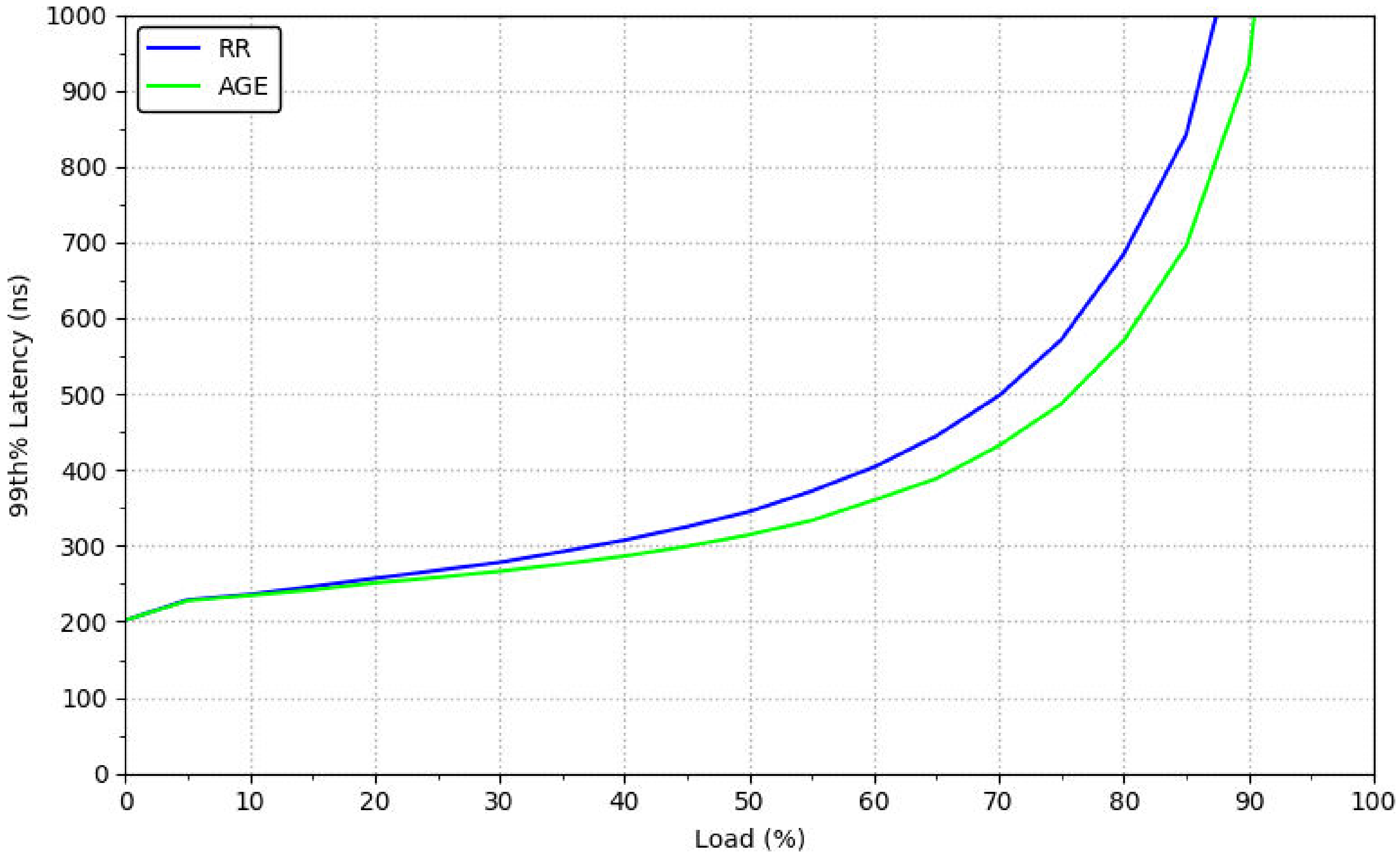
– This simulation is an application sending uniform random traffic and randomly sizes messages



SSPlot: Load-Latency-Compare

Load vs. latency across multiple sweeps

- This is the result of many sweeps of simulations across injection rate (one sweep for “RR” and one for “AGE”)
- This plot is like the Load-Latency plot but compares across multiple sweeps
- This particular setup shows median latency (any latency distribution can be chosen)



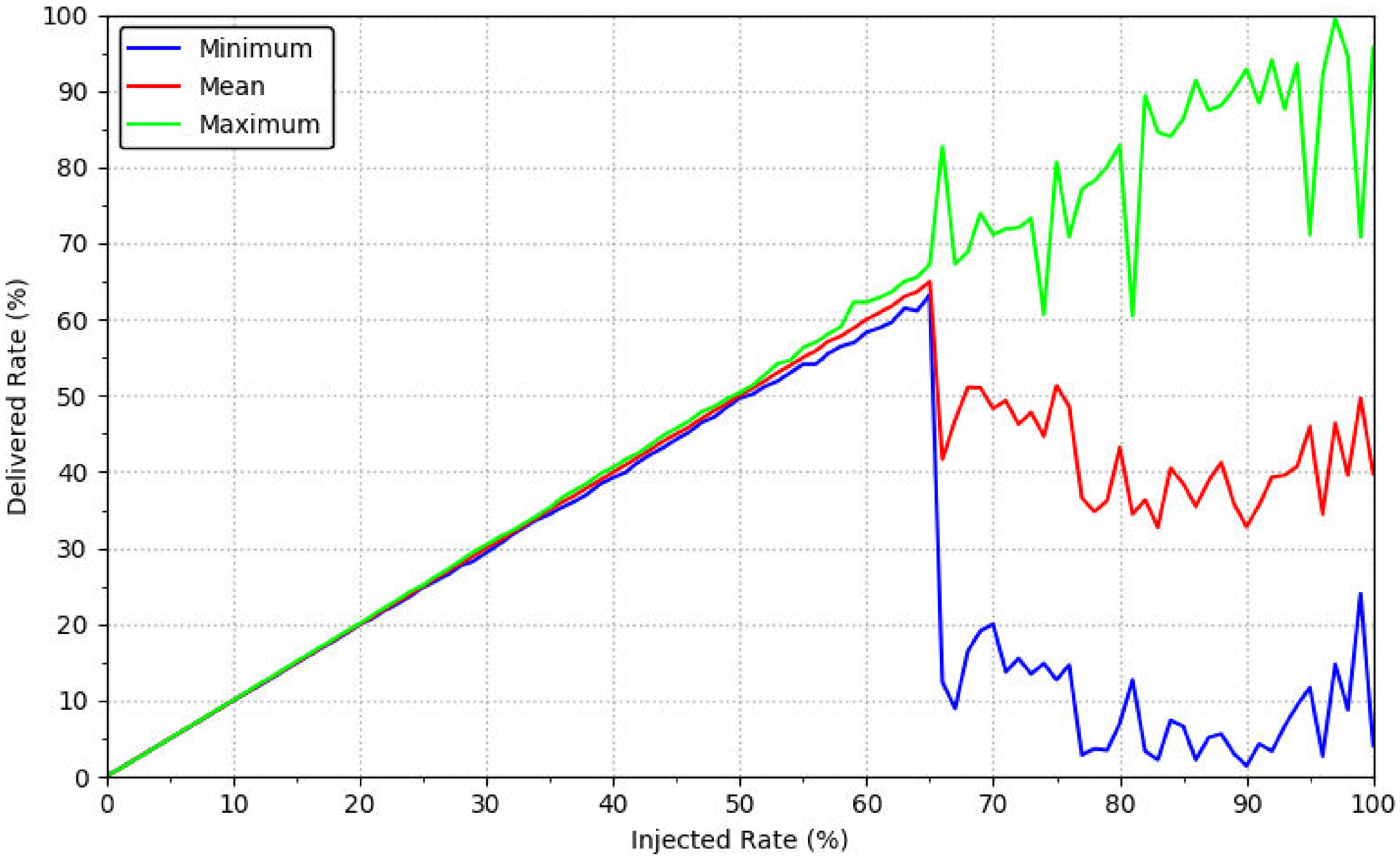
SSPlot: Load-Rate

Offered rate vs. delivered rate (min, mean, max)

– This is the result of a sweep of simulations across injection rate

– This simulation is an application sending traffic over a torus network that stresses the bisection

– At 65% injection rate, the network becomes saturated. Severe bandwidth unfairness occurs due to round-robin arbitration



Hewlett Packard Enterprise

SSSweep

– SSSweep automates the entire simulation pipeline process

– Users define independent simulation variables and corresponding functions to apply the variable

```
algs = ['oblivious', 'adaptive']

def set_alg(alg, config):
        return ('network.protocol_classes[0].routing.adaptive=bool={}'
                .format('true' if alg == 'adaptive' else 'false'))

sweeper.add_variable('Routing Algorithm', 'RA', algs, set_alg)
```

– Users define the type of plots they'd like

– SSSweep creates all configurations and uses Taskrun to run all tasks

– SSSweep generates a static HTML/CSS/Javscript web site for plot viewing

SuperSim: Extensible Flit-Level Simulation of Large-Scale Interconnection Networks

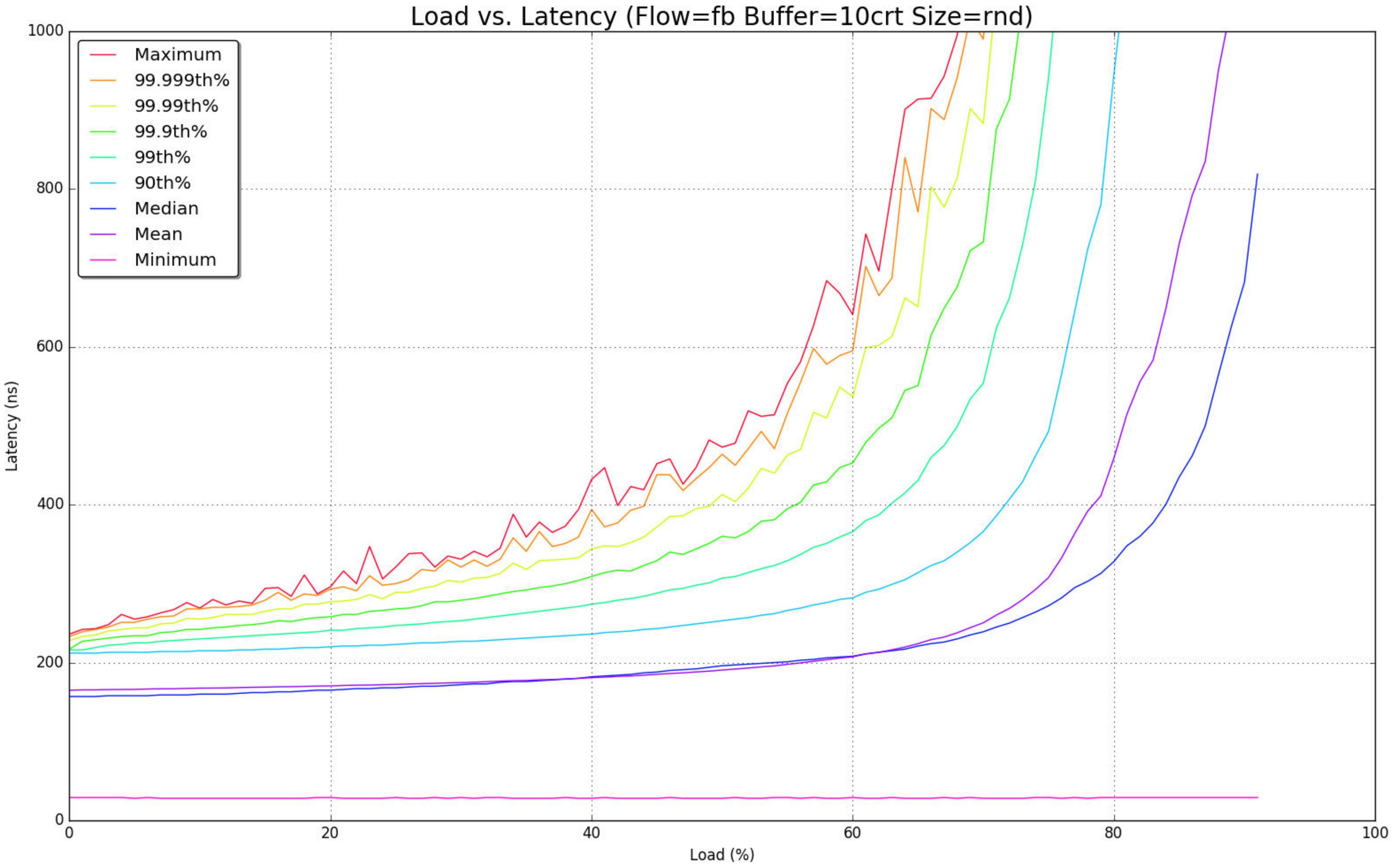
www.github.com/hewlettpackard/supersim

Hewlett Packard Enterprise

Backup Slides

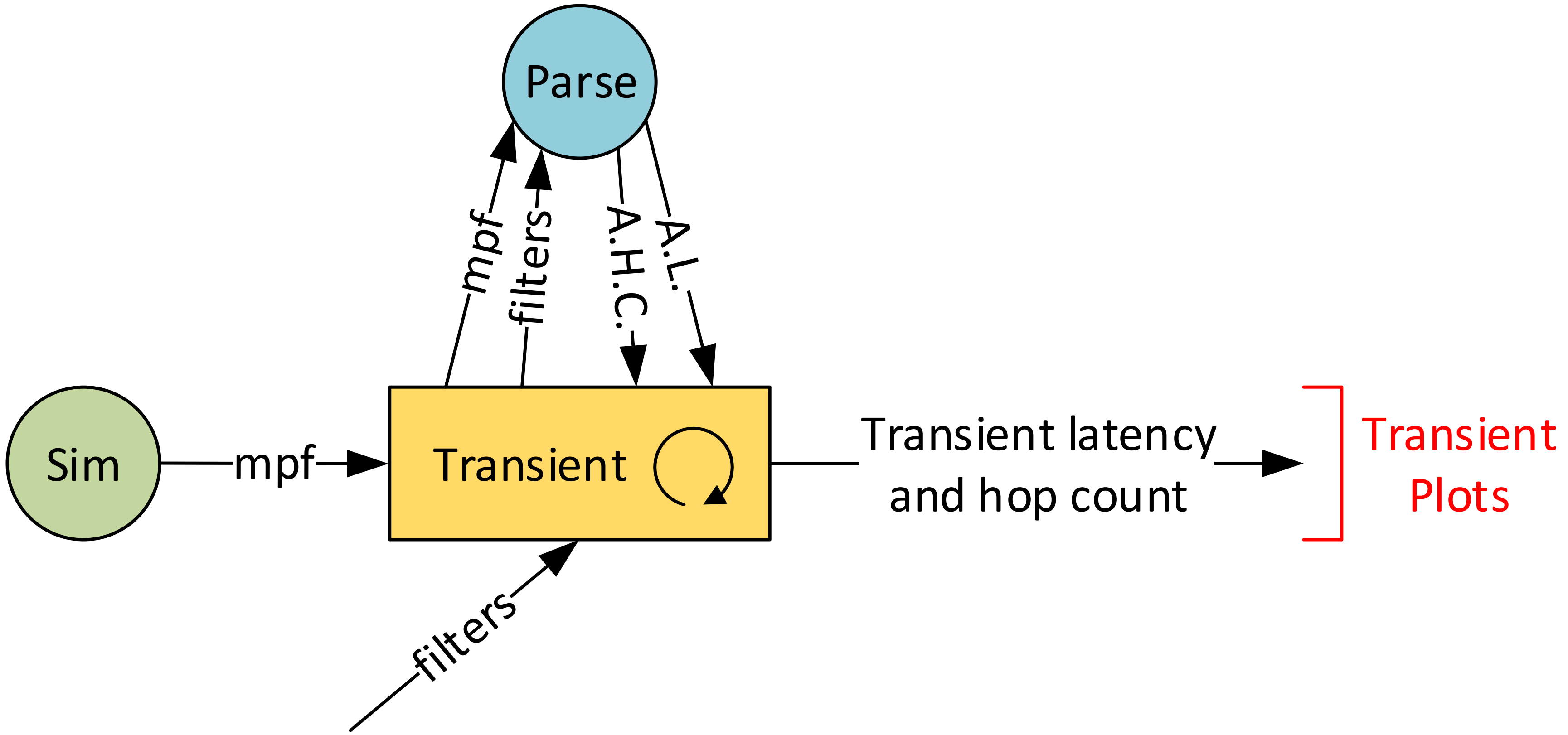
Focus on Real Issues of Large-Scale Networks

Analyze latency distributions rather than just average latency



SSParse: Transient Tool

– SSParse includes a wrapper tool that uses the main SSParse executable to generate a transient analysis



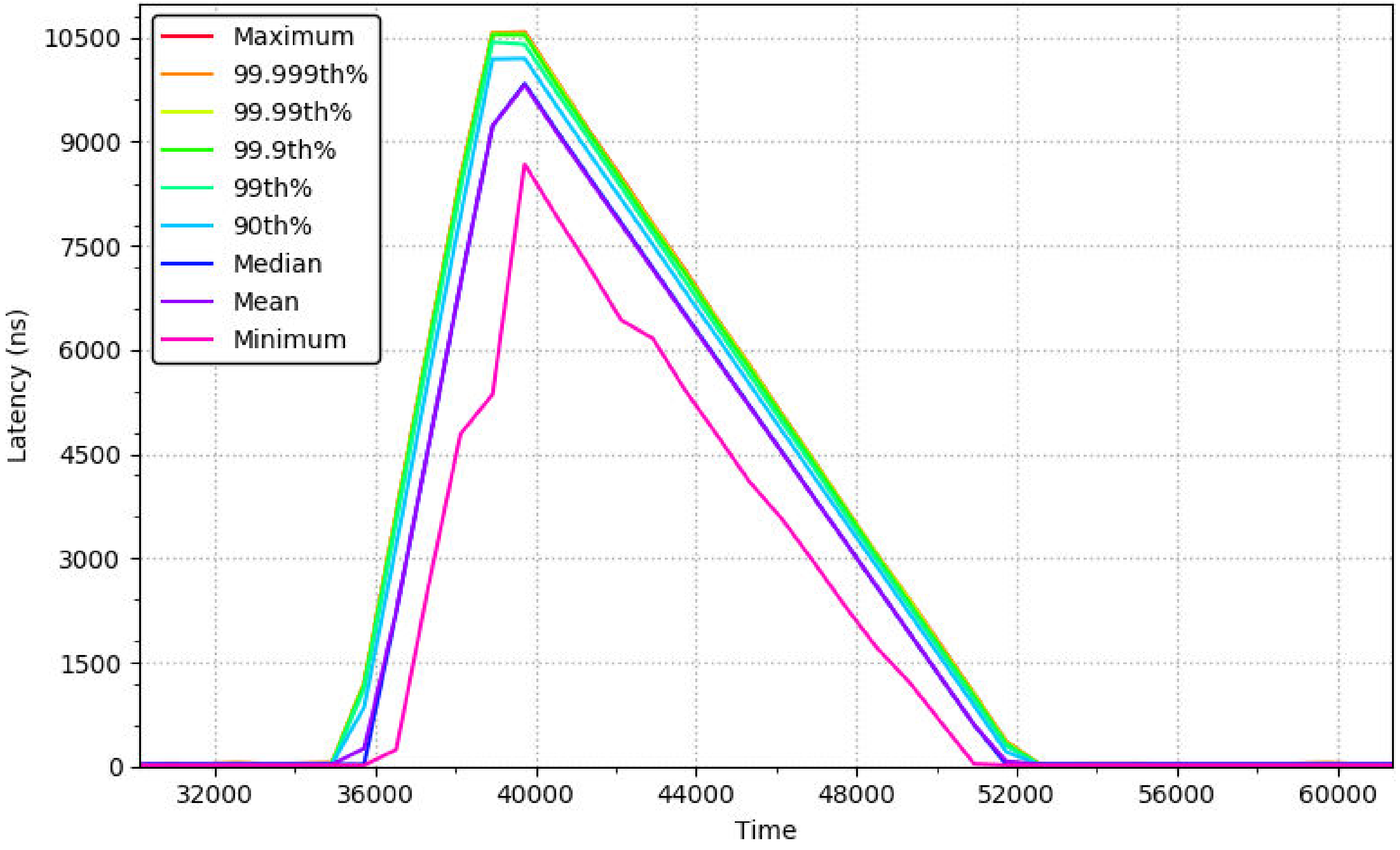
SSPlot: Time-Latency

Time vs. latency at all distributions

– This is the result of a single simulation

– In this simulation there are two applications. Application 0 sends uniform random traffic at 10% the whole time. Application 1 sends bit complement traffic (adversarial) in a pulse starting at time ~35,000

– These results show the traffic only for Application 0



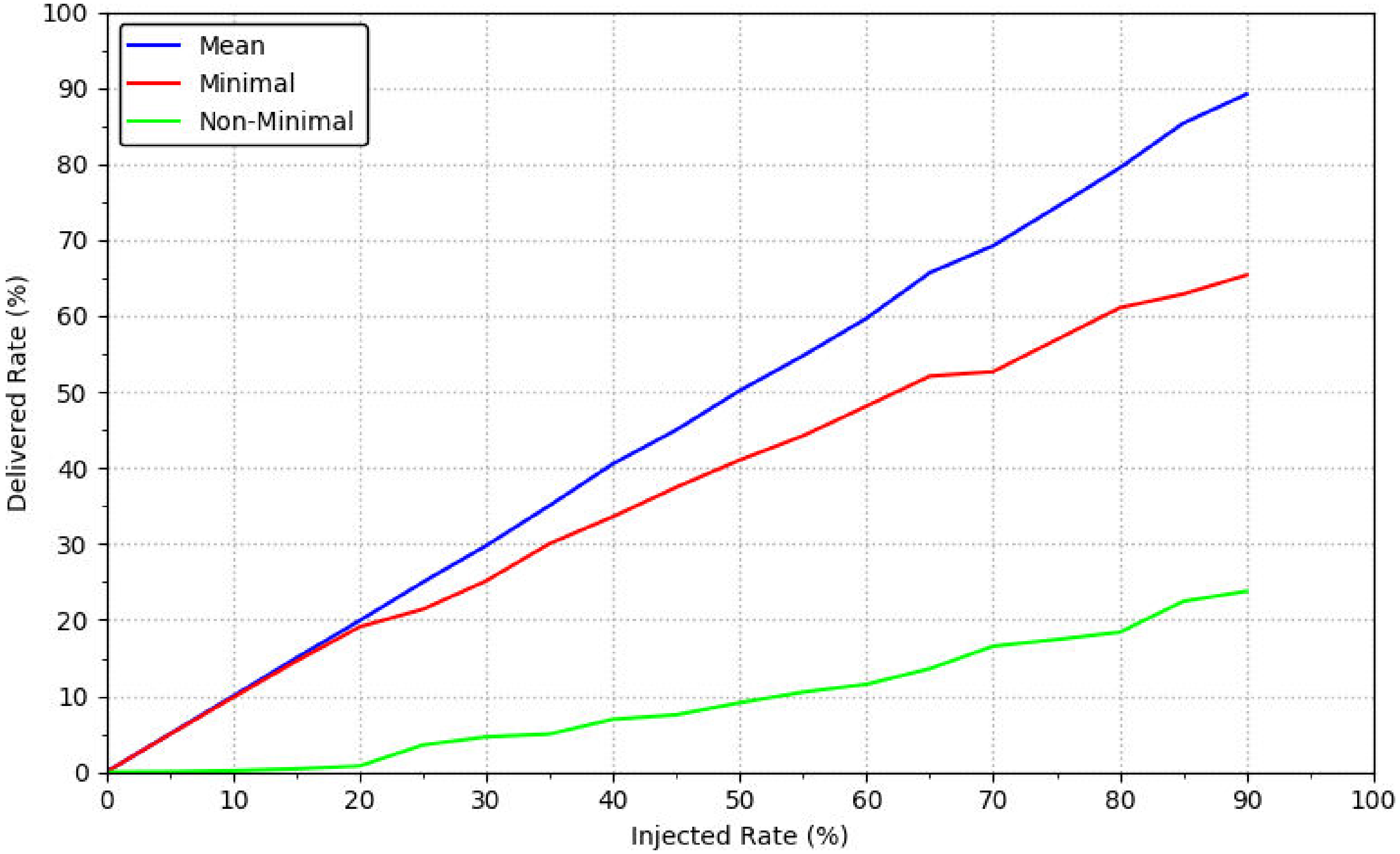
Hewlett Packard Enterprise

SSPlot: Load-Rate-Percent

Offered rate vs. delivered rate (total, minimal, non-minimal)

– This is the result of a sweep of simulations across injection rate

– This simulation is an application sending uniform random traffic and randomly sizes messages

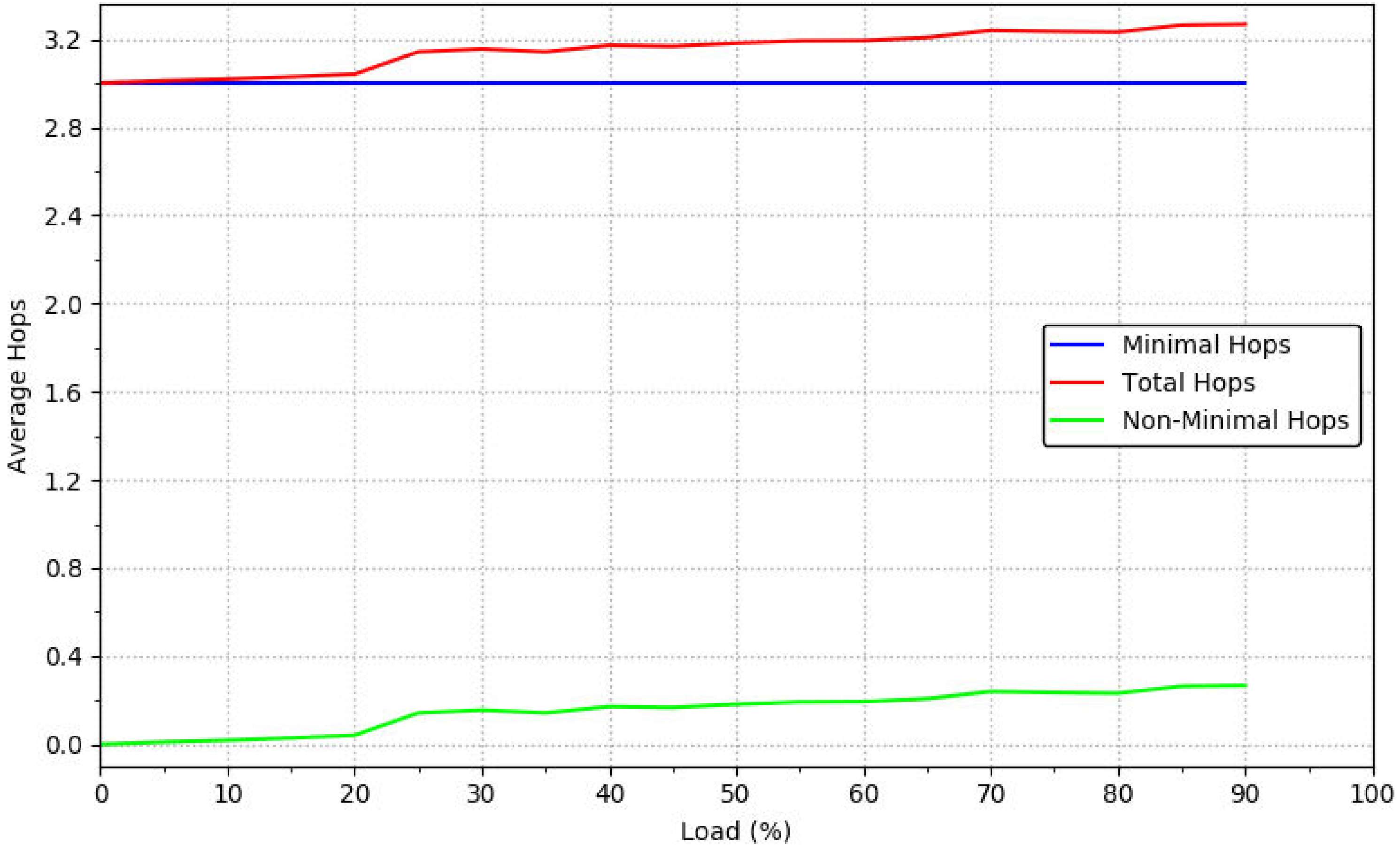


SSPlot: Load-Average-Hops

Load vs. average hops

– This is the result of a sweep of simulations across injection rate

– This simulation is an application sending uniform random traffic and randomly sizes messages

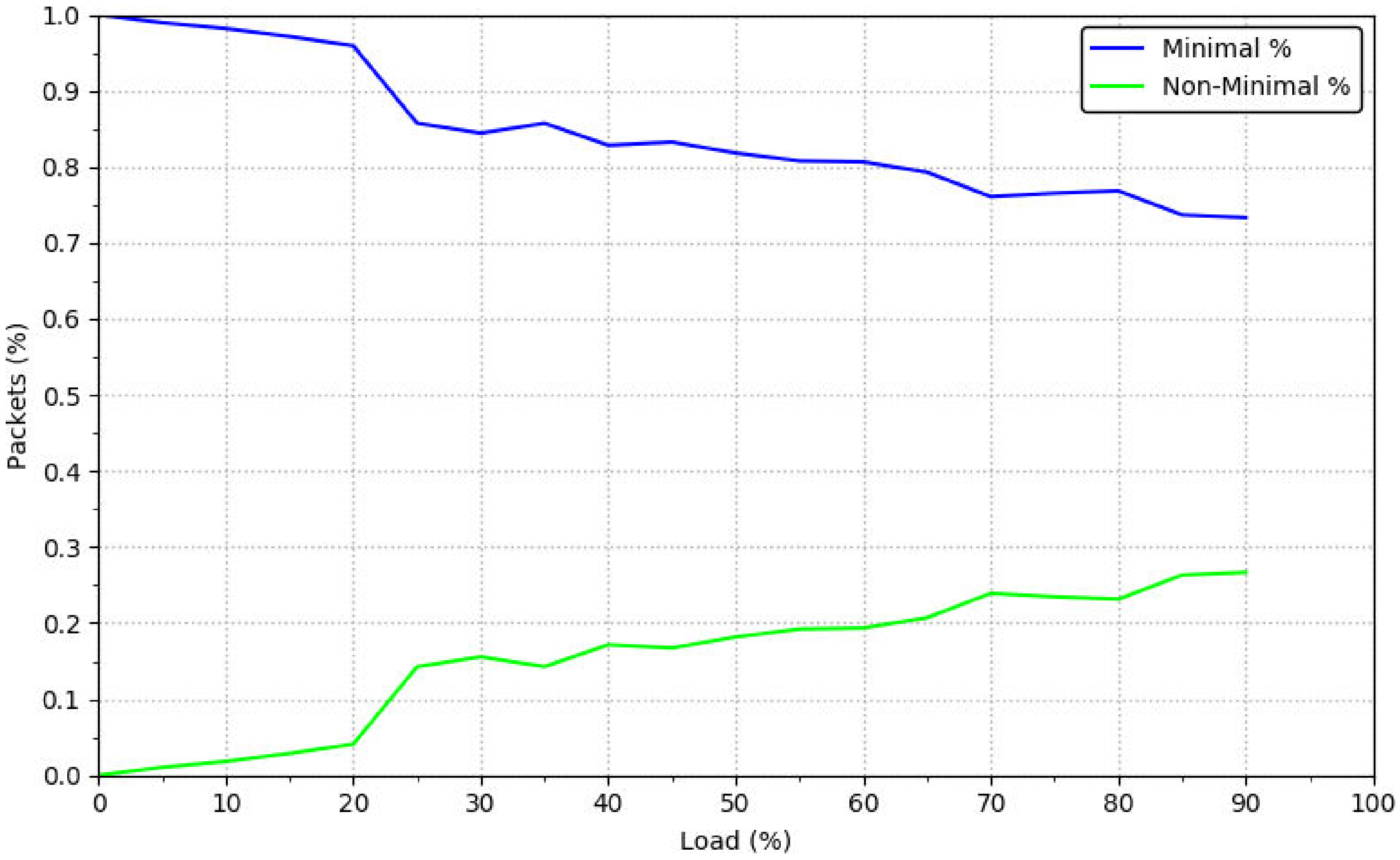


SSPlot: Load-Percent-Minimal

Load vs. minimal and non-minimal percentages

– This is the result of a sweep of simulations across injection rate

– This simulation is an application sending uniform random traffic and randomly sizes messages



Hewlett Packard Enterprise