RC3
Recursively Cautious Congestion Control

Radhika Mittal, Justine Sherry,
Sylvia Ratnasamy, Scott Shenker
UC Berkeley

Roadmap

- *Isn't congestion control a solved problem?*
- *Scope for performance gains*
- *Design Details*
- *Simulation Results*
- *Linux Implementation and Evaluation*
- *Challenges and Future*

Roadmap

- **Isn't congestion control a solved problem?**
- *Scope for performance gains*
- *Design Details*
- *Simulation Results*
- *Linux Implementation and Evaluation*
- *Challenges and Future*

Short Flow Completion Time

- "Being fast really matters. Users really respond to speed."
  - *0.5 sec delay caused a 20% drop in traffic – Google*
  - *2 sec slowdown changed queries/user by -1.8% and revenue/user by -4.3% – Bing*
  - *5 sec speedup resulted in a 25% increase in page views and 7-12% increase in revenue – Shopzilla*

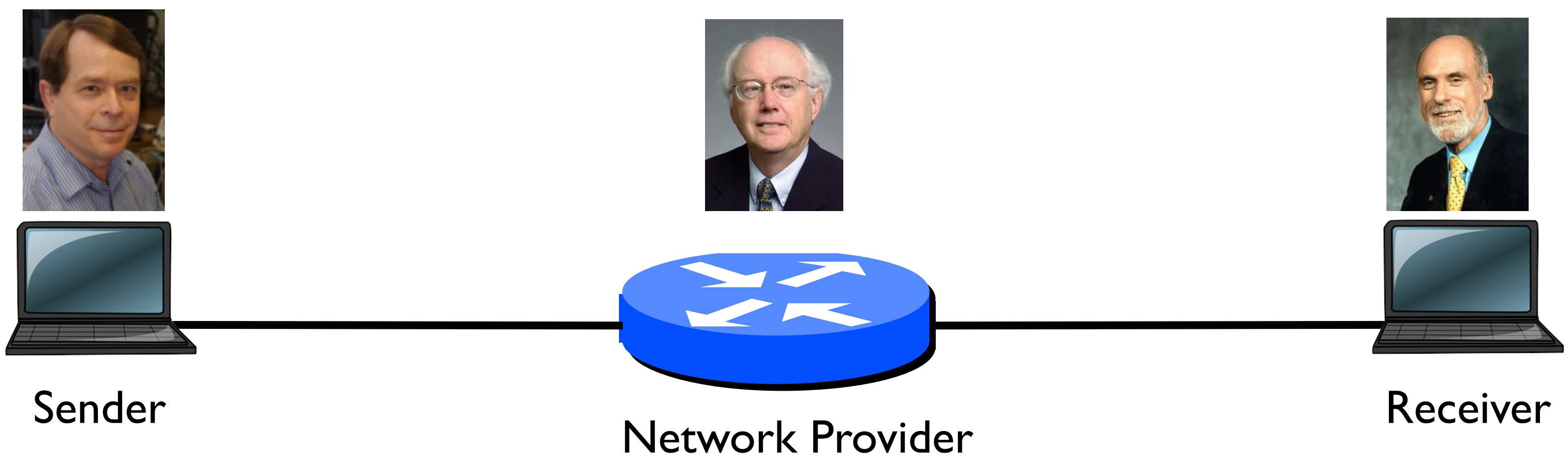
- James Hamilton's Blog

RC3 in a nutshell

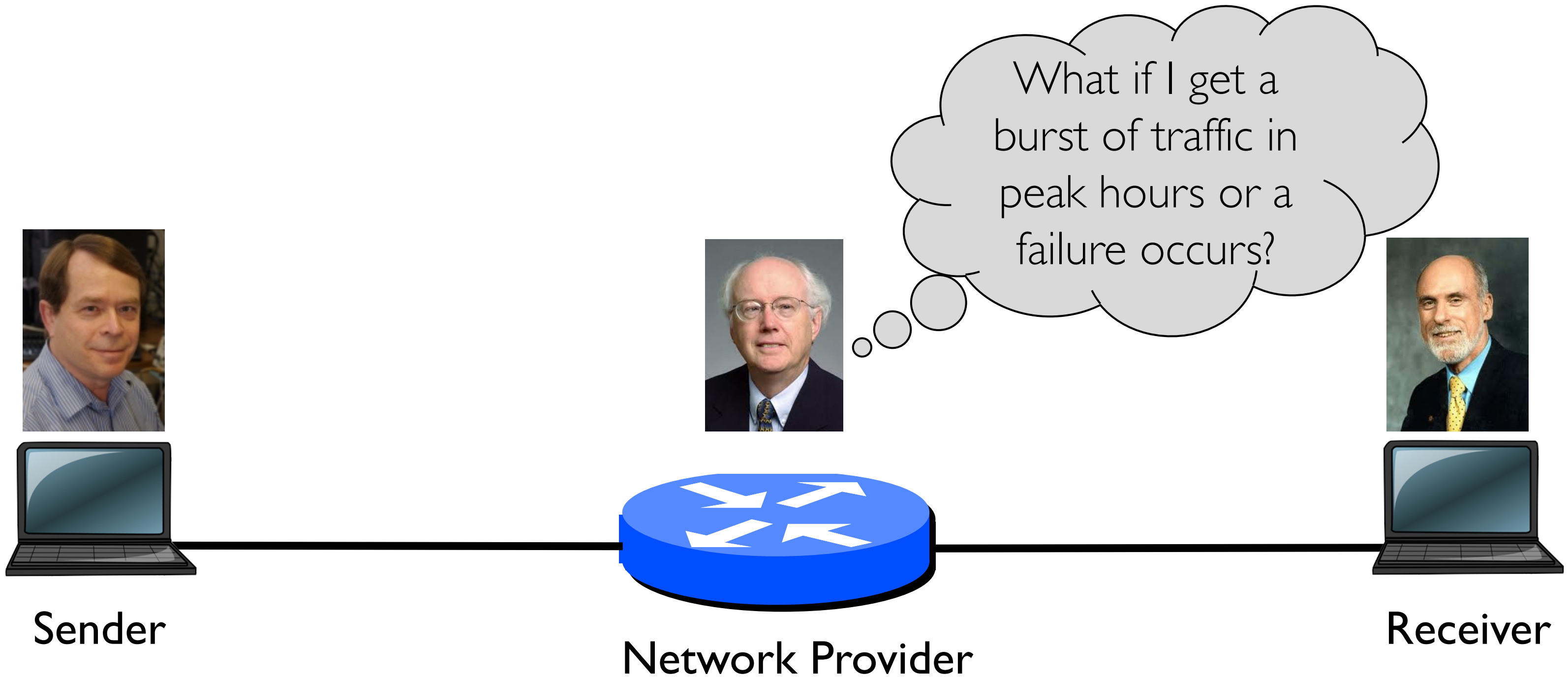
Send *additional* packets from the flow using low priority service (*WQoS*), filling up only the *spare capacity* in the network

- 40-80% Reduction in Flow Completion Time
- No harm to the regular high priority traffic
- Better use of Network Resources

Example Scenario

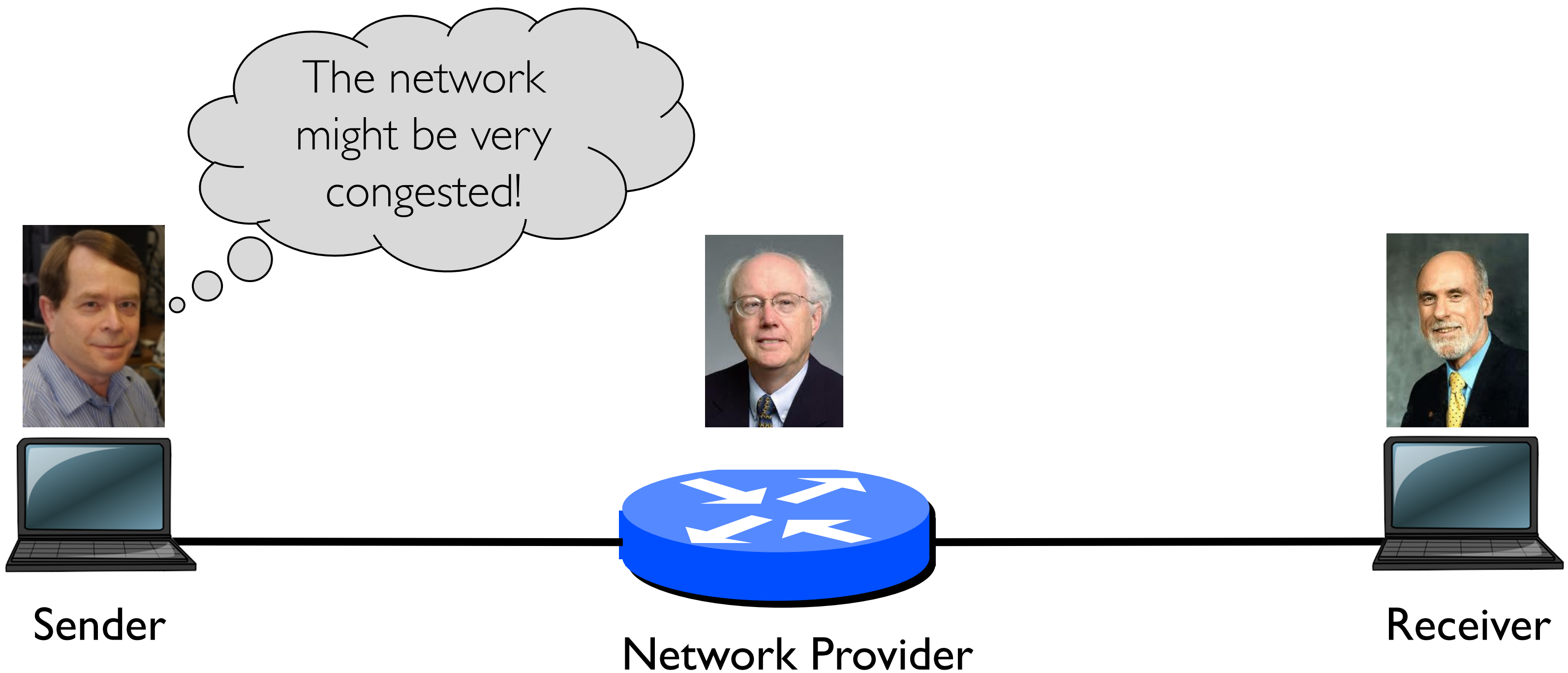


Network Provider Viewpoint



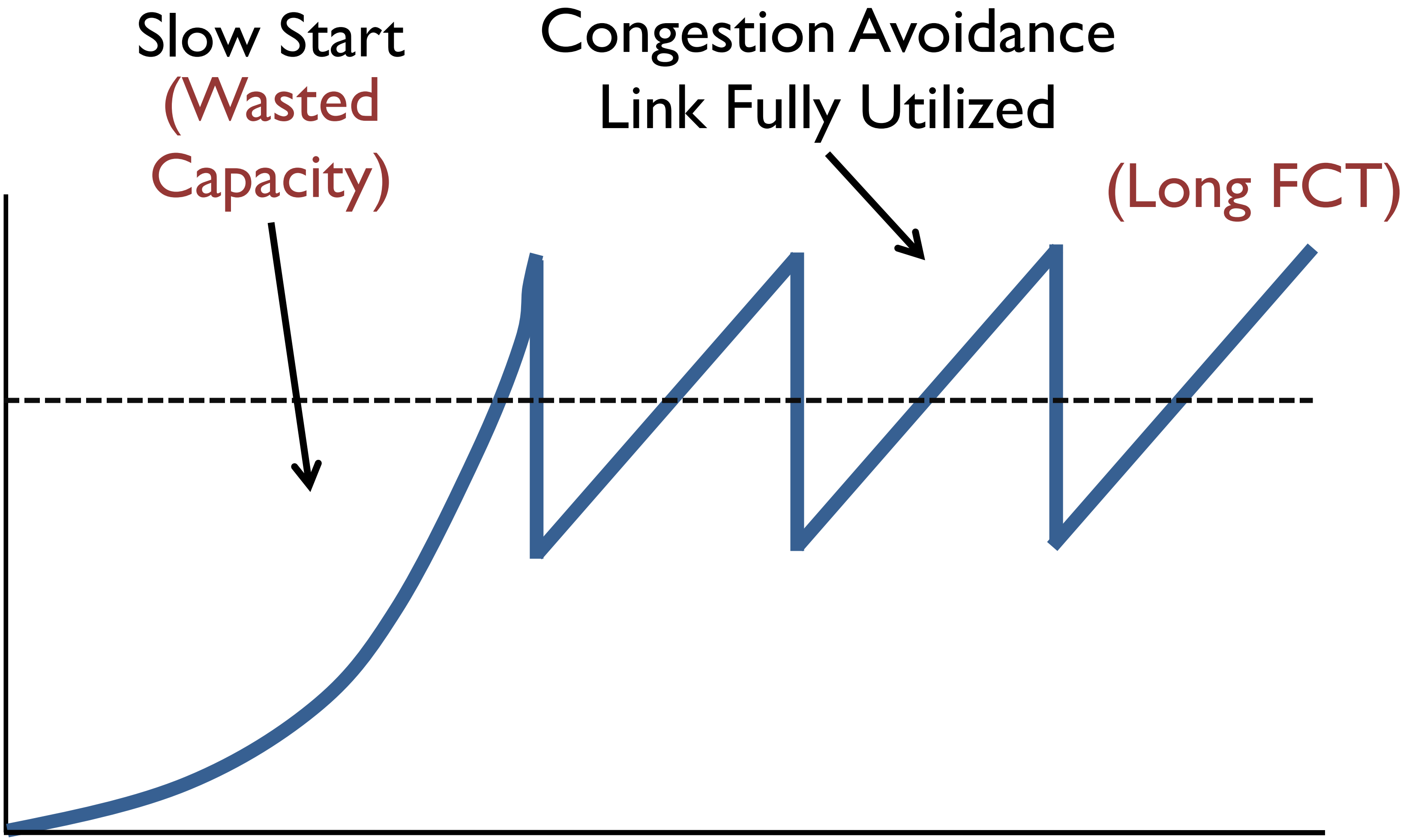
Must overprovision
30-50% average link utilization

Endhost Viewpoint



Must ramp-up cautiously

TCP


Slow Start
(Wasted Capacity)
Congestion Avoidance
Link Fully Utilized
(Long FCT)

The Root Cause

Two Goals of Congestion Control

- Fill the pipe for high throughput
- Do no harm to other flows

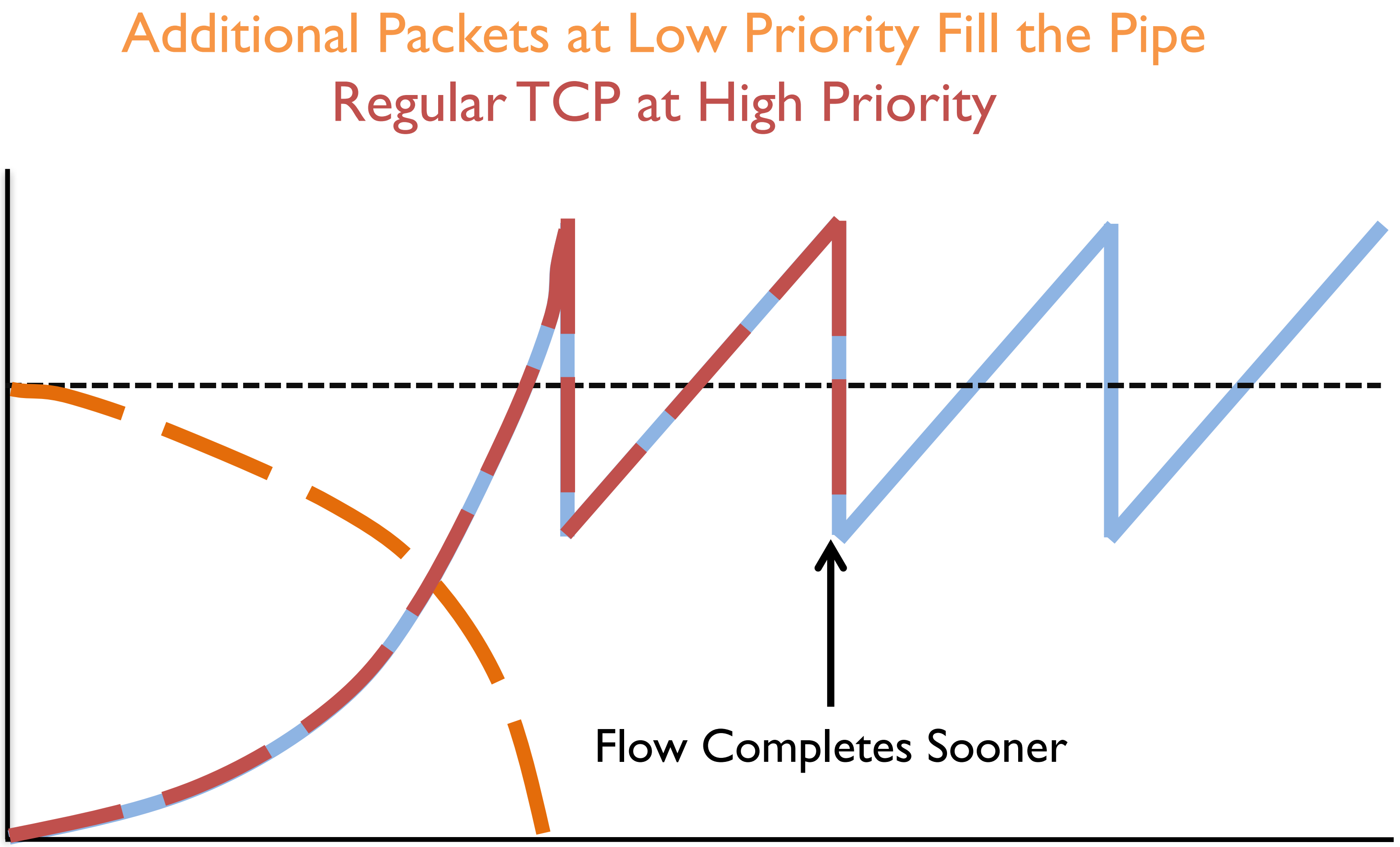
Traditional Approach

- Single mechanism tries to balance the two conflicting goals

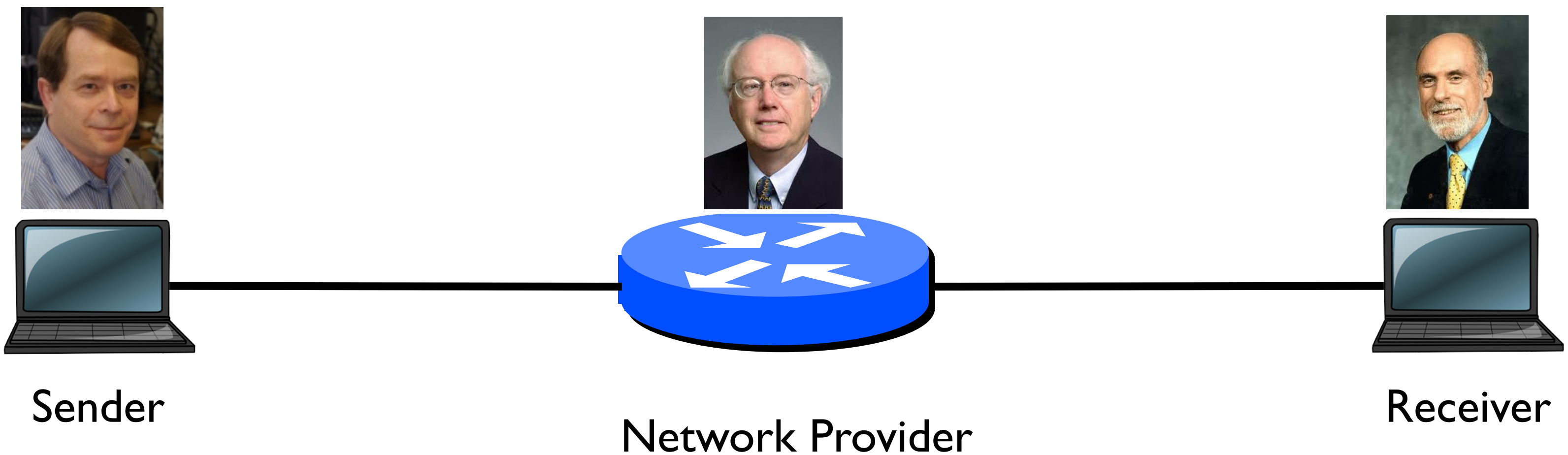
RC3: Decouple these goals using priorities

- Fill the pipe at lower priority
- Do no harm at higher priority

RC3 in action

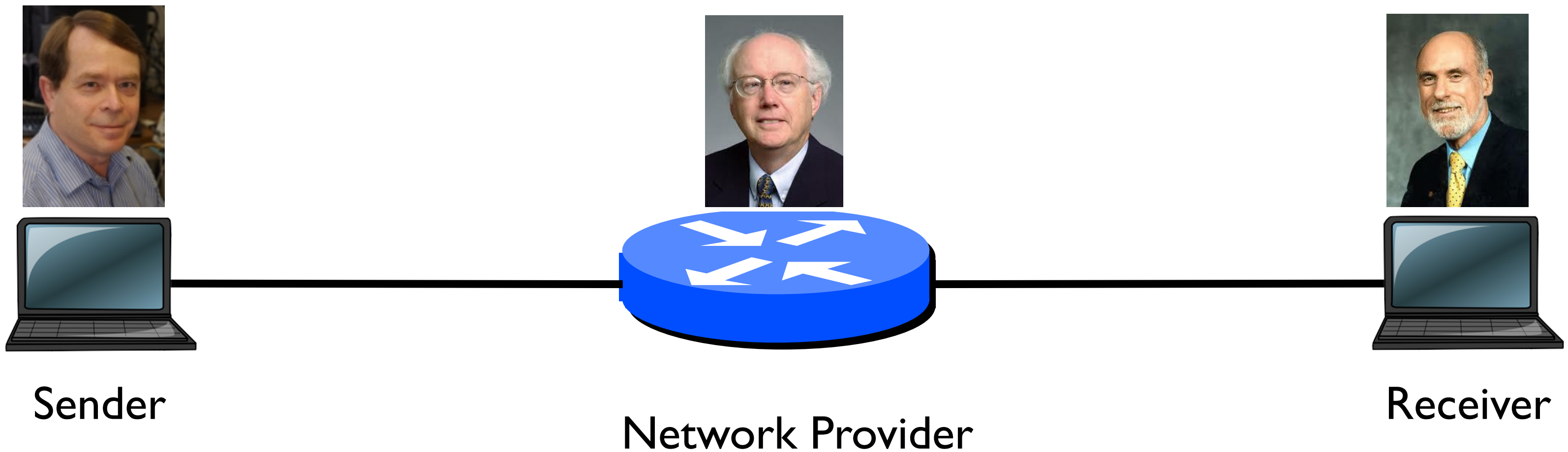


Example: FCT with Slow Start



7 packets flow (with initial congestion window of 1 segment) completes in 3RTTs under slow start

Example: FCT with RC3

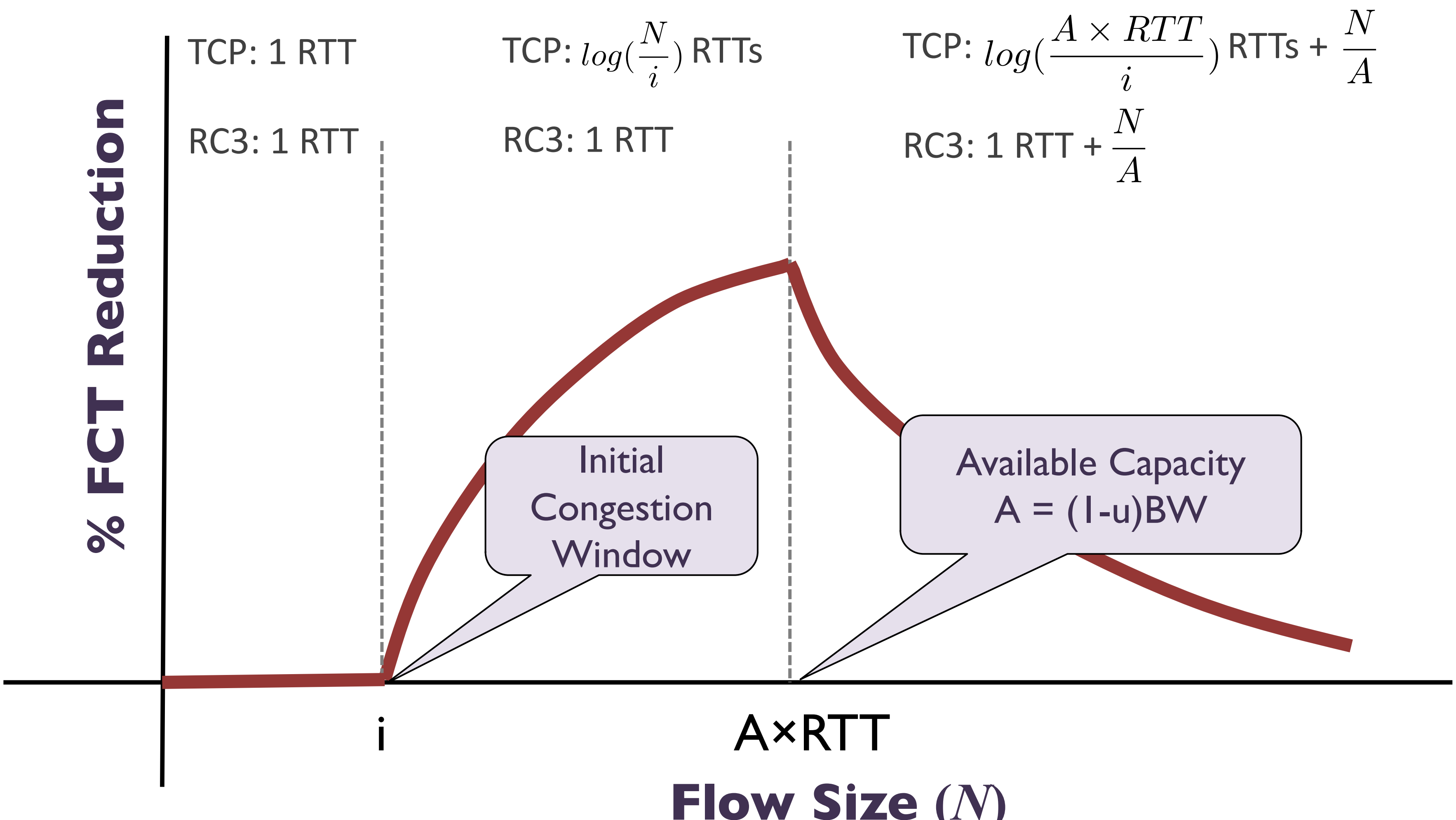


Remaining 6 packets sent at lower priority with the 1st packet
Flow completes in 1RTT

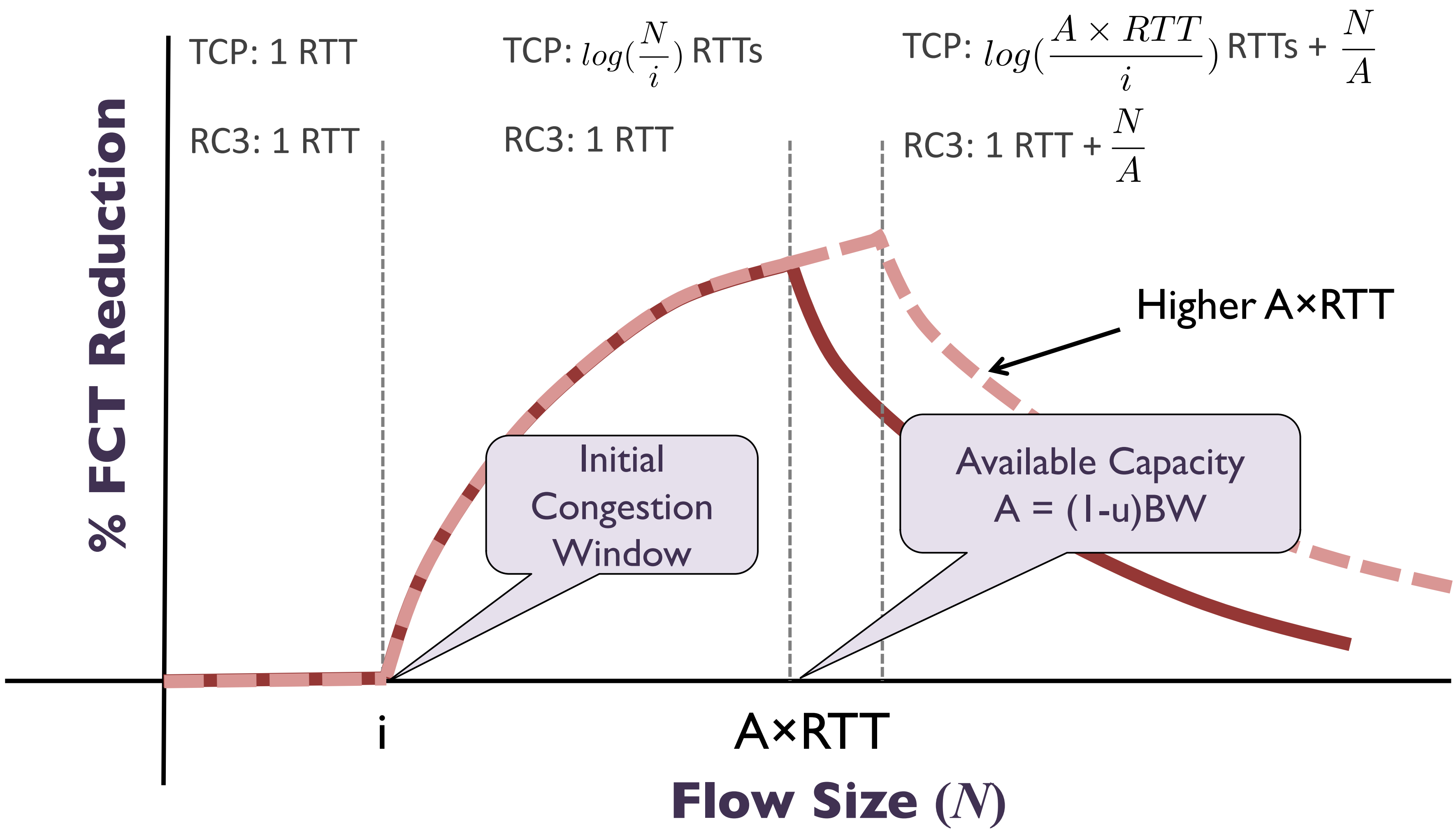
Roadmap

- *Isn't congestion control a solved problem?*
  - *Conflicting goals of high throughput and friendliness decoupled through priorities*
- **Scope for performance gains**
- *Design Details*
- *Simulation Results*
- *Linux Implementation and Evaluation*
- *Challenges and Future*

Theoretical Model



Parameter Sensitivity: AxRTT



Roadmap

- *Isn't congestion control a solved problem?*
  - *Conflicting goals of high throughput and friendliness decoupled through priorities*
- *Scope for performance gains*
  - *Increases with increasing RTTxBW*
- **Design Details**
- *Simulation Results*
- *Linux Implementation and Evaluation*
- *Challenges and Future*

WQoS Implementation

Routers offer several layers of worse service

- Use Priority Queues
- Support already present

Packets carry priority (possibly) in DSCP field

- Priority 0 – default (highest)
- Priority 1, Priority 2, Priority 3,….

RC3 Design

RC3 runs two parallel control loops

- TCP control loop

  Transmits packets that obey unmodified TCP logic at highest priority

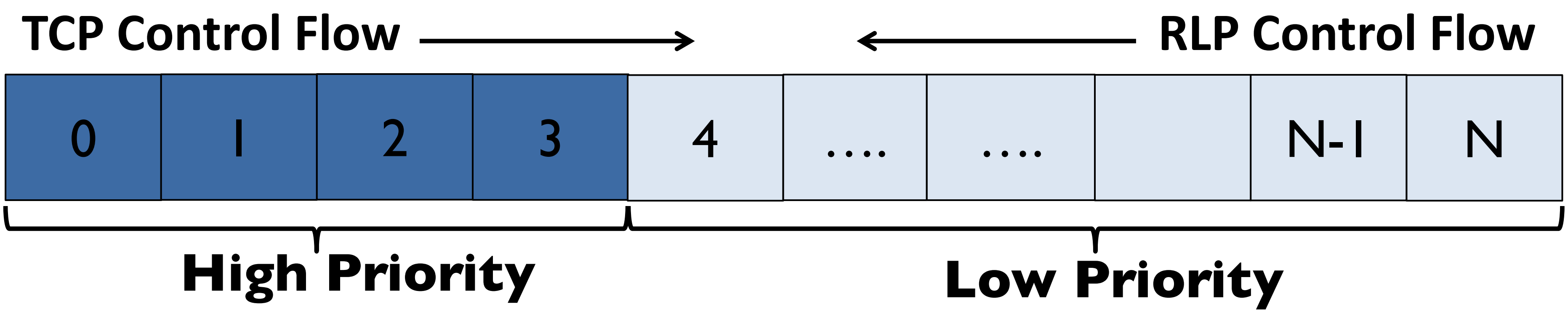
- Recursive Low Priority (RLP) control loop

  Transmits additional packets at low priority

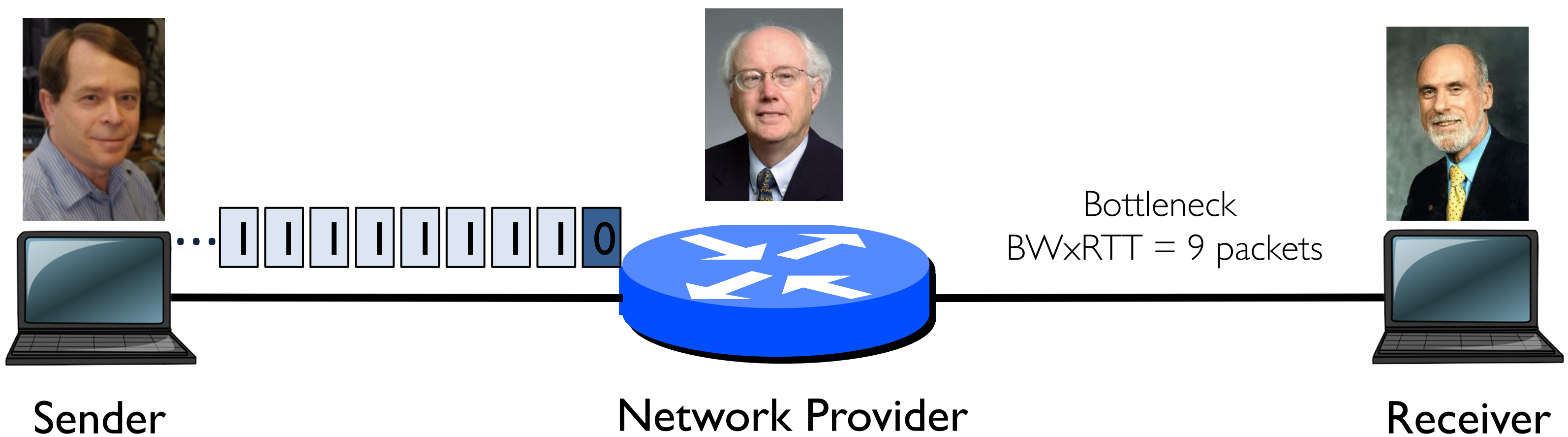
What packets are sent at low priority?

Minimum overlap between packets sent by the two control loops for maximum gains

- RLP starts from the last packet in buffer
- Goes in reverse order

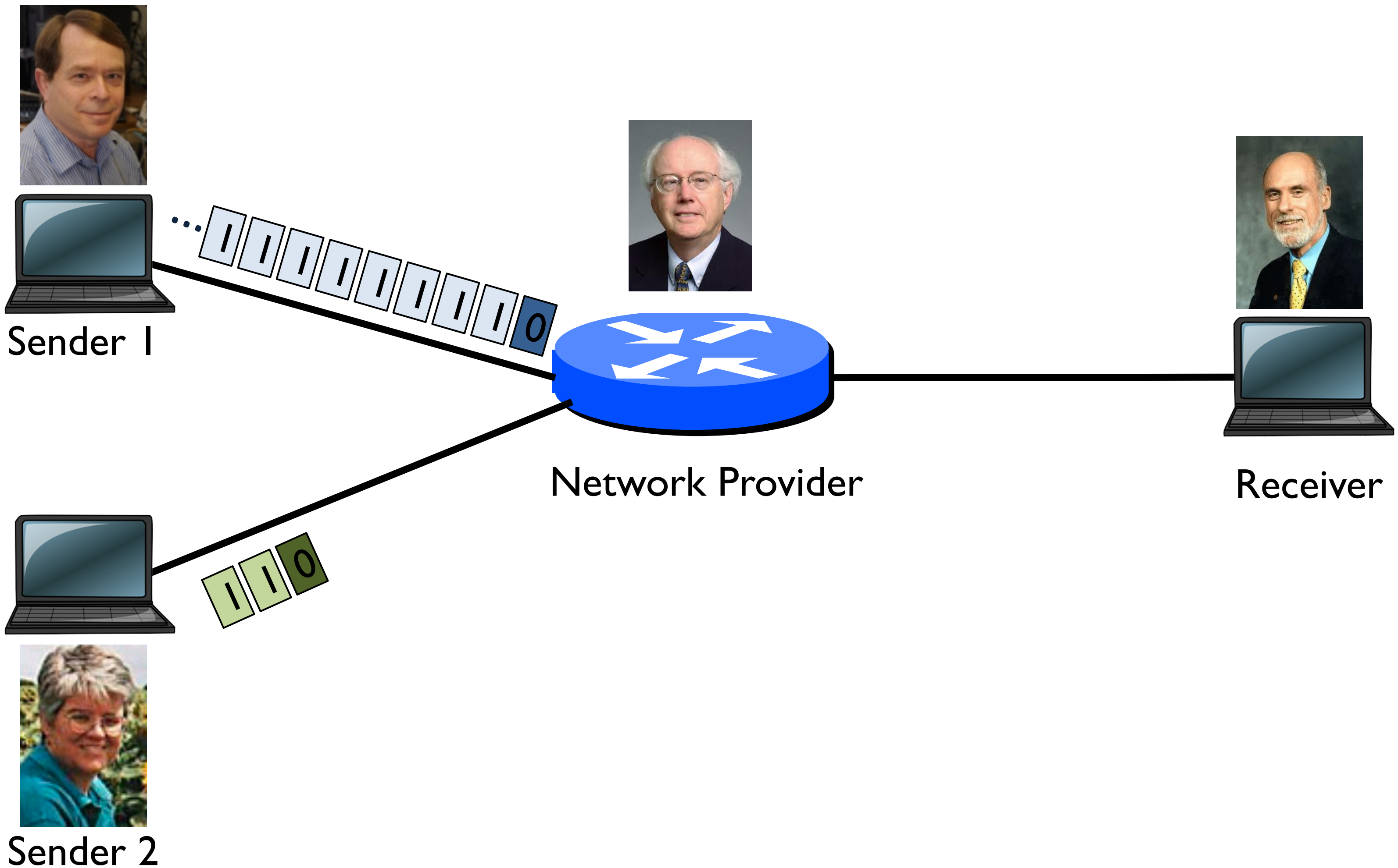


Single Flow

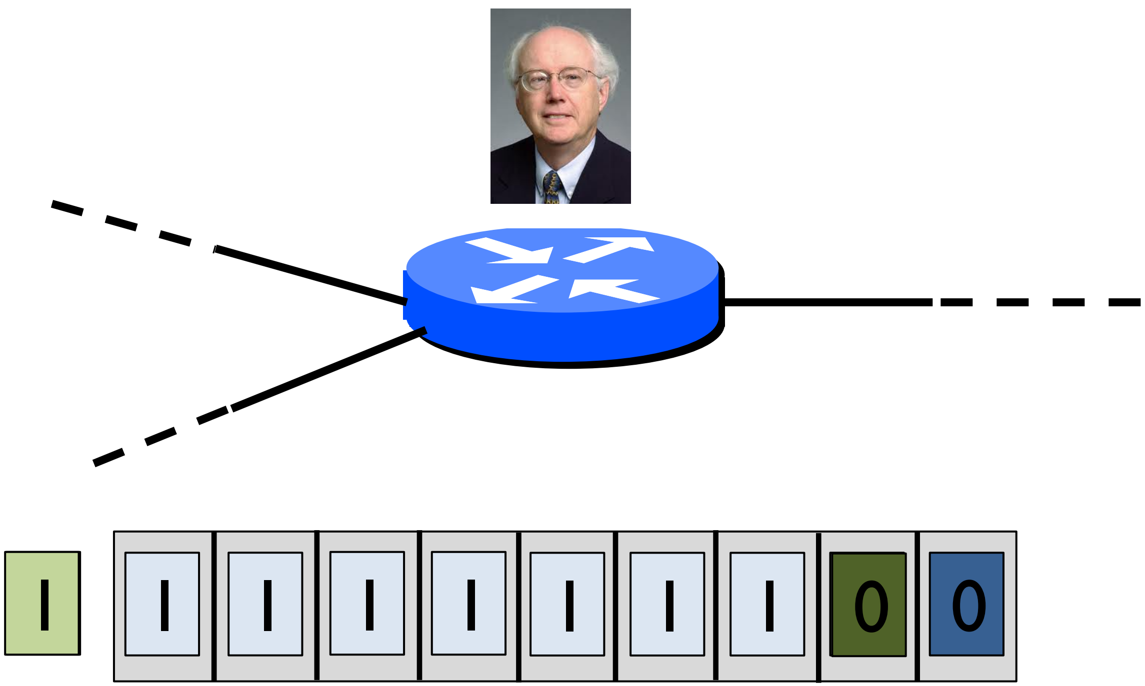


...
1
1
1
1
1
1
1
1
0
Bottleneck
BWxRTT = 9 packets
Sender
Network Provider
Receiver

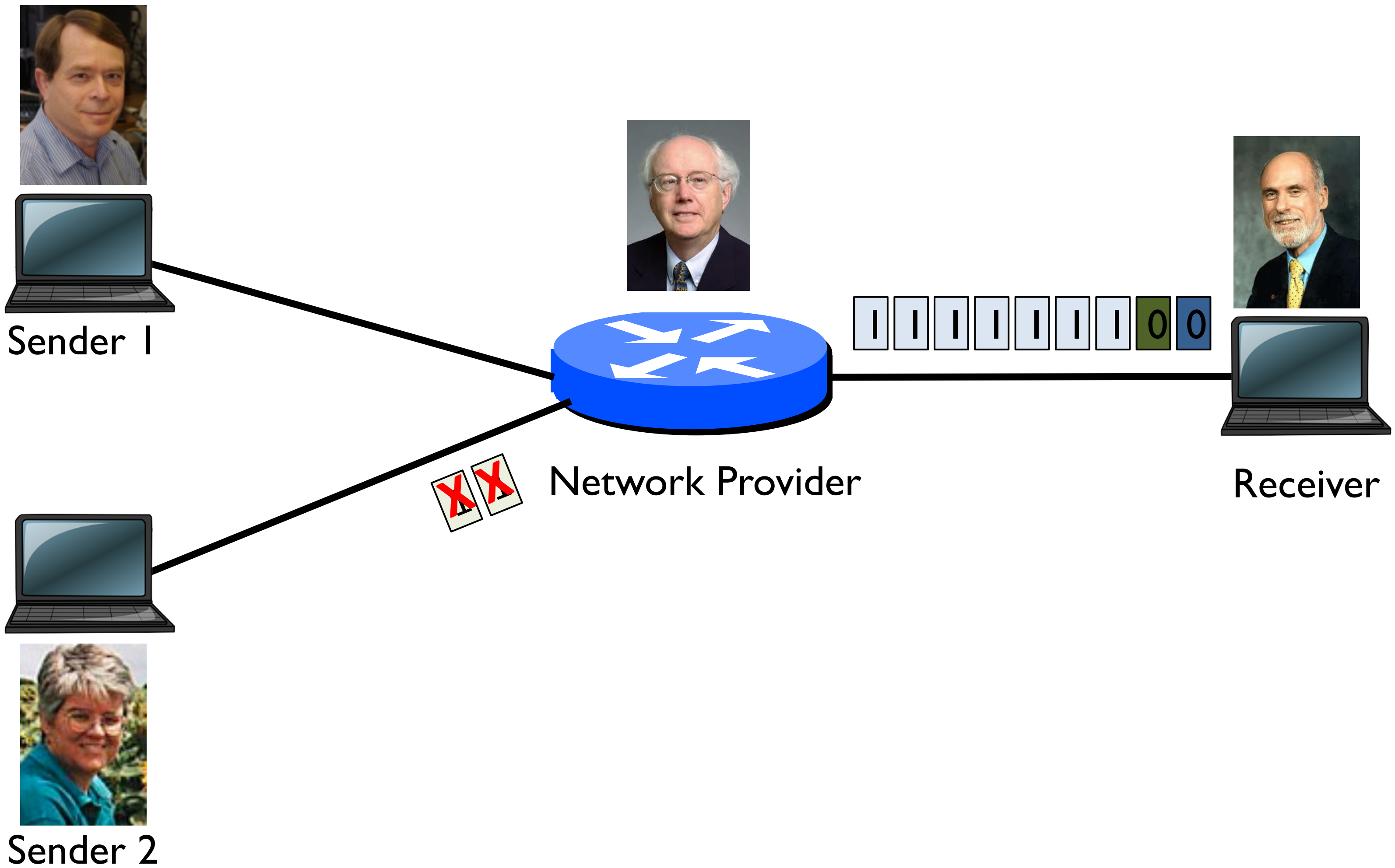
Multiple Flows?



Router's Priority Queue

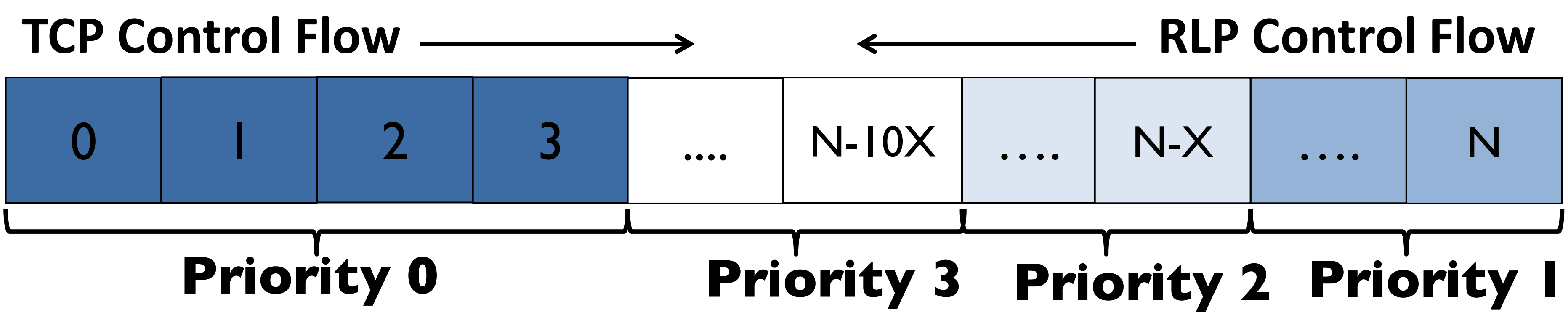


Multiple Flows?



Recursively Cautious Congestion Control

- Use multiple priority levels
- Send exponentially larger number of packets at each priority level



RC3 Design: Quick Recap

Two parallel control loops

- Regular TCP
- Recursive Low Priority (RLP)

Minimum overlap between the two control loops

- Send low priority packets from the end in reverse order

Max-min fairness across flows

- Use multiple priority levels

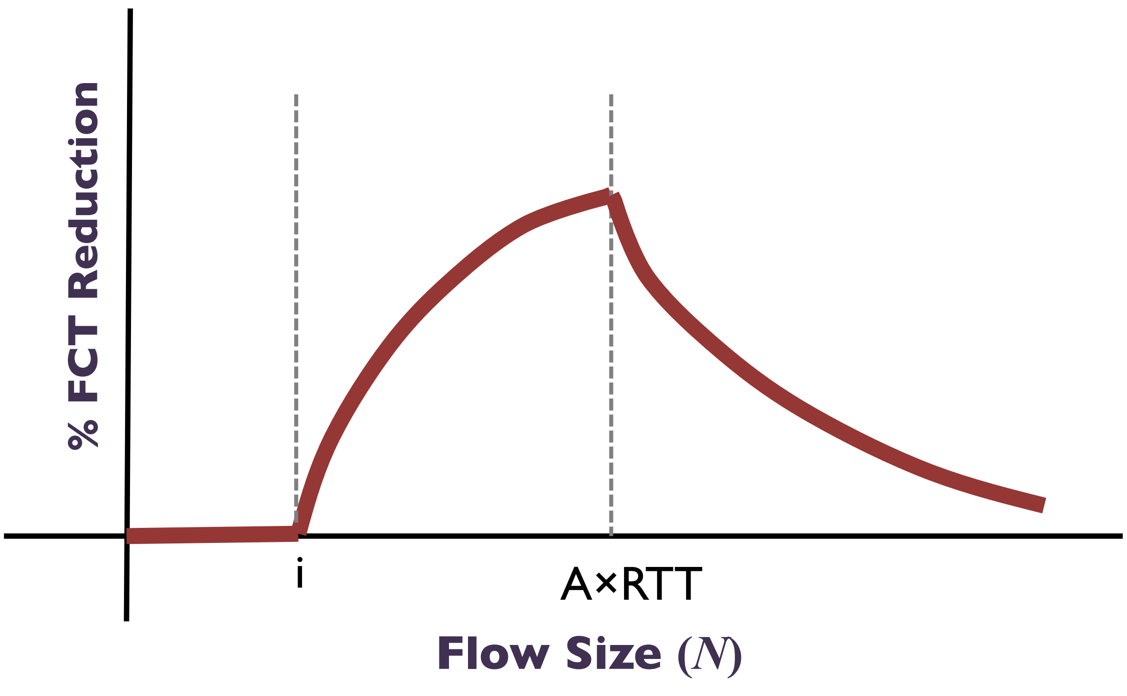
Roadmap

- *Isn't congestion control a solved problem?*
  - *Conflicting goals of high throughput and friendliness decoupled through priorities*
- *Scope for performance gains*
  - *Increases with increasing RTTxBW*
- *Design Details*
  - *Additional packets sent backwards from the end using multiple low priority levels*
- **Simulation Results**
- *Linux Implementation and Evaluation*
- *Challenges and Future*

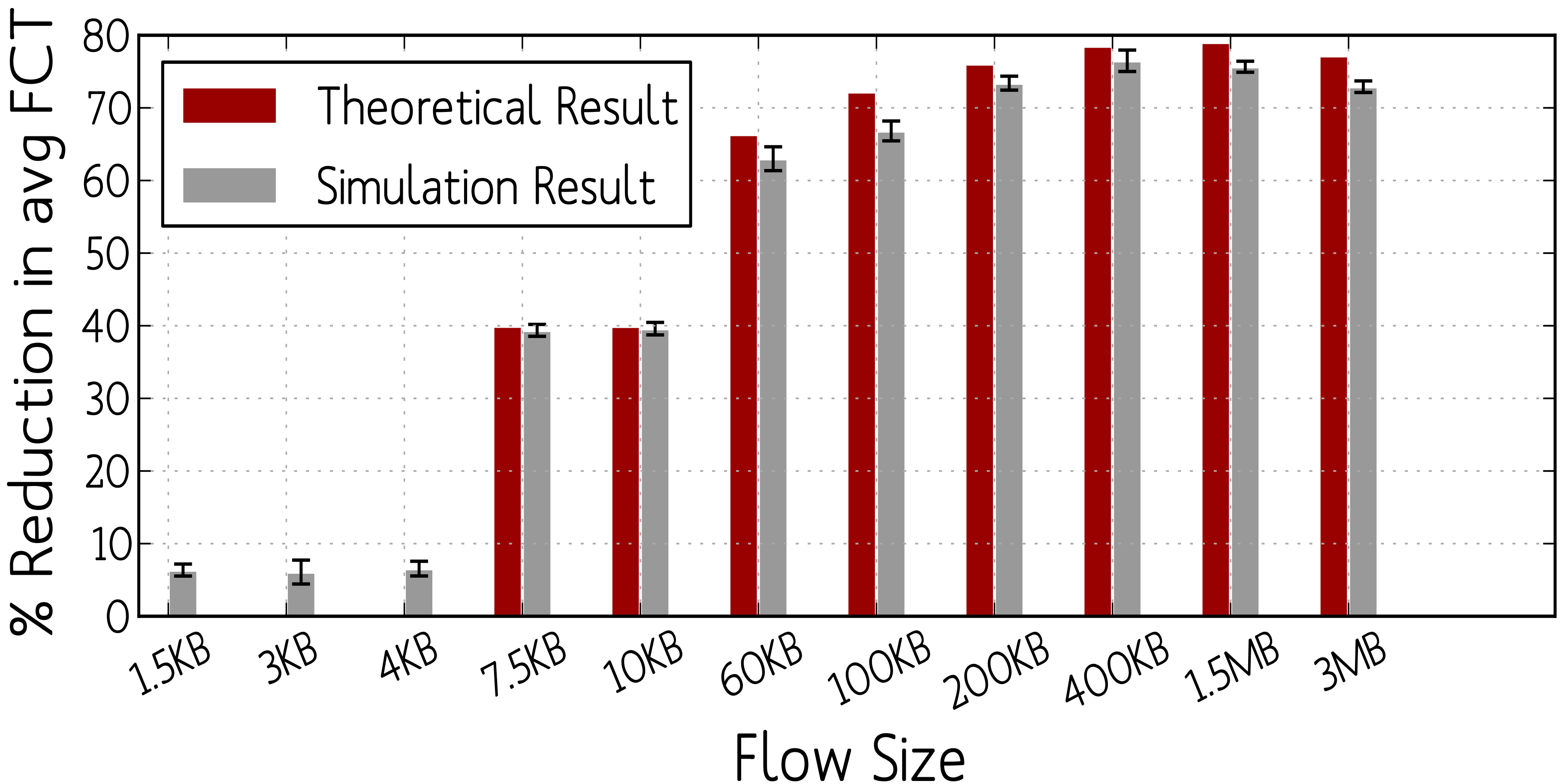
Simulation Setup

- Multi-hop **Internet-2** network topology
  - 10 core nodes, 100 end hosts
- **1Gbps** bottleneck bandwidth
- **40ms** average RTT
- Baseline is **30%** average link utilization
- Pareto flow size distribution with Poisson inter-arrival
- Initial Congestion Window of **4 segments**

Comparing baseline simulation results with the theoretical model

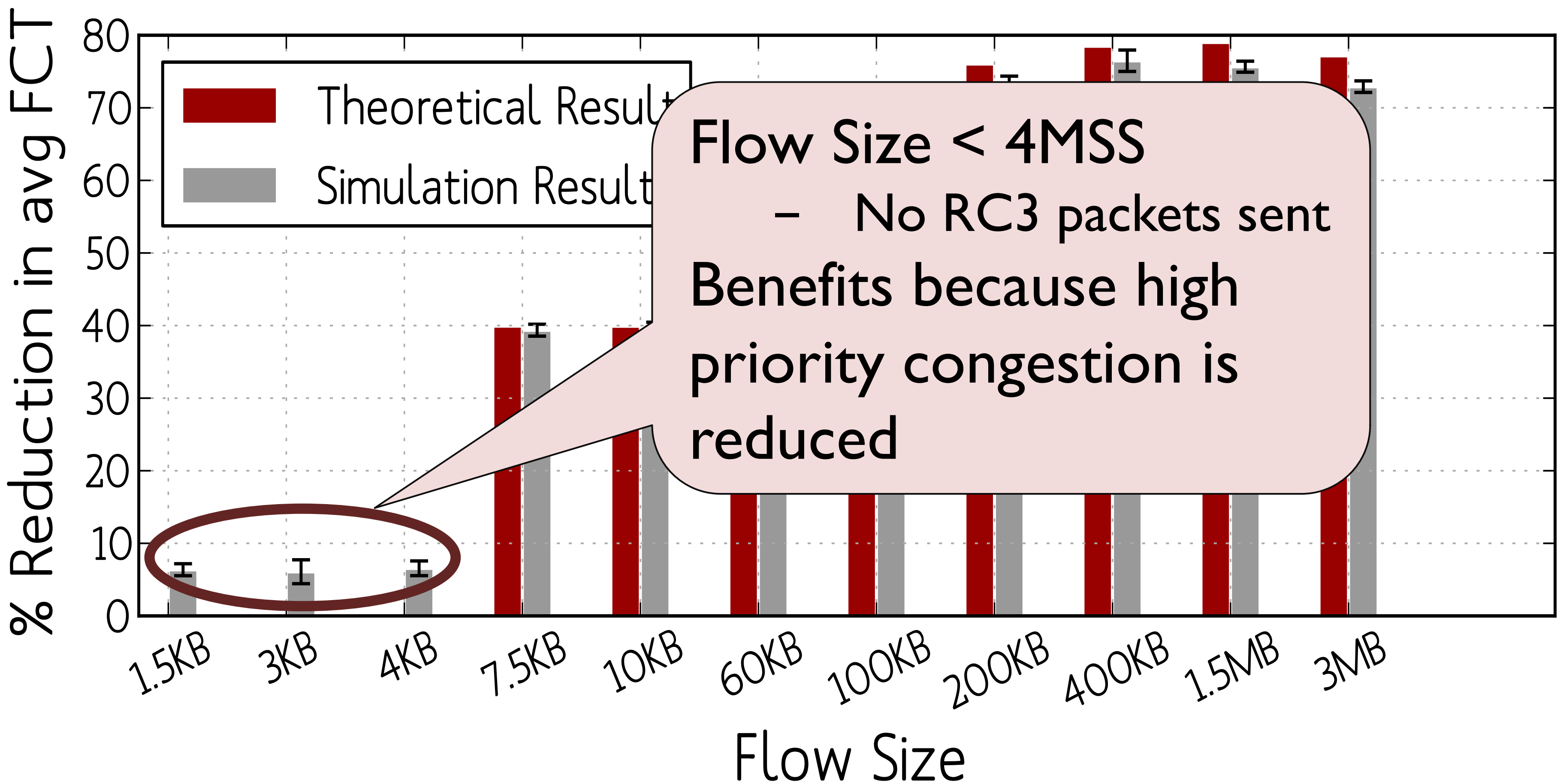


Baseline

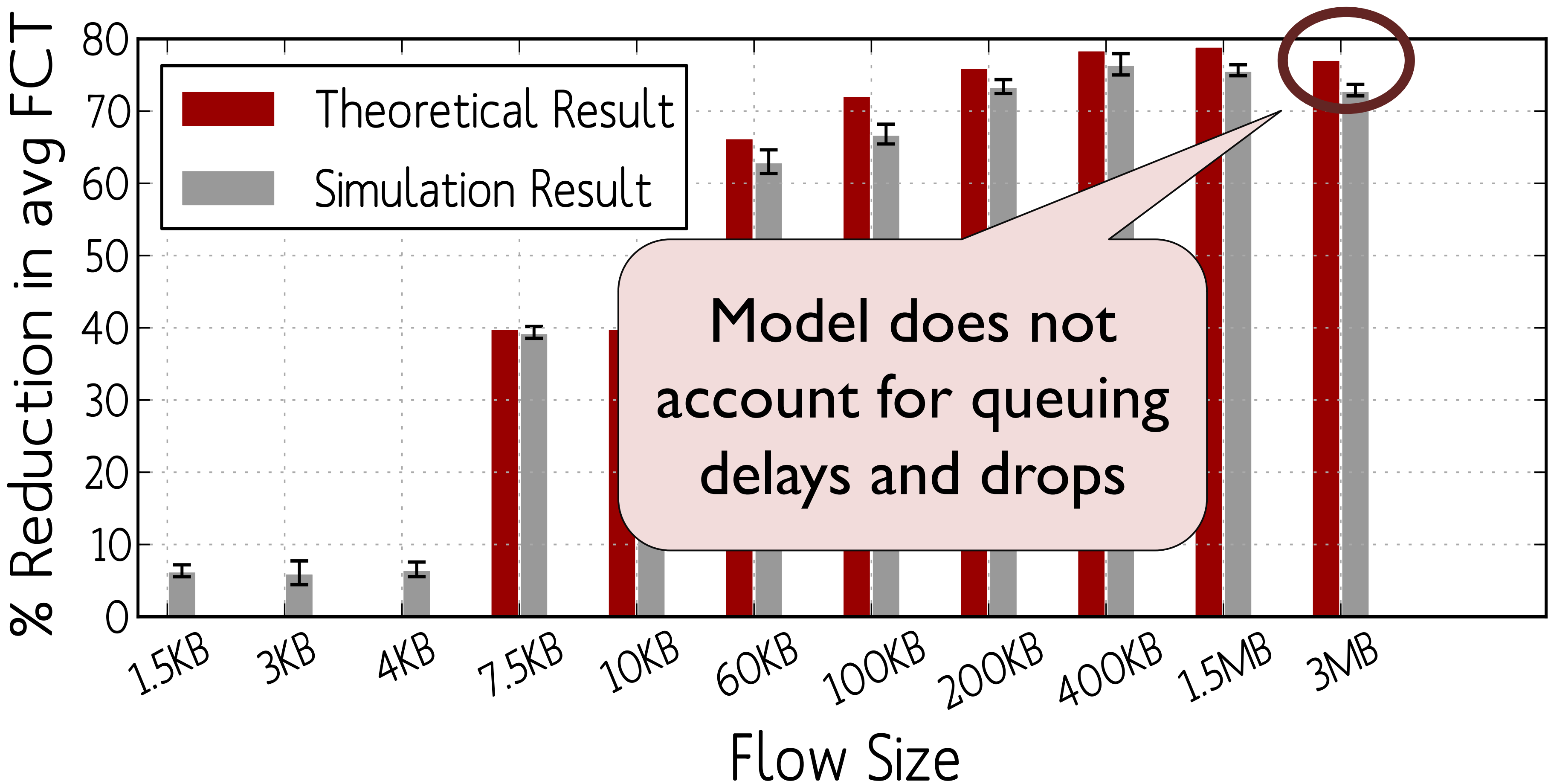


% Reduction in avg FCT
80
70
60
50
40
30
20
10
0
Theoretical Result
Simulation Result
1.5KB
3KB
4KB
7.5KB
10KB
60KB
100KB
200KB
400KB
1.5MB
3MB
Flow Size

Baseline

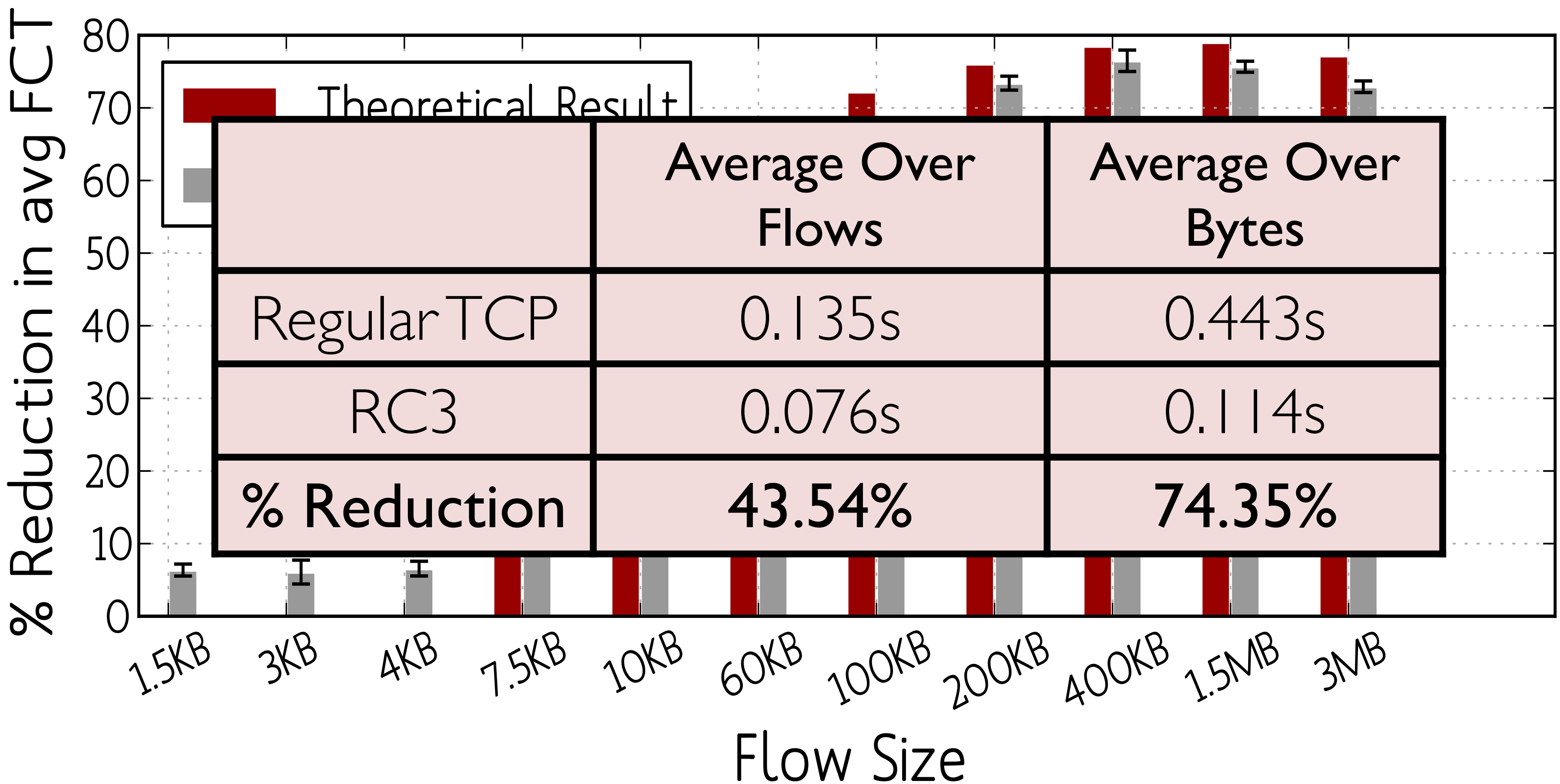

% Reduction in avg FCT
0
10
20
30
40
50
60
70
80
Theoretical Result
Simulation Result
Flow Size < 4MSS
– No RC3 packets sent
Benefits because high priority congestion is reduced
1.5KB
3KB
4KB
7.5KB
10KB
60KB
100KB
200KB
400KB
1.5MB
3MB
Flow Size

Baseline



% Reduction in avg FCT
0
10
20
30
40
50
60
70
80
Theoretical Result
Simulation Result
Model does not account for queuing delays and drops
1.5KB
3KB
4KB
7.5KB
10KB
60KB
100KB
200KB
400KB
1.5MB
3MB
Flow Size

Baseline



	Average Over Flows	Average Over Bytes
Regular TCP	0.135s	0.443s
RC3	0.076s	0.114s
% Reduction	**43.54%**	**74.35%**

Comparing RC3 with other schemes

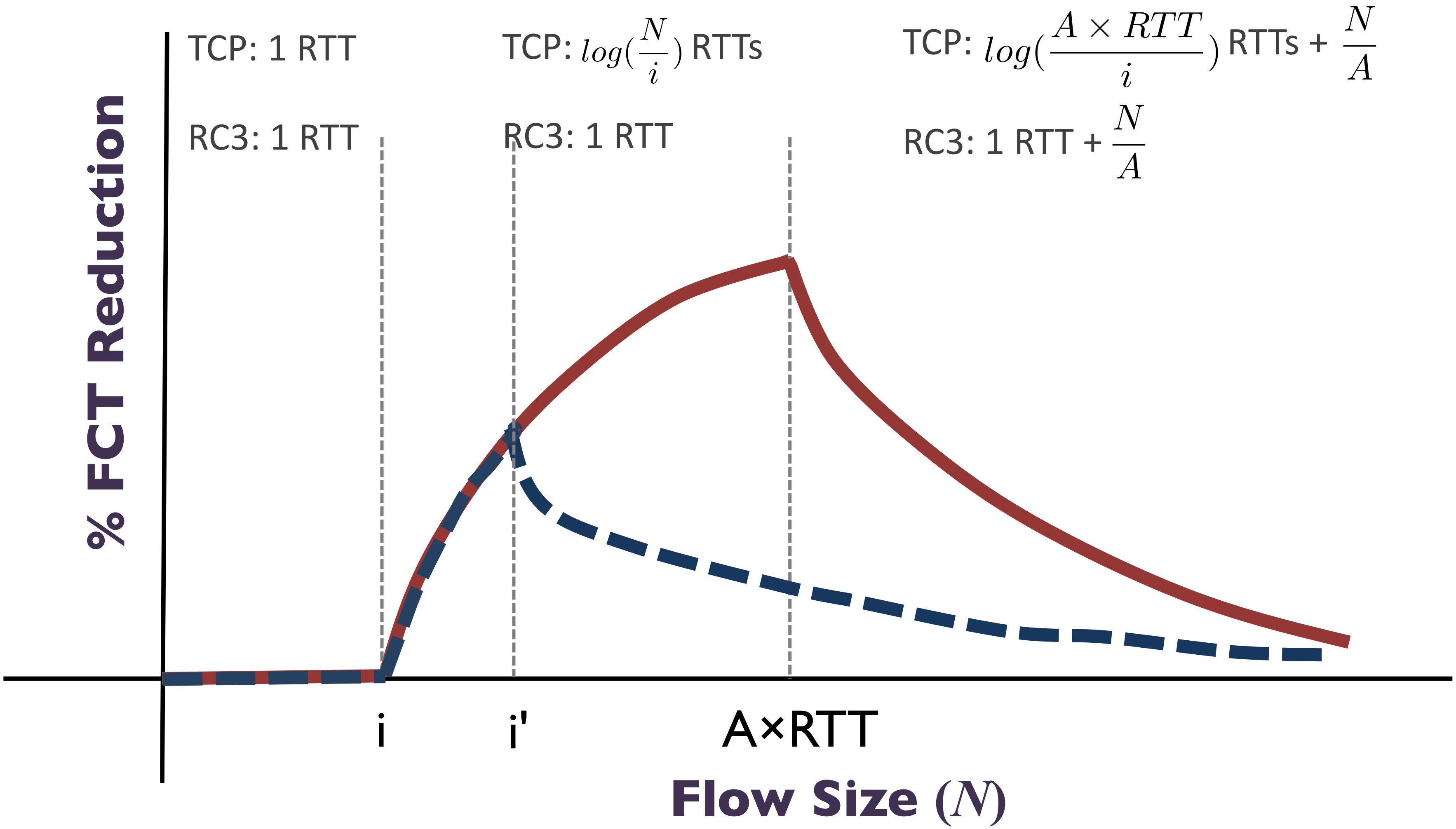
RC3 in comparison

- Increasing the initial congestion window
- Rate Control Protocol (RCP)

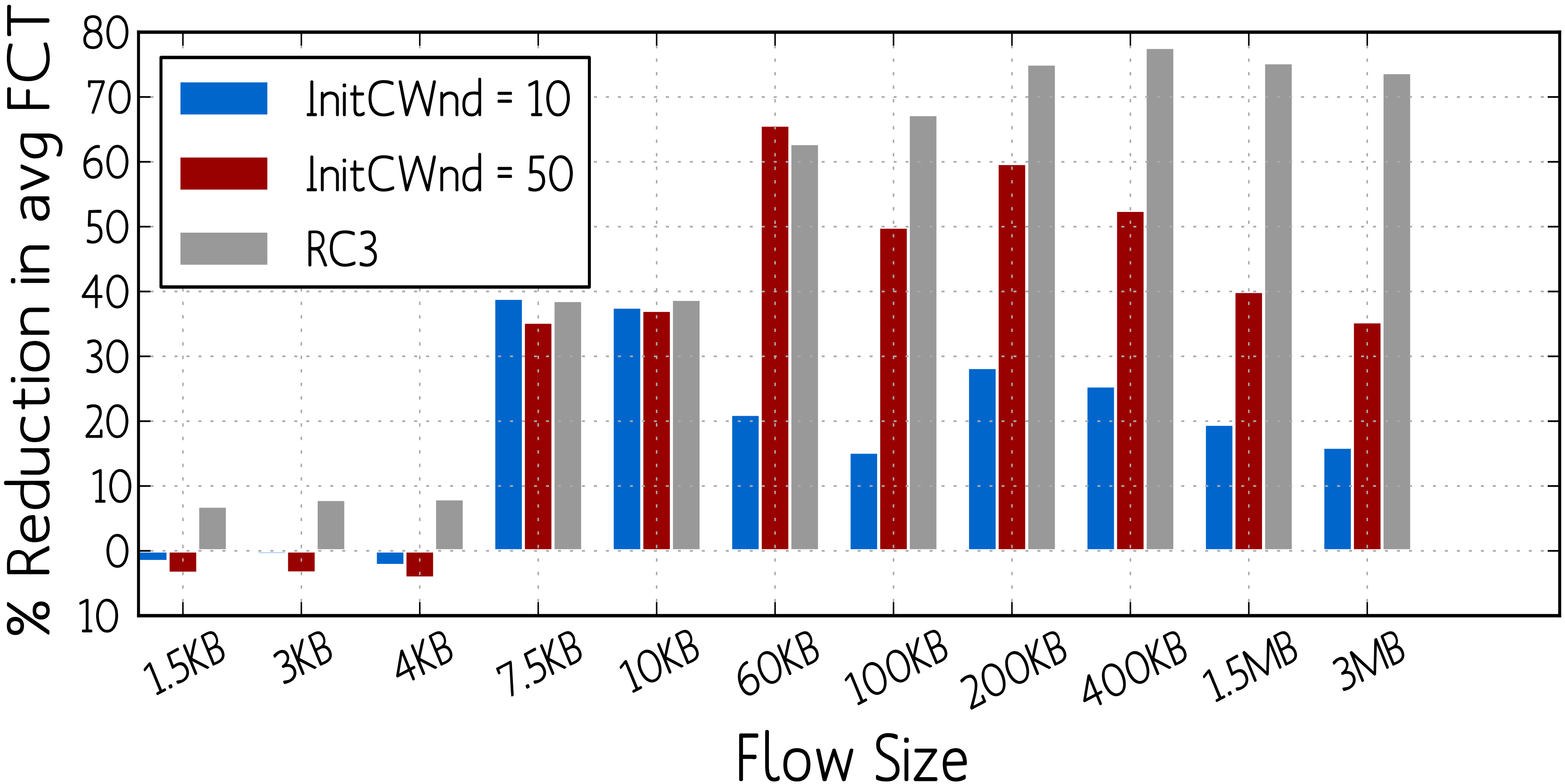
RC3 in comparison

- Increasing the initial congestion window
- Rate Control Protocol (RCP)

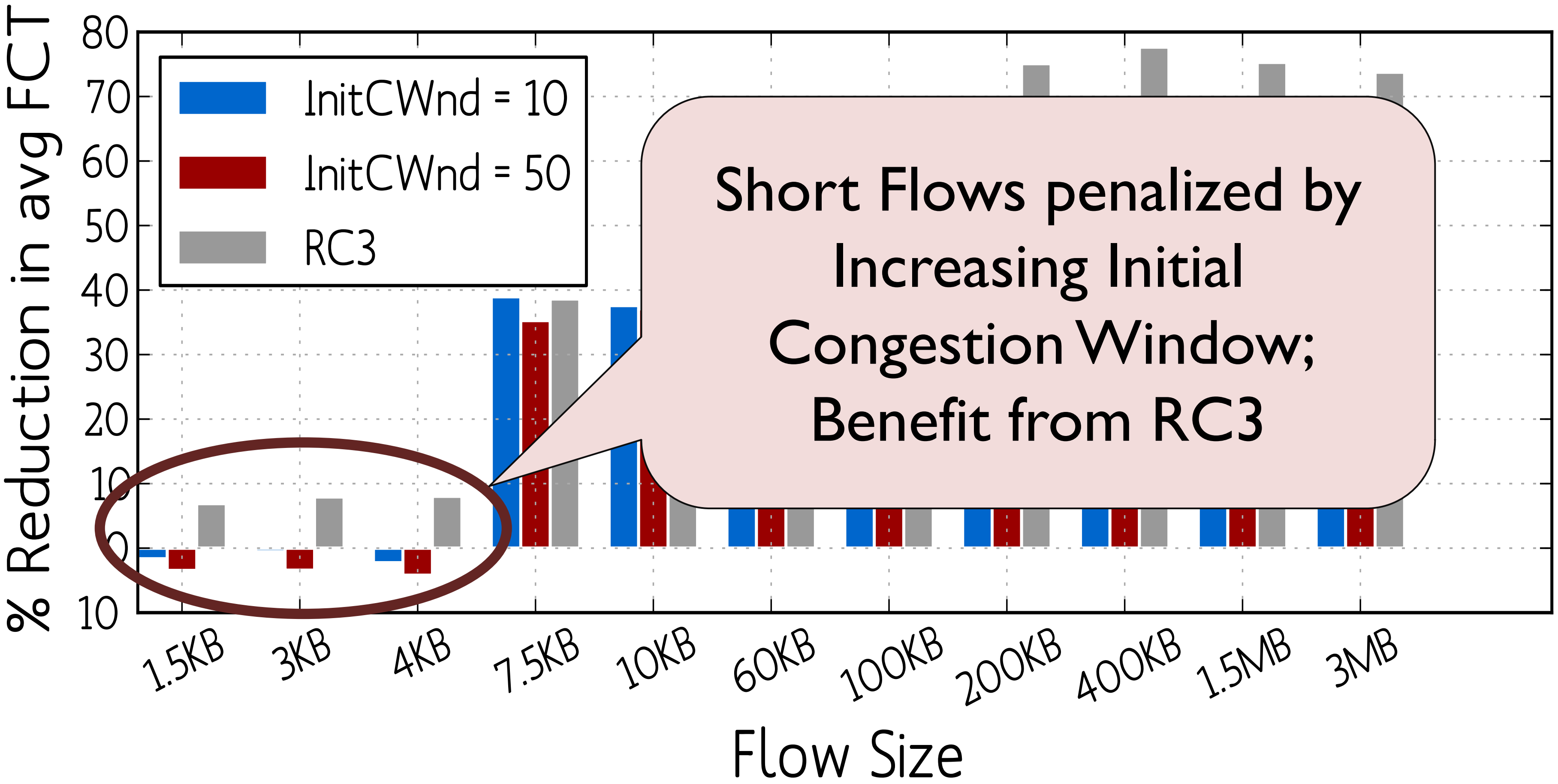
Comparison: Increasing InitCWnd



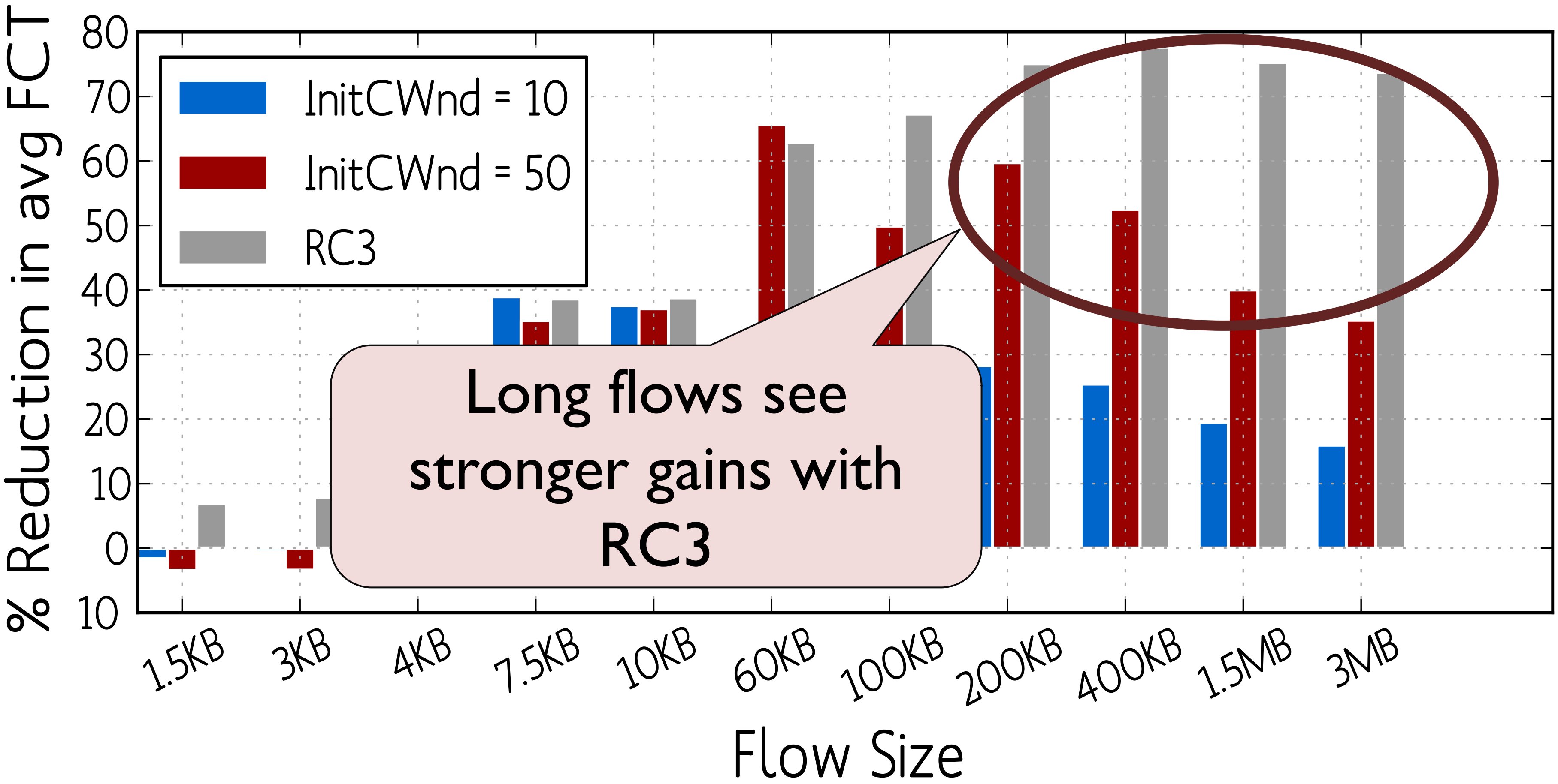
Comparison: Increasing InitCwnd



Comparison: Increasing InitCwnd



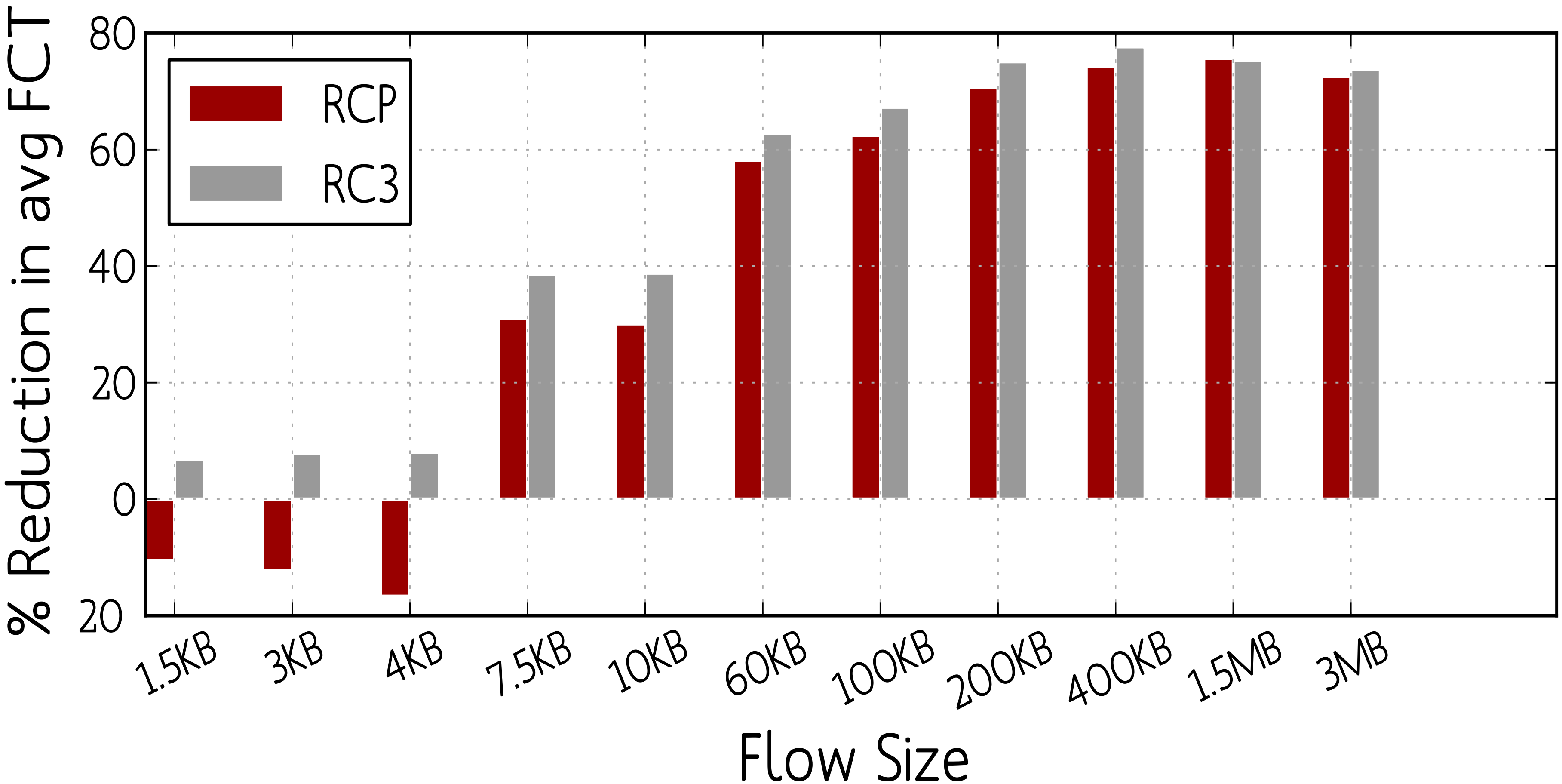
Comparison: Increasing InitCwnd



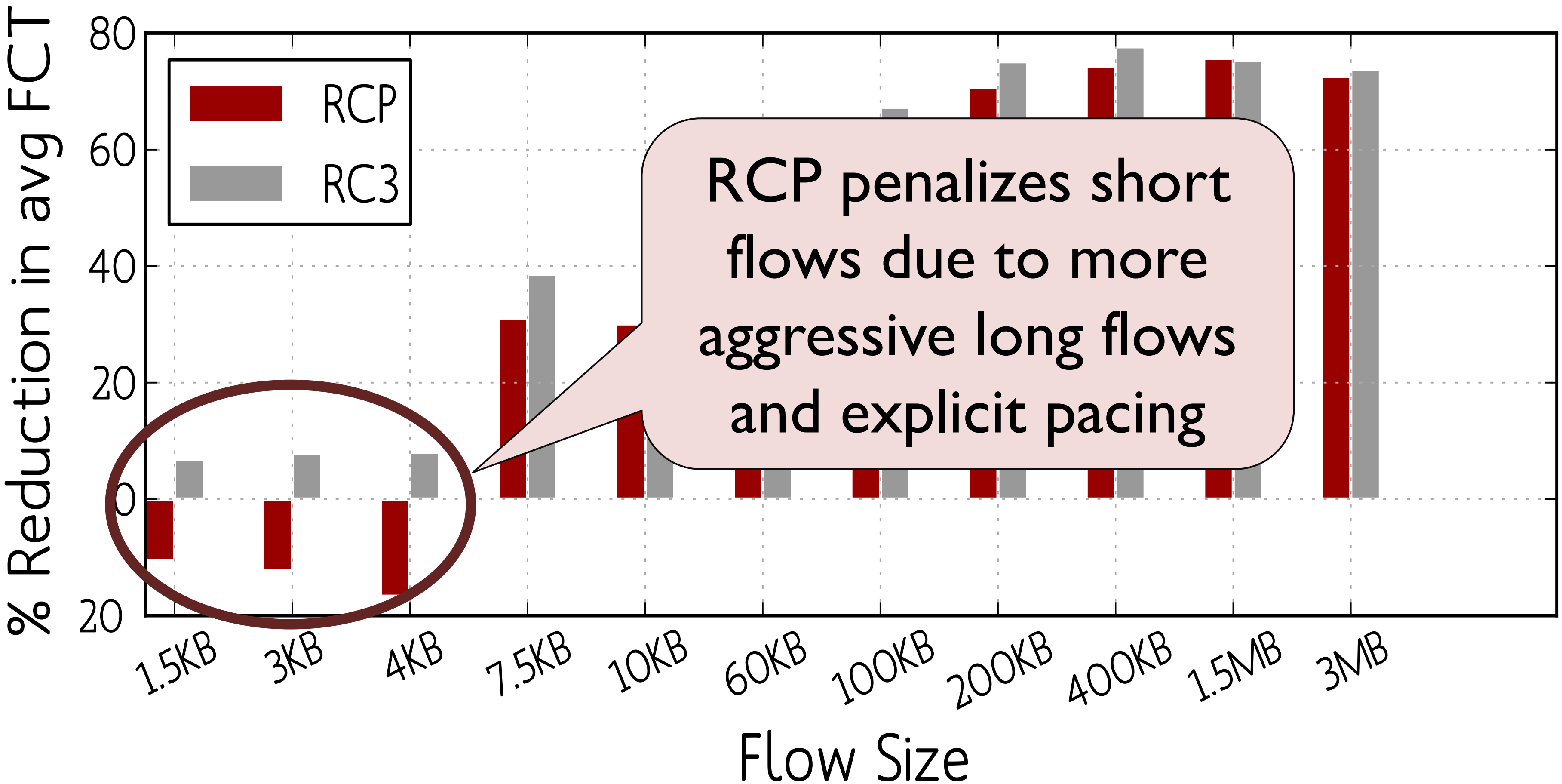
RC3 in comparison

- Increasing the initial congestion window
- Rate Control Protocol (RCP)

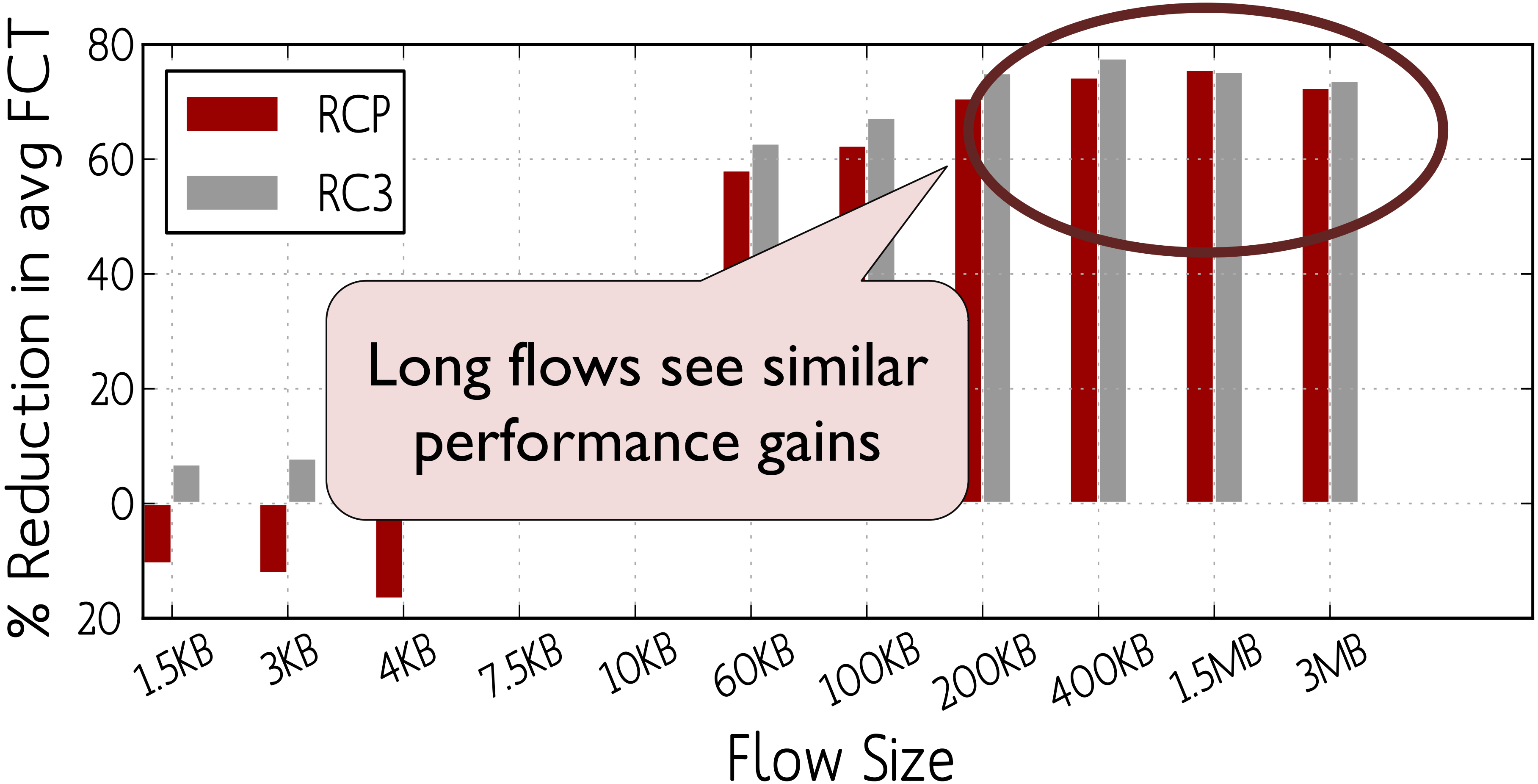
Comparison: RCP



Comparison: RCP



Comparison: RCP



Stress Testing RC3

- Varying Link Utilization
- Varying RTTxBW
- More Topologies
- Different Workload
- Link Heterogeneity
- Random Losses
- Varying Priority Assignments
- Application Pacing
- Comparison with traditional QoS

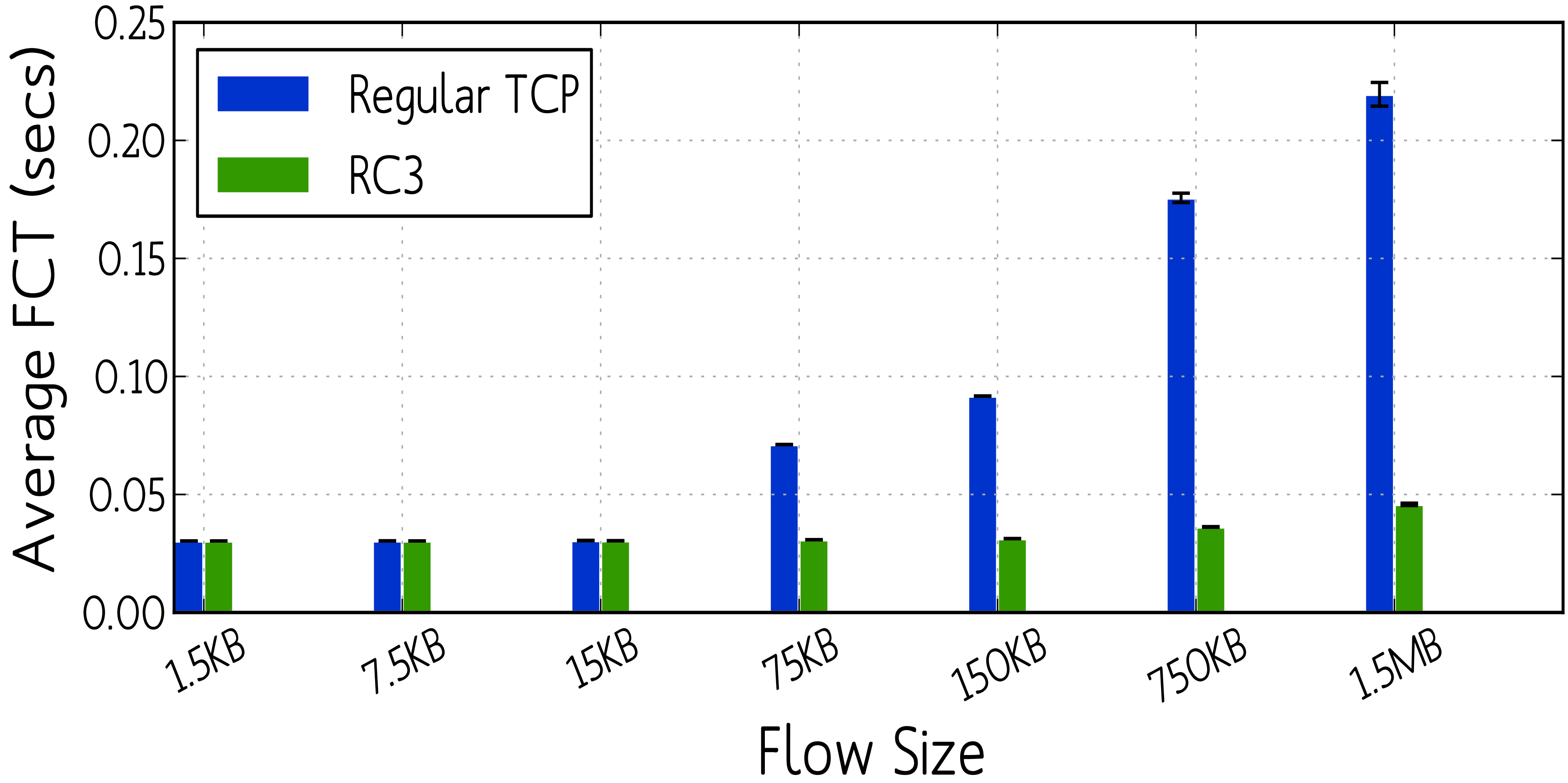
Roadmap

- *Isn't congestion control a solved problem?*
  - *Conflicting goals of high throughput and friendliness decoupled through priorities*
- *Scope for performance gains*
  - *Increases with increasing RTTxBW*
- *Design Details*
  - *Additional packets sent backwards from the end using multiple low priority levels*
- *Simulation Results*
  - *40-80% reduction in FCT over baseline TCP implementation*
- **Linux Implementation and Evaluation**
- *Challenges and Future*

RC3 in Implementation

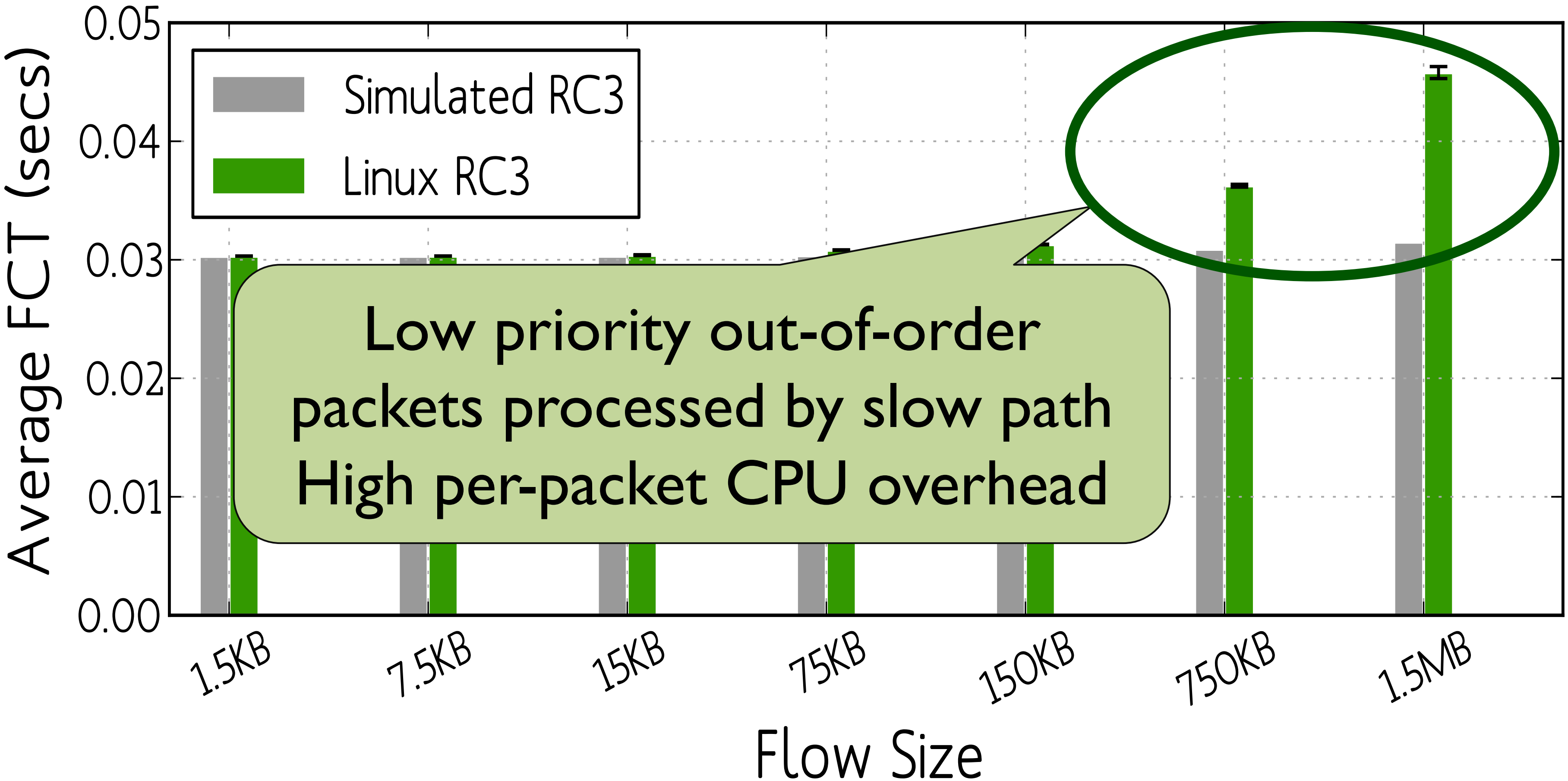
- Implemented in Linux 3.2 kernel
- 121 additional LOC
  - Sending Data Packets: 74 LOC
  - Receiving Data Packets and Acks: 47 LOC
- Agnostic to the underlying TCP algorithm
  - Can be Tahoe, Reno, NewReno, BIC, CUBIC etc

Evaluation



Average FCT (secs)
0.00
0.05
0.10
0.15
0.20
0.25
Regular TCP
RC3
1.5KB
7.5KB
15KB
75KB
150KB
750KB
1.5MB
Flow Size

Evaluation

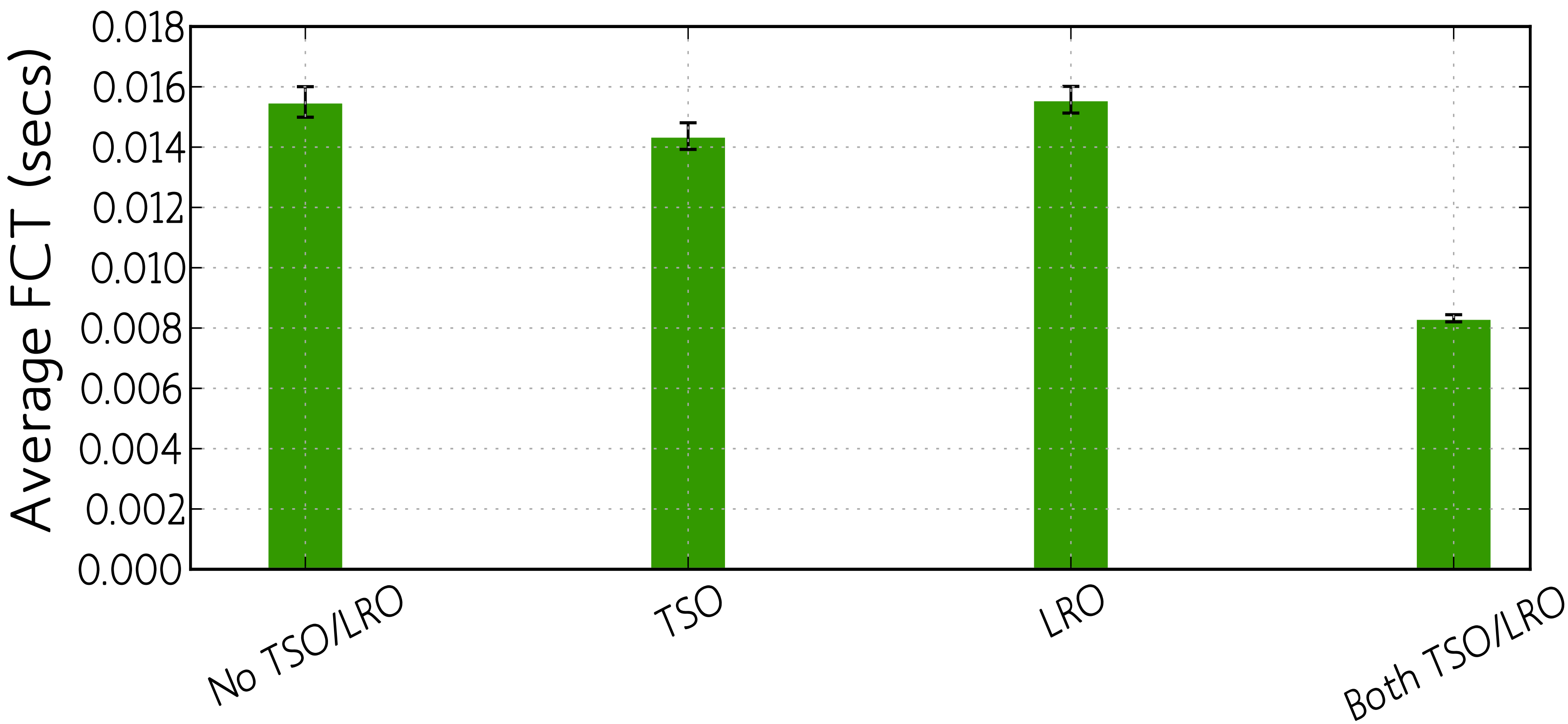


Average FCT (secs)
0.05
0.04
0.03
0.02
0.01
0.00
Simulated RC3
Linux RC3
Low priority out-of-order
packets processed by slow path
High per-packet CPU overhead
1.5KB
7.5KB
15KB
75KB
150KB
750KB
1.5MB
Flow Size

Leveraging NIC Offloading

- TCP Segmentation Offload (TSO)
  - Multiple segments processed by sender stack as a single chunk
- Large Receive Offload (LRO)
  - Multiple segments received aggregated into a single chunk
- RC3 supports offloading to reduce CPU overhead
  - Logically treat each chunk as a single packet at the sender
  - This allows aggregation of segments at the receiver

Leveraging NIC Offloading



Roadmap

- *Isn't congestion control a solved problem?*
  - *Conflicting goals of high throughput and friendliness decoupled through priorities*
- *Scope for performance gains*
  - *Increases with increasing RTTxBW*
- *Design Details*
  - *Additional packets sent backwards from the end using multiple low priority levels*
- *Simulation Results*
  - *40-80% reduction in FCT over baseline TCP implementation*
- *Linux Implementation and Evaluation*
  - *Simple modifications, agnostic to the underlying congestion control algorithm*
- **Challenges and Future**

Where RC3 is of little help…

- Low delay bandwidth product
- Very heavily utilized links
- Small queue buffer size at the bottleneck
- Application pacing

Deployment Concerns

- Partial Priorities Support
- Middleboxes [Honda et. al. 2011, Flach et al 2013]
- Wireless

Future

- Performance gains increase with BWxRTT
  - Likely to increase with time
- Futuristic datacenter bandwidth of 100Gbps
  - 45% reduction in average FCT (over flows)
  - 66% reduction in average FCT (over bytes)

Summary

- Send additional packets from a flow using several layers of low priority service
- Uses only the spare capacity in the network without affecting the regular traffic
- Gives 40-80% reduction in FCTs over baseline TCP