IEEE ICNP 2018

Micro-burst in Data Centers: Observations, Analysis, and Mitigations

Danfeng Shan, Fengyuan Ren, Peng Cheng, Ran Shu, and Chuanxiong Guo

Tsinghua University

Microsoft Research

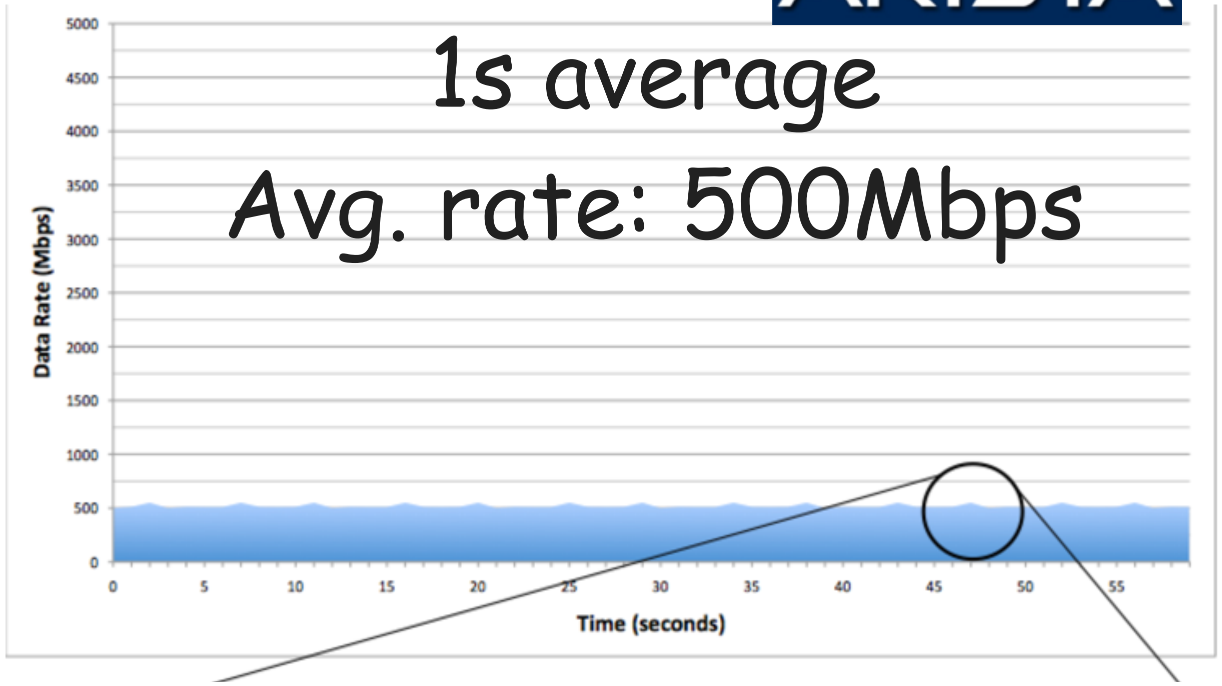
ByteDance

Outline

- Background
- Methodology of Observing Micro-bursts
- Observing and Analyzing Micro-bursts
- Mitigating Micro-bursts
- Conclusion

Micro-burst Traffic

ARISTA

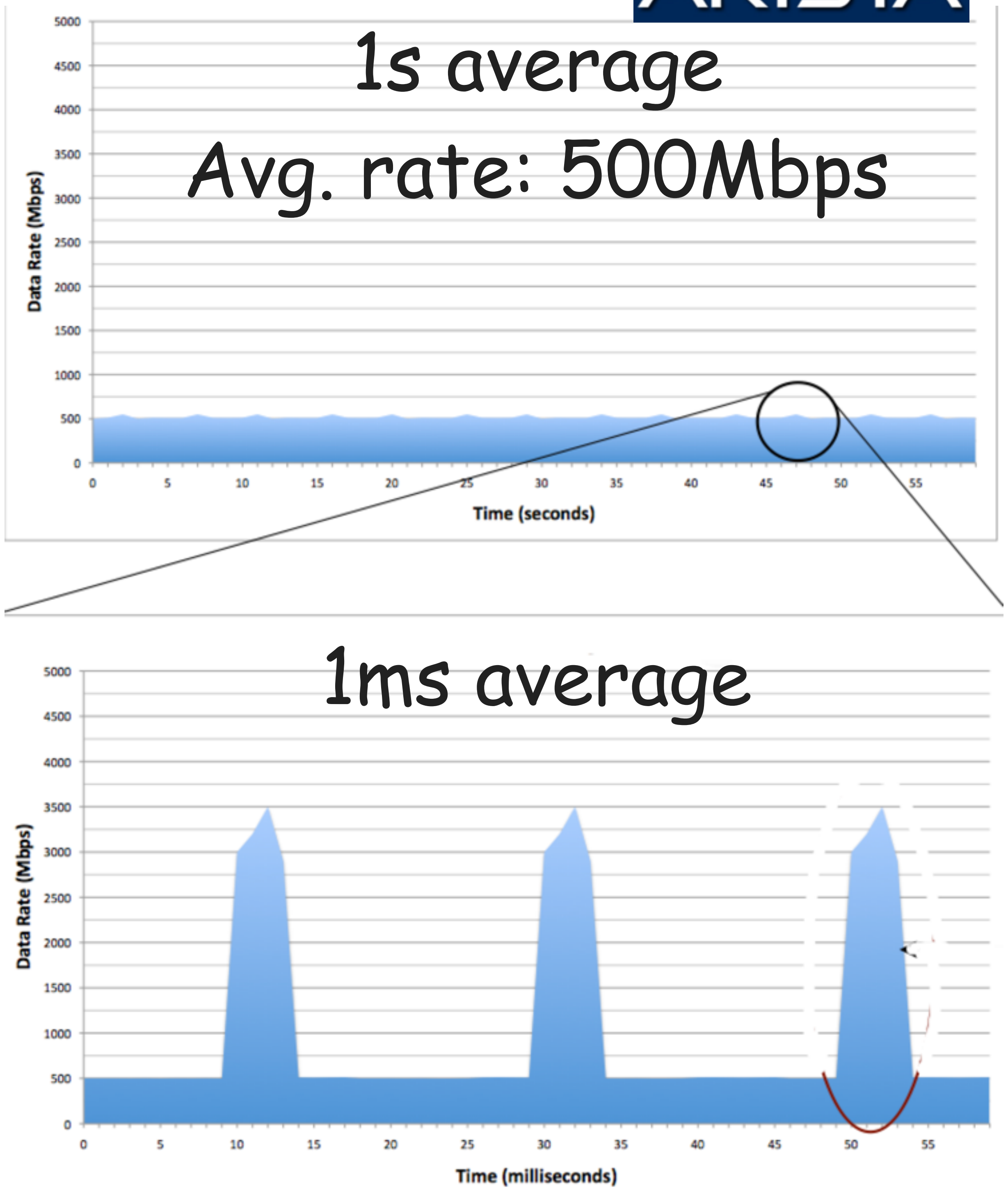


1s average

Avg. rate: 500Mbps

Micro-burst Traffic

ARISTA



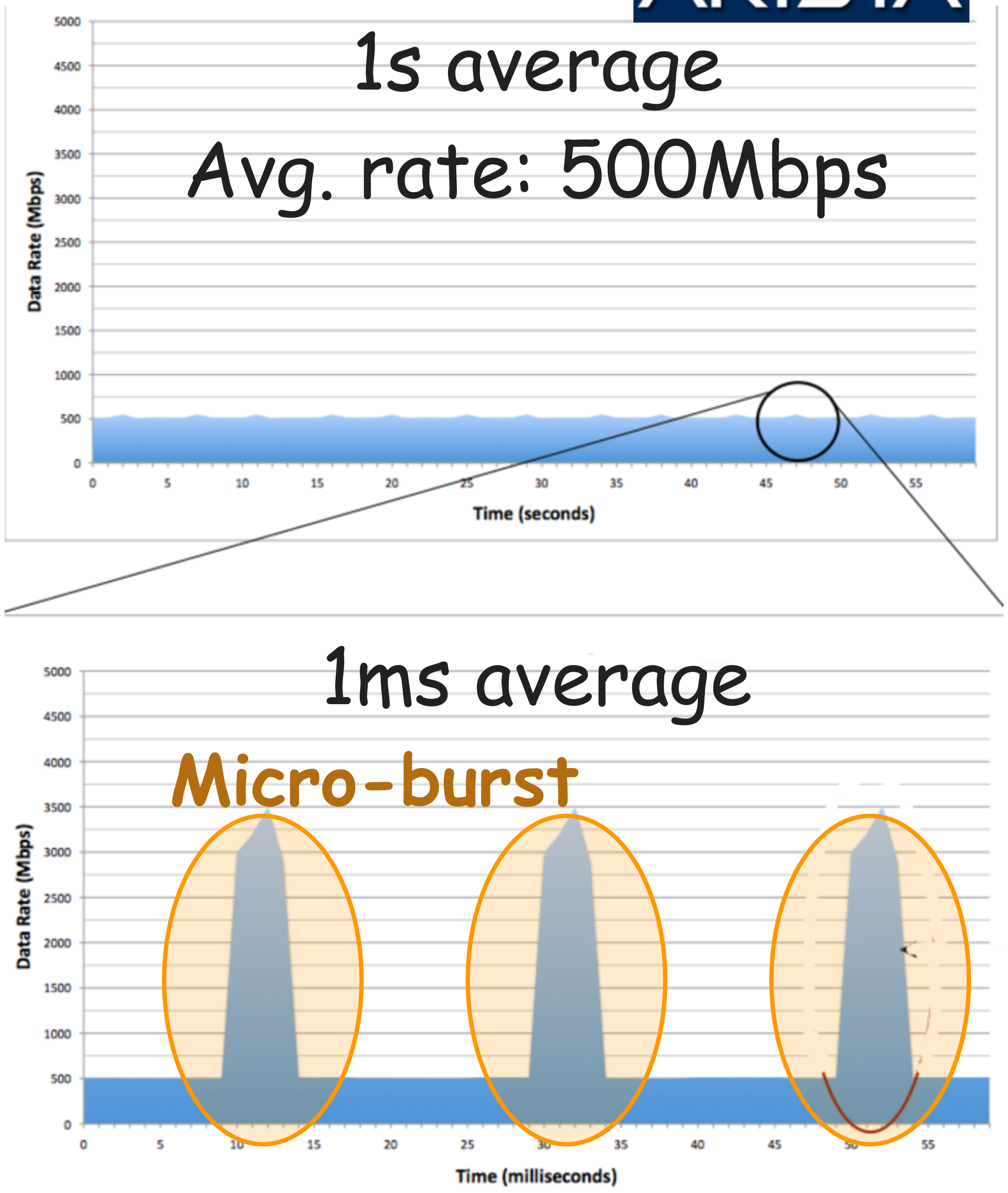
1s average

Avg. rate: 500Mbps

1ms average

Micro-burst Traffic

ARISTA



1s average

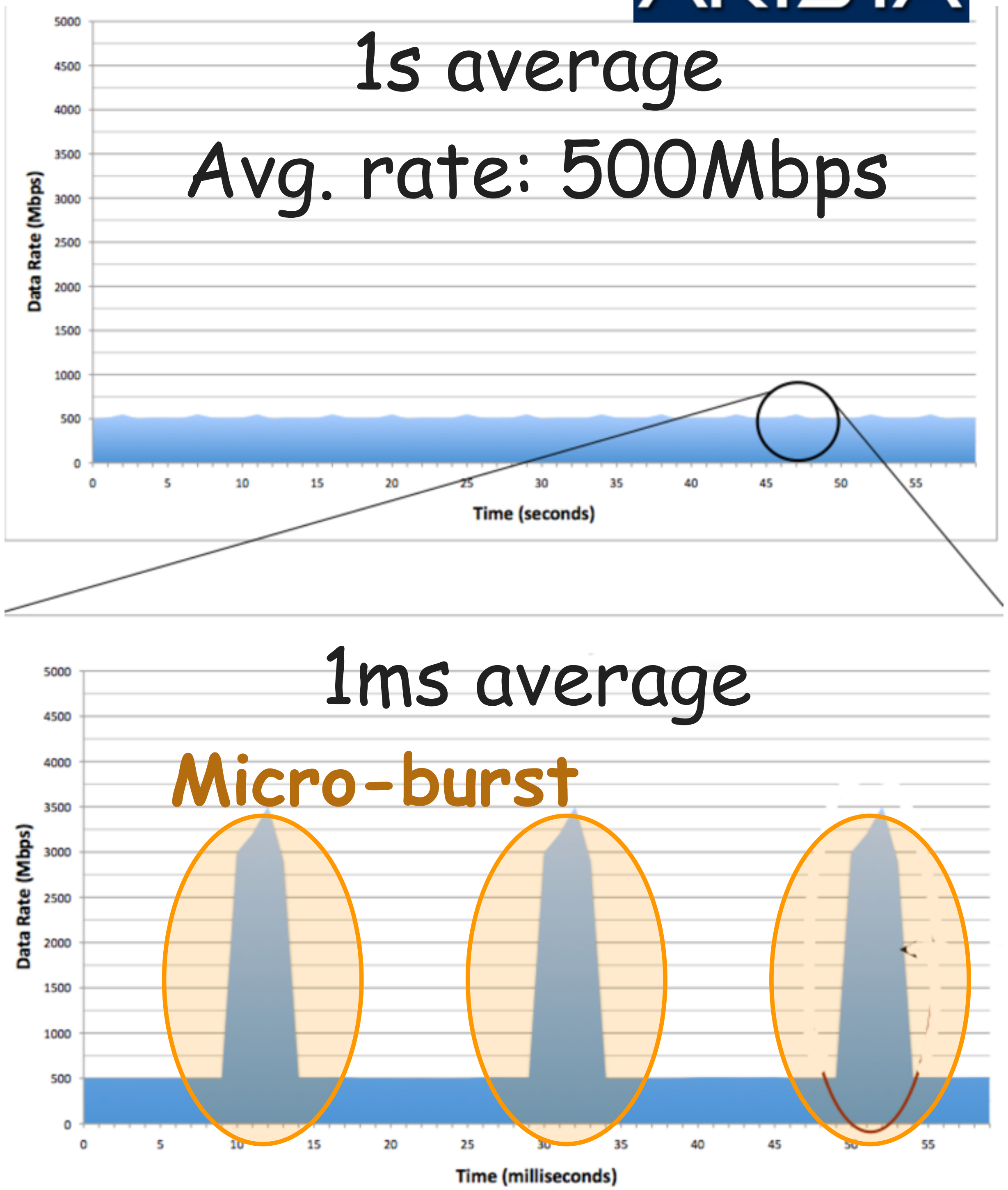
Avg. rate: 500Mbps

1ms average

Micro-burst

Micro-burst Traffic

ARISTA


1s average

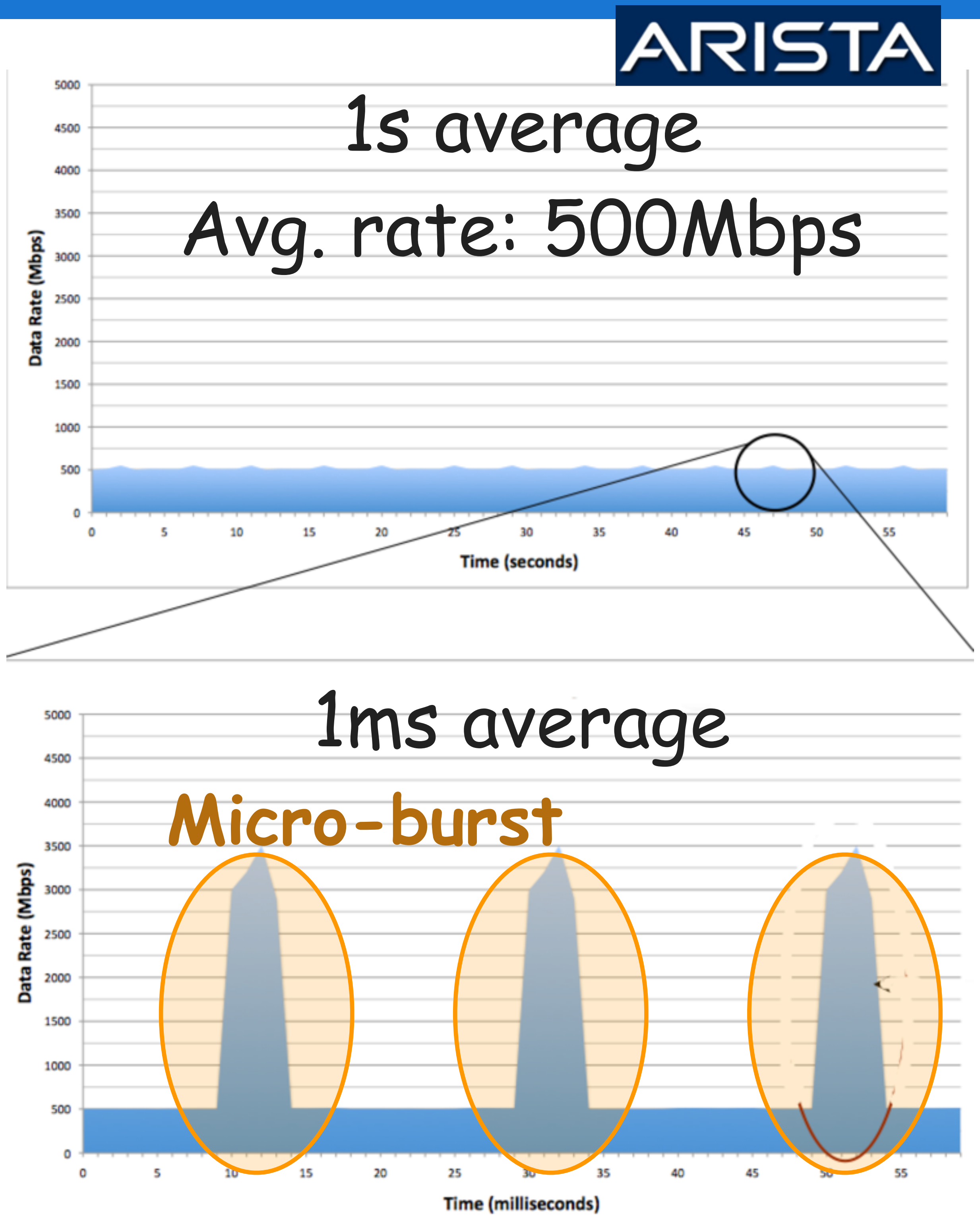
Avg. rate: 500Mbps

1ms average

Micro-burst

Duration: 4.5ms (micro)

Micro-burst Traffic


1s average

Avg. rate: 500Mbps

1ms average

Micro-burst

Hard: detect

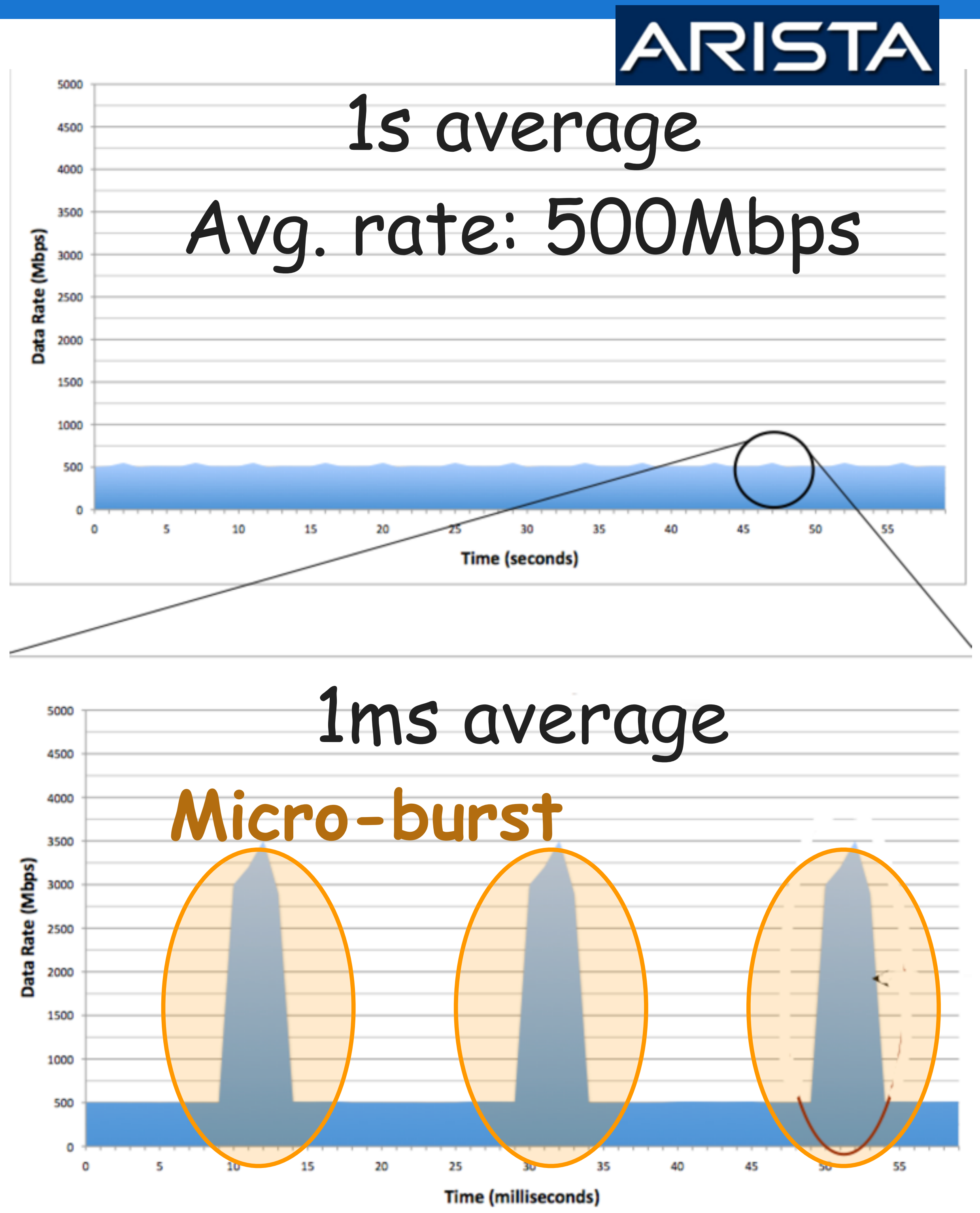
Duration: 4.5ms (micro)

Micro-burst Traffic

Hard: detect

Duration: 4.5ms (micro)

Rate: 3.5Gbps (burst)



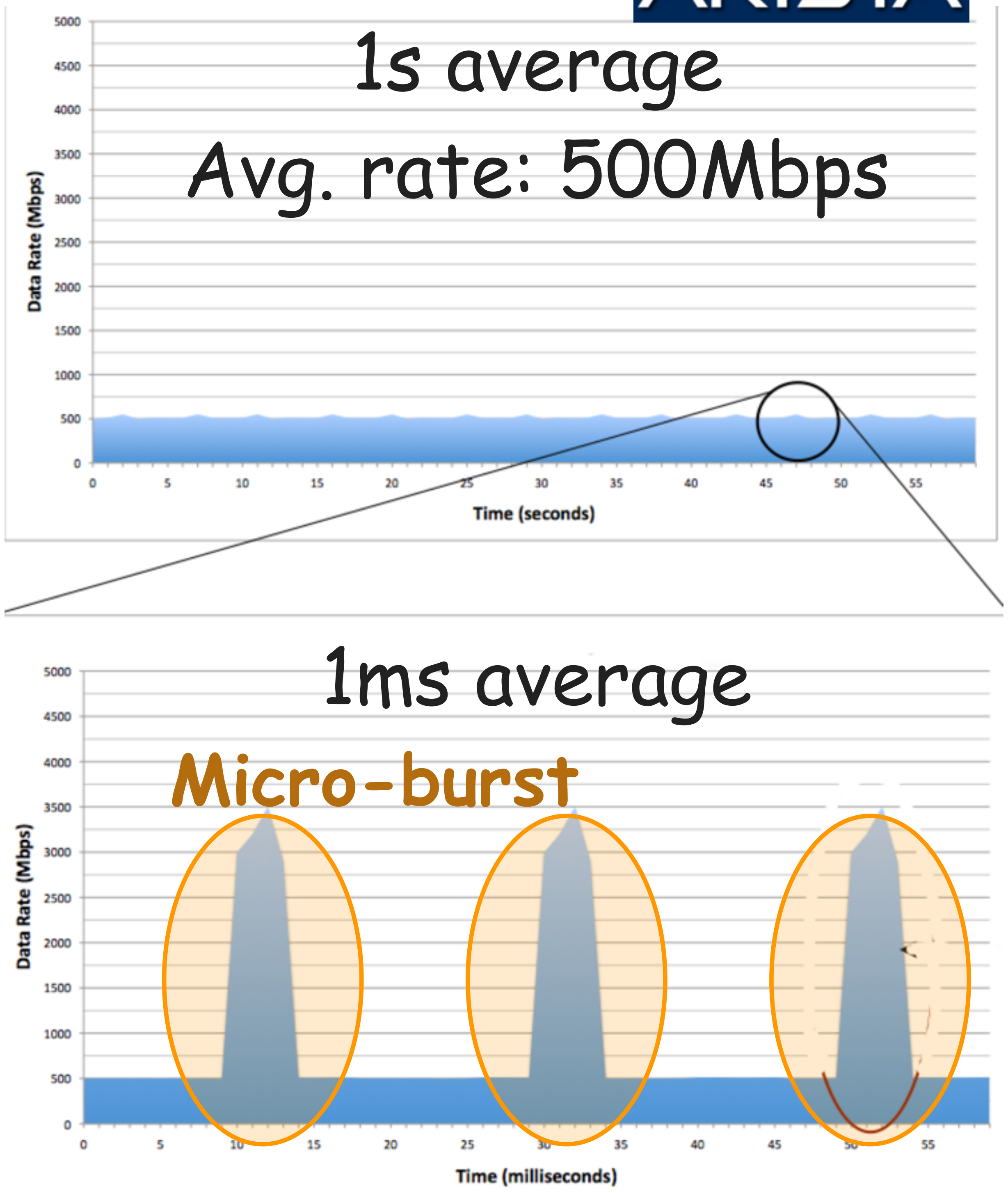
1s average

Avg. rate: 500Mbps

1ms average

Micro-burst

Micro-burst Traffic



1s average

Avg. rate: 500Mbps

1ms average

Micro-burst

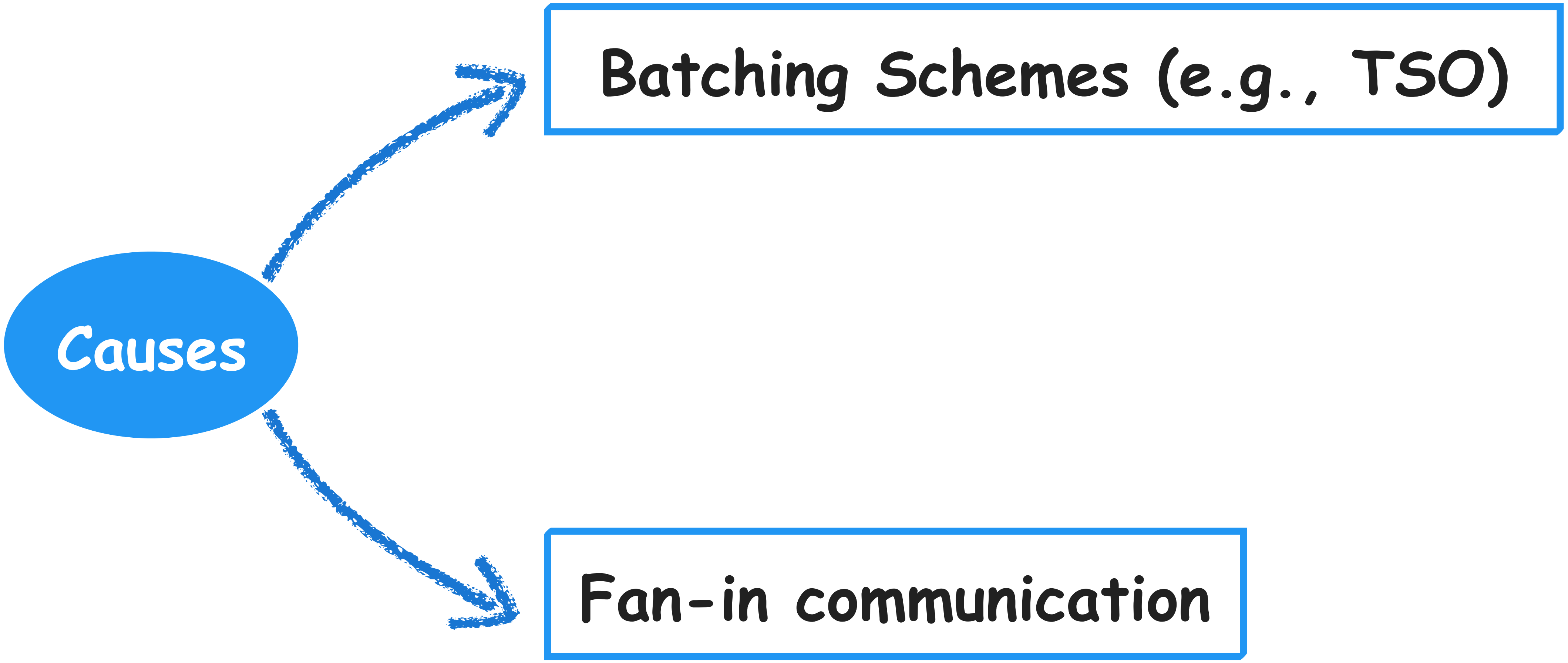
Hard: detect

Duration: 4.5ms (micro)

Easy: packet dropping

Rate: 3.5Gbps (burst)

Micro-burst Traffic

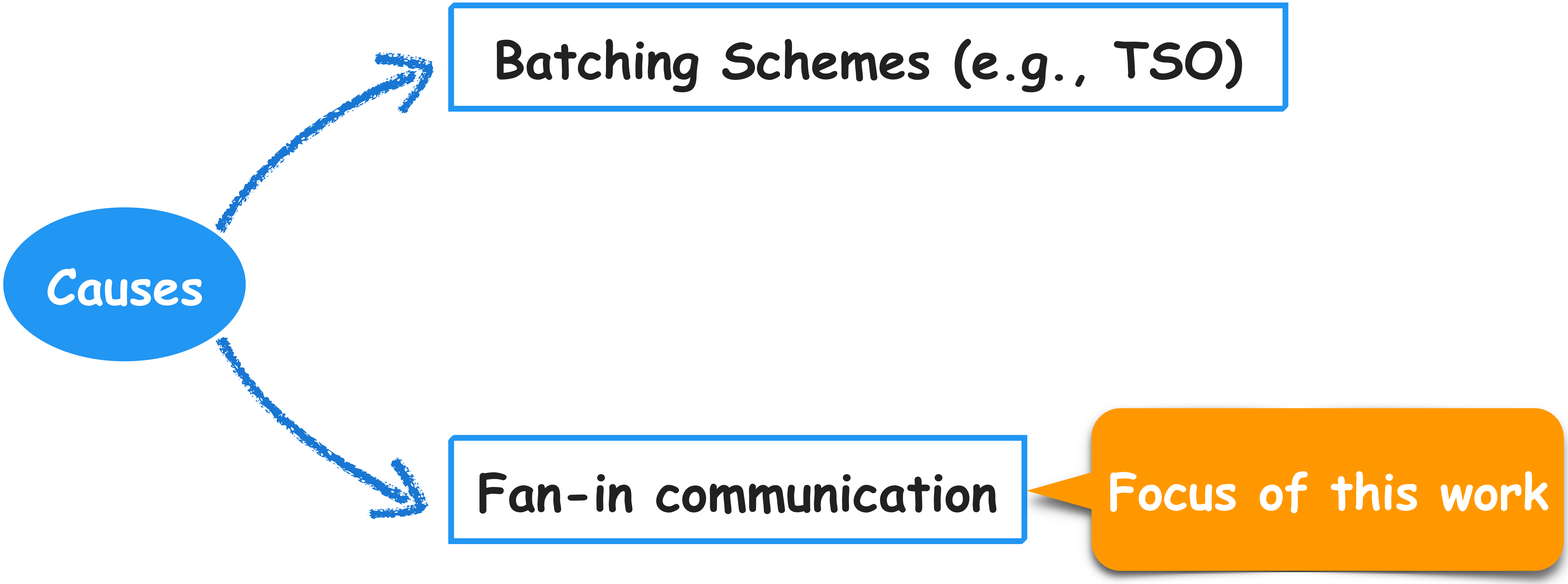


Causes

Batching Schemes (e.g., TSO)

Fan-in communication

Micro-burst Traffic



Causes

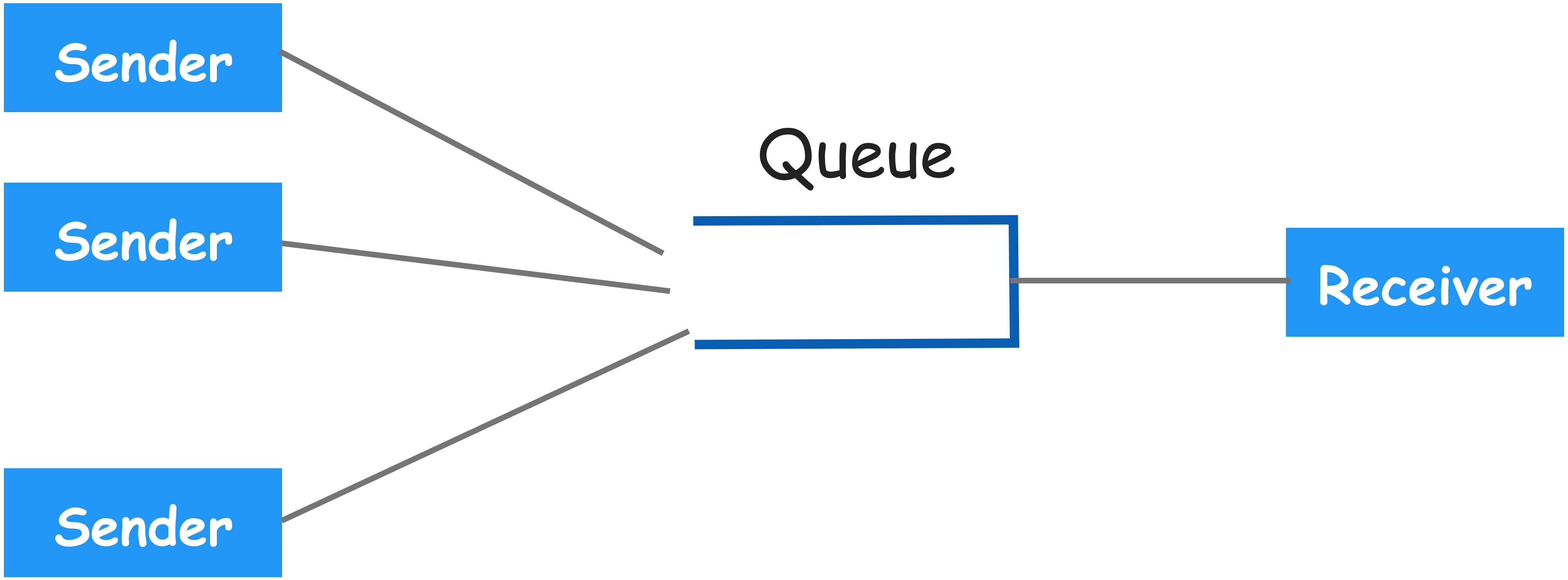
Batching Schemes (e.g., TSO)

Fan-in communication

Focus of this work

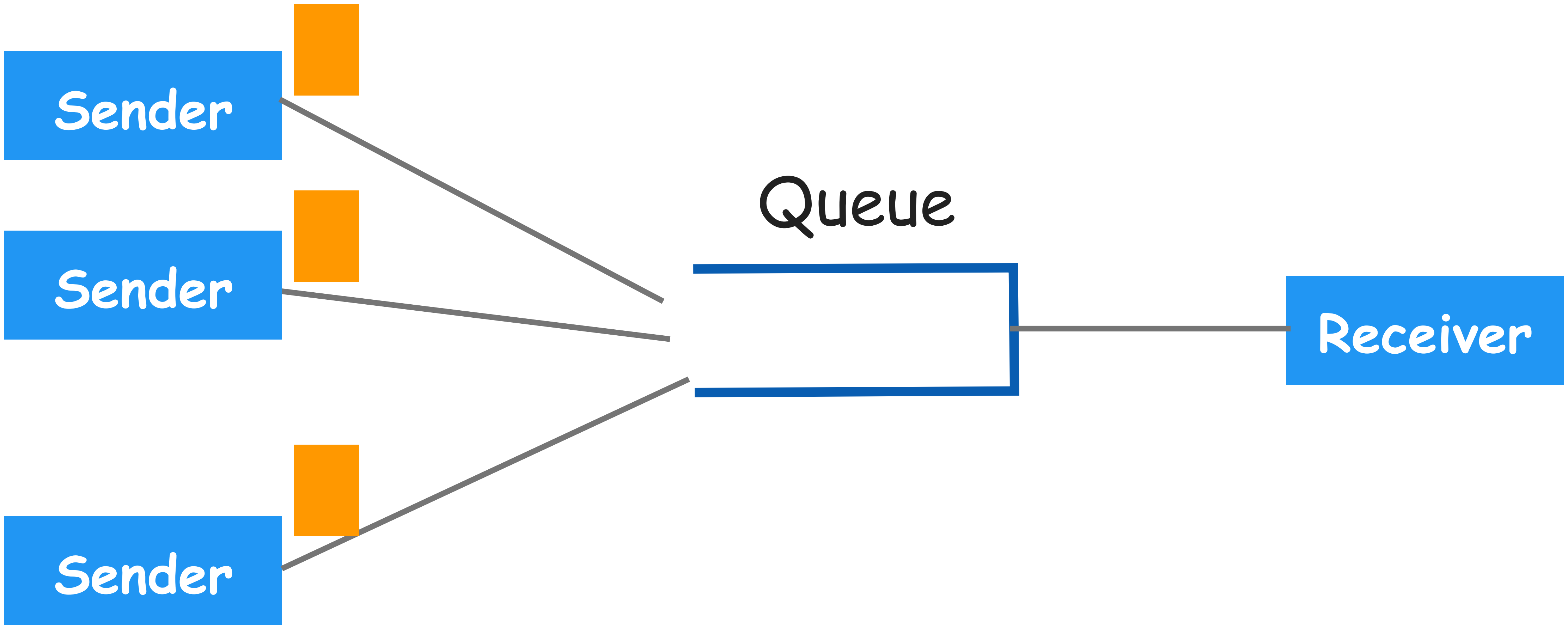
Micro-burst Traffic

Fan-in Communication: Distributed Storage, MapReduce, Web Search, Memcached Systems, Distributed Machine Learning ……



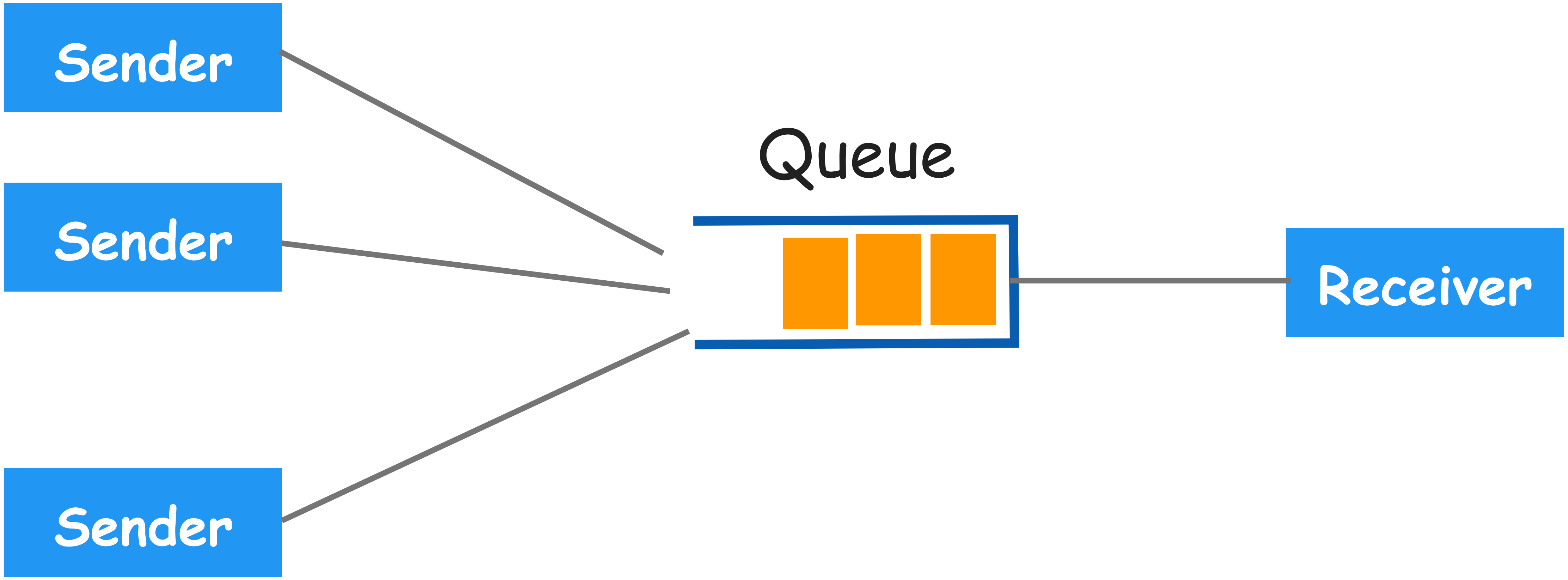
Micro-burst Traffic

Fan-in Communication: Distributed Storage, MapReduce, Web Search, Memcached Systems, Distributed Machine Learning ……



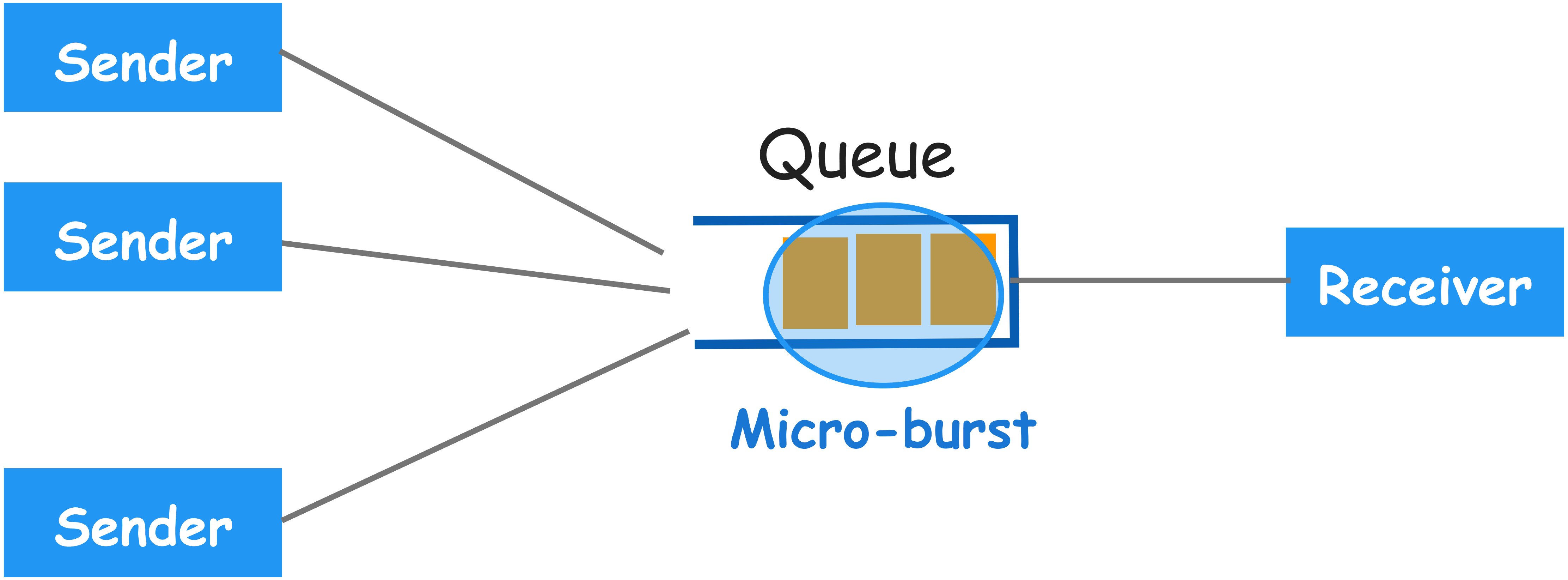
Micro-burst Traffic

Fan-in Communication: Distributed Storage, MapReduce, Web Search, Memcached Systems, Distributed Machine Learning ……



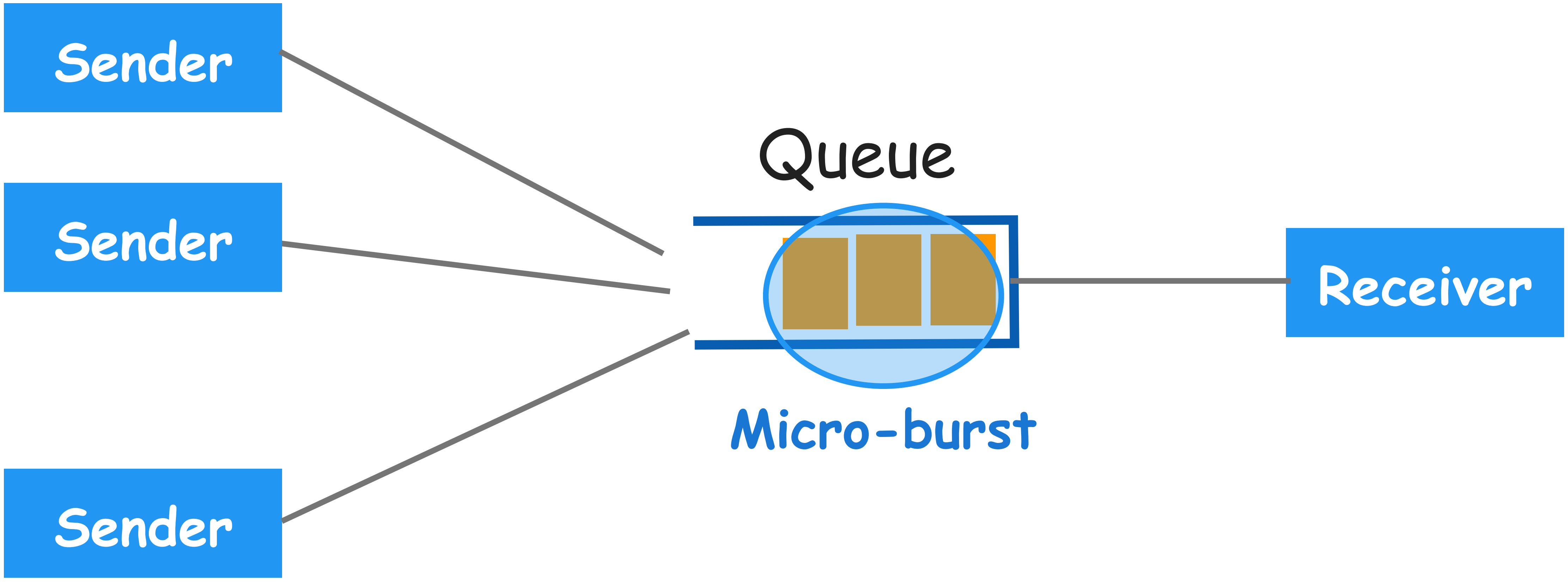
Micro-burst Traffic

Fan-in Communication: Distributed Storage, MapReduce, Web Search, Memcached Systems, Distributed Machine Learning ……



Micro-burst Traffic

Fan-in Communication: Distributed Storage, MapReduce, Web Search, Memcached Systems, Distributed Machine Learning ……



Characteristics of Micro-burst ?

Outline

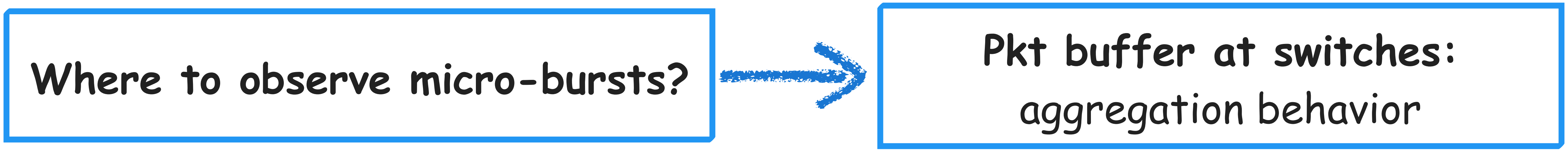
- Background
- Methodology of Observing Micro-bursts
- Observing and Analyzing Micro-bursts
- Mitigating Micro-bursts
- Conclusion

Methodology

Where to observe micro-bursts?

How to observe micro-bursts?

Methodology

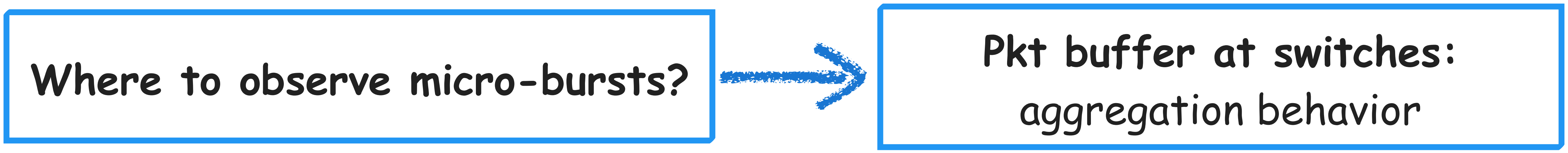


Where to observe micro-bursts?

Pkt buffer at switches:
aggregation behavior

How to observe micro-bursts?

Methodology



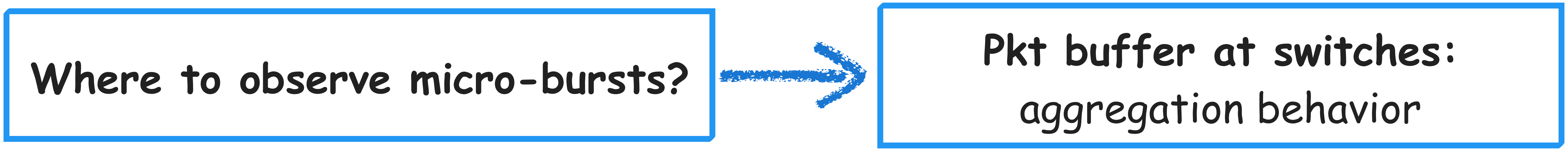
Where to observe micro-bursts?

Pkt buffer at switches:
aggregation behavior

How to observe micro-bursts?

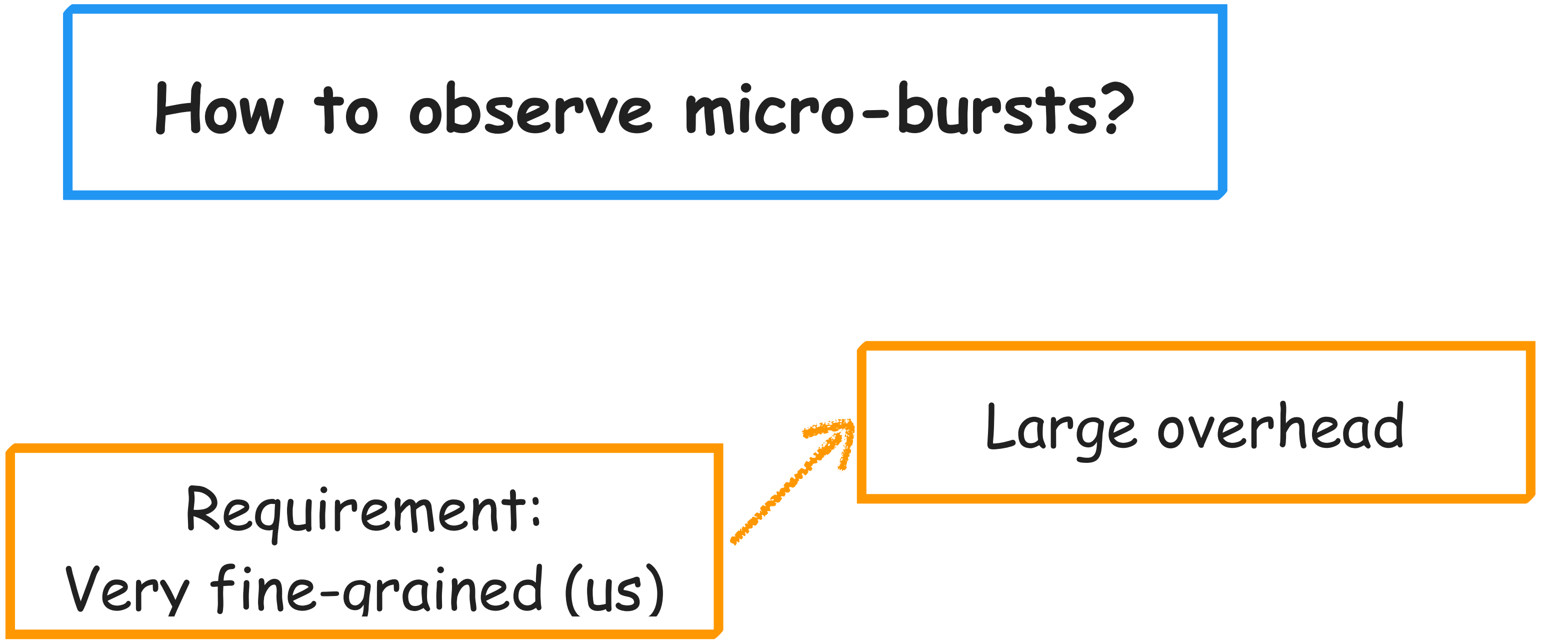
Requirement:
Very fine-grained (us)

Methodology



Where to observe micro-bursts?

Pkt buffer at switches:
aggregation behavior

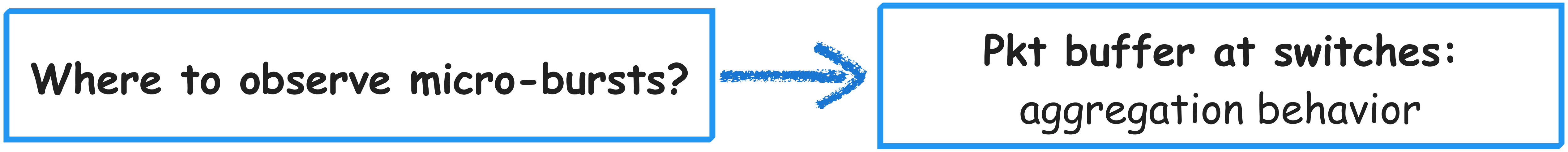


How to observe micro-bursts?

Requirement:
Very fine-grained (us)

Large overhead

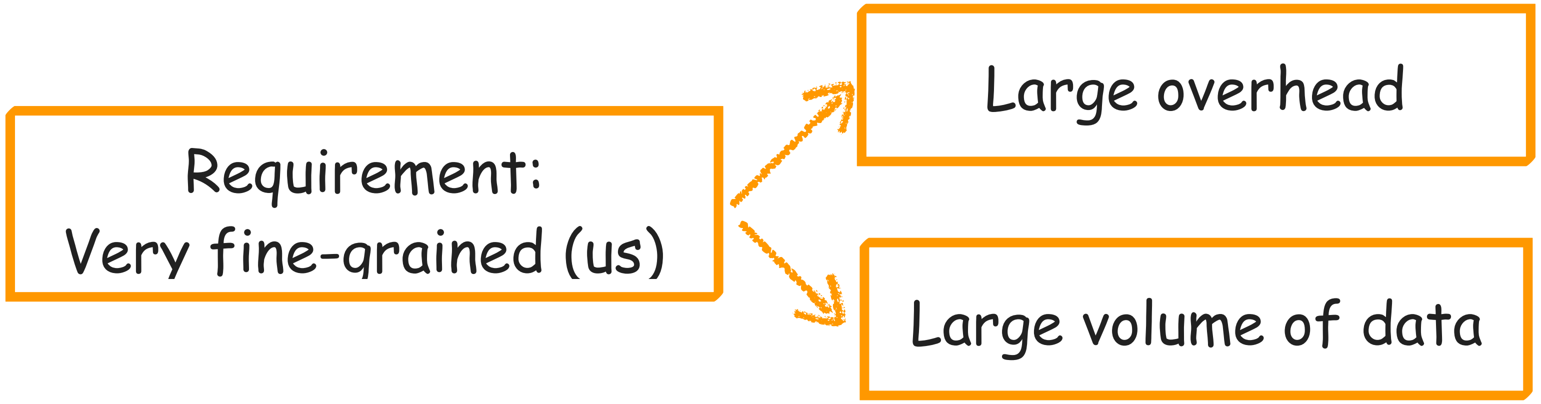
Methodology



Where to observe micro-bursts?

Pkt buffer at switches:
aggregation behavior

How to observe micro-bursts?

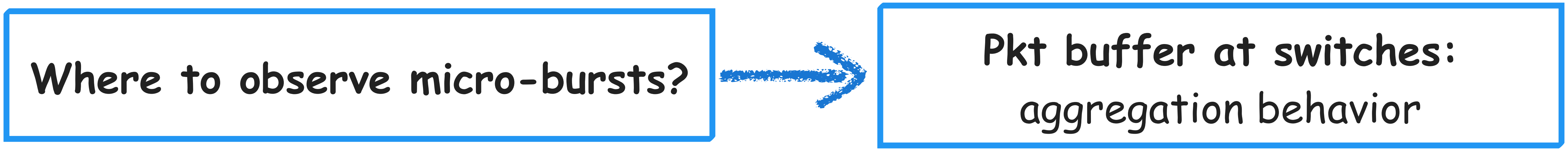


Requirement:
Very fine-grained (us)

Large overhead

Large volume of data

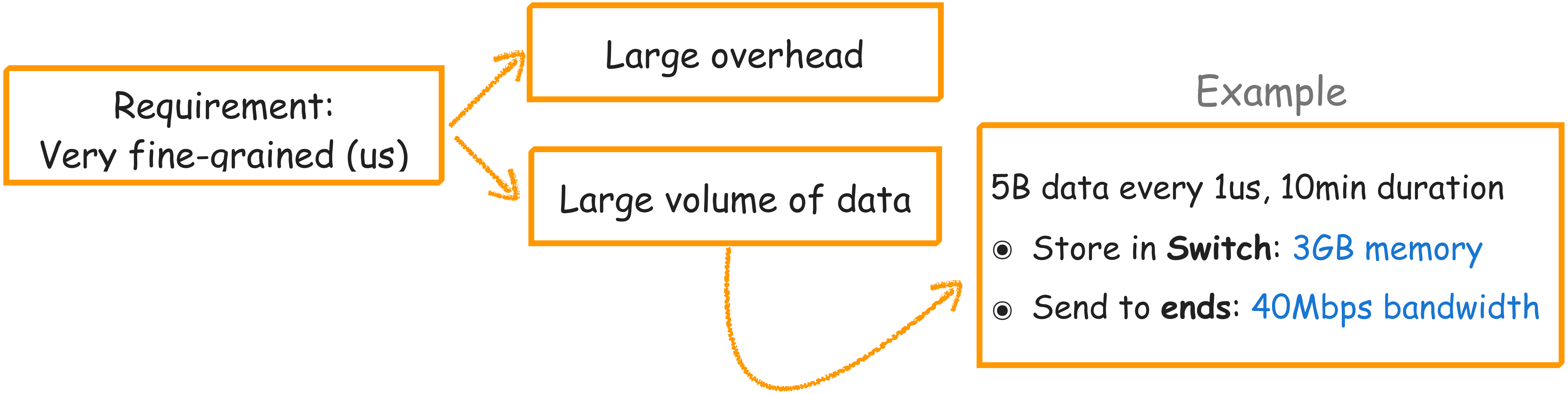
Methodology



Where to observe micro-bursts?

Pkt buffer at switches:
aggregation behavior

How to observe micro-bursts?



Requirement:
Very fine-grained (us)

Large overhead

Large volume of data

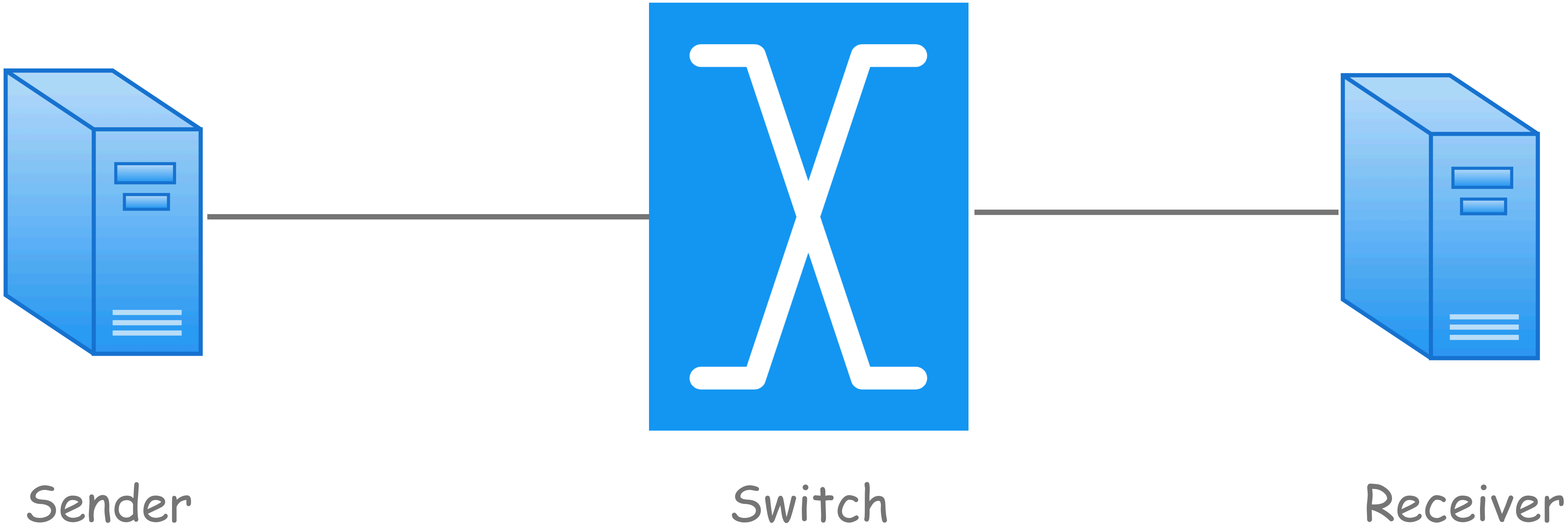
Example

5B data every 1us, 10min duration

- Store in **Switch**: 3GB memory
- Send to **ends**: 40Mbps bandwidth

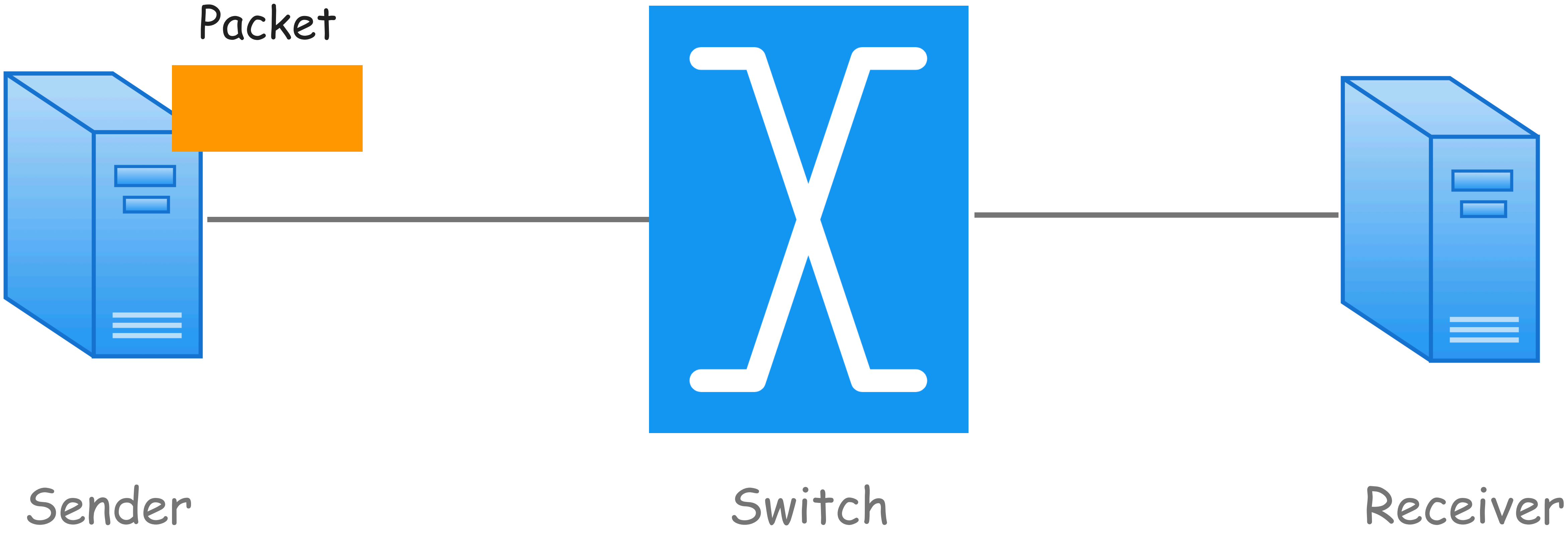
Methodology

— How to observe micro-burst



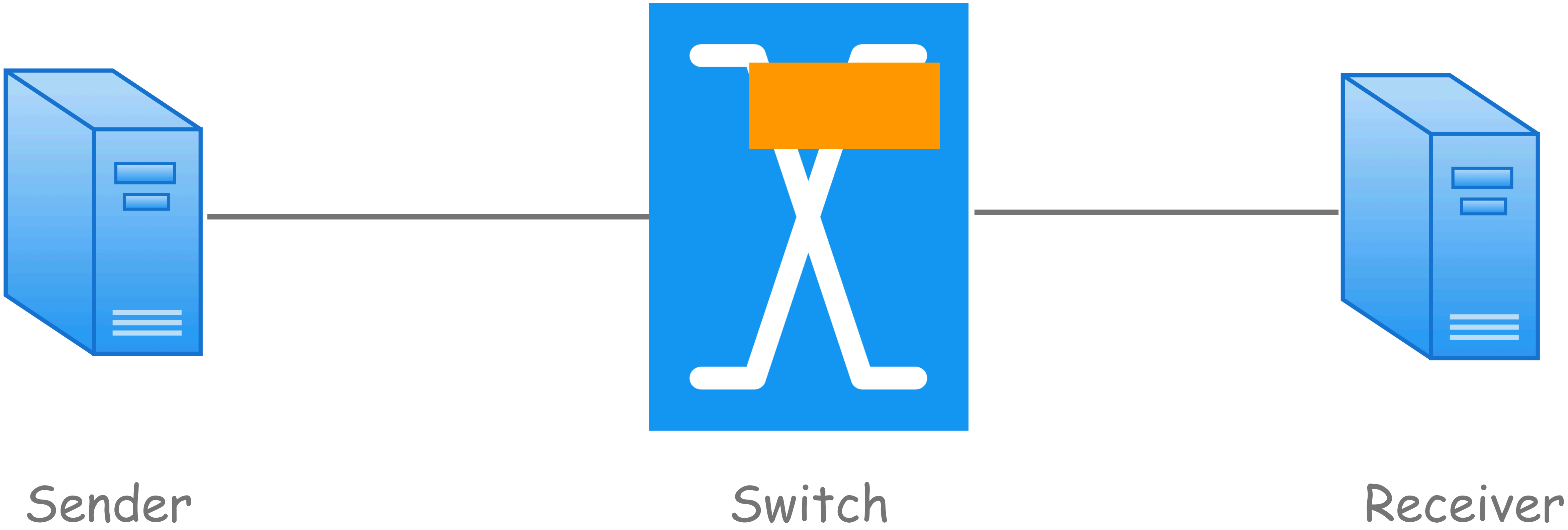
Methodology

— How to observe micro-burst



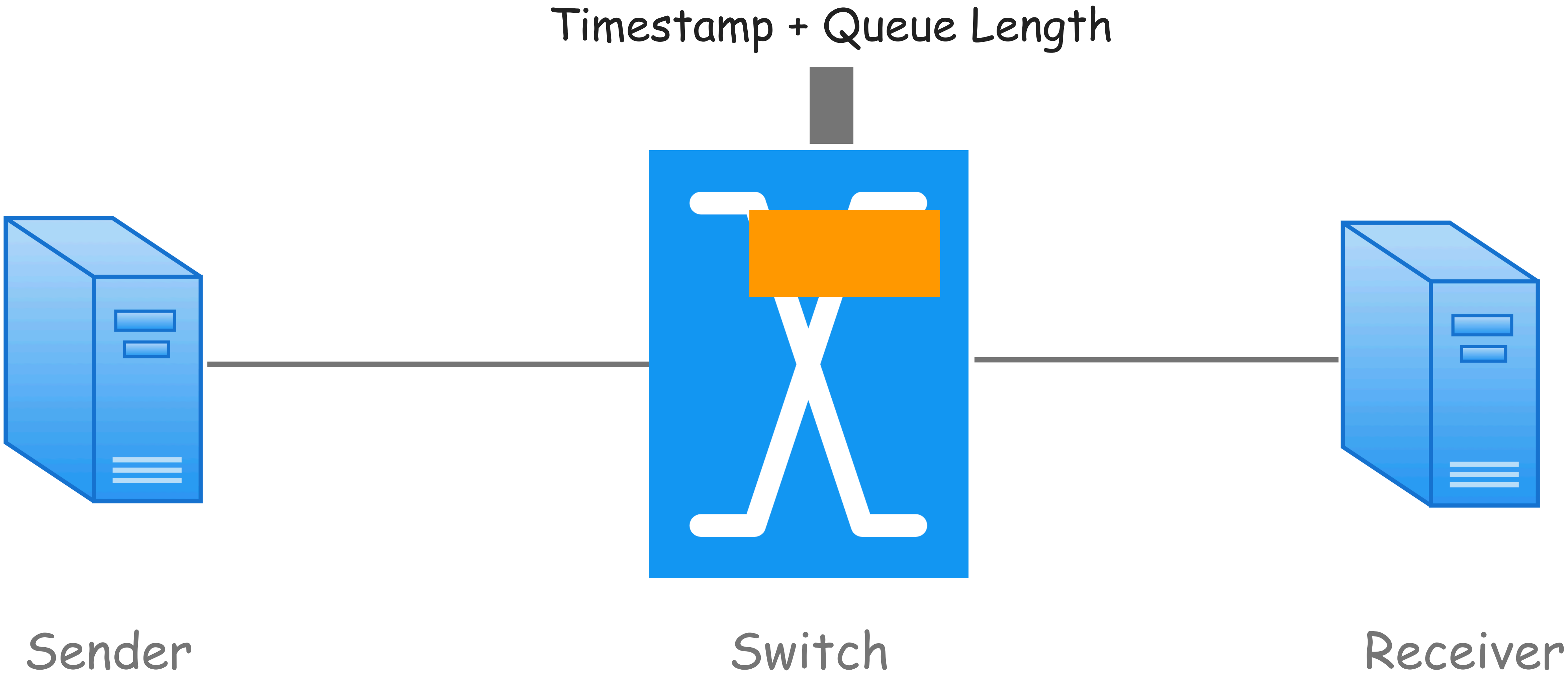
Methodology

— How to observe micro-burst



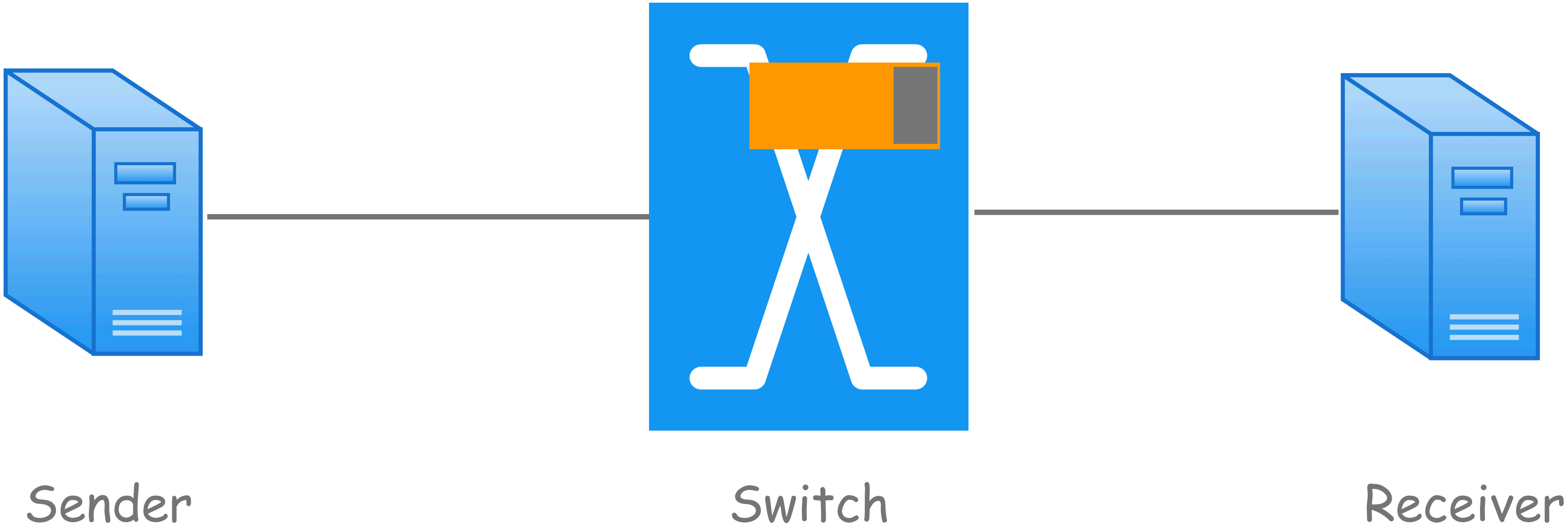
Methodology

— How to observe micro-burst



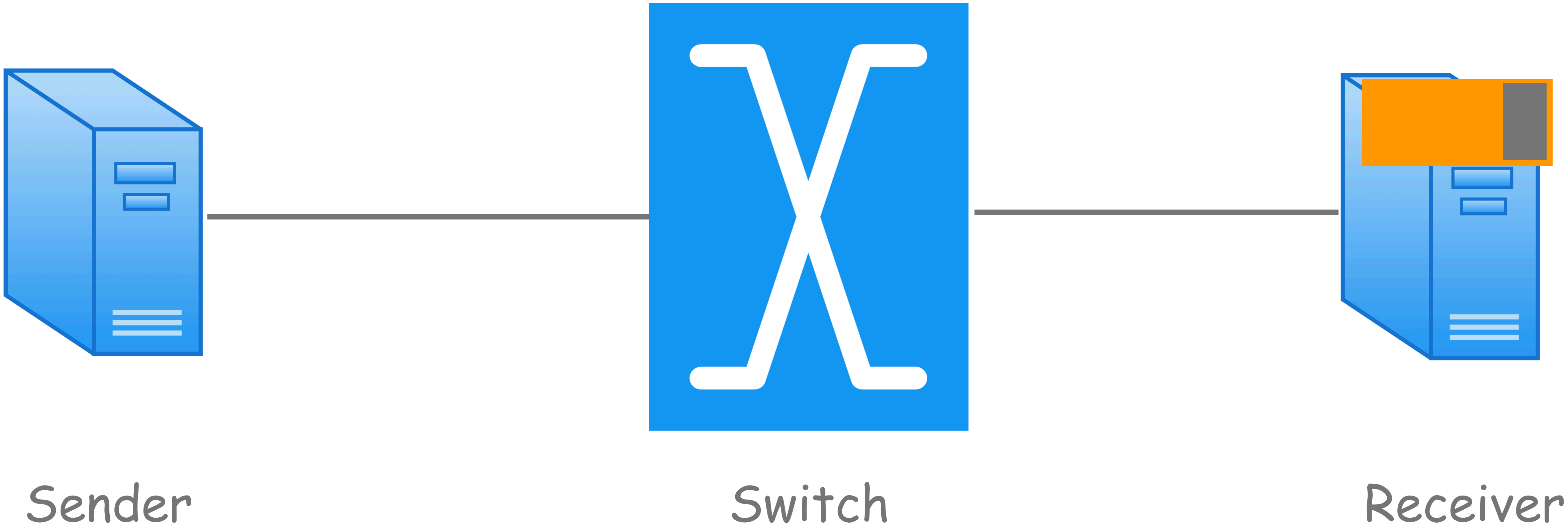
Methodology

— How to observe micro-burst



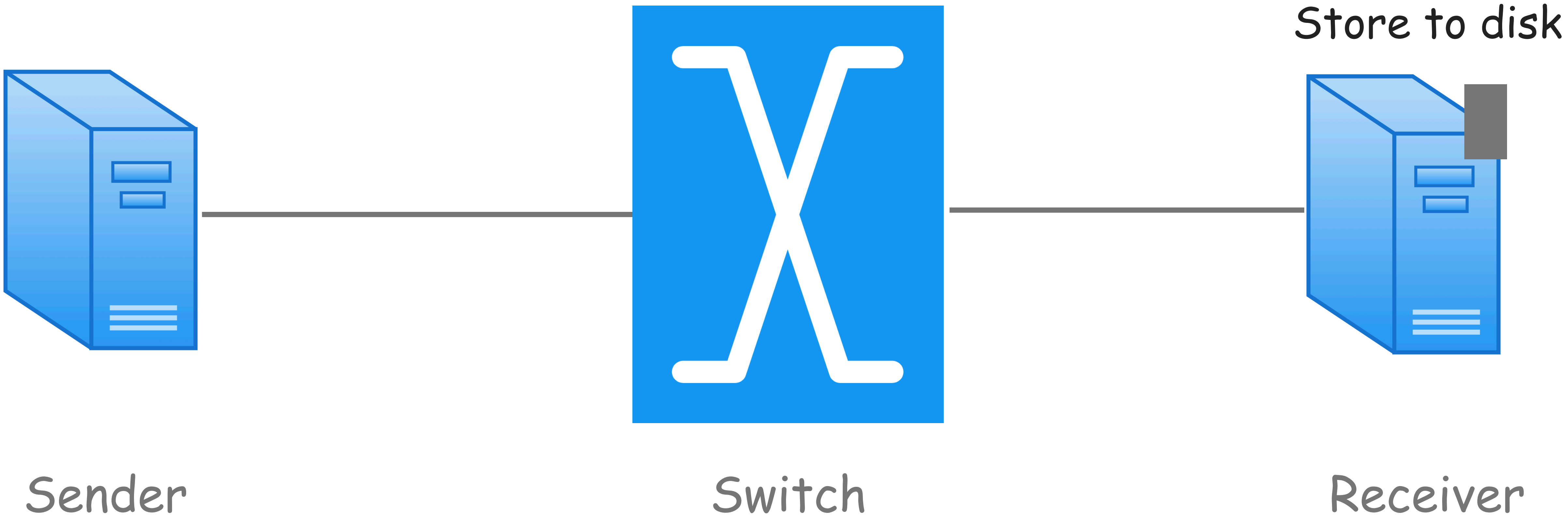
Methodology

— How to observe micro-burst



Methodology

— How to observe micro-burst



Methodology

— How to observe micro-burst

Benefits:

✓ No need to consume switch memory
✓ No need to consume bandwidth
✓ Low overhead to switch

Methodology

— How to observe micro-burst

Benefits:

- ✓ No need to consume switch memory
- ✓ No need to consume bandwidth
- ✓ Low overhead to switch

NetFPGA
Implementaion

Methodology

— How to observe micro-burst

Benefits:

- ✓ No need to consume switch memory
- ✓ No need to consume bandwidth
- ✓ Low overhead to switch

NetFPGA
Implementaion

Latency: +8ns

Methodology

— How to observe micro-burst

Benefits:

- ✓ No need to consume switch memory
- ✓ No need to consume bandwidth
- ✓ Low overhead to switch

NetFPGA Implementaion

Latency: +8ns

Resource Usage

Resources	ECN Switch	+Qlen Monitor
Slice Flip Flops	14738	14777
LUTs	18048	19050

Methodology

— How to observe micro-burst

Benefits:

✓ No need to consume switch memory
✓ No need to consume bandwidth
✓ Low overhead to switch

NetFPGA Implementaion

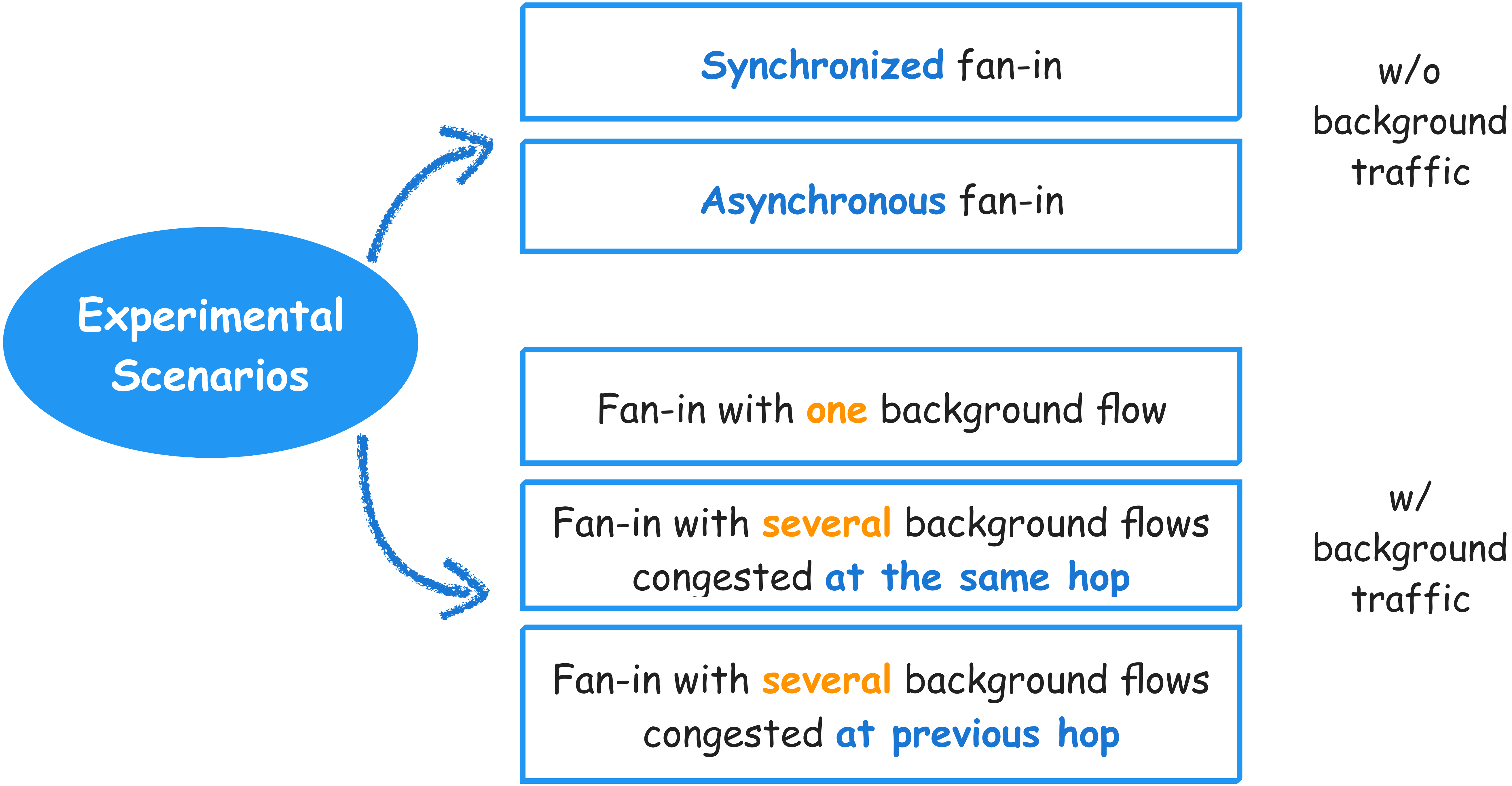
Latency: +8ns

Resource Usage

Resources	ECN Switch	+Qlen Monitor
Slice Flip Flops	14738	14777
LUTs	18048	19050

+8.3%

Methodology



Experimental Scenarios

Synchronized fan-in

Asynchronous fan-in

w/o background traffic

Fan-in with one background flow

Fan-in with several background flows congested at the same hop

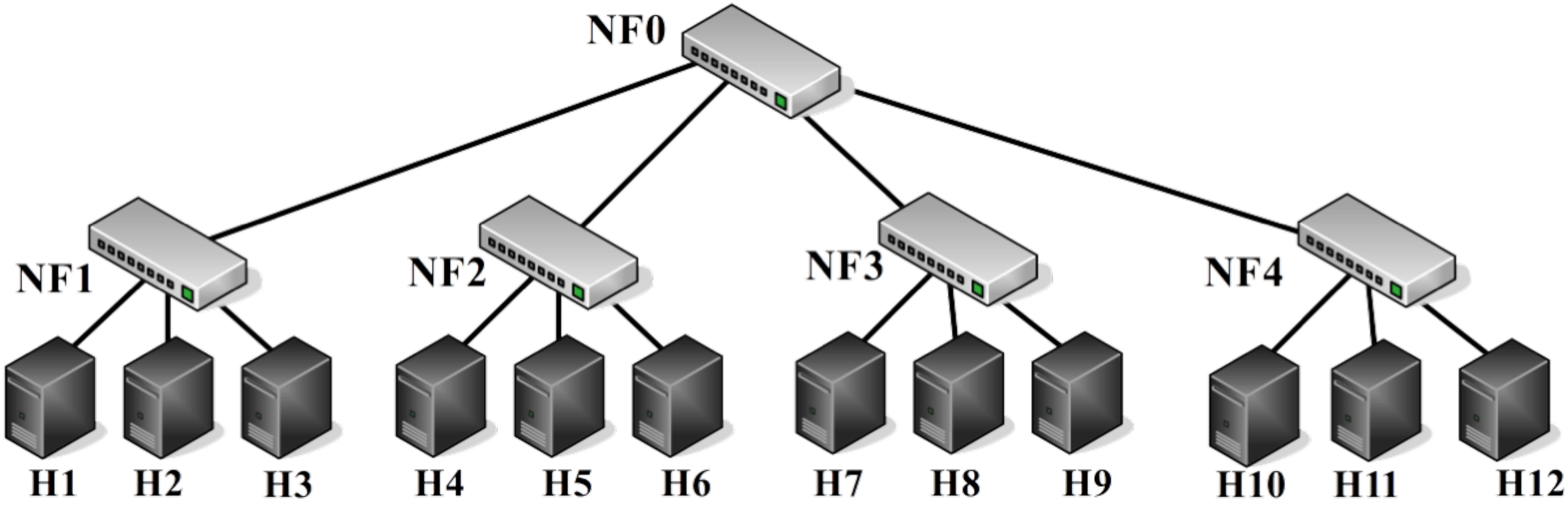
Fan-in with several background flows congested at previous hop

w/ background traffic

Outline

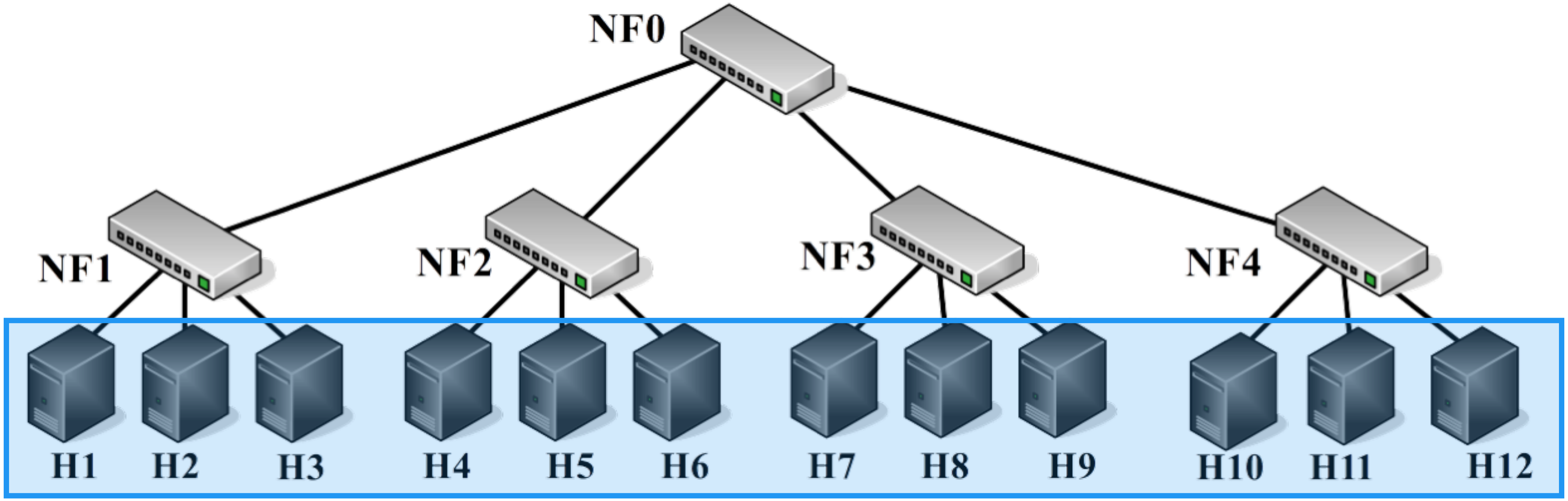
- Background
- Methodology of Observing Micro-bursts
- Observing and Analyzing Micro-bursts
- Mitigating Micro-bursts
- Conclusion

Experiment Settings



Testbed

Experiment Settings

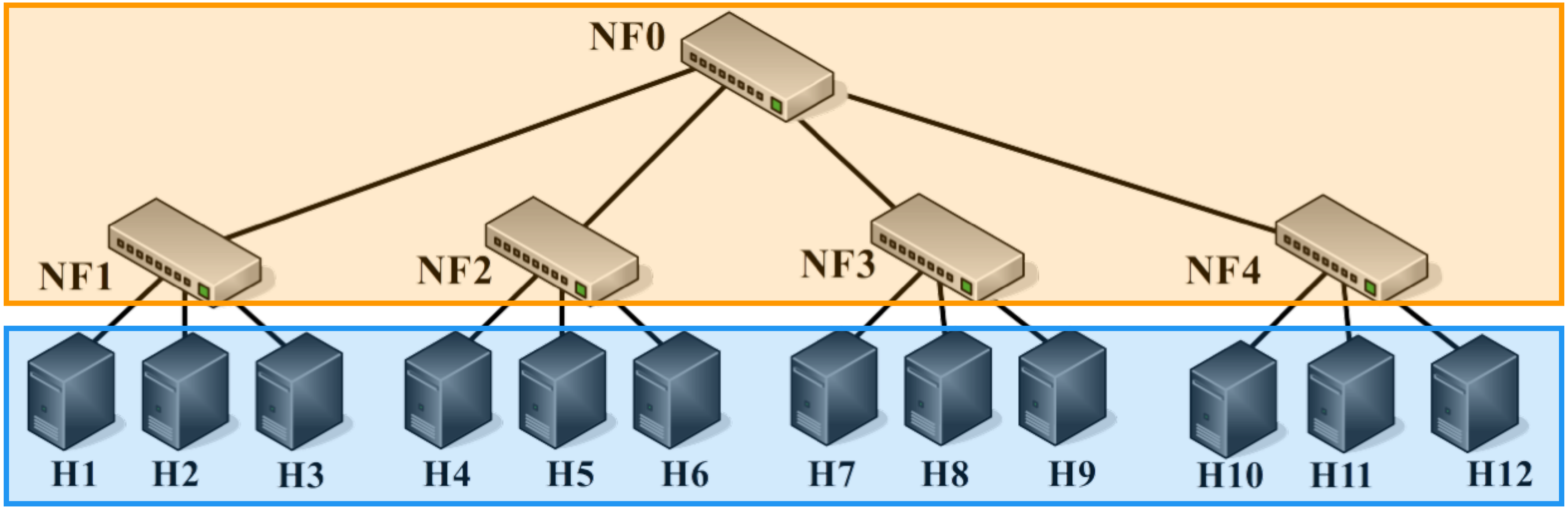


12 servers: CentOS, Linux 2.6.38

Testbed

Experiment Settings

4 NetFPGA cards (1Gbps):
512KB buffer, queue length monitoring

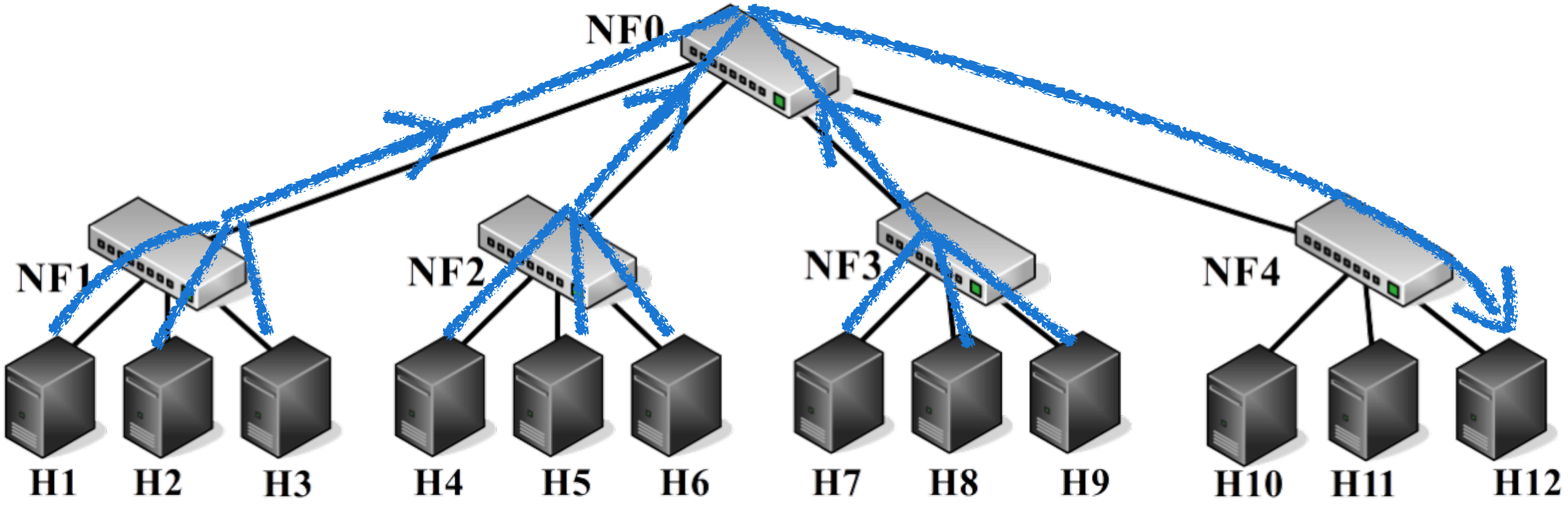


12 servers: CentOS, Linux 2.6.38

Testbed

Observations

— Synchronized fan-in

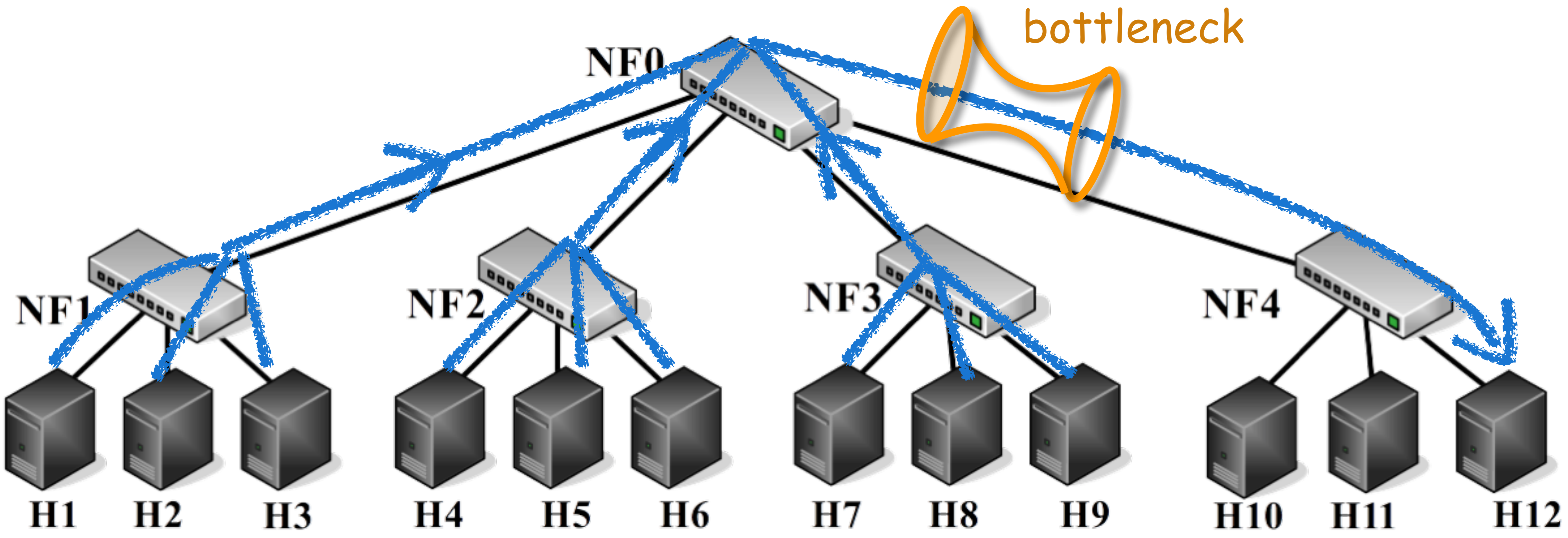


Traffic: H1-9 —> H12, 18 flows

Experiment Traffic

Observations

— Synchronized fan-in

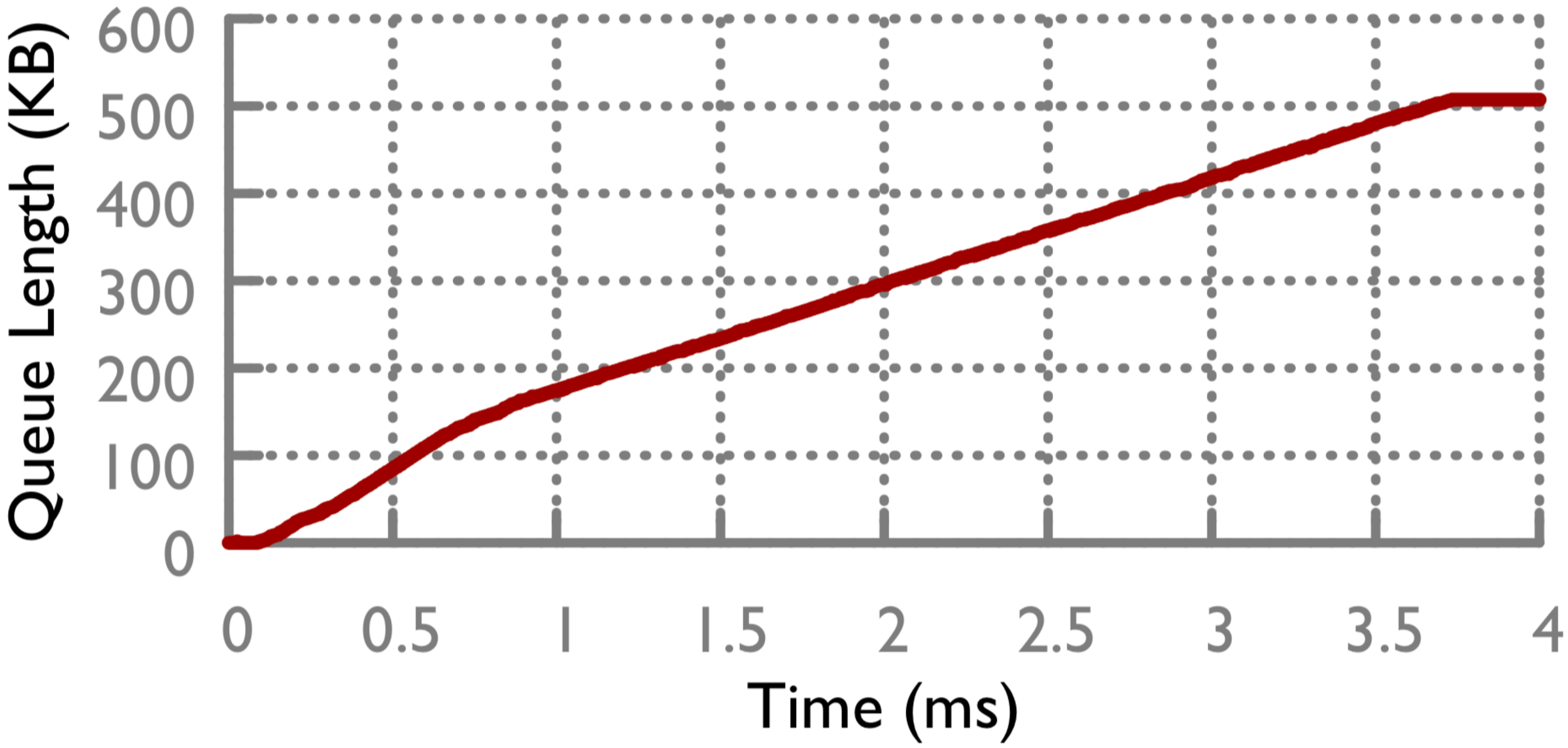


Traffic: H1-9 —> H12, 18 flows

Experiment Traffic

Observations

— Synchronized fan-in

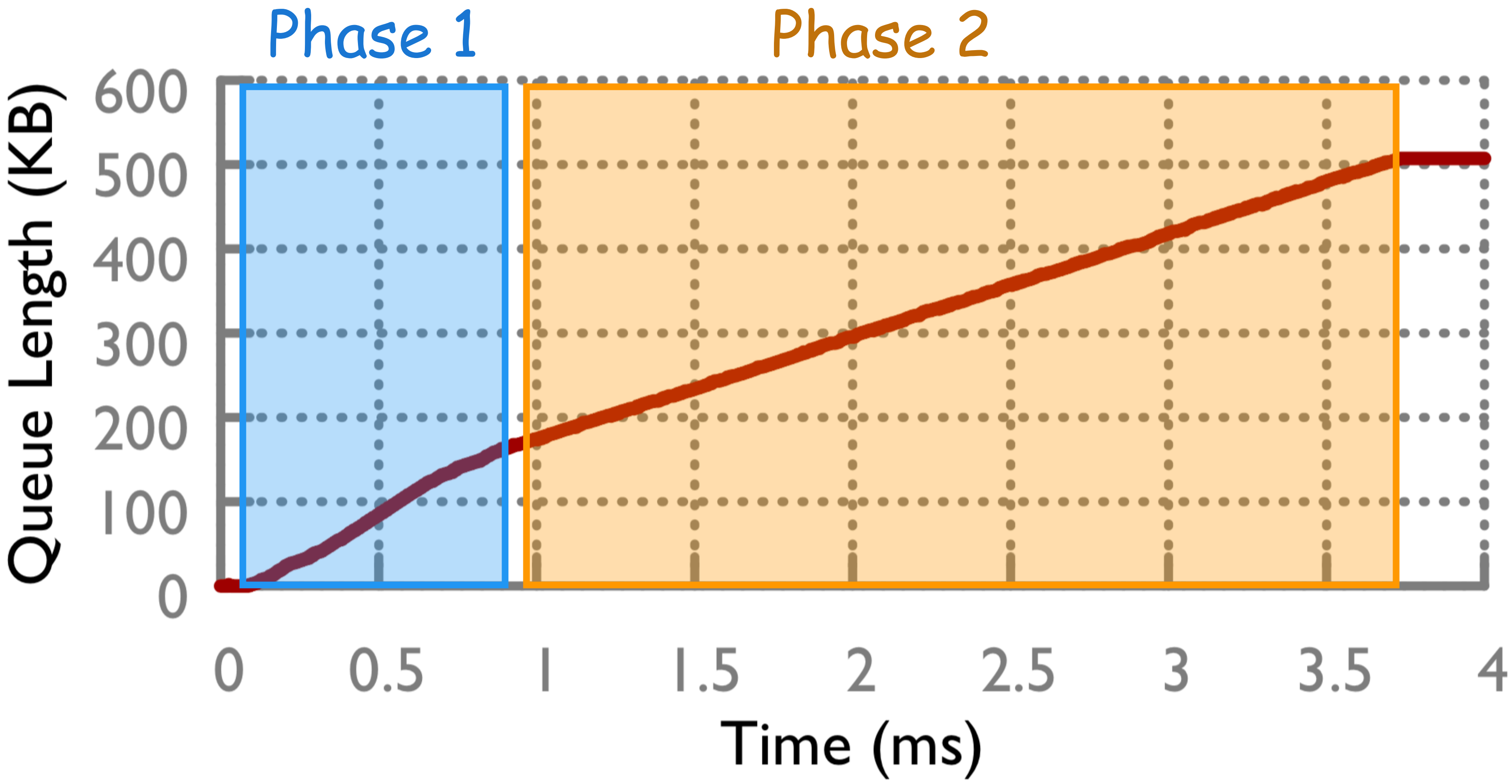


Queue Length Evolution

Experiment Results

Observations

— Synchronized fan-in

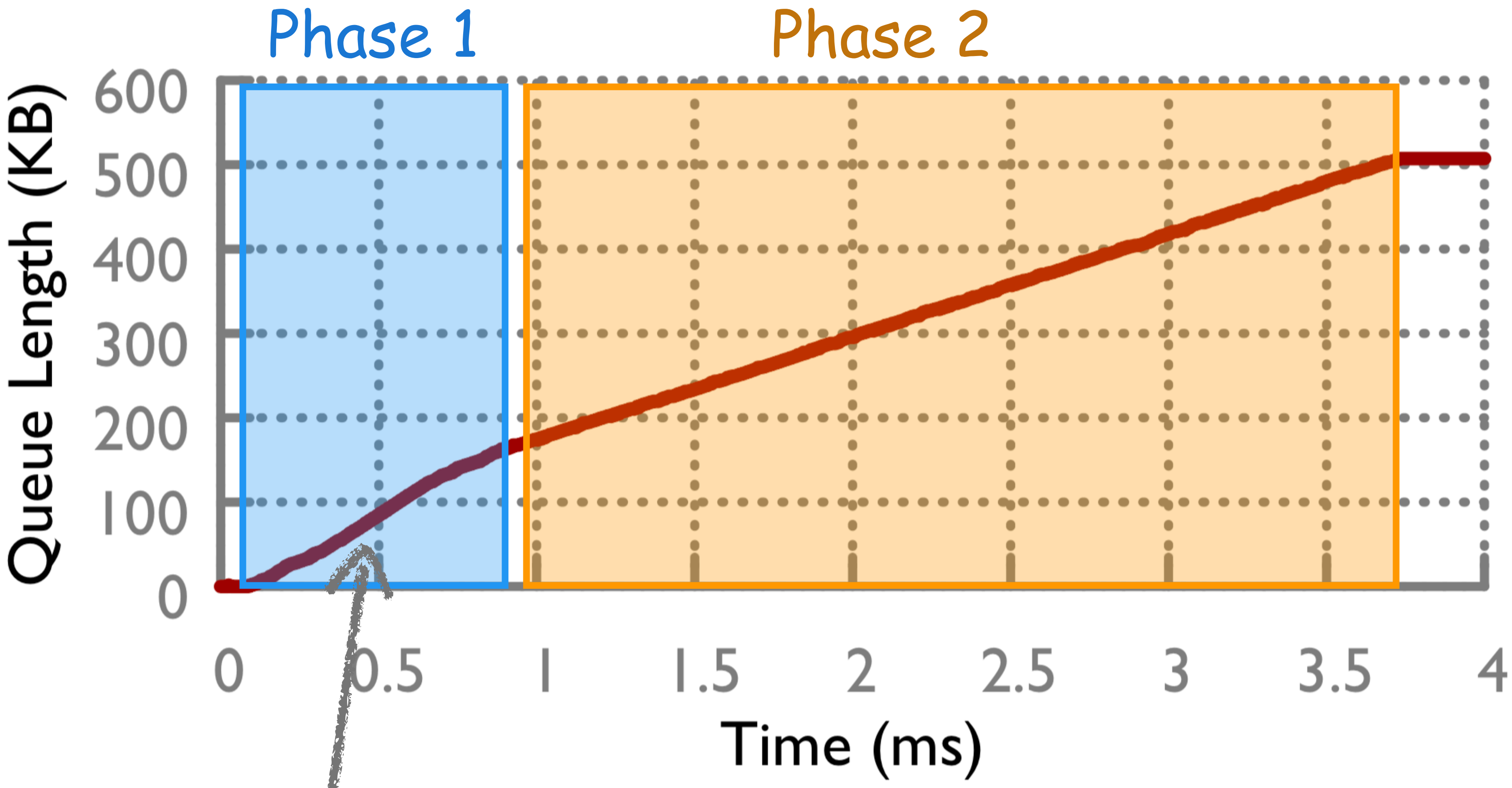


Queue Length Evolution

Experiment Results

Observations

— Synchronized fan-in



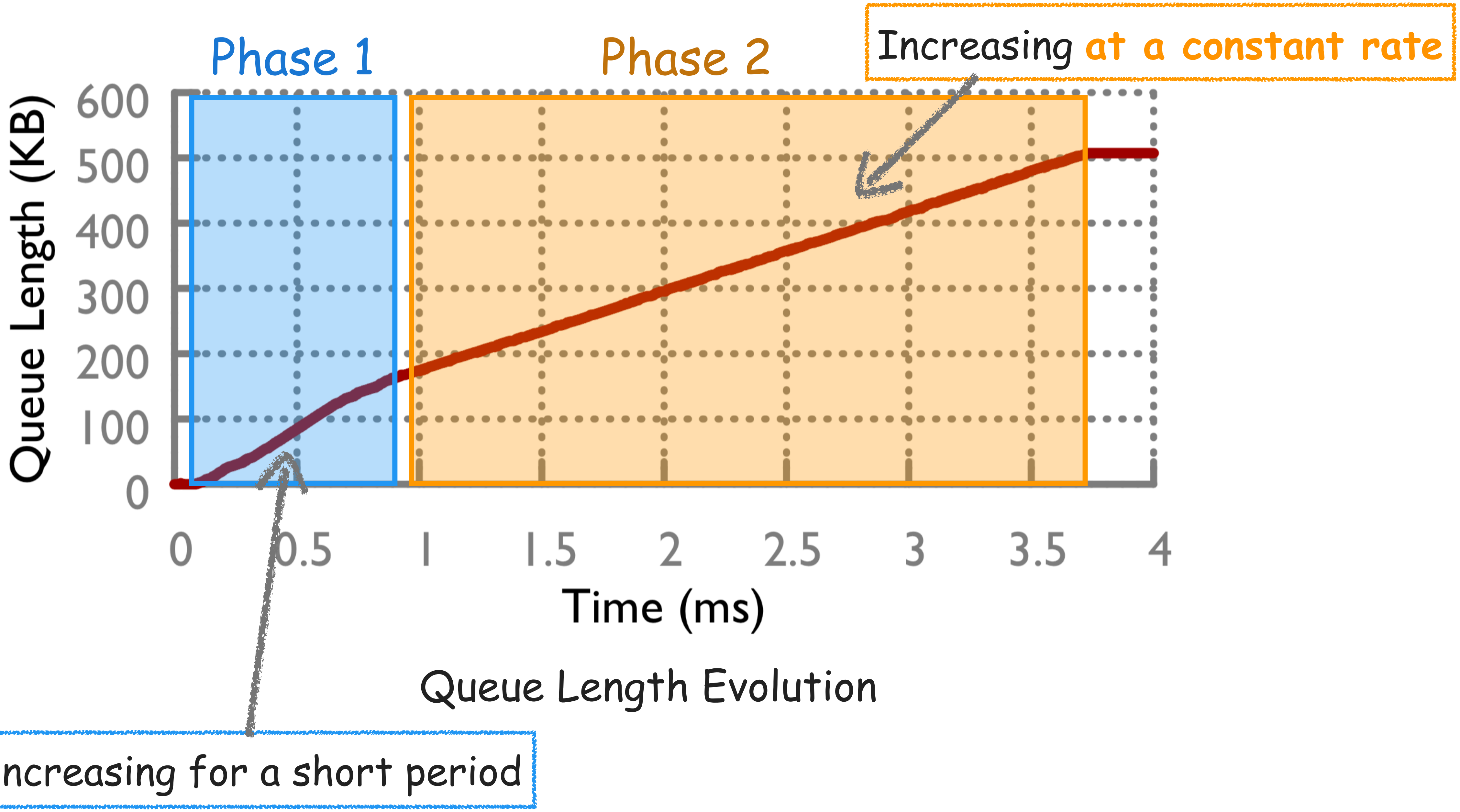
Queue Length Evolution

Sharply increasing for a short period

Experiment Results

Observations

— Synchronized fan-in



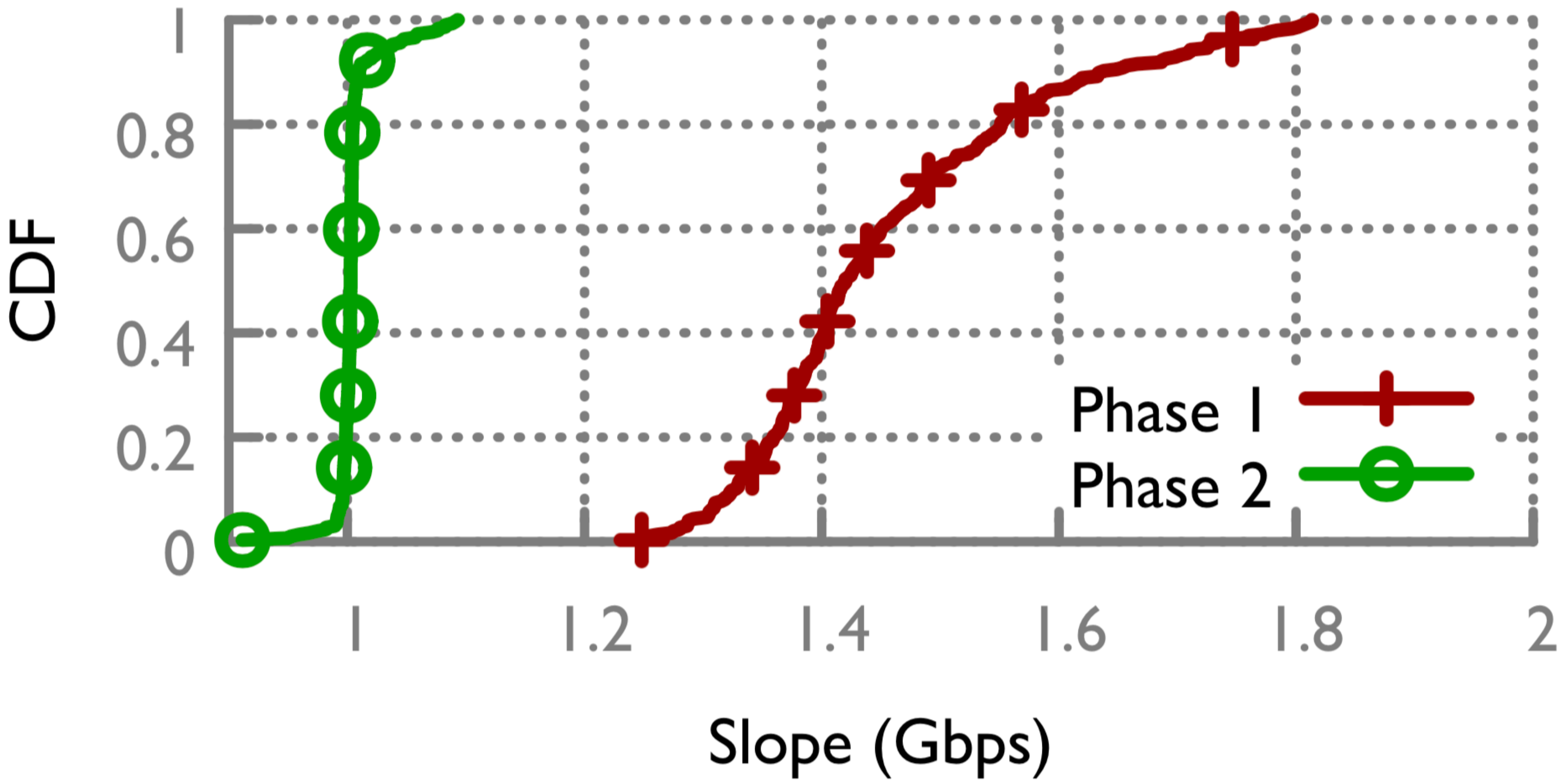
Queue Length Evolution

Experiment Results

Observations

— Synchronized fan-in

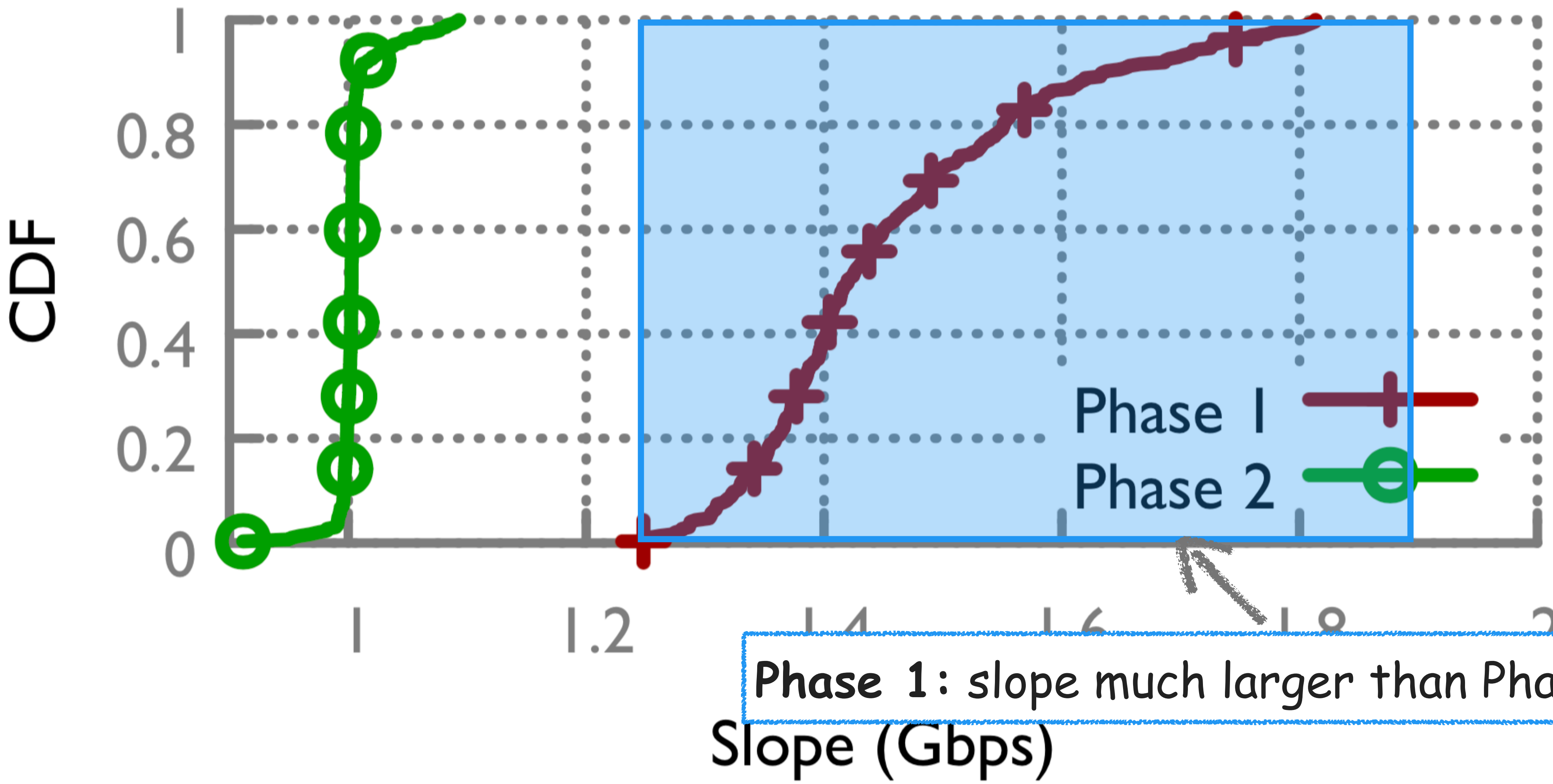
Slope: queue length increasing rate



Observations

— Synchronized fan-in

Slope: queue length increasing rate



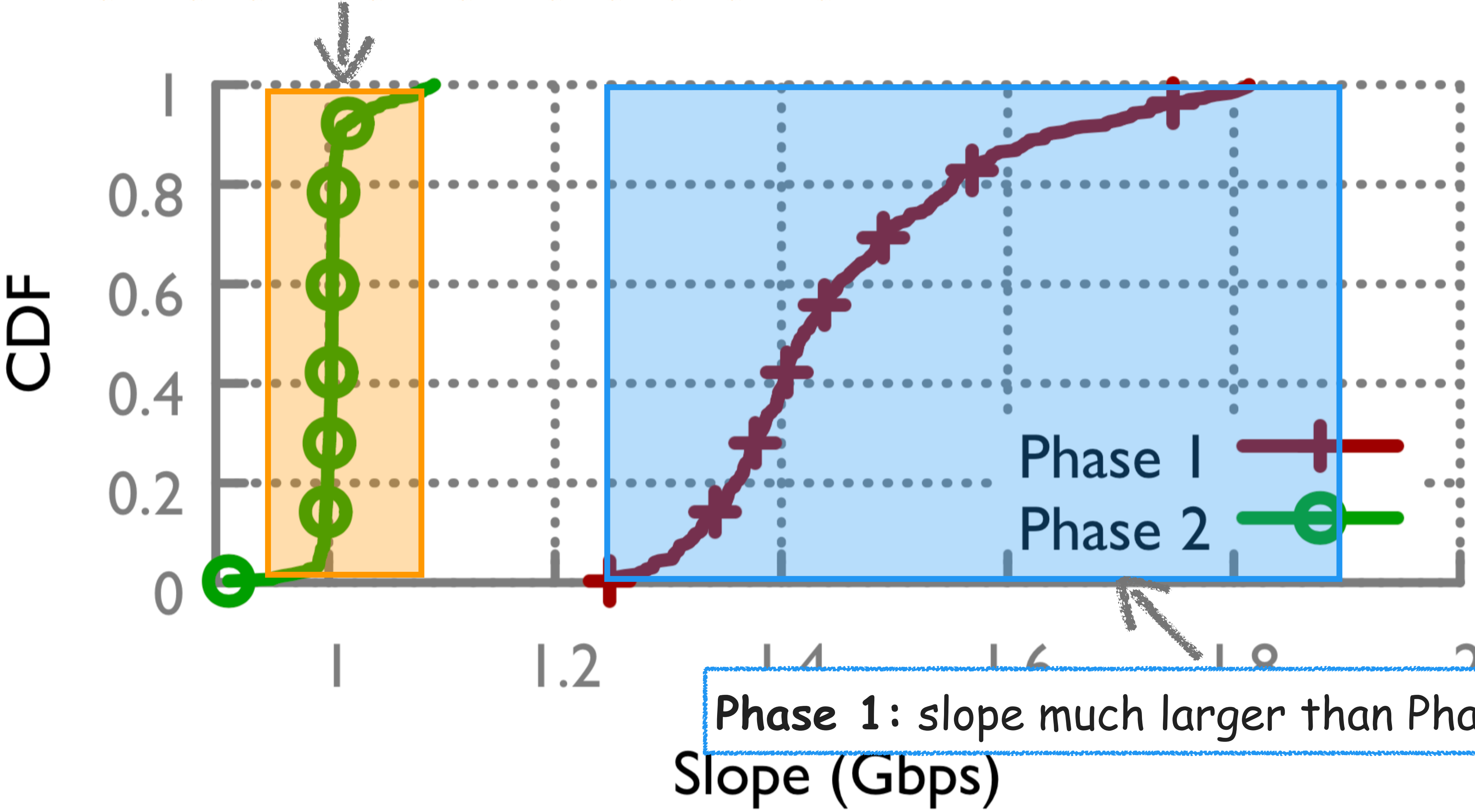
Phase 1: slope much larger than Phase 2

Observations

— Synchronized fan-in

Slope: queue length increasing rate

Phase 2: Slope = 1Gbps (bottleneck capacity)



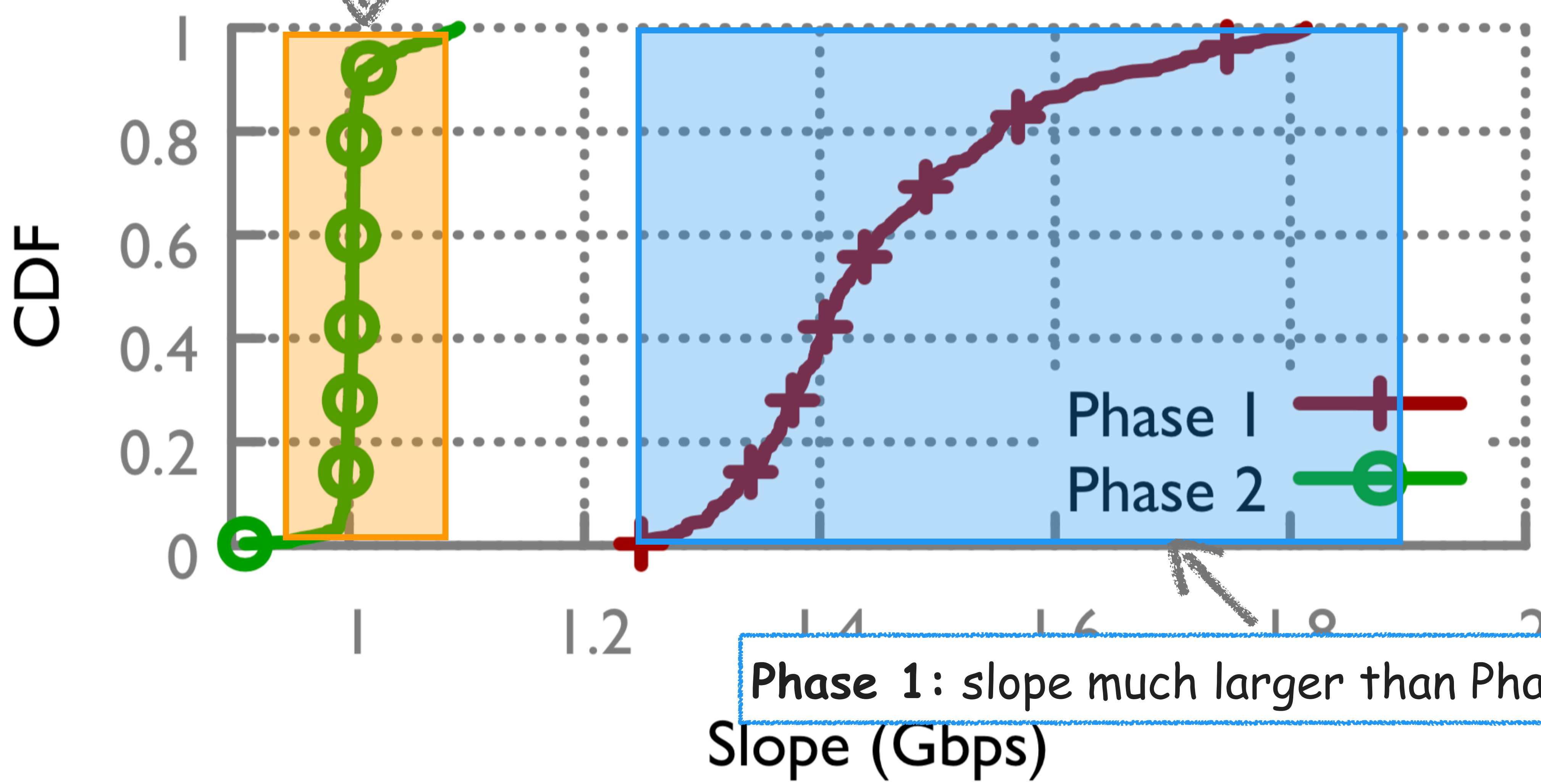
Phase 1: slope much larger than Phase 2

Observations

— Synchronized fan-in

Slope: queue length increasing rate

Phase 2: Slope = 1Gbps (bottleneck capacity)

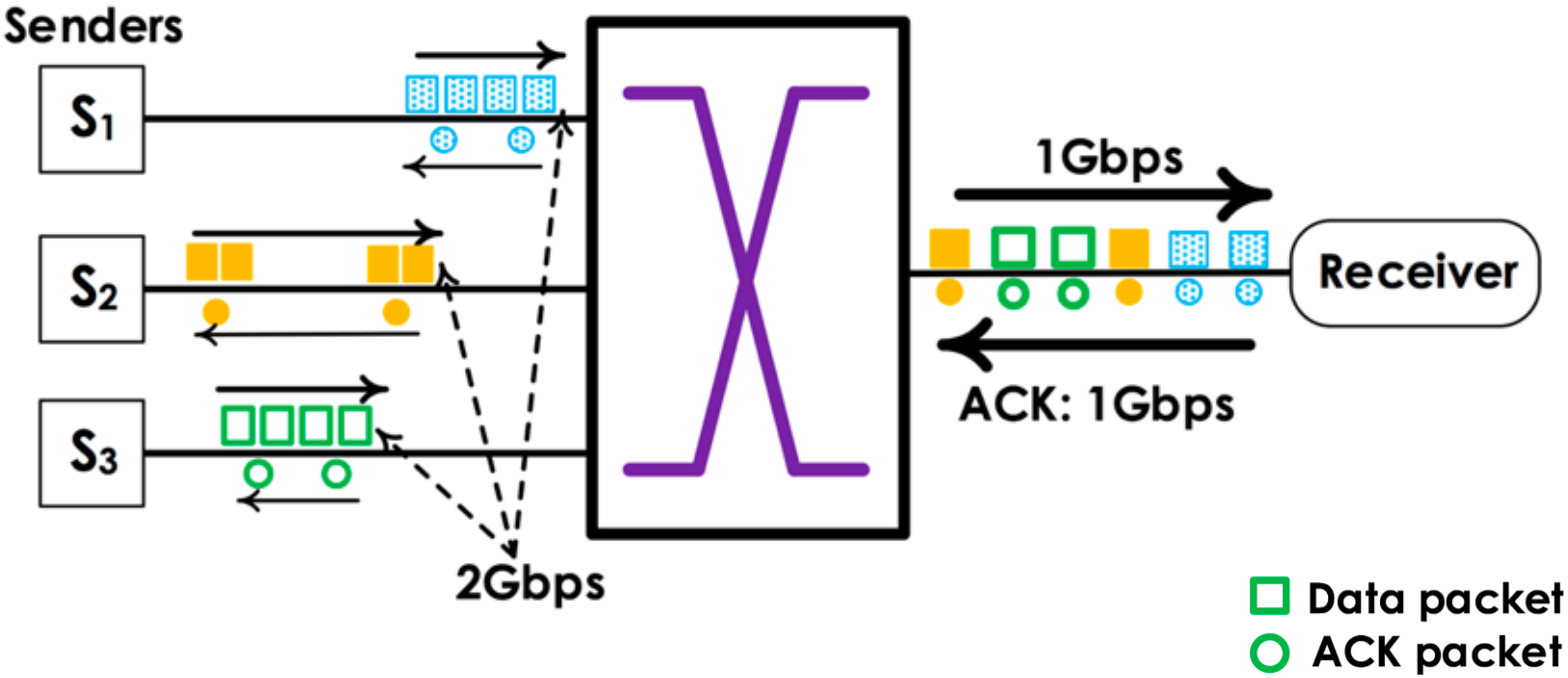


Phase 1: slope much larger than Phase 2

Why???

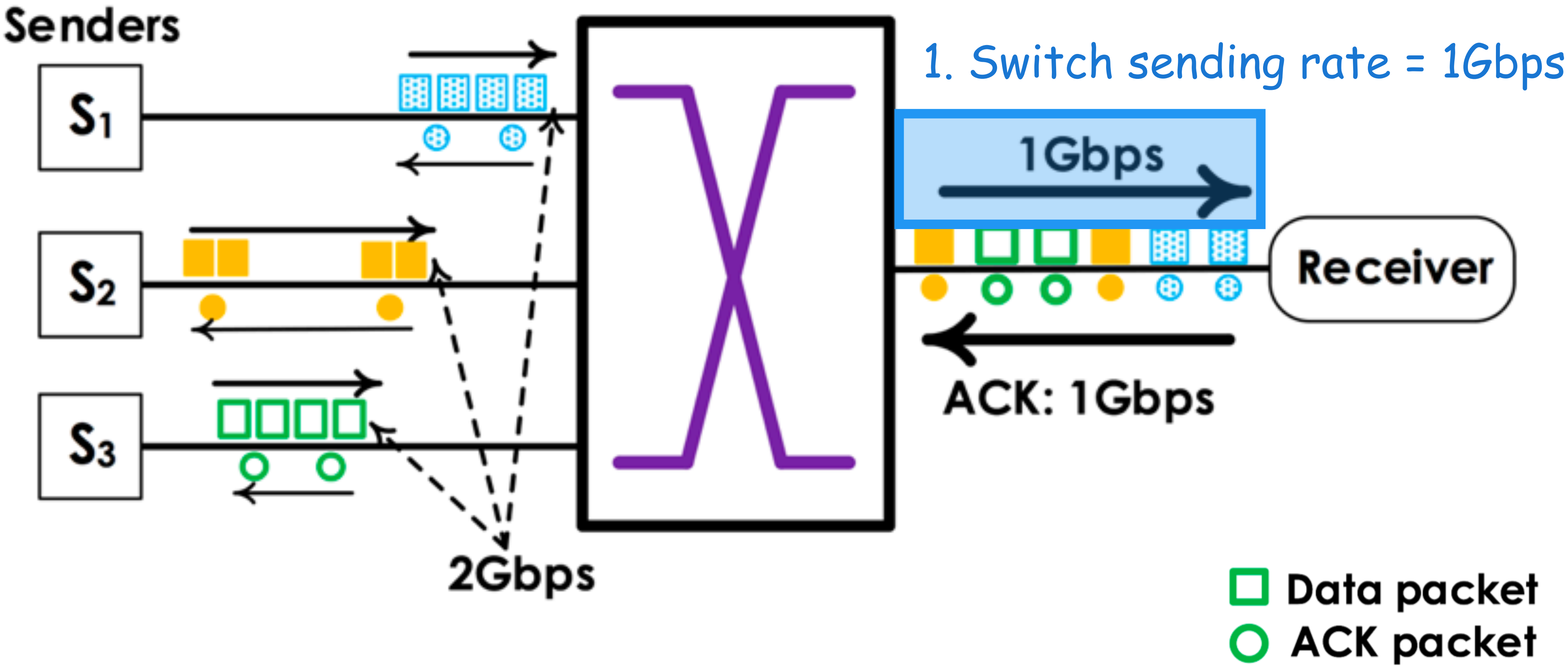
Observations

— Synchronized fan-in



Observations

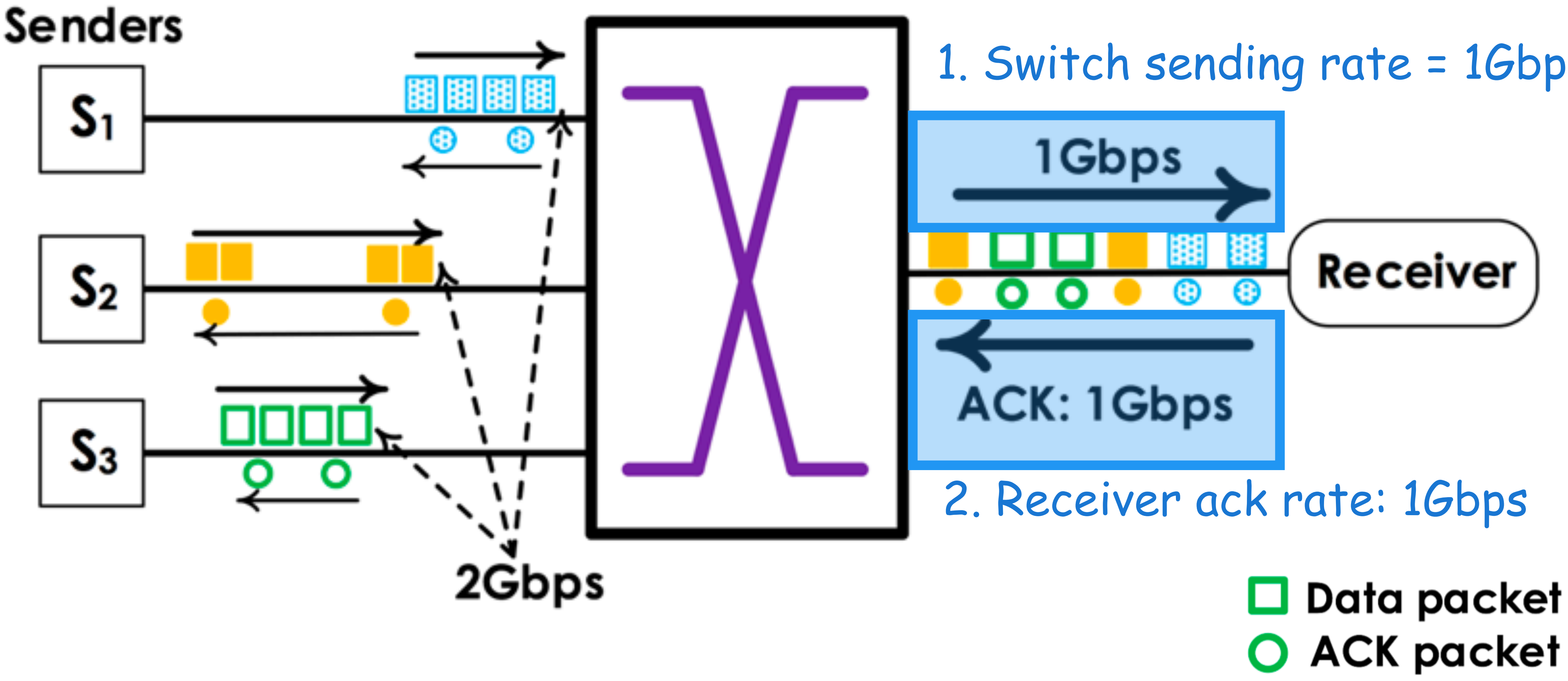
— Synchronized fan-in



1. Switch sending rate = 1Gbps

Observations

— Synchronized fan-in



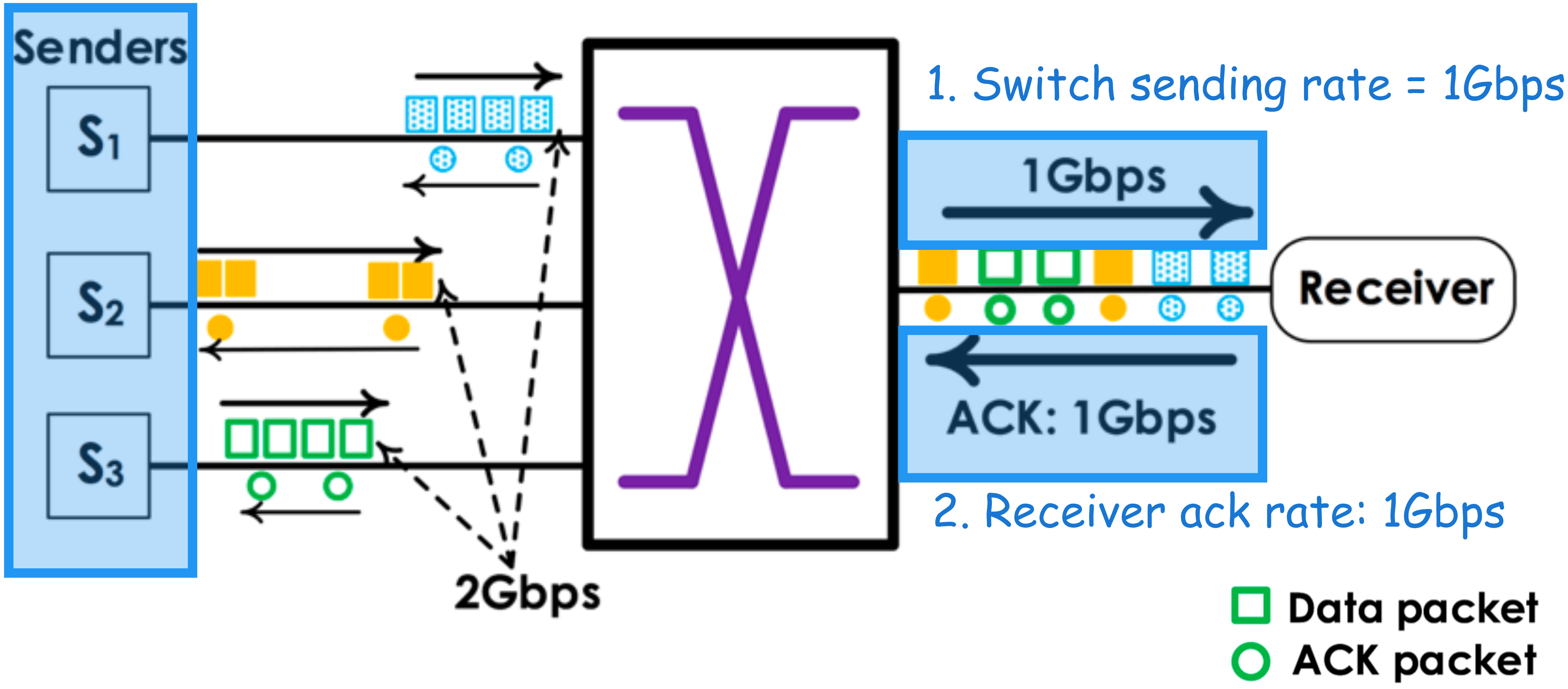
1. Switch sending rate = 1Gbps

2. Receiver ack rate: 1Gbps

Observations

— Synchronized fan-in

3. Sender: 1 ACK —> 2 Data packet
(slow start)



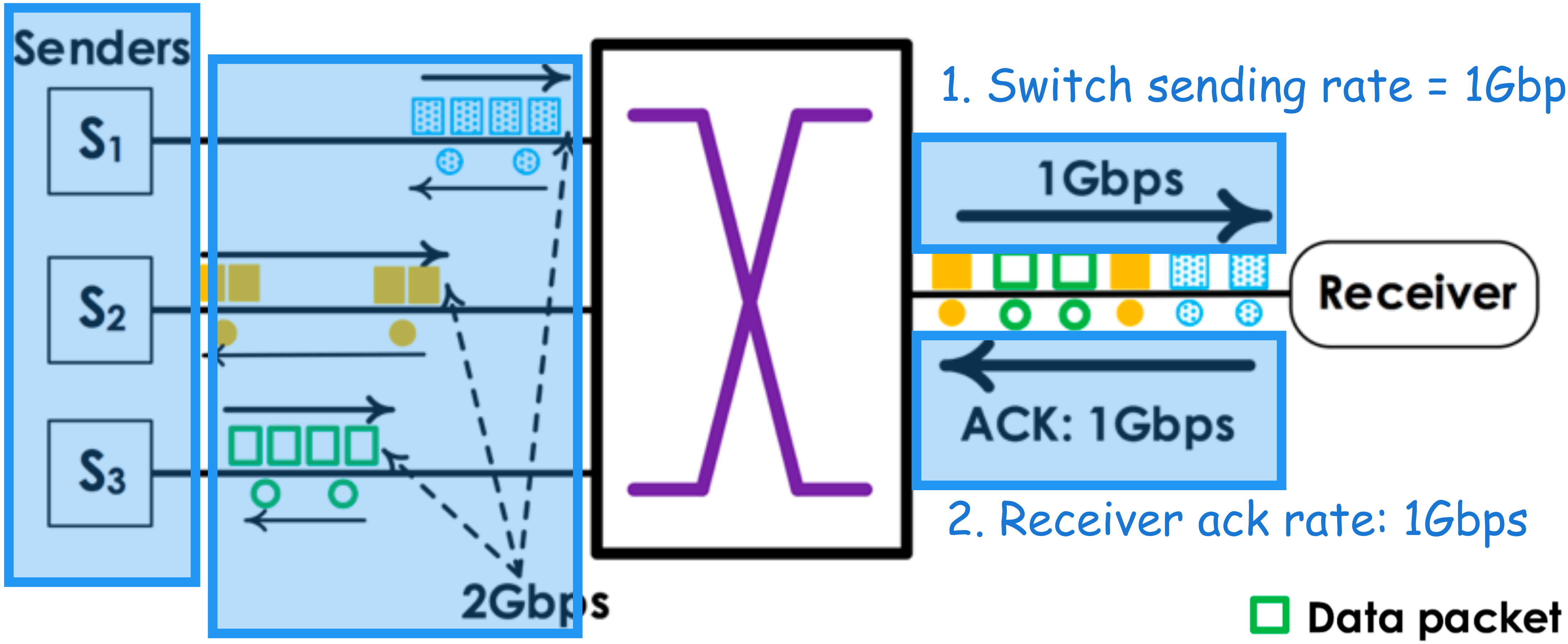
1. Switch sending rate = 1Gbps

2. Receiver ack rate: 1Gbps

Observations

— Synchronized fan-in

3. Sender: 1 ACK —> 2 Data packet
(slow start)



1. Switch sending rate = 1Gbps

2. Receiver ack rate: 1Gbps

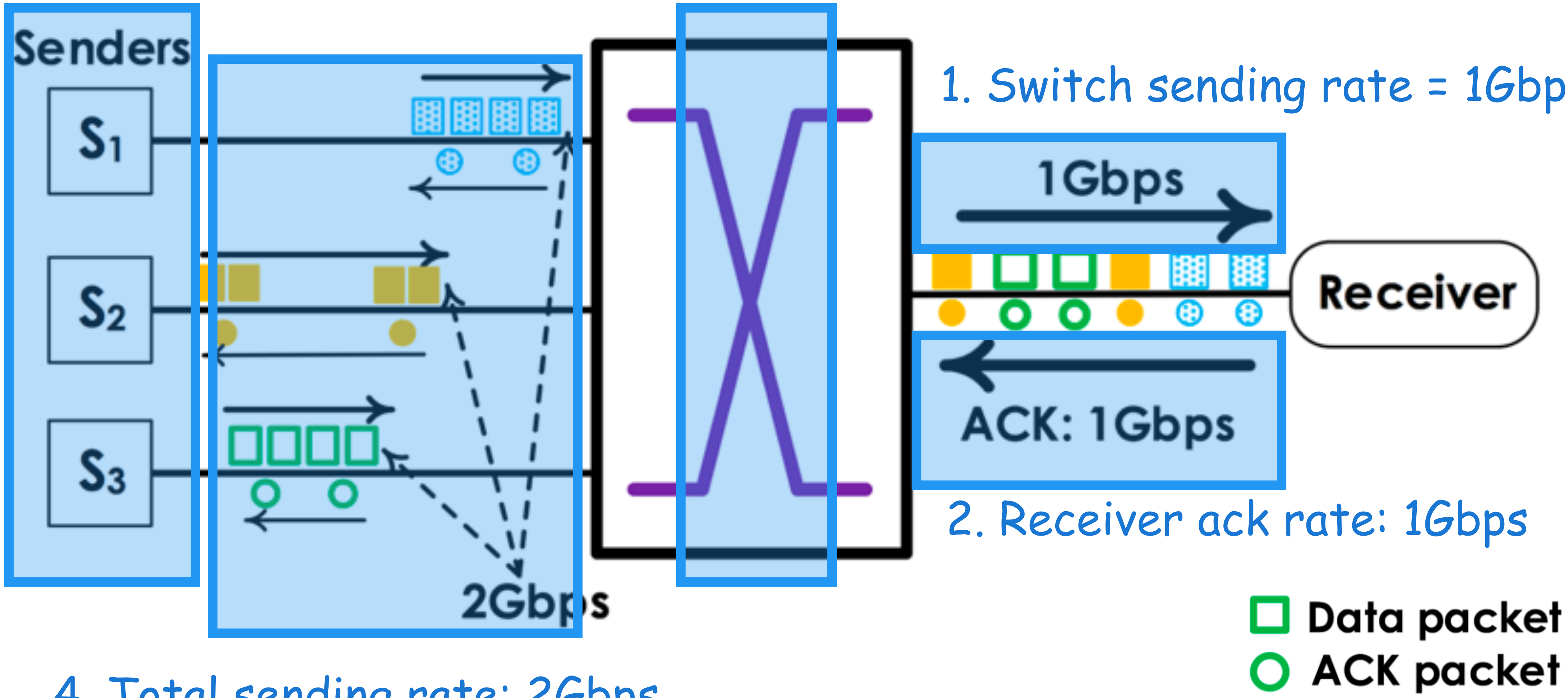
4. Total sending rate: 2Gbps

Observations

— Synchronized fan-in

3. Sender: 1 ACK —> 2 Data packet (slow start)

5. Qlen increasing rate: 1Gbps



1. Switch sending rate = 1Gbps

2. Receiver ack rate: 1Gbps

4. Total sending rate: 2Gbps

Observations

— Synchronized fan-in

Phase 2:
slope = 1Gbps

- **Bottleneck capacity** limits the receiving rate
- ***ACK-clocking system*** evenly spread the packets
- ***Congestion Control*** doubles the total sending rate

Observations

— Synchronized fan-in

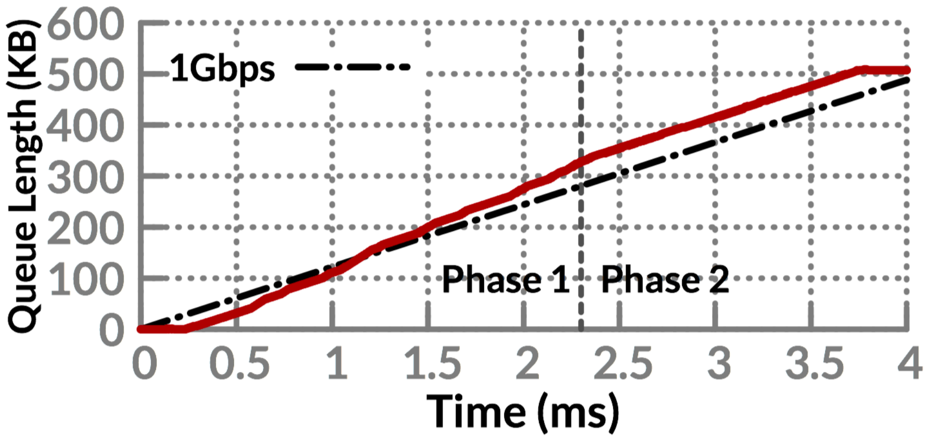
Phase 2:
slope = 1Gbps

- **Bottleneck capacity** limits the receiving rate
- ***ACK-clocking system*** evenly spread the packets
- ***Congestion Control*** doubles the total sending rate

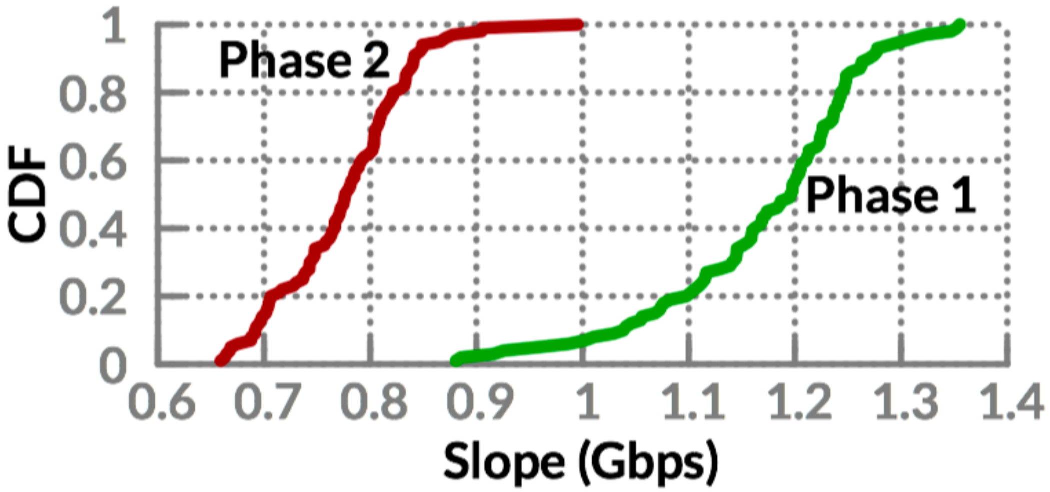
Phase 1:
slope larger than Phase 2

- Senders are sending **1st round** of packets
- Uncontrolled by self-clocking system

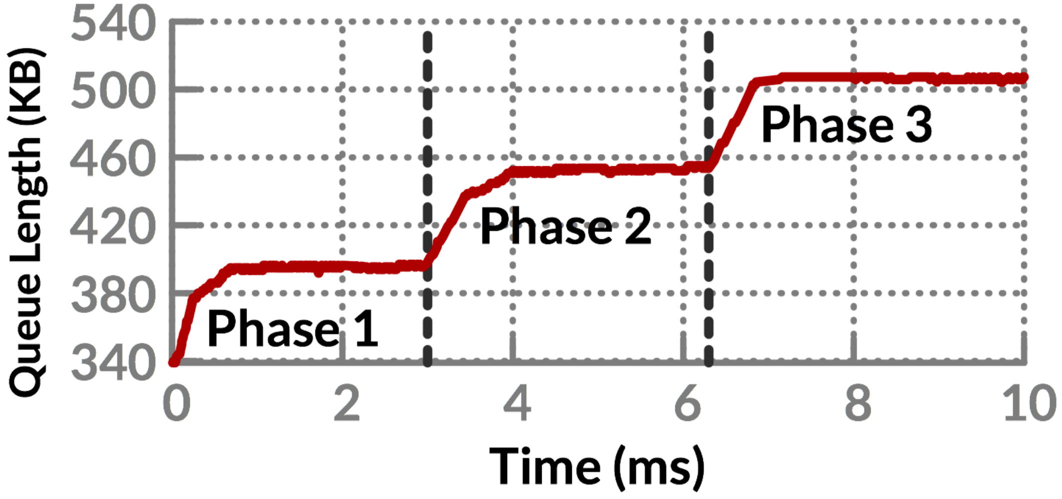
Observations



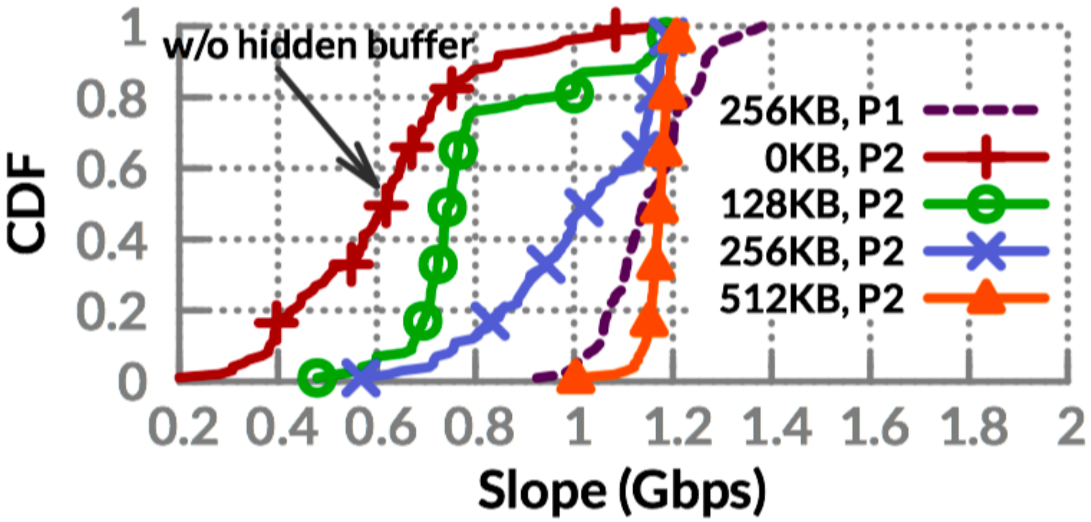
Asynchronous fan-in



w/ one background flow



w/ several background flows
congested at the same hop



w/ several background flows
congested at previous hop

Summary of Observations

Phase 2 Behavior

1. Without background flows
   - **Slope** = bottleneck capacity
2. With one background flow, or several background flows congested at the same hop
   - **Slope** < bottleneck capacity
3. With several background flows congested at previous hop
   - **slope** > bottleneck capacity
   - **Slope** <= 2*bottleneck capacity

Summary of Observations

Phase 2 Behavior

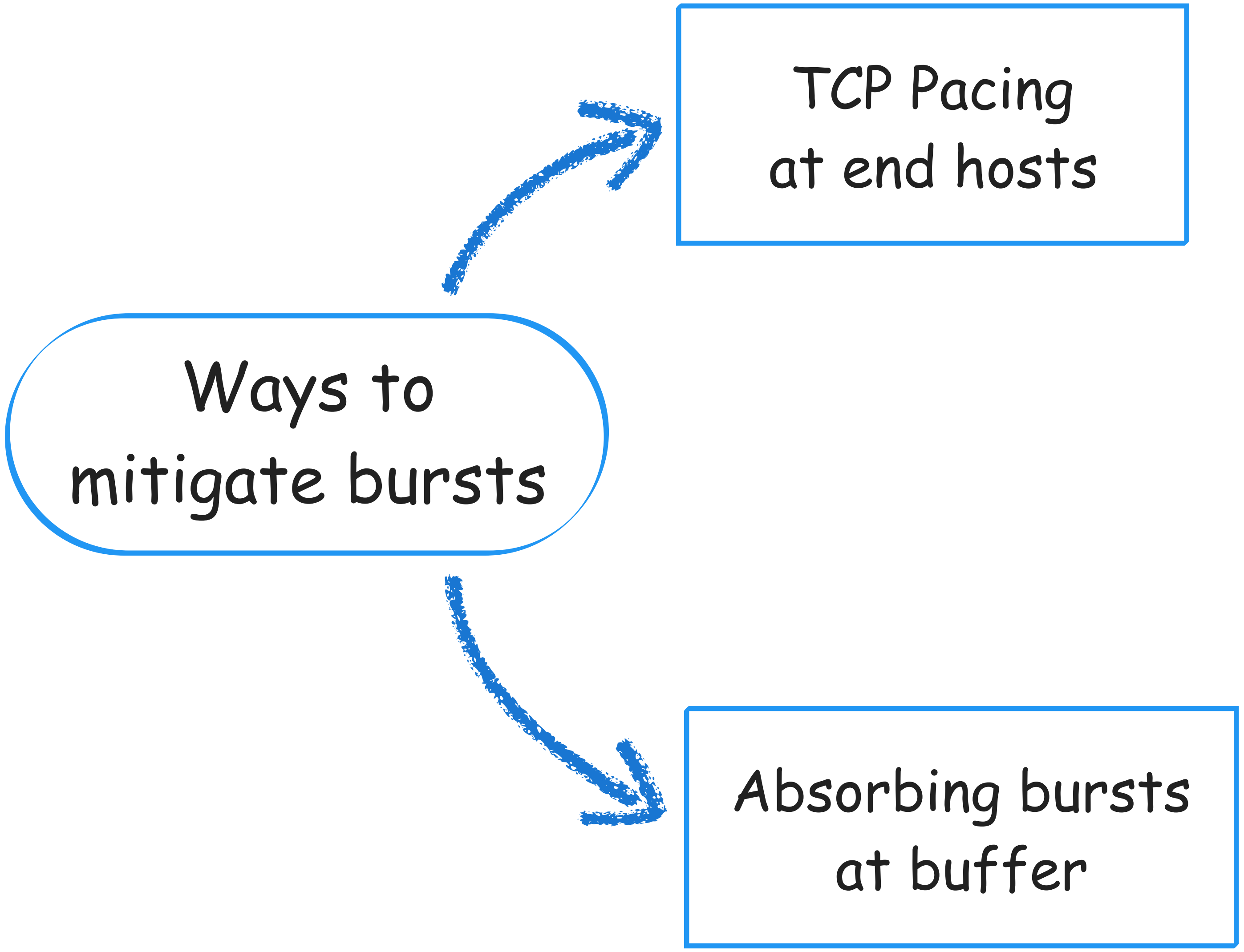
1. Without background flows
   - **Slope** = bottleneck capacity
2. With one background flow, or several background flows congested at the same hop
   - **Slope** < bottleneck capacity
3. With several background flows congested at previous hop
   - **slope** > bottleneck capacity
   - **Slope** <= 2*bottleneck capacity

Slope describes the dynamic behavior of micro-bursts

Implications

Ways to mitigate bursts

Implications

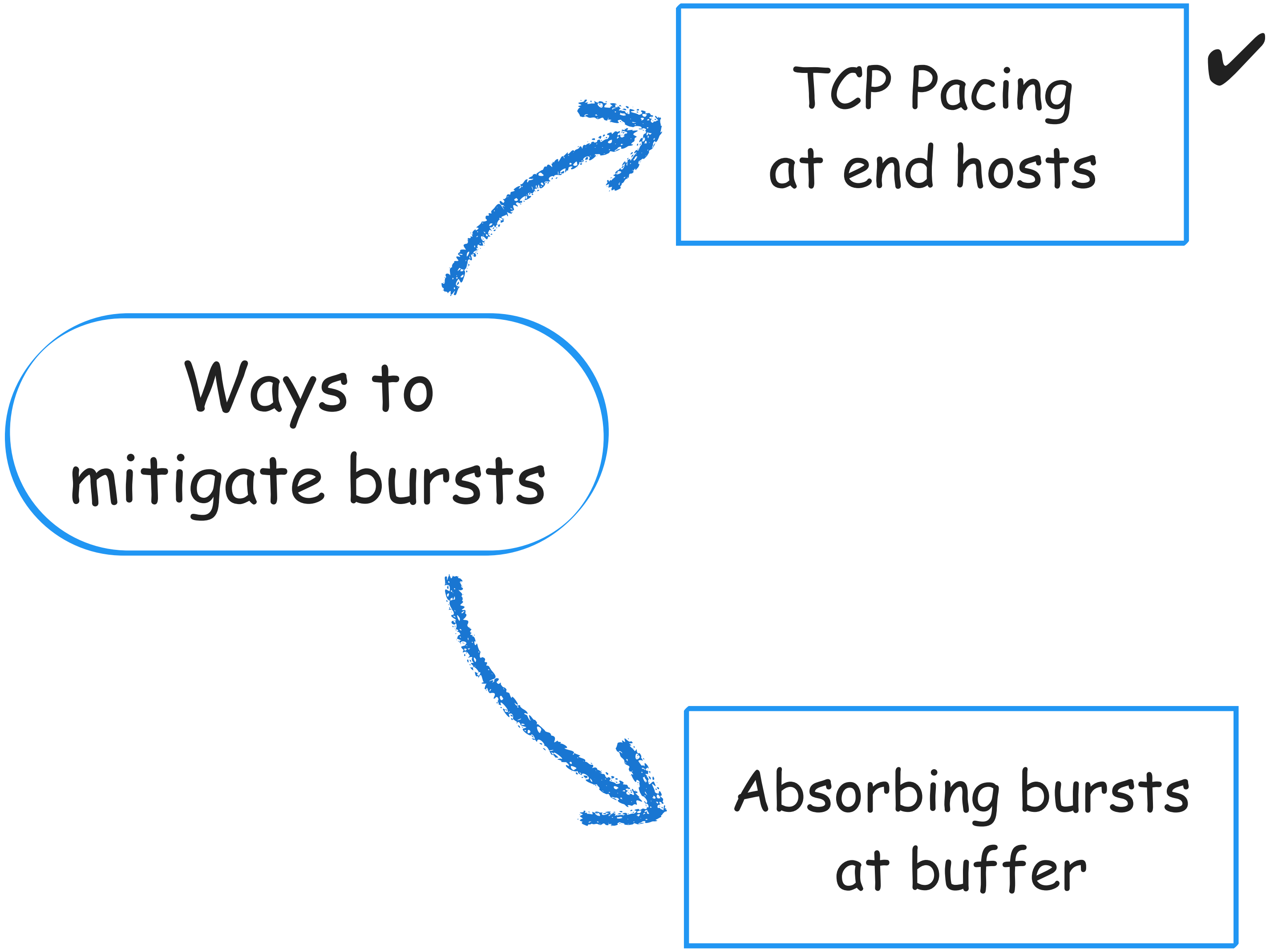


Ways to mitigate bursts

TCP Pacing at end hosts

Absorbing bursts at buffer

Implications

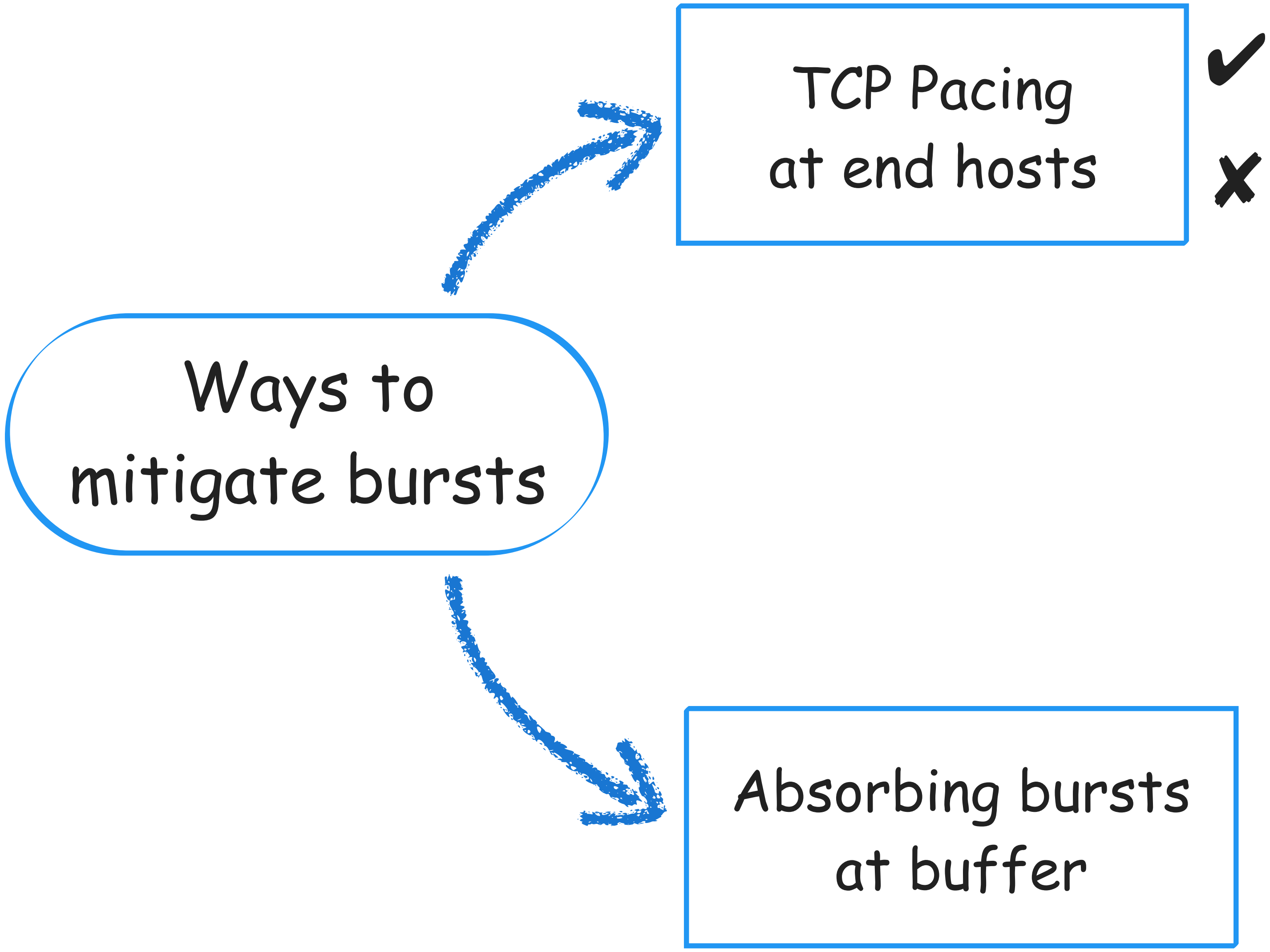

Ways to mitigate bursts

TCP Pacing at end hosts

Absorbing bursts at buffer

✔ burstiness inside a single flow

Implications



Ways to mitigate bursts

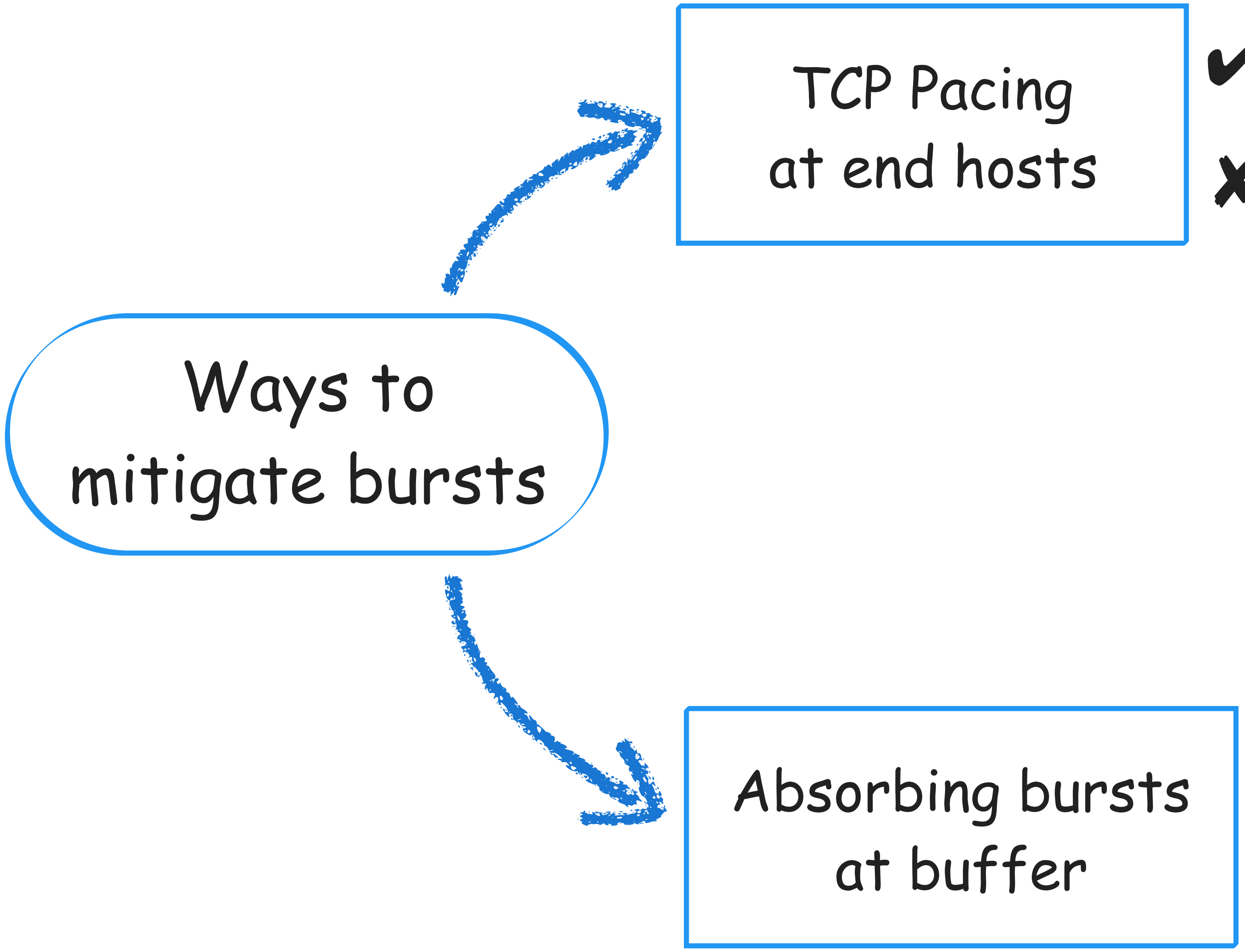
TCP Pacing at end hosts

✔ burstiness inside a single flow

✘ fan-in: burstiness from multiple flows

Absorbing bursts at buffer

Implications



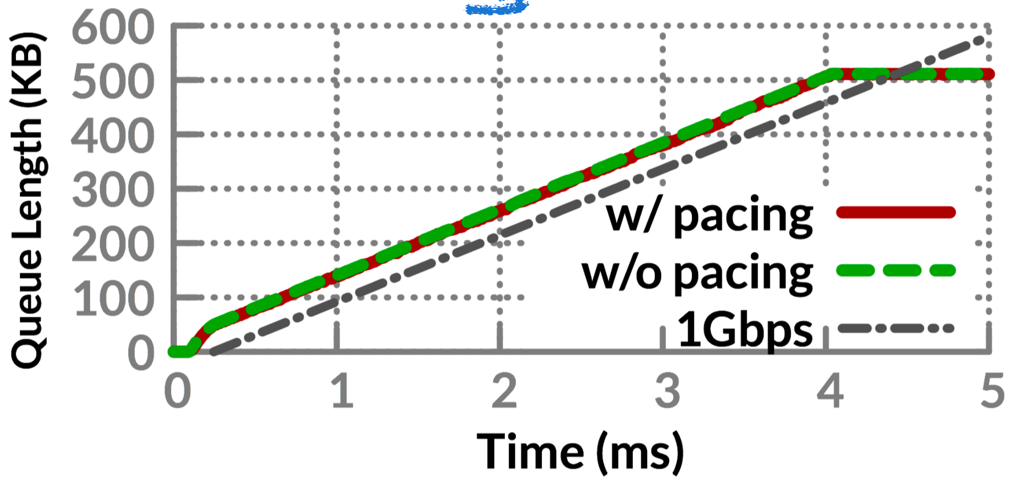
Ways to mitigate bursts

TCP Pacing at end hosts

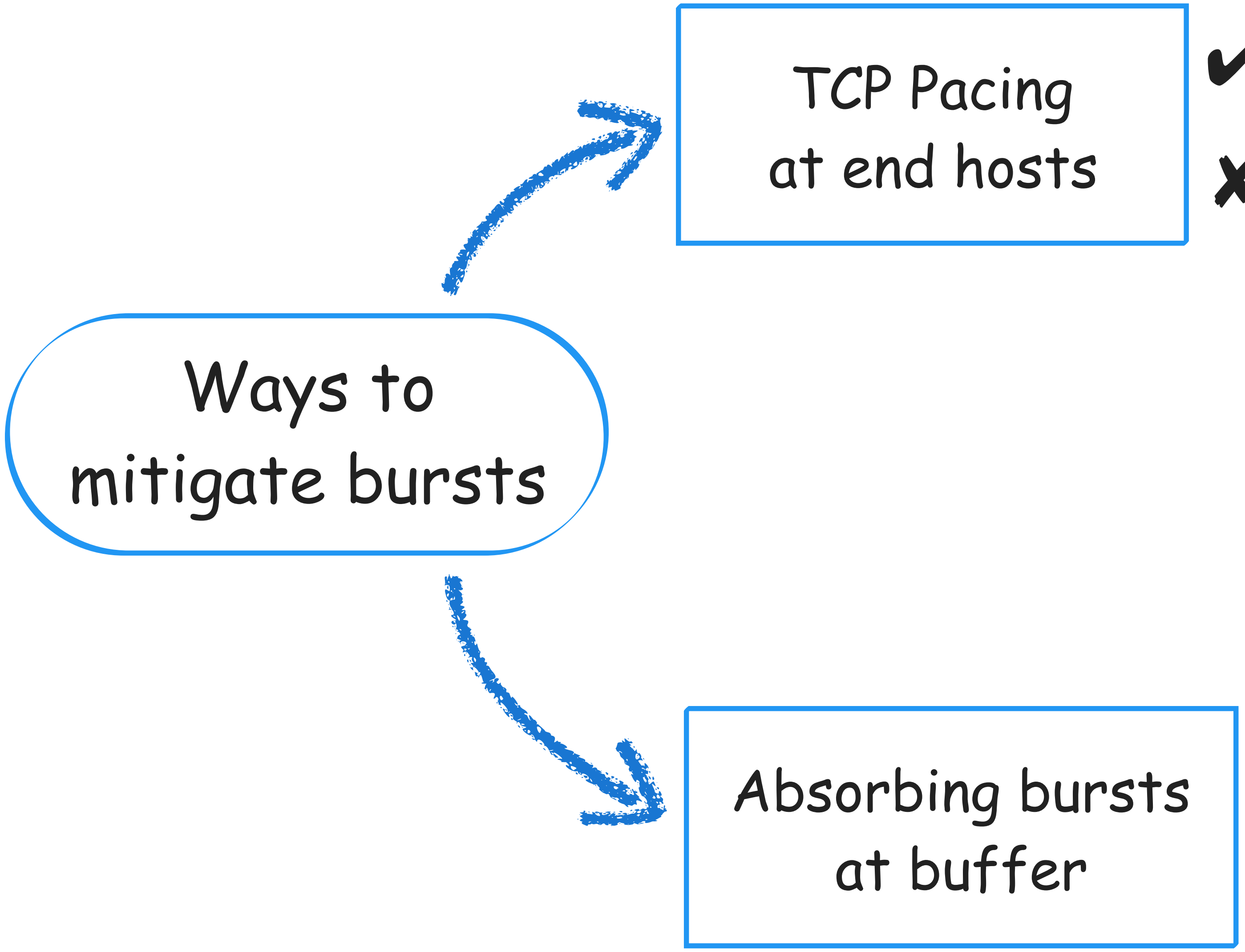
Absorbing bursts at buffer

✔ burstiness inside a single flow

✘ fan-in: burstiness from multiple flows



Implications



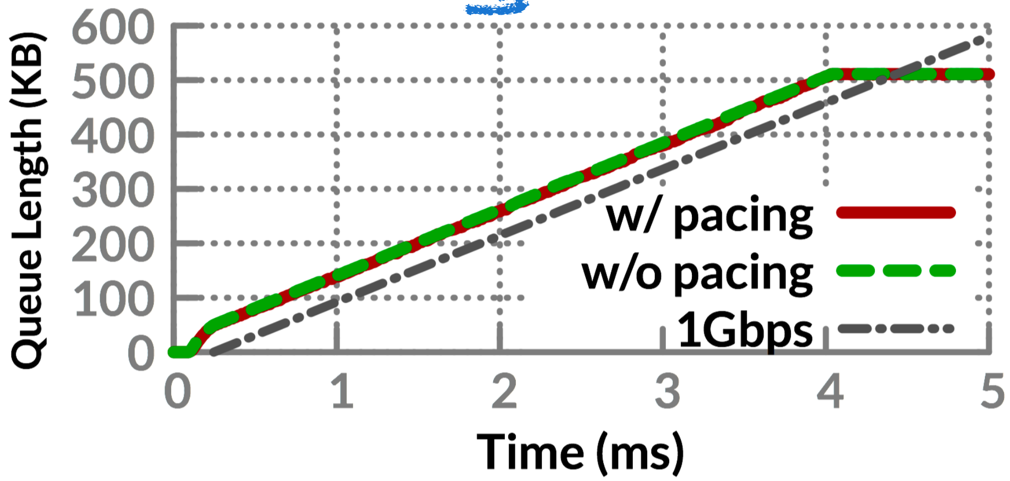
Ways to mitigate bursts

TCP Pacing at end hosts

Absorbing bursts at buffer

✔ burstiness inside a single flow

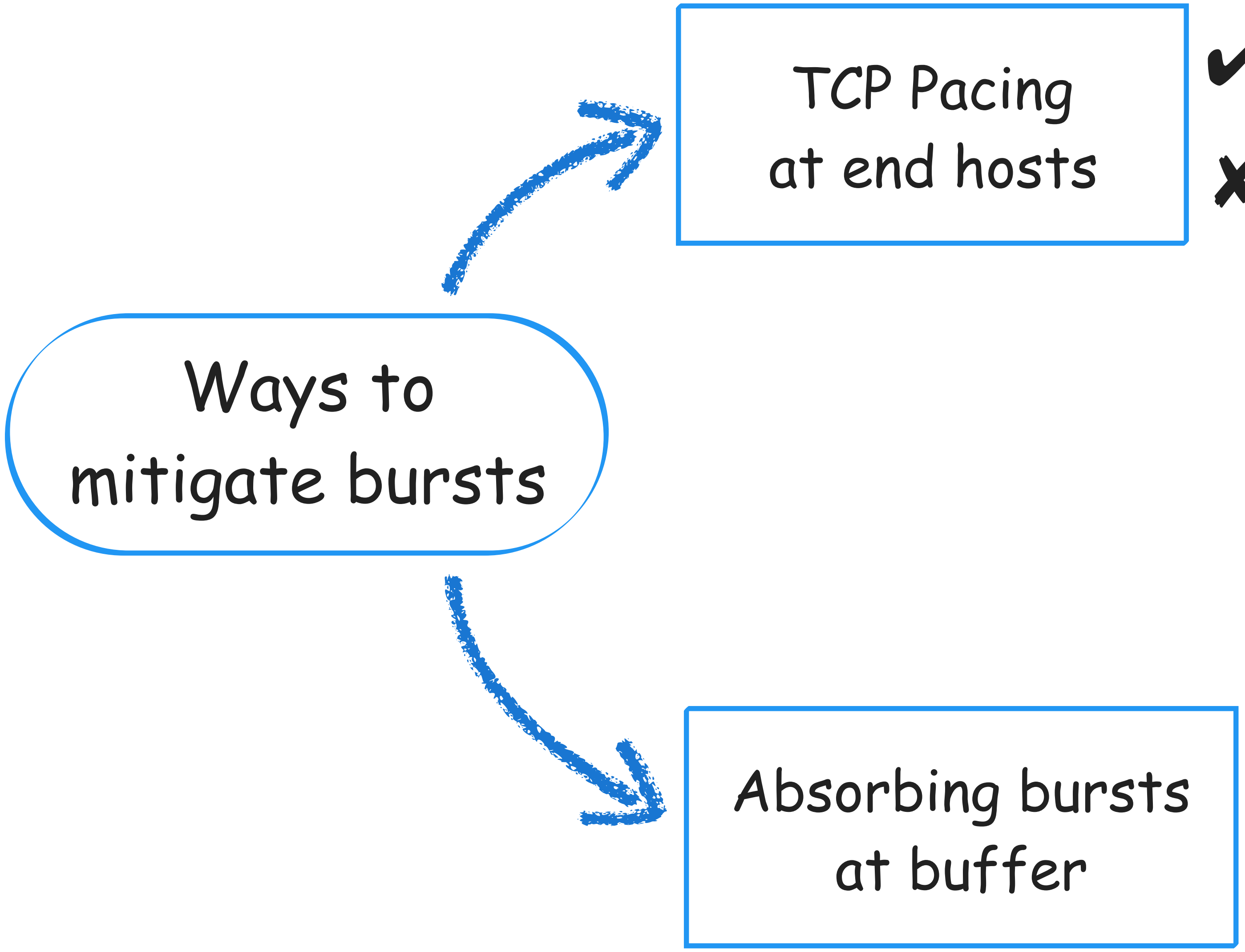
✘ fan-in: burstiness from multiple flows



Absorb one pkt —> another two pkt

🙁

Implications

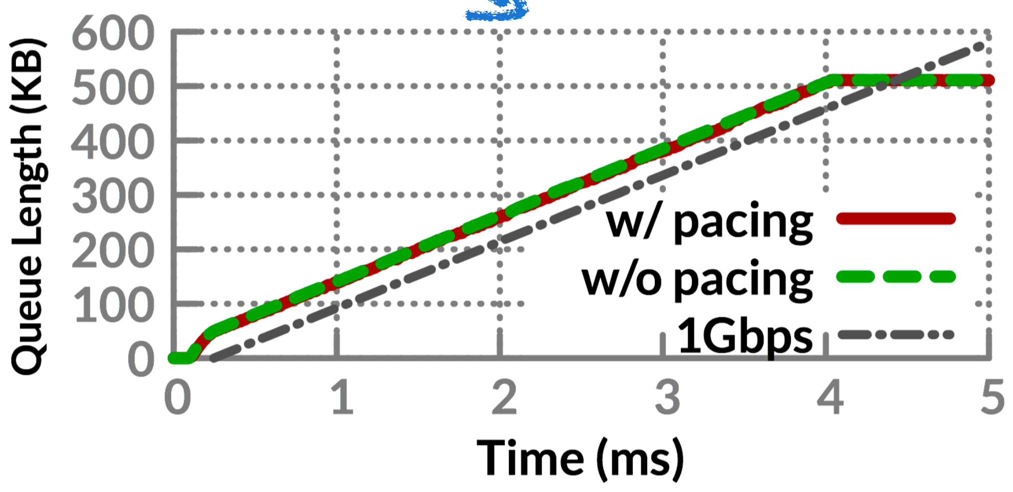


Ways to mitigate bursts

TCP Pacing at end hosts

✔ burstiness inside a single flow

✘ fan-in: burstiness from multiple flows



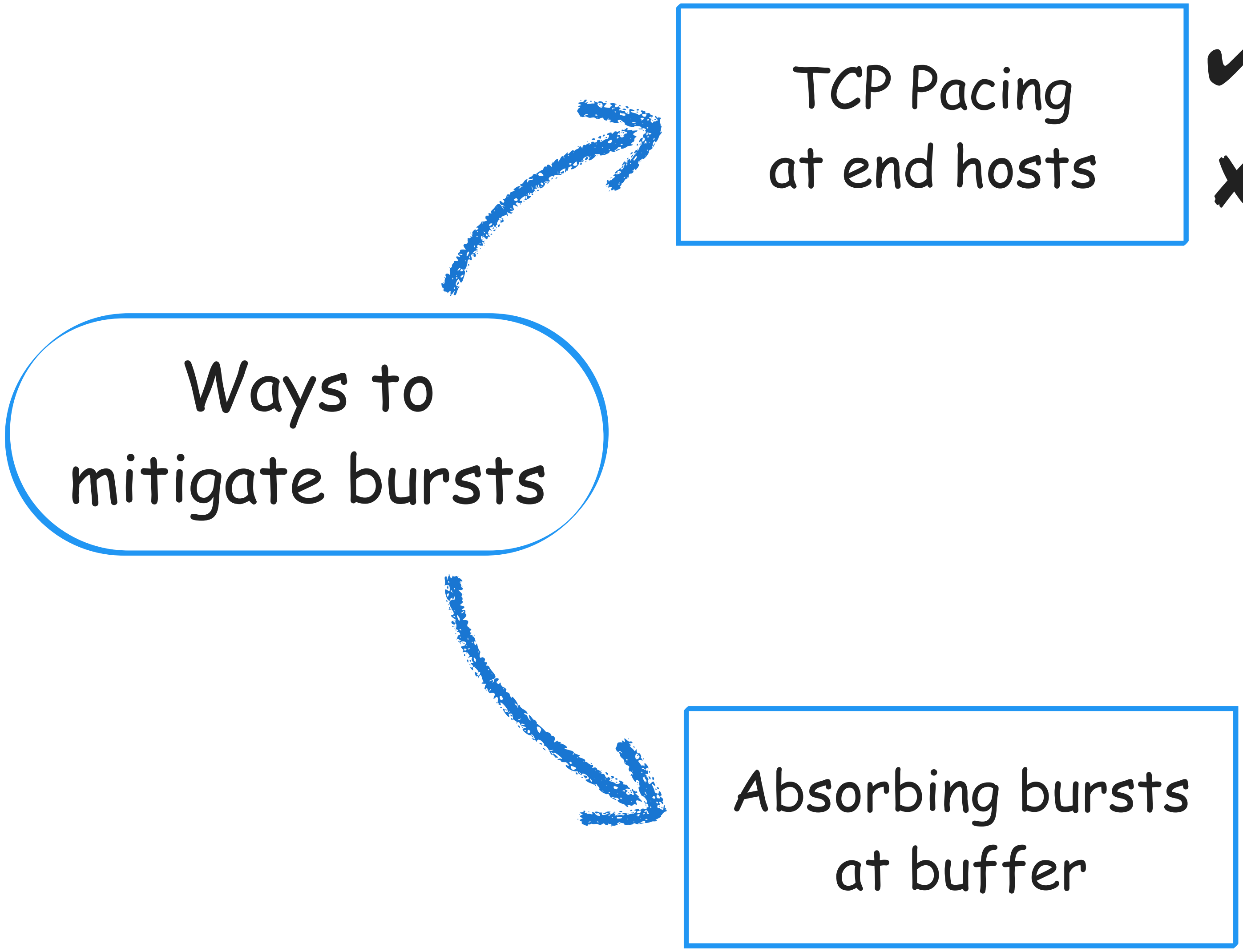
Absorbing bursts at buffer

Absorb one pkt —> another two pkt

🙁

How to mitigate micro-bursts?

Implications



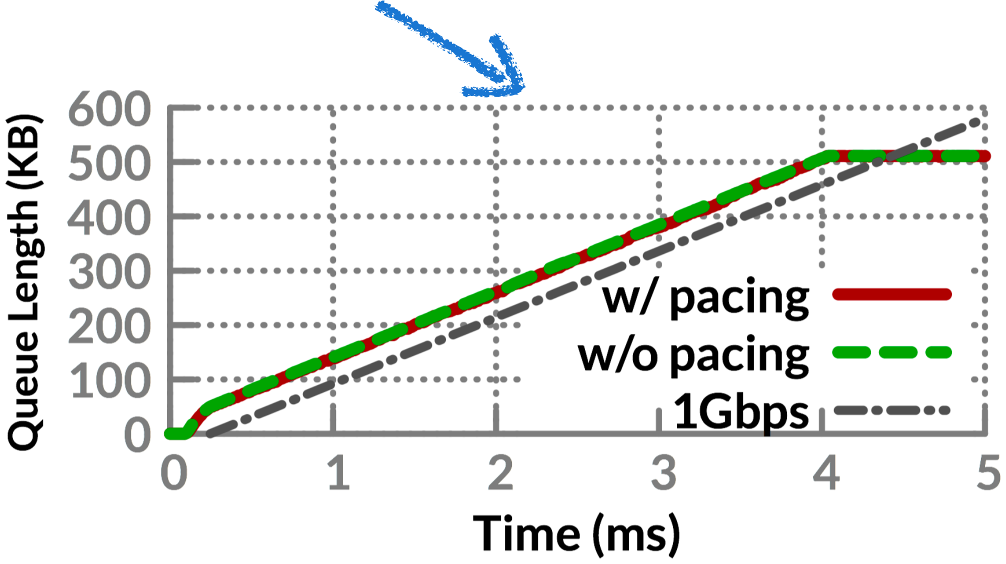
Ways to mitigate bursts

TCP Pacing at end hosts

Absorbing bursts at buffer

✔ burstiness inside a single flow

✘ fan-in: burstiness from multiple flows



Absorb one pkt —> another two pkt

🙁

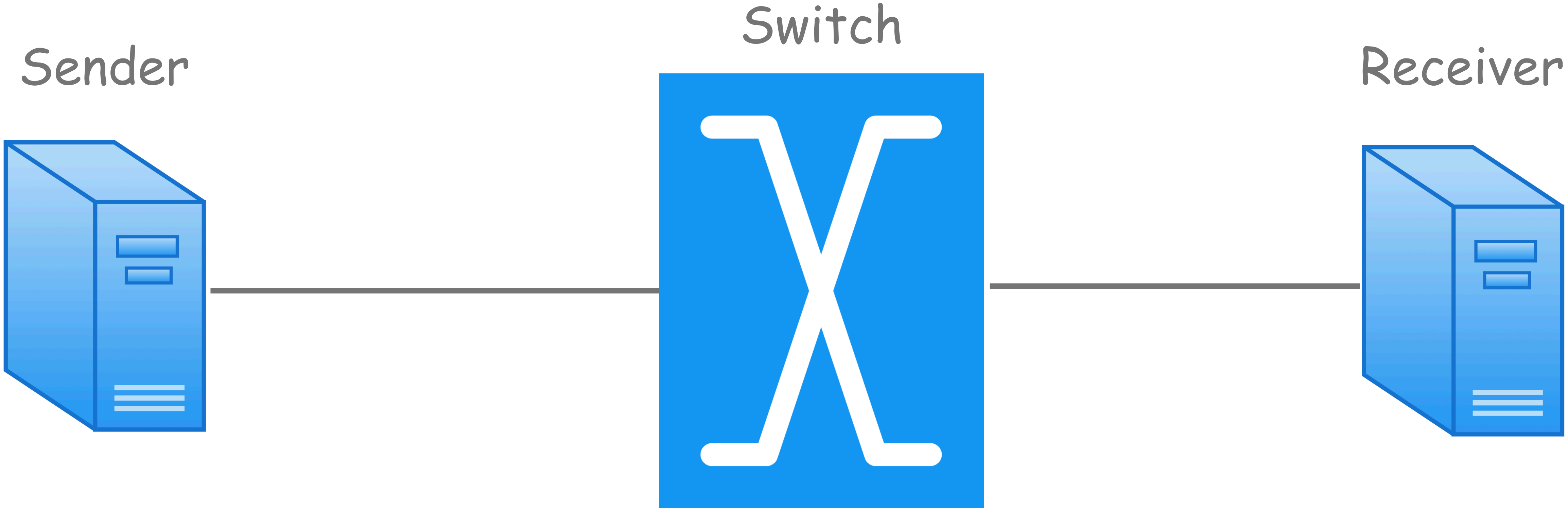
How to mitigate micro-bursts? → **Senders slow down in time**

Outline

- Background
- Methodology of Observing Micro-bursts
- Observing and Analyzing Micro-bursts
- Mitigating Micro-bursts
- Conclusion

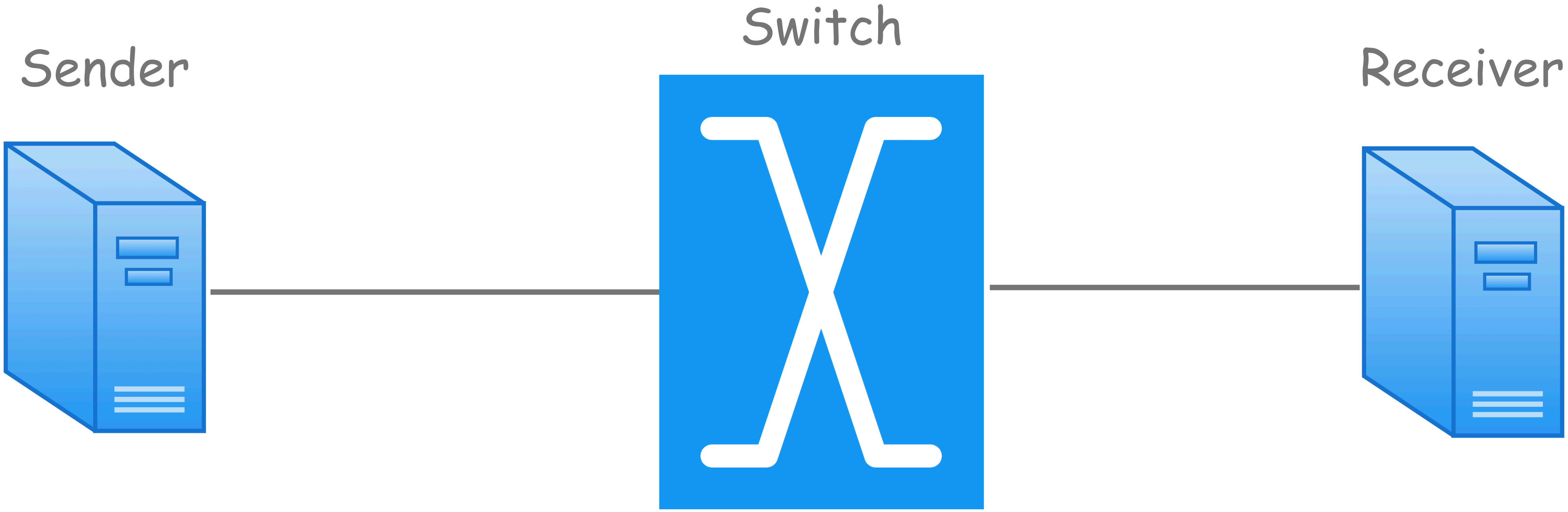
Mitigating Micro-bursts

How to mitigate micro-bursts?



Mitigating Micro-bursts

How to mitigate micro-bursts?

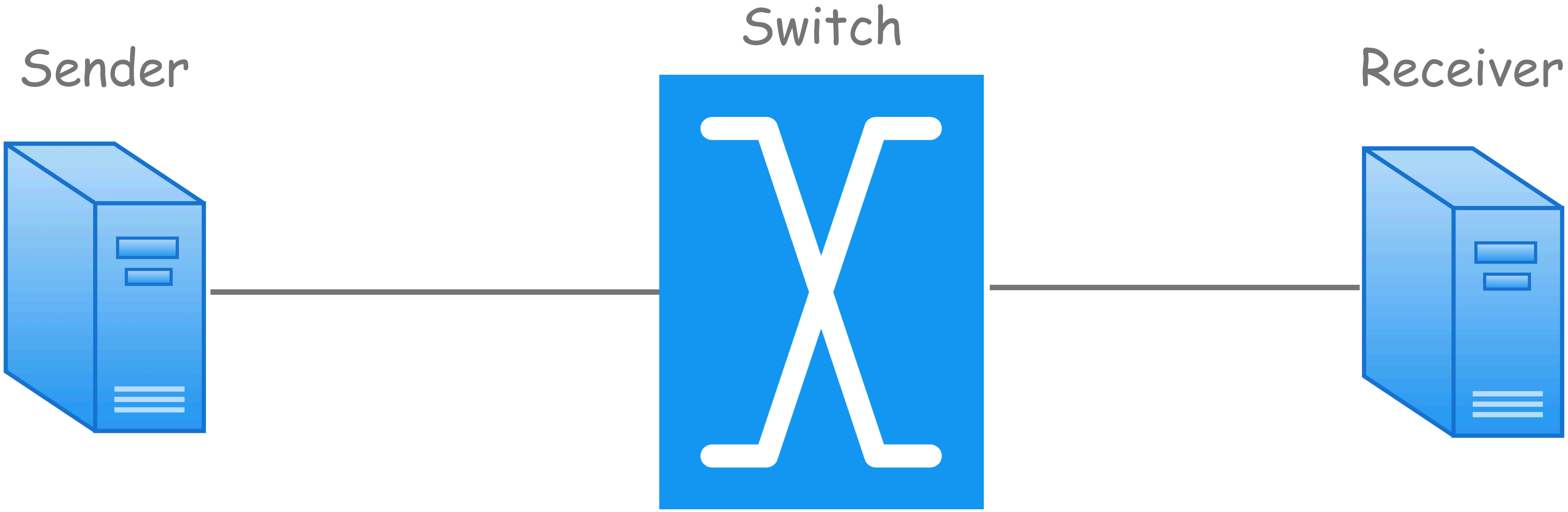


Switch:

Notify senders as soon as possible

Mitigating Micro-bursts

How to mitigate micro-bursts?



Senders: ←

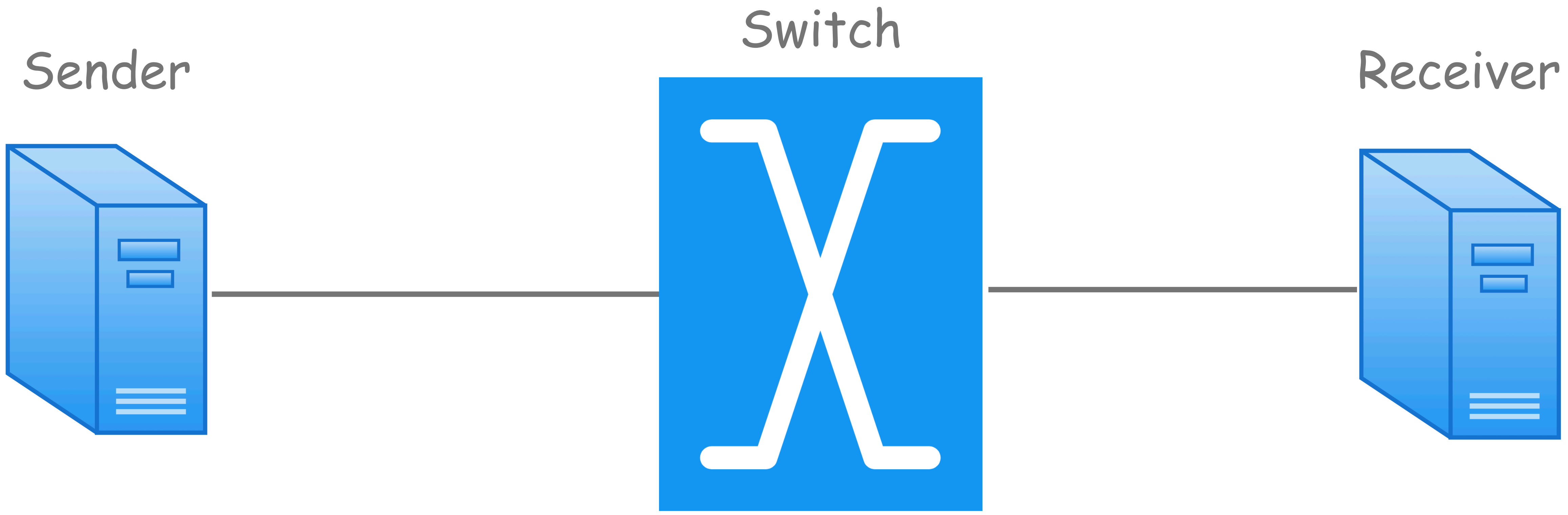
Slow down

Switch:

Notify senders as soon as possible

Mitigating Micro-bursts

How to mitigate micro-bursts?



Sender

Switch

Receiver

Senders: ←

Slow down

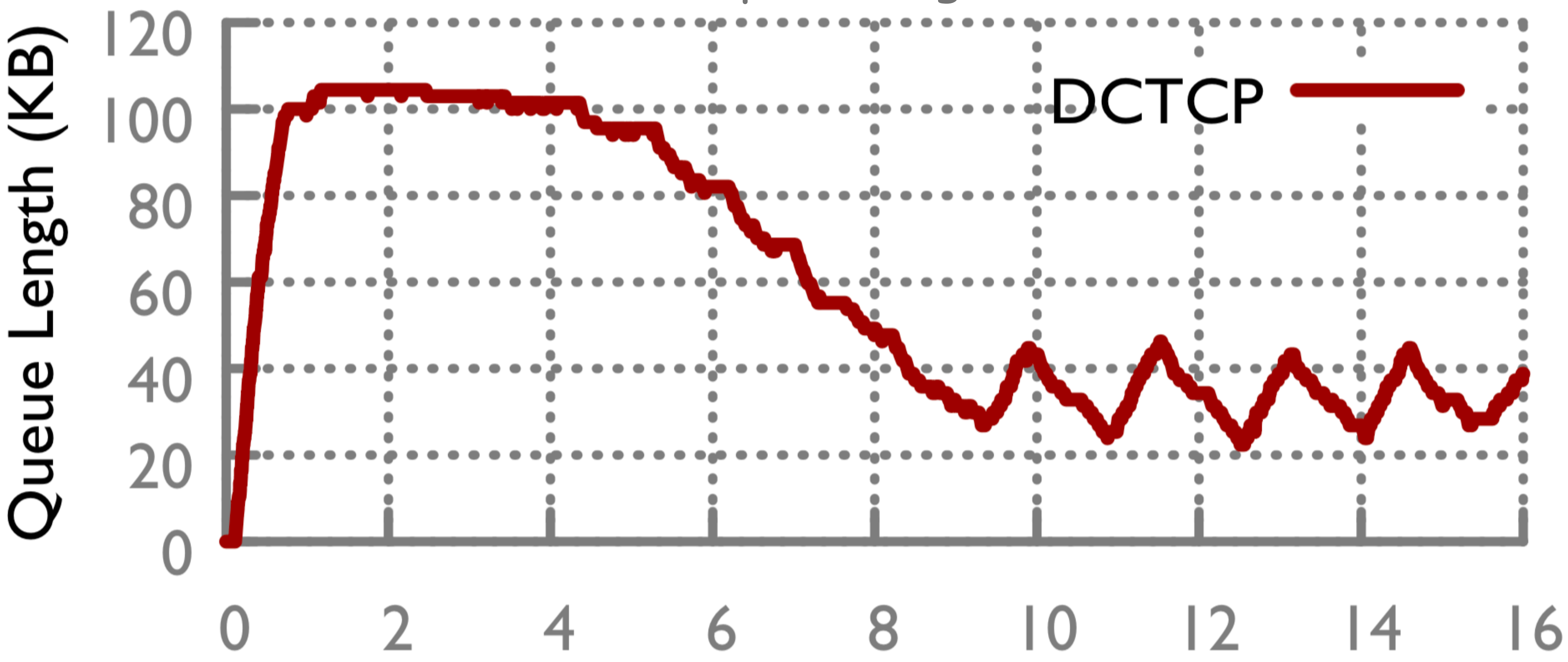
Switch:

Notify senders as soon as possible

How? ECN marking

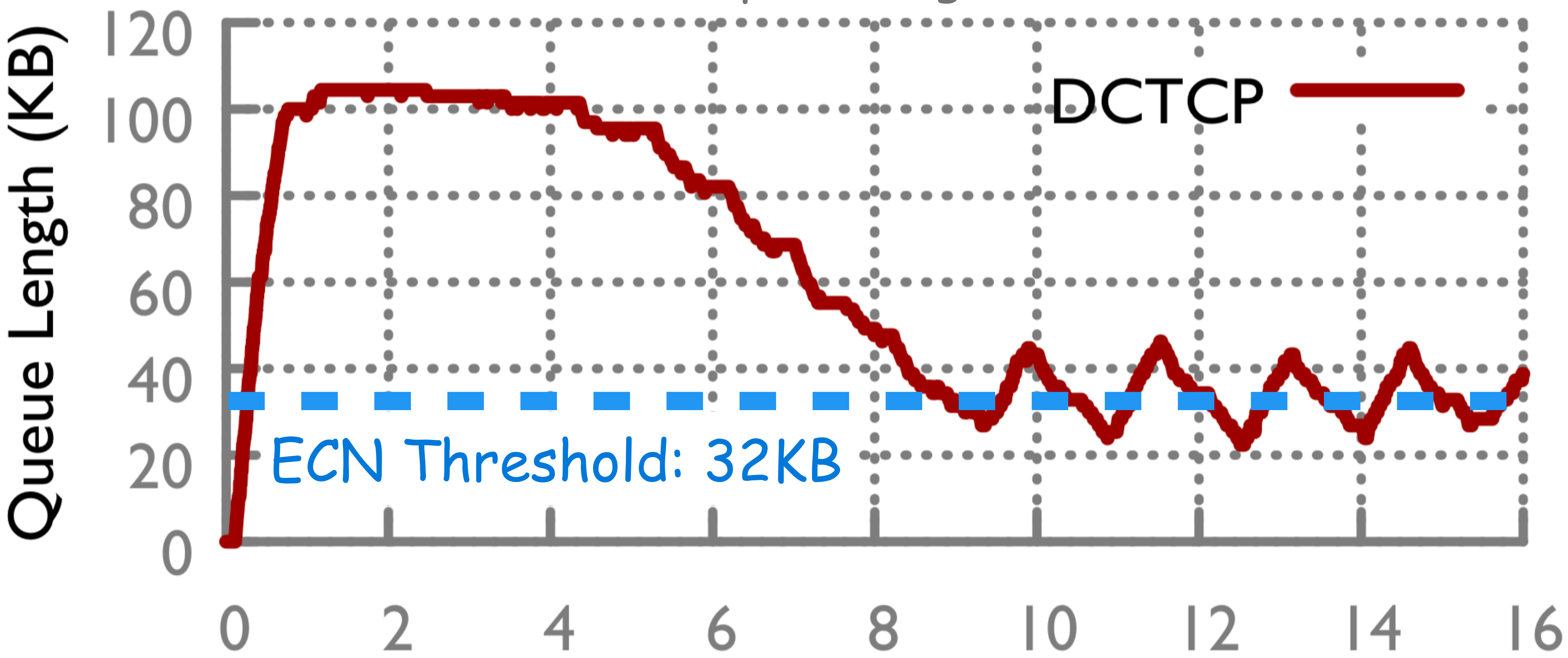
Mitigating Micro-bursts

Exp. settings: 9 senders —> 1 receiver



Mitigating Micro-bursts

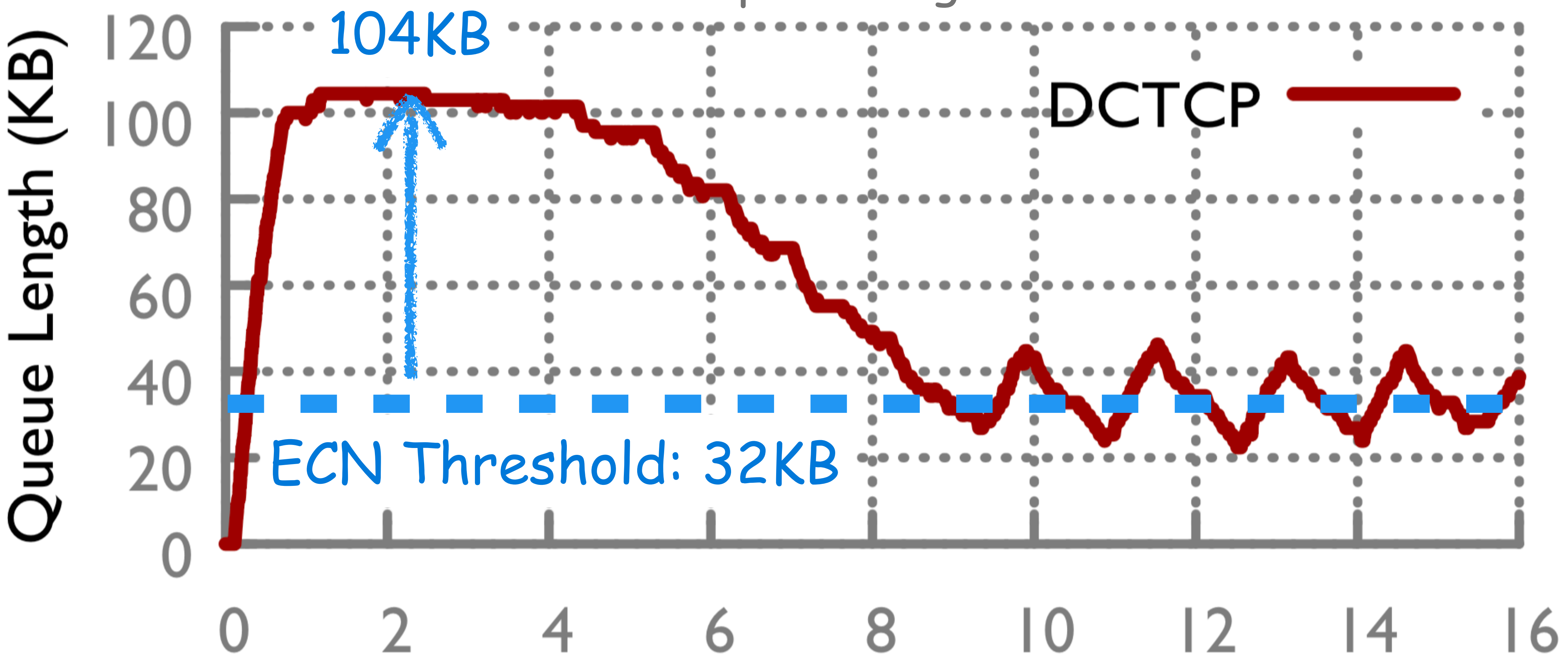
Exp. settings: 9 senders —> 1 receiver



ECN Threshold: 32KB

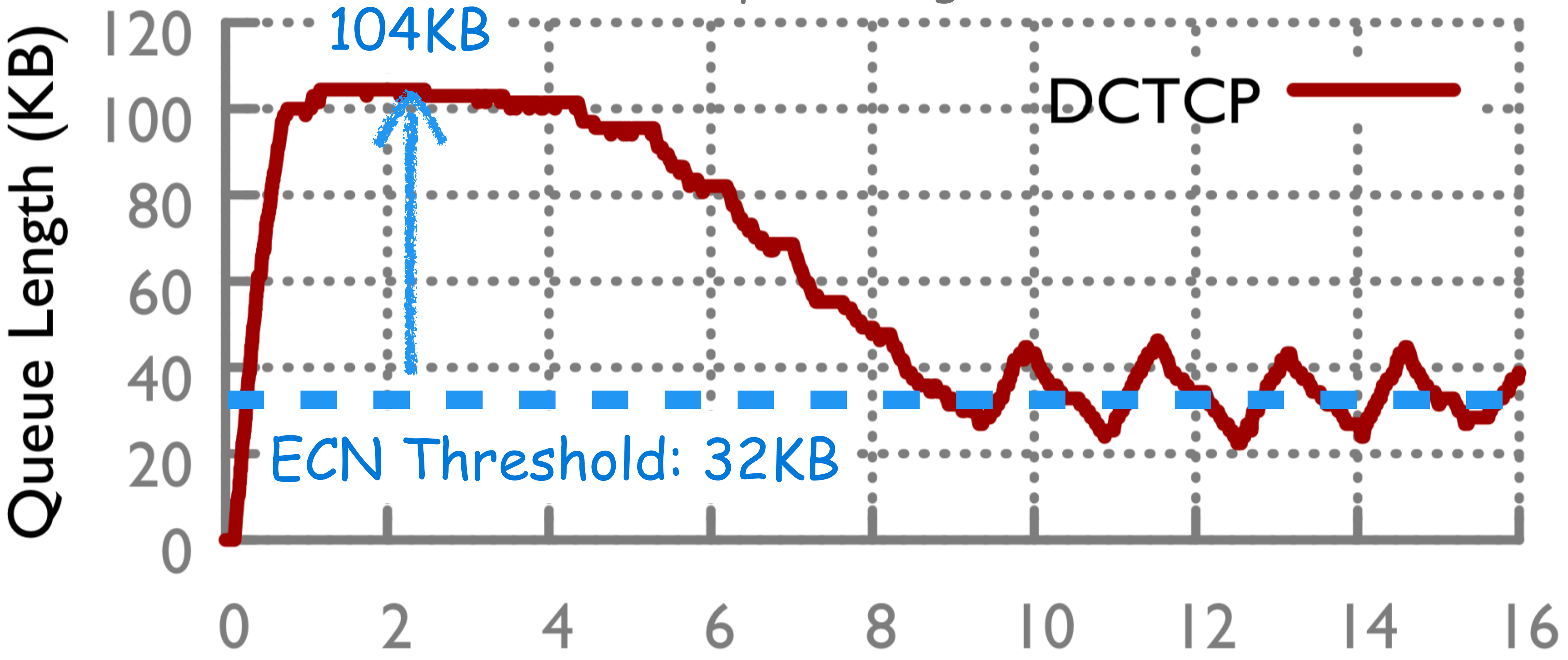
Mitigating Micro-bursts

Exp. settings: 9 senders —> 1 receiver



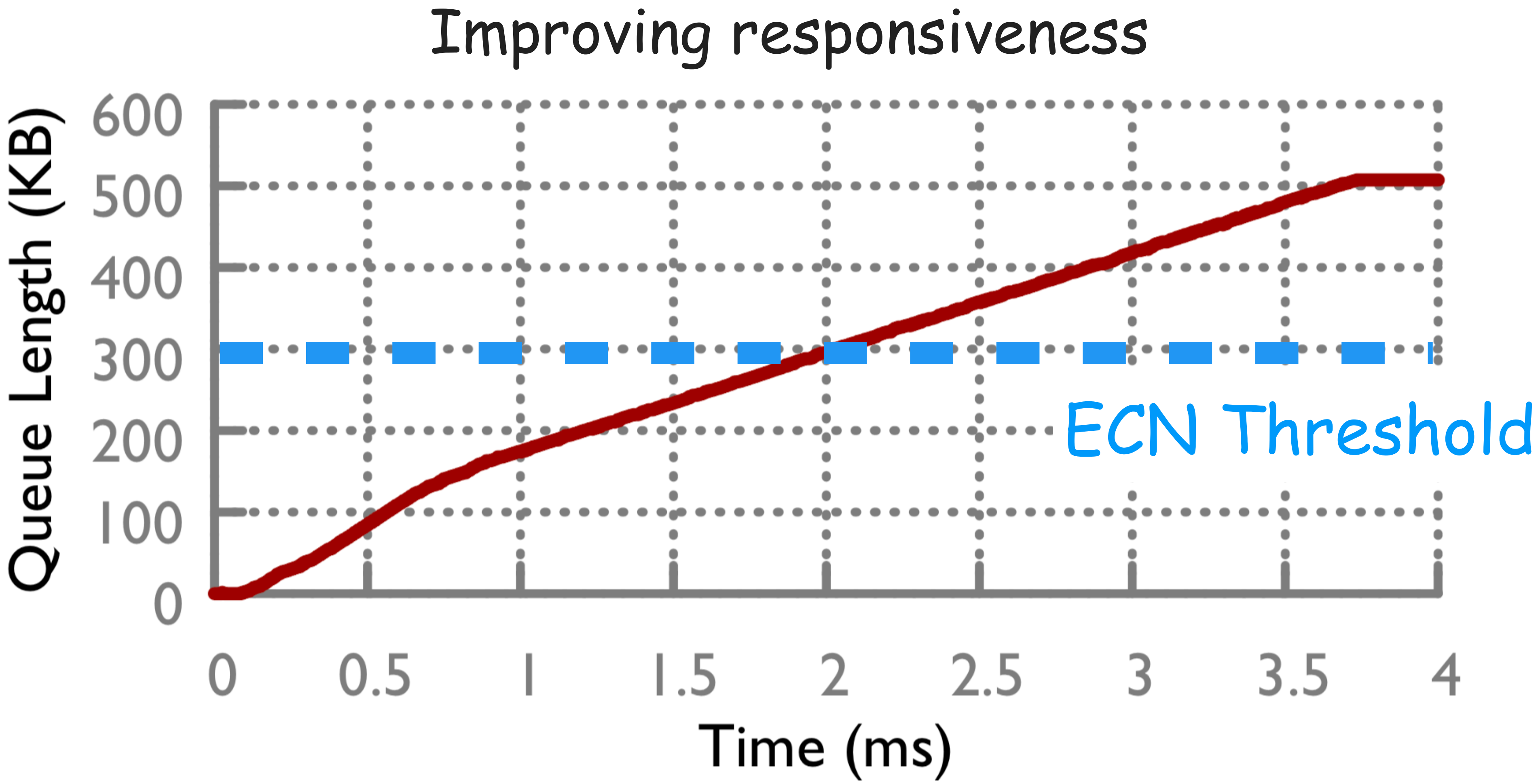
Mitigating Micro-bursts

Exp. settings: 9 senders —> 1 receiver

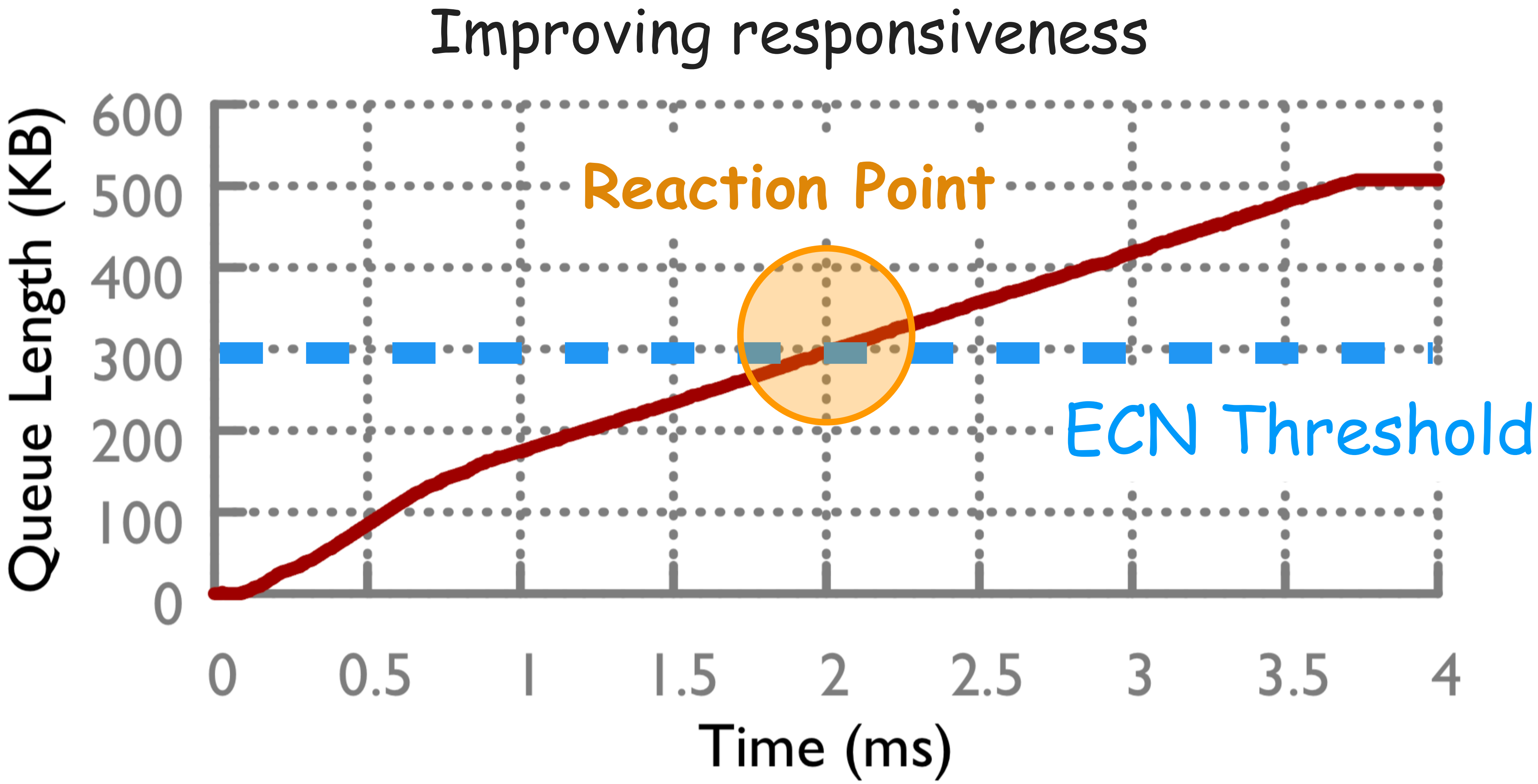


Not responsive enough ☹

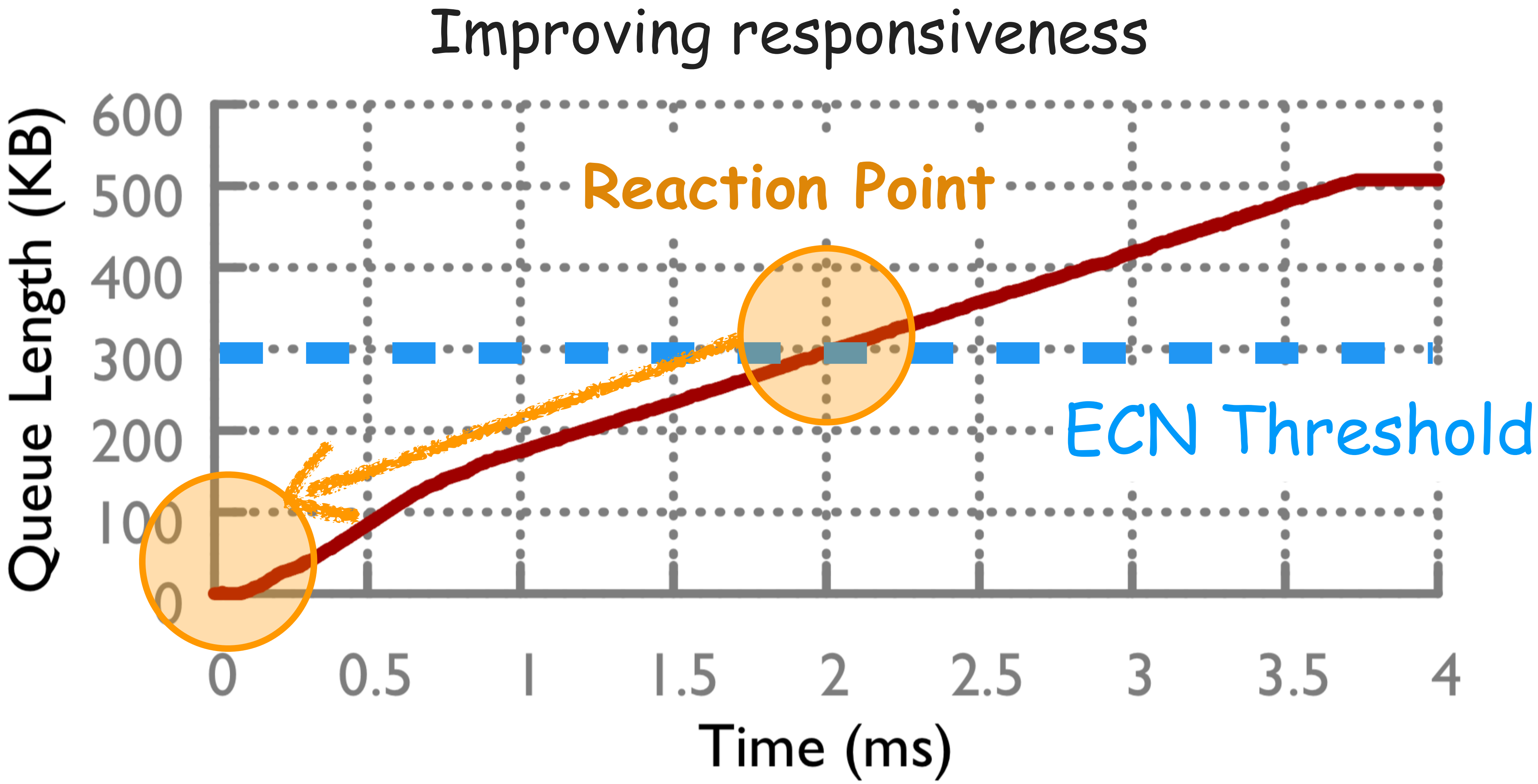
Mitigating Micro-bursts



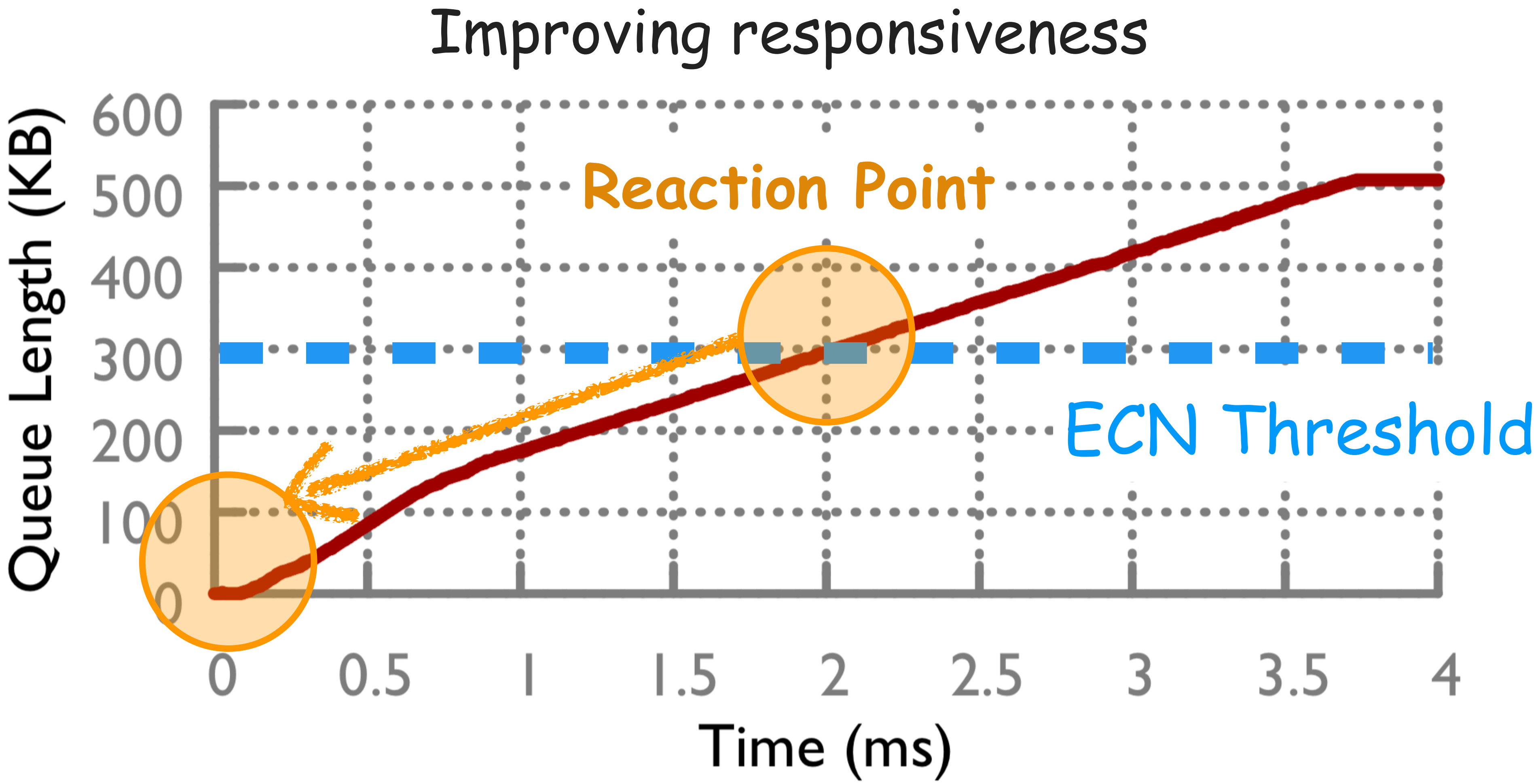
Mitigating Micro-bursts



Mitigating Micro-bursts



Mitigating Micro-bursts



Reduce ECN threshold —> Throughput Loss ☹

Mitigating Micro-bursts

S-ECN: slope-based ECN marking scheme

- Stochastically mark packets
- The bigger the slope, the larger the marking probability

Mitigating Micro-bursts

S-ECN: slope-based ECN marking scheme

- Stochastically mark packets
- The bigger the slope, the larger the marking probability

↓

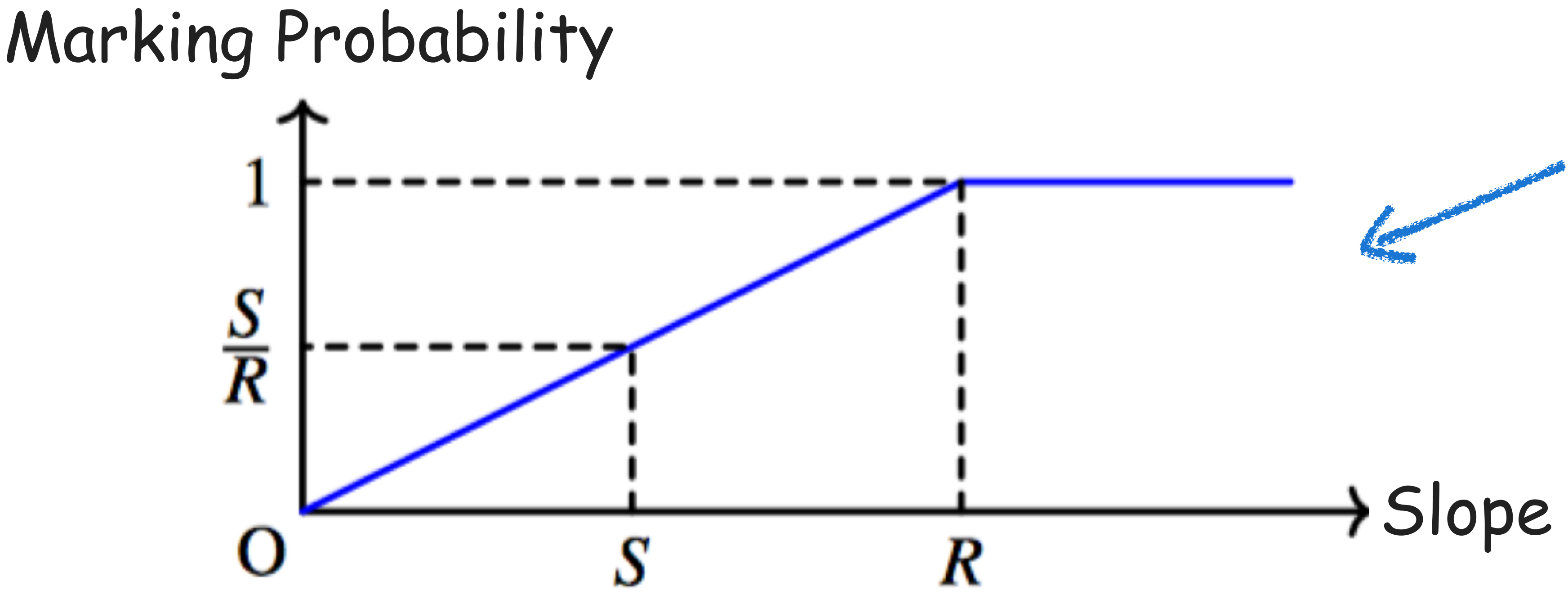
Send slope to senders

Mitigating Micro-bursts

S-ECN: slope-based ECN marking scheme

- Stochastically mark packets
- The bigger the slope, the larger the marking probability

Send slope to senders



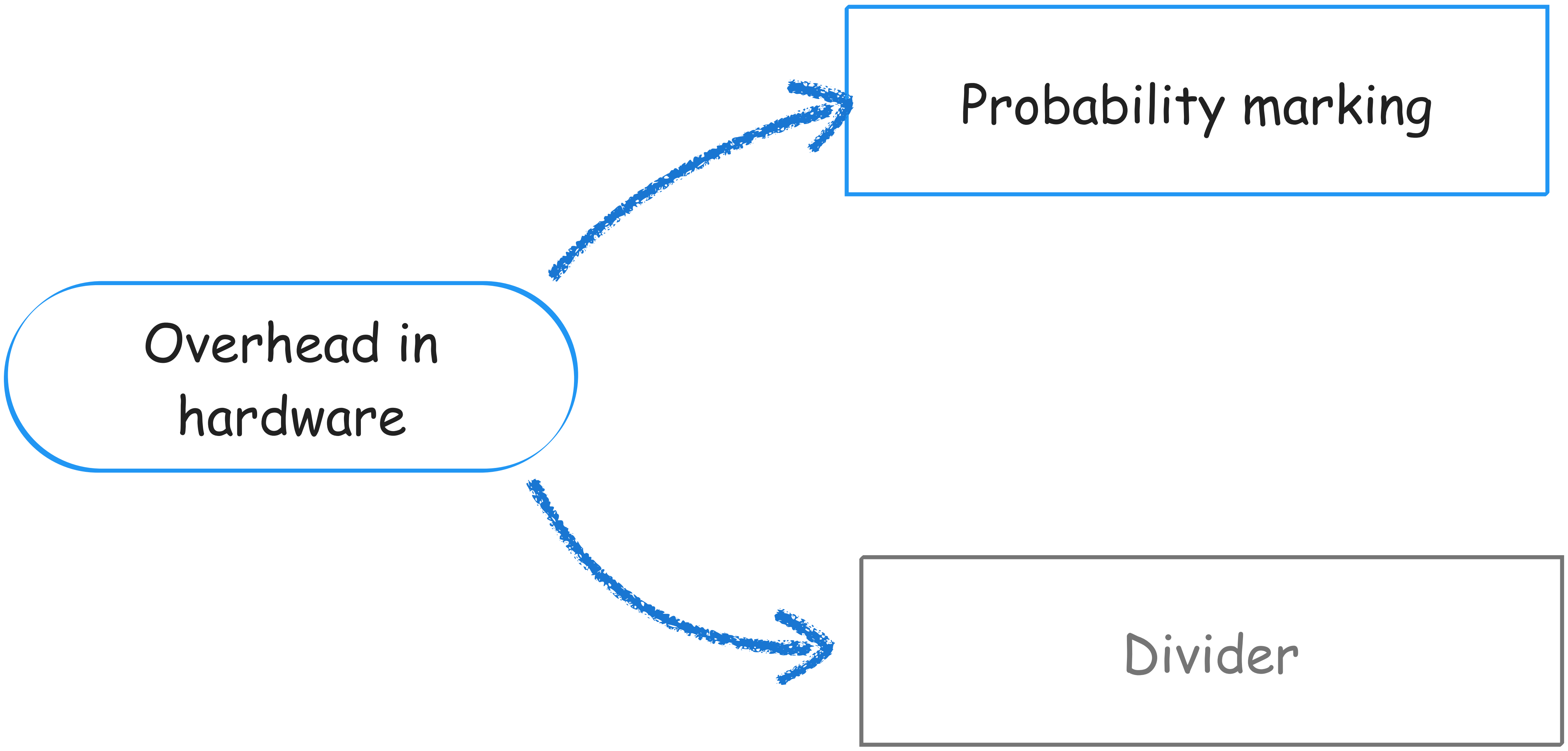
$$Prob = \begin{cases} 0, & s \leqslant 0, \\ \frac{s}{R}, & 0 < s < R, \\ 1, & s \geqslant R \end{cases}$$

Prob: Marking Probability

s: slope

R: port speed

Implementing S-ECN

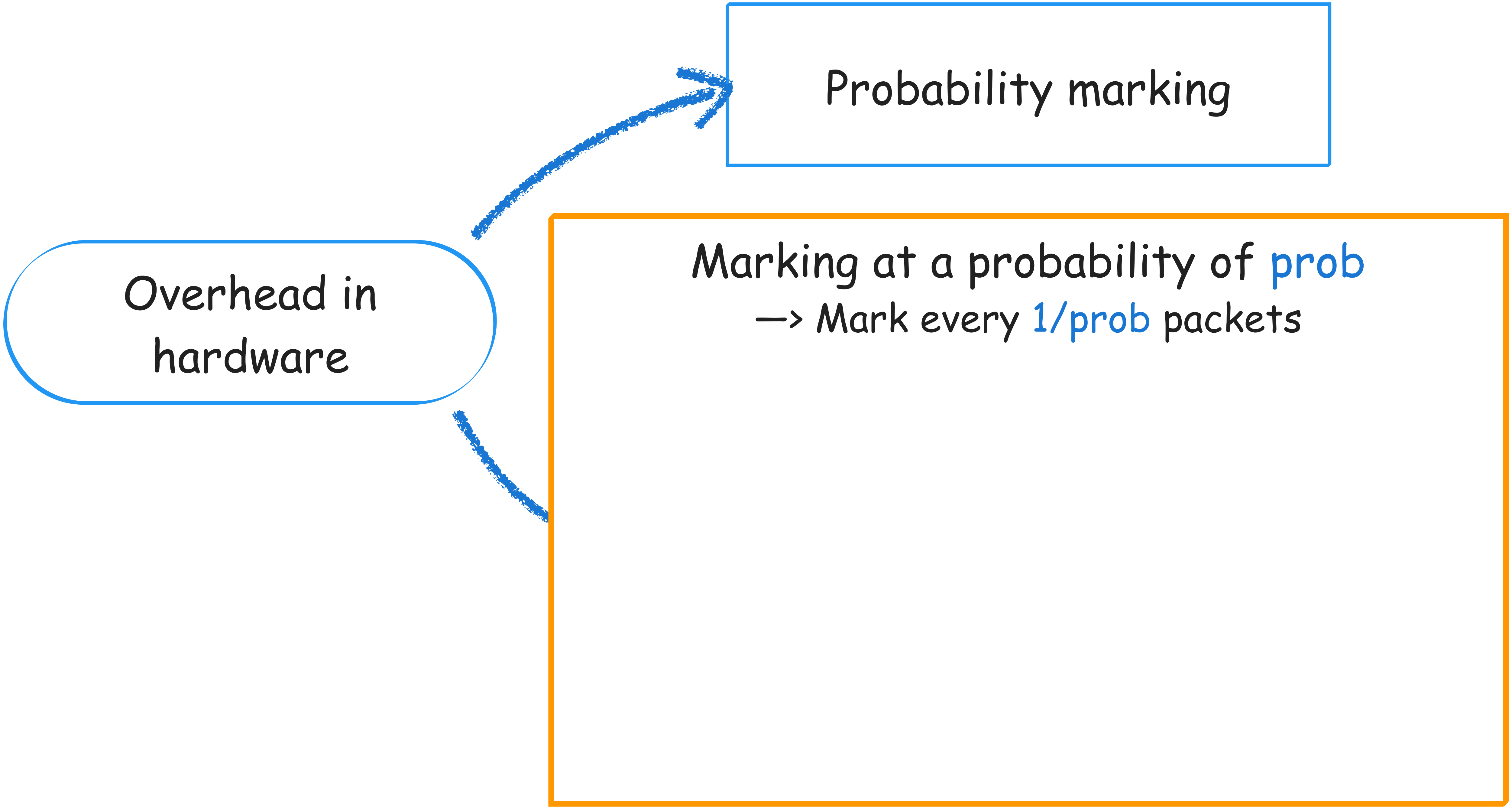


Overhead in hardware

Probability marking

Divider

Implementing S-ECN

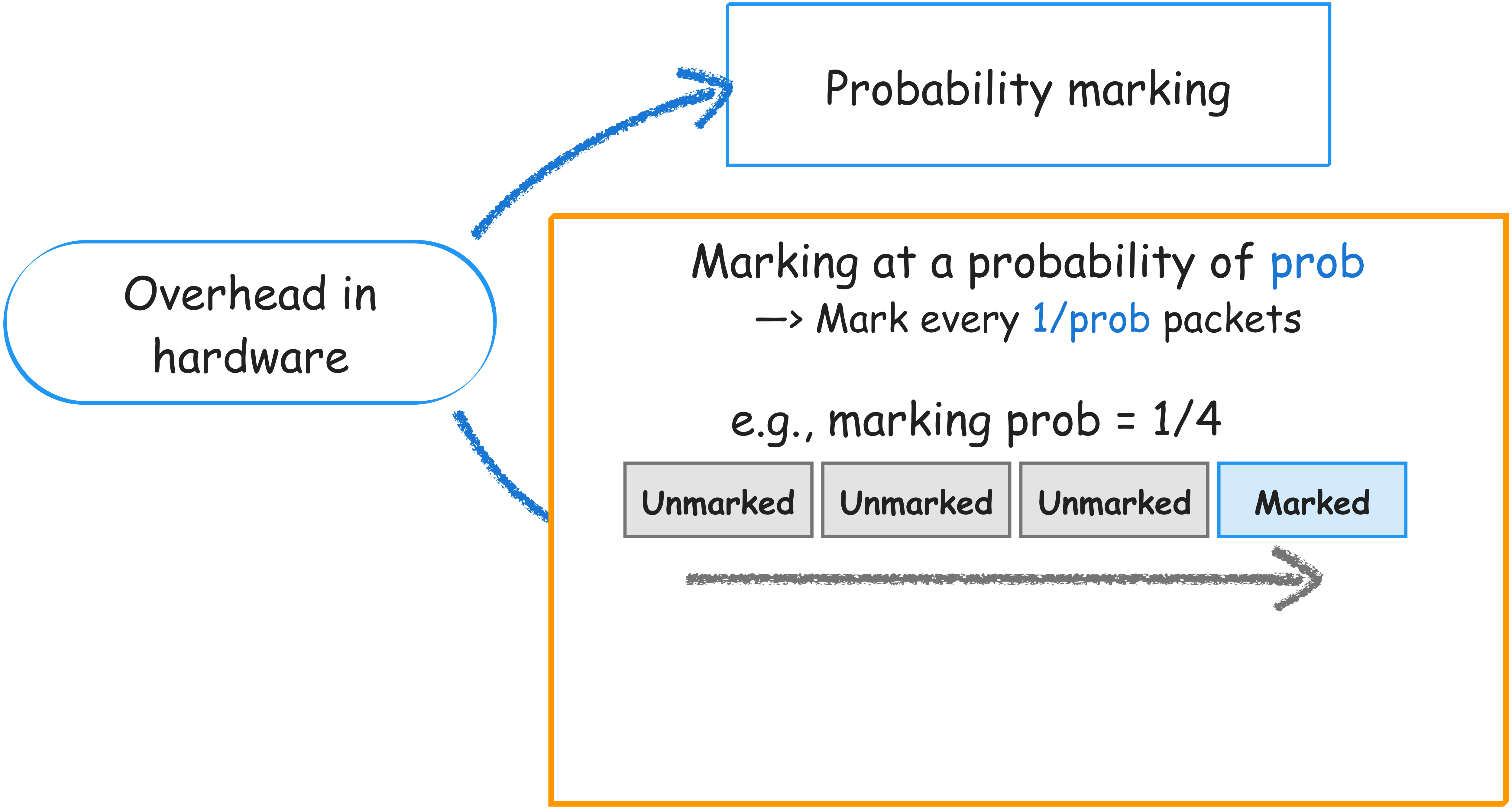


Overhead in hardware

Probability marking

Marking at a probability of prob
—> Mark every 1/prob packets

Implementing S-ECN



Overhead in hardware

Probability marking

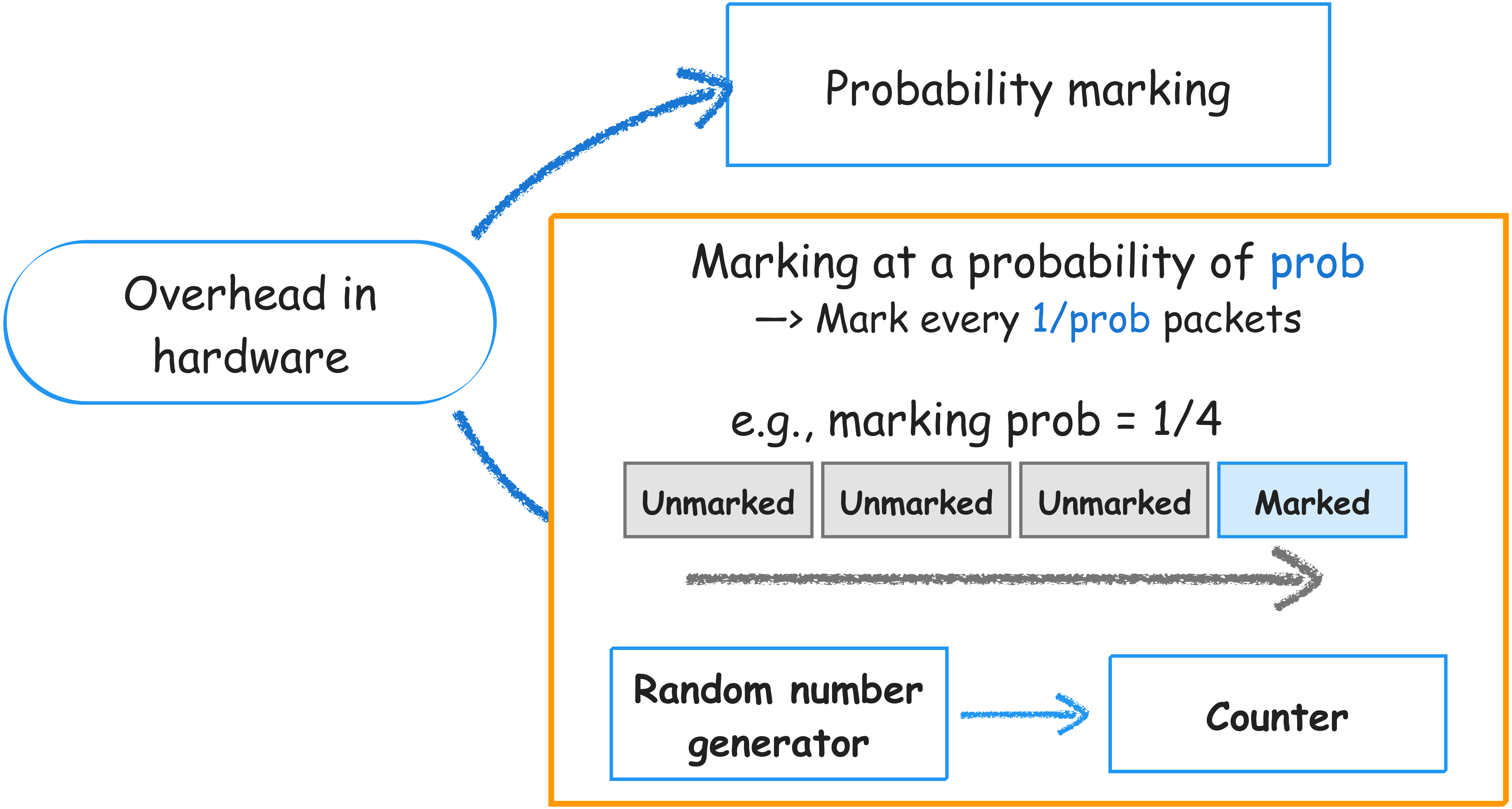
Marking at a probability of prob

—> Mark every 1/prob packets

e.g., marking prob = 1/4

Unmarked	Unmarked	Unmarked	Marked

Implementing S-ECN



Overhead in hardware

Probability marking

Marking at a probability of prob

—> Mark every 1/prob packets

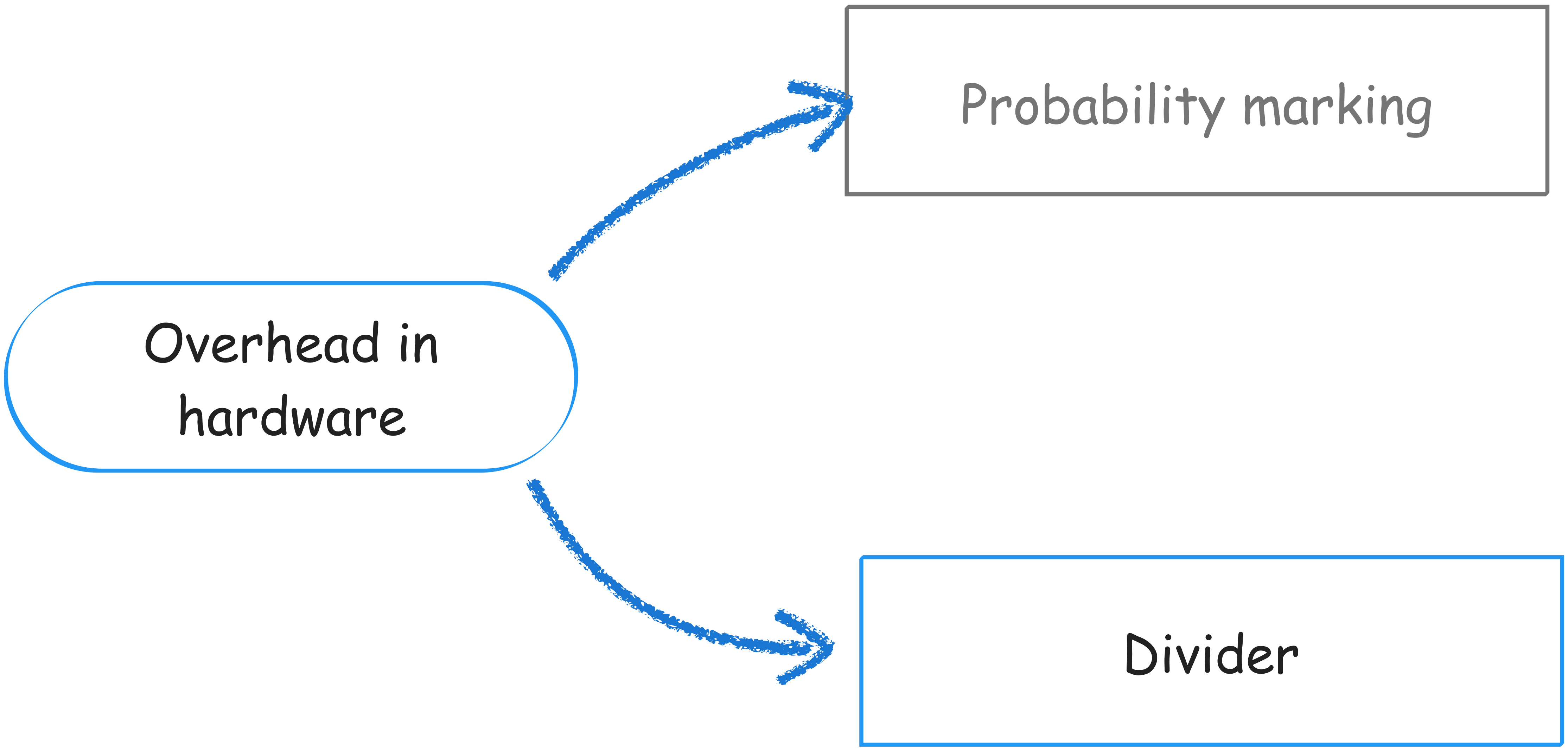
e.g., marking prob = 1/4

Unmarked	Unmarked	Unmarked	Marked

Random number generator

Counter

Implementing S-ECN

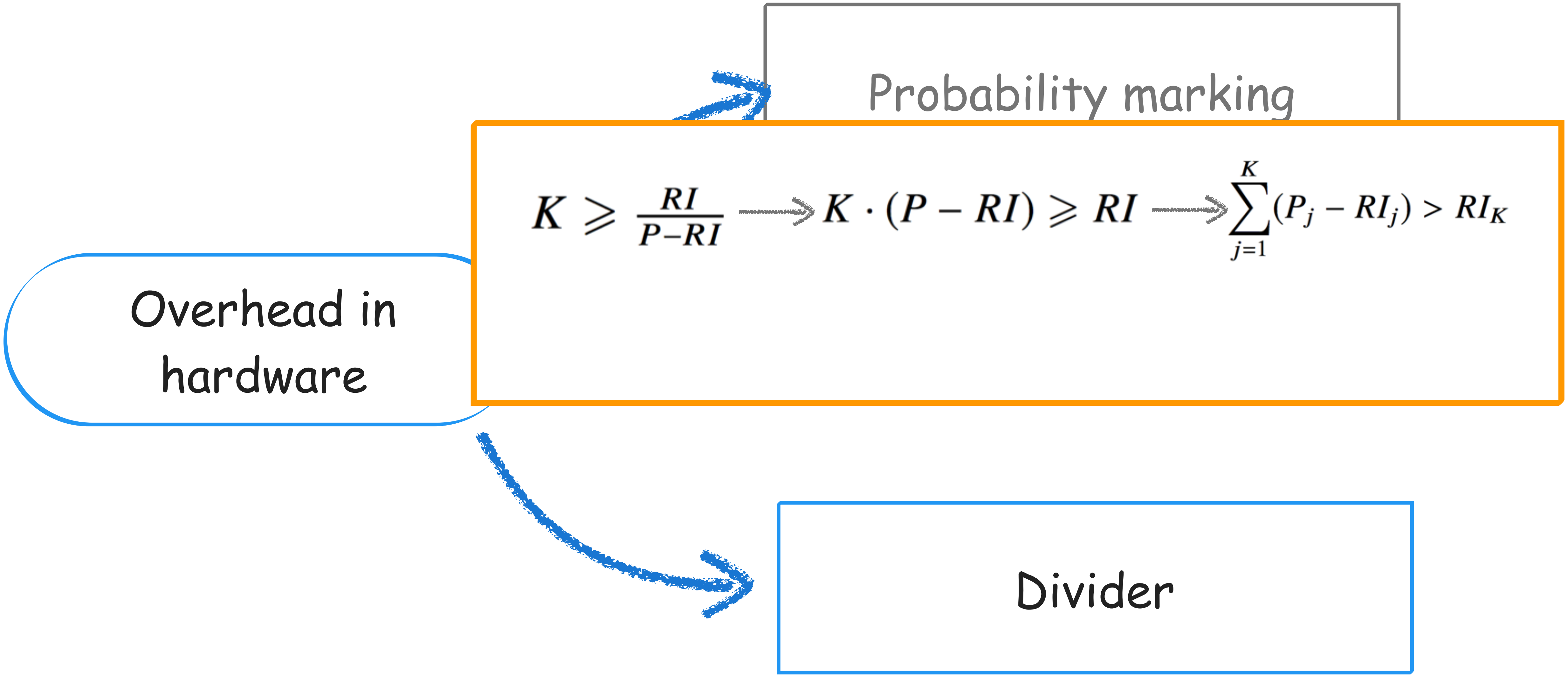


Overhead in hardware

Probability marking

Divider

Implementing S-ECN



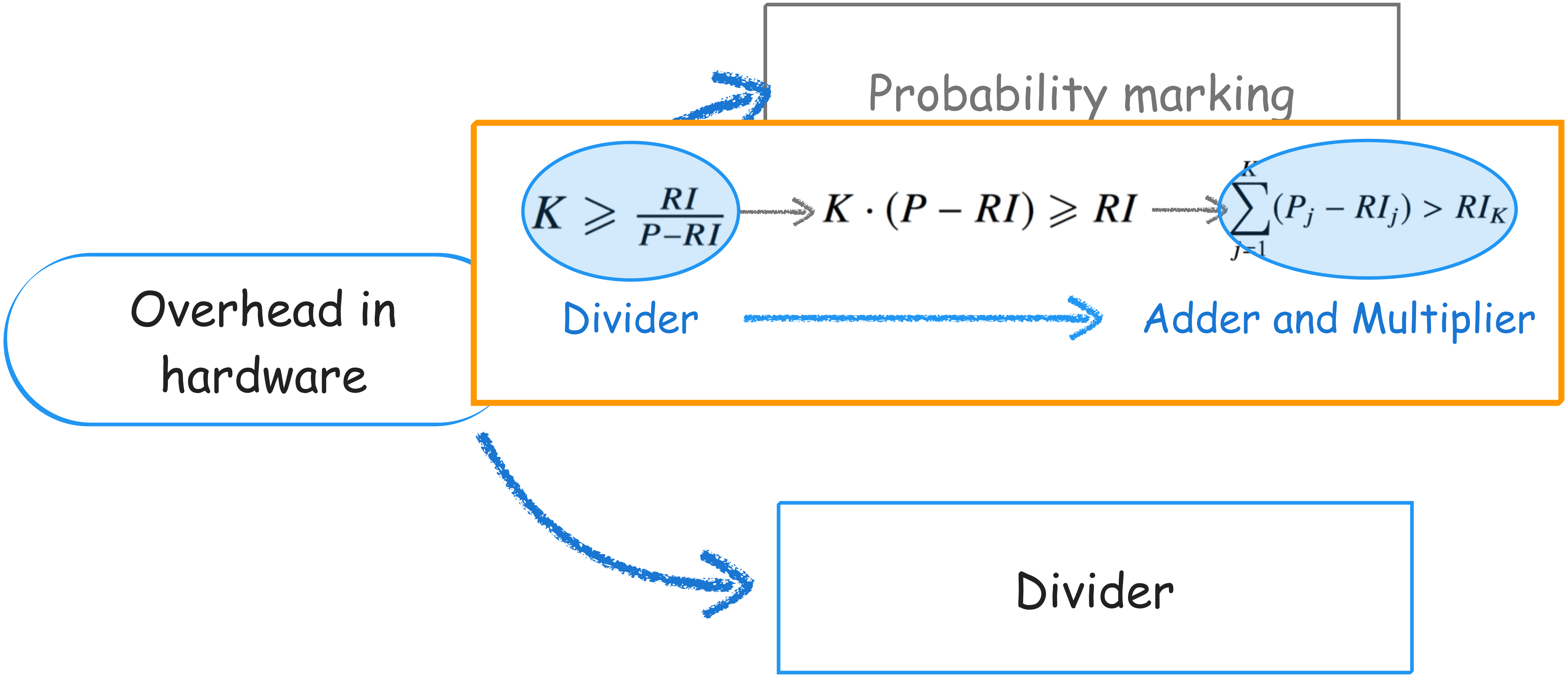
Overhead in hardware

Probability marking

$$K \geqslant \frac{RI}{P-RI} \dashrightarrow K \cdot (P - RI) \geqslant RI \dashrightarrow \sum_{j=1}^{K}(P_j - RI_j) > RI_K$$

Divider

Implementing S-ECN



Overhead in hardware

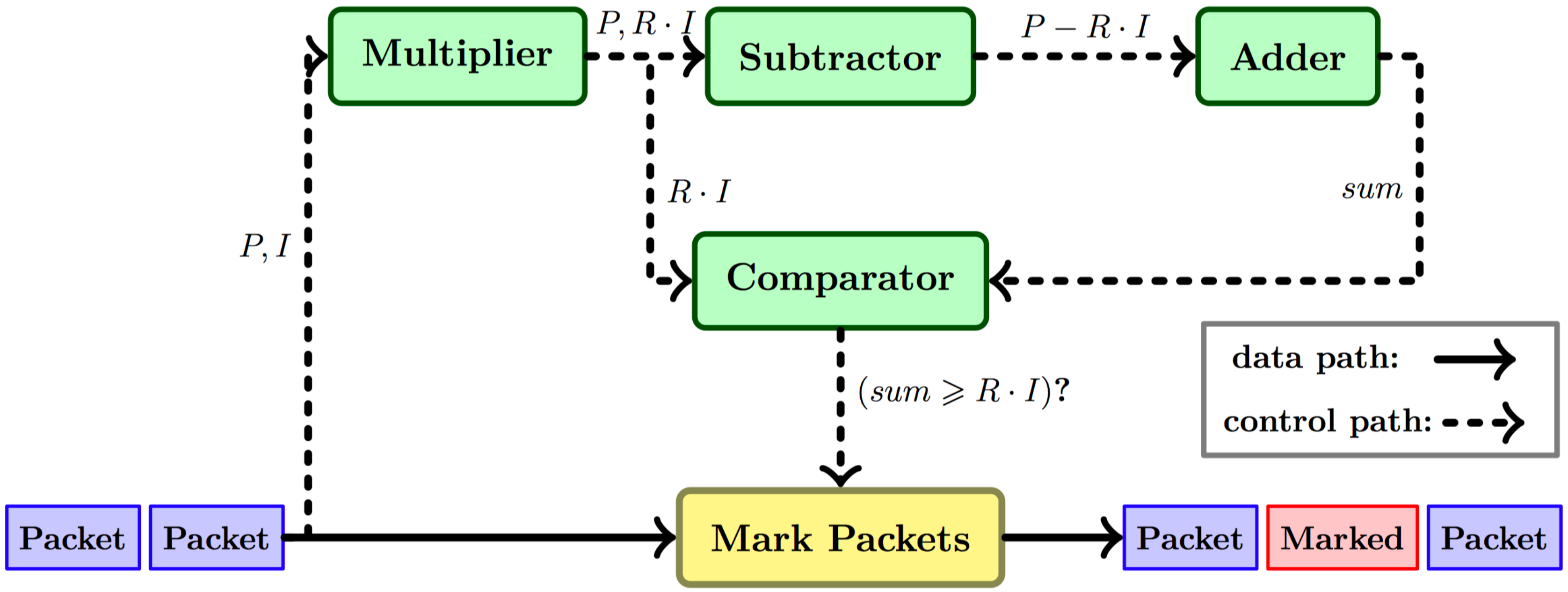
Probability marking

$$K \geqslant \frac{RI}{P-RI} \dashrightarrow K \cdot (P - RI) \geqslant RI \dashrightarrow \sum_{j=1}^{K}(P_j - RI_j) > RI_K$$

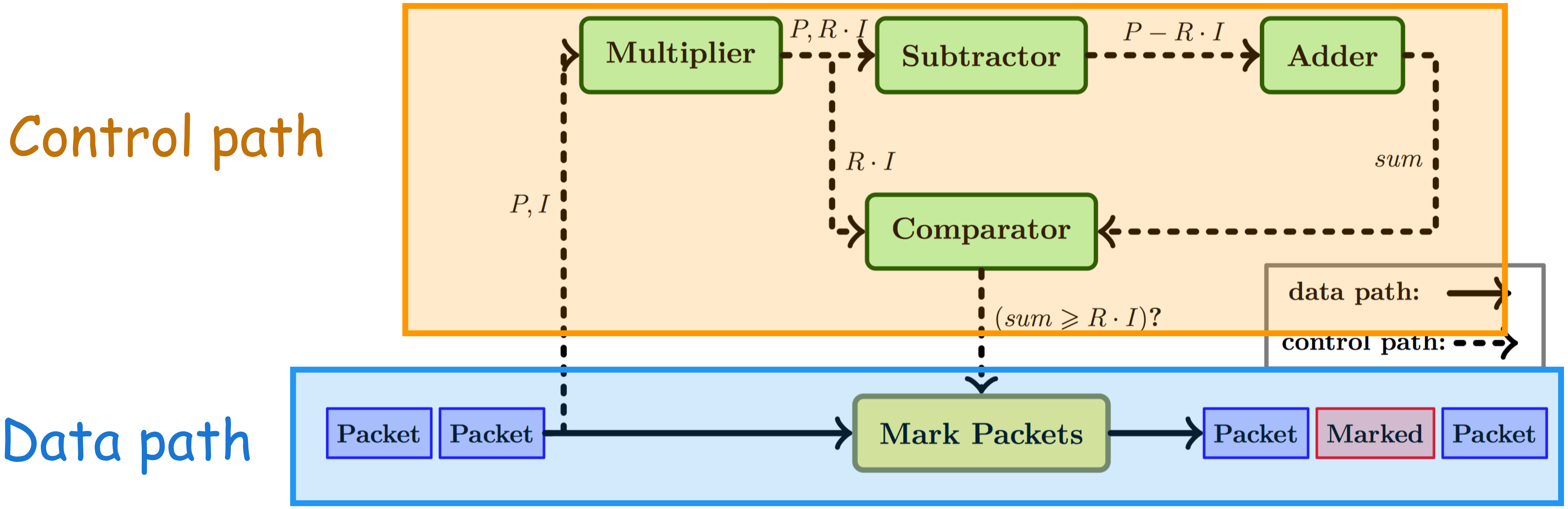
Divider → Adder and Multiplier

Divider

Implementing S-ECN



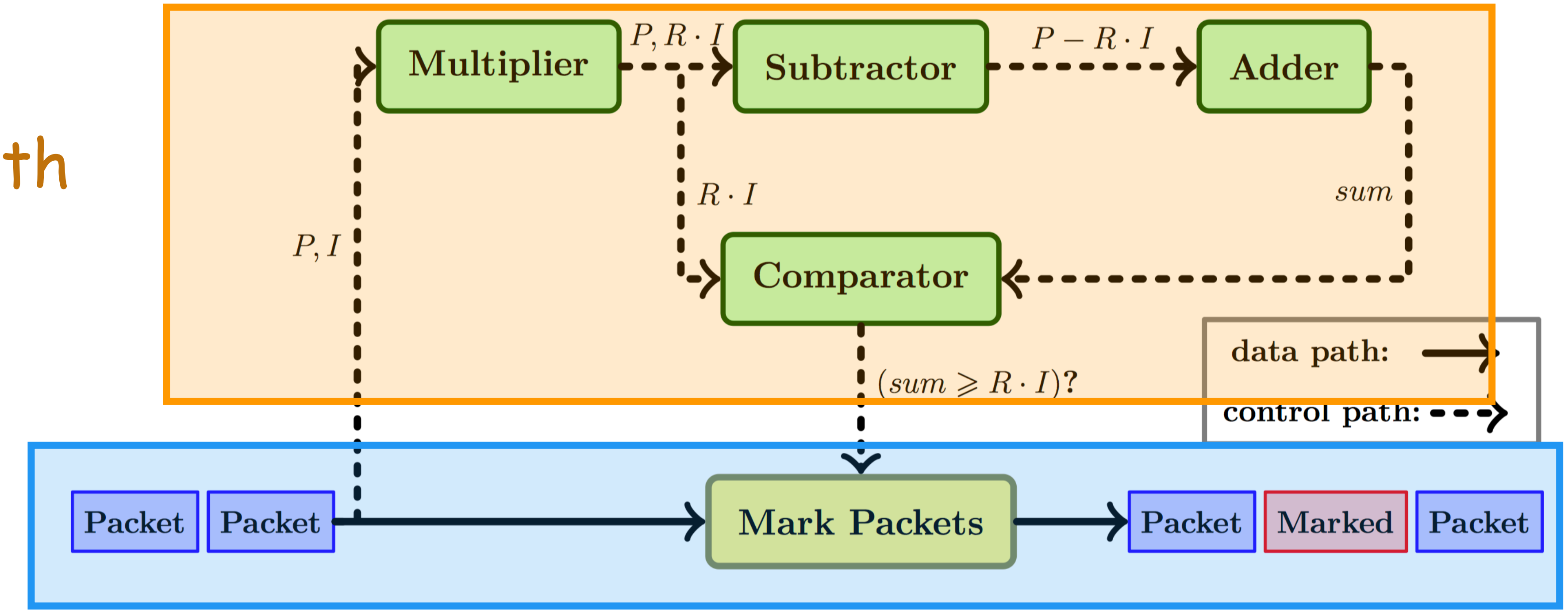
Implementing S-ECN



Implementing S-ECN

Control path

Data path

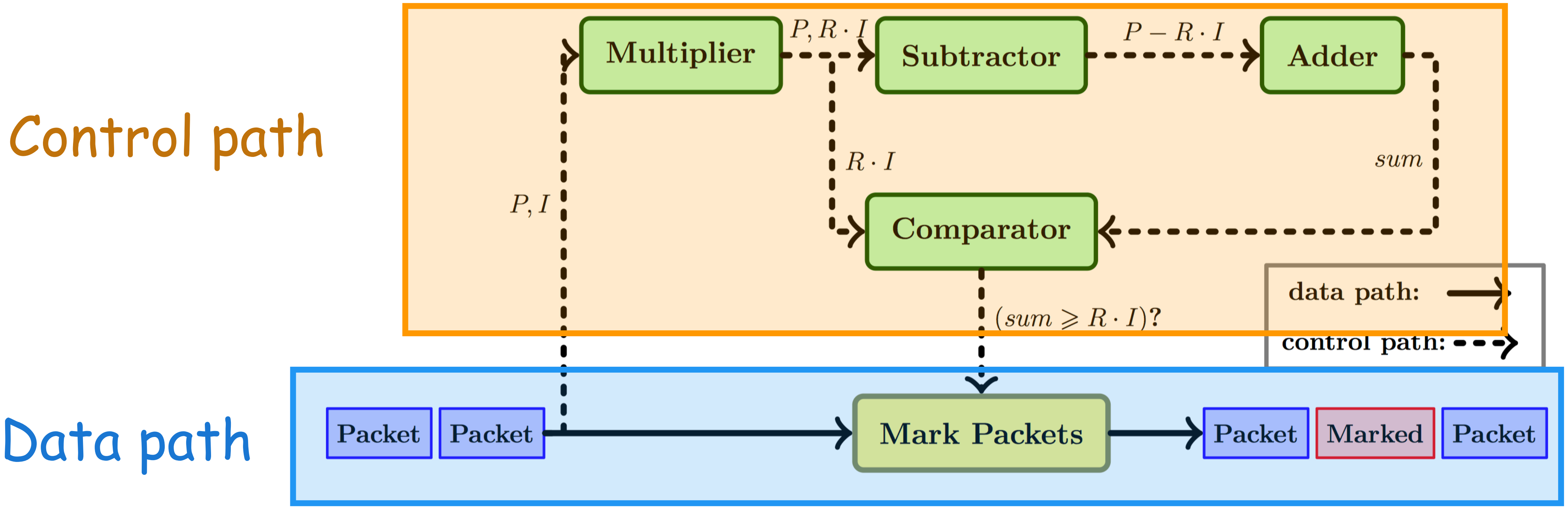


NetFPGA Implementation

Resource Usage

Resources	ECN Switch	+S-ECN
Slice Flip Flops	14738	14700
LUTs	18048	18544

Implementing S-ECN



NetFPGA Implementation

Resource Usage

Resources	ECN Switch	+S-ECN
Slice Flip Flops	14738	14700
LUTs	18048	18544

+6%

Evaluation

Protocols Compared

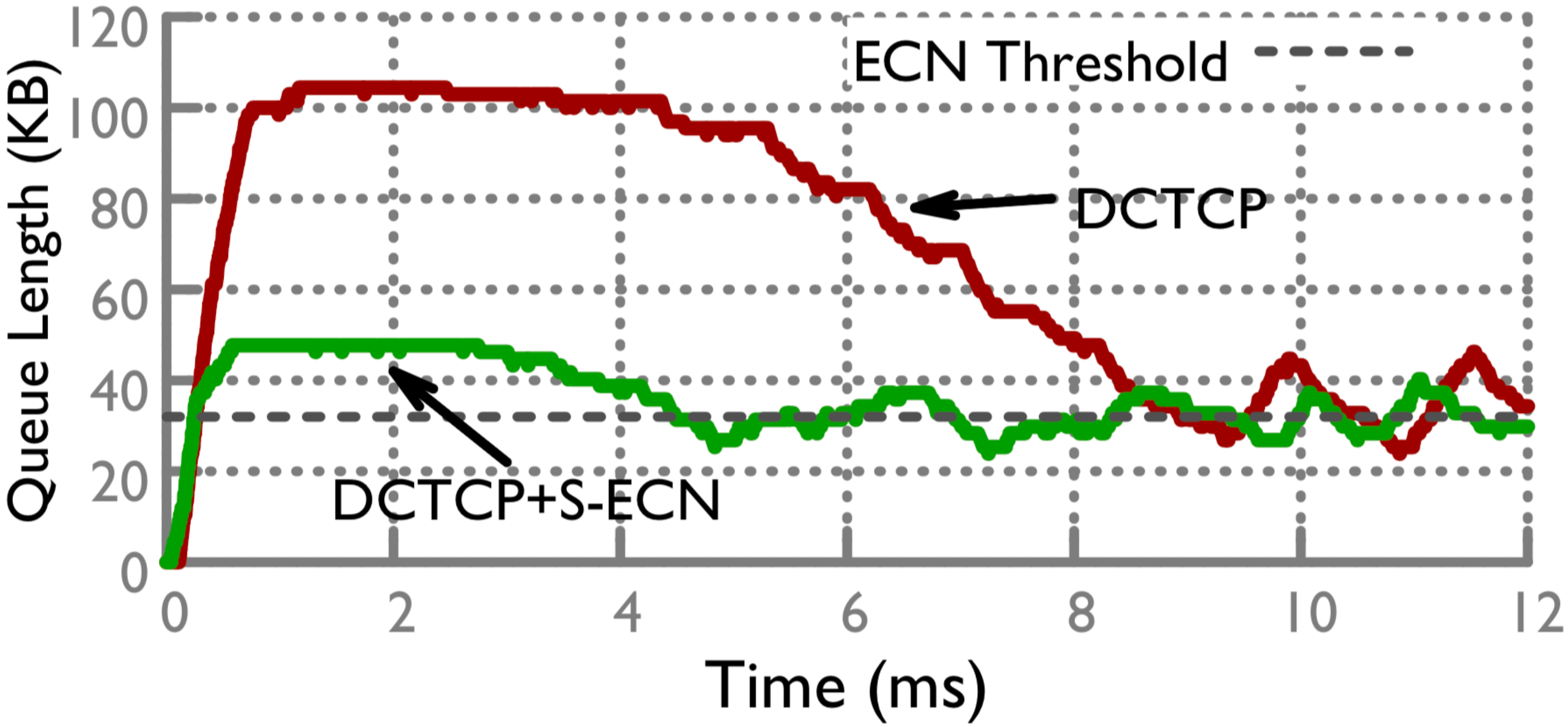
Protocols	End Host Algorithm	Switch Settings
DCTCP	DCTCP	Mark <—> Qlen >= K K = 32KB

DCTCP+S-ECN	DCTCP	if Qlen < K: S-ECN
		if Qlen >= K: Mark
		K= 32KB

Evaluation

— Suppression of sharp queue increasing

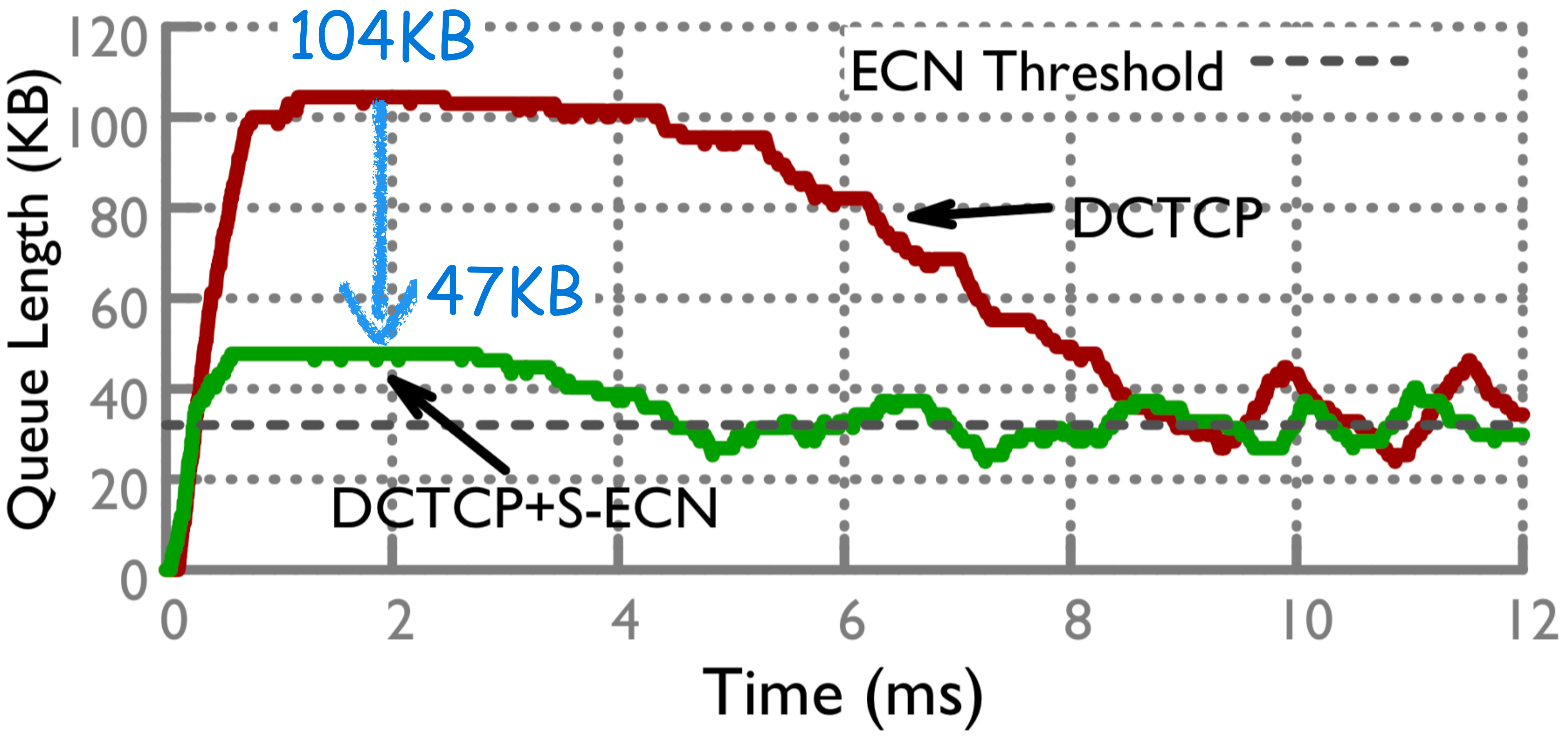
Settings: 9 senders —> 1 receiver



Evaluation

— Suppression of sharp queue increasing

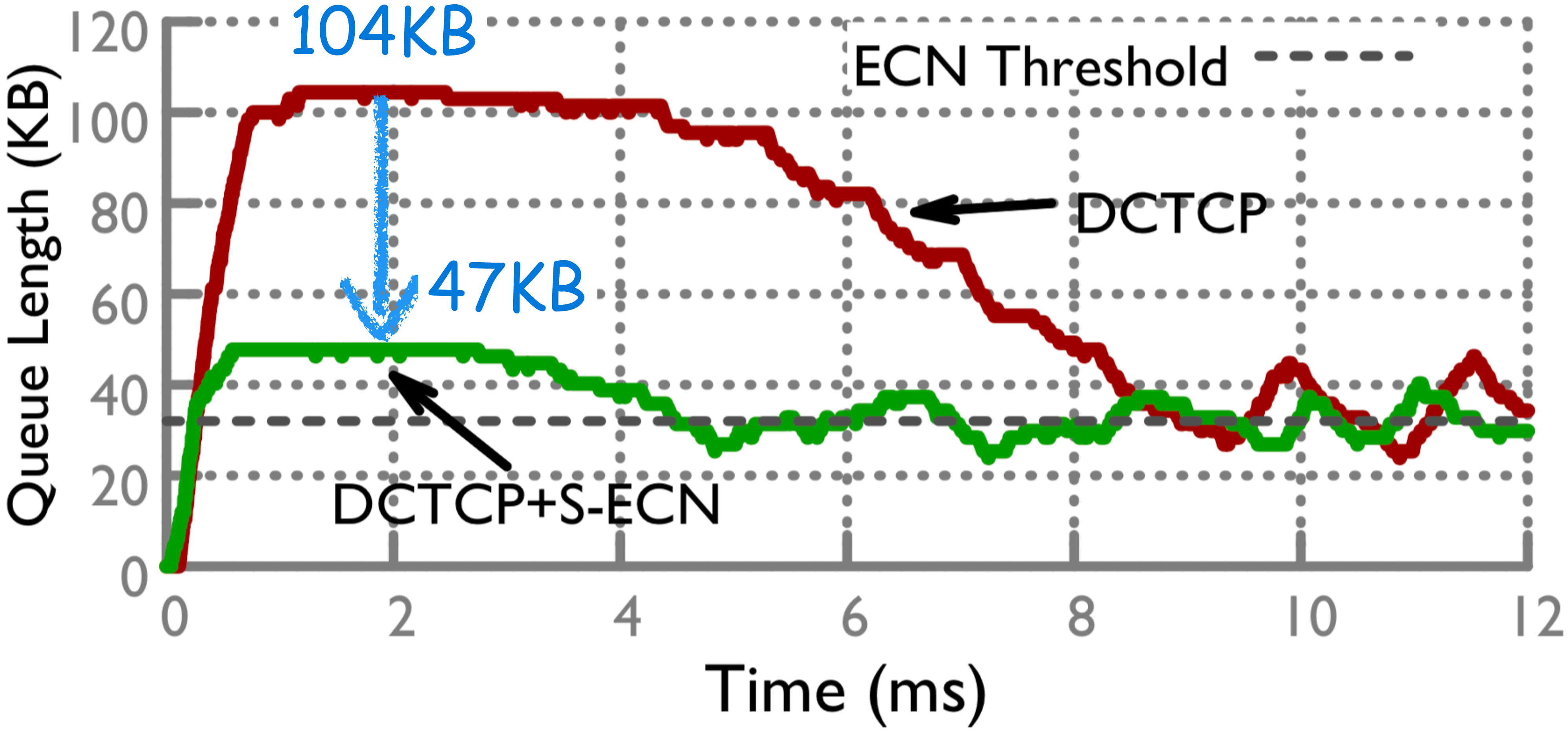
Settings: 9 senders —> 1 receiver



Evaluation

— Suppression of sharp queue increasing

Settings: 9 senders —> 1 receiver

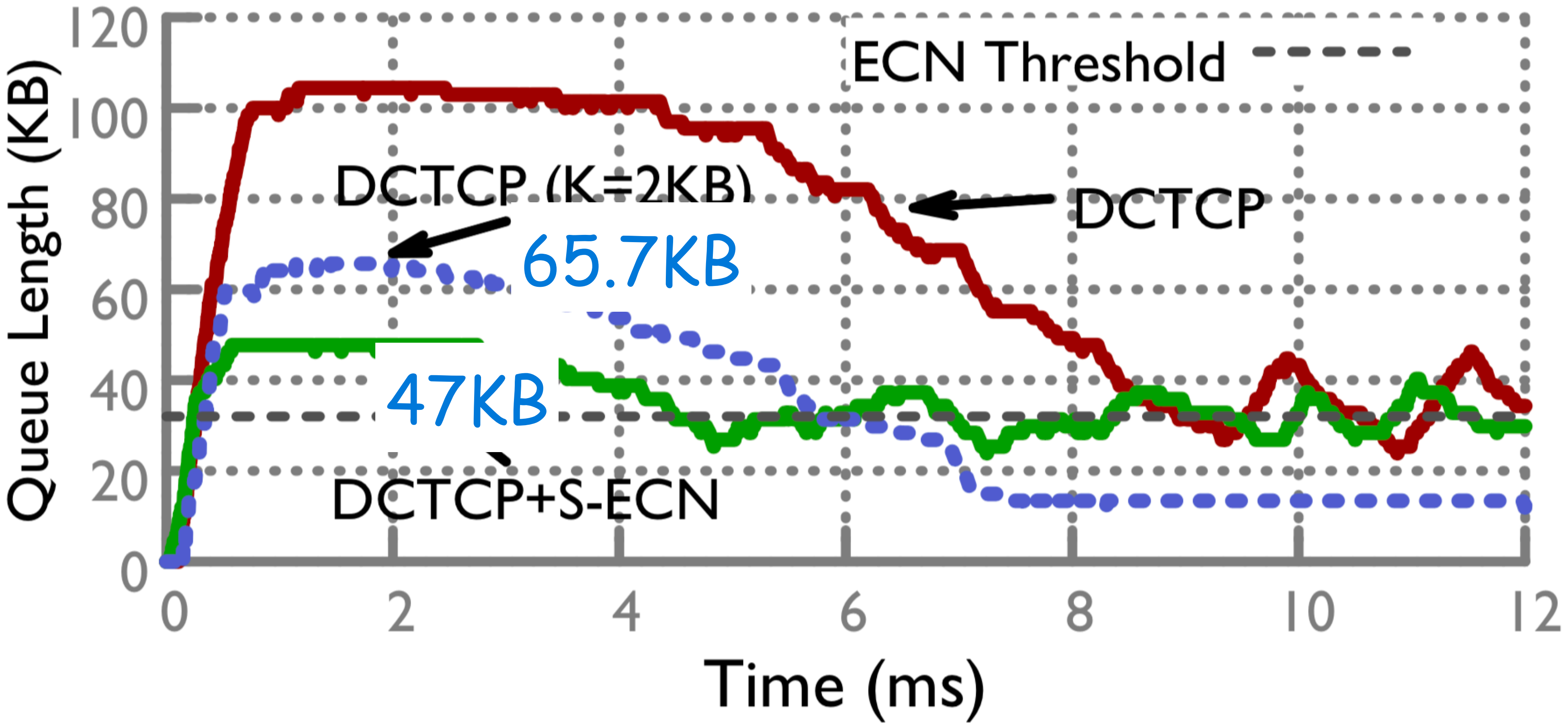


Queue length increment reduced by over 2x

Evaluation

— Suppression of sharp queue increasing

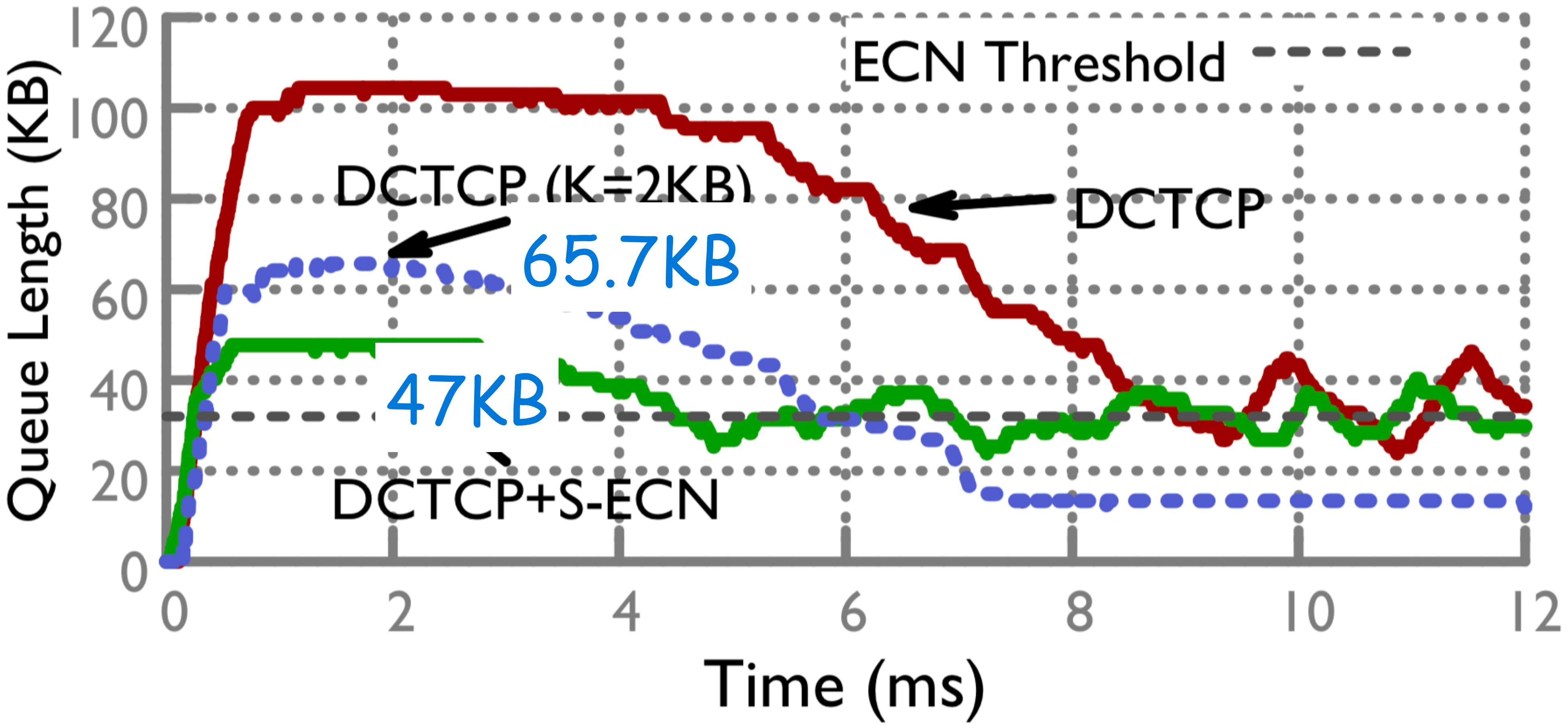
S-ECN vs. Extremely low ECN threshold



Evaluation

— Suppression of sharp queue increasing

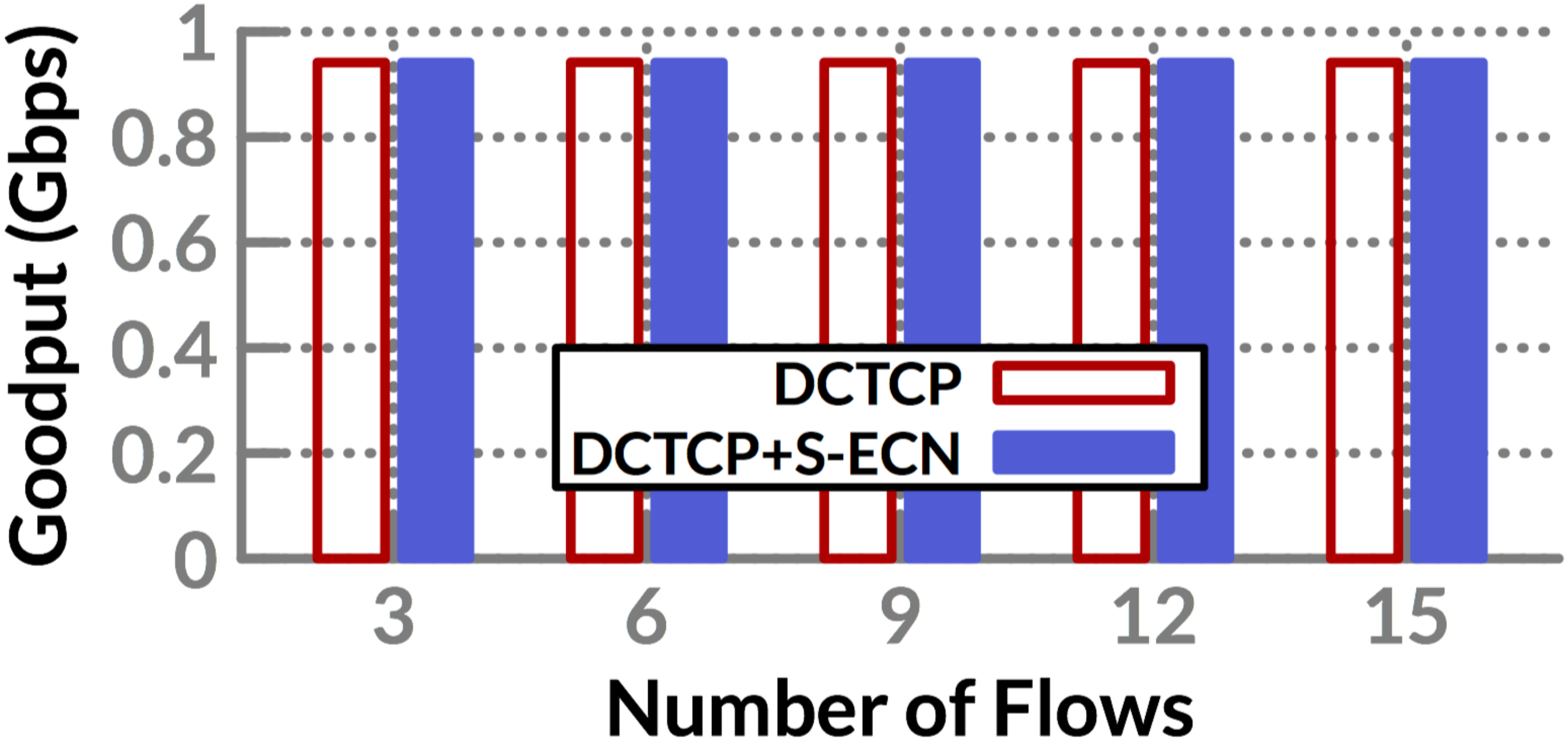
S-ECN vs. Extremely low ECN threshold



S-ECN is more responsive

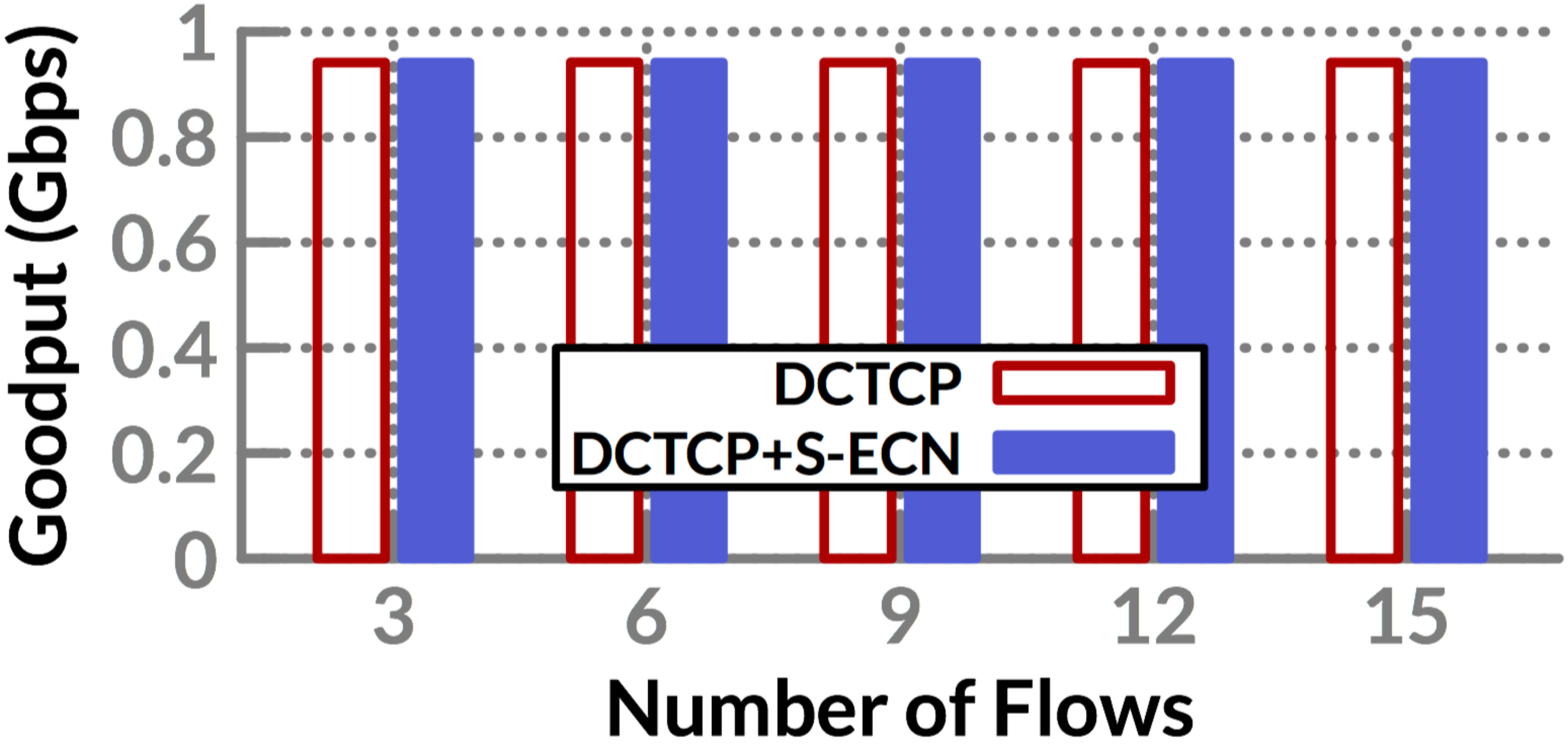
Evaluation

— Network Utilization



Evaluation

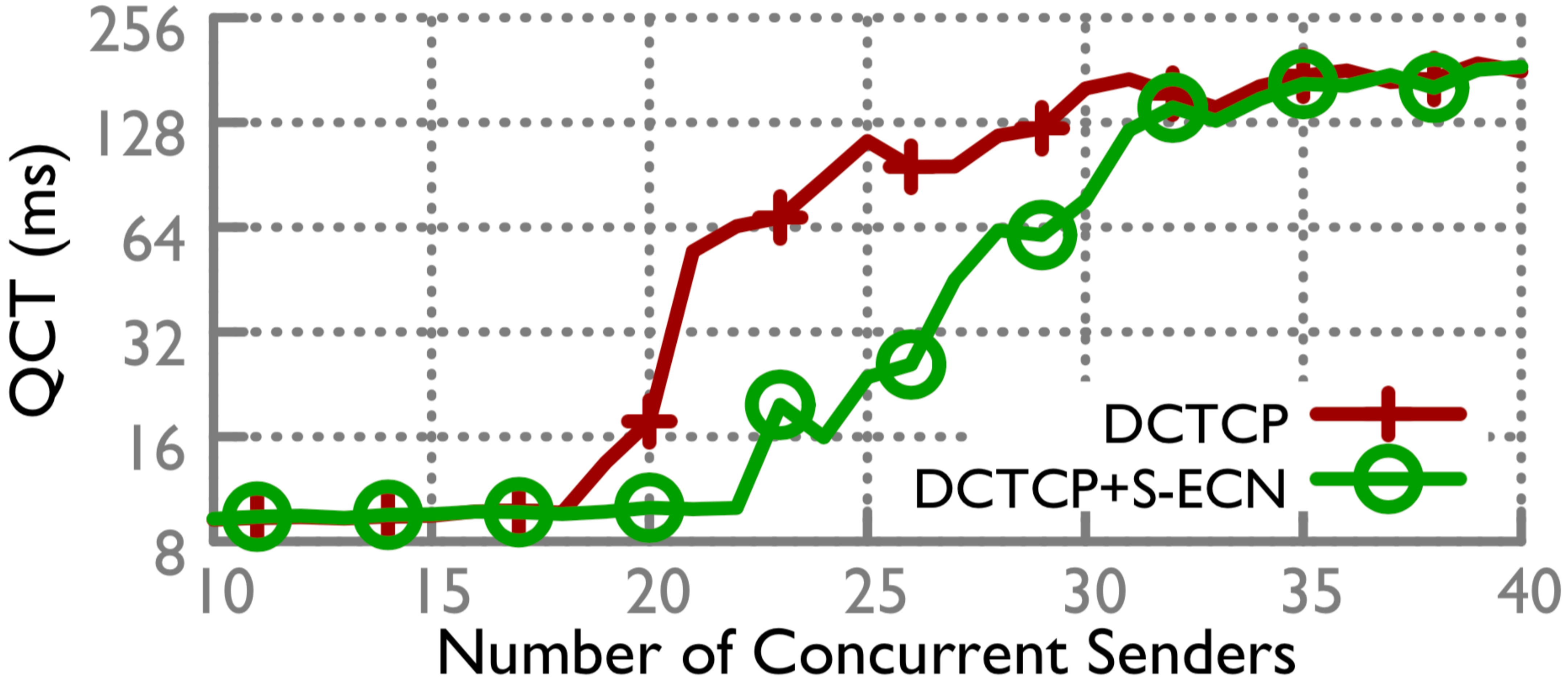
— Network Utilization



S-ECN can fully utilize network

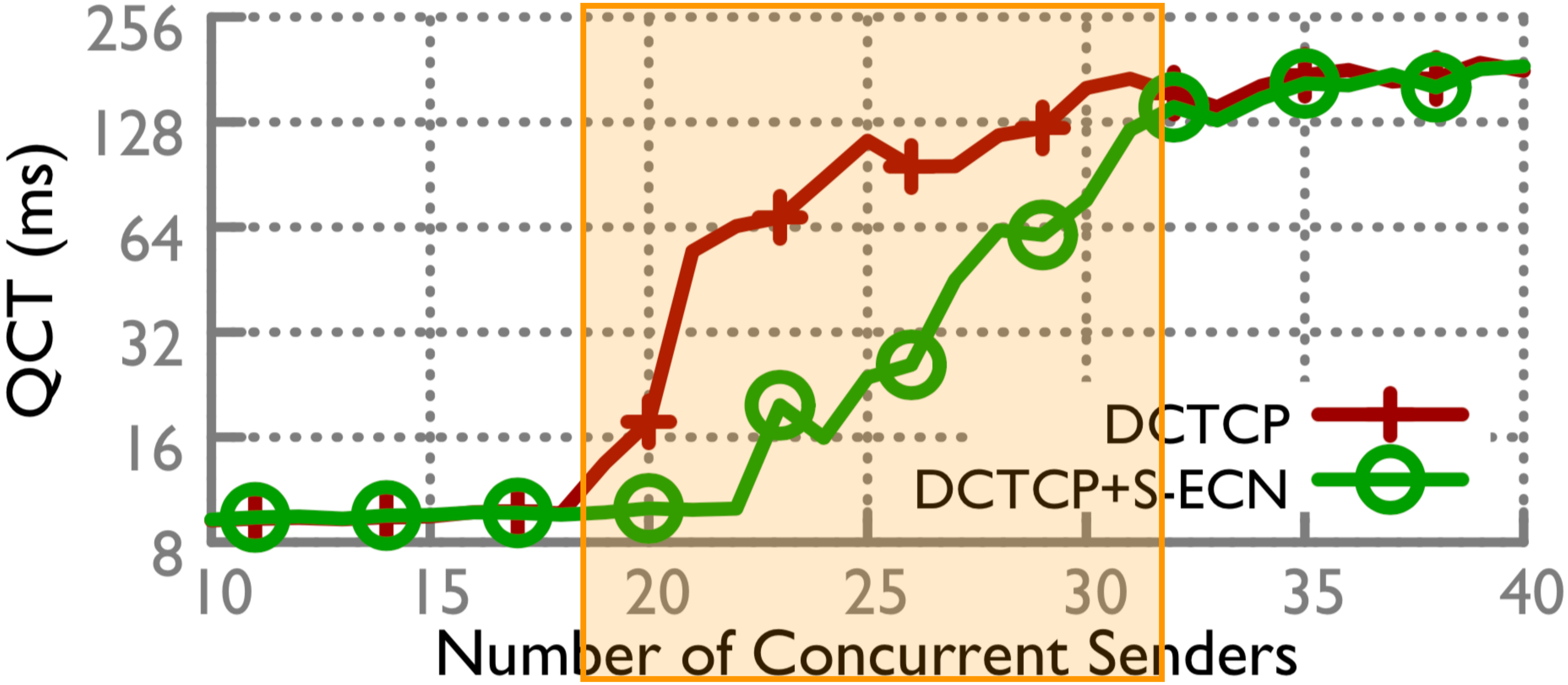
Evaluation
— Incast Performance

Exp. settings: query for total 1MB data, buffer size: 128KB



Evaluation
— Incast Performance

Exp. settings: query for total 1MB data, buffer size: 128KB



Better Incast Performance

Evaluation
— Benchmark Traffic

From DCTCP paper

Query Traffic (many-to-one):

- One server queries all other servers for total 100KB data
- Query arrival: Poisson

Background Traffic (one-to-one):

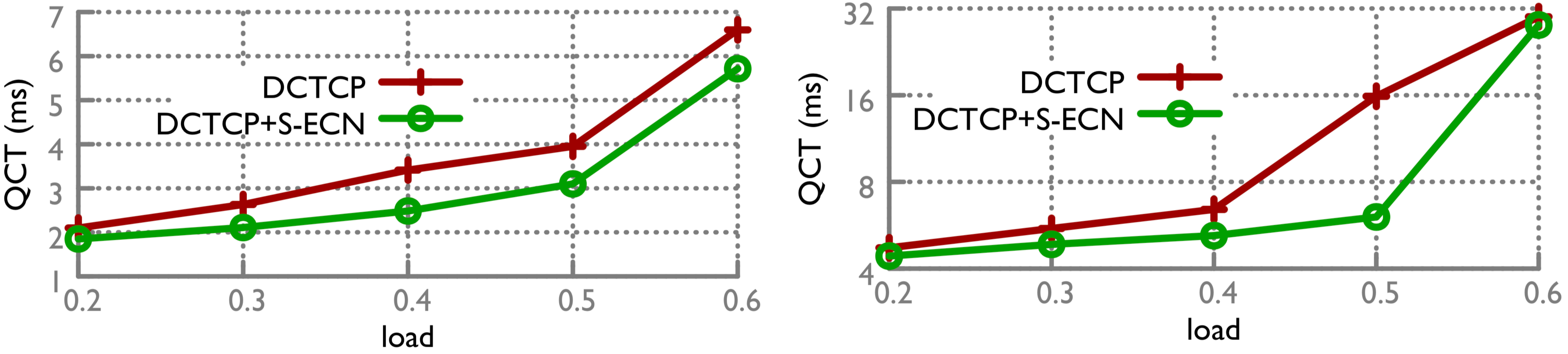
- Randomly choose sender and receiver
- Flow arrival: Poisson
- Flow size distribution

Evaluation
— Benchmark Traffic

Query Completion Time (QCT) of query traffic

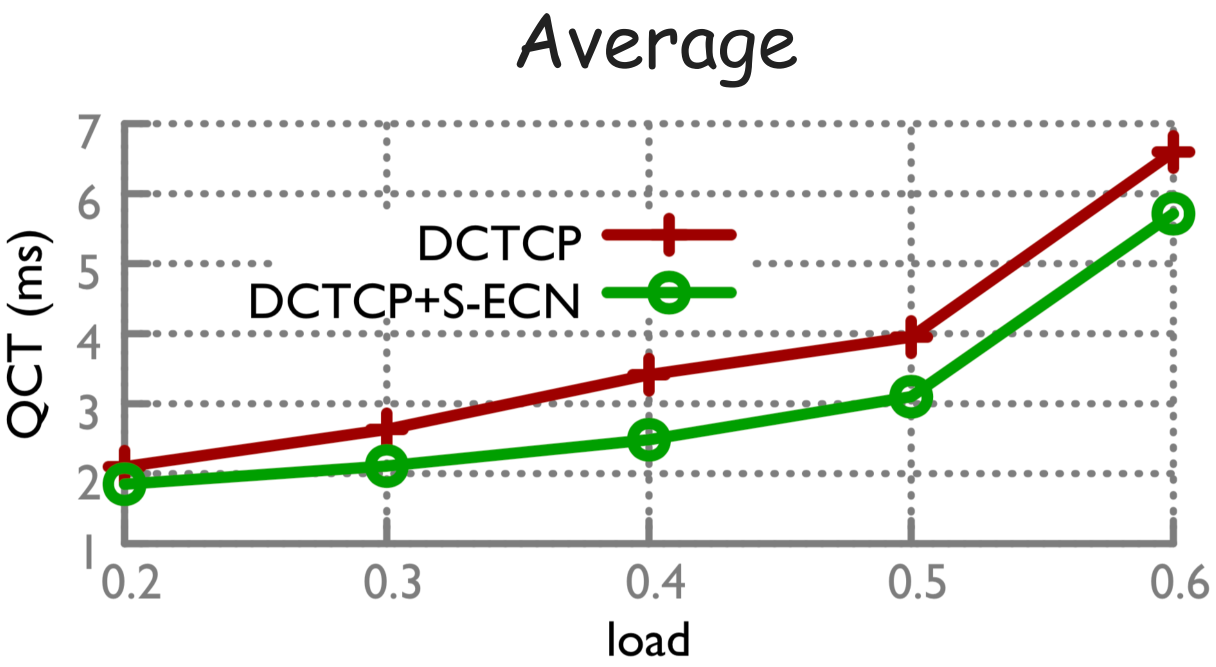
Average

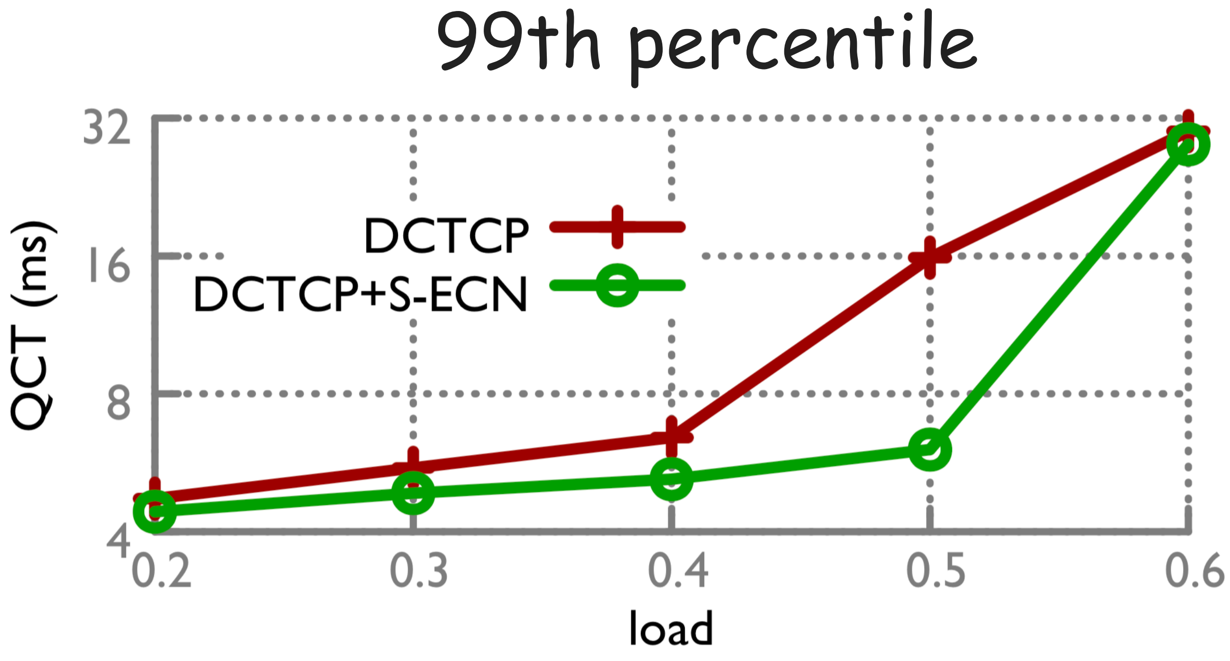
99th percentile



Evaluation
— Benchmark Traffic

Query Completion Time (QCT) of query traffic

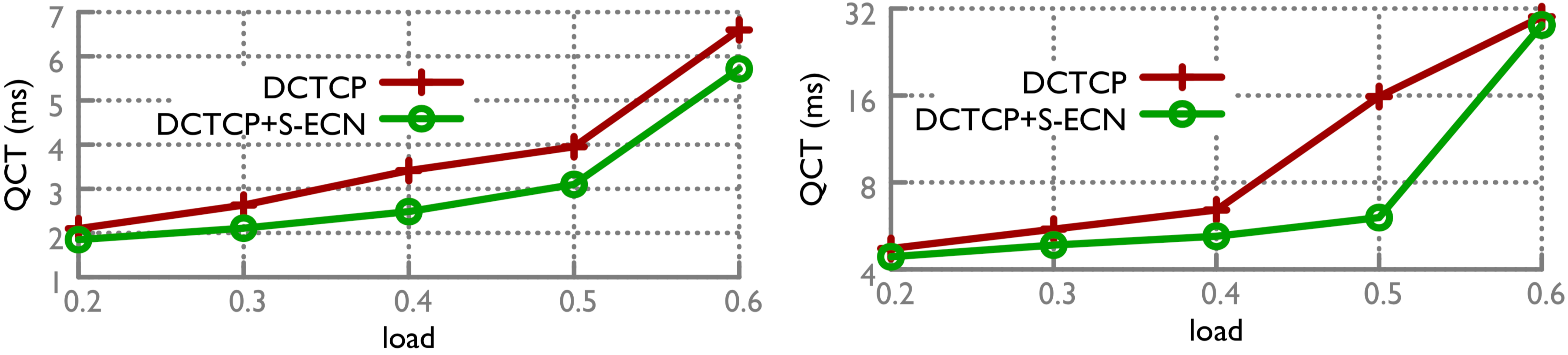




Avg. query completion time: reduced by ~12%-27%

Evaluation
— Benchmark Traffic

Query Completion Time (QCT) of query traffic



Average

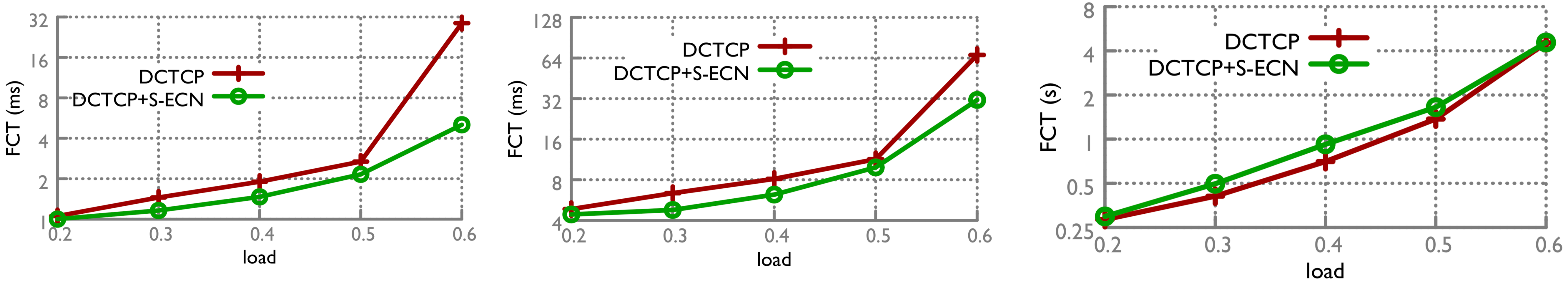
99th percentile

Avg. query completion time: reduced by ~12%-27%

99th percentile: reduced by ~6%-62%

Evaluation
— Benchmark Traffic

Flow Completion Time (FCT) of background traffic



(0,100KB]: Average

(0,100KB]: 99th percentile

(10MB, +∞)

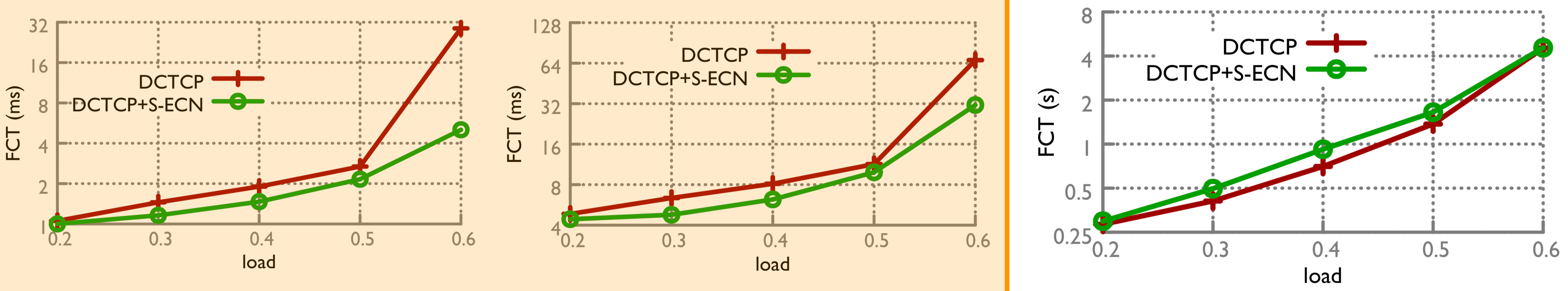
Evaluation
— Benchmark Traffic

Flow Completion Time (FCT) of background traffic

(0,100KB]: Average

(0,100KB]: 99th percentile

(10MB, +∞)

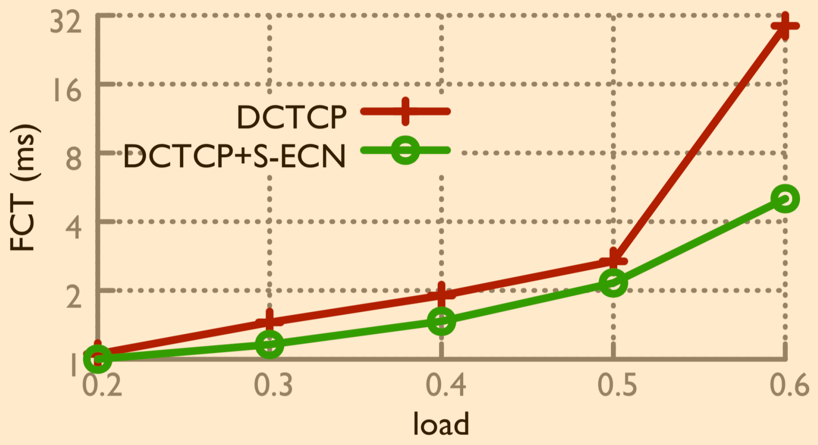


Small flows: finish faster

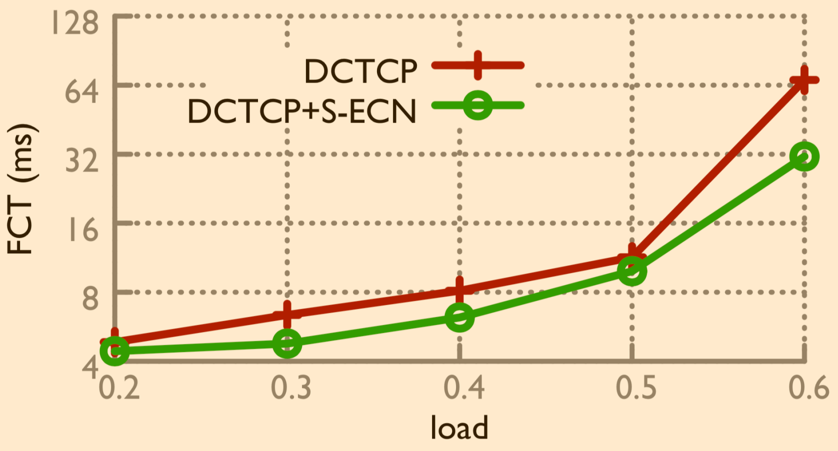
Evaluation
— Benchmark Traffic

Flow Completion Time (FCT) of background traffic

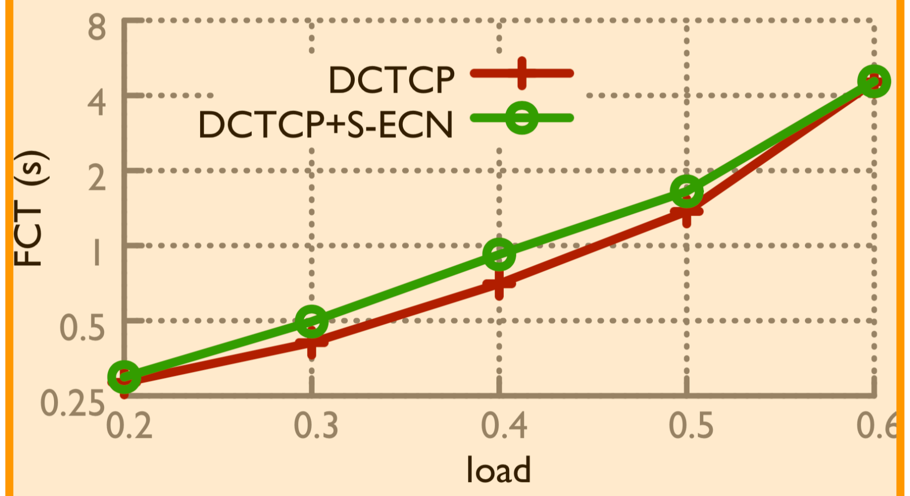
(0,100KB]: Average



(0,100KB]: 99th percentile



(10MB, +∞)

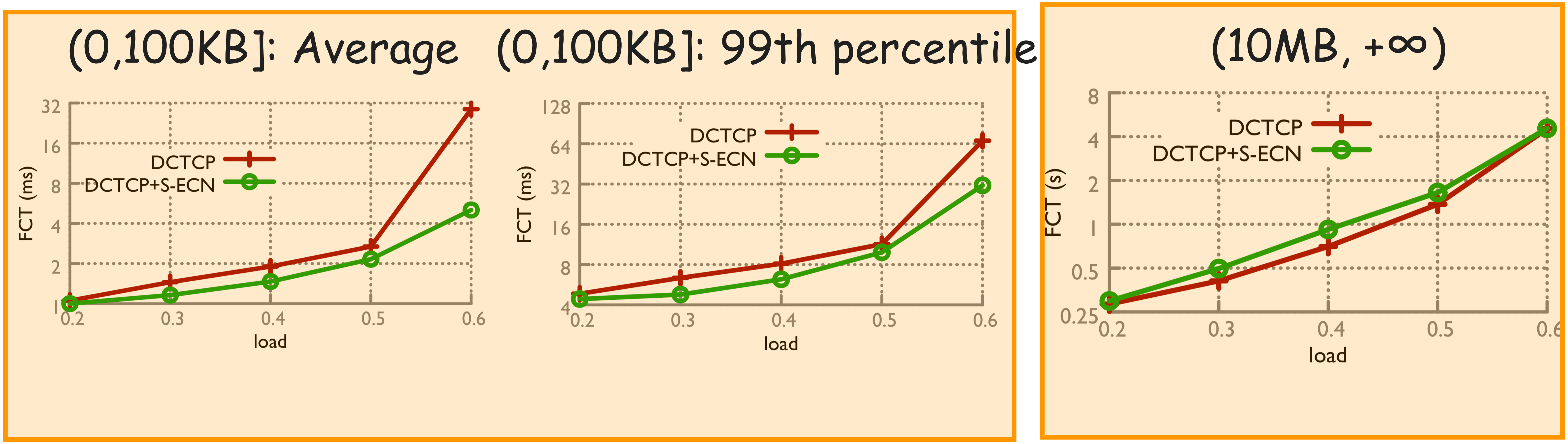


Small flows: finish faster

Large flows: finish slower

Evaluation
— Benchmark Traffic

Flow Completion Time (FCT) of background traffic



(0,100KB]: Average

(0,100KB]: 99th percentile

(10MB, +∞)

Small flows: finish faster

Large flows: finish slower

☹ **Future Work**

Conclusion

- Observing and Analyzing dynamic behaviors of micro-burst
  - The self-clocking system, congestion control, and bottleneck link capacity jointly dominate the evolution of micro-burst
  - Dynamic behaviors of micro-burst can be described by slope of queue length evolution
  - **Implications**: Conventional burst mitigation approaches are ineffective

- S-ECN marking Scheme
  - Probability marking scheme based-on slope
  - suppressed sharp queue length increasing by 2x

Thank you!