Towards High Fidelity Network Emulation

Lianjie Cao, Xiangyu Bu, **Sonia Fahmy**, Siyuan Cao

Department of Computer Science

PURDUE UNIVERSITY

This work has been sponsored in part by NSF grant CNS-1319924

How to experimentally evaluate an idea?

Network Simulator

ns-2 NETWORK SIMULATOR

ns-3 NETWORK SIMULATOR

OPNET

Network Testbed

emulab total network testbed

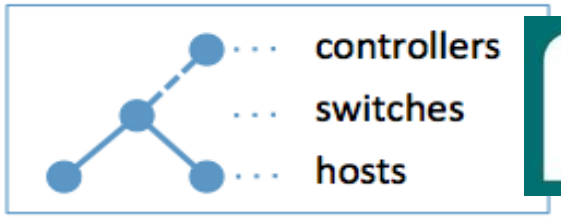
The DETER Project

geni Exploring Networks of the Future

PLANETLAB
An open platform for developing, deploying, and accessing planetary-scale services

Network Emulator

> sudo mn



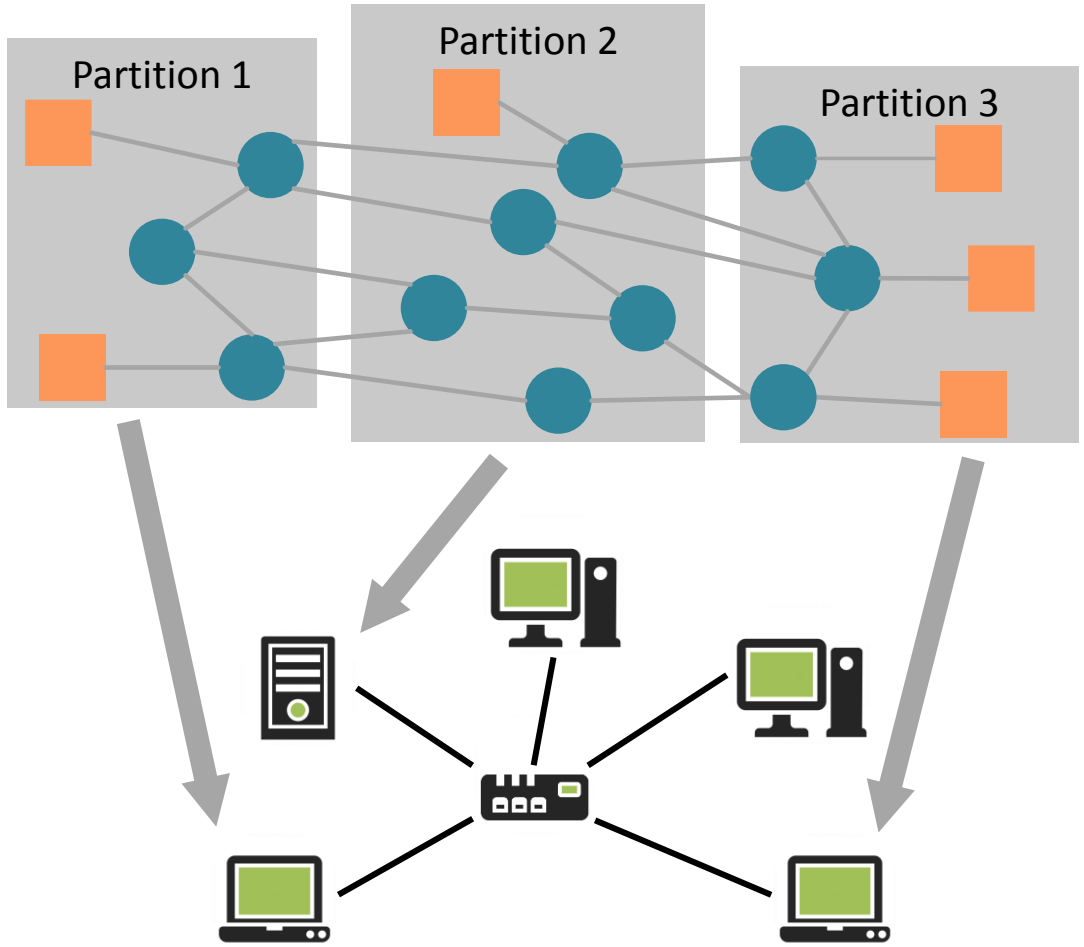
EstiNet

DOT
Distributed OpenFlow Testbed

How to map a networked app onto infrastructure?

Emulated hosts

Containers, virtual machines, ……



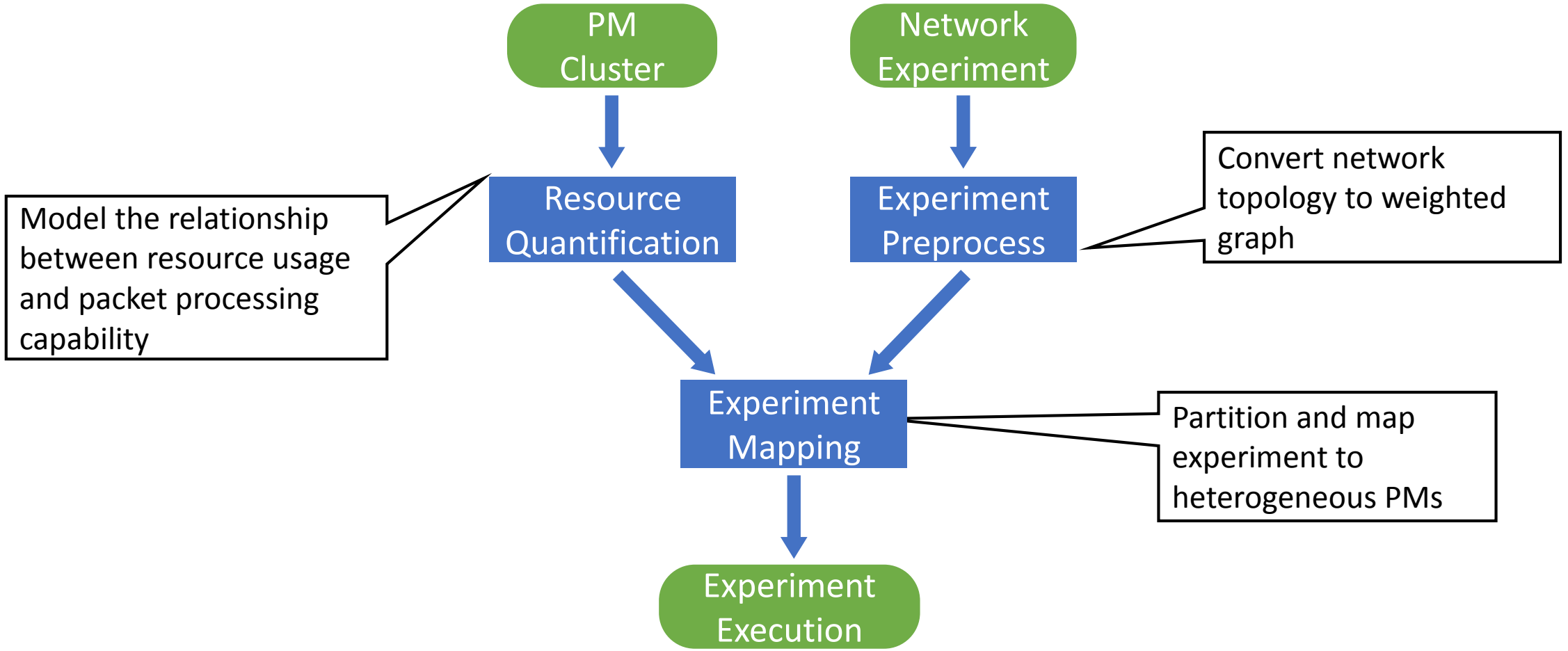
Emulate network devices

Open vSwich, Indigo Virtual Switch, ……

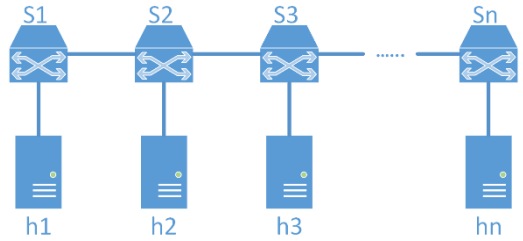
Problem

- In a distributed network emulator running on heterogeneous PMs, how we can profile the physical resources and map a network experiment onto the PMs with high performance fidelity?
- Challenges
  - How to quantify the physical resources in heterogeneous cluster?
  - How to partition a network experiment to achieve high performance fidelity?
  - How to allow resource multiplexing on the same cluster?
- Design principles
  - Integrity and fidelity
  - Best effort
  - Judicious use of resources

Design


PM Cluster
Network Experiment
Resource Quantification
Experiment Preprocess
Model the relationship between resource usage and packet processing capability
Convert network topology to weighted graph
Experiment Mapping
Partition and map experiment to heterogeneous PMs
Experiment Execution

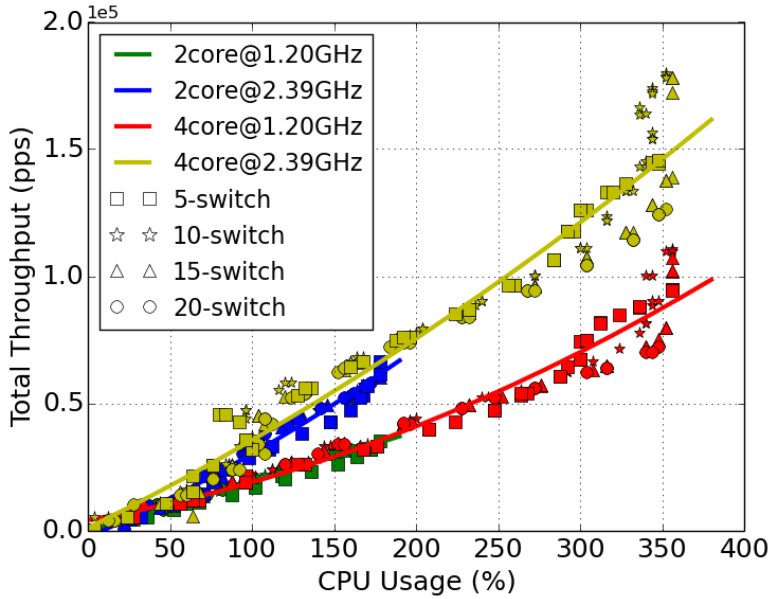
Resource Quantification



Traffic Generation
(packet sizes, software switches, topology sizes, etc.)

Resource Usage Collection
(CPU, memory, network)

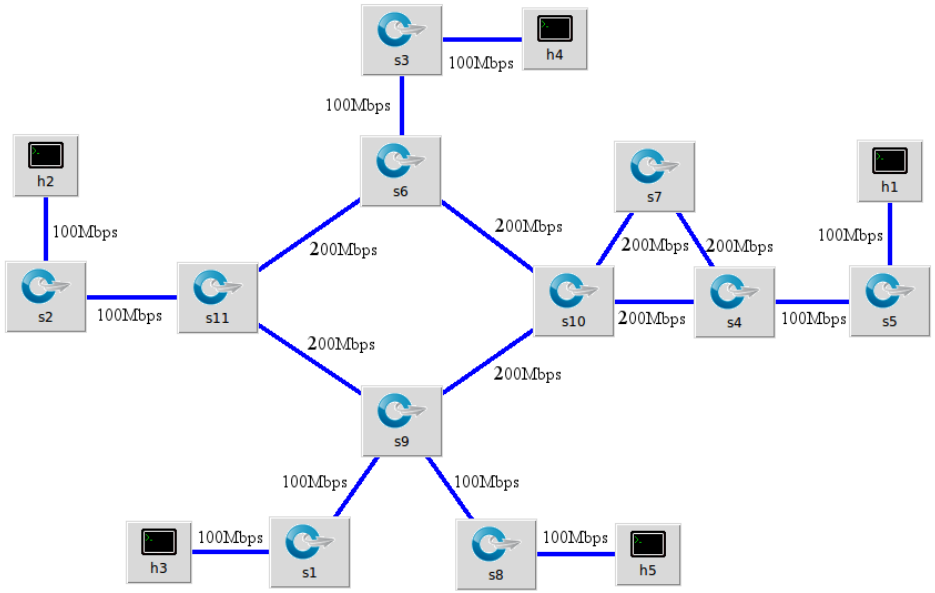
Model Fitting
(Polynomial regression)



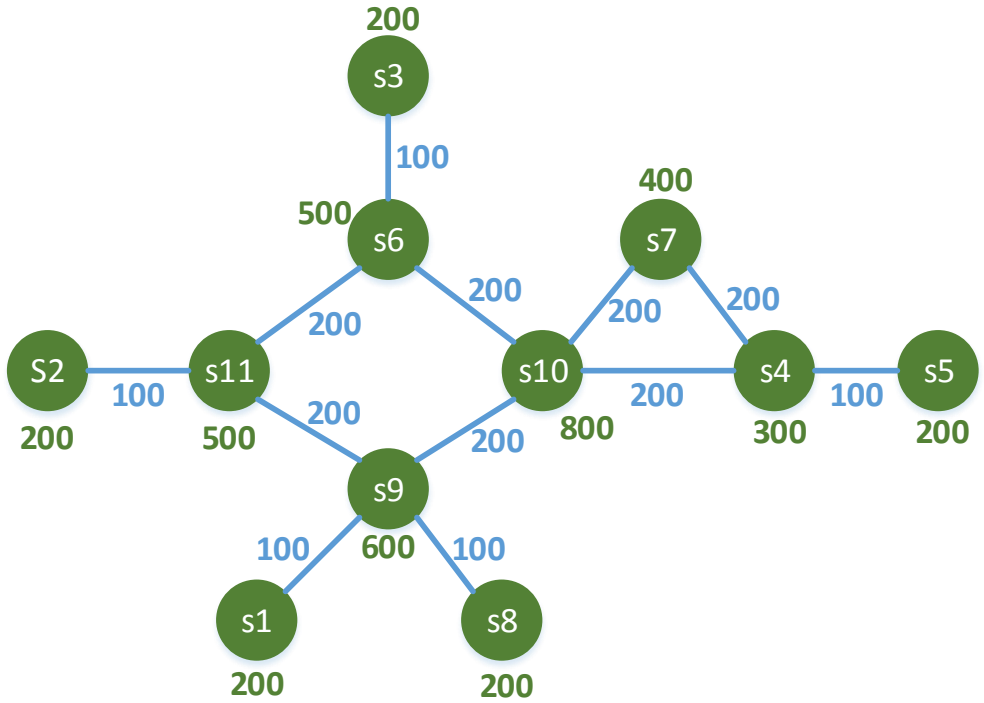
$$P_{2core@1.20GHz}(u) = 0.0168u^2 + 192.944u - 286.828$$
$$P_{2core@2.39GHz}(u) = 0.425u^2 + 285.166u - 2709.699$$
$$P_{4core@1.20GHz}(u) = 0.359u^2 + 112.275u + 4061.292$$
$$P_{4core@2.39GHz}(u) = 0.279u^2 + 316.796u + 948.393$$

Topology Abstraction

Mininet Topology



Graph *G=(V, E)*



- Collapse end hosts to adjacent switches
- Edge weight $w((a,b))$ = link bandwidth
- Vertex weight $w(v) = \sum w(a,b)$, where a=v or b=v

Partitioning and Mapping

- Objectives
  - Avoid PM overload ➔ performance fidelity
  - Maximize PM utilization ➔ resource multiplexing
  - Minimize edge cut ➔ traffic localization
- Input
  - Weighted graph *G = (V, E)*
  - Host resource requirements (*e.g.*, CPU)
  - Information of k PMs (e.g., PM capacity functions and CPU shares)
- Output
  - Subgraphs $S_1, S_2, \ldots\ldots, S_{k'}$, where $k' < k$

Mapping Algorithm

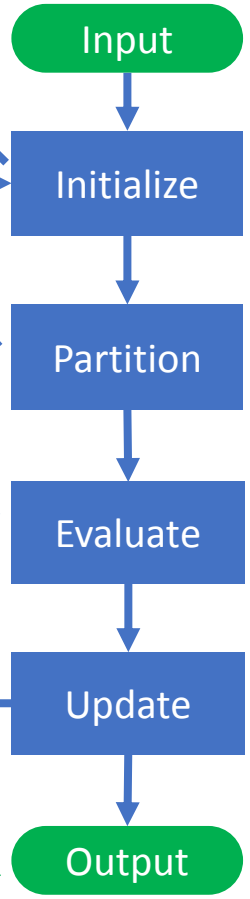
1. Compute packet processing capacity
2. Select minimal # of PMs
3. Normalize host CPU and capacity of selected PMs

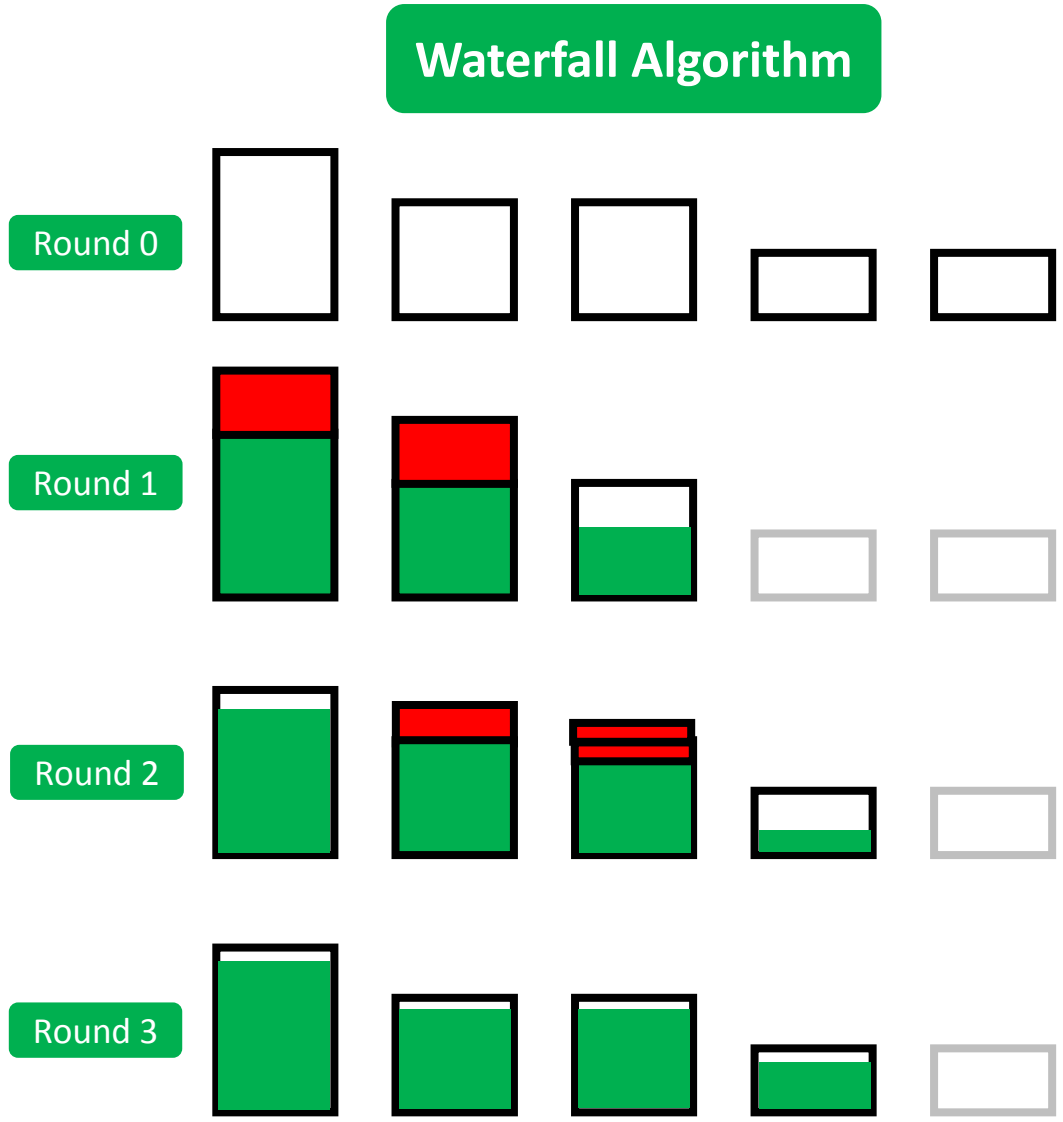
Invoke METIS with normalized input

1. Compute host CPU and capacity used on each selected PM
2. Convert capacity value to CPU for switches
3. Store and rank this result in a hash set based on i) # of PM, ii) overutilization and underutilization and iii) edge cut
4. Create new exploration branch if new PM needed

1. Decrease PM target CPU share if overloaded
2. Increase PM target CPU share to compensate for deductions from overloaded PMs

1. No new branches
2. Termination counter exhausted for all branches
3. Select best result as output





Evaluation

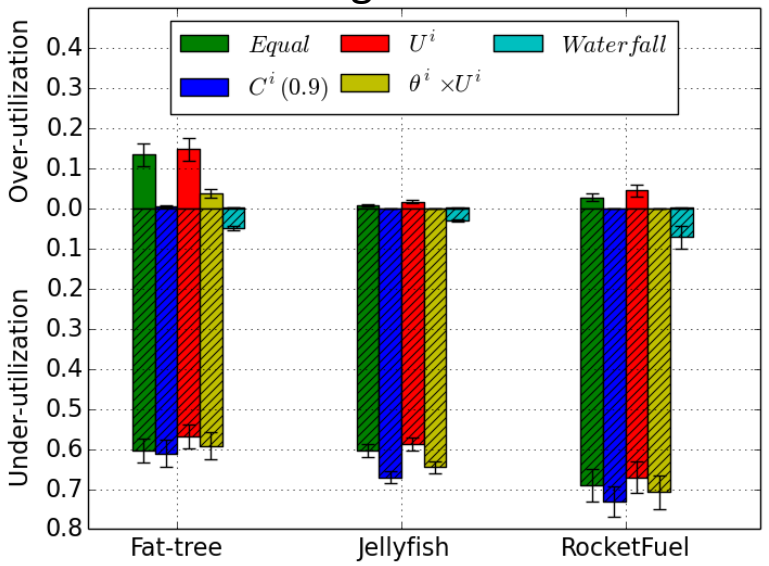
- Simulation
  - Evaluate Waterfall with various network topologies and cluster configurations
- DDoS experiments
  - Evaluate Waterfall with testbed experiments
- Comparison
  - *Equal* ➔ Equal-sized partitioning using METIS
  - U^i ➔ Use max CPU shares of PMs for METIS
  - $\theta^i \times U^i$ ➔ Use adjusted max CPU shares of PMs for METIS
  - $C^i(0.9)$ ➔ Use 90% of max packet processing capacity for METIS
  - *SwitchBin* ➔ Default choice of Mininet cluster mode

Simulation

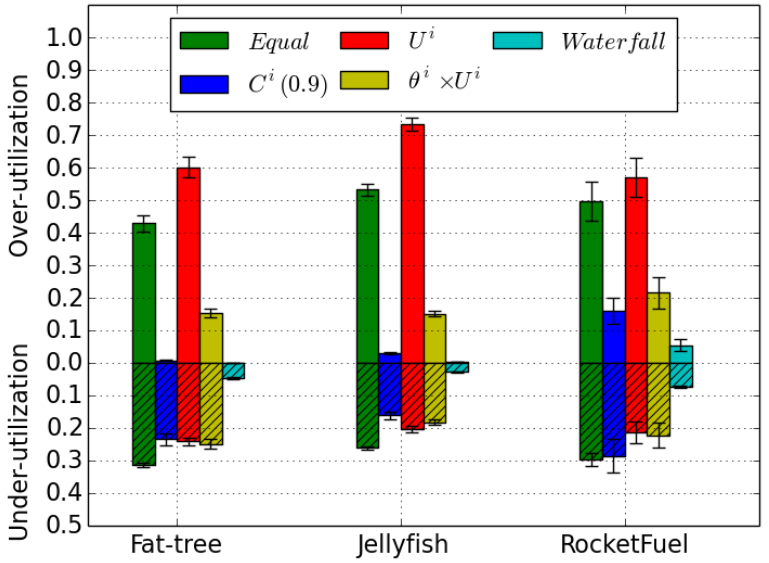
- Network topologies
  - RocketFuel, Jellyfish and Fat-tree
  - 41 ~ 670 nodes and 96 ~ 6072 edges
- Simulated clusters
  - Large clusters: 21 PMs (sufficient resources)
  - Medium clusters: one cluster per topology (just enough resources)
  - Small clusters: one cluster for each topology (insufficient resources)
- Metrics
  - Degree of overutilization: $\frac{\hat{u}^i - U^i}{U^i}$
  - Degree of underutilization: $\frac{U^i - \hat{u}^i}{U^i}$ for large and medium clusters
  - Standard error of overutilization for small clusters
  - Edge cut

Simulation Results

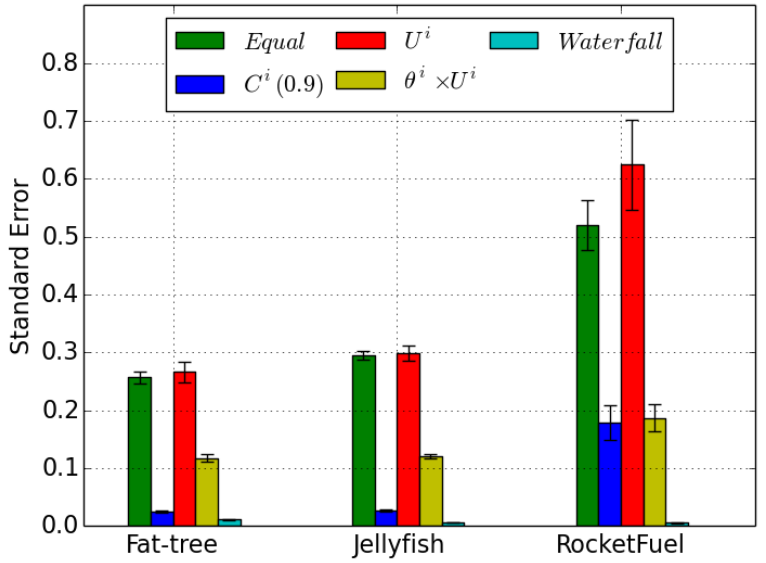
Large Cluster



Medium Cluster



Small Cluster



Select fewer PMs

↓

Judicious use of resources

Low overutilization and underutilization

↓

Integrity and fidelity

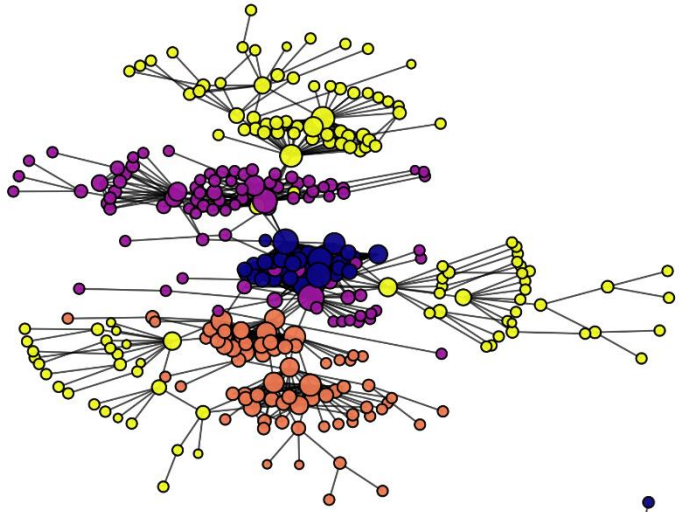
Balance overutilization on heterogeneous PMs

↓

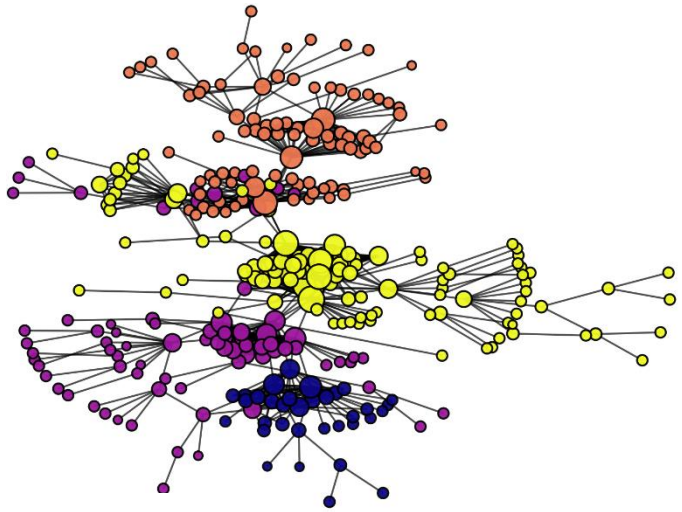
Best effort

RocketFuel Example

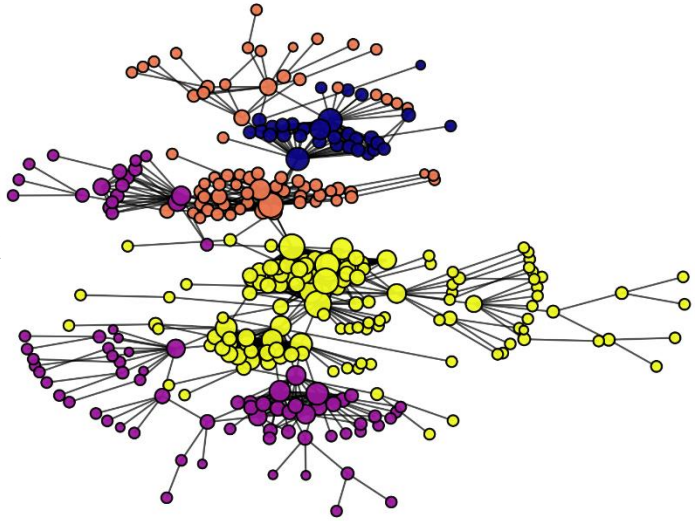
Equal



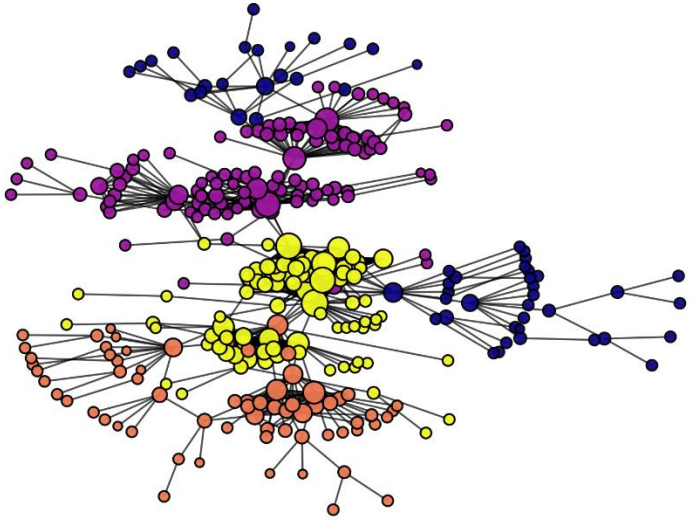
U^i



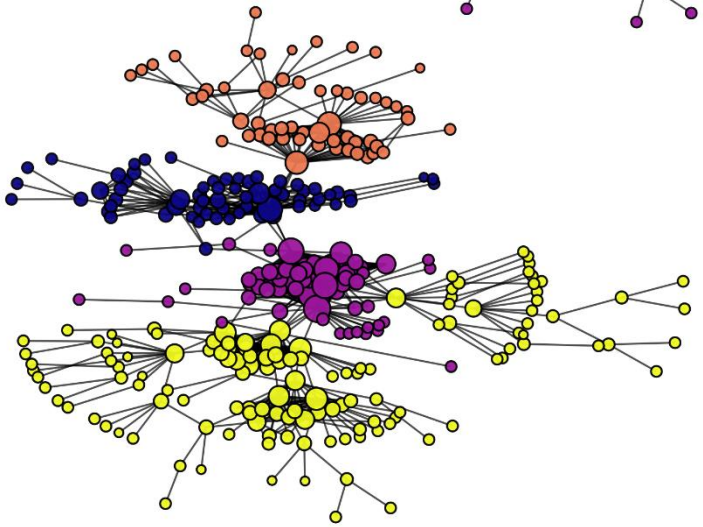
$\theta^i \times U^i$



$C^i(0.9)$



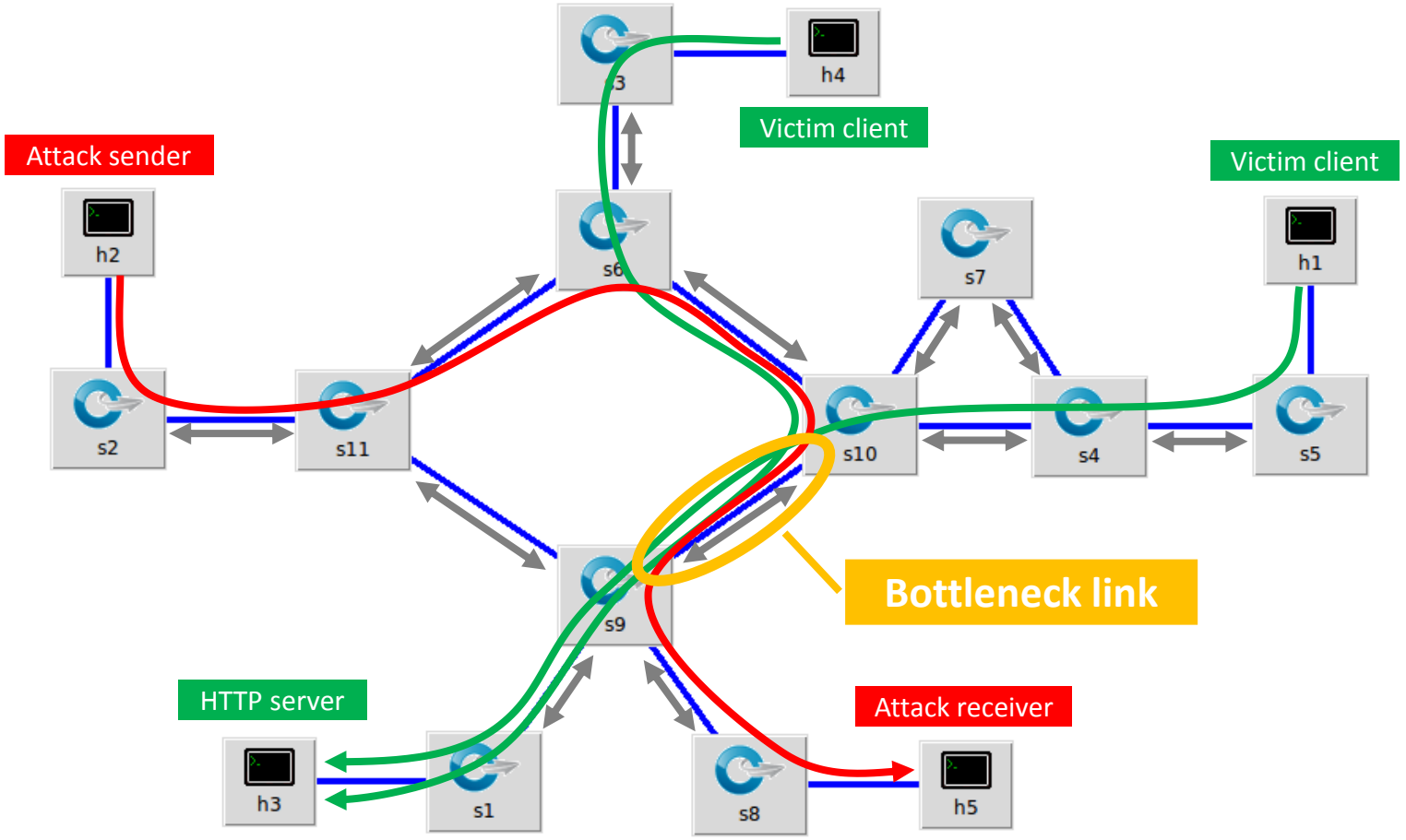
Waterfall



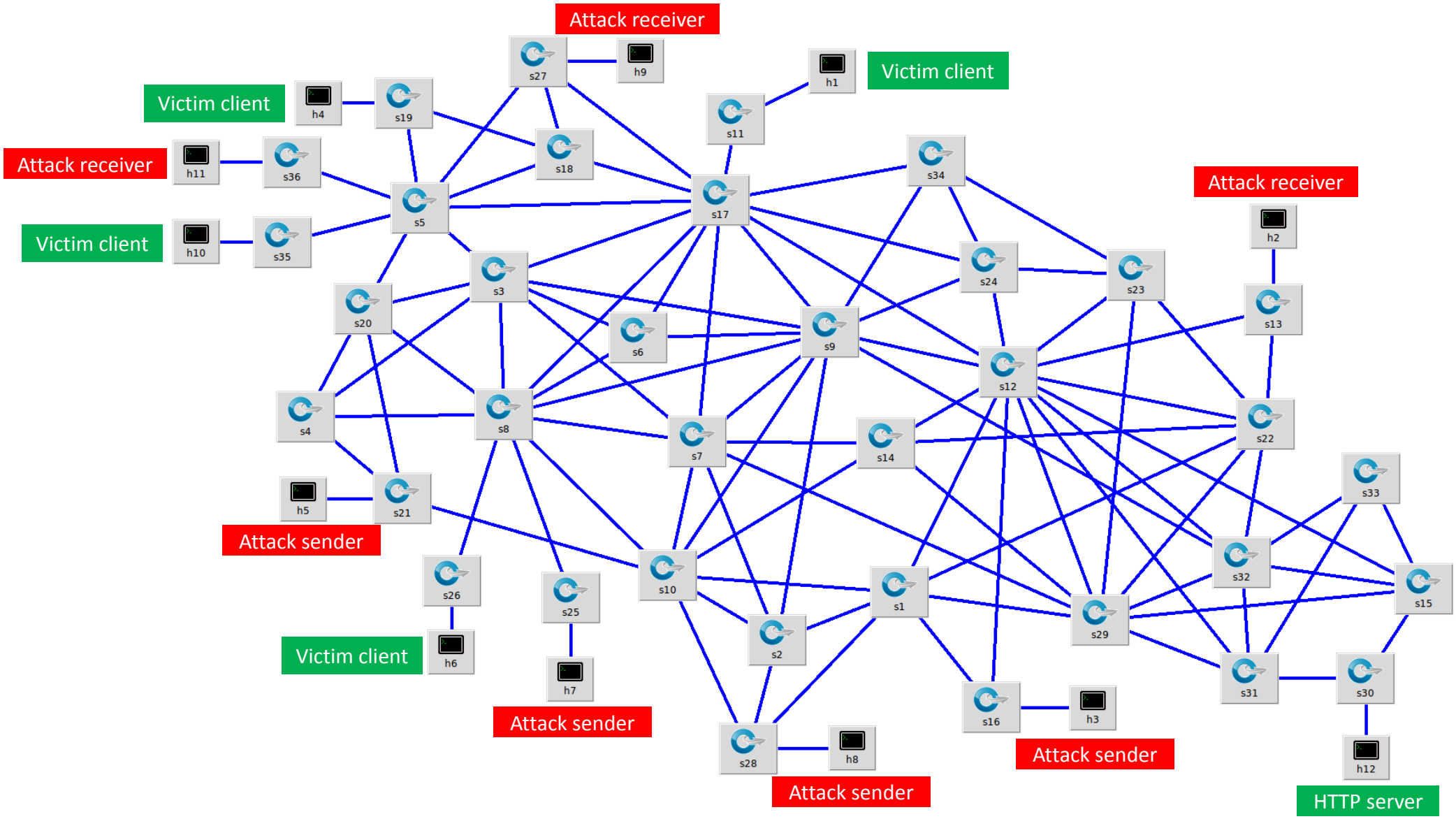
DDoS Experiments

- Network topologies
  - Small-scale topology: RocketFuel with 11 switches and 5 hosts
  - Medium-scale topology: RocketFuel with 36 switches and 12 hosts
- Network traffic
  - Background traffic: UDP traffic on all links
  - HTTP traffic: HTTP traffic between victim clients and HTTP server
  - Attack traffic: UDP traffic between attack senders and receivers
  - HTTP traffic and attack traffic share certain bottleneck links
- Metrics
  - CPU utilization of PMs
  - Link utilization of experimental topology
  - Completed HTTP requests

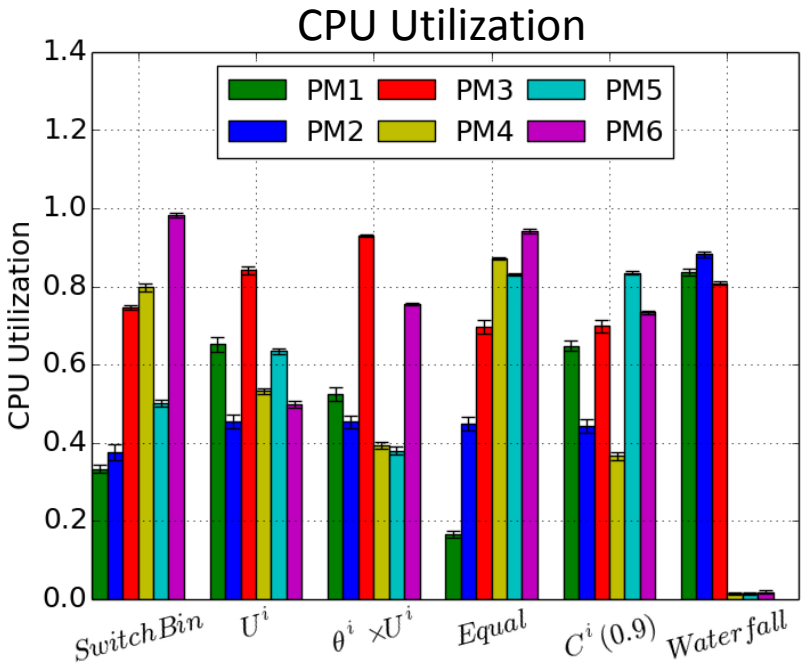
Small-scale DDoS

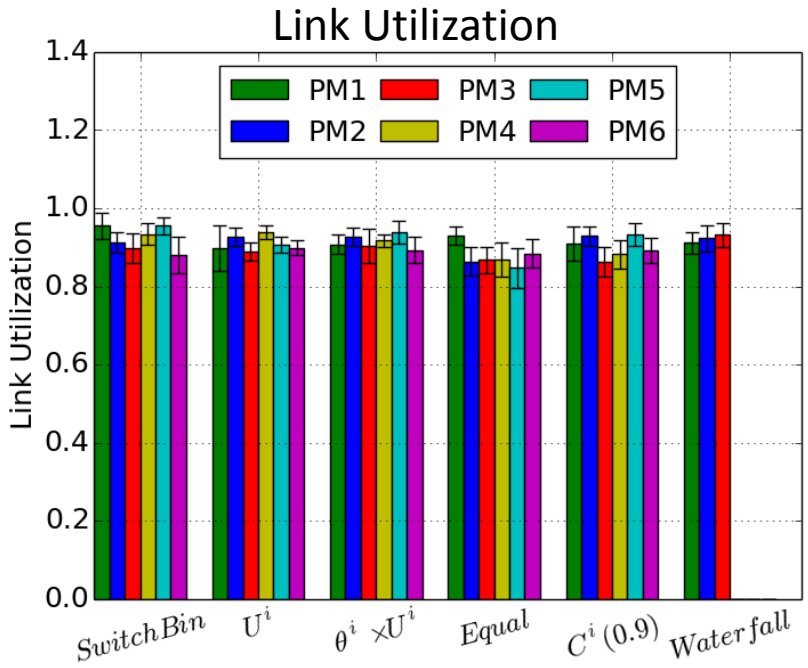


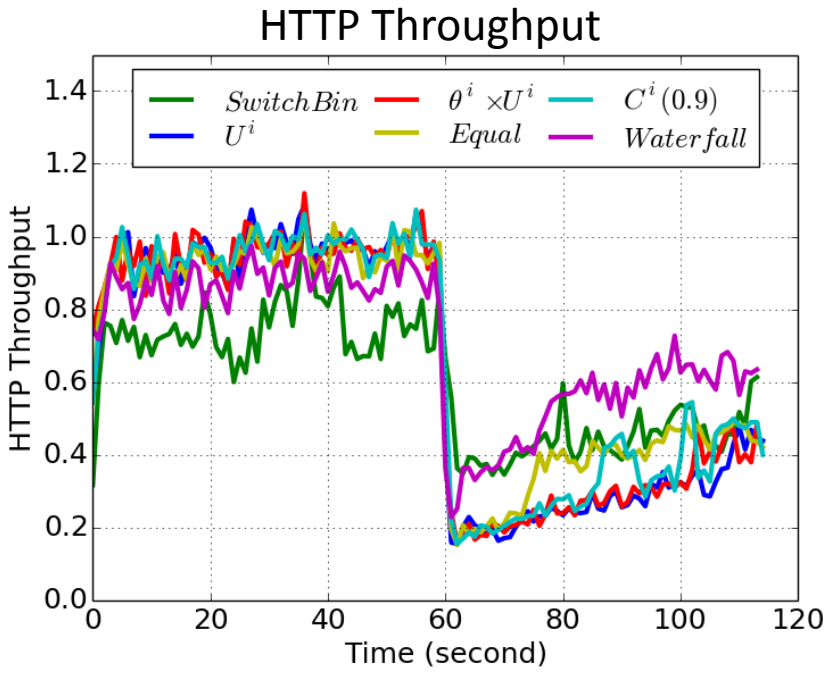
Medium-scale DDoS



Results for Small-scale DDoS







< 90% CPU usage on selected PMs

> 90% link utilization

HTTP throughput drop

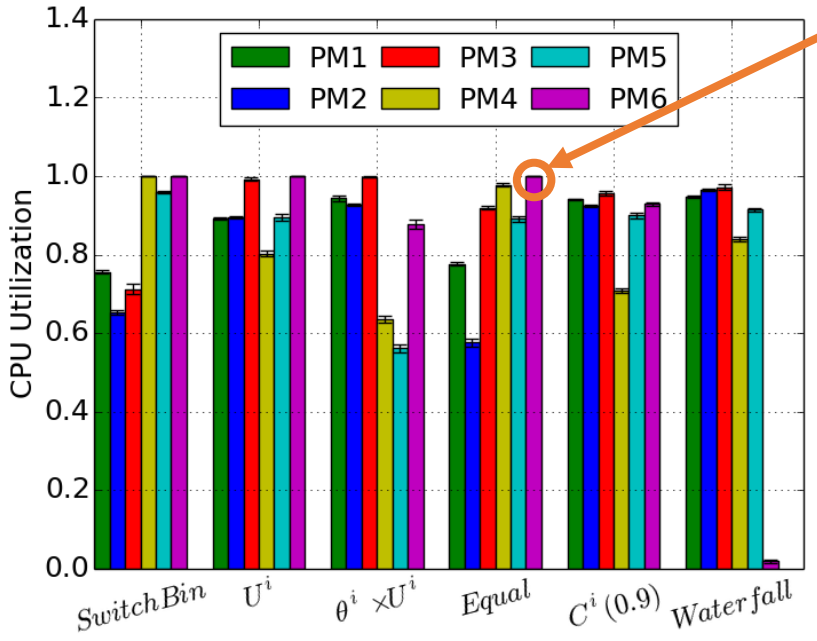
No PM overload

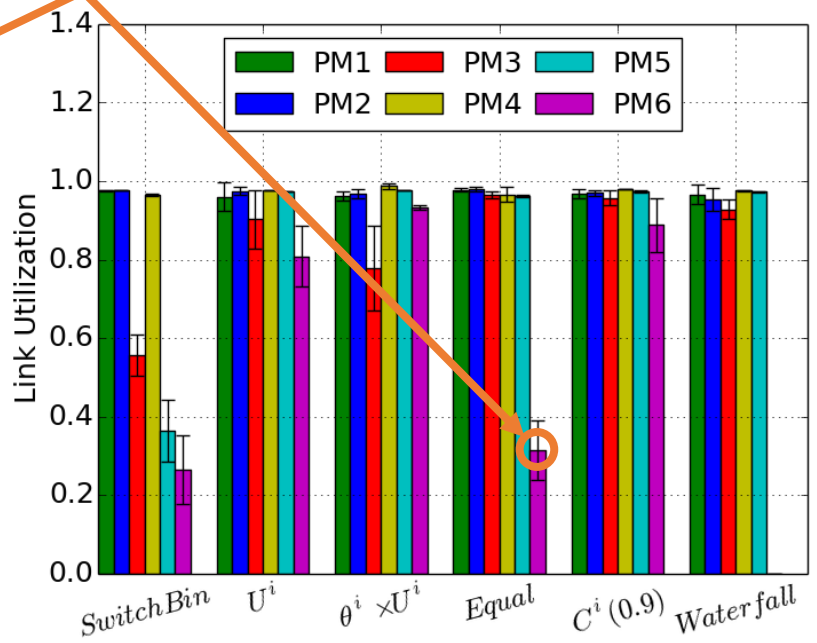
High performance fidelity

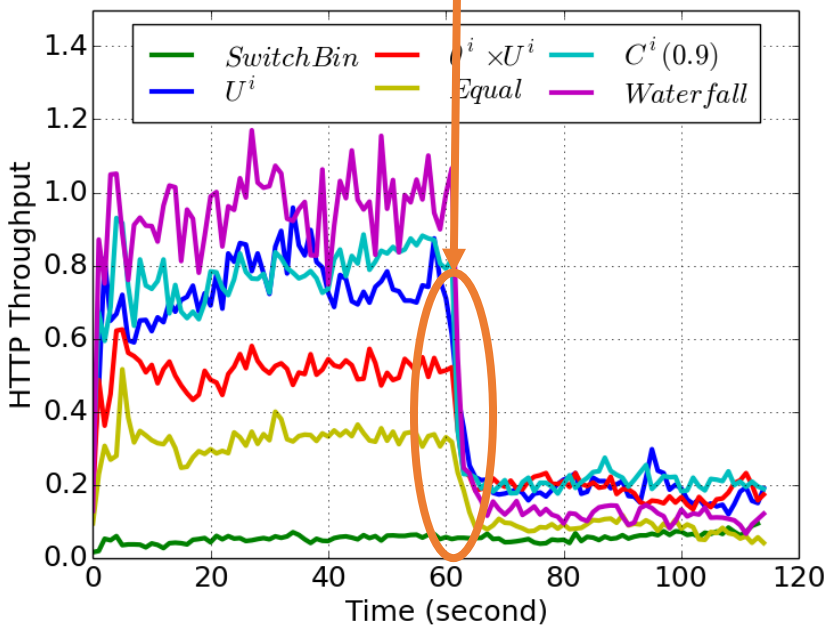
Results for Medium-scale DDoS

Performance fidelity loss due to PM overload

Insufficient CPU allocation to end hosts







Waterfall

- Selects fewer PMs
- Achieves more balanced resource usage
- Allocates desired resources to hosts and switches
- Maintains high performance fidelity

Related Work

- Graph partitioning
  - Kernighan-Lin (KL) algorithm: Kernighan@BSTJAN'70.
  - Spectral algorithms: Pothen@SIMAX'90, Hendrickson@SISC'95.
  - Multilevel algorithms/software: Barnard@PPSC'93, Hendrickson@SC'95, METIS, Chaco.

- Network embedding and virtualization
  - VM placement: Jiang@INFOCOM'12, Kuo@INFOCOM'14.
  - Virtual network embedding: Chowdhury@ToN'12 (ViNEYard), Yu@CCR'08.

- Testbed mapping
  - Ricci@CCR'03 (Emulab assign), Mirkovic@IMC'12 (DETER assign+), Yao@CNS'13 (EasyScale), Yan@SOSR'15 (VT-Mininet).

Conclusions

- Proposed a framework for mapping a distributed task (or emulation experiment) onto a cluster of possibly heterogeneous machines
- Quantified packet processing capability
- Designed waterfall algorithm to map and partition a network experiment
- Evaluated our framework via simulations and DDoS experiments

Thank you!

Questions?