# FIFO Service with Differentiated Queueing

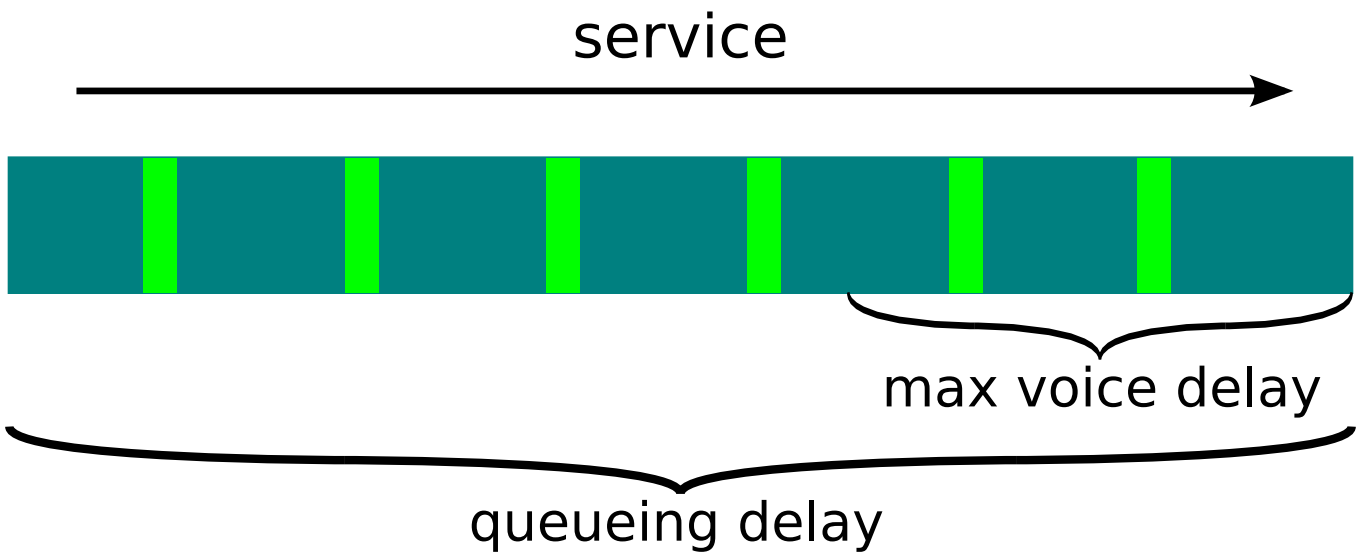
Martin Karsten

David R. Cheriton School of Computer Science
University of Waterloo
(currently on Sabbatical at University of Kaiserslautern)

ANCS 2011

# Motivation

- residential access link – concurrent flows:
  - file transfer (using TCP)
  - voice call
- link buffer fully utilized



# Table of Contents

1 Motivation

2 Problem Statement

3 Algorithm

4 Implementation

5 Evaluation

6 Wrap Up

# Table of Contents

1 Motivation

2 Problem Statement

3 Algorithm

4 Implementation

5 Evaluation

6 Wrap Up

# Rate Neutrality

- avoid control and accounting overhead
  $=>$ avoid preferential treatment of traffic
- use shared FIFO service as benchmark
  - service rates proportional to arrival rates
- rate control at edge/end nodes: TCP, etc.

# Rate Neutrality

- avoid control and accounting overhead
  $=>$ avoid preferential treatment of traffic
- use shared FIFO service as benchmark
  - service rates proportional to arrival rates
- rate control at edge/end nodes: TCP, etc.
- classical packet scheduling?
  - absolute: needs rate allocation (signalling)
  - priority: preferential treatment

  $=>$ distorts edge/end control

# Delay Control

- delay control without rate increase?
  –> packet discard
- rate neutral with packet discard?
  –> preserve service

# Delay Control

- delay control without rate increase?
  –> packet discard
- rate neutral with packet discard?
  –> preserve service
- proposal: multi-class queueing system
  - maximum queueing delay per class
  - preserve service within class
  - throughput similar to corresponding FIFO

# Incentive Compatibility

- end/edge systems freely choose service class
- no preferential treatment
- lower delay = less buffer = higher loss
  $=>$ *strategy-proof*

# Incentive Compatibility

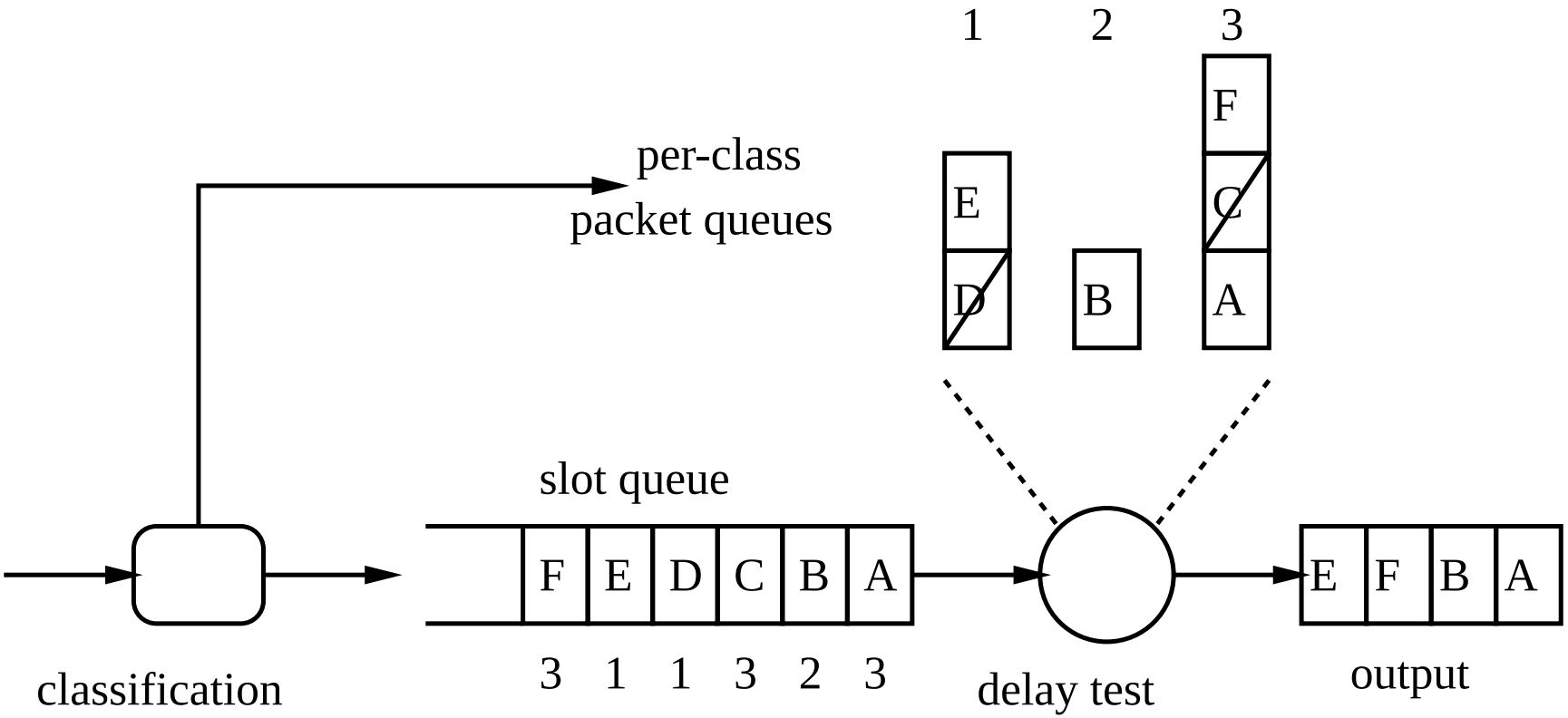
- end/edge systems freely choose service class
- no preferential treatment
- lower delay = less buffer = higher loss
  => *strategy-proof*

- not addressed: indirect effects (TCP, etc.)
  - smaller RTT –> higher sending rate
  - enforce transparency at router?
    hard-code router policy for specfic e2e mechanism?
    sound architecture? modular design?

# Use Cases

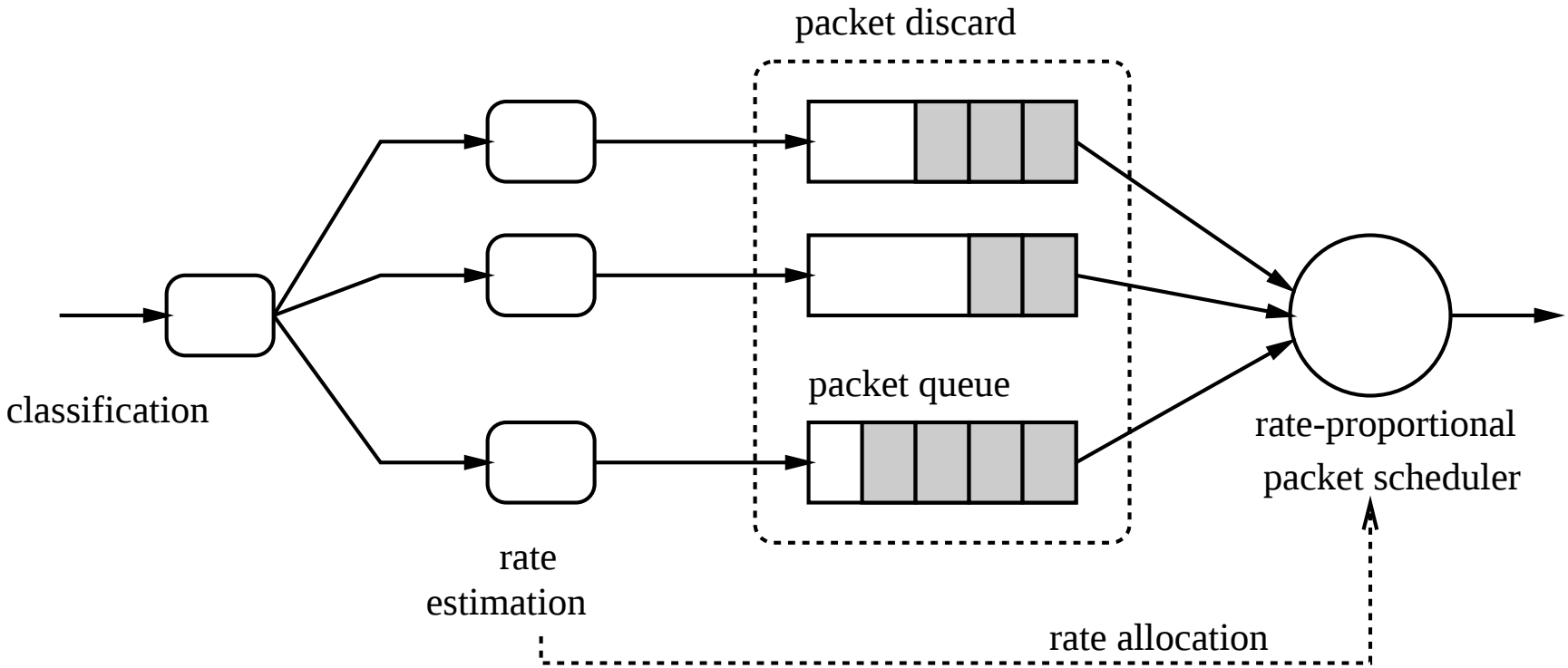
- isolated deployment
  - peering exchange, residential gateway
- edge-based load control
  - IETF PCN architecture
- small router buffers
  - experimentation and transition
- small number of (standardized) delay classes

# Conceptual Design




- late delay test: worst-case linear complexity

# Alternative Conceptual Design




- rate estimation/allocation: complexity, accuracy, time lag

# Table of Contents

1 Motivation

2 Problem Statement

3 Algorithm

4 Implementation

5 Evaluation

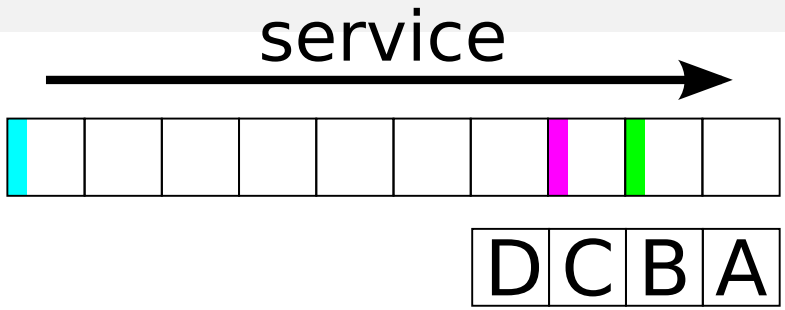
6 Wrap Up

# Virtually Isolated FIFO Queueing

- manage FIFO queue of *virtual slots*
  - virtual slot: packet in regular FIFO
    here: right to send at some point in time
- admit packet, if virtual slot available in queue
  - with suitable service time
  - store in packet queue (sorted by service time)
- purge unused virtual slots from system
  - avoid virtual buffer hogging

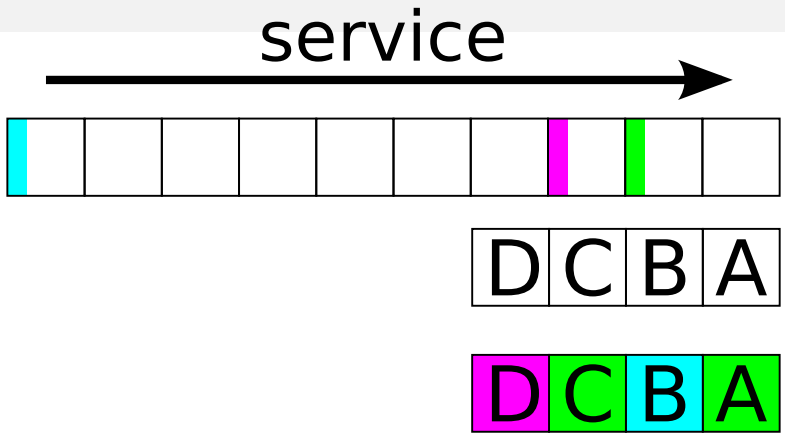
# VIFQ Operation

- buffer
- arrival



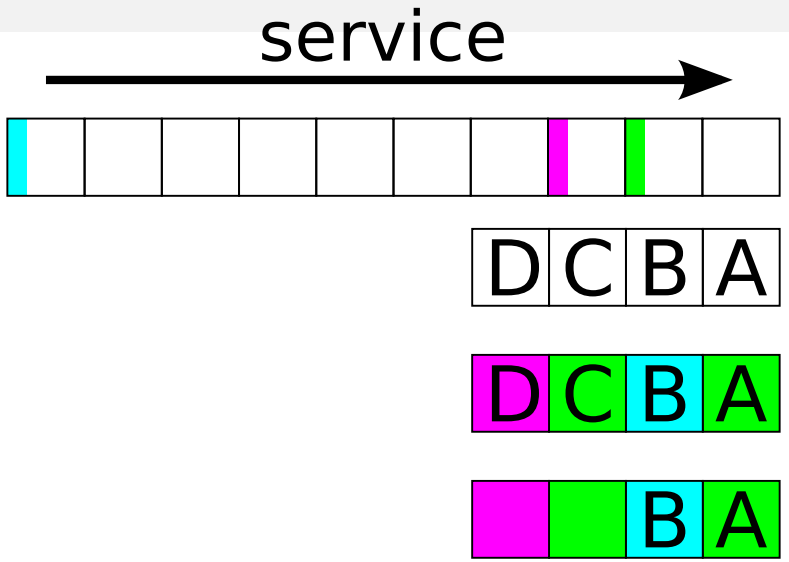
# VIFQ Operation

- buffer
- arrival
- classification



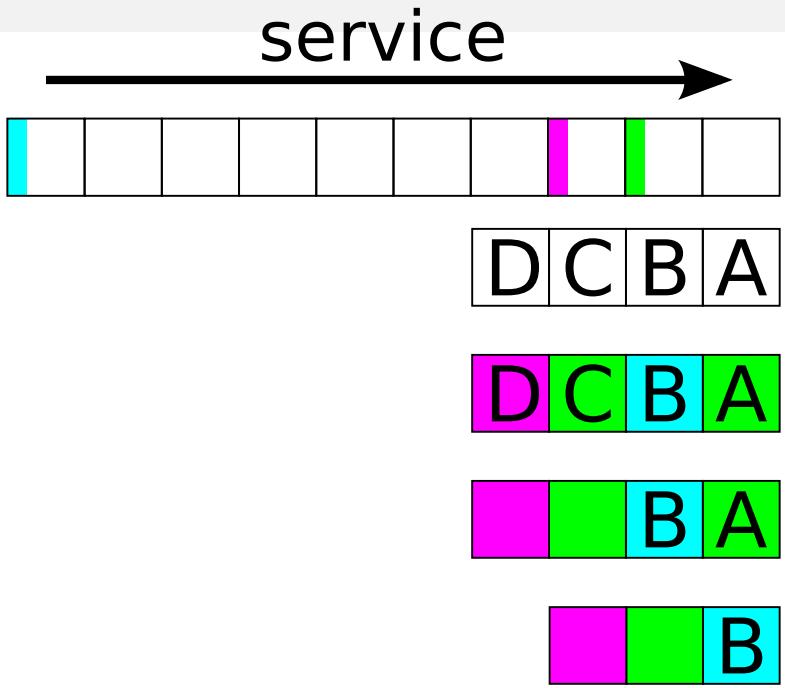
# VIFQ Operation

- buffer
- arrival
- classification
- packet/slot queue



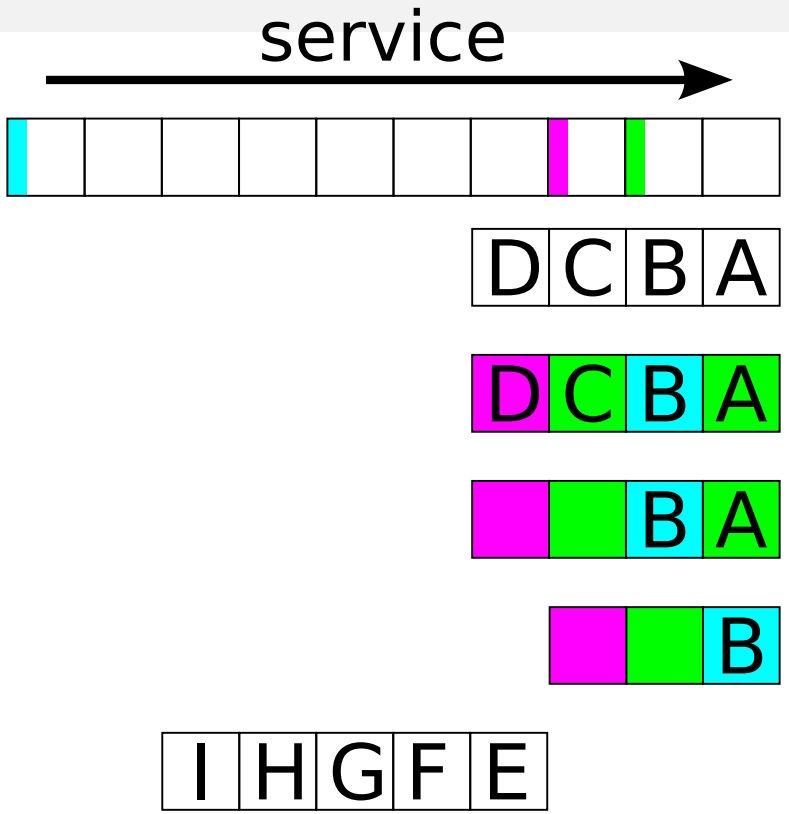
# VIFQ Operation

- buffer
- arrival
- classification
- packet/slot queue
- service



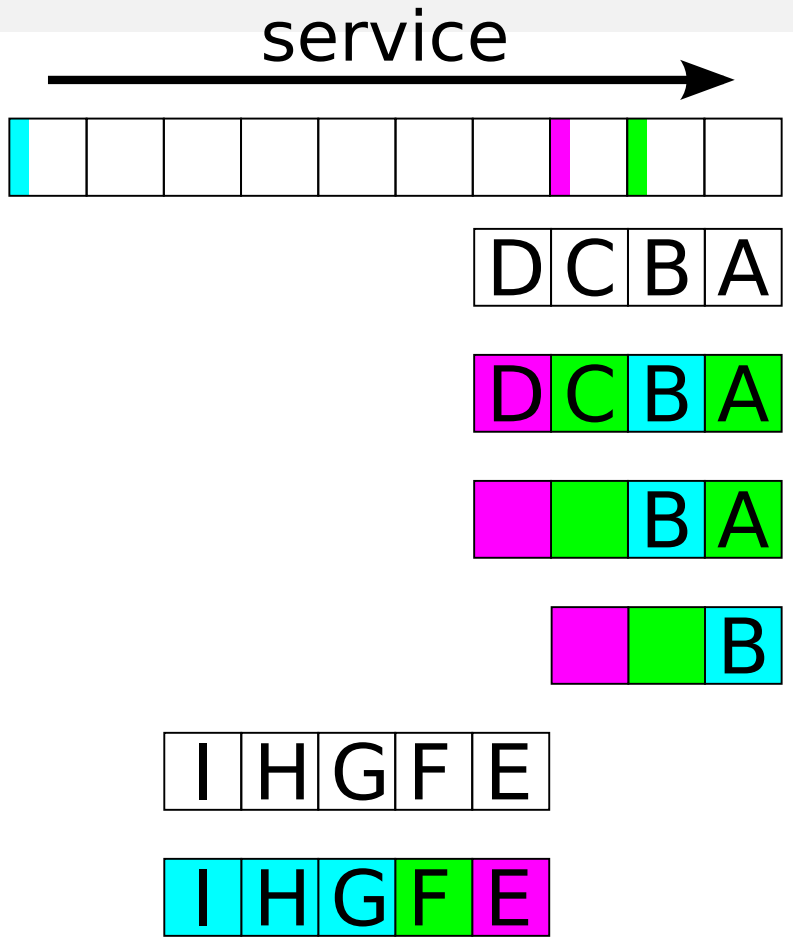
# VIFQ Operation

- buffer
- arrival
- classification
- packet/slot queue
- service
- arrival



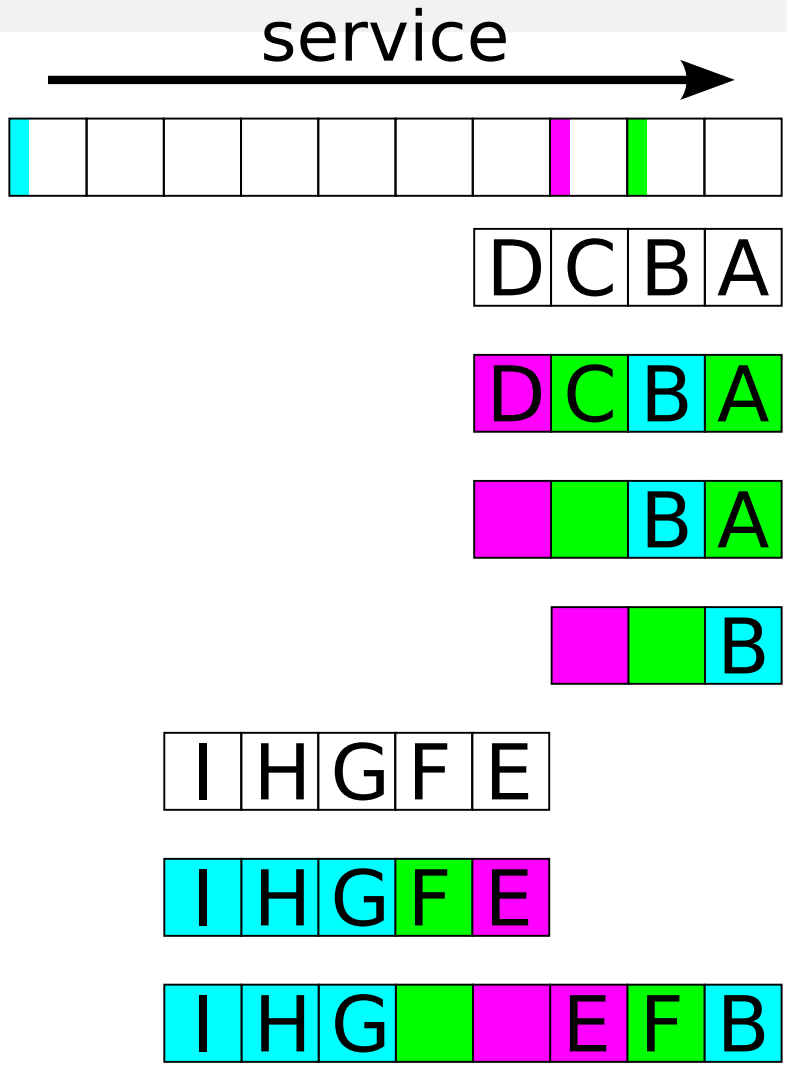
# VIFQ Operation

- buffer
- arrival
- classification
- packet/slot queue
- service
- arrival
- classification



# VIFQ Operation

- buffer
- arrival
- classification
- packet/slot queue
- service
- arrival
- classification
- packet/slot queue



# Table of Contents

1 Motivation

2 Problem Statement

3 Algorithm

4 Implementation

5 Evaluation

6 Wrap Up

# Complexity in Arrival Routine

- two loops proportional to arriving packet length
  1. purge unused slots
  2. reconcile different packet lengths
- packet-amortized constant complexity
  - router designed to handle minimum size packets
  - extra CPU capacity for larger packets

# Sorted Packet Queue

- hardware-assisted priority queue => O(1)
  - timer wheel with *find-first-set* instruction
- software tree/heap-based priority queue
- worst-case: O(logN) in small number of classes

# Table of Contents

1 Motivation

2 Problem Statement

3 Algorithm

4 Implementation

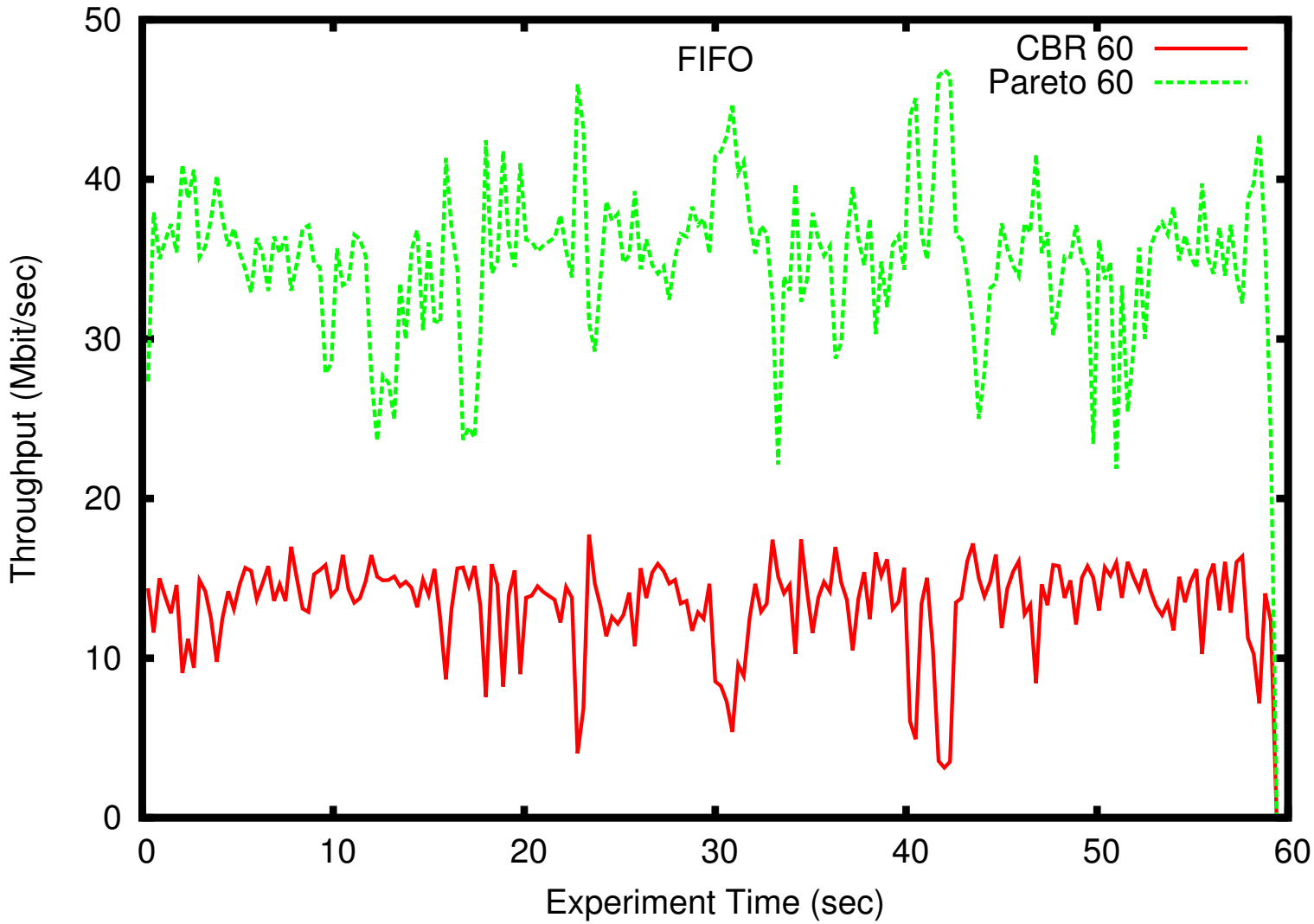
5 Evaluation

6 Wrap Up

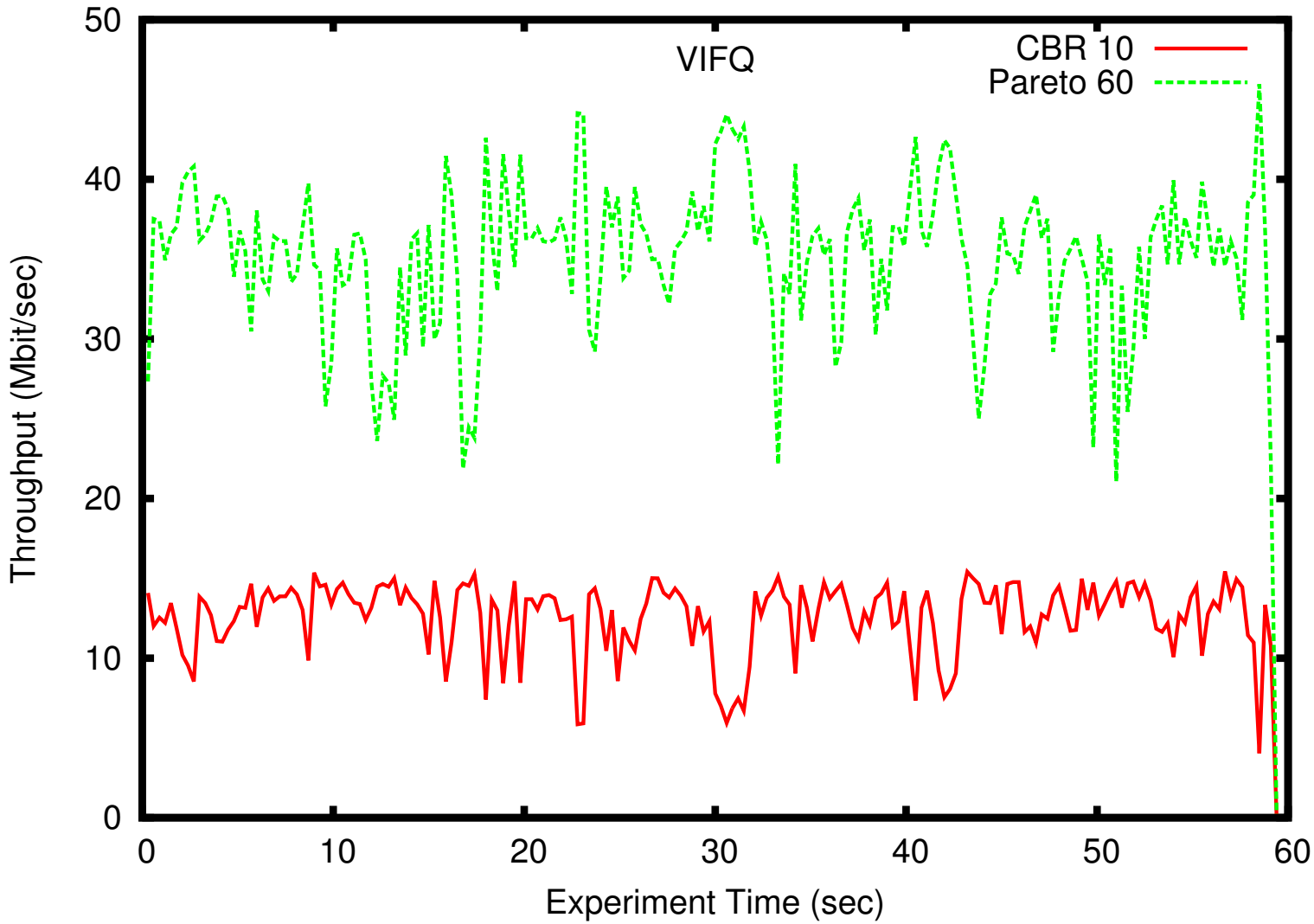
# Simulation Parameters

- dumbbell topology
- 50 Mbit/s bottleneck
- 60 msec roundtrip propagation delay
- methodology
  - compare throughput, utilization, etc., with FIFO
  - verify delay differentiation

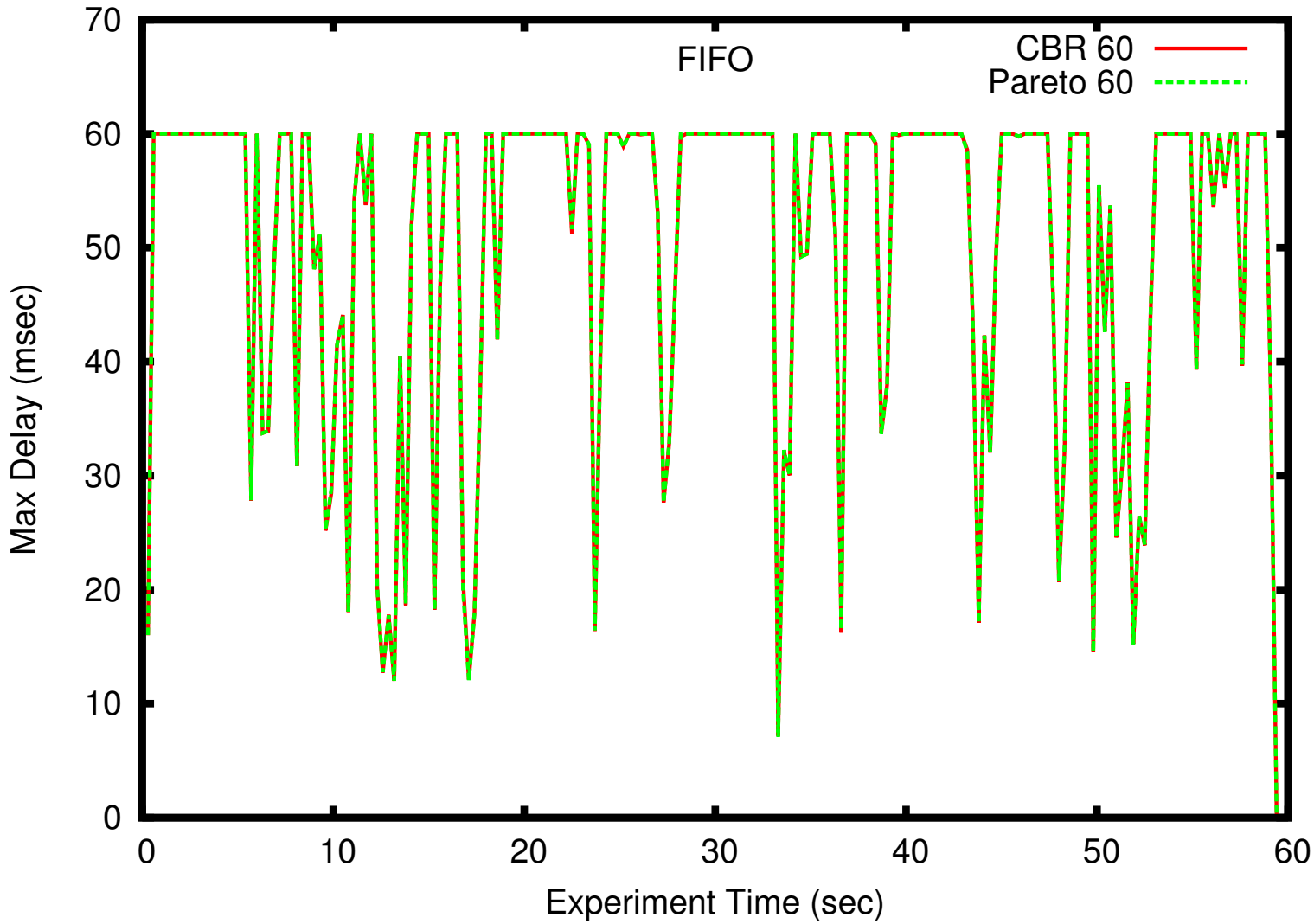
# CBR vs. Pareto - Example



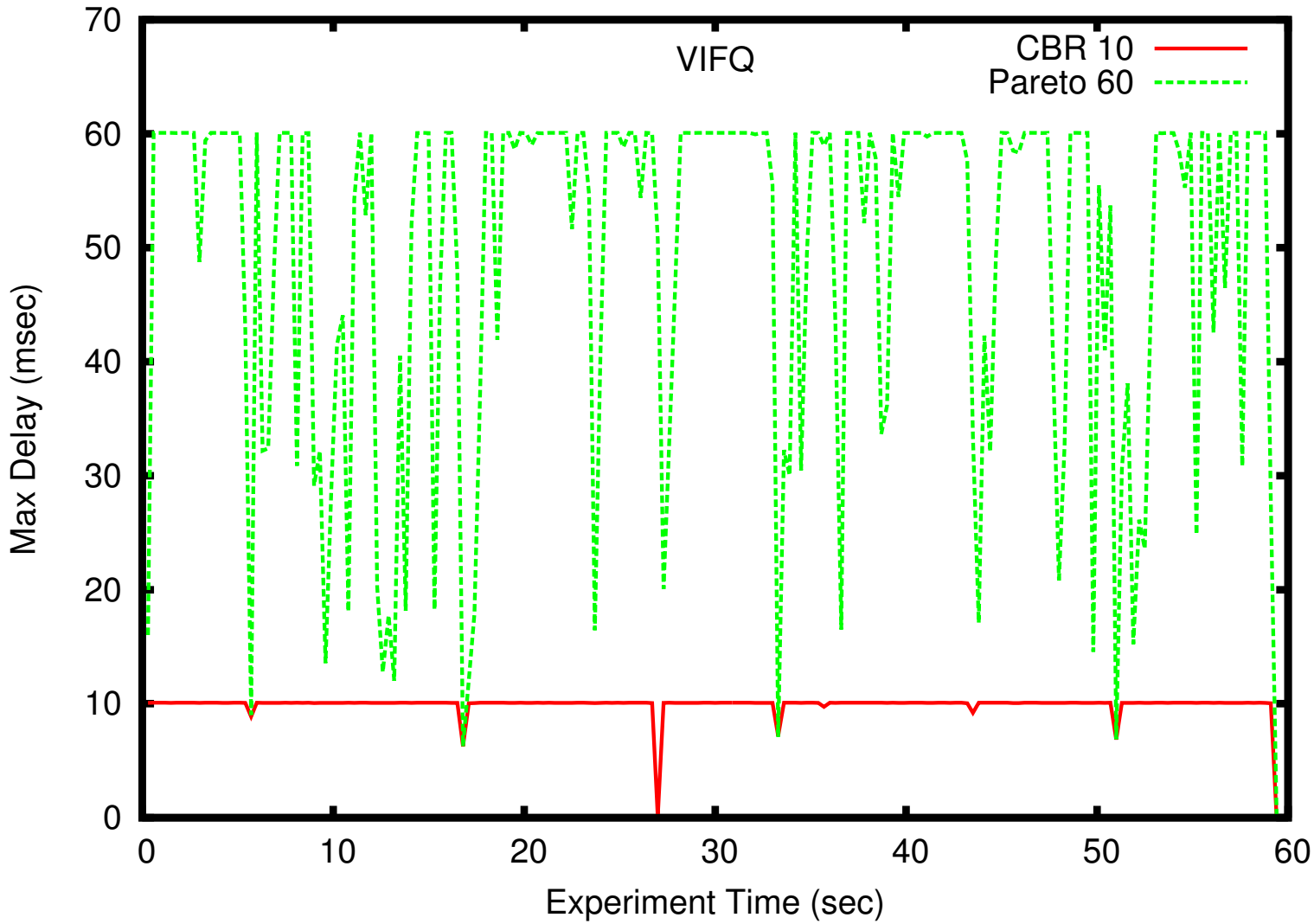
# CBR vs. Pareto - Example



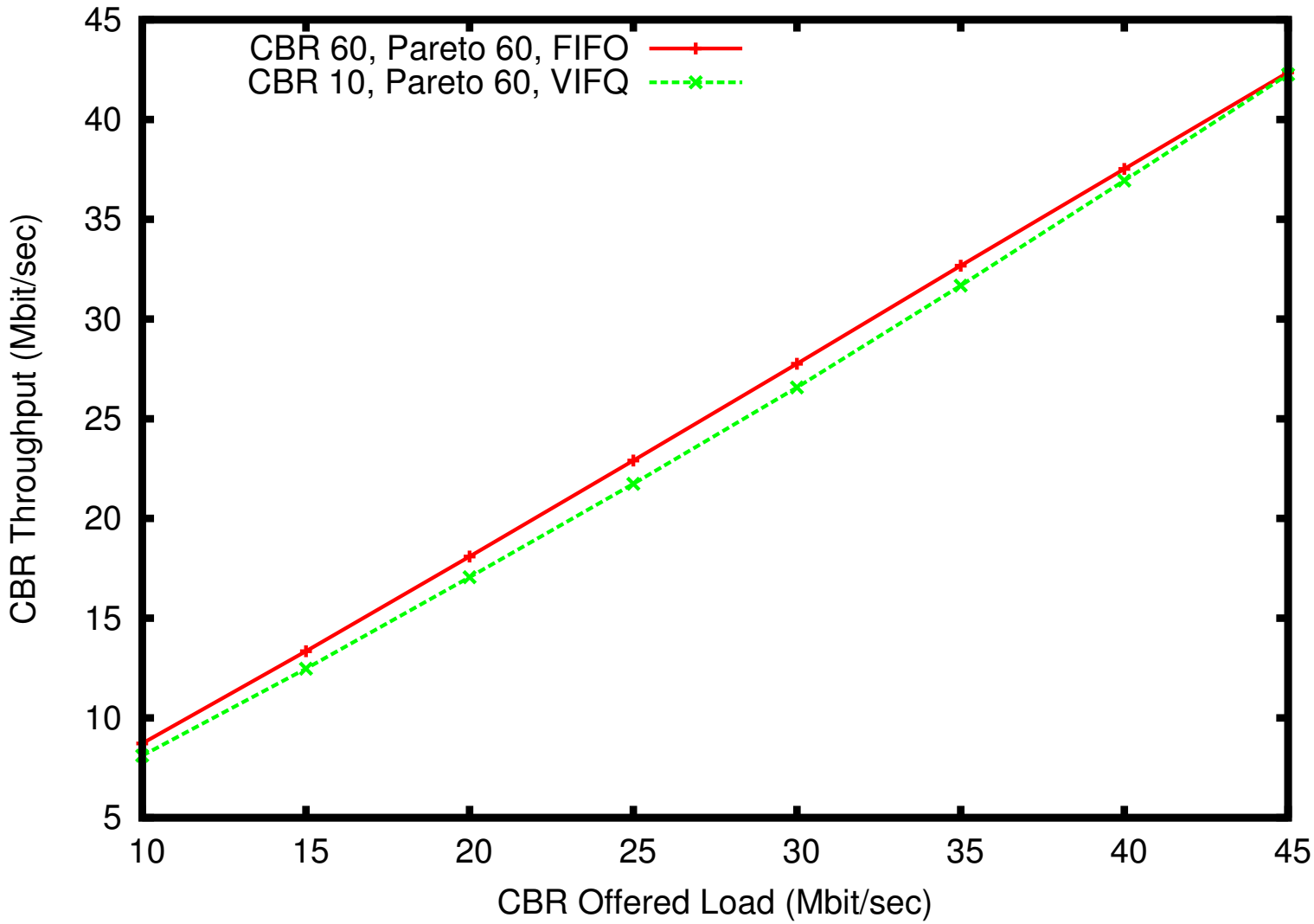
# CBR vs. Pareto - Example



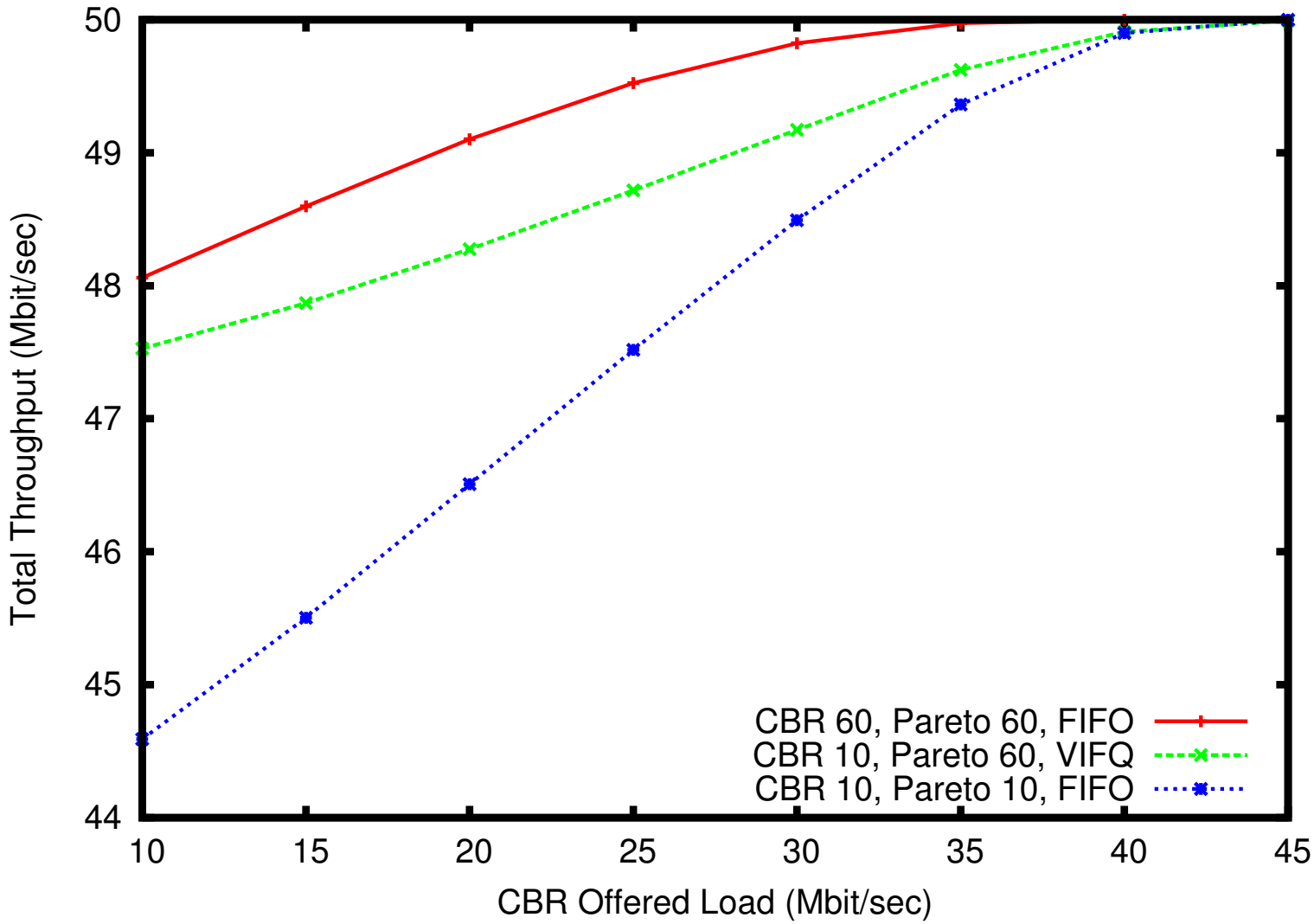
# CBR vs. Pareto - Example



# CBR vs. Pareto - Throughput



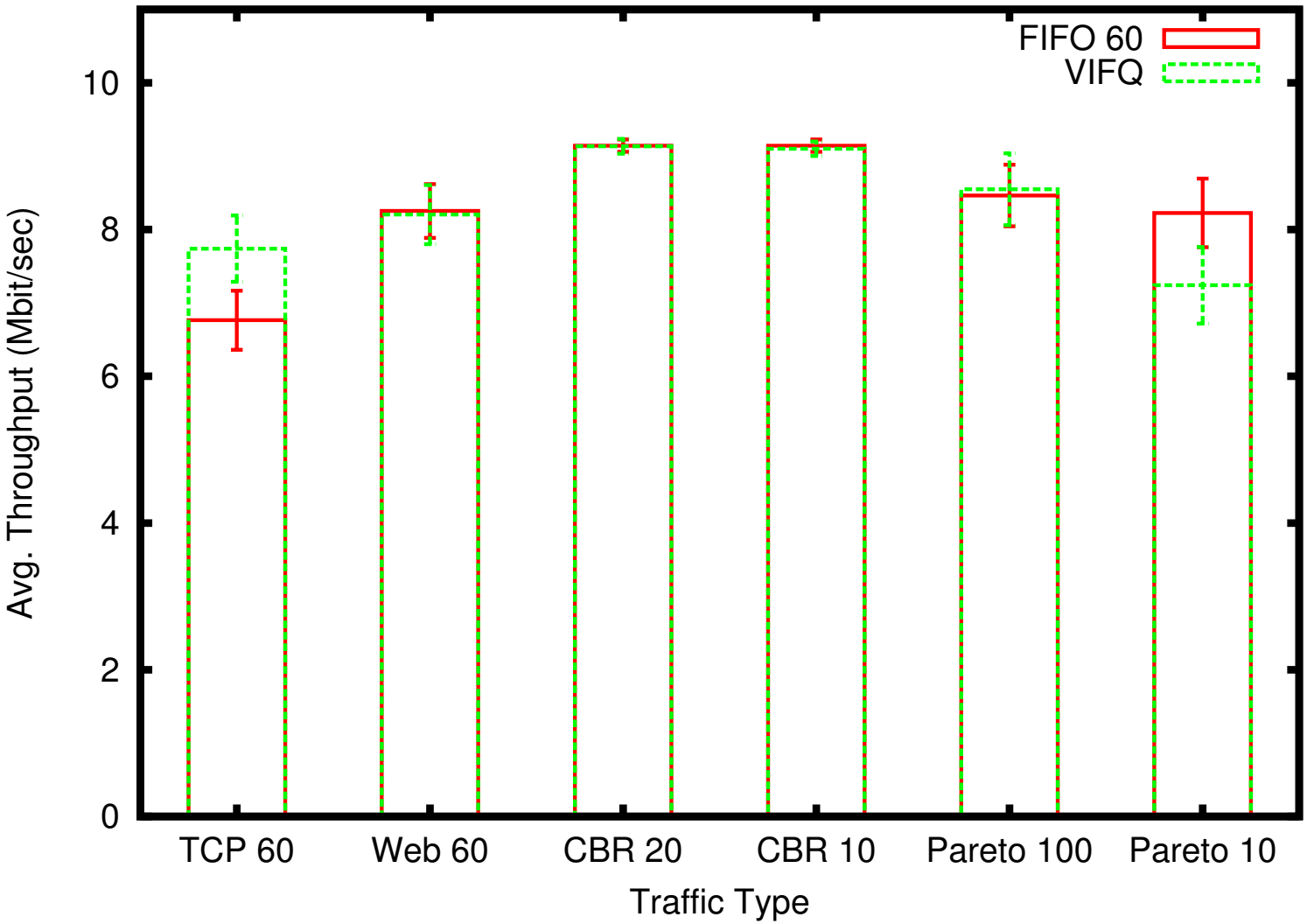
# CBR vs. Pareto - Utilization



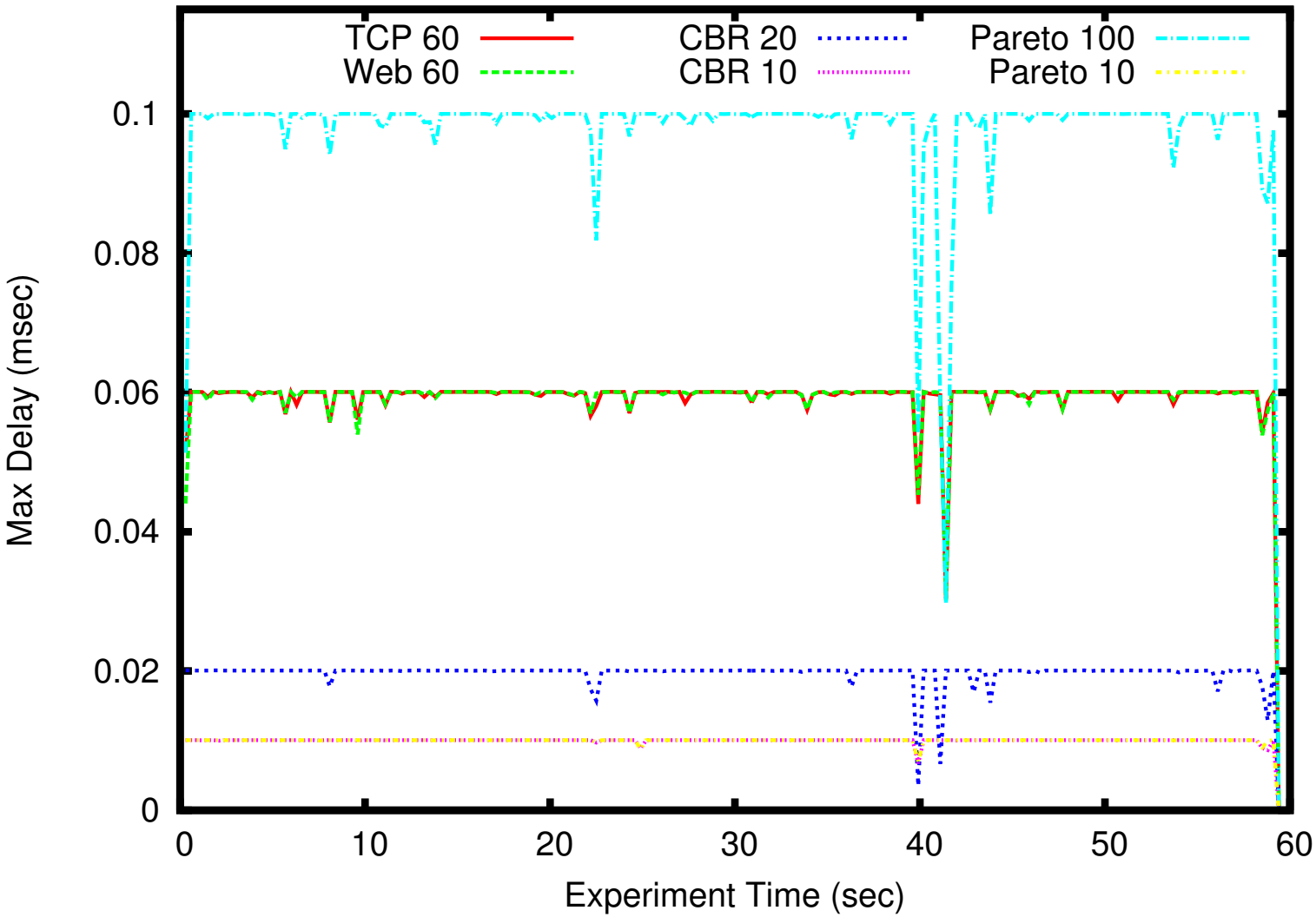
# Multiple Classes

| delay class | traffic type |
|---|---|
| 60 msec | 50 long-term, greedy TCP flows |
| 60 msec | 10/sec short (web) TCP flows, 100KB |
| 20 msec | CBR, 20% load |
| 10 msec | CBR, 20% load |
| 100 msec | Pareto, 20% load |
| 10 msec | Pareto, 20% load |

# Multiple Classes - Throughput



# Multiple Classes - Delay



# Table of Contents

1 Motivation

2 Problem Statement

3 Algorithm

4 Implementation

5 Evaluation

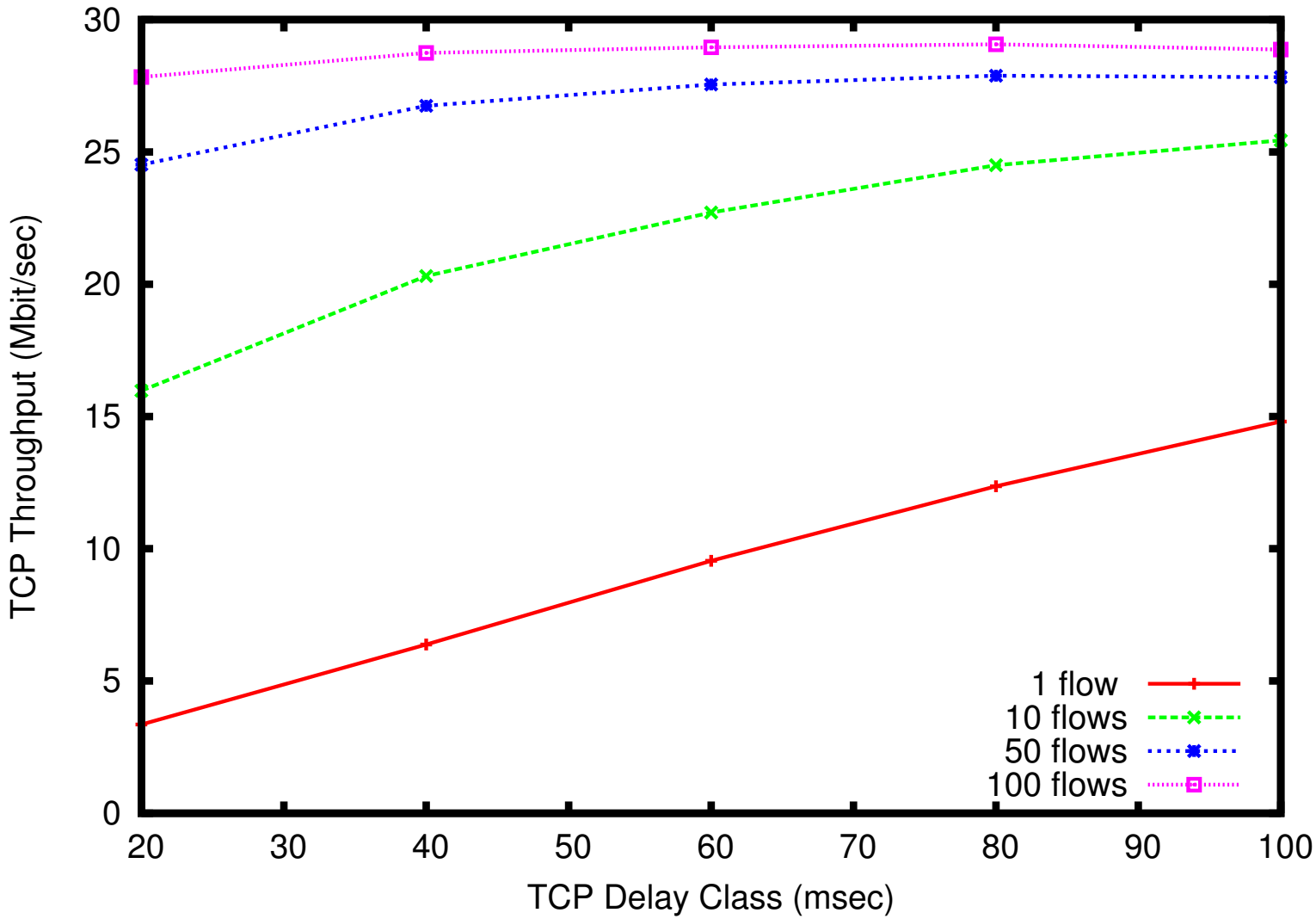
6 Wrap Up

# Wrap Up

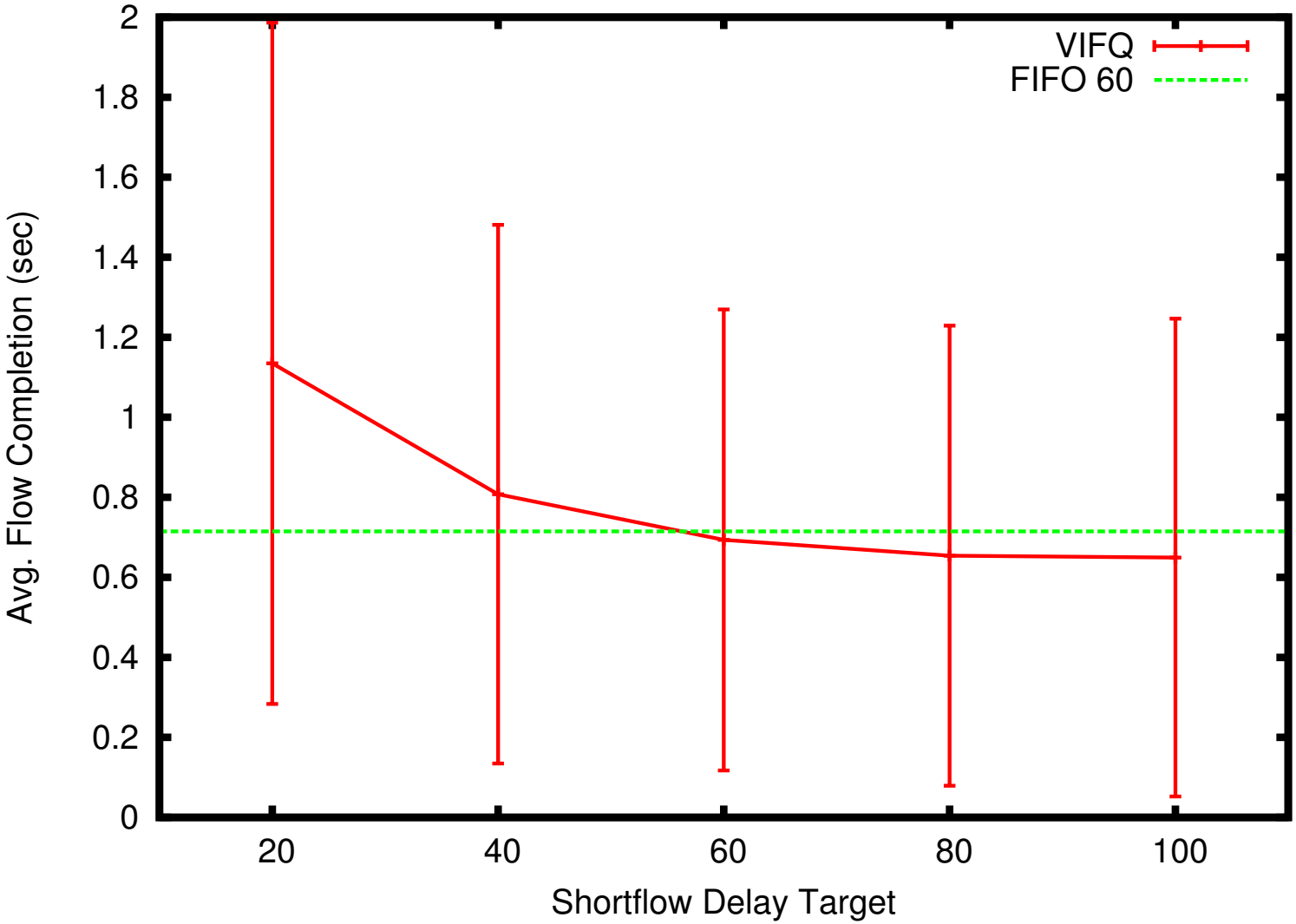
- new approach to differentiated delay control
  - rate neutral, incentive-compatible
- VIFQ concept: versatile building block
- VIFQ algorithm: simple and feasible
- initial evaluation results promising
- next steps: scenarios, modelling, implementation

# Extra Slides

# TCP vs. Pareto - Throughput



# Short Flows - Completion Time



# TCP/TFRC - Throughput