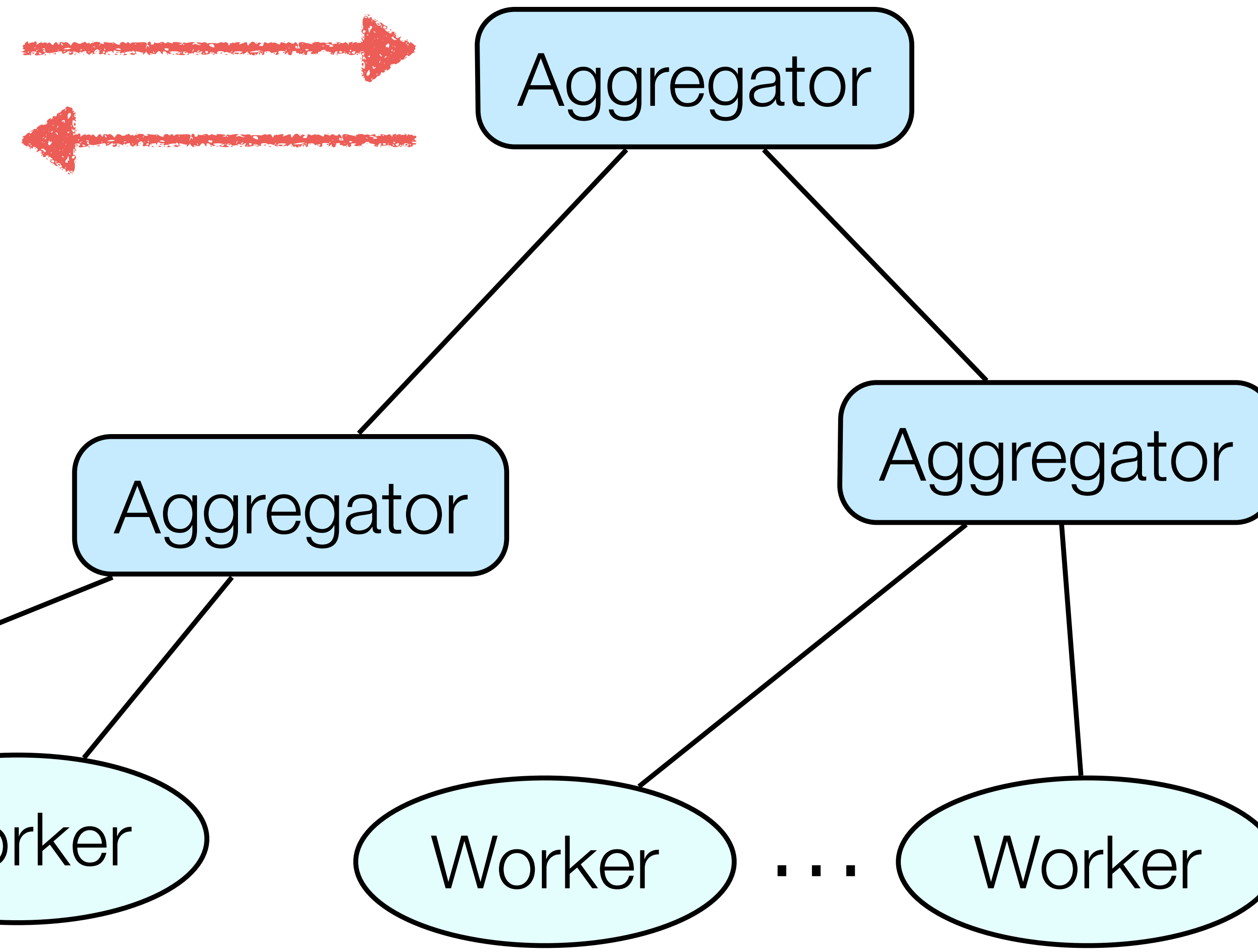
RepFlow: Minimizing Flow Completion Times with Replicated Flows in Data Centers

Hong Xu*, Baochun Li+

*City University of Hong Kong
+University of Toronto

IEEE INFOCOM, Toronto, May 1, 2014

Background

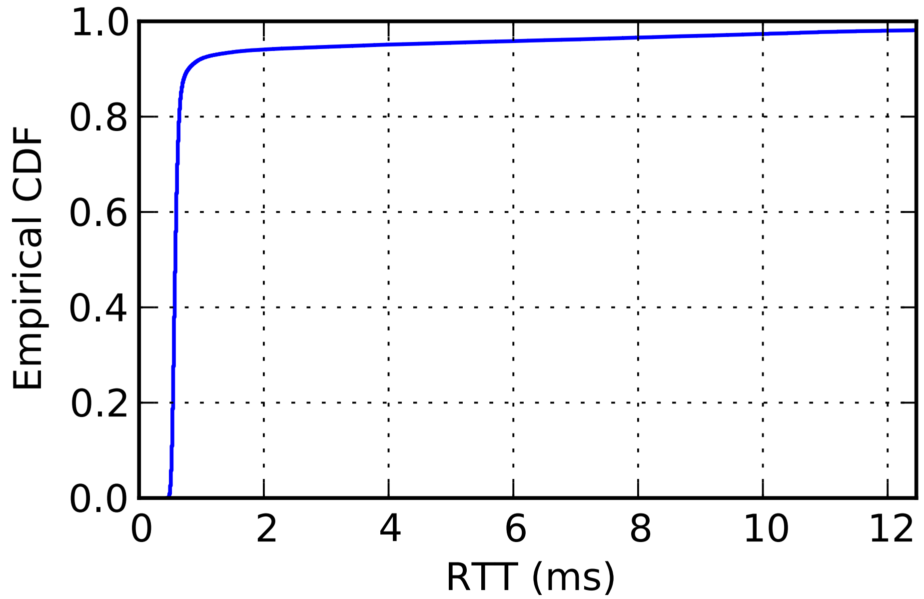


Low latency for a huge number of short query/response flows, especially the tail latency (e.g. 99-th)

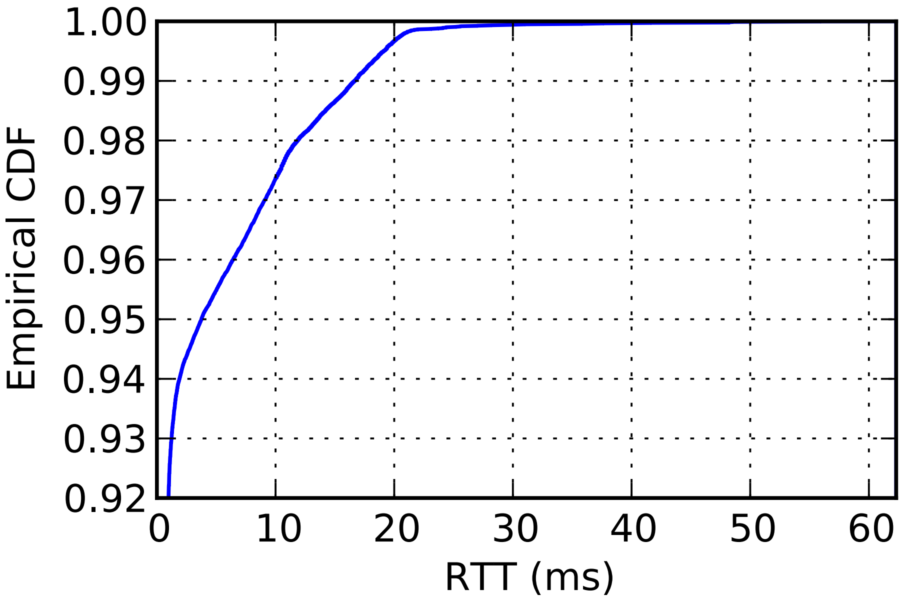
Background

- Current data center transport is ill-fitted for the task

RTT measurement in EC2 us-west-2c, 100K samples



Mean RTT: 0.5ms

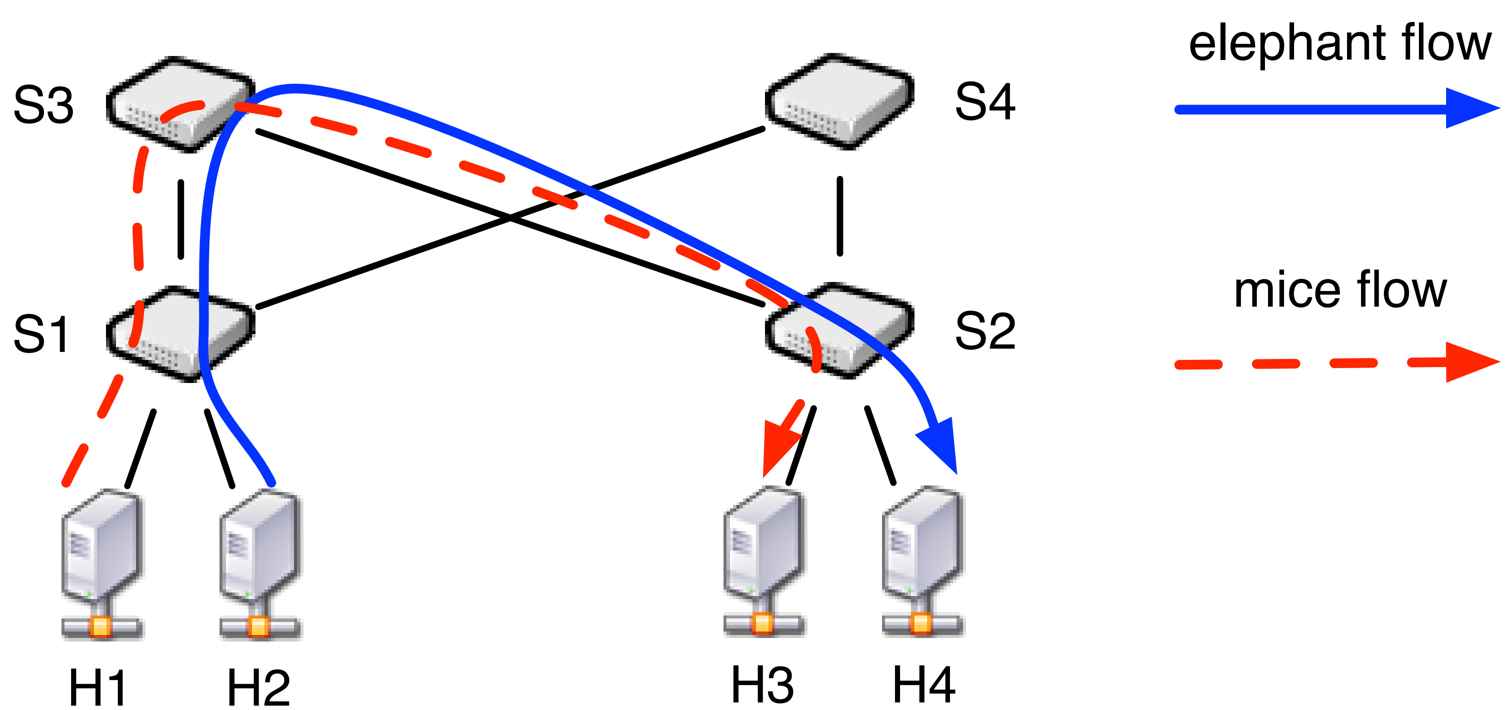


99-th RTT: 17ms

- Corroborated by measurements from many existing papers

Background

- Culprit: Head-of-line blocking in switch ports with ECMP, resulting in long queueing delay



- Tail latency is even worse with elephants colliding on the same path due to ECMP

Related work

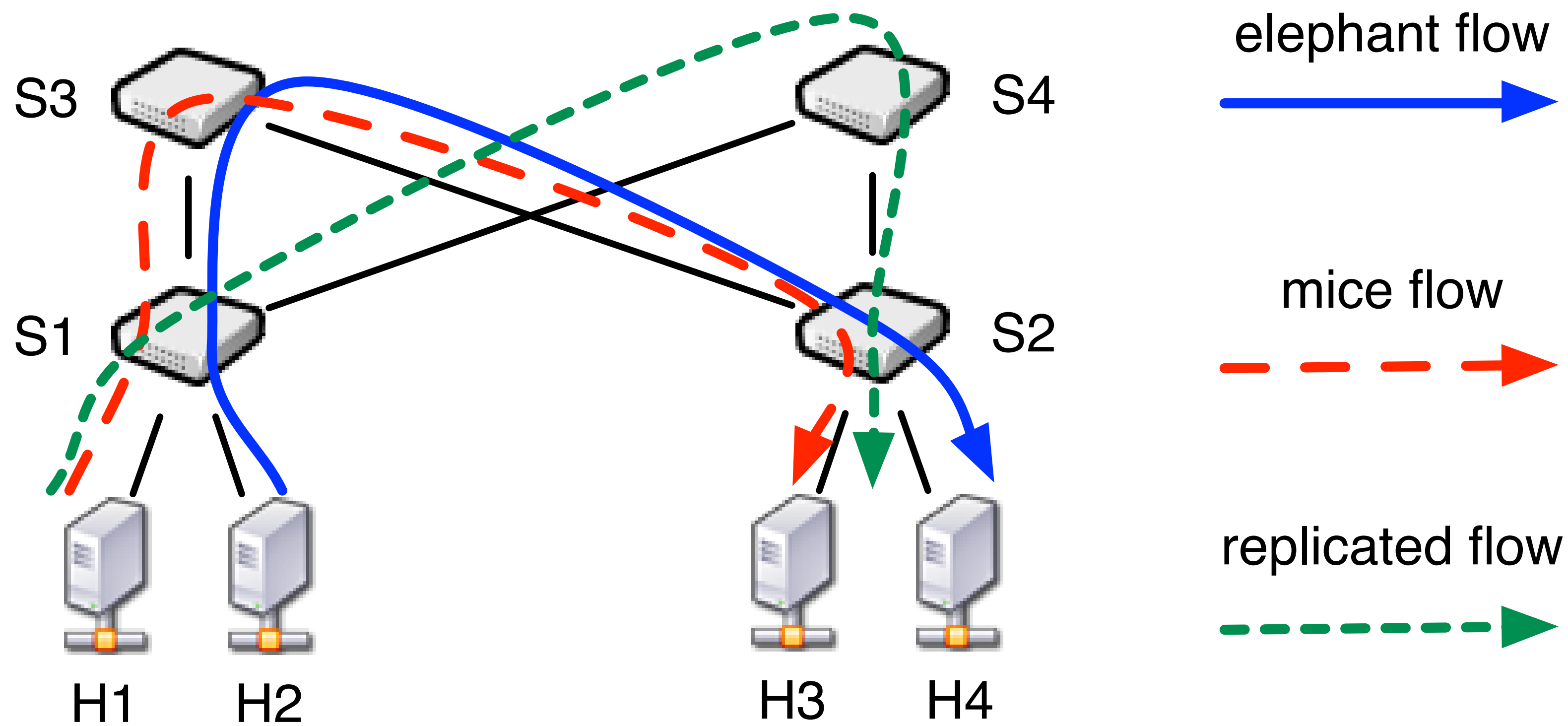
- Reducing (tail) latency in data center networks is an important problem
  - Reduce queue length: *DCTCP (2010), HULL (2012)*
  - Prioritize scheduling: *D^3 (2011), PDQ (2012), DeTail (2012), pFabric (2013)*
- They all require **modifications to end-hosts and/or switches**, making it difficult to deploy in reality

A practical and effective low latency transport

RepFlow

RepFlow in a nutshell

- Replicate each mice flow to exploit multipath diversity



- No two paths are exactly the same – The power of two choices, M Mitzenmacher

- Clos based topologies provide many equal-cost paths

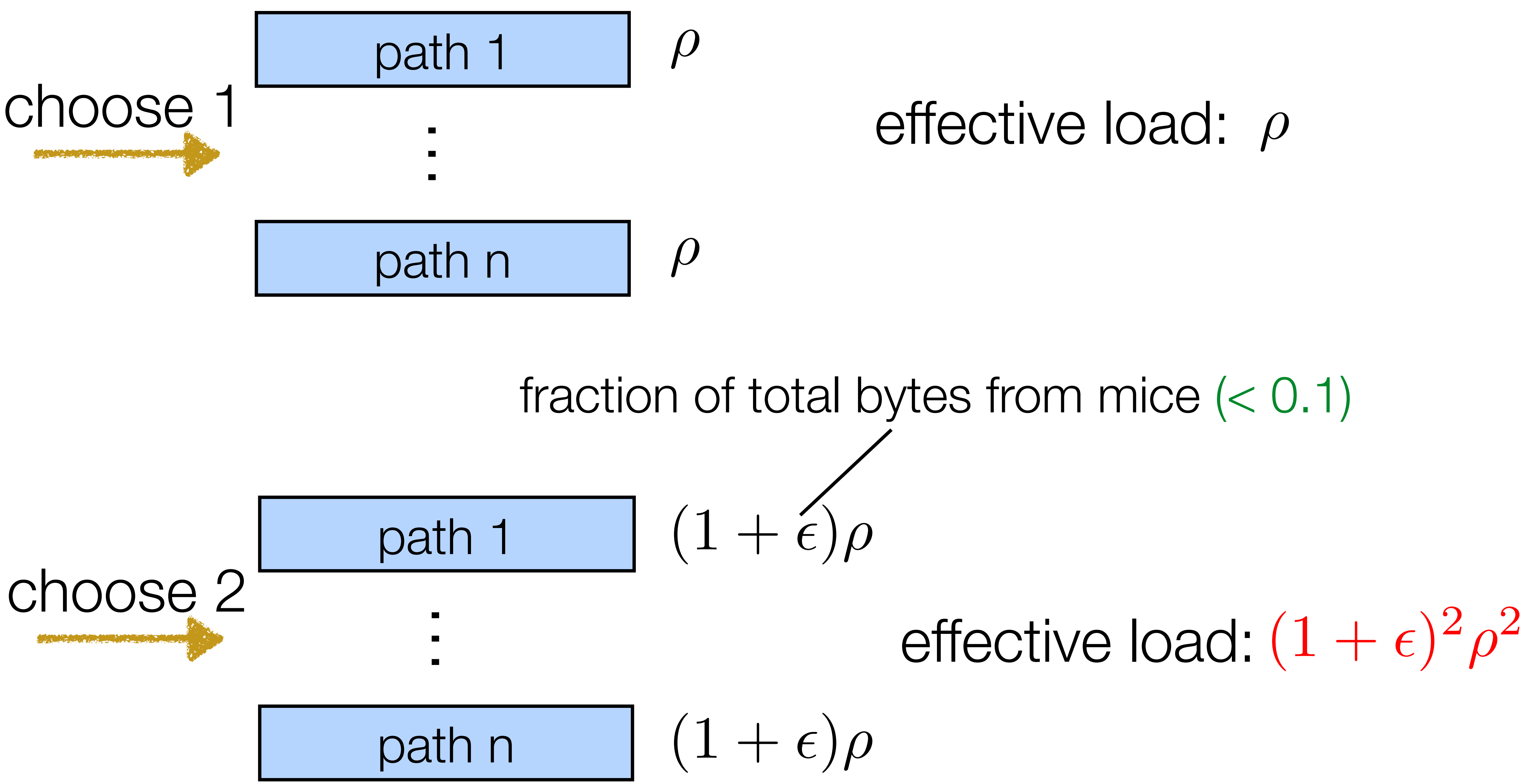
RepFlow’s design

- Which flows?
  - Less than 100KB, consistent with many existing papers
  - Replicate all flows at first, stop after 100KB
- When?
  - Always! (We’ll come back about the overhead issue)
- How?
  - Two TCP sockets, different src ports so they will be hashed to distinct paths by ECMP

Is RepFlow practical?

- Application layer implementation
  - No change to end-hosts or switches
  - Works with any TCP variant and ECMP
  - Circumvent the difficult path selection problem for mice flows
- Is RepFlow effective?

Simplified queueing analysis



Packet-level NS-3 simulations

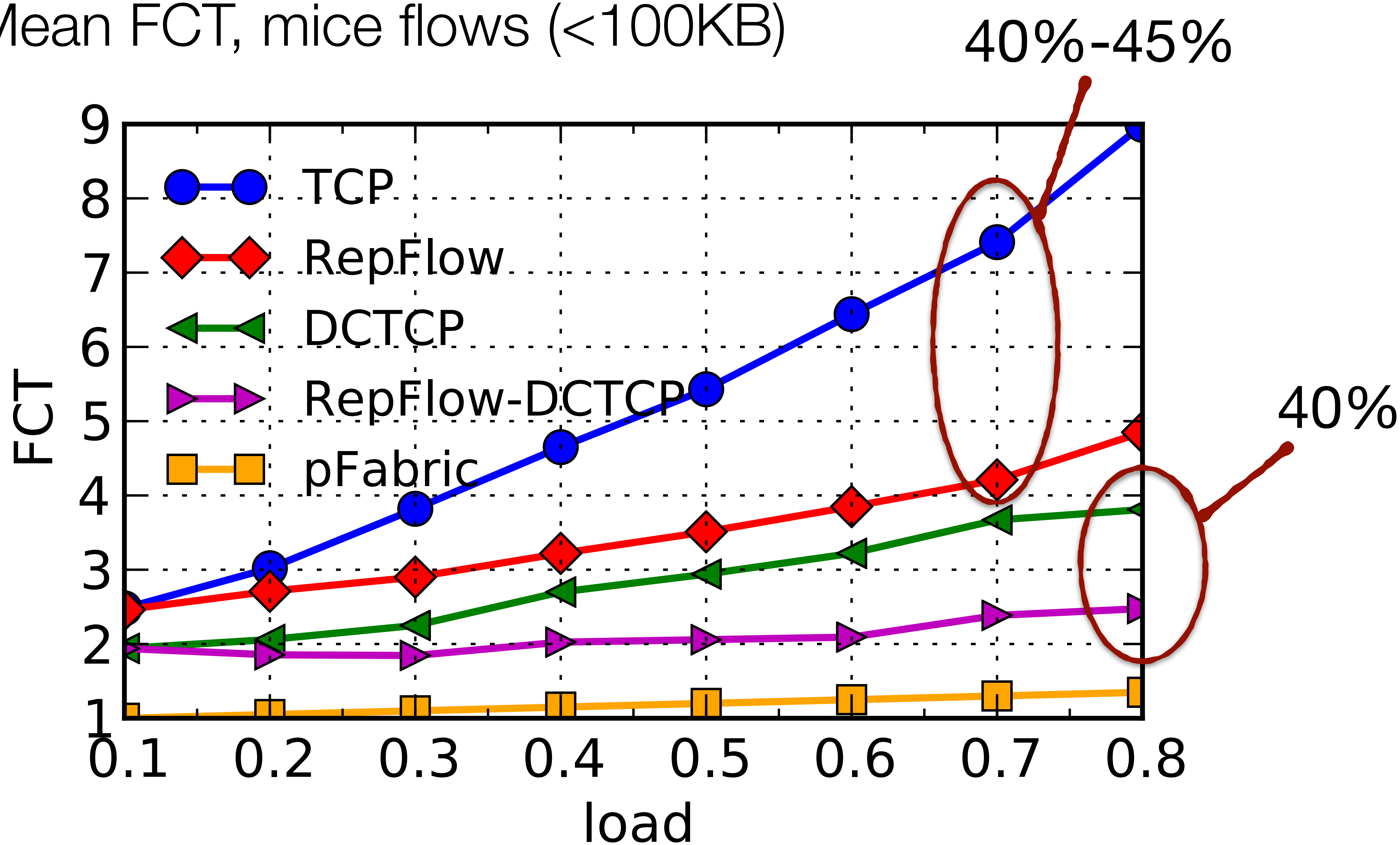
- Topology: 16-pod 1Gbps fat-tree, 1,024 hosts
- Traffic pattern: Poisson, random src/dst, 0.5s worth
- Flow size distribution:
    - Web search cluster from DCTCP paper
        - >95% bytes are from 30% flows large than 1MB
    - Data mining cluster from VL2 paper (not shown here)
        - >95% bytes are from 3.6% flows large than 35MB

Benchmarks

- TCP: TCP NewReno, initial window 12KB, DropTail queues with 100 packet buffer
- RepFlow
- DCTCP: source code from authors of D2TCP
- RepFlow-DCTCP: RepFlow on top of DCTCP
- pFabric: state-of-the-art, near-optimal FCT with priority queueing, source code obtained from authors

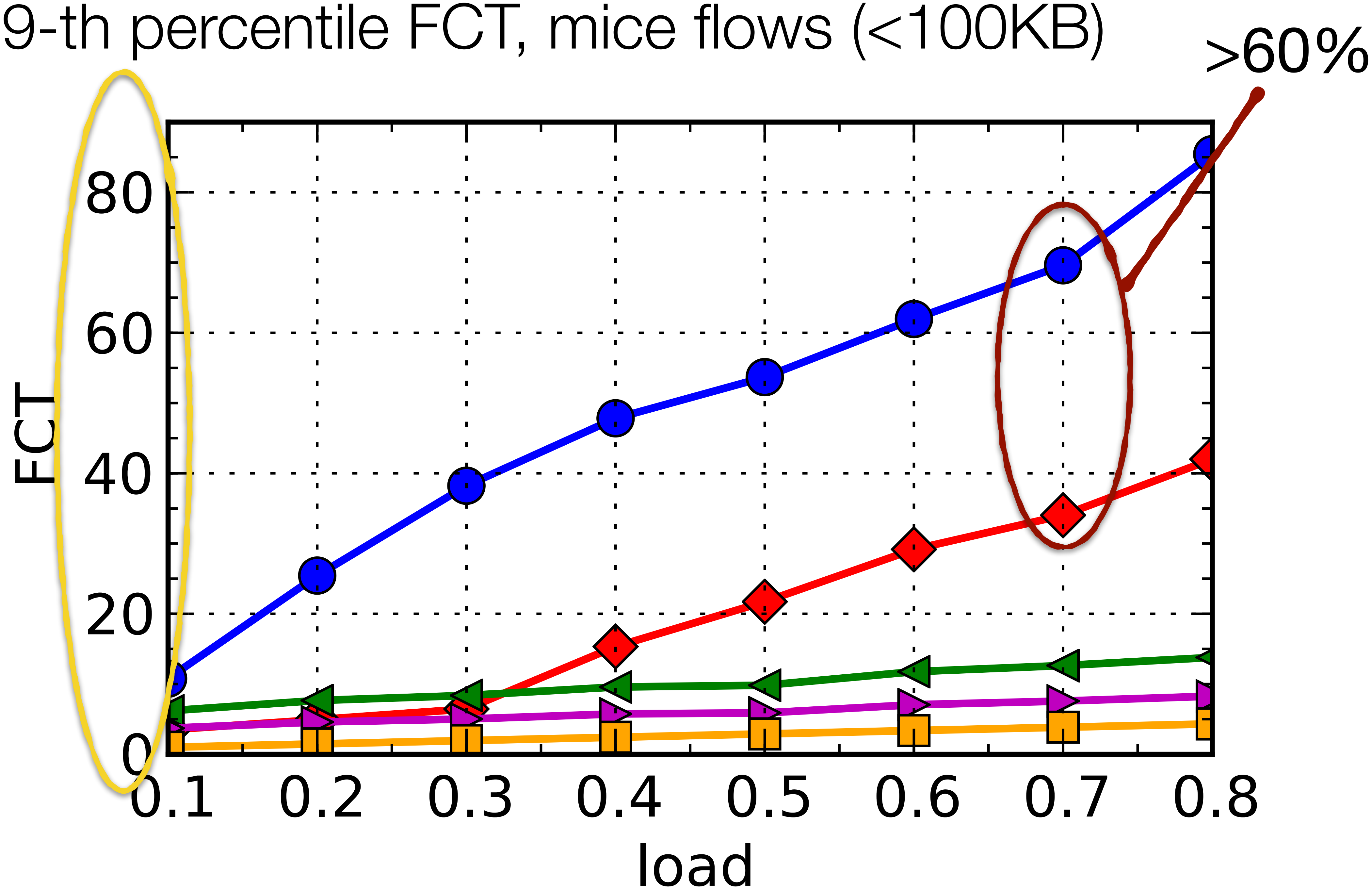
Results [1/4]

- Mean FCT, mice flows (<100KB)



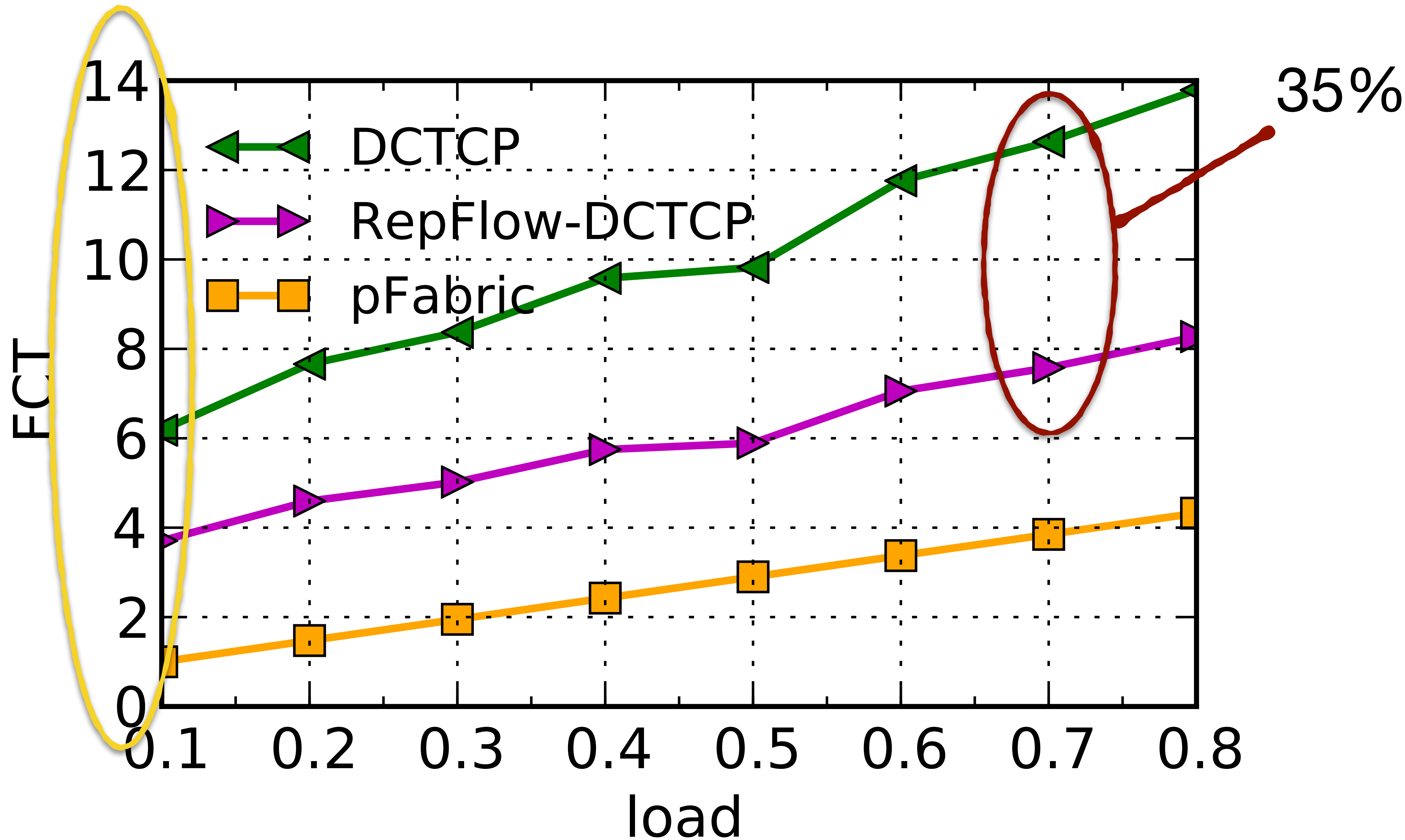
Results [2/4]

- 99-th percentile FCT, mice flows (<100KB)



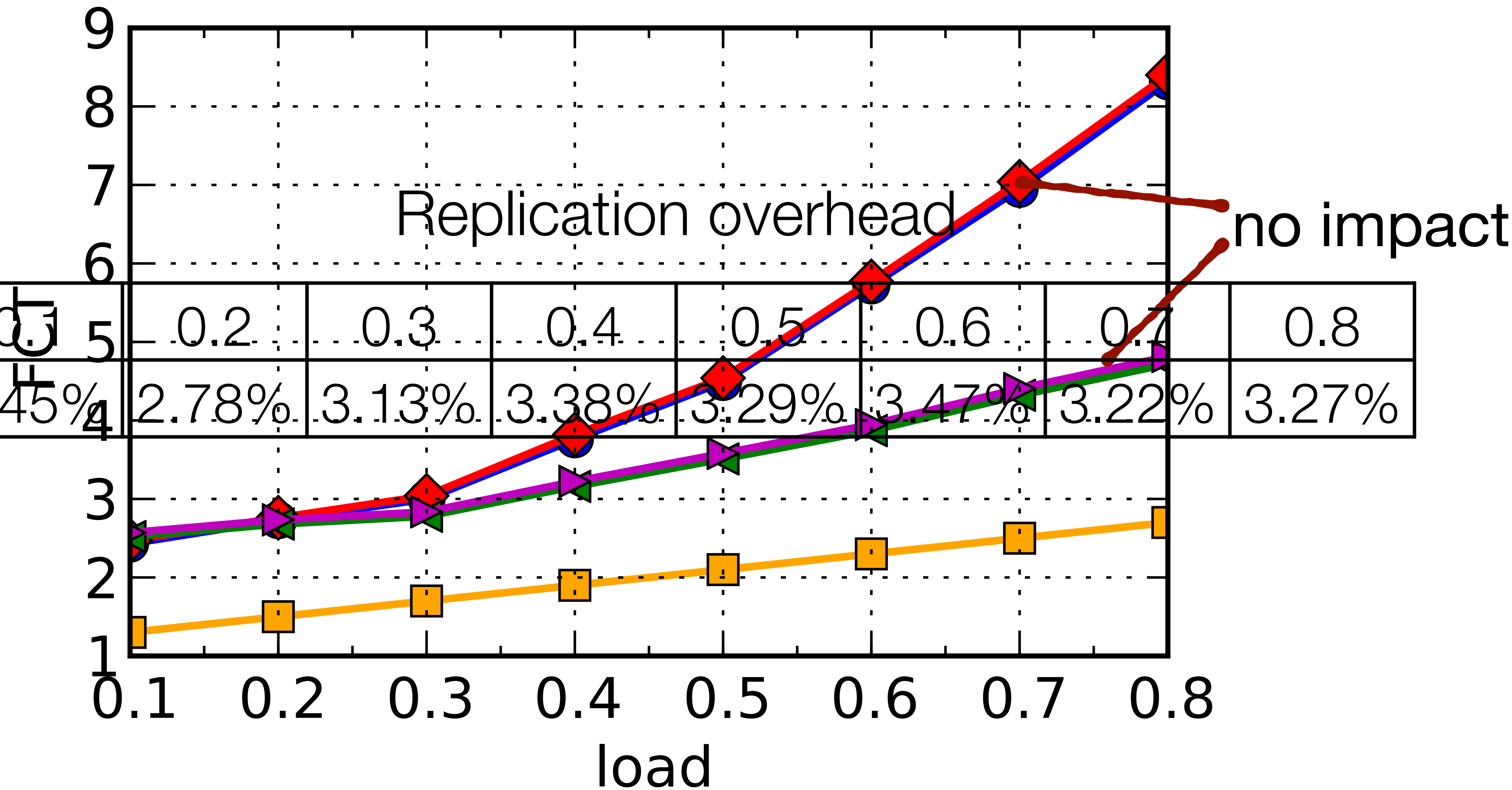
Results [3/4]

- 99-th percentile FCT, mice flows (<100KB), DCTCP



Results [4/4]

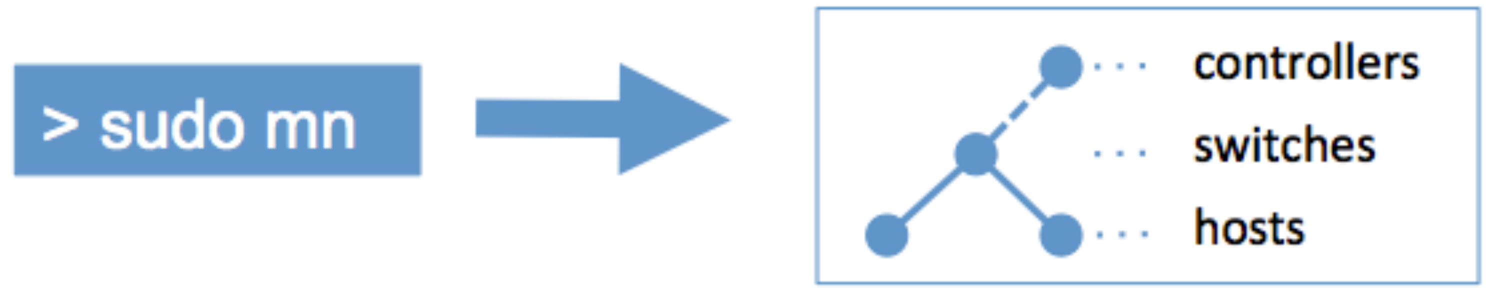
- Mean FCT, elephant flows (>=100KB)



Replication overhead

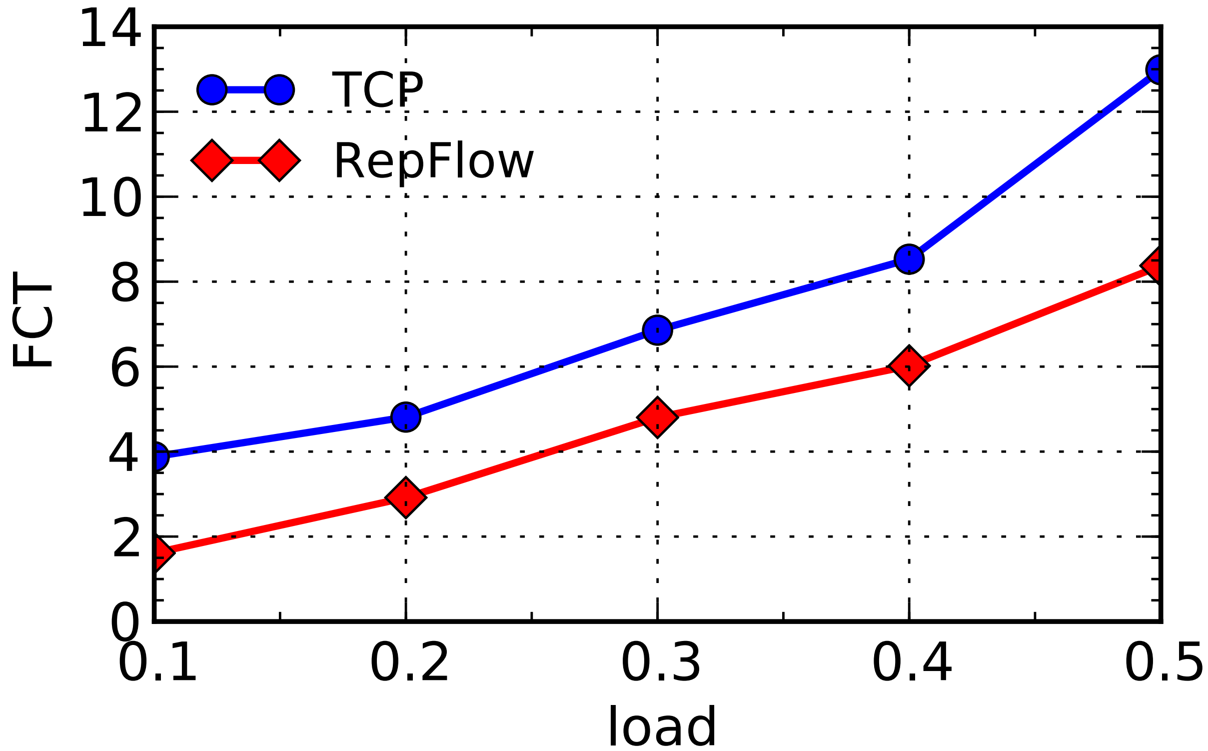
0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
3.45%	2.78%	3.13%	3.38%	3.29%	3.47%	3.22%	3.27%

Mininet experiments

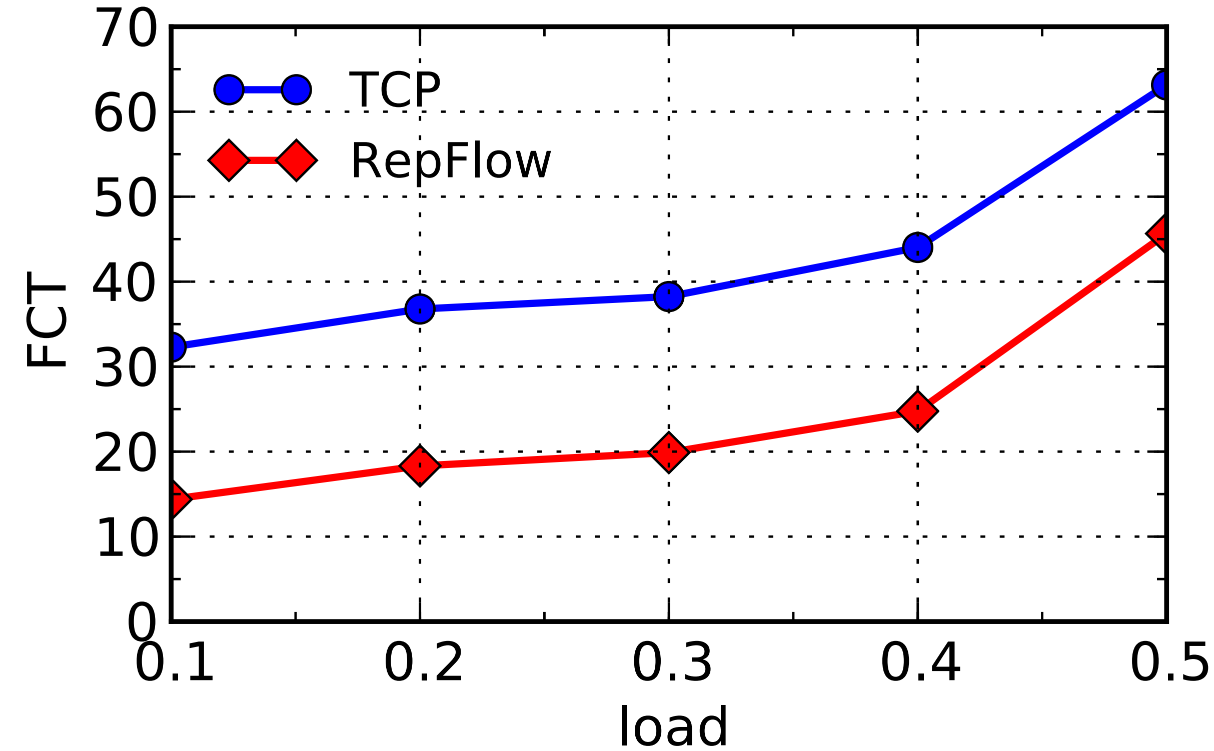


- A high-fidelity emulator based on Linux container based virtualization, running real kernel, switch and application code on a single machine
- Each virtual host runs two threads, one for regular flows and the other for replicated ones
- 4-pod fat-tree, 20Mbps link 1ms delay, buffer size 50 packets

Mininet results



Mean FCT



99-th percentile FCT

Results are in line with simulation results, with 25%-50% improvements.

Almost done…

- Takeaway: *RepFlow is a practical and effective low latency data center transport using replication*
- Redundancy for latency is generally applicable
  - Vulimiri et al., Low latency via redundancy, CoNEXT 2013
- Future work
  - Implementation of RepFlow as a general application library
  - Queueing models for FCT in data center networks

Thank you!

Henry Xu
City University of Hong Kong