Flexplane: An Experimentation Platform for Resource Management in Datacenters

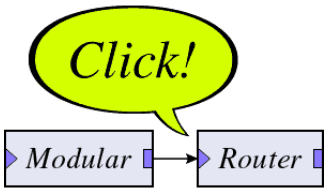
Amy Ousterhout, Jonathan Perry,
Hari Balakrishnan, Petr Lapukhov

Datacenter Networks

- Applications have diverse requirements
- Dozens of new resource management schemes
  - Low latency: DCTCP
  - Min FCT: PDQ, RCP, pFabric, PERC
  - Deadlines: D^3, D^2TCP
- Difficult to experiment with schemes in real networks
  - Requires changes to hardware routers

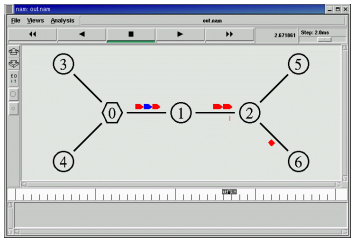
Experimentation with Resource Management

- Experimentation in real networks
  - Software routers - limited **throughput**
  - Programmable hardware - limited **flexibility**



Experimentation with Resource Management

- Experimentation in simulation (e.g., ns, OMNeT++)
  - Does not **accurately** model real network stacks, NICs, and distributed applications
  - Does not run in real time

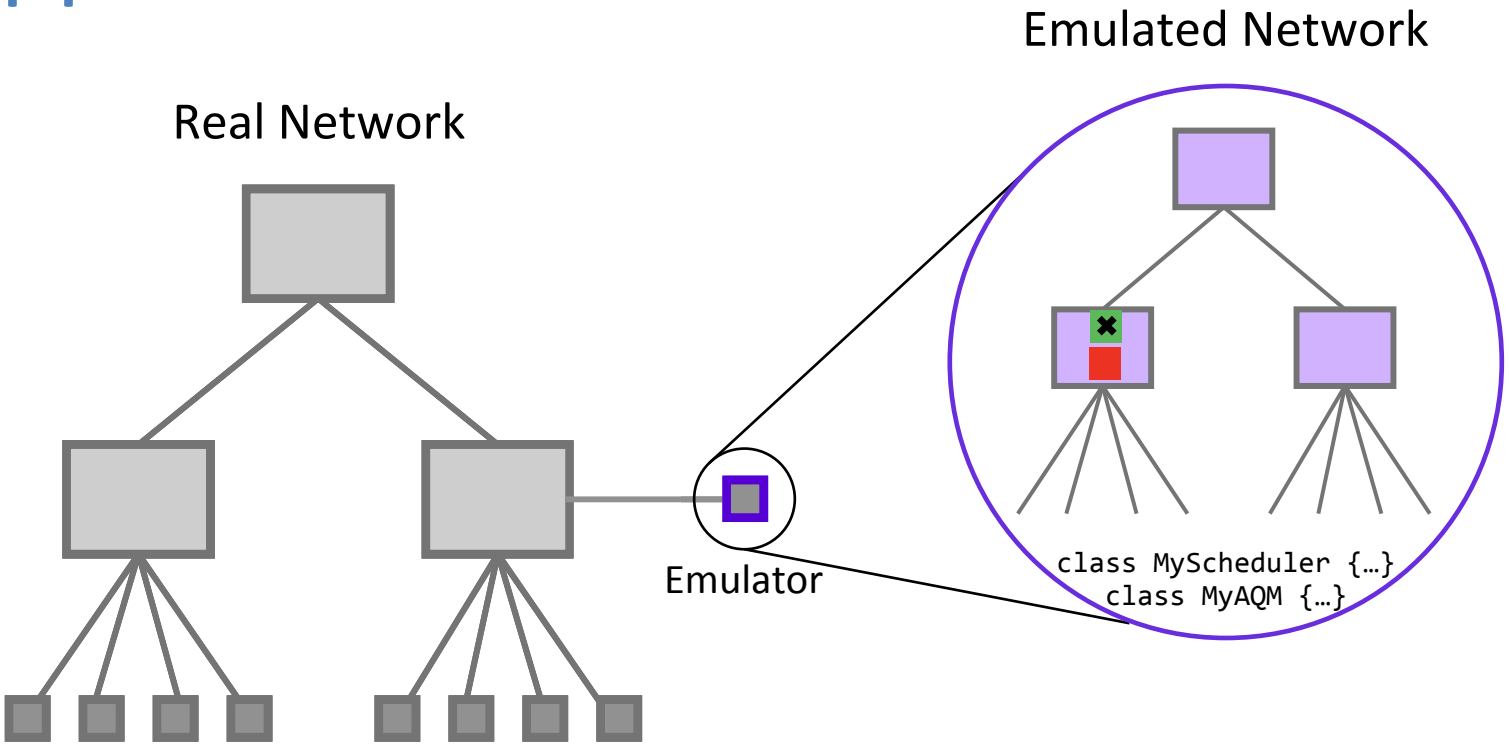


No existing approach to experimentation provides **accuracy**, **flexibility**, and **high throughput**

Our Contributions

- Key idea: whole-network emulation
- Flexplane: a platform for faithful experimentation with resource management schemes
  - Accurate – predicts behavior of hardware
  - Flexible – express schemes in C++
  - High throughput – 761 Gbits/s

Approach: Whole-Network Emulation

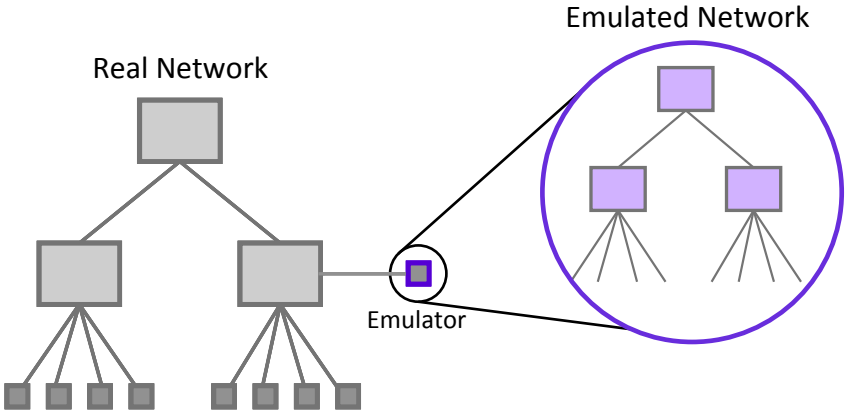


Abstract Packets

- Resource management schemes are *data-independent*
- Concise representation of one MTU
  - Source, destination, flow, ID
  - Custom per-scheme fields

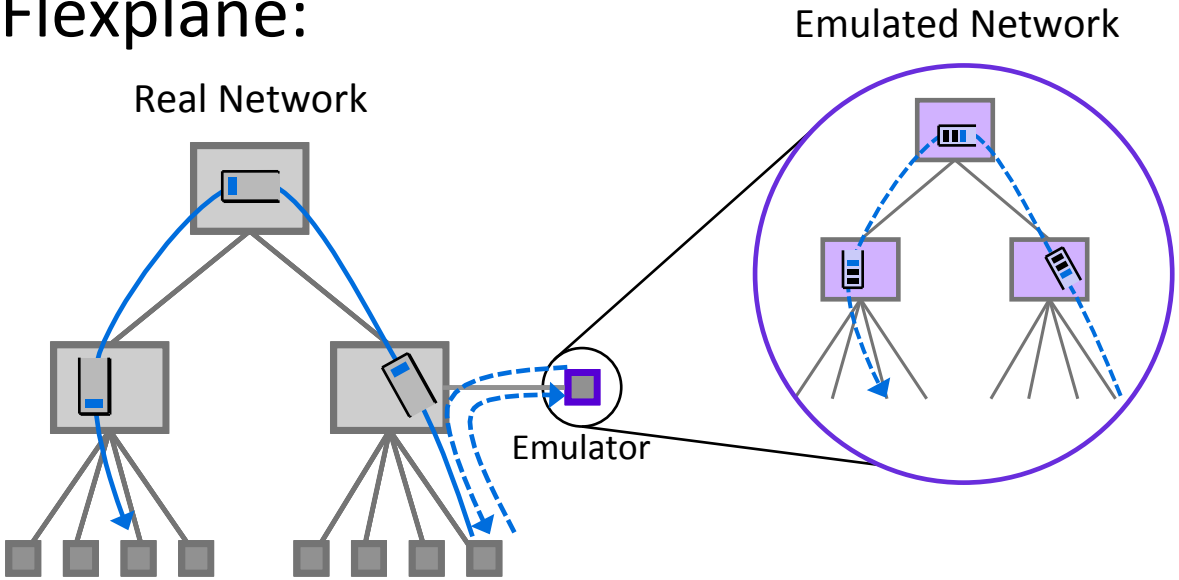
Emulator

- Real-time network simulator
- Faster than standard network simulators
  - Time divided into abstract-packet-sized timeslots
  - Omits endpoint software



Accuracy

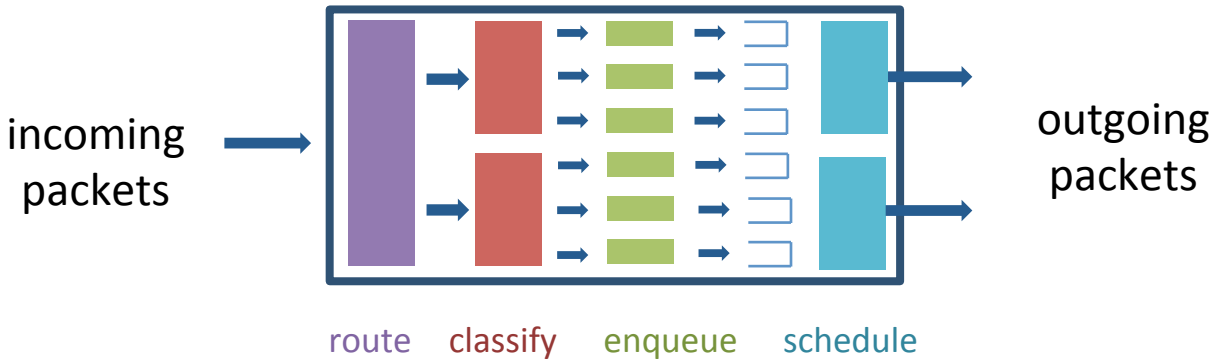
- Goal: predict behavior of a hardware network
- Hardware latency:
- Added latency of Flexplane:
  - RTT to emulator
  - Unloaded delay
  - Queuing delay in real network



Flexplane API

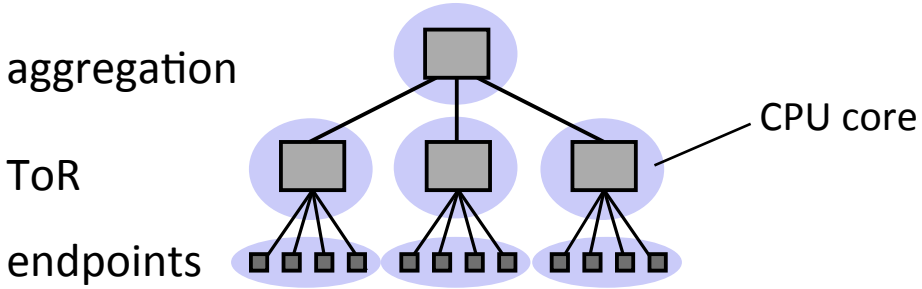
- Decouples schemes from framework

Emulator	int route(AbstractPkt *pkt) int classify(AbstractPkt *pkt, int port) enqueue(AbstractPkt *pkt, int port, int queue) AbstractPkt *schedule(int output_port)
Endpoints	prepare_request(sk_buff *skb, char *request_data) prepare_to_send(sk_buff *skb, char *allocation_data)



Multicore Emulator Architecture

- Pin network components (routers, endpoints) to cores
- Communication via FIFO queues
- Router state not shared across cores



Implementation

- Emulator uses Intel DPDK for low-latency NIC access
- Endpoints run a Linux qdisc

Evaluation

- Accuracy
- Utility
- Emulator throughput

Flexplane is Accurate

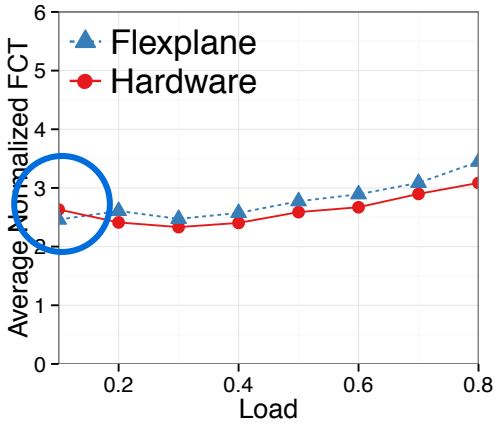
- Bulk TCP: 5 senders, 1 receiver
- Throughput 9.2-9.3 Gbits/s vs. 9.4 Gbits/s in hardware
- Similar queue occupancies

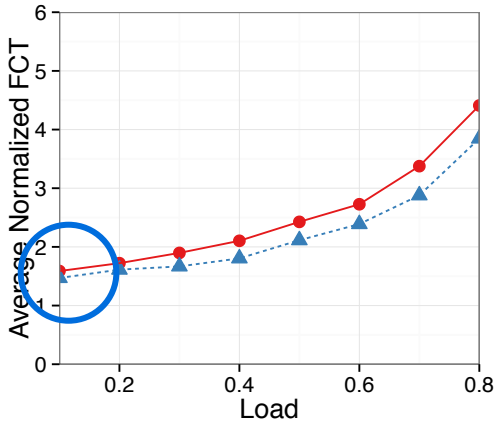
	Median Queue Occupancies (MTUs)	
	Hardware	Flexplane
DropTail	931	837
RED	138	104
DCTCP	61	51

Flexplane is Accurate

- RPC web search workload

DCTCP (0, 100KB]





DCTCP (10MB, ∞)

- Accurate to within 2-14% for loads up to 60%
- Observe behavior not visible in simulations

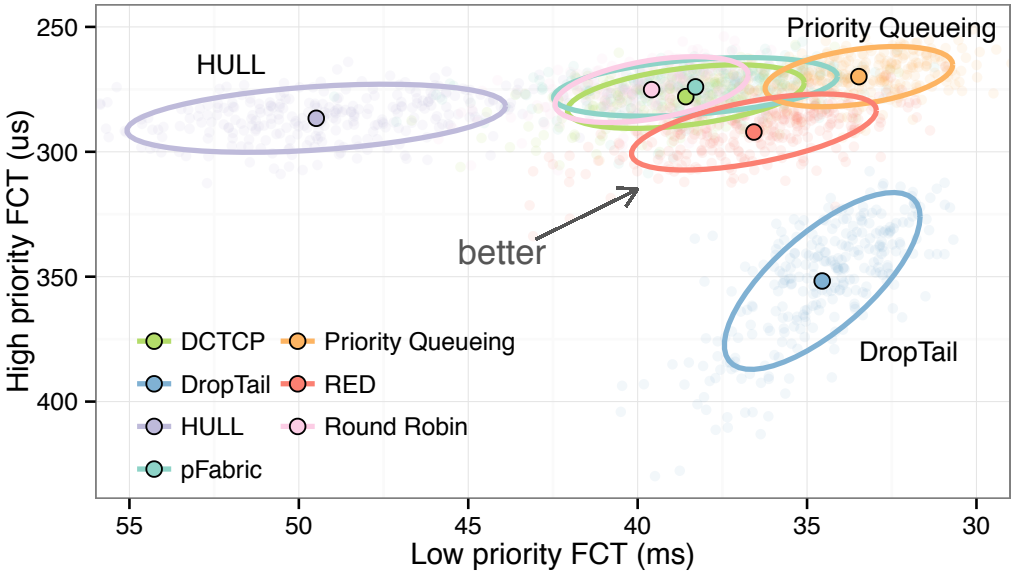
Flexplane is Easy to Use

- Implemented several schemes in dozens of lines of code

scheme	LOC
drop tail queue manager	39
RED queue manager	125
DCTCP queue manager	43
priority queueing scheduler	29
round robin scheduler	40
HULL scheduler	60
pFabric QM, queues, scheduler	251

Flexplane Enables Experimentation

- Evaluating trade-offs between resource management schemes



Flexplane Enables Experimentation

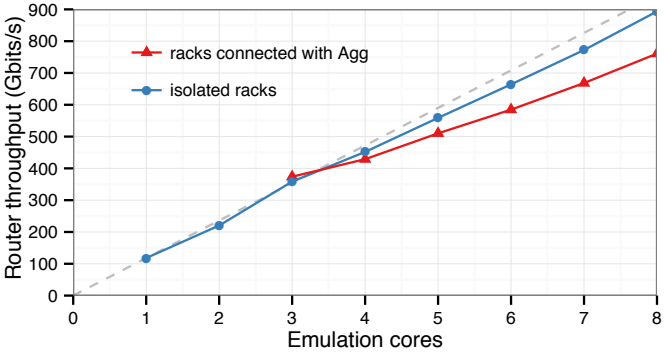
- Experiment with real distributed applications such as Spark

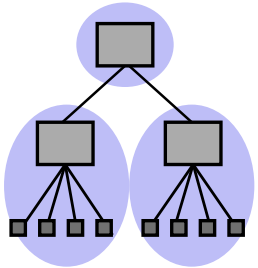
	% Change in Completion Time Relative to DropTail	
	Coordinate descent	Sort
DCTCP	+4.4%	-4.8%
HULL	+29.4%	-2.6%

- Performance depends on network and CPU

Emulator Throughput

- Emulator provides 761 Gbits/s of aggregate throughput with 10 total cores





- 81x as much throughput per clock cycle as RouteBricks

Flexplane: an Experimentation Platform

- Whole-network emulation
- Flexplane: a platform for faithful experimentation with resource management schemes
  - Accuracy, flexibility, and high throughput

https://github.com/aousterh/flexplane