Scalable Emulation of SDN Applications with Simulation Symbiosis

Jason Liu, Florida International University - USA
Cesar Marcondes, Federal University of São Carlos - BRAZIL

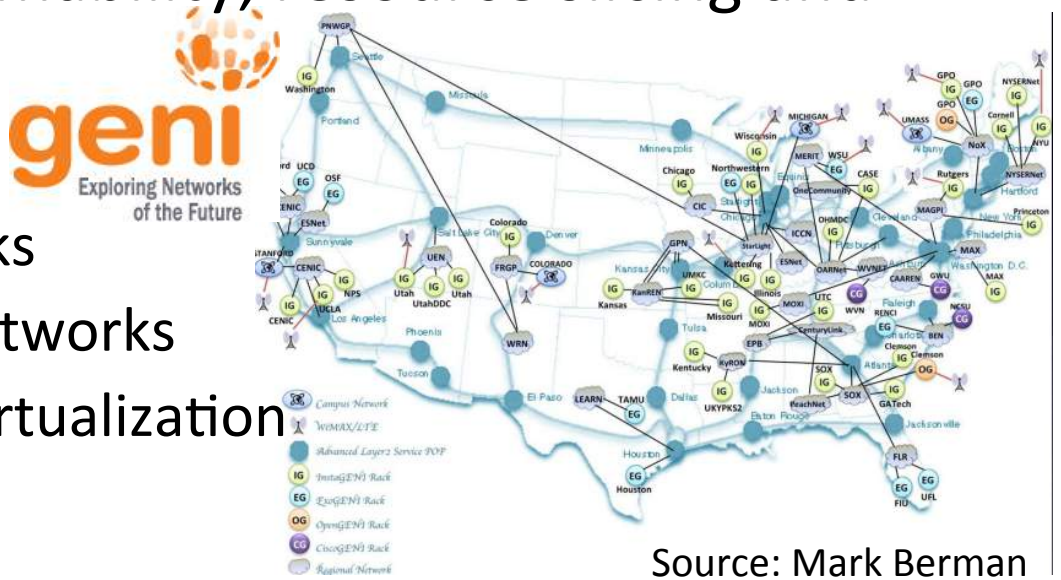
US NSF grant CNS-1346688
Brazil FAPESP Scholarship 2014/00708-6

Collaborations with Cesar's Group Related to Future Internet

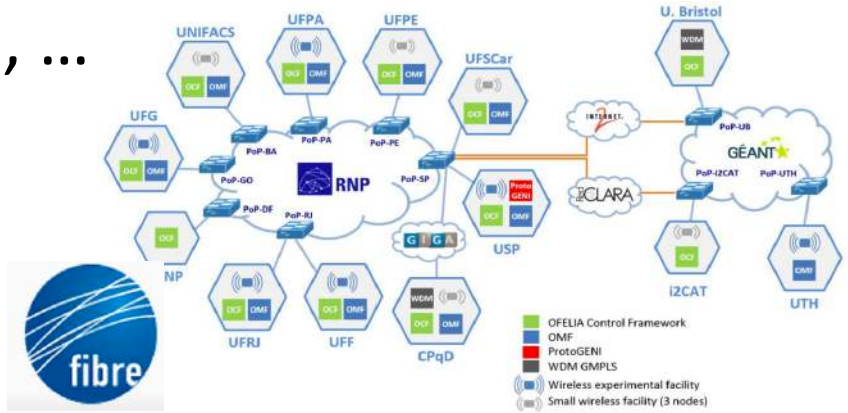
- SDN Conflux (this talk is one result)
- Superconductivity of virtual systems
- SDN TCP
- Traffic engineering and management

Motivation

- **Future Internet** (FI) testbeds
  - Allow deep programmability, resource slicing and federation
  - Applied research in:
    - Data Center Networks
    - Software Defined Networks
    - Network Function Virtualization
- Examples:
  - GENI, FIBRE, …
  - CCNIE
  - NSFClouds



Source: Mark Berman



Limitations

- For example, MapReduce optimization
  - There are >220 parameters to configure before running a job
  - Difficult to reconfigure (domain skills)
  - Not scalable

Initial setup	lower bound	upper bound	Parameter name
200	100	1,600	mapred.child.java.opts
100	80	1,120	io.sort.mb
10	8	200	io.sort.factor
0.05	0.03	0.15	io.sort.record.percent
4,096	2,048	32,768	io.file.buffer.size
0.7	0.5	0.8	mapred.job.shuffle.input.buffer.percent
0.66	0.4	0.8	mapred.job.shuffle.merge.percent
1,000	0	1,200	mapred.inmem.merge.threshold
0	0	0.8	mapred.job.reduce.input.buffer.percent

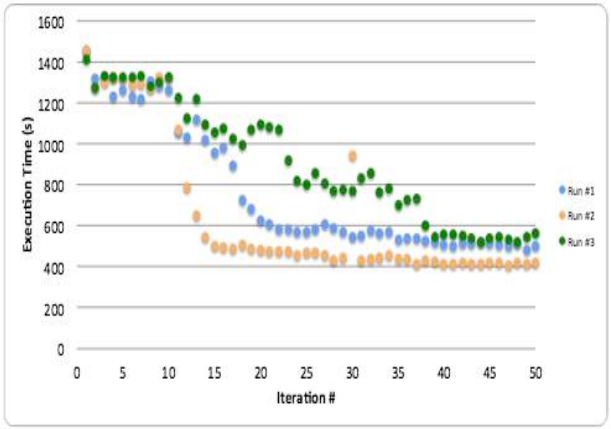
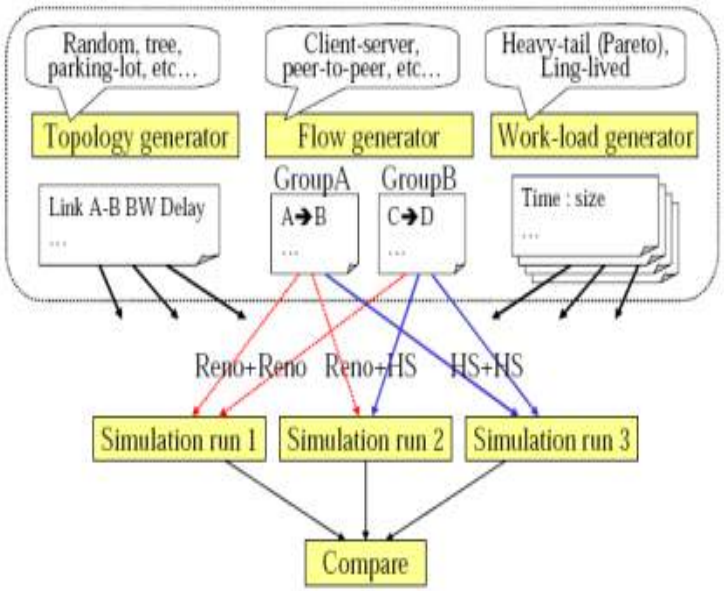


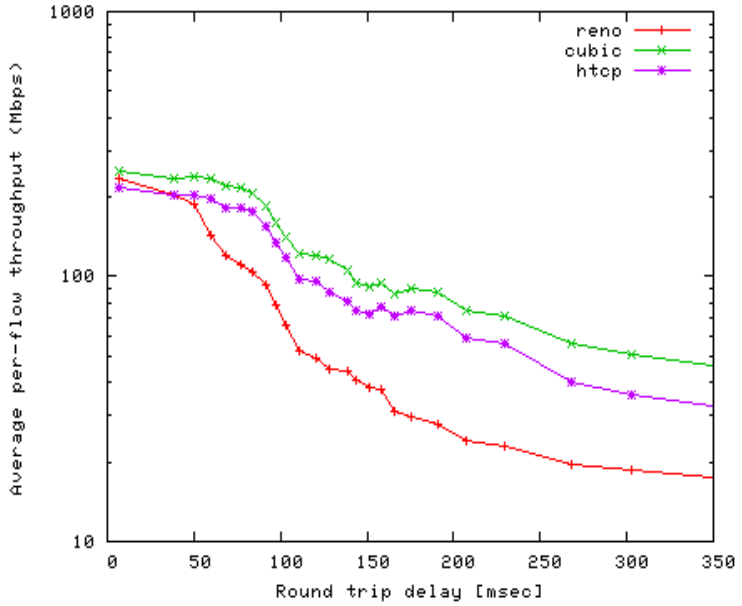
Fig. 5: Plot for the execution time for three runs of 50 iterations using COBYLA.

Desani, D., Gil-Costa, V., Marcondes C., Senger, H. "Black-box Optimization of Hadoop Parameters Using Derivative-free Optimization Methods". Submitted to CloudCom 2015.

Limitations

- For another example, transport layer protocol
  - Insufficient aggregate bandwidth in testbeds
  - Forced to change settings: bottleneck, delay and cross-traffic conditions





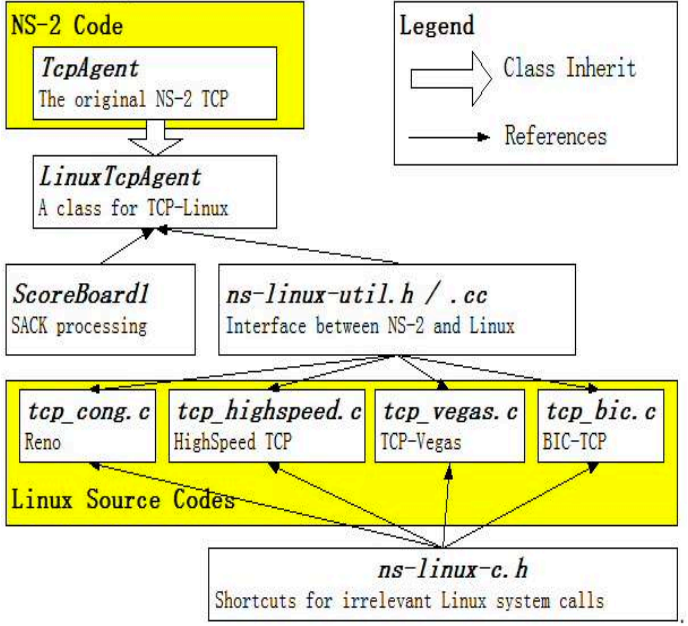
Shimonishi, H.; Sanadidi M.; Murase T. "Assessing Interactions among Legacy and High-Speed TCP Protocols". International Workshop on Protocols for Fast Long-Distance Networks (PFLDNet2007)

Testbeds and Simulation

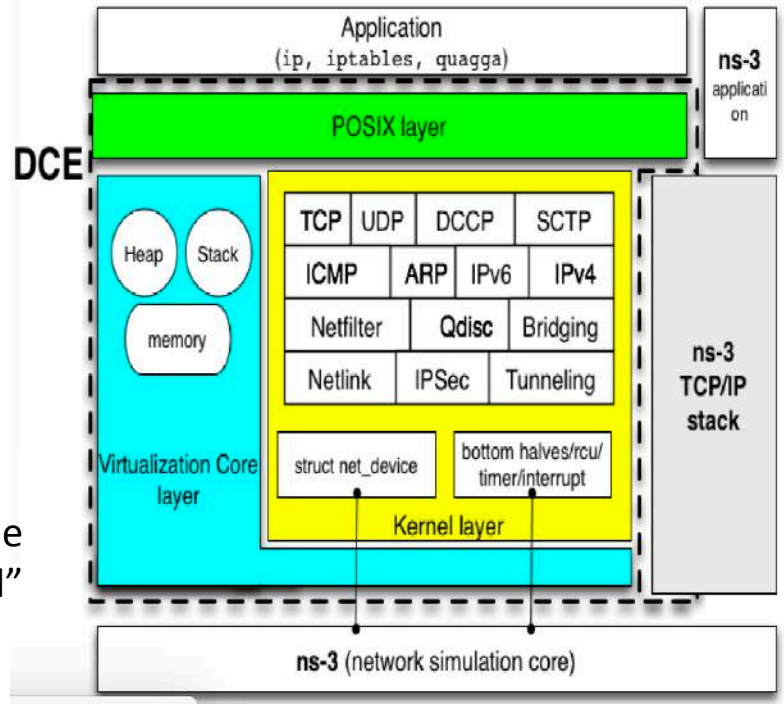
- Survey of SIGCOMM papers (2007-2013)
  - Physical Testbeds and Simulation
    - A large proportion of the evaluative work
    - Often used in complementary roles in the evaluative studies
  - Testbeds
    - For small-scale real-world studies
  - Simulation
    - For speculating scaling properties
    - Use simplified models
- Questions are:
  - how to bring them even closer?
  - How to achieve flexibility, scale, and realism?

Direct Code Execution in ns-3

- Recompile application code
- Provide kernel space code as shared library
- Use simulation clock
- Results show scalability issues (too costly to run)

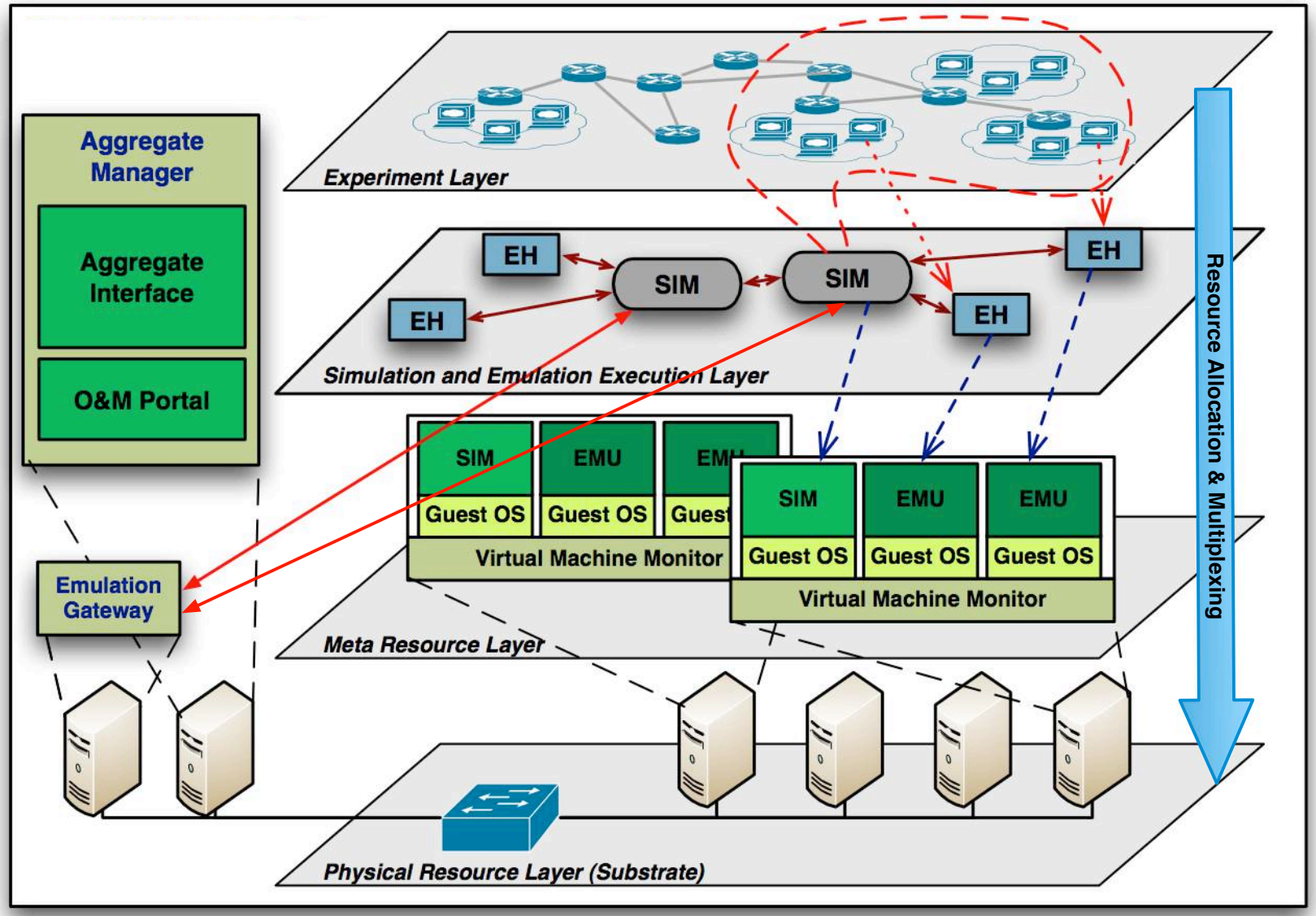


TCP Linux in ns-2 Implementation - Caltech



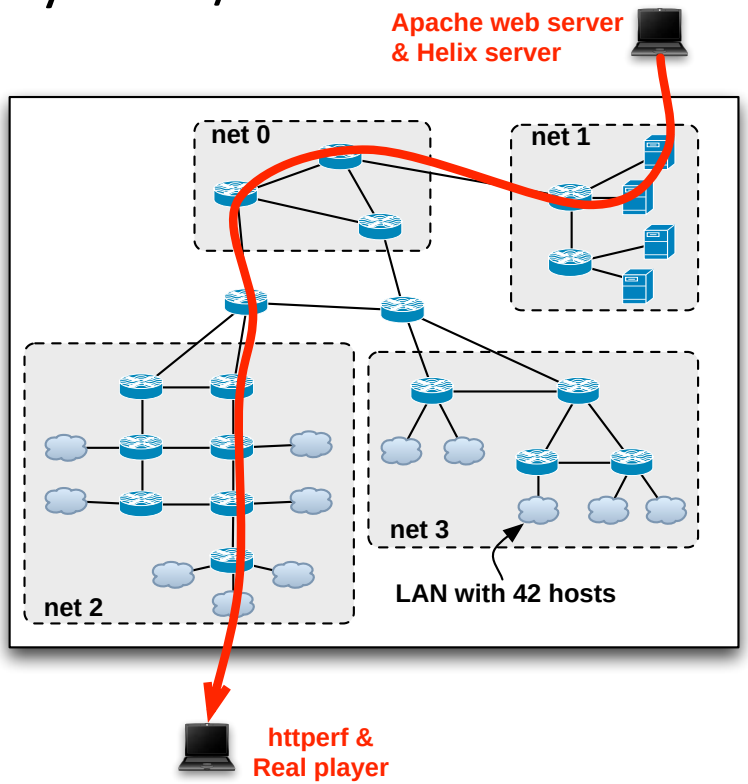
H. Tazaki. “Extracted from ns-3 Direct Code Execution tutorial”

PrimoGENI for Hybrid Experimentation

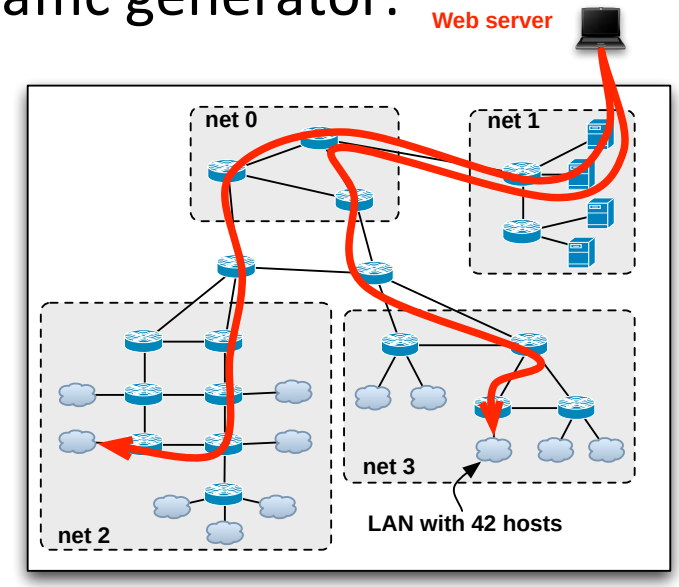


Some Use Cases

A delay node/network:

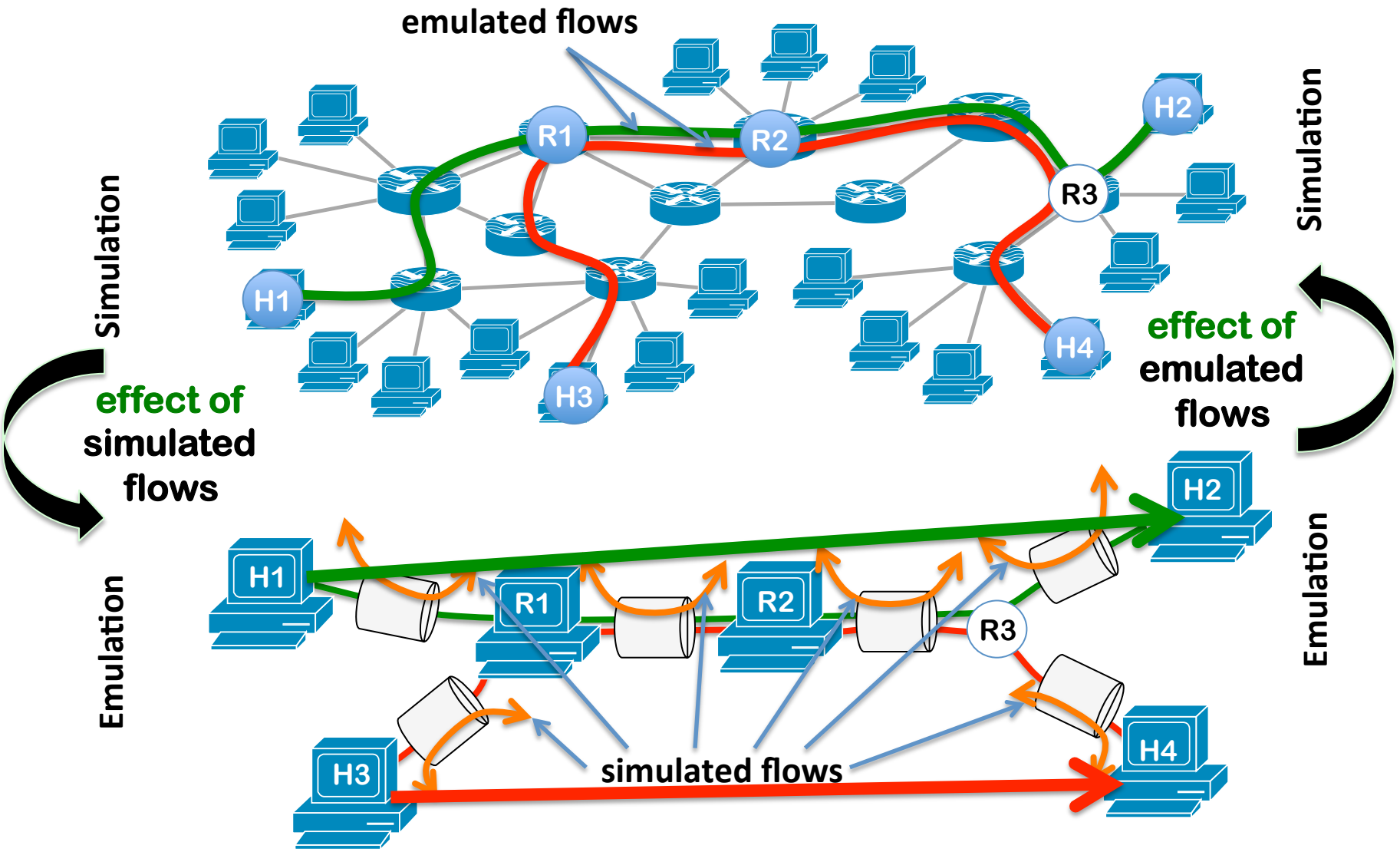


A traffic generator:

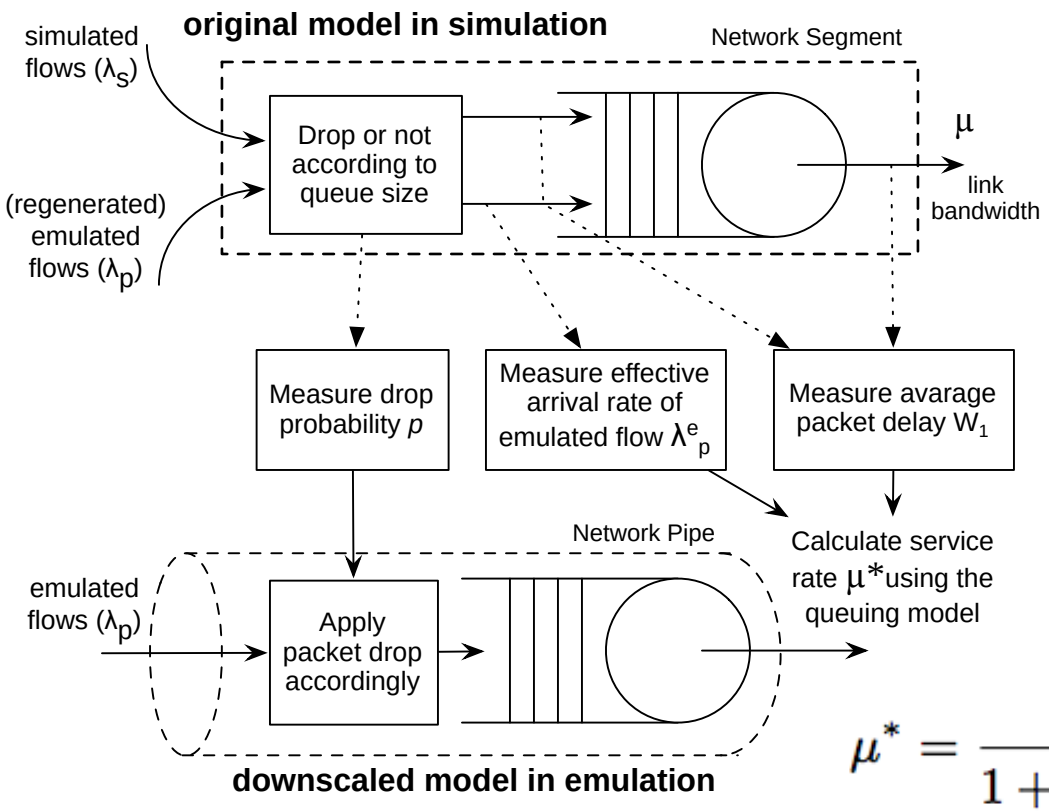


A virtual distributed environment:

Our Symbiotic Approach in a Nutshell



Steady-State Queuing Model



Single-link Segment

Assuming M/D/1 Queue

$$\mu^* = \frac{\lambda_p^e}{1 + W_1\lambda_p^e - \sqrt{1 + W_1^2(\lambda_p^e)^2}}$$

What About Transient Effect?

- Well, steady-state does not work!
- Closed-form solution for transient effect is rather elusive, even for Poisson arrivals
- We invent a “control nob” to dynamically adjust μ* from measurements

Adjust for Transient Effect

excess queue length

avg pkt delay in emulation

avg pkt delay in simulation

$$\Delta L(t) = \mu^*(t)(W_2(t) - W_1(t))$$

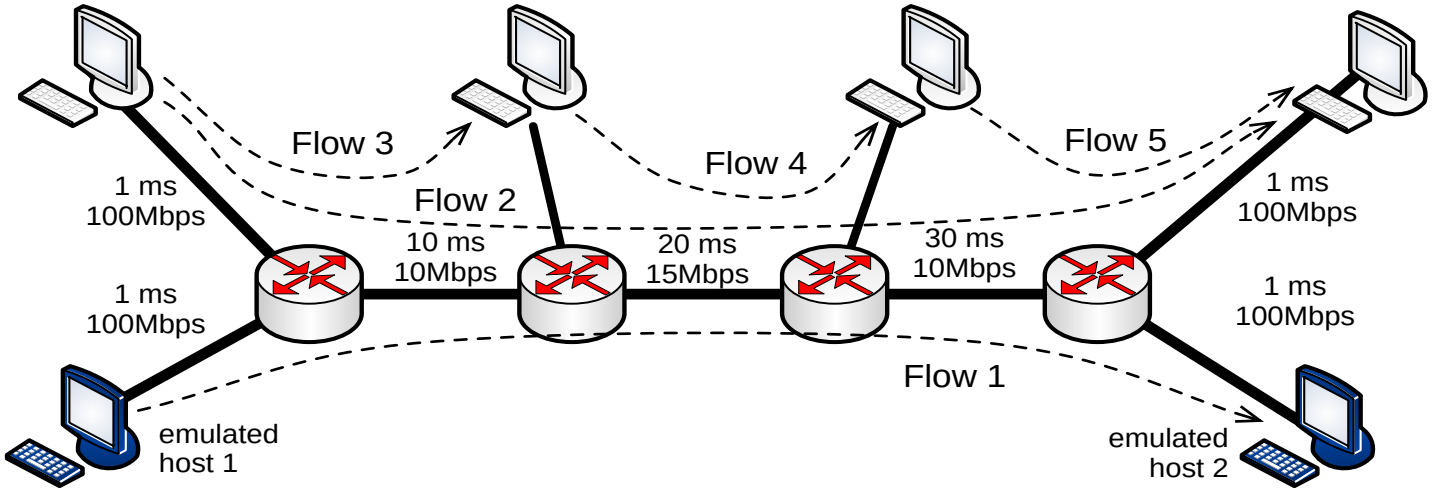
excess service rate

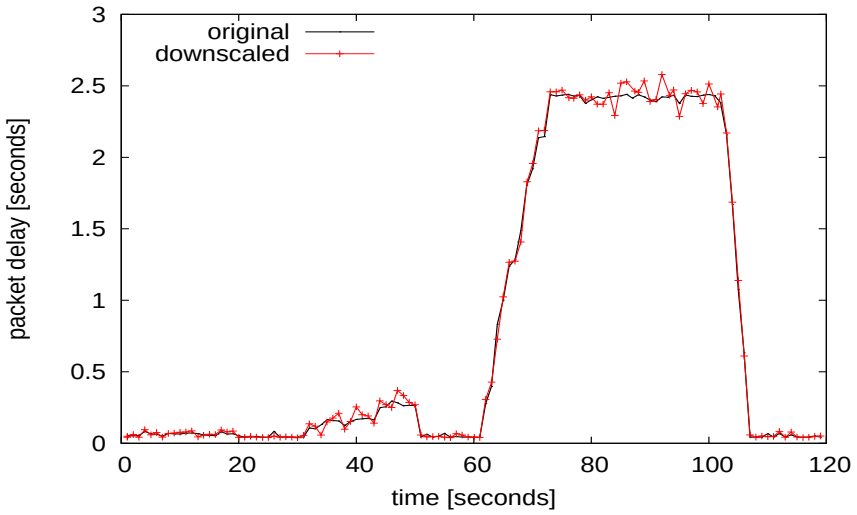
$$\Delta\mu^*(t) = \frac{\Delta L(t)}{\Delta T} = \frac{\mu^*(t)(W_2(t) - W_1(t))}{\Delta T}$$

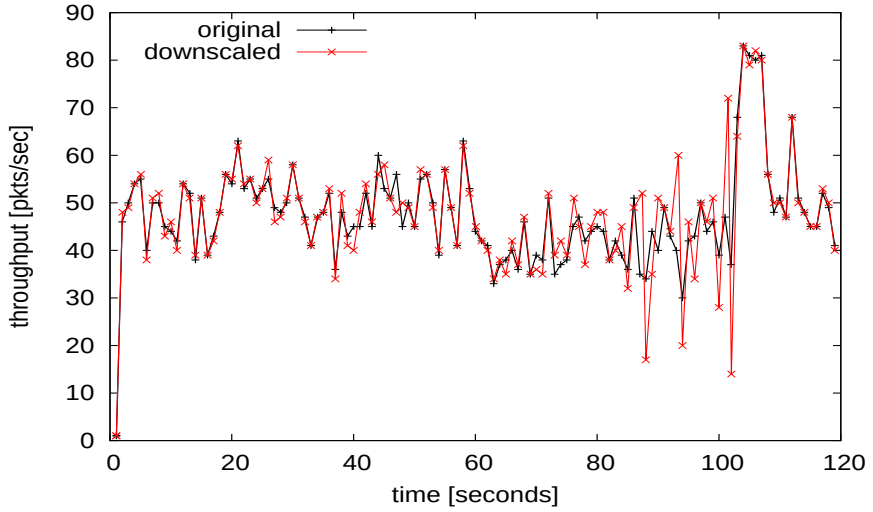
$$\begin{aligned} \hat{\mu}(t) &= \mu^*(t) + \Delta\mu^*(t) \\ &= \frac{\lambda_p^e(\Delta T + W_2(t) - W_1(t))}{\Delta T(1 + W_1(t)\lambda_p^e(t) - \sqrt{1 + W_1^2(t)\lambda_p^e(t)^2})} \end{aligned}$$

- The adjustment effectually forces the emulation system to "track" the simulated network conditions at each update interval.
- The result can be extended to network segments with multiple links.

Validation


Flow 3
Flow 2
Flow 4
Flow 5
Flow 1
1 ms
100Mbps
1 ms
100Mbps
10 ms
10Mbps
20 ms
15Mbps
30 ms
10Mbps
1 ms
100Mbps
1 ms
100Mbps
emulated
host 1
emulated
host 2


original
downscaled
packet delay [seconds]
time [seconds]
0
0.5
1
1.5
2
2.5
3
0
20
40
60
80
100
120


original
downscaled
throughput [pkts/sec]
time [seconds]
0
10
20
30
40
50
60
70
80
90
0
20
40
60
80
100
120

Symbiosize This: Mininet

- Use lightweight containers (network namespaces)
  - Can create 100s of containers per host
- Easy to program setup
- Uniform and easy API
- API can be explored to create a distributed emulation, for larger scale experiments
- However, still quite resource limited

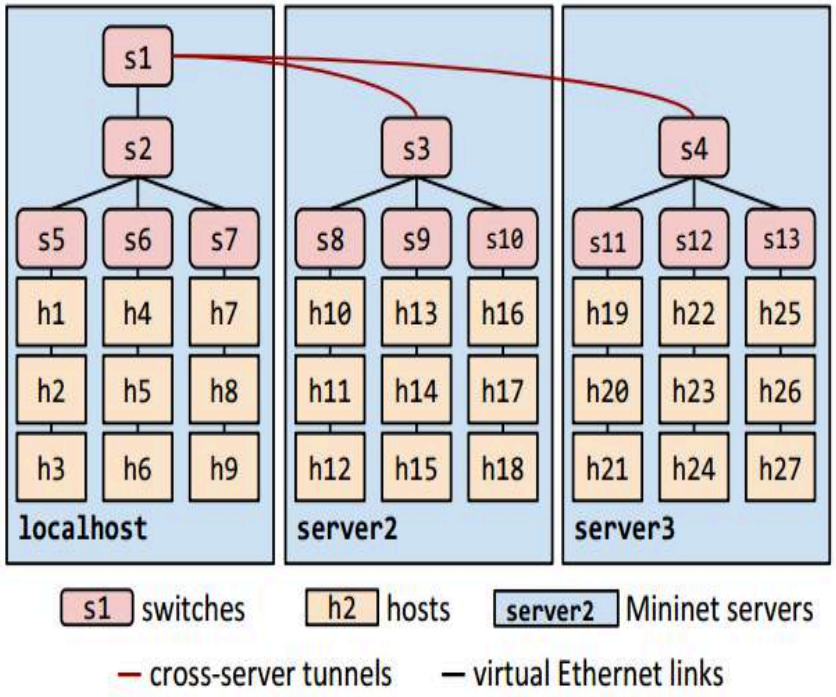


Figure 1: Sample Mininet Cluster

```
topo = Tree(depth=3, fanout=3)
servers = ['localhost','server2','server3']
net = MininetCluster(topo=topo, servers=servers)
net.start()
CLI(net)
net.stop()
```

...or via simple command line options:

```
# mn --topo tree,depth=3,fanout=3
     --cluster localhost,server2,server3
```

Downscaling process

- Start by marking emulated hosts that will be instantiated as containers and capable of directly running any app (iperf)
- Downscaling process
  - Prune the network model and remove all hosts, routers, and links not traversed by emulated traffic
  - compresses the pruned topology by combining the intermediate nodes and links visited by the same set of emulated flows into a single network pipe
  - Example: network segment between h1 and r2 - single pipe
- In our workflow, we need downscaled file

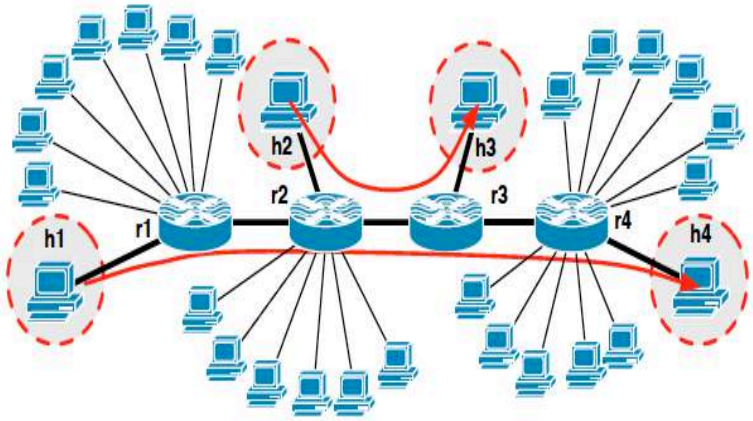


Fig. 2: A target virtual network with emulated traffic identified.

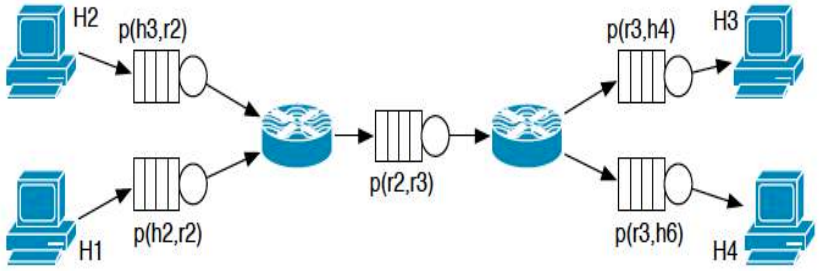


Fig. 3: A downscaled network model to run in Mininet.

Distributed Option

- Symbiotic approach can support distributed emulation
- Multiple mininet instances operating in parallel
- Unbounded amount of traffic to be affected by emulation
  - Example: flow h2 to h3 can be emulated in a separate Mininet than flow from h1 to h4.
- Note that the state of the network pipe, p(r2; r3), is mirrored on both instances; that is, they will be controlled by the simulator with the identical link properties.

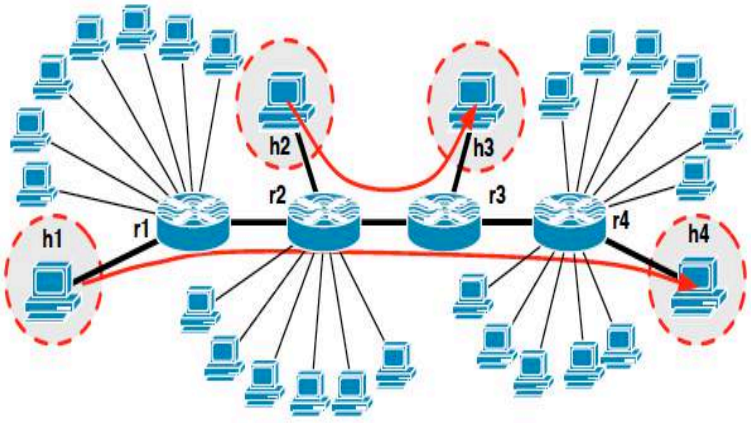


Fig. 2: A target virtual network with emulated traffic identified.

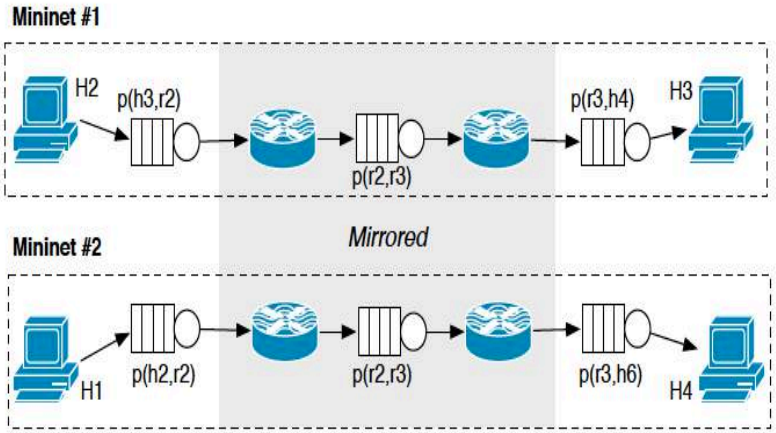


Fig. 5: Downscaled models for two Mininet instances.

Mininet Symbiosis

- In our prototype we make use of:
  - Mininet
  - Primex Simulator
- We keep similar mininet API
- Under the hood:
  - **traffic monitor** used to collect measurements at each queue traversed by the emulated flows, including packet drop probability, arrival rate of emulated flow, queuing delay
  - **traffic generator** collect information from Mininet about the traffic demand d from applications for each emulated flow

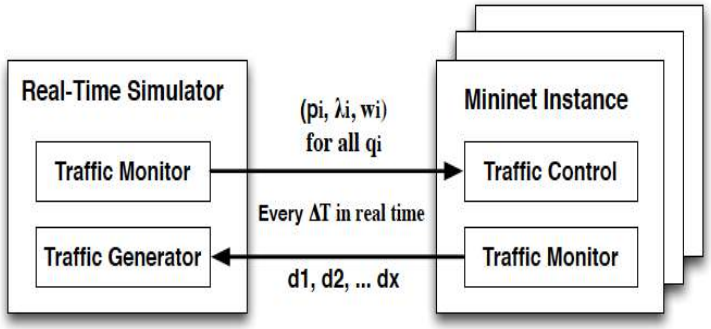


Fig. 4: Mininet symbiosis setup.

```
#!/usr/bin/python

import threading
import sys
import socket
from mininet.net import Mininet
from mininet.node import OVSController
from mininet.link import TCLink

def traffic_controller( net, mn_pipes_file, sock ):
def traffic_monitor( net, mn_pipes_file, demand_file,

def main():
    # Parsing downscale file
    queues, pipes, flows = file.parseDownscale()

    # Create a mininet network
    net = Mininet( controller=OVSController )
    net.addController( 'c0' )

    info( '*** Adding hosts\n' )
    for sflow in flows
        hosts[i] = net.addHost()
        hosts[i+1] = net.addHost()

        for pipe in pipes
            totalbw = min(totalbw, queue[2])
            totaldelay += int(queue[1])

        linkopts = dict( bw=totalbw, delay='%sms' % to
        net.addLink( hosts[0], hosts[1], cls=TCLink, *

    info( '\n*** Starting network\n' )
    net.start()
    CLI(net)
    net.stop()
```

Conclusions

- Symbiotic simulation provides promising tradeoff
  - a realistic environment for running network applications
  - simulation to provide more flexible, large, and complex network scenarios
- Feasibility study while we currently undergo a full-scale implementation
- We are interested in simulator be able to connect with multiple Mininet instances to support large-scale experiments
- Use symbiosis to study bandwidth-intensive OpenFlow applications (Future Internet), which would otherwise be difficult to realize in the traditional simulation or emulation testbeds