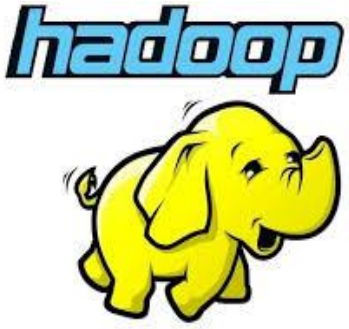
Enabling ECN in Multi-Service Multi-Queue Data Centers

Wei Bai, Li Chen, Kai Chen, Haitao Wu (Microsoft)

SING Group @ Hong Kong University of Science and Technology

Background

- Data Centers
  - Many services with diverse network requirements



Background

- Data Centers
  - Many services with diverse network requirements
- ECN-based Transports

ECN = Explicit Congestion Notification

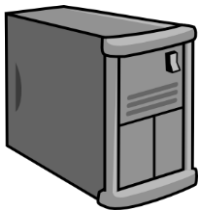
Background

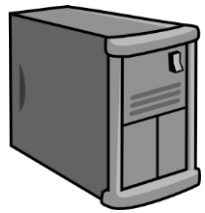
- Data Centers
  - Many services with diverse network requirements
- ECN-based Transports
  - Achieve high throughput & low latency
  - Widely deployed: DCTCP, DCQCN, etc.

ECN-based Transports

ECN-based Transports

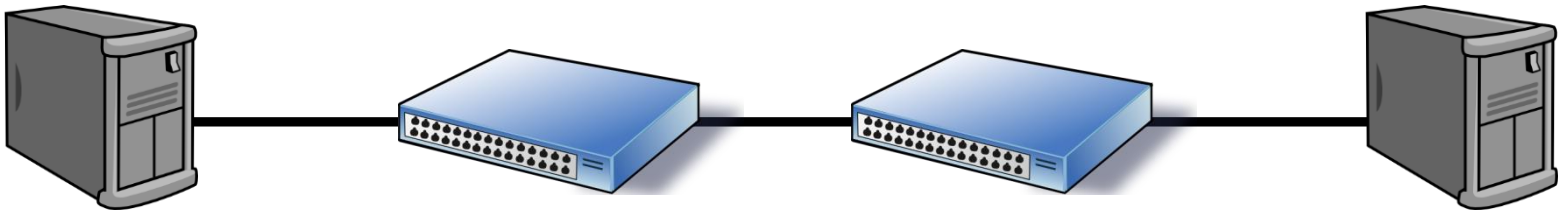
- ECN-enabled end-hosts
  - React to ECN by adjusting sending rates





ECN-based Transports

- ECN-enabled end-hosts
  - React to ECN by adjusting sending rates
- ECN-aware switches
  - Perform ECN marking based on Active Queue Management (AQM) policies



ECN-based Transports

- ECN-enabled end-hosts
  - React to ECN by adjusting sending rates
- ECN-aware switches
  - Perform ECN marking based on Active Queue Management (AQM) policies

ECN-aware Switches

- Adopt RED to perform ECN marking

RED = Random Early Detection

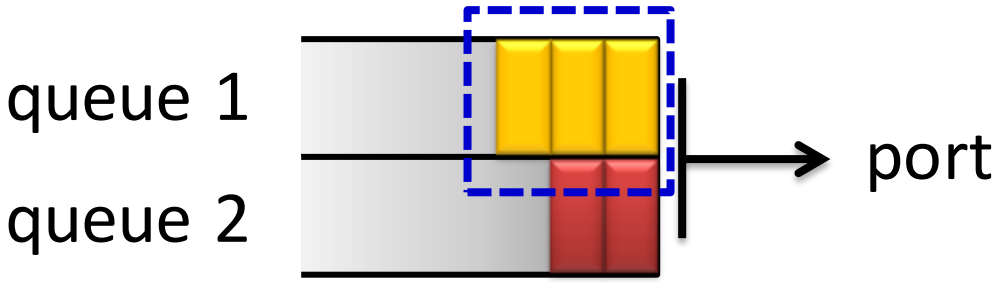
ECN-aware Switches

- Adopt RED to perform ECN marking
  - Per-queue/port/service-pool ECN/RED

Track buffer occupancy of different egress entities

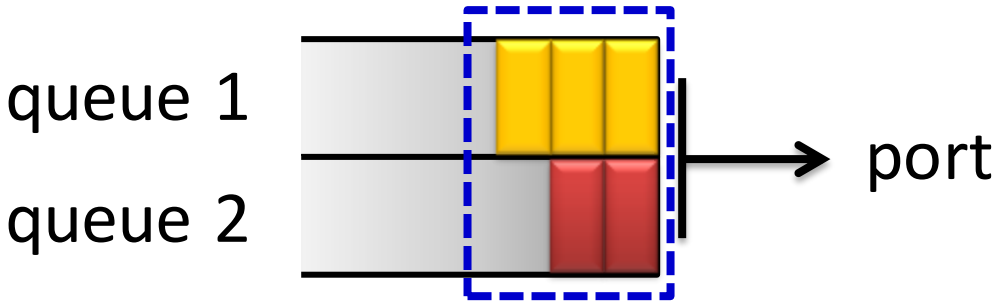
ECN-aware Switches

- Adopt RED to perform ECN marking
  - Per-**queue**/port/service-pool ECN/RED



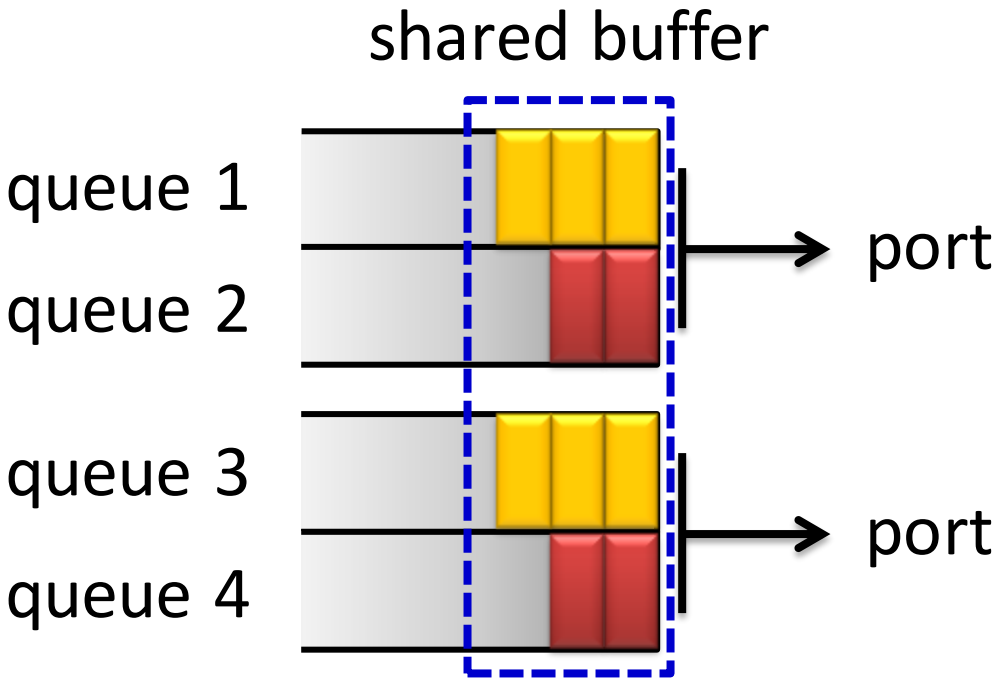
ECN-aware Switches

- Adopt RED to perform ECN marking
  - Per-queue/**port**/service-pool ECN/RED



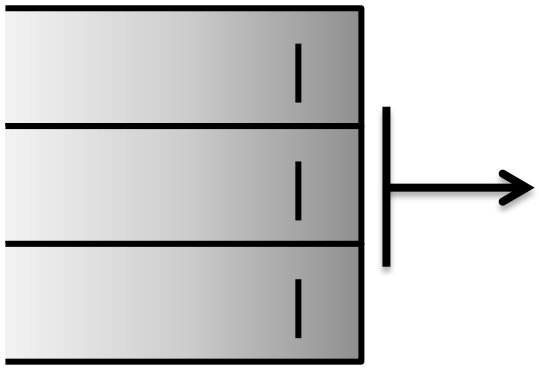
ECN-aware Switches

- Adopt RED to perform ECN marking
  - Per-queue/port/**service-pool** ECN/RED



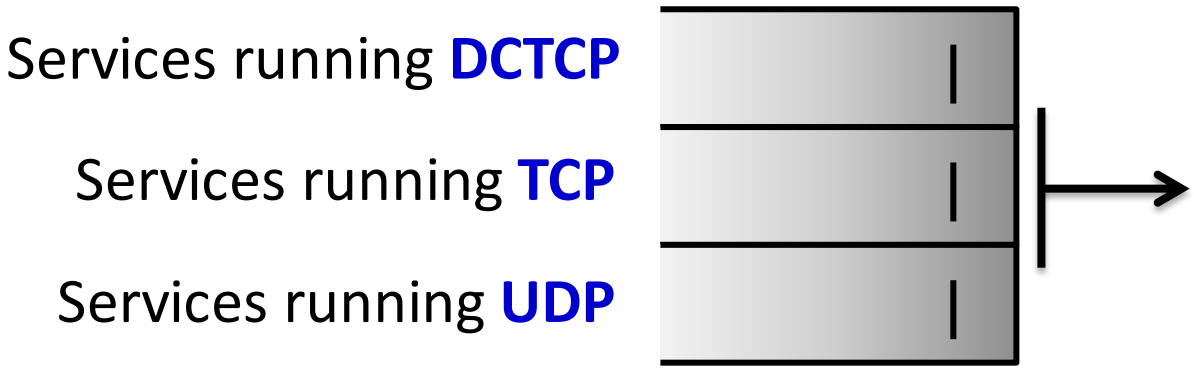
ECN-aware Switches

- Adopt RED to perform ECN marking
  - Per-queue/port/service-pool ECN/RED
- Leverage multiple queues to classify traffic
  - Isolate traffic from different services/applications



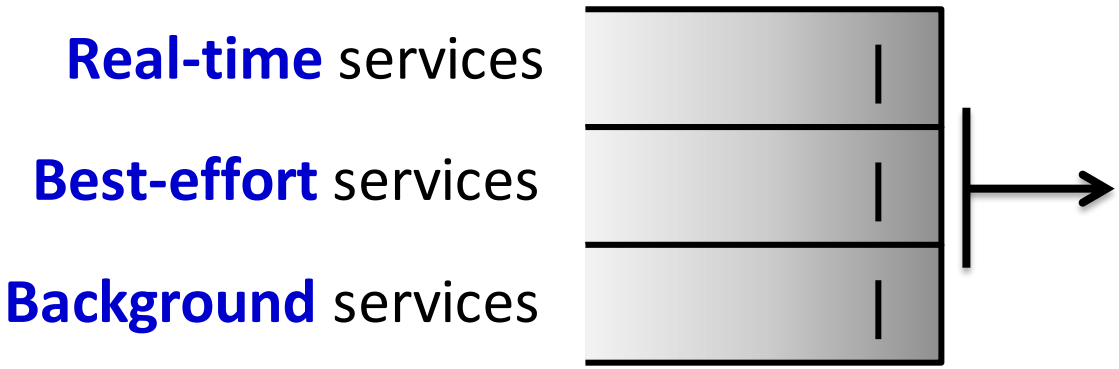
ECN-aware Switches

- Adopt RED to perform ECN marking
  - Per-queue/port/service-pool ECN/RED
- Leverage multiple queues to classify traffic
  - Isolate traffic from different services/applications



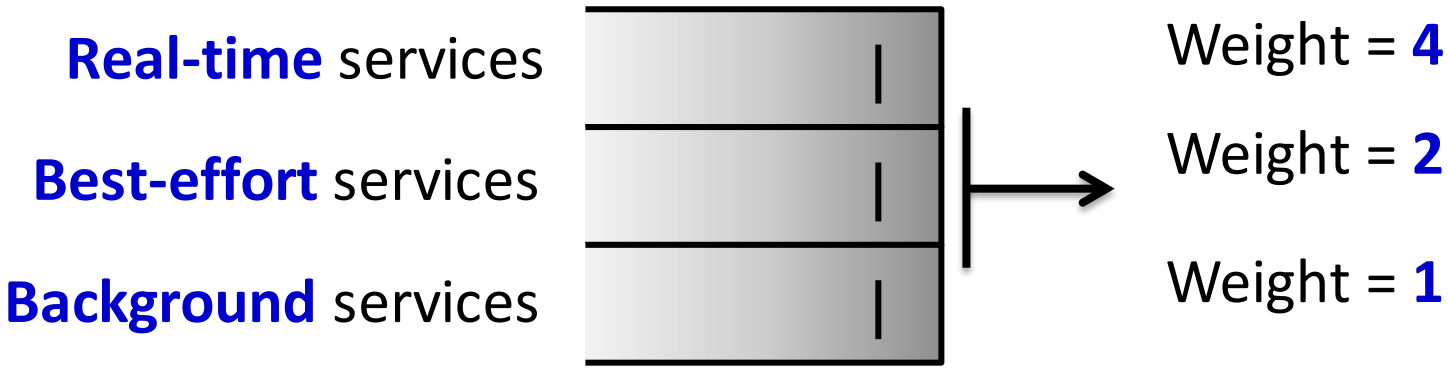
ECN-aware Switches

- Adopt RED to perform ECN marking
  - Per-queue/port/service-pool ECN/RED
- Leverage multiple queues to classify traffic
  - Isolate traffic from different services/applications



ECN-aware Switches

- Adopt RED to perform ECN marking
  - Per-queue/port/service-pool ECN/RED
- Leverage multiple queues to classify traffic
  - Isolate traffic from different services/applications
  - Weighted max-min fair sharing among queues



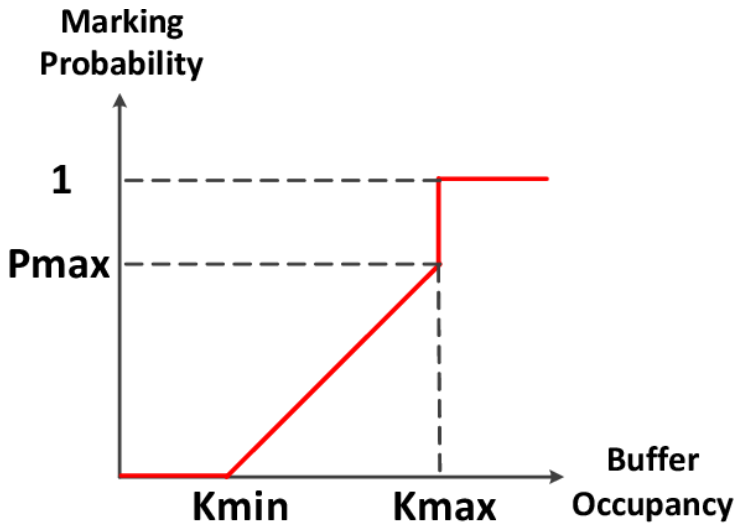
ECN-aware Switches

- Adopt RED to perform ECN marking
  - Per-queue/port/service-pool ECN/RED
- Leverage multiple queues to classify traffic
  - Isolate traffic from different services/applications
  - Weighted max-min fair sharing among queues

Perform ECN marking in multi-queue context

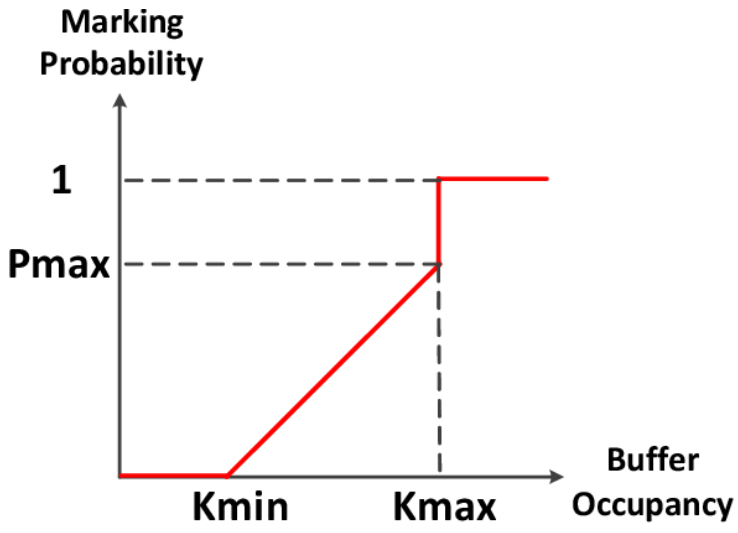
ECN marking with Single Queue

ECN marking with Single Queue

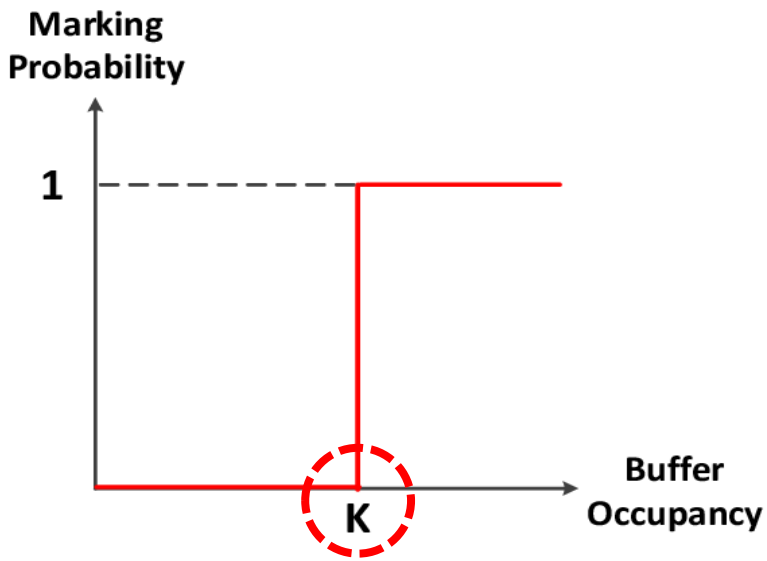


RED Algorithm

ECN marking with Single Queue



RED Algorithm



Practical Configuration

(e.g., DCTCP)

ECN marking with Single Queue

- To achieve 100% throughput

$$K \geq C \times RTT \times \lambda$$

ECN marking with Single Queue

- To achieve 100% throughput

$$K \geq C \times RTT \times \lambda$$

Determined by congestion control algorithms

ECN marking with Single Queue

- To achieve 100% throughput

$$K \geq C \times RTT \times \lambda$$

Standard ECN marking threshold

ECN marking with Single Queue

- To achieve 100% throughput

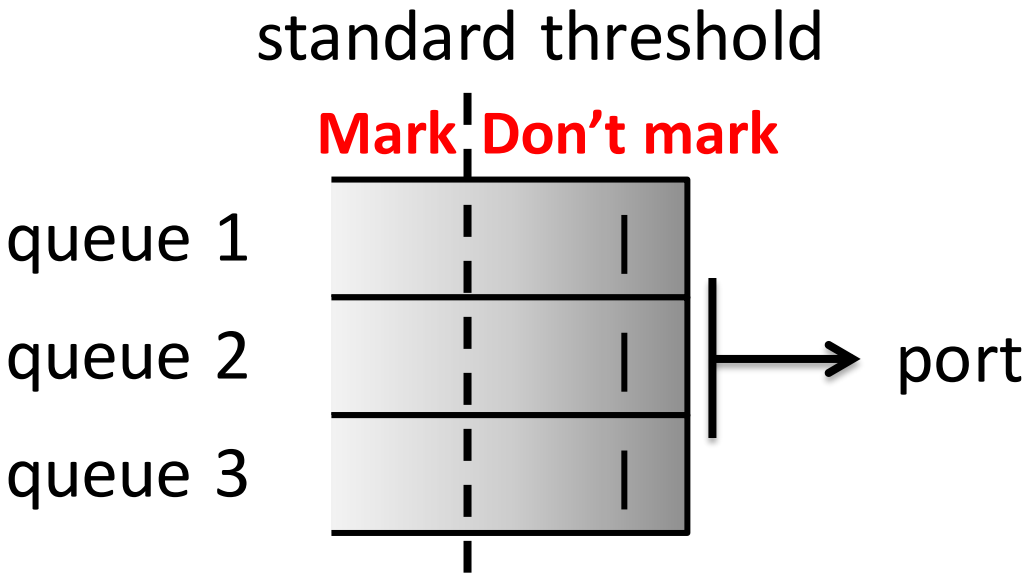
$$K \geq C \times RTT \times \lambda$$

The standard threshold is relatively stable in DCN, e.g., 65 packets for 10G network (DCTCP paper)

ECN marking with Multi-Queue (1)

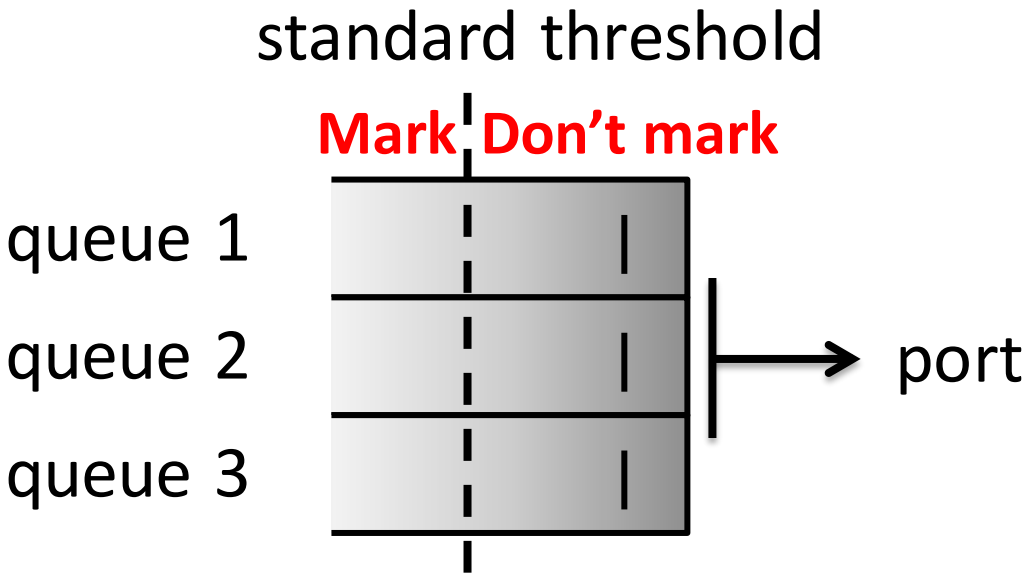
ECN marking with Multi-Queue (1)

- Per-queue with the standard threshold
  - $K_{queue(i)} = C \times RTT \times \lambda$



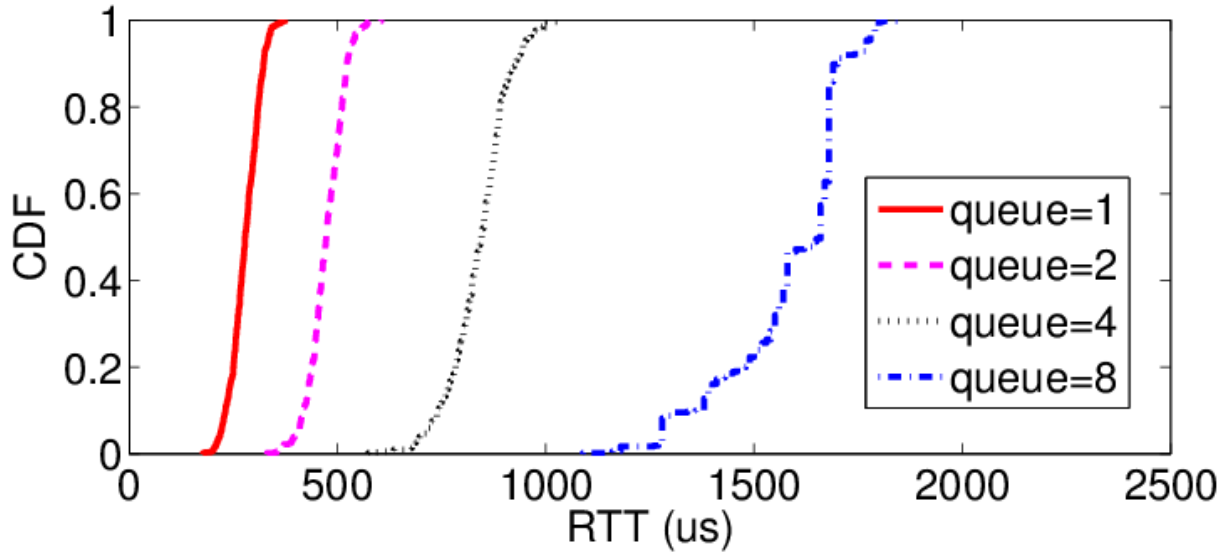
ECN marking with Multi-Queue (1)

- Per-queue with the standard threshold
  - $K_{queue(i)} = C \times RTT \times \lambda$
  - **Increase packet latency**



ECN marking with Multi-Queue (1)

- Per-queue with the standard threshold
  - $K_{queue(i)} = C \times RTT \times \lambda$
  - **Increase packet latency**

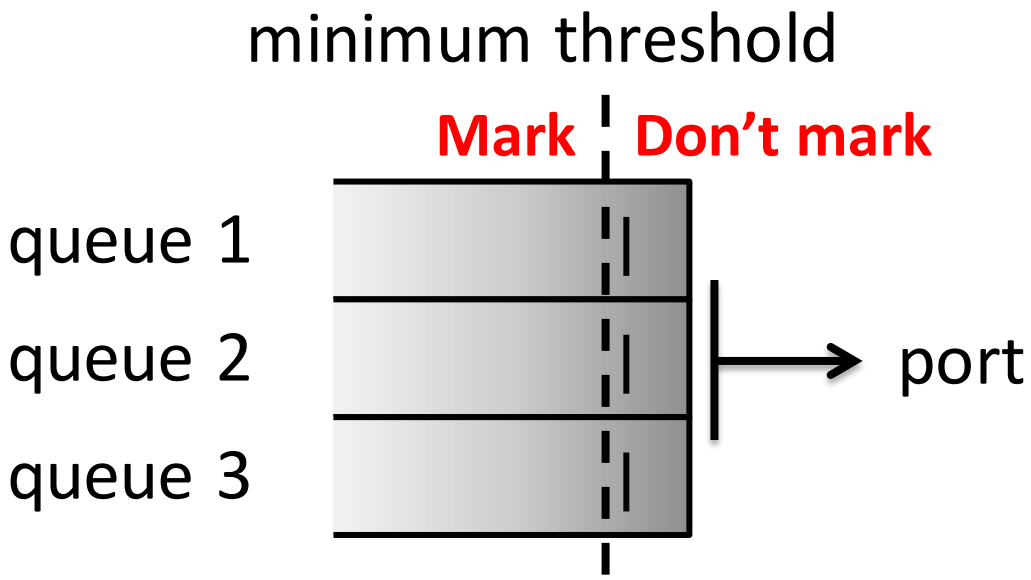


Evenly classify 8 long-lived flows into a varying number of queues

ECN marking with Multi-Queue (2)

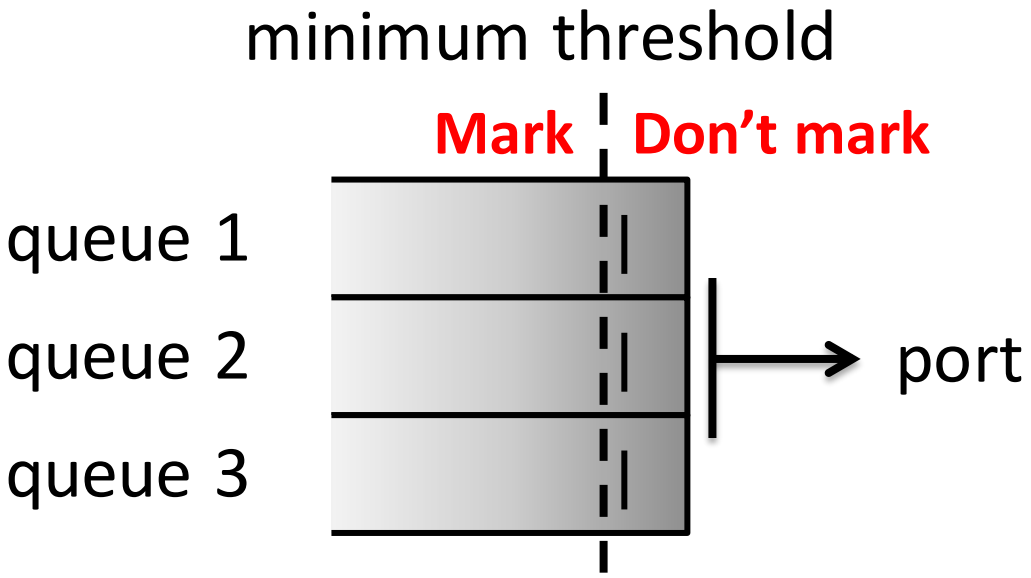
- Per-queue with the minimum threshold
  - $K_{queue(i)} = C \times RTT \times \lambda \times w_i / \sum w_j$

Normalized weight



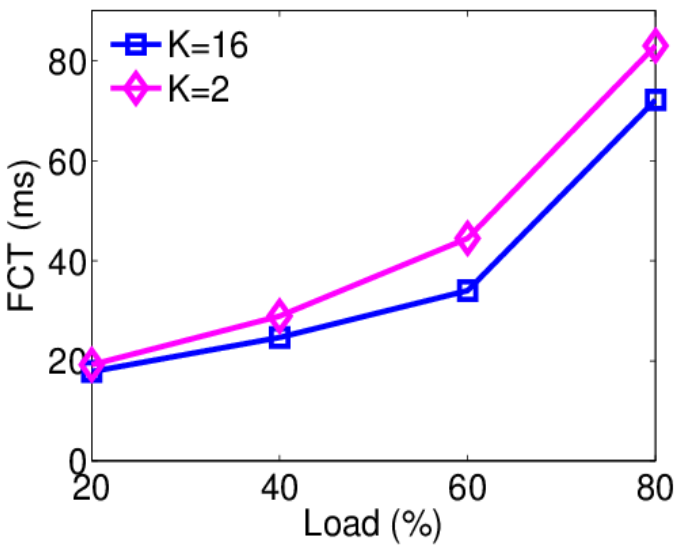
ECN marking with Multi-Queue (2)

- Per-queue with the minimum threshold
  - $K_{queue(i)} = C \times RTT \times \lambda \times w_i / \sum w_j$
  - **Degrade throughput**

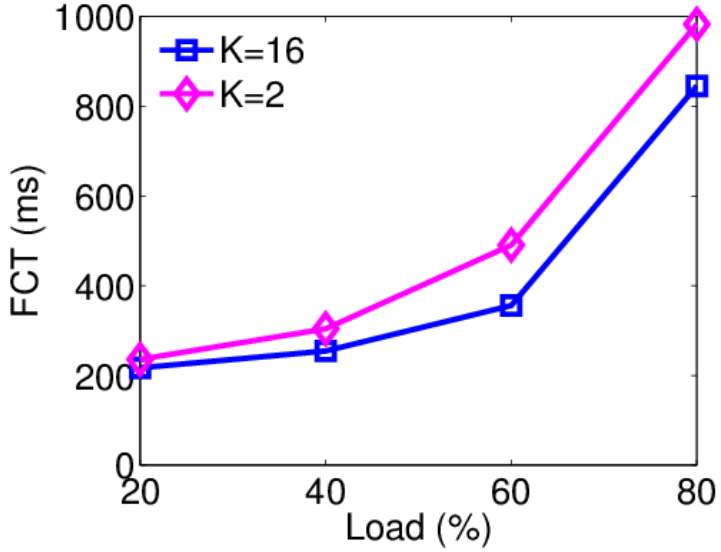


ECN marking with Multi-Queue (2)

- Per-queue with the minimum threshold
  - $K_{queue(i)} = C \times RTT \times \lambda \times w_i / \sum w_j$
  - **Degrade throughput**



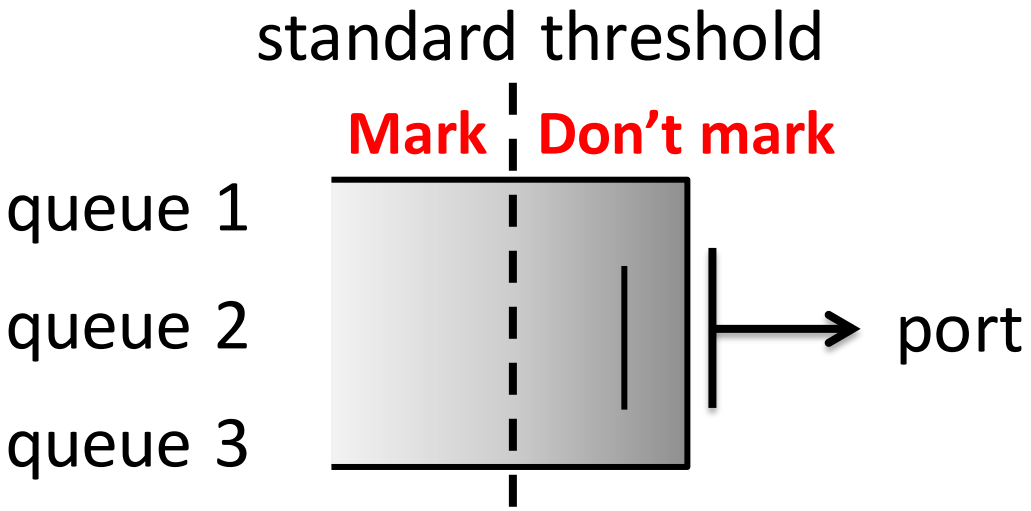
Overall Average FCT



Average FCT (>10MB)

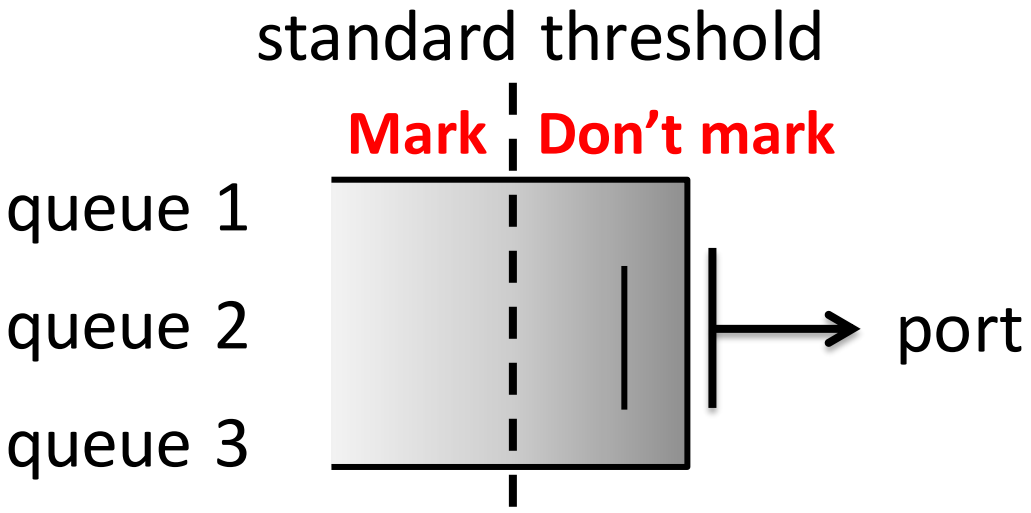
ECN marking with Multi-Queue (3)

- Per-port
  - $K_{port} = C \times RTT \times \lambda$



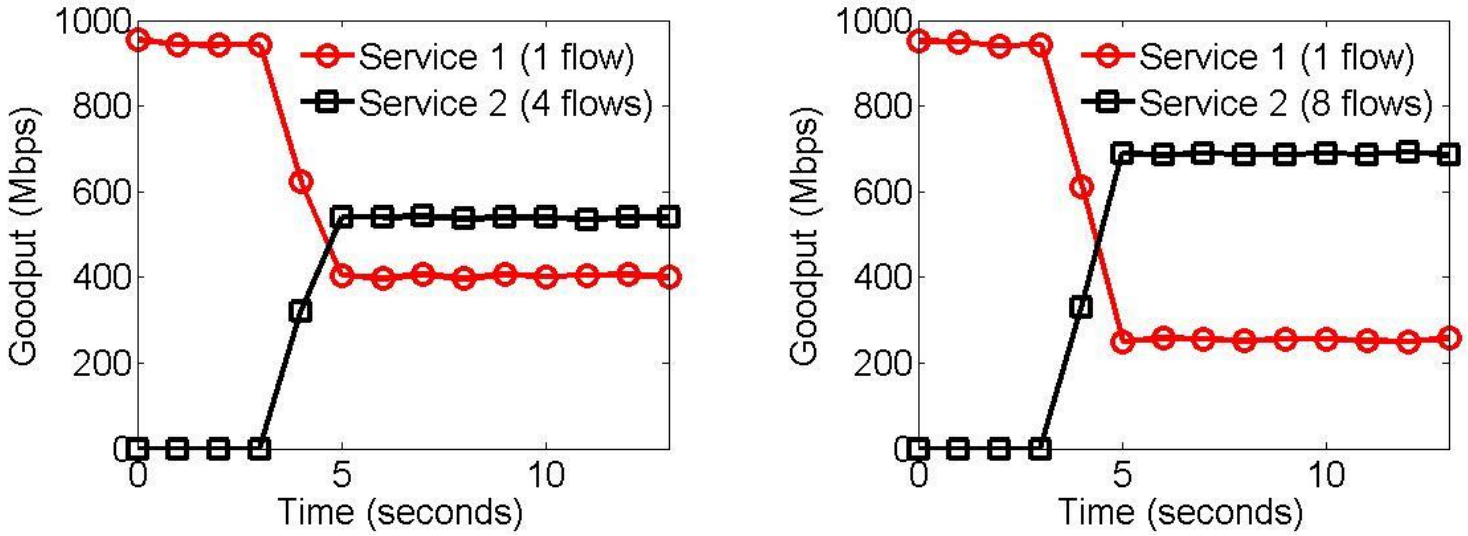
ECN marking with Multi-Queue (3)

- Per-port
  - $K_{port} = C \times RTT \times \lambda$
  - **Violate weighted fair sharing**



ECN marking with Multi-Queue (3)

- Per-port
  - $K_{port} = C \times RTT \times \lambda$
  - **Violate weighted fair sharing**



Both services have a equal-weight dedicated queue on the switch

Question

- Can we design an ECN marking scheme with following properties:
  - Deliver low latency
  - Achieve high throughput
  - Preserve weighted fair sharing
  - Compatible with legacy ECN/RED implementation

Question

- Can we design an ECN marking scheme with following properties:
  - Deliver low latency
  - Achieve high throughput
  - Preserve weighted fair sharing
  - Compatible with legacy ECN/RED implementation

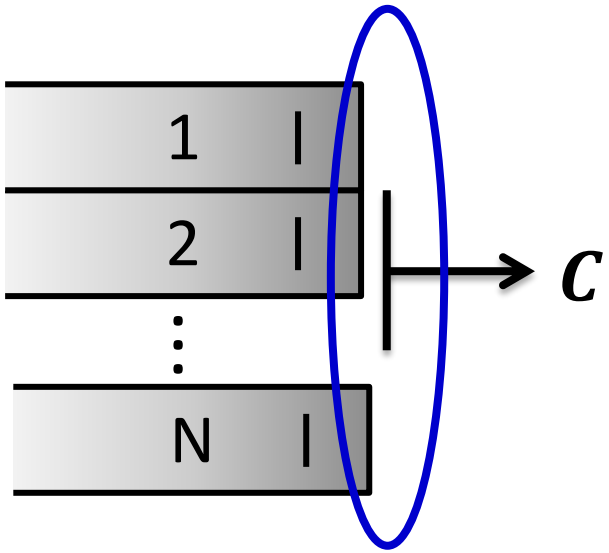
Our answer: MQ-ECN

MQ-ECN’S DESIGN

Start from GPS Model

- N queues share the link with capacity C

Generalized Processor Sharing (GPS)



Start from GPS Model

- N queues share the link with capacity C

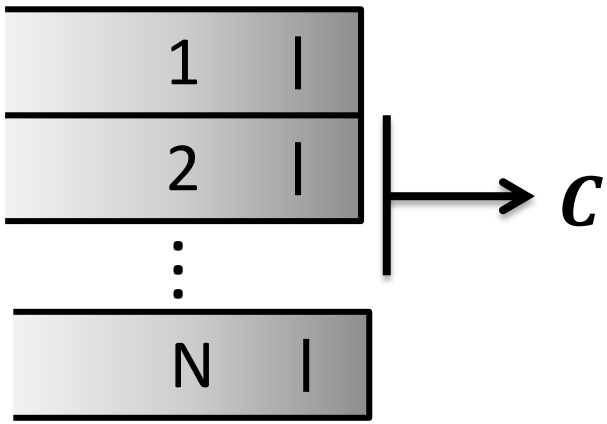
Input Rate

r_1

r_2

$\vdots$

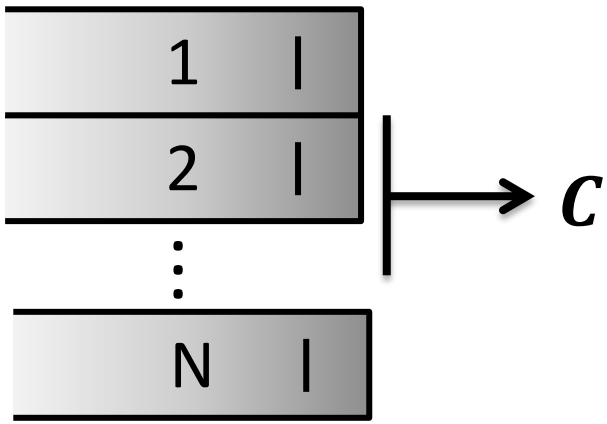
r_N



Start from GPS Model

- N queues share the link with capacity C

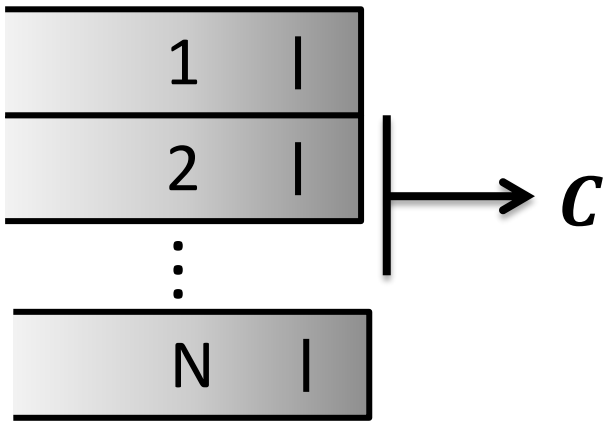
Input Rate	Weight
r_1	w_1
r_2	w_2
$\vdots$	$\vdots$
r_N	w_N



Start from GPS Model

- N queues share the link with capacity C
- $C = \sum_{i=1}^{N} \min(r_i, w_i\alpha)$ Weighted Fair Share Rate

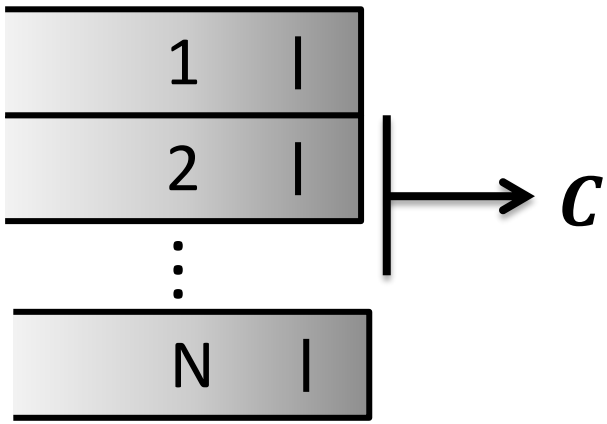
Input Rate	Weight	Output Rate
r_1	w_1	$\min(r_1, w_1\alpha)$
r_2	w_2	$\min(r_2, w_2\alpha)$
$\vdots$	$\vdots$	$\vdots$
r_N	w_N	$\min(r_N, w_N\alpha)$



Start from GPS Model

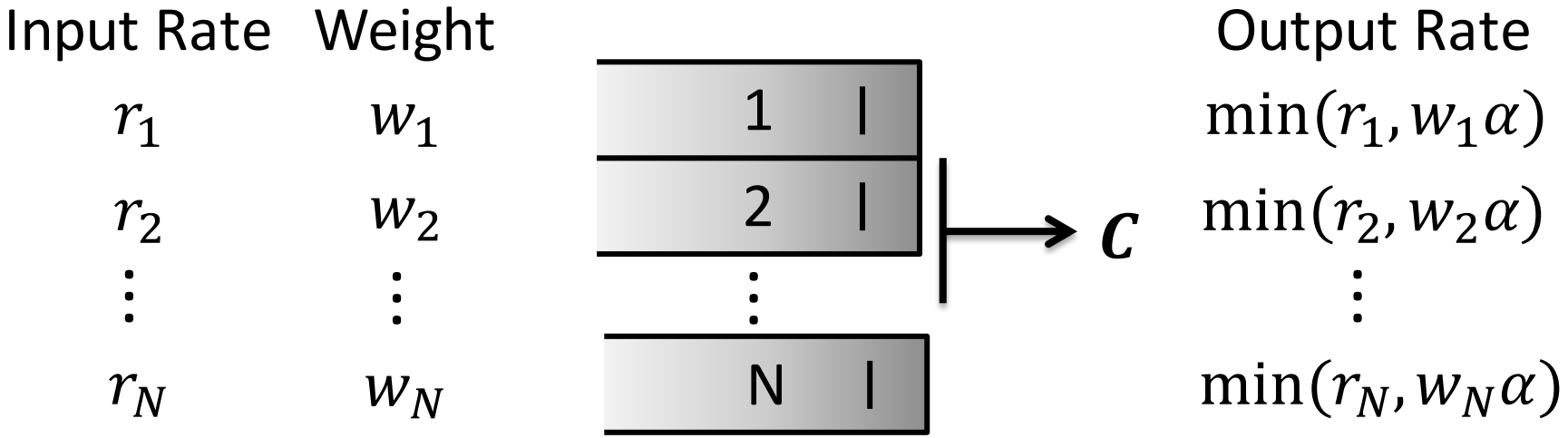
- N queues share the link with capacity C
- $C = \sum_{i=1}^{N} \min(r_i, w_i\alpha)$
- Use ECN to throttle queue i if $r_i > w_i\alpha$

Input Rate	Weight	Output Rate
r_1	w_1	$\min(r_1, w_1\alpha)$
r_2	w_2	$\min(r_2, w_2\alpha)$
$\vdots$	$\vdots$	$\vdots$
r_N	w_N	$\min(r_N, w_N\alpha)$



Start from GPS Model

- N queues share the link with capacity C
- $C = \sum_{i=1}^{N} \min(r_i, w_i\alpha)$
- $K_{queue(i)} = w_i\alpha \times RTT \times \lambda$

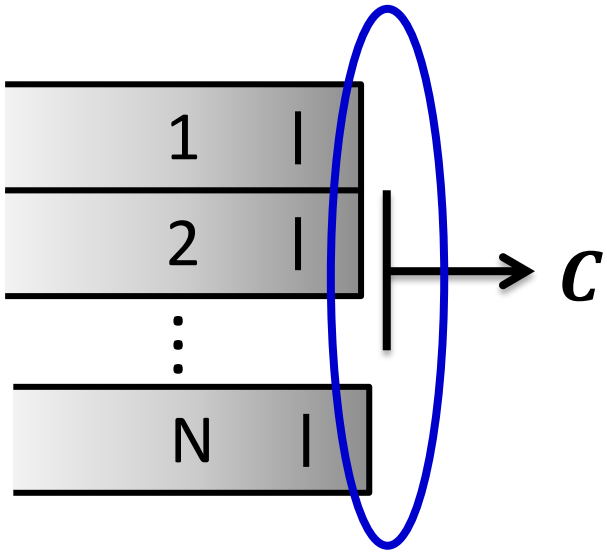


Start from GPS Model

- N queues share the link with capacity C
- $C = \sum_{i=1}^{N} \min(r_i, w_i\alpha)$
- $K_{queue(i)} = w_i\alpha \times RTT \times \lambda$

bit-by-bit round robin

Input Rate	Weight	Output Rate
r_1	w_1	$\min(r_1, w_1\alpha)$
r_2	w_2	$\min(r_2, w_2\alpha)$
$\vdots$	$\vdots$	$\vdots$
r_N	w_N	$\min(r_N, w_N\alpha)$

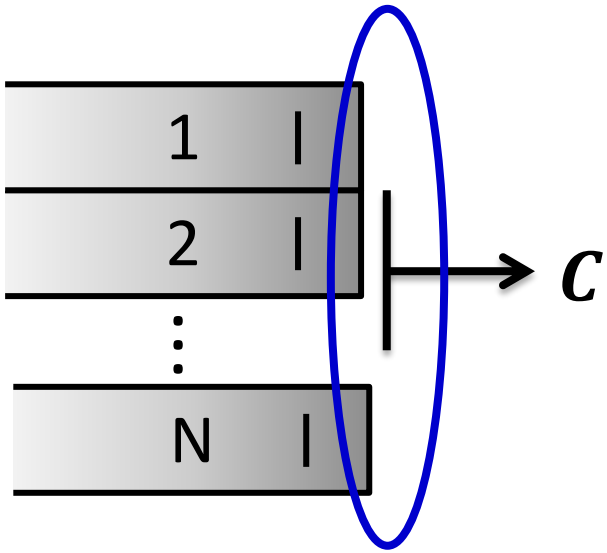


Start from GPS Model

- N queues share the link with capacity C
- $C = \sum_{i=1}^{N} \min(r_i, w_i\alpha)$
- $K_{queue(i)} = w_i\alpha \times RTT \times \lambda$

bit-by-bit round robin

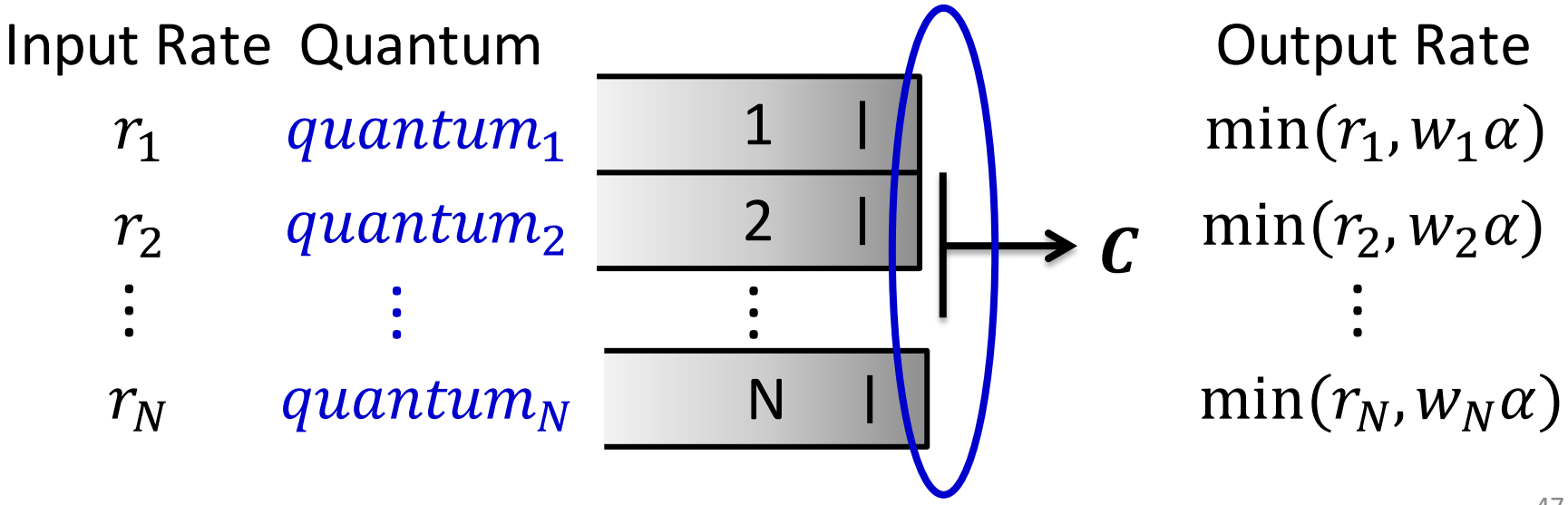
Input Rate	Quantum	Output Rate
r_1	w_1 bits	$\min(r_1, w_1\alpha)$
r_2	w_2 bits	$\min(r_2, w_2\alpha)$
⋮	⋮	⋮
r_N	w_N bits	$\min(r_N, w_N\alpha)$



Start from GPS Model

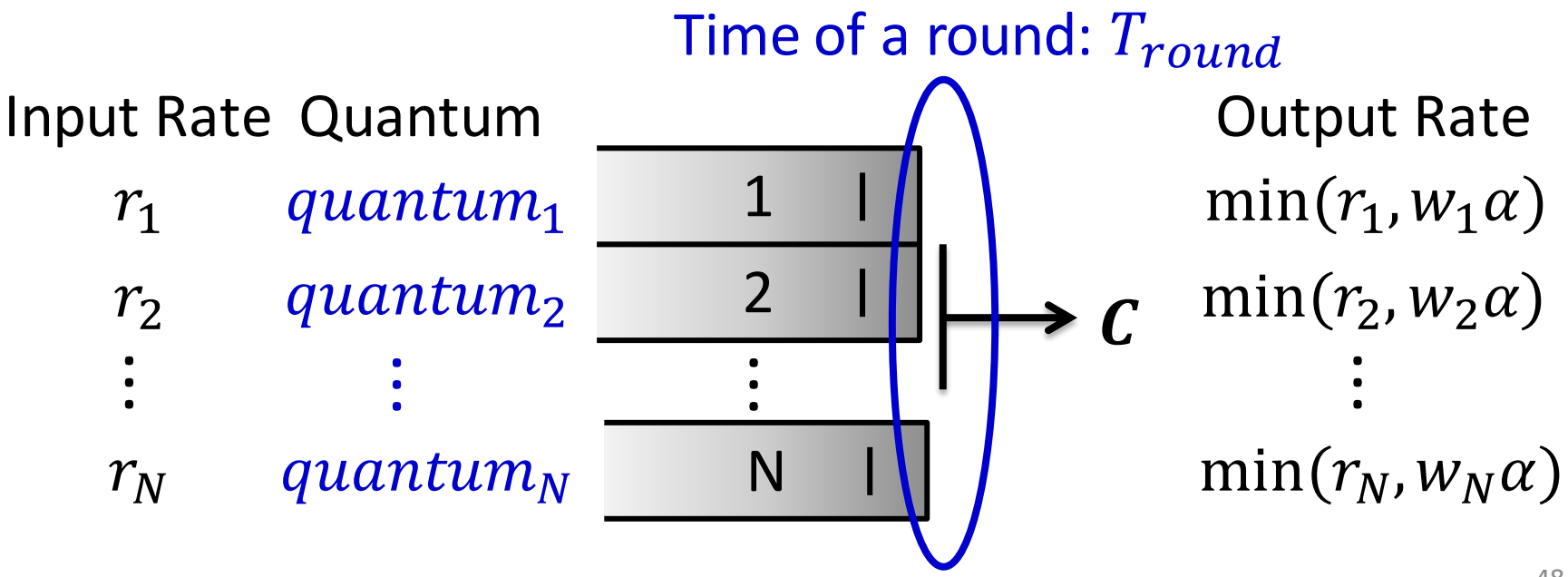
- N queues share the link with capacity C
- $C = \sum_{i=1}^{N} \min(r_i, w_i \alpha)$
- $K_{queue(i)} = w_i \alpha \times RTT \times \lambda$

Time of a round: T_{round}



Start from GPS Model

- N queues share the link with capacity C
- $C = \sum_{i=1}^{N} \min(r_i, w_i\alpha)$
- $K_{queue(i)} = quantum_i / T_{round} \times RTT \times \lambda$



MQ-ECN

- $K_{queue(i)} = quantum_i / T_{round} \times RTT \times \lambda$

MQ-ECN

- $K_{queue(i)} = quantum_i / T_{round} \times RTT \times \lambda$

Why does it work?

MQ-ECN

- $K_{queue(i)} = quantum_i / T_{round} \times RTT \times \lambda$
  - Deliver low latency
  - Achieve high throughput

$K_{queue(i)}$ adapts to traffic dynamics

MQ-ECN

- $K_{queue(i)} = quantum_i / T_{round} \times RTT \times \lambda$
  - Deliver low latency
  - Achieve high throughput
  - Preserve weighted fair sharing

$K_{queue(i)}$ is in proportion to the weight

MQ-ECN

- $K_{queue(i)} = quantum_i / T_{round} \times RTT \times \lambda$
  - Deliver low latency
  - Achieve high throughput
  - Preserve weighted fair sharing
  - Compatible with legacy ECN/RED implementation

Per-queue ECN/RED with *dynamic thresholds*

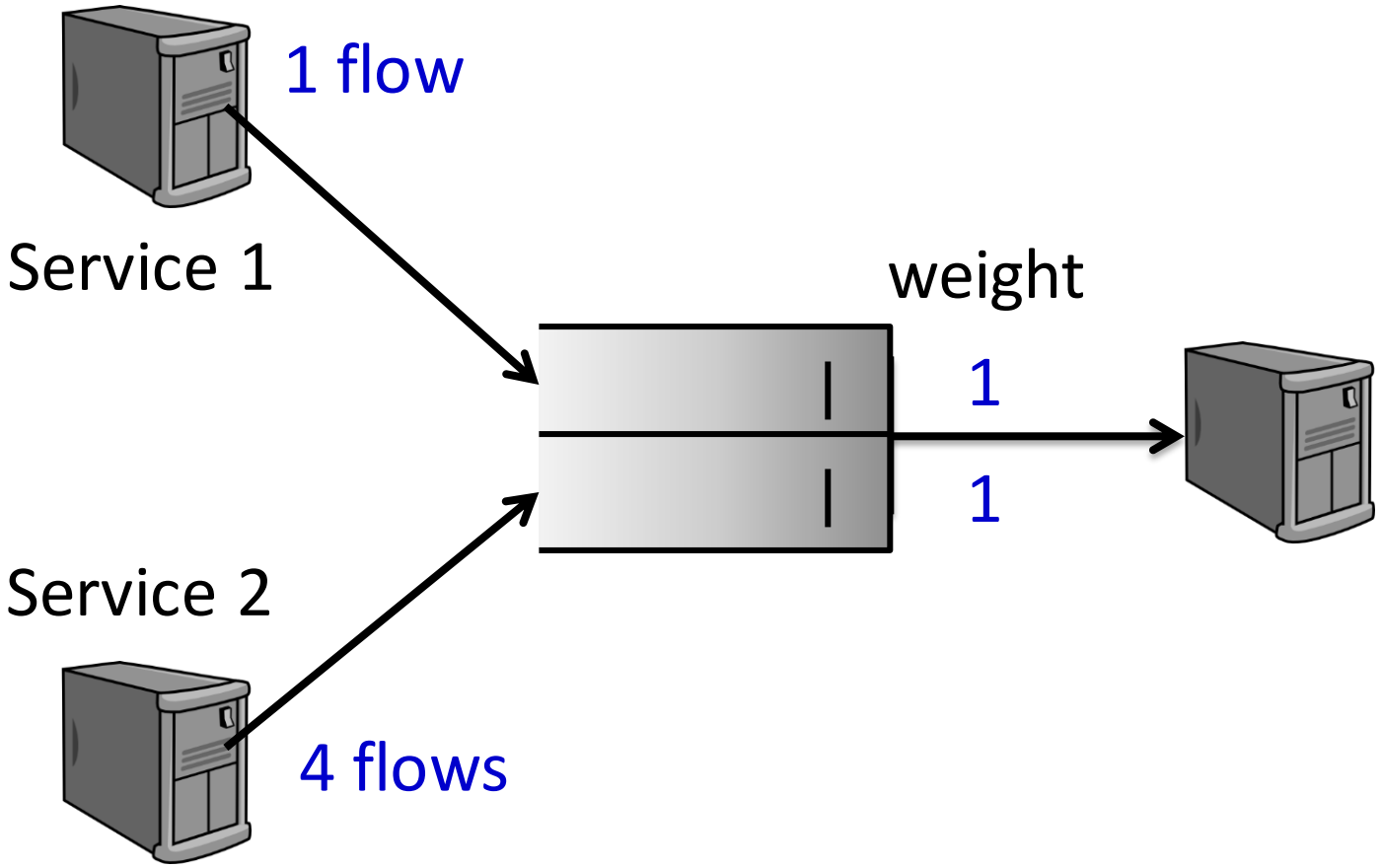
MQ-ECN

- $K_{queue(i)} = quantum_i / T_{round} \times RTT \times \lambda$
  - Deliver low latency
  - Achieve high throughput
  - Preserve weighted fair sharing
  - Compatible with legacy ECN/RED implementation
- More details
  - Handle inaccurate estimation of T_{round}
  - Apply to round-robin packet schedulers
    - DWRR, WRR, etc.

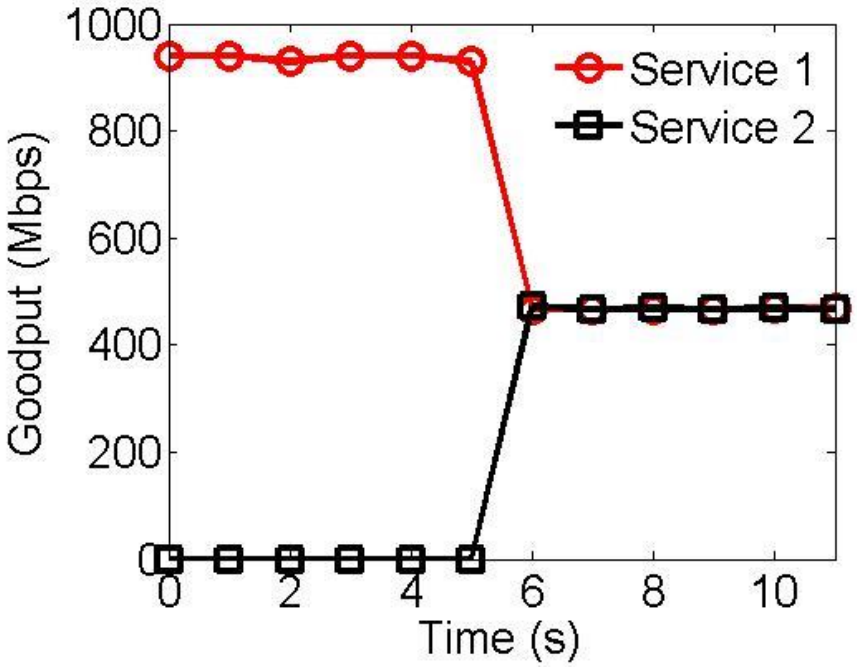
Testbed Evaluation

- MQ-ECN software prototype
  - Linux qdisc kernel module performing DWRR
- Testbed setup
  - 9 servers are connected to a server-emulated switch with 9 NICs
  - End-hosts use DCTCP as the transport protocol
- Benchmark traffic
  - Web search (DCTCP paper)
- More results in large-scale simulations

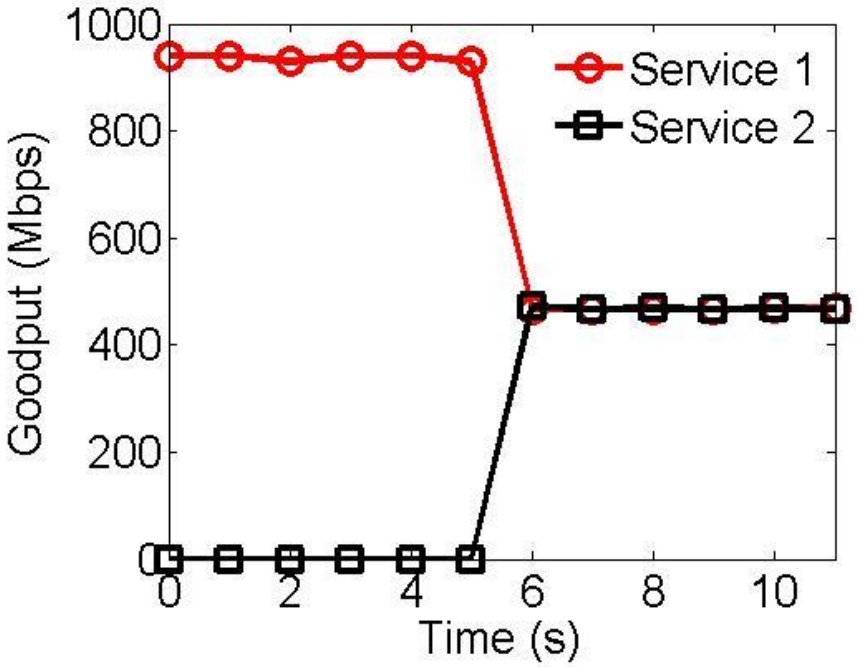
Static Flow Experiment



Static Flow Experiment

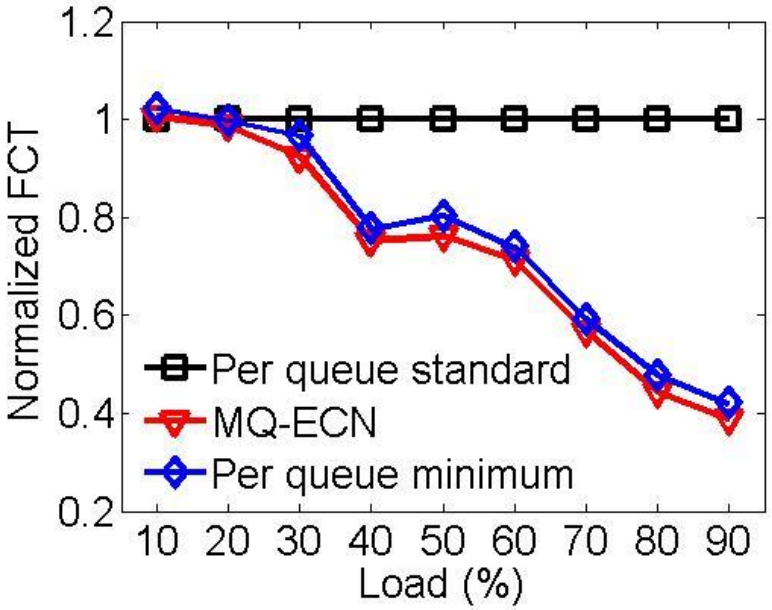


Static Flow Experiment

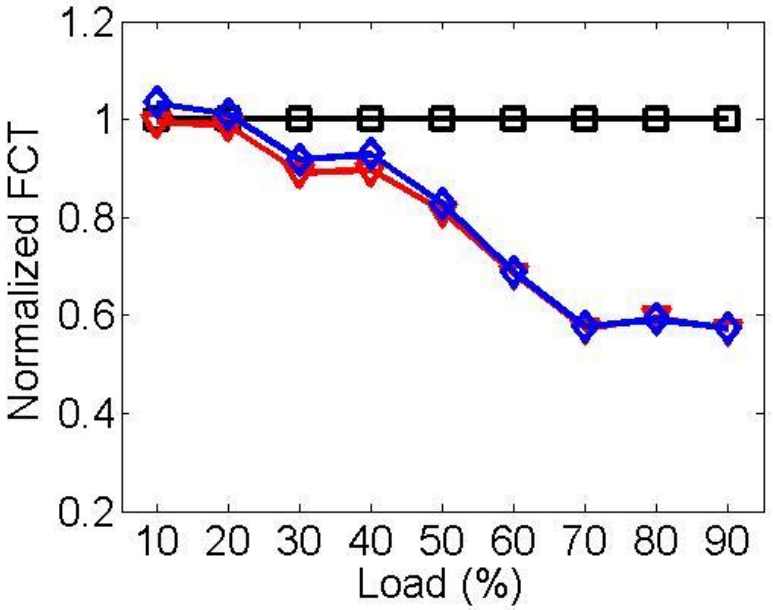


MQ-ECN preserves weighted fair sharing

Realistic Traffic: Small Flows (<100KB)

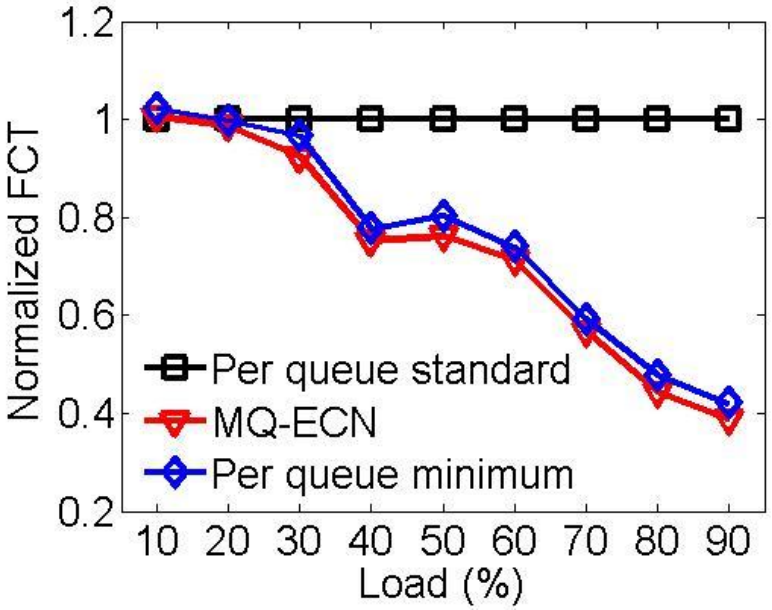


Balanced traffic pattern

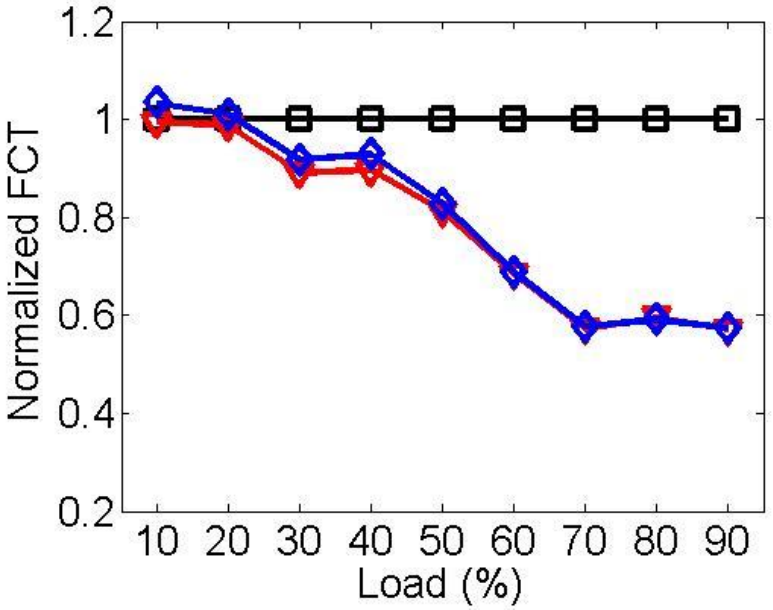


Unbalanced traffic pattern

Realistic Traffic: Small Flows (<100KB)



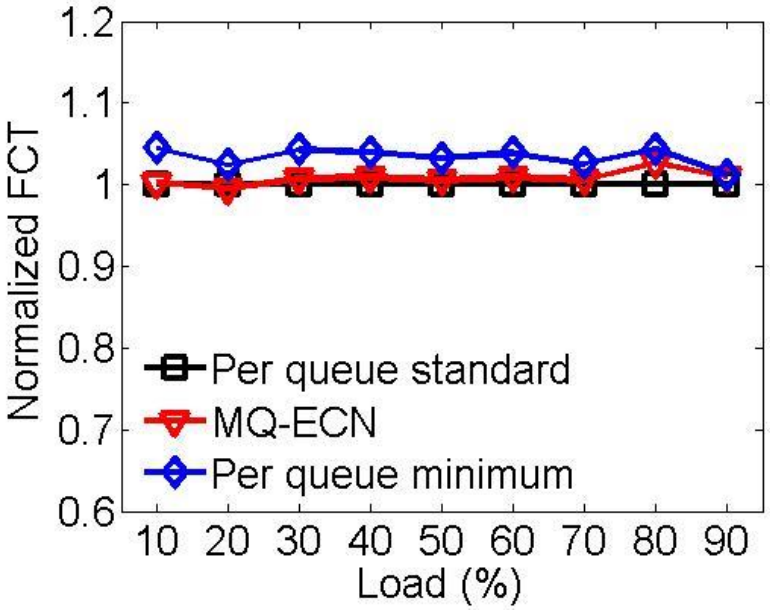
Balanced traffic pattern



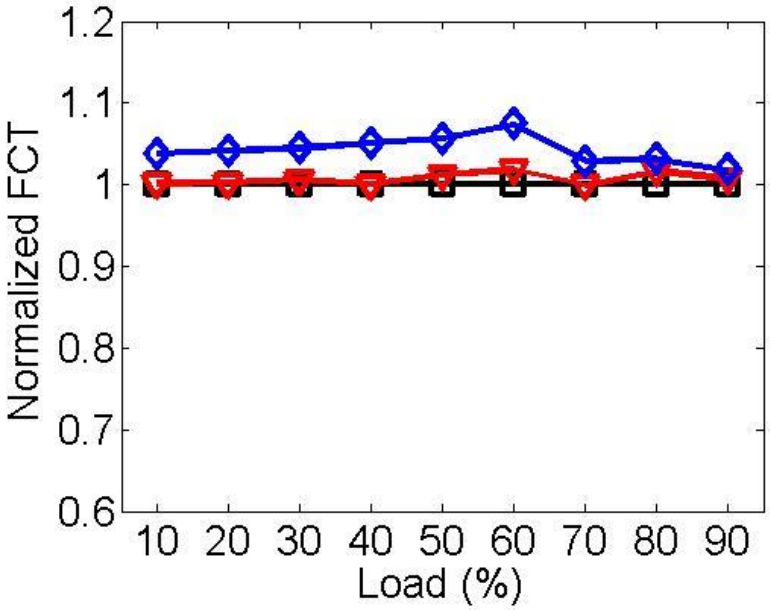
Unbalanced traffic pattern

MQ-ECN achieves low latency

Realistic Traffic: Large Flows (>10MB)

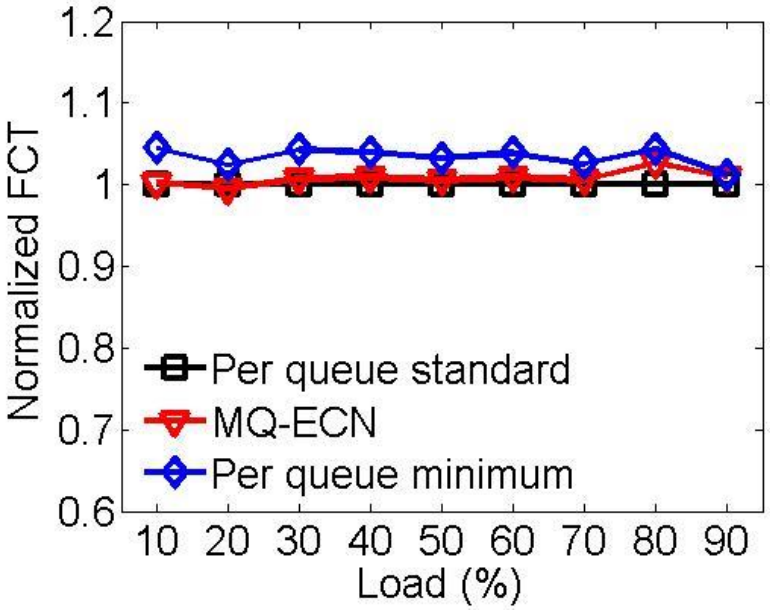


Balanced traffic pattern

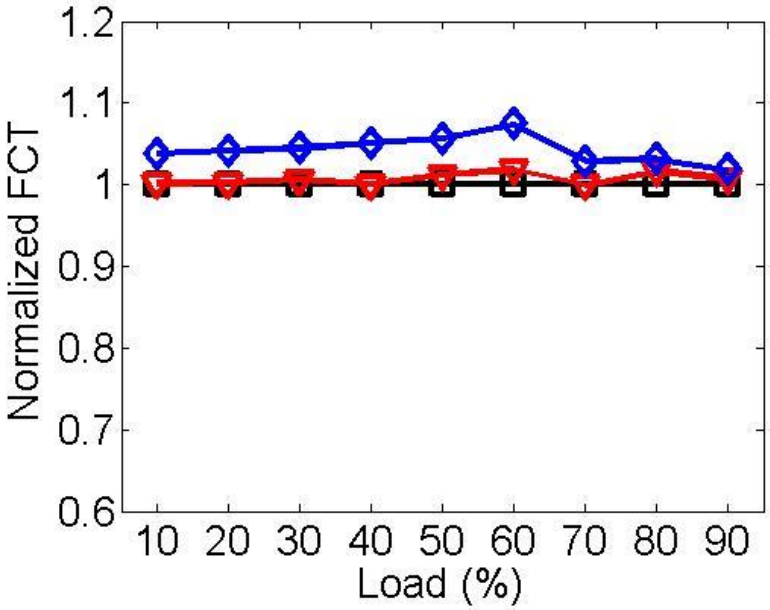


Unbalanced traffic pattern

Realistic Traffic: Large Flows (>10MB)



Balanced traffic pattern



Unbalanced traffic pattern

MQ-ECN achieves high throughput

Conclusions

- Identify performance impairments of existing ECN/RED schemes in multi-queue context
- MQ-ECN: simple, yet effective
  - High throughput, low latency, weighted fair sharing
  - Currently, apply to round-robin packet schedulers
- ECN marking scheme for arbitrary schedulers is an important ongoing effort
- Code: http://sing.cse.ust.hk/projects/MQ-ECN

Thanks!

Support Arbitrary Packet Schedulers

- Find a general solution to estimate per-queue *effective* draining rate
  - Draining rate should be measured when the queue keeps congested
    - $K_{queue(i)} = \boldsymbol{w_i \alpha} \times RTT \times \lambda$
  - Measurement window is hard to decide