Adapting TCP for Reconfigurable Datacenter Networks

Matthew K. Mukerjee*†, **Christopher Canel***, Weiyang Wang○, Daehyeok Kim*‡,

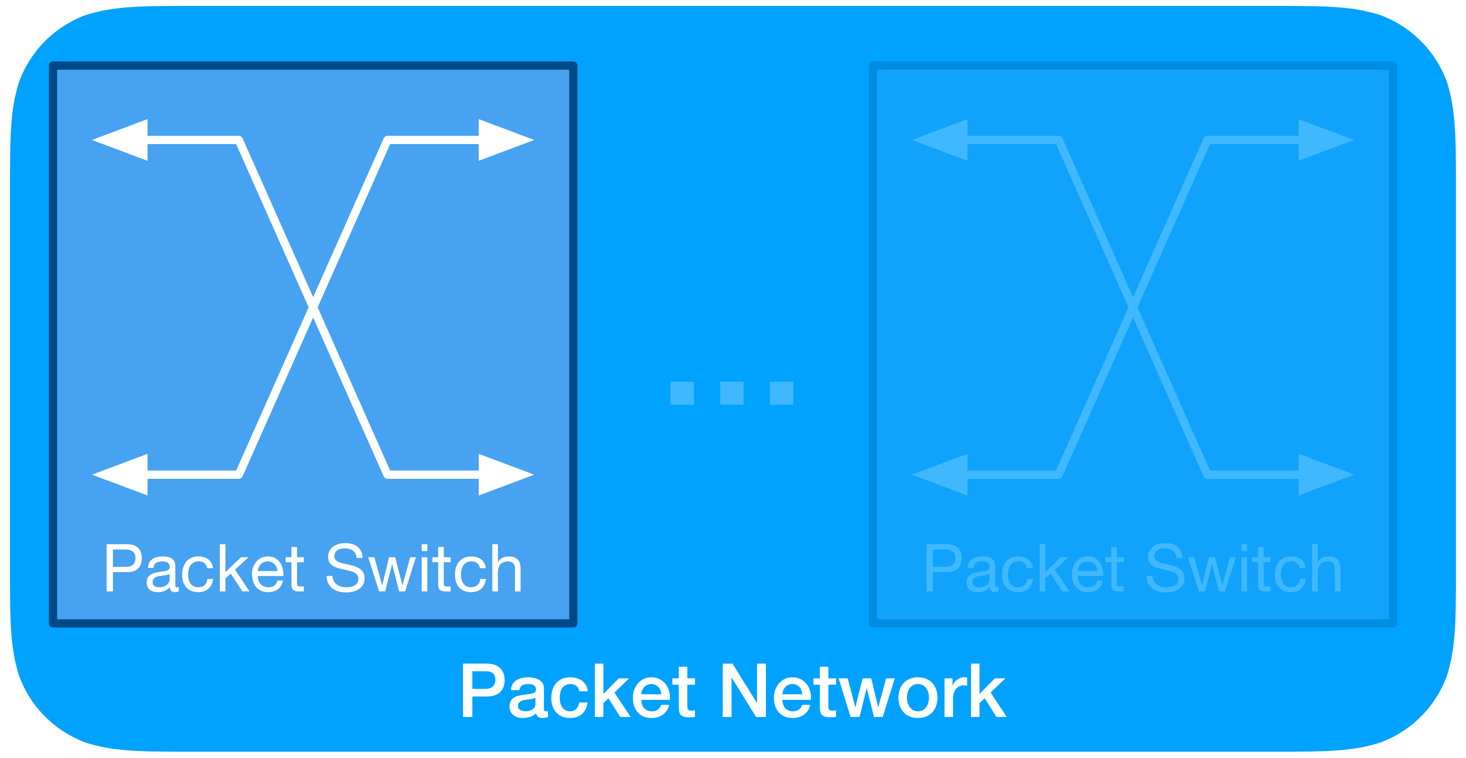
Srinivasan Seshan*, Alex C. Snoeren○

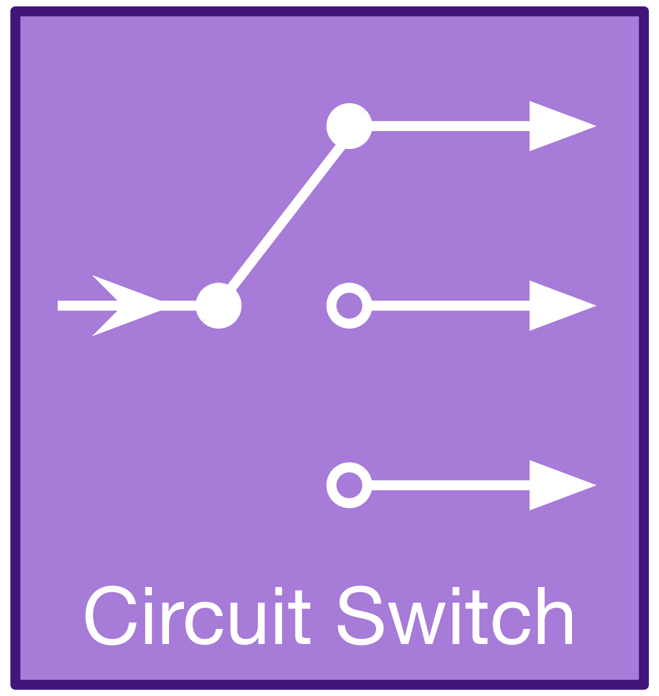
**Carnegie Mellon University, ○UC San Diego, †Nefeli Networks, ‡Microsoft Research*

February 26, 2020

Reconfigurable Datacenter Network (RDCN)

all-to-all connectivity

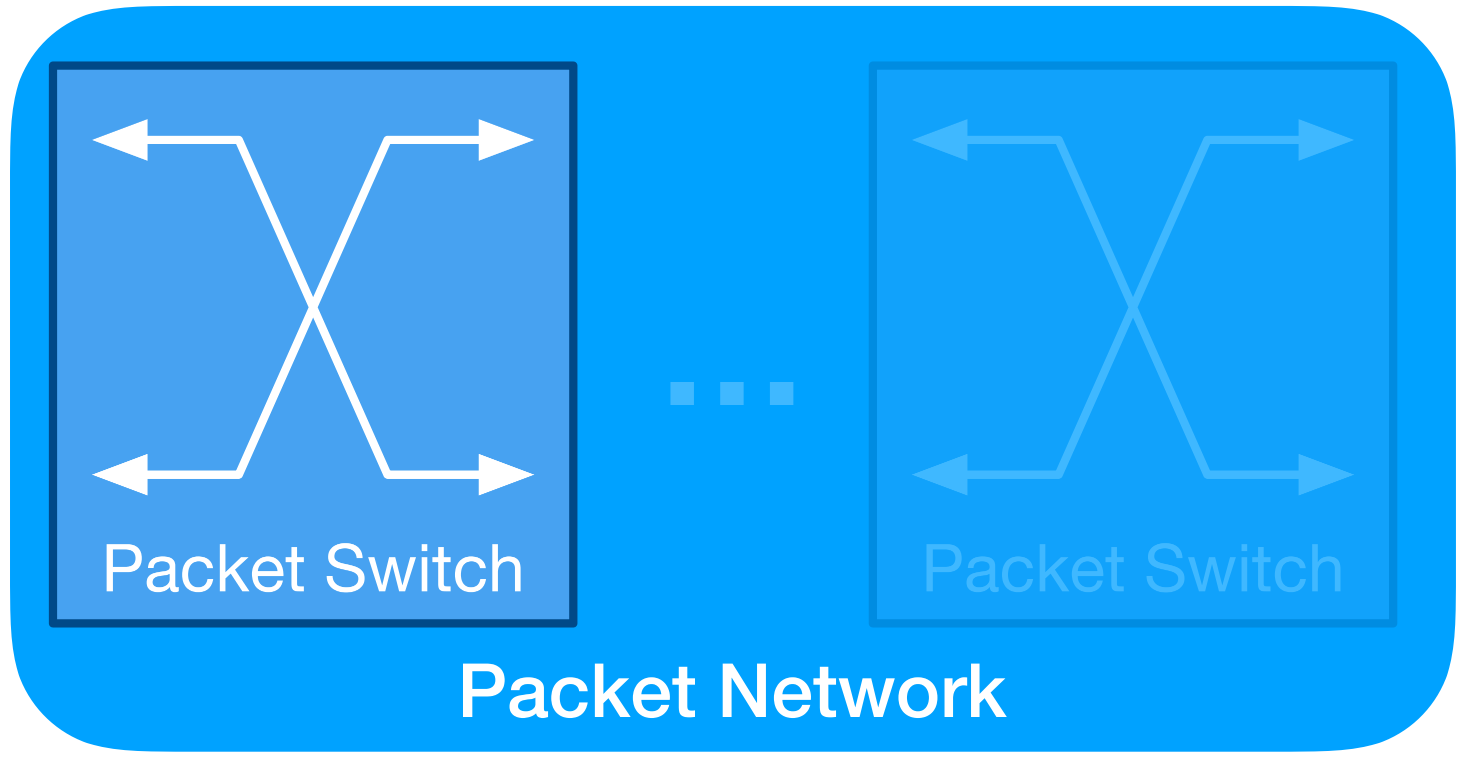


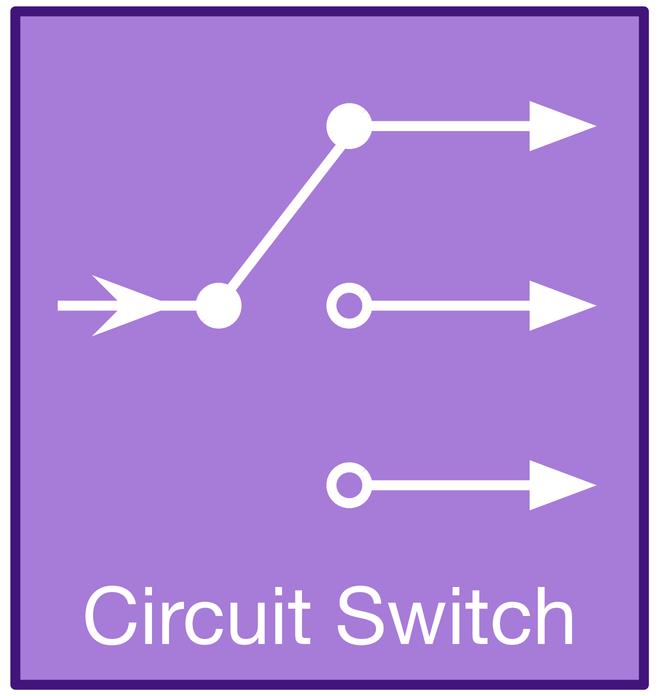


higher bandwidth,
between certain racks

Reconfigurable Datacenter Network (RDCN)

all-to-all connectivity

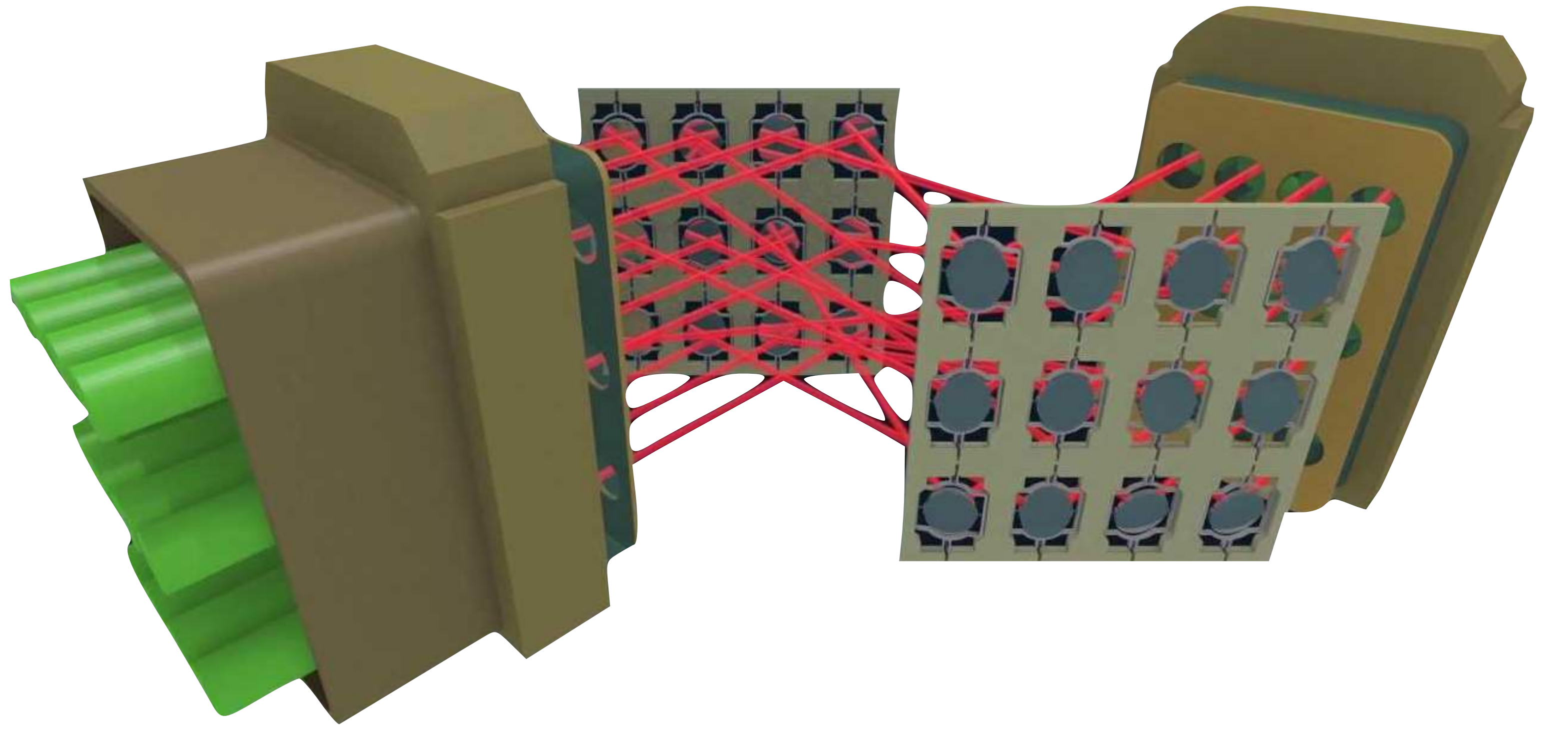




higher bandwidth, between certain racks

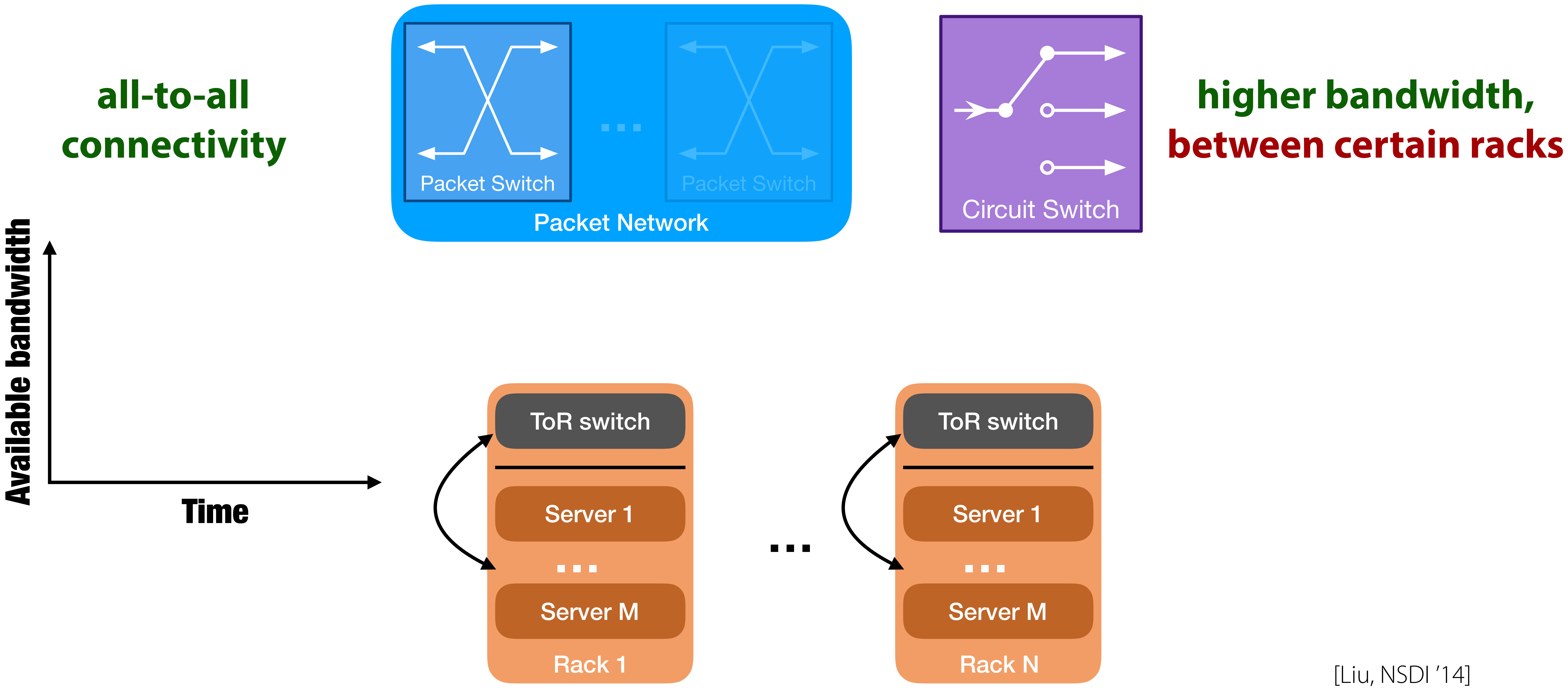
60GHz wireless

free-space optics



optical circuits

Reconfigurable Datacenter Network (RDCN)

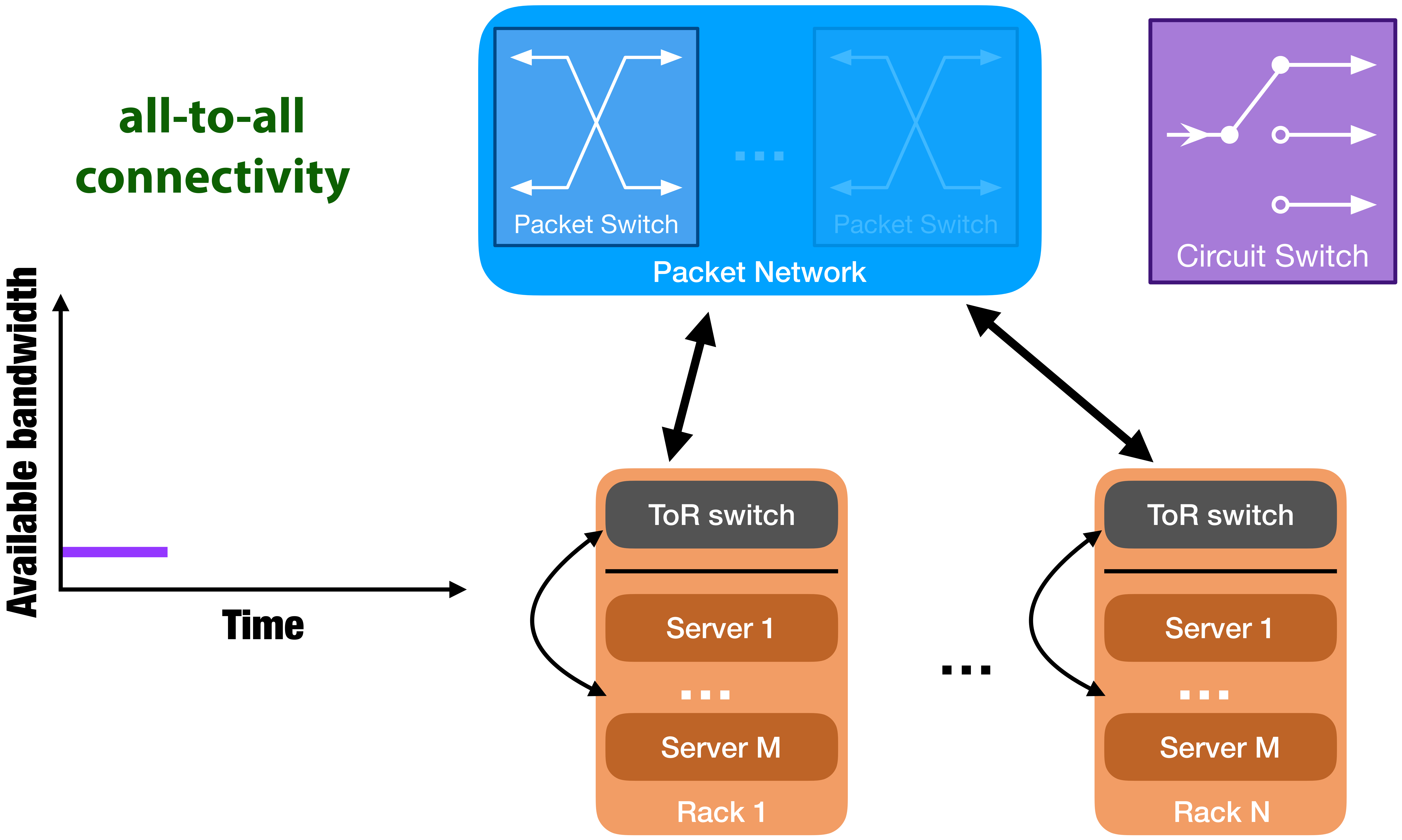


[Liu, NSDI '14]

Reconfigurable Datacenter Network (RDCN)

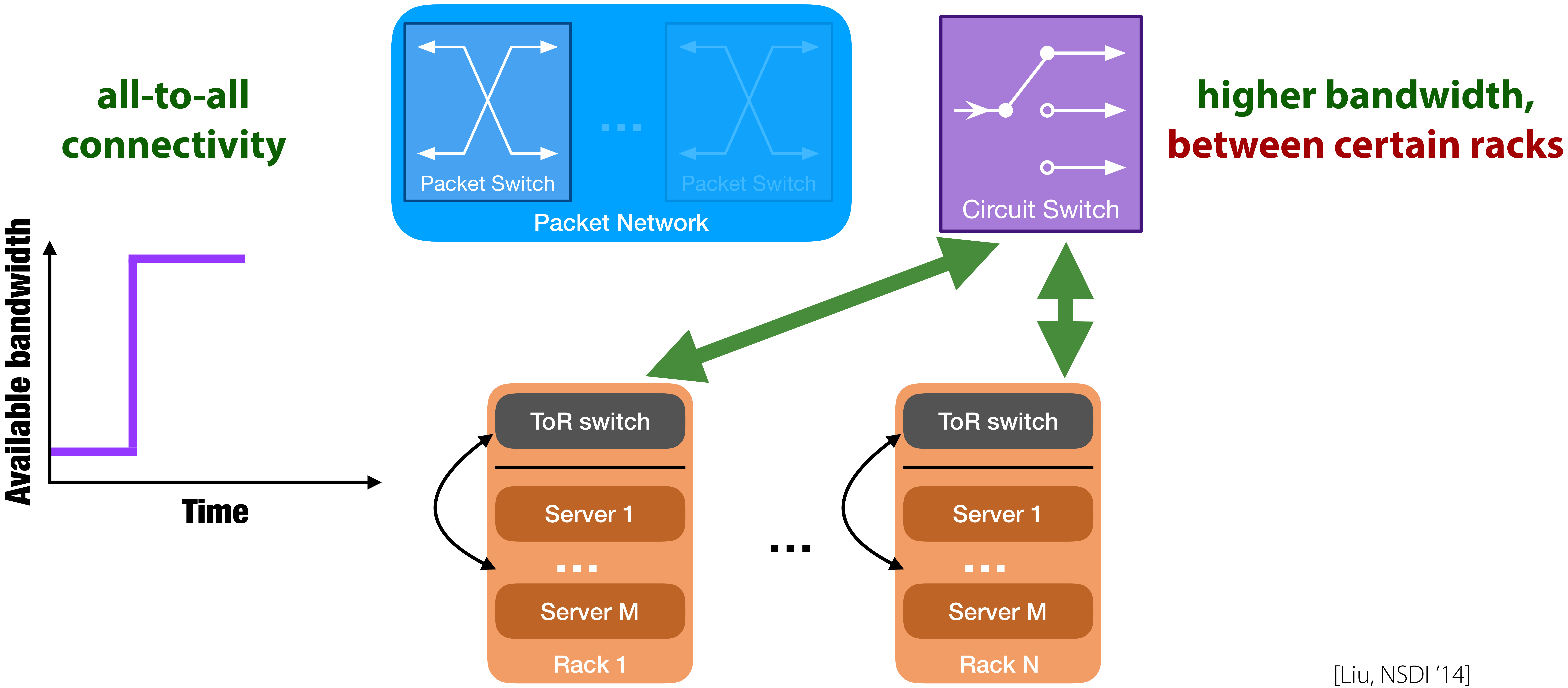
all-to-all connectivity

higher bandwidth, between certain racks



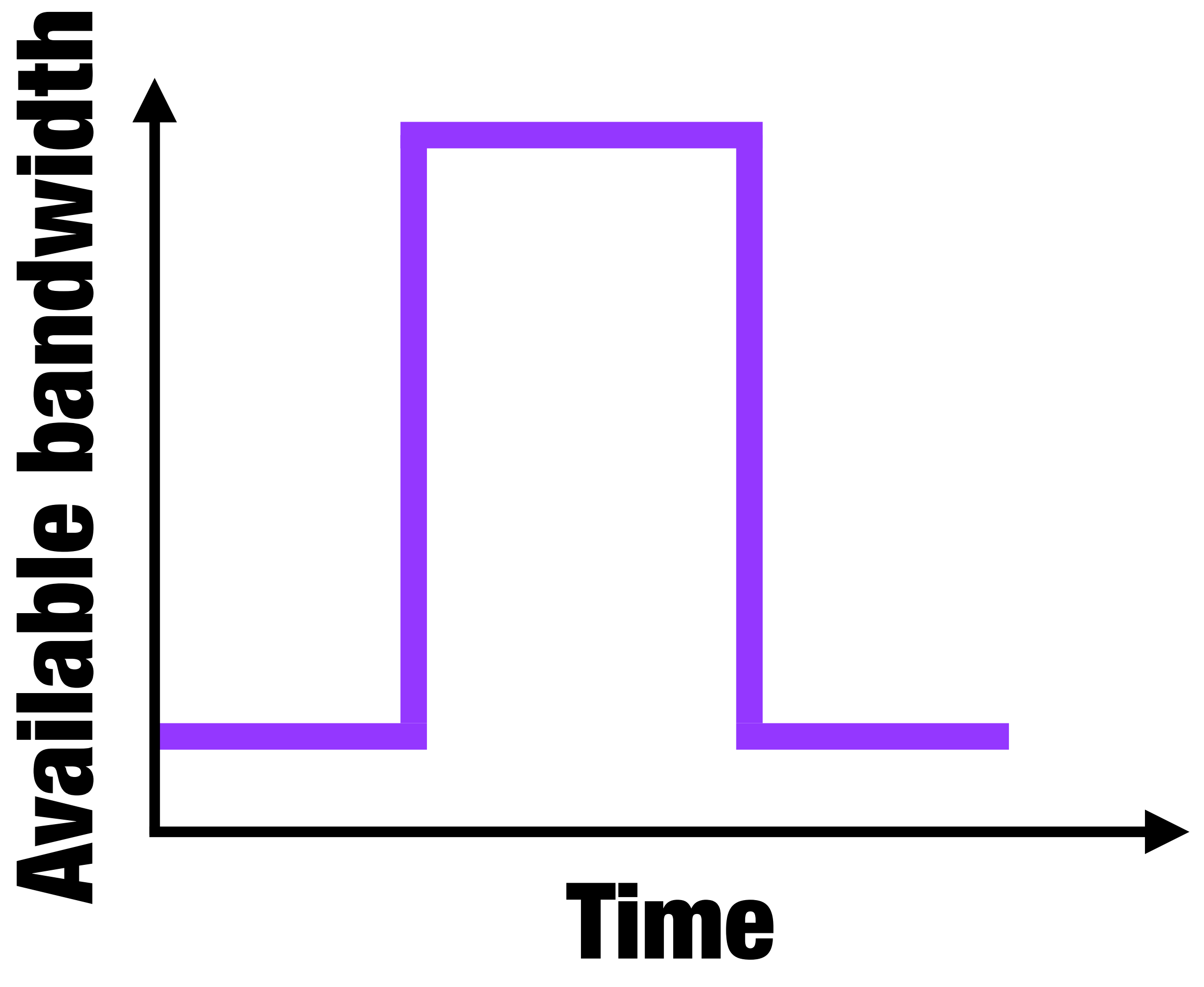
[Liu, NSDI '14]

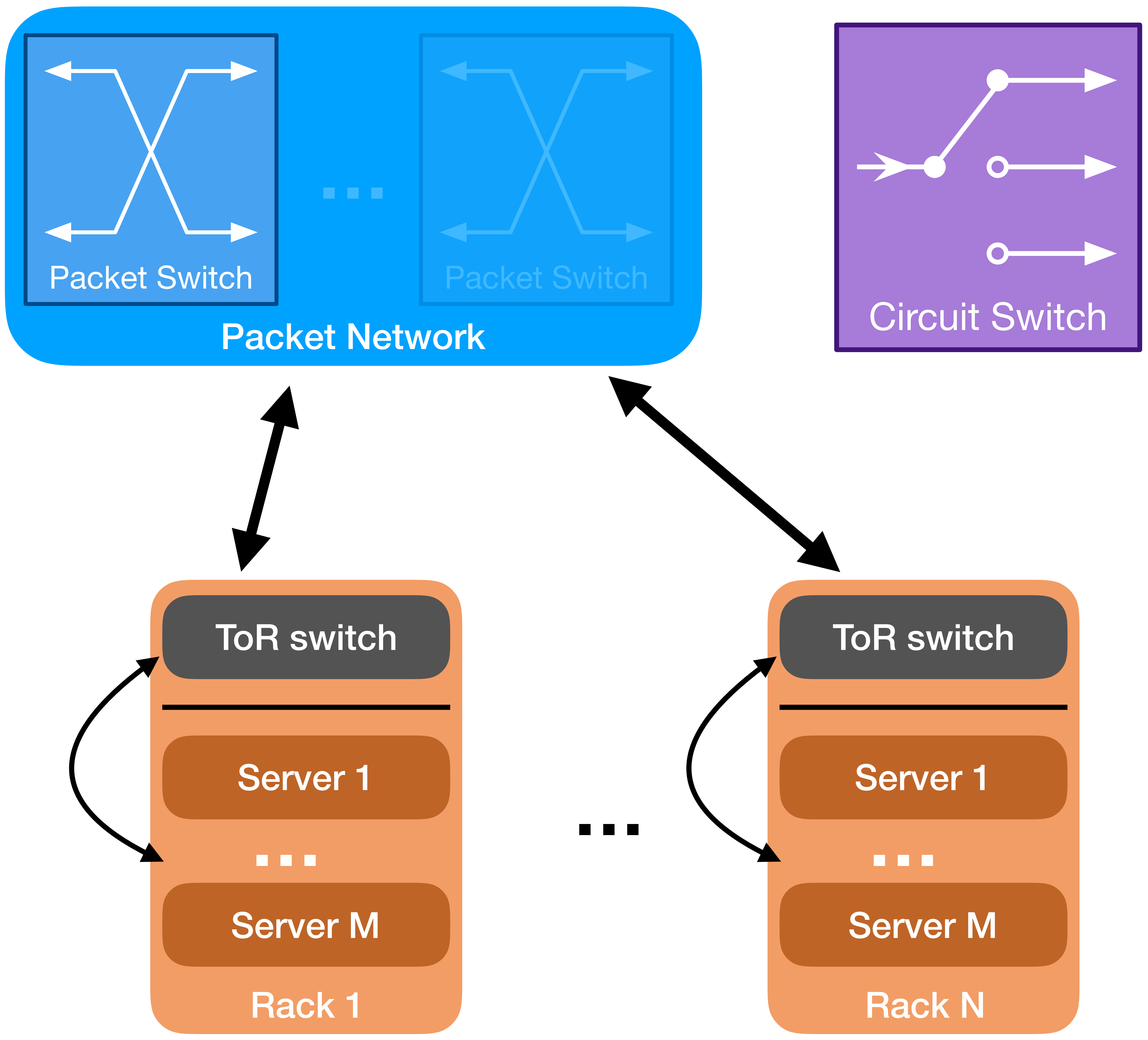
Reconfigurable Datacenter Network (RDCN)



[Liu, NSDI '14]

Reconfigurable Datacenter Network (RDCN)

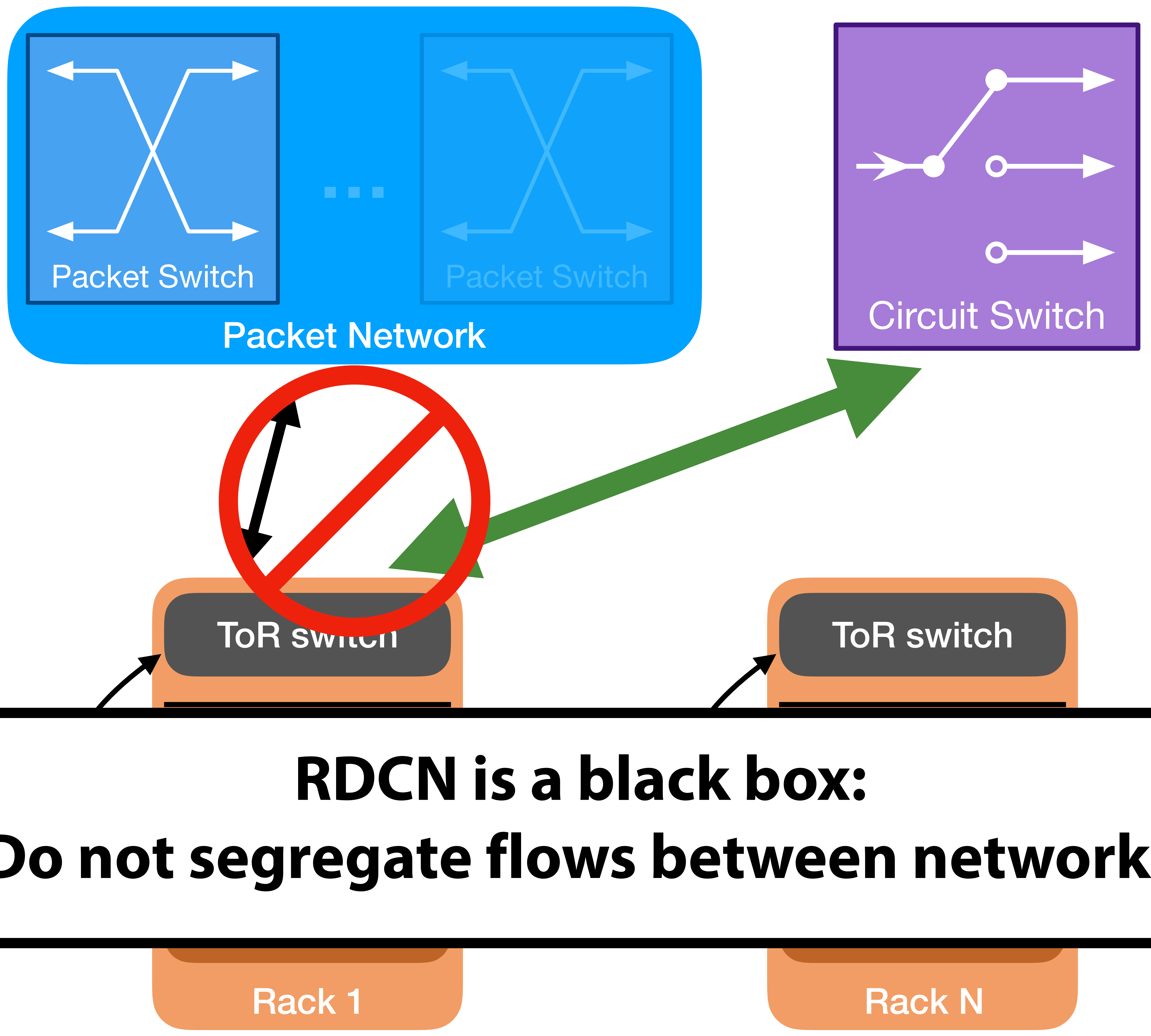




[Liu, NSDI '14]

Reconfigurable Datacenter Network (RDCN)

all-to-all connectivity



higher bandwidth, between certain racks

RDCN is a black box:
Do not segregate flows between networks

[Liu, NSDI '14]

2010: RDCNs speed up DC workloads

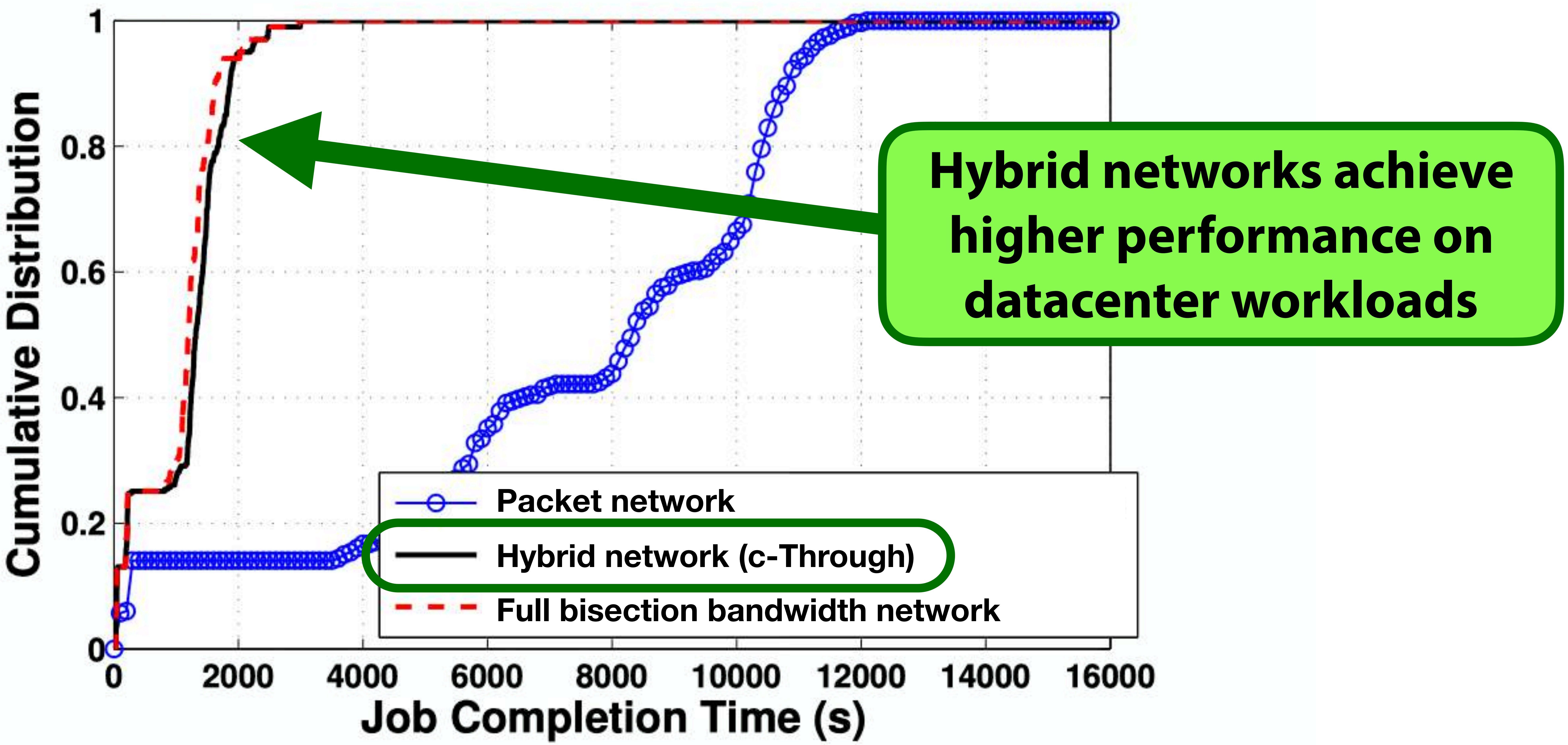
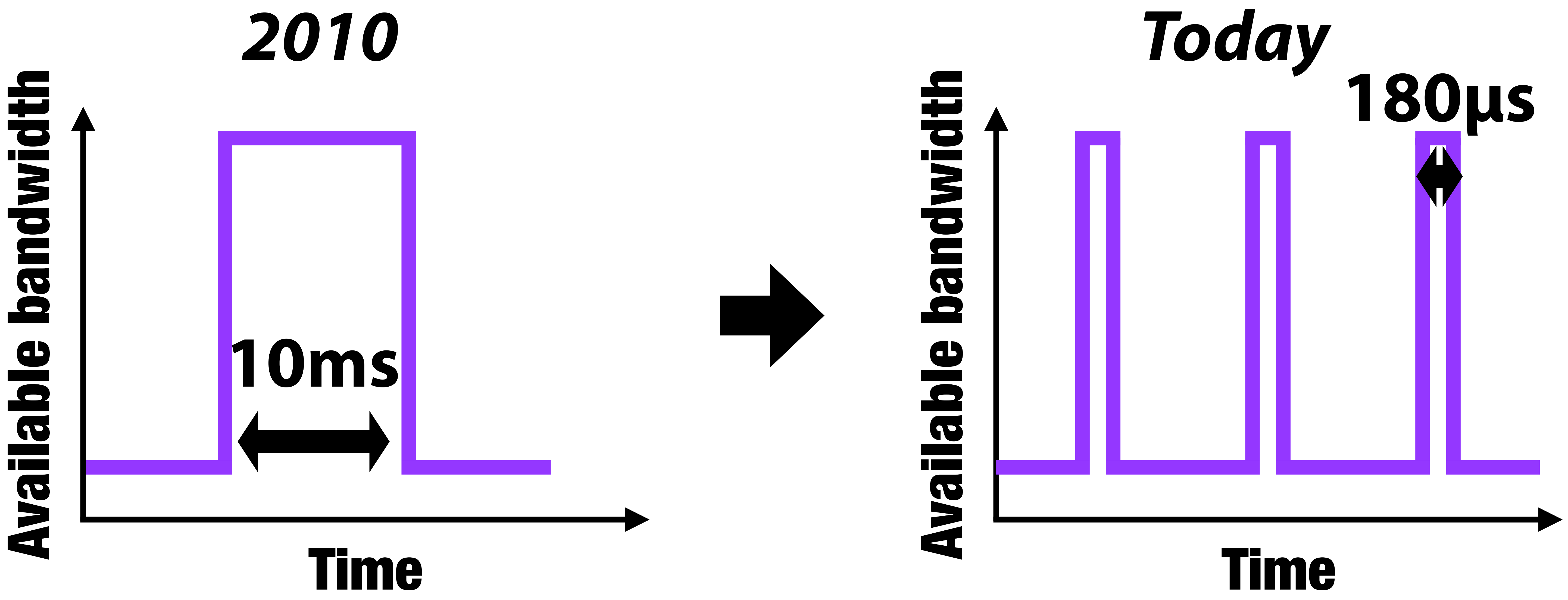


Figure 9: The completion of Hadoop Gridmix tasks

[Wang, SIGCOMM '10]

Today's RDCNs reconfigure 10x as often

Advances in circuit switch technology have led to a 10x reduction in reconfiguration delay ⇒ today, circuits can reconfigure much more frequently

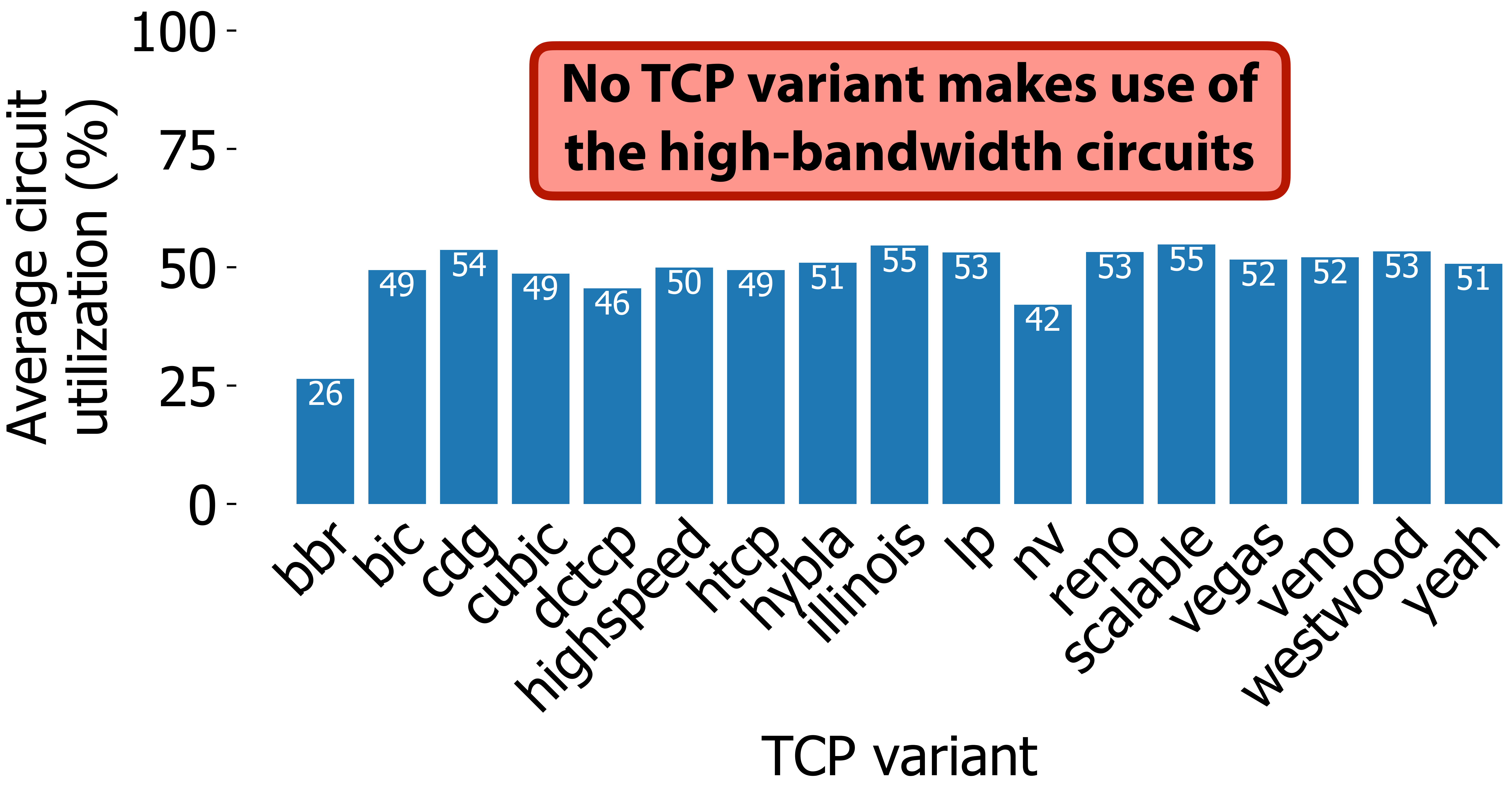


Better for datacenters: More flexibility to support dynamic workloads
Better for hosts: Less data must be available to saturate higher bandwidth NW

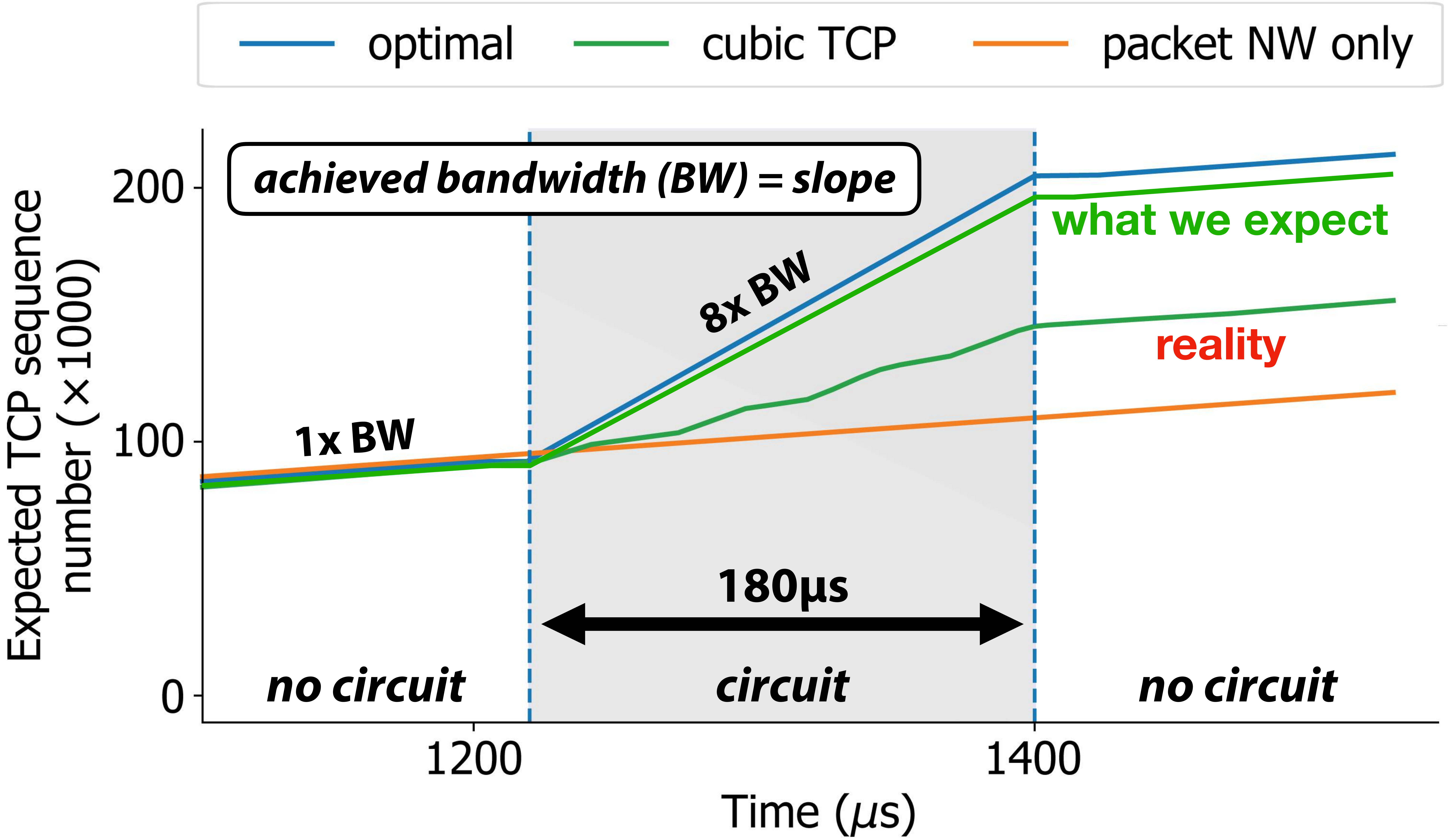
[Porter, SIGCOMM '13]

Short-lived circuits pose a problem for TCP

16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s



TCP cannot ramp up during short circuits



What is the problem?

All TCP variants are designed to adapt to changing network conditions

- E.g., congestion, bottleneck links, RTT

But ***bandwidth fluctuations*** in modern RDCN are an order of magnitude ***more frequent*** (10x shorter circuit duration) and ***more substantial*** (10x higher bandwidth) than TCP is designed to handle

- RDCNs break the implicit assumption of relatively-stable network conditions

This requires an ***order-of-magnitude shift in how fast TCP reacts***

This talk: Our 2-part solution

In-network: Use information about upcoming circuits to transparently "trick" TCP into ramping up more aggressively

- High utilization, at the cost of tail latency

At endhosts: New TCP variant, ***reTCP***, that explicitly reacts to circuit state changes

- Mitigates tail latency penalty

The two techniques can be deployed separately, but work best together

This talk: Our 2-part solution

In-network: Use information about upcoming circuits to transparently "trick" TCP into ramping up more aggressively

- High utilization, at the cost of tail latency

At endhosts: New TCP variant, ***reTCP***, that explicitly reacts to circuit state changes

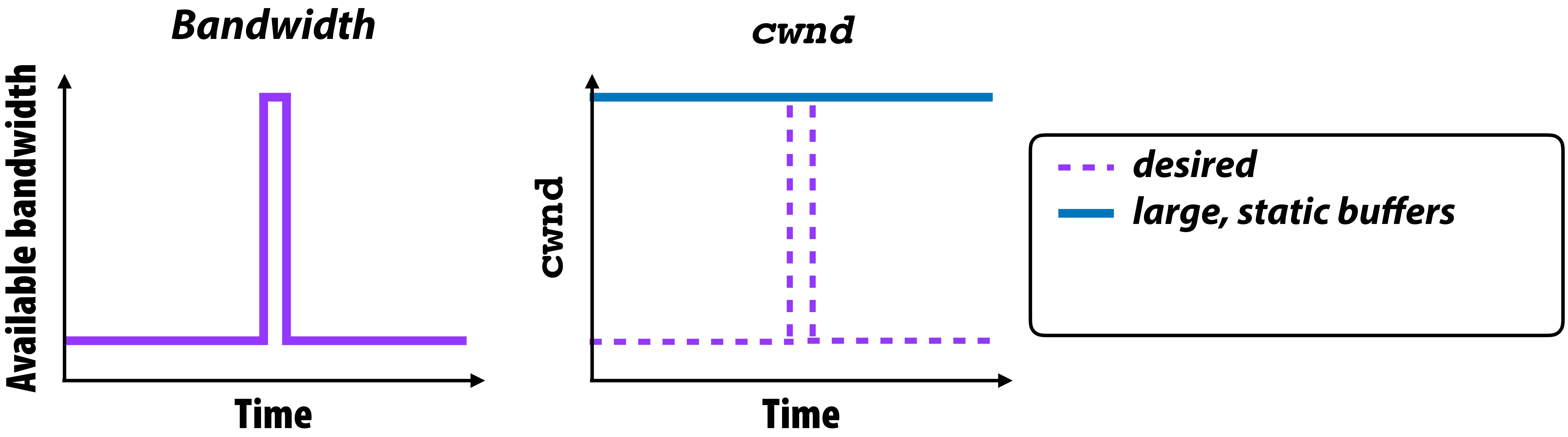
- Mitigates tail latency penalty

The two techniques can be deployed separately, but work best together

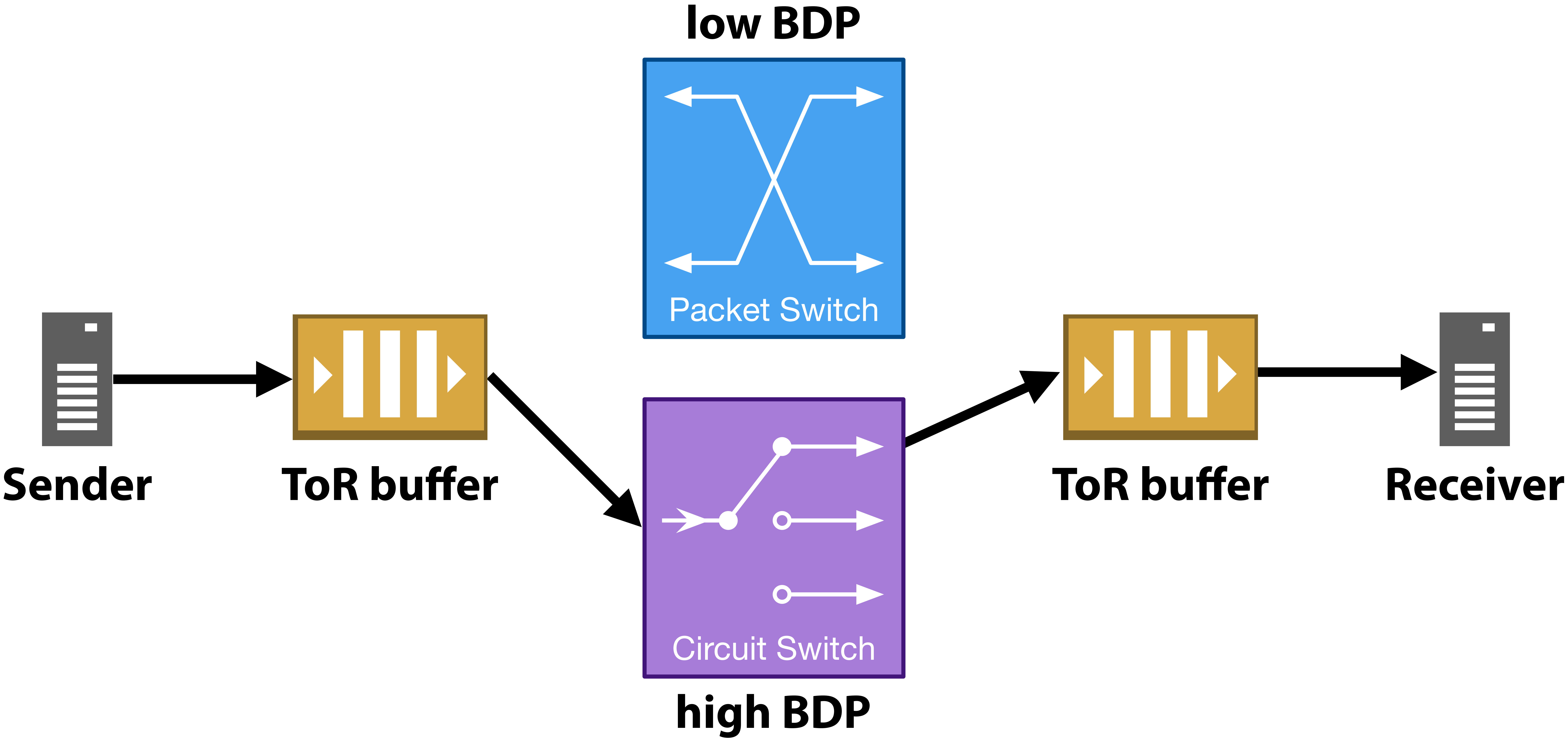
Naïve idea: Enlarge switch buffers

Want we want: TCP's congestion window (**`cwnd`**) to parallel the BW fluctuations

First attempt: Make **`cwnd`** large all the time ***How?*** Use large ToR buffers

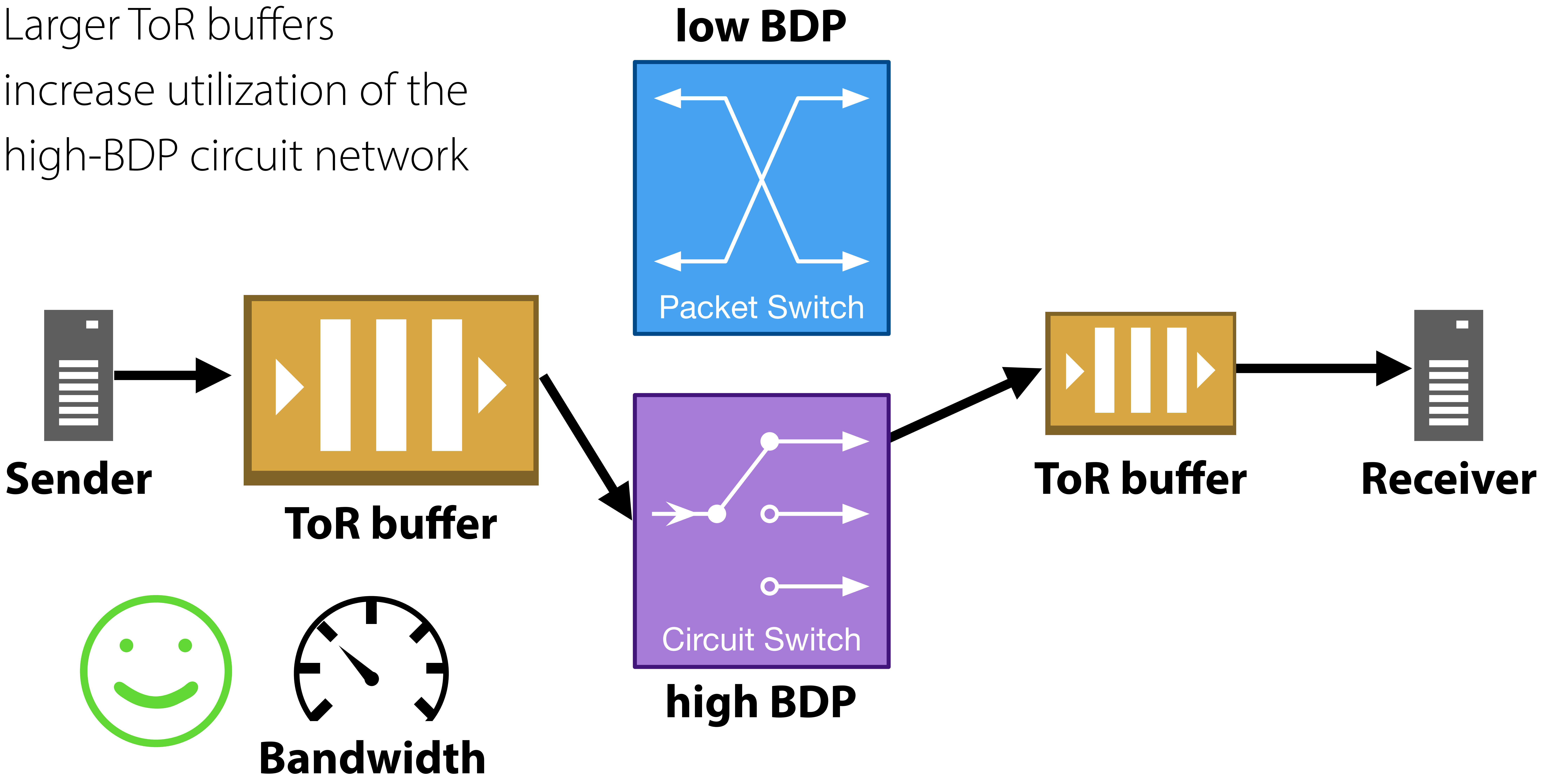


Naïve idea: Enlarge switch buffers

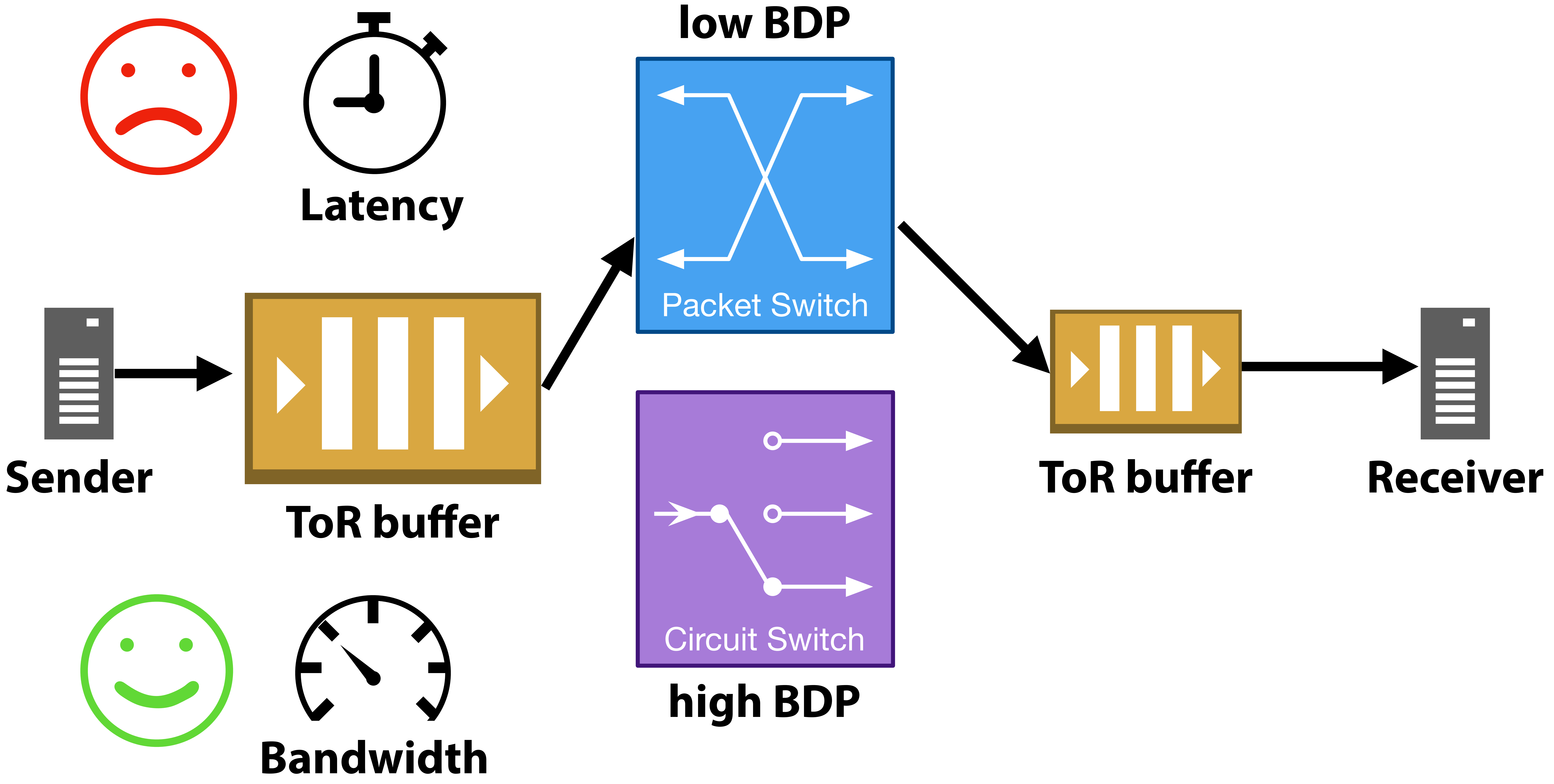


Naïve idea: Enlarge switch buffers

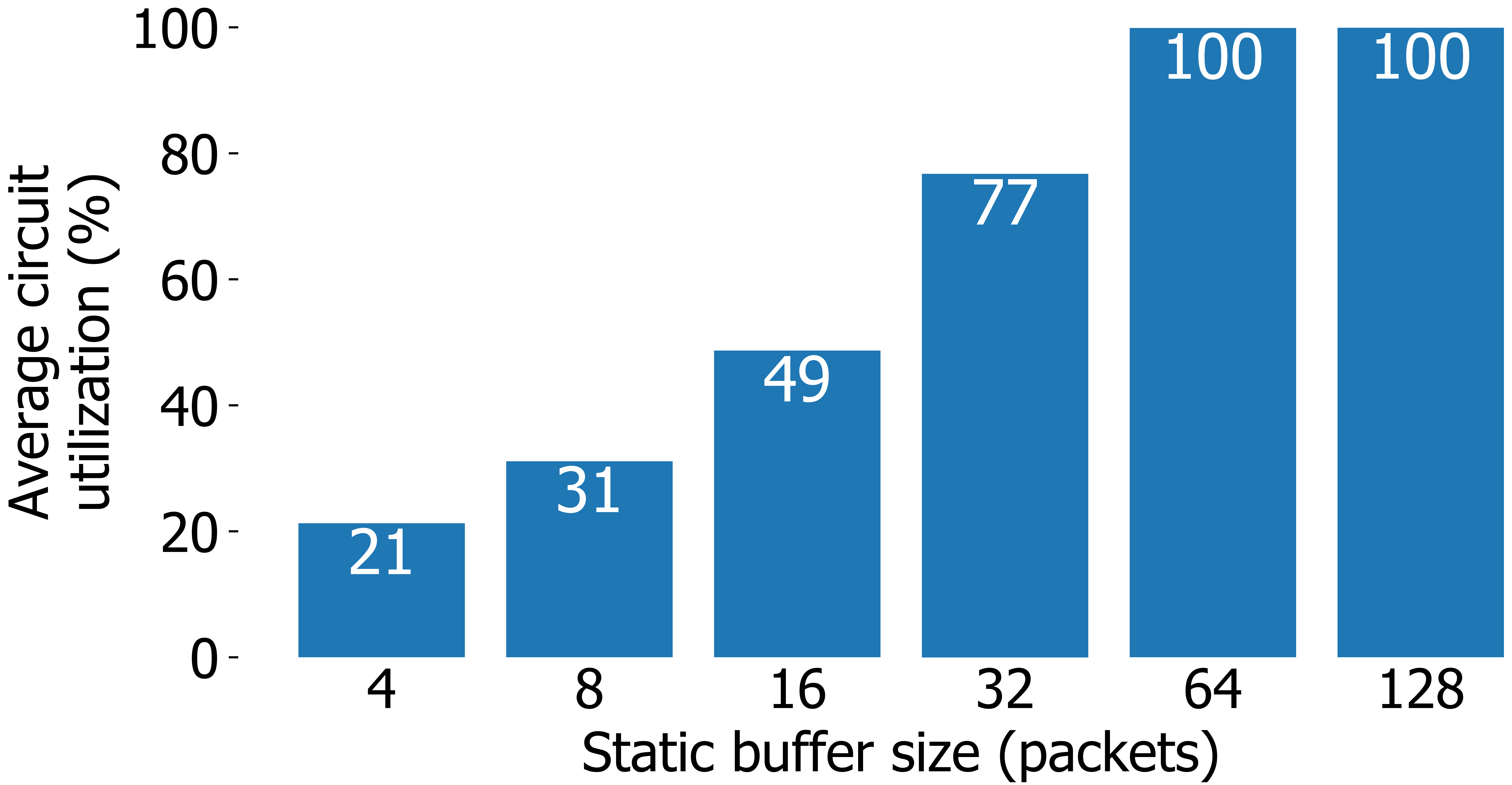
Larger ToR buffers increase utilization of the high-BDP circuit network



Naïve idea: Enlarge switch buffers

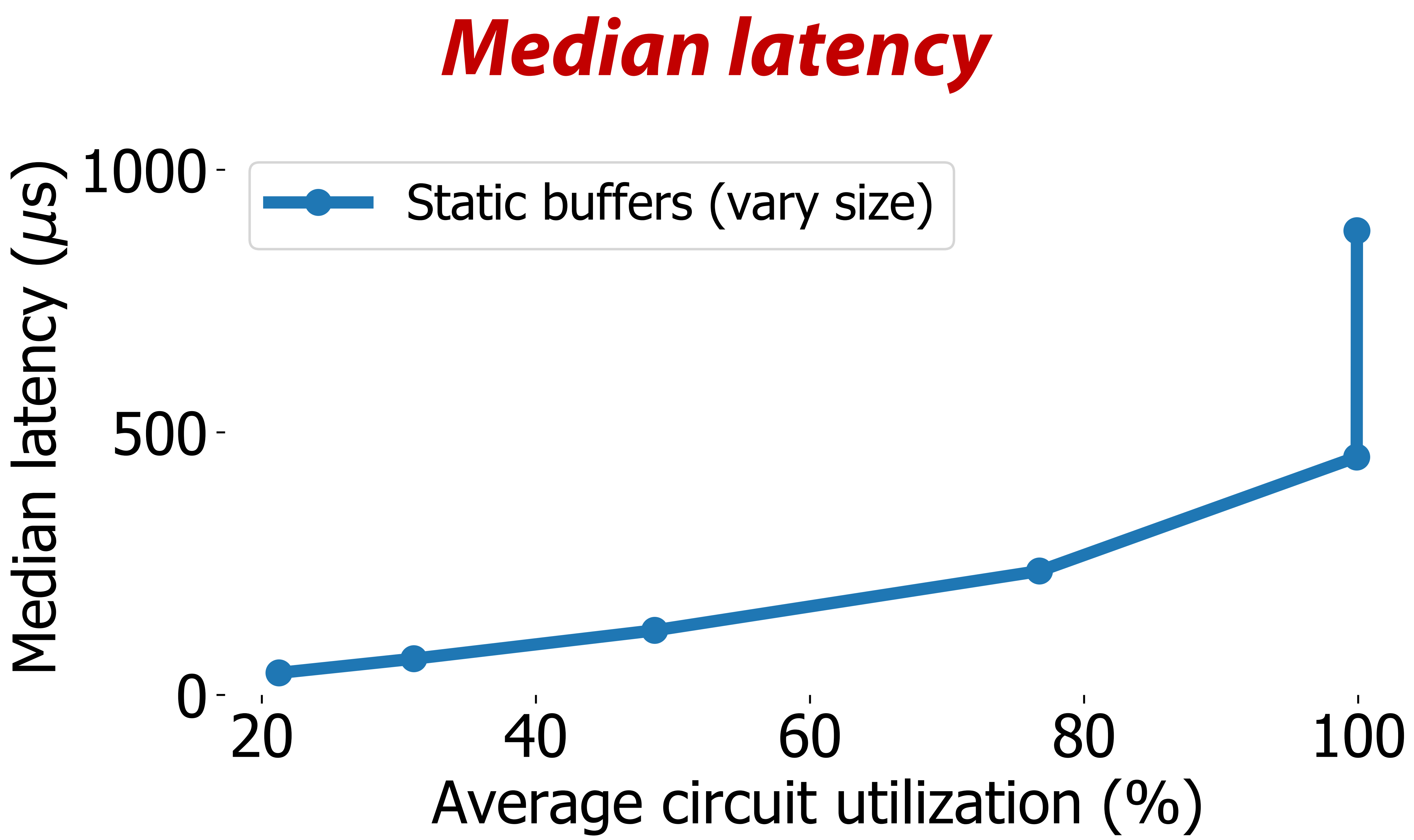


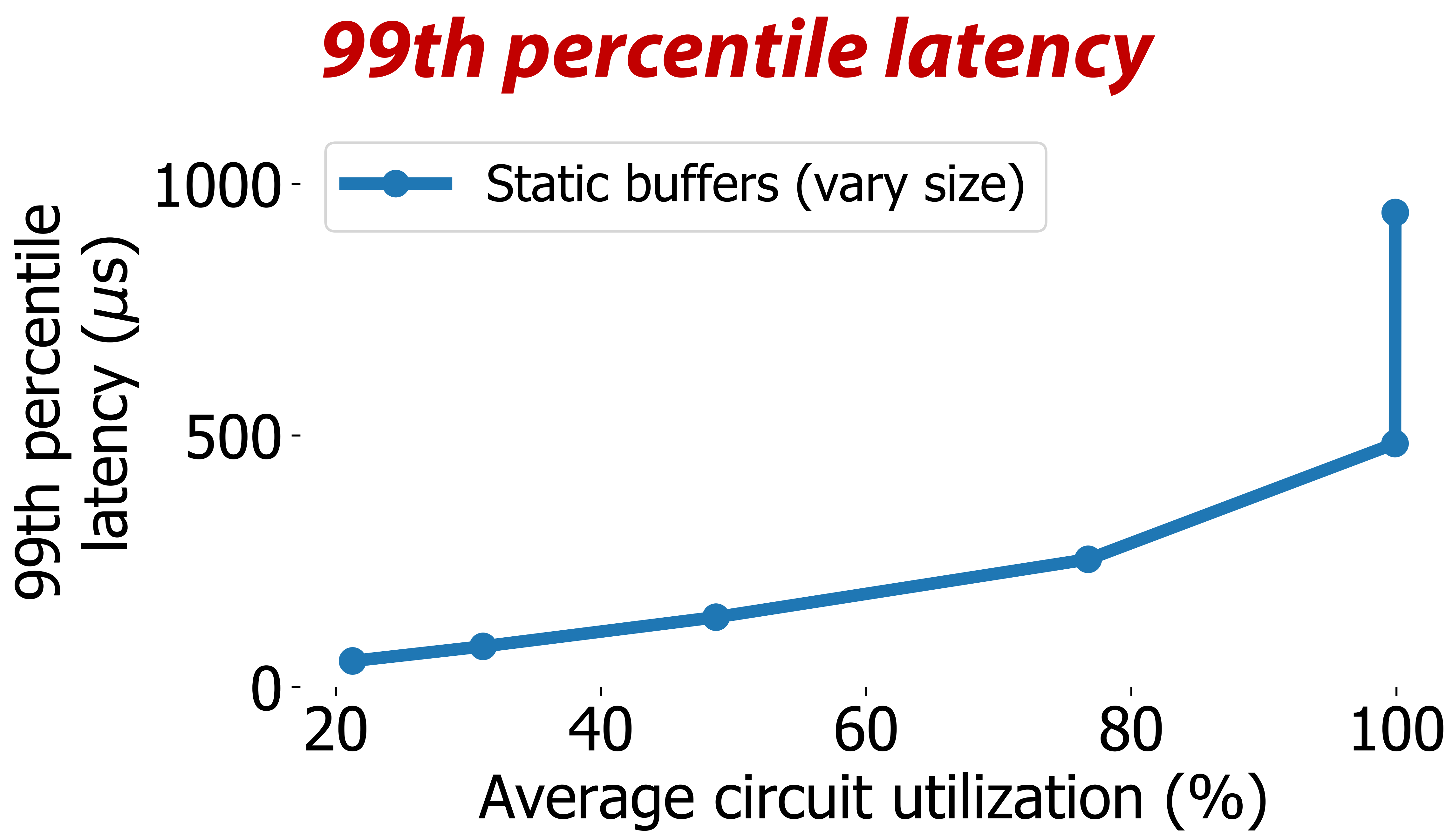
Large queues increase utilization…



16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s

…but result in high latency





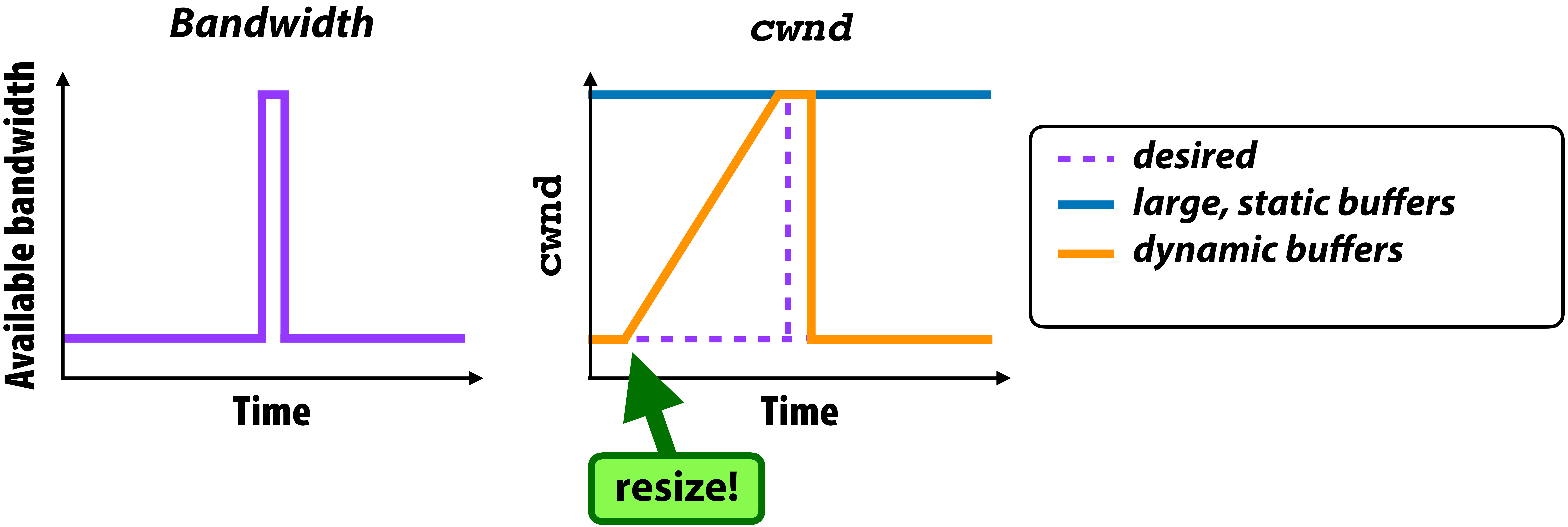
How can we improve this latency?

16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s

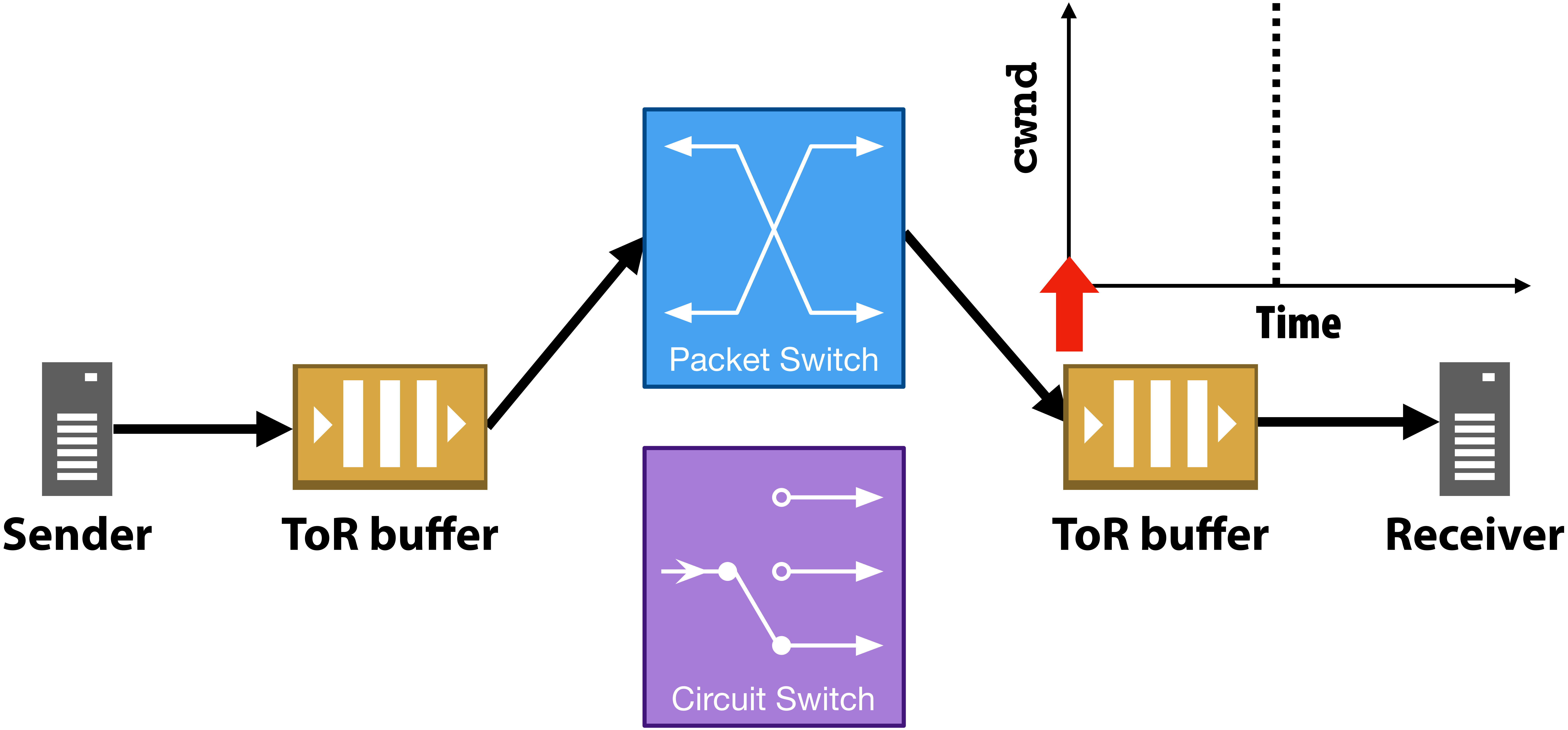
Use large buffers only when circuit is up

Dynamic buffer resizing: Before a circuit begins, transparently enlarge ToR buffers

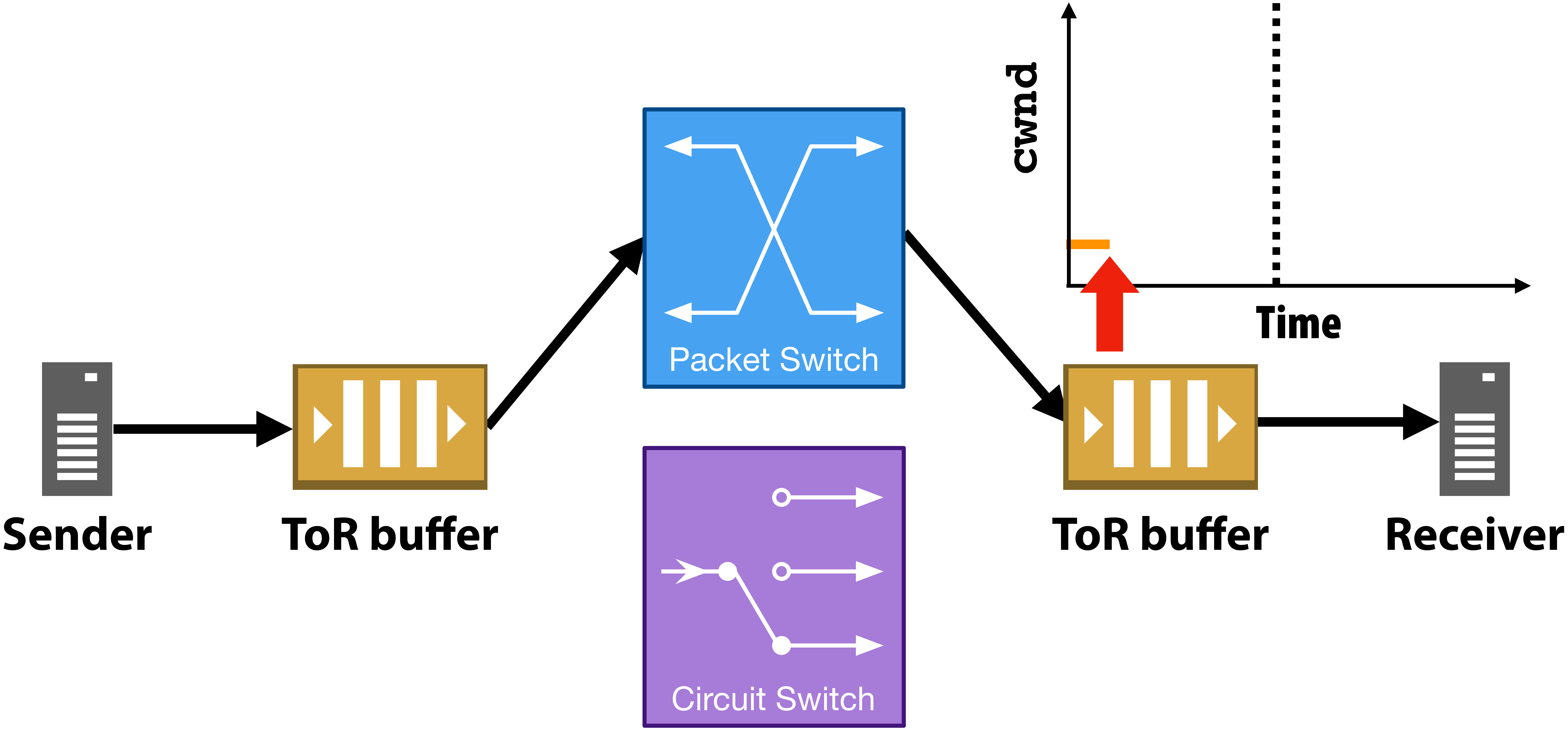
Full circuit utilization with a latency degradation only during ramp-up period



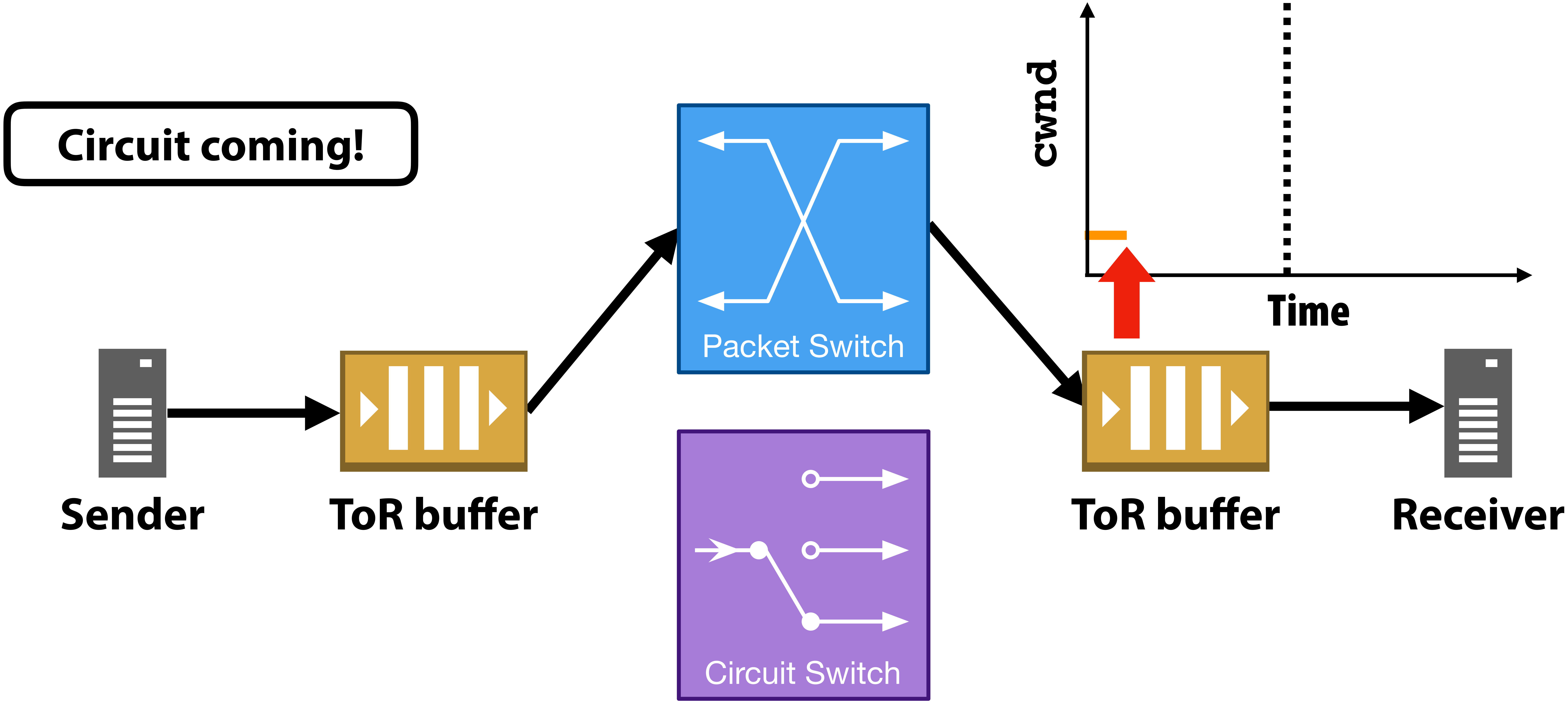
Resize ToR buffers before circuit begins



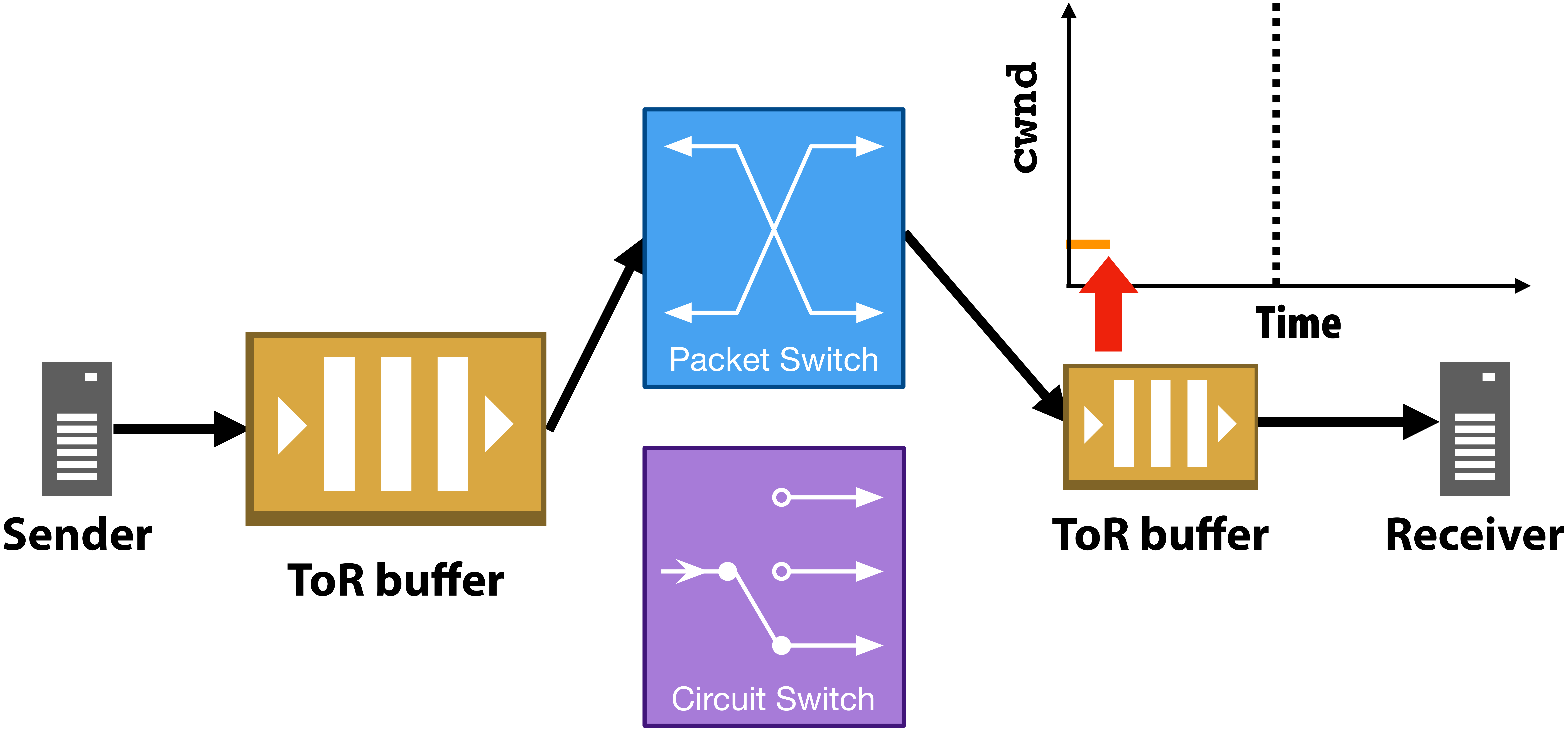
Resize ToR buffers before circuit begins



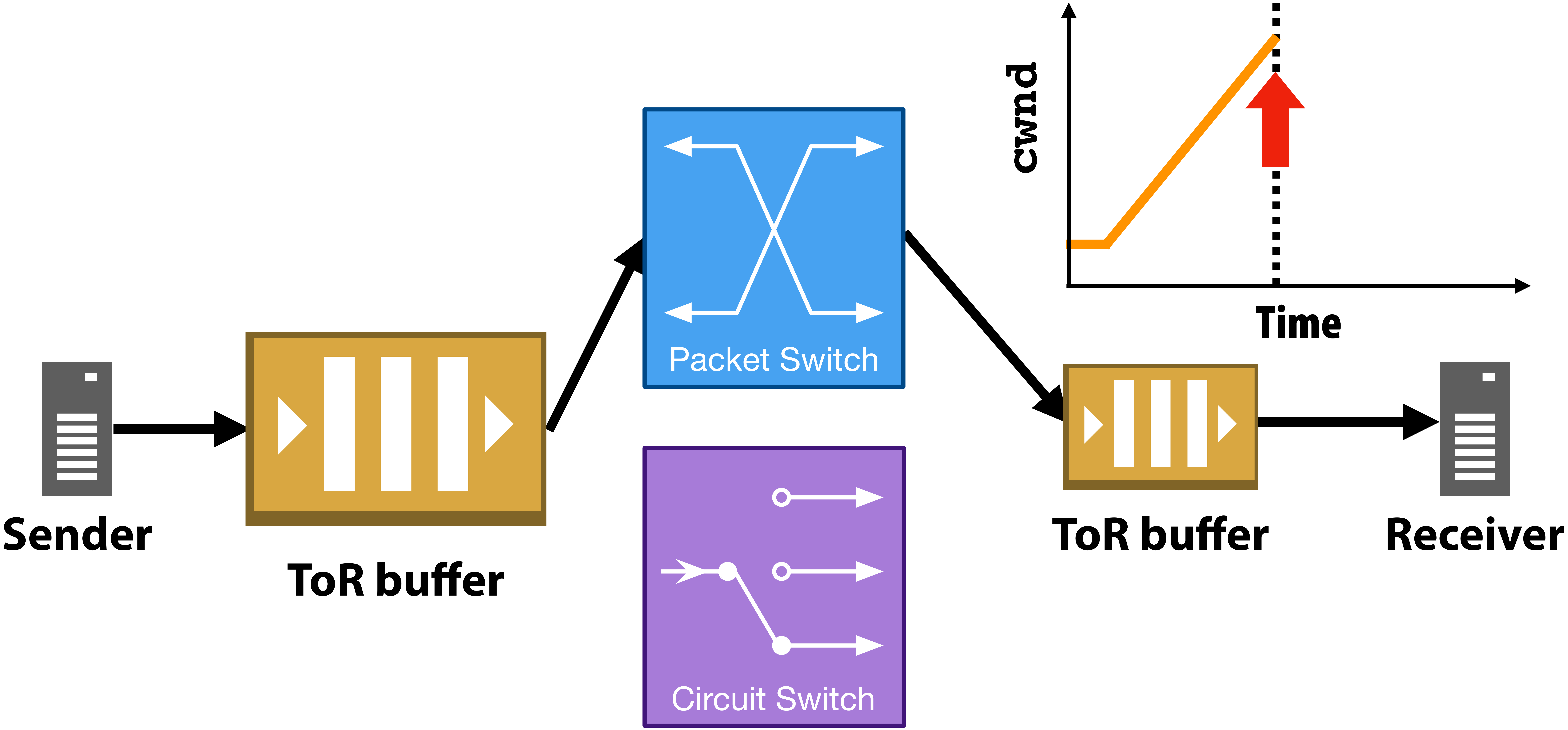
Resize ToR buffers before circuit begins



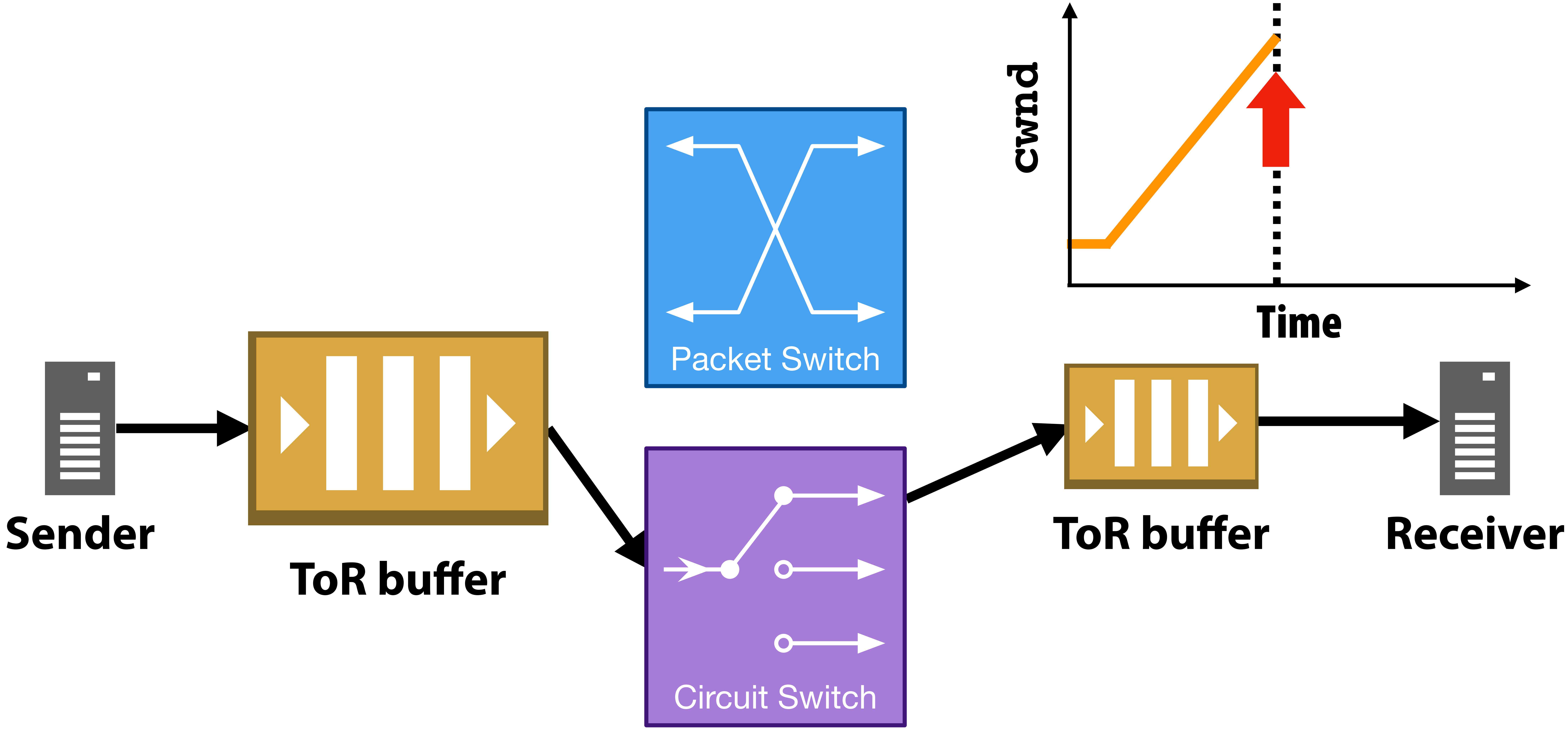
Resize ToR buffers before circuit begins



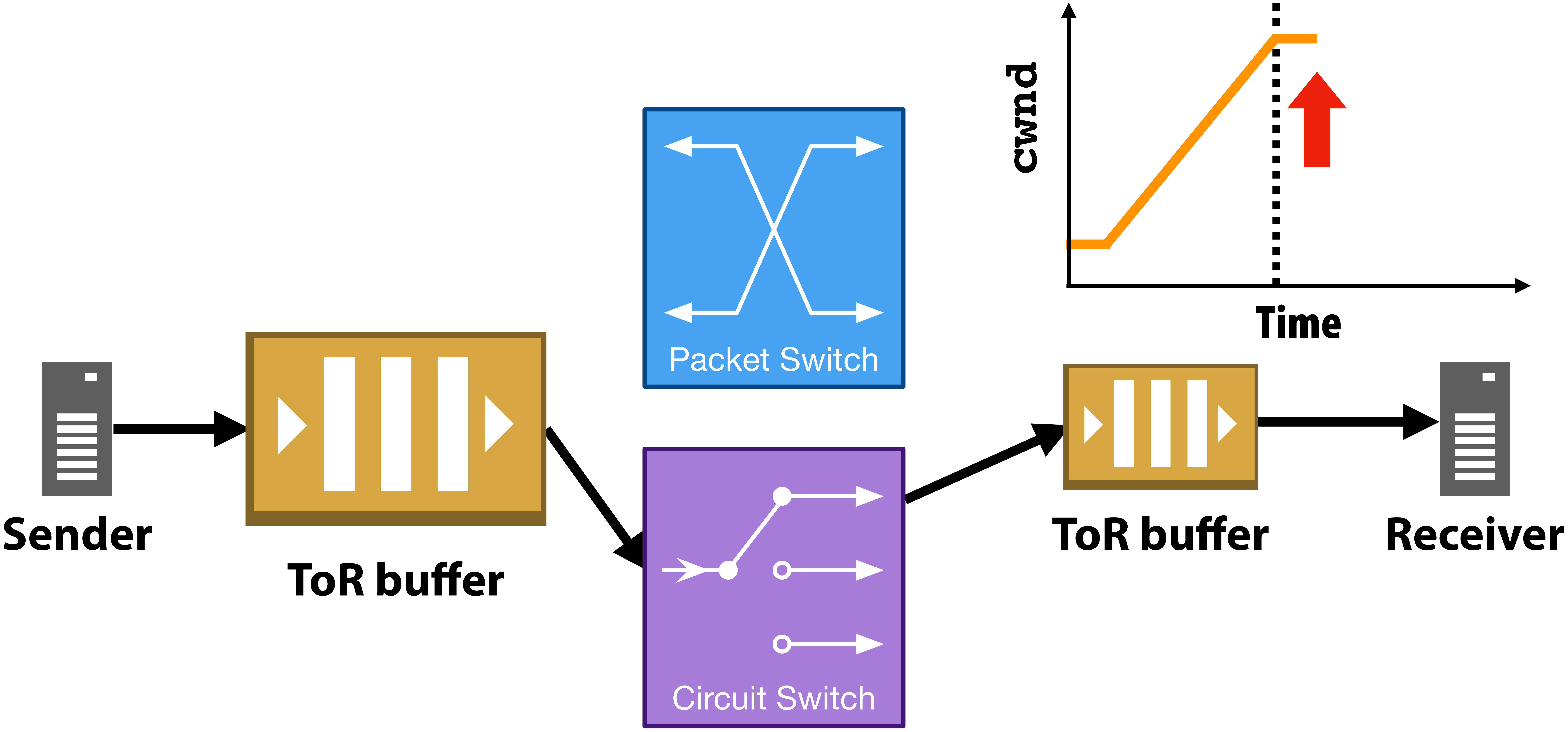
Resize ToR buffers before circuit begins



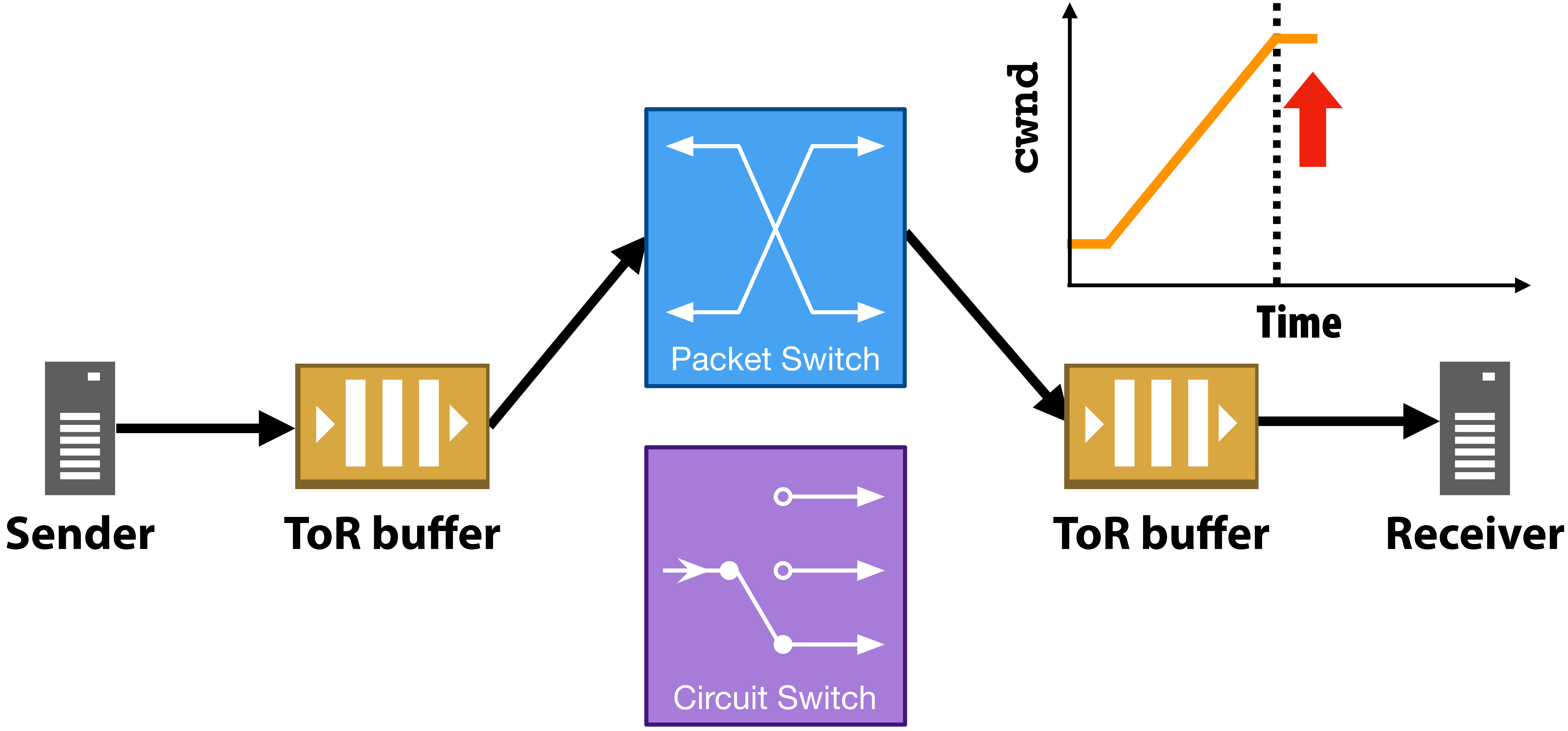
Resize ToR buffers before circuit begins



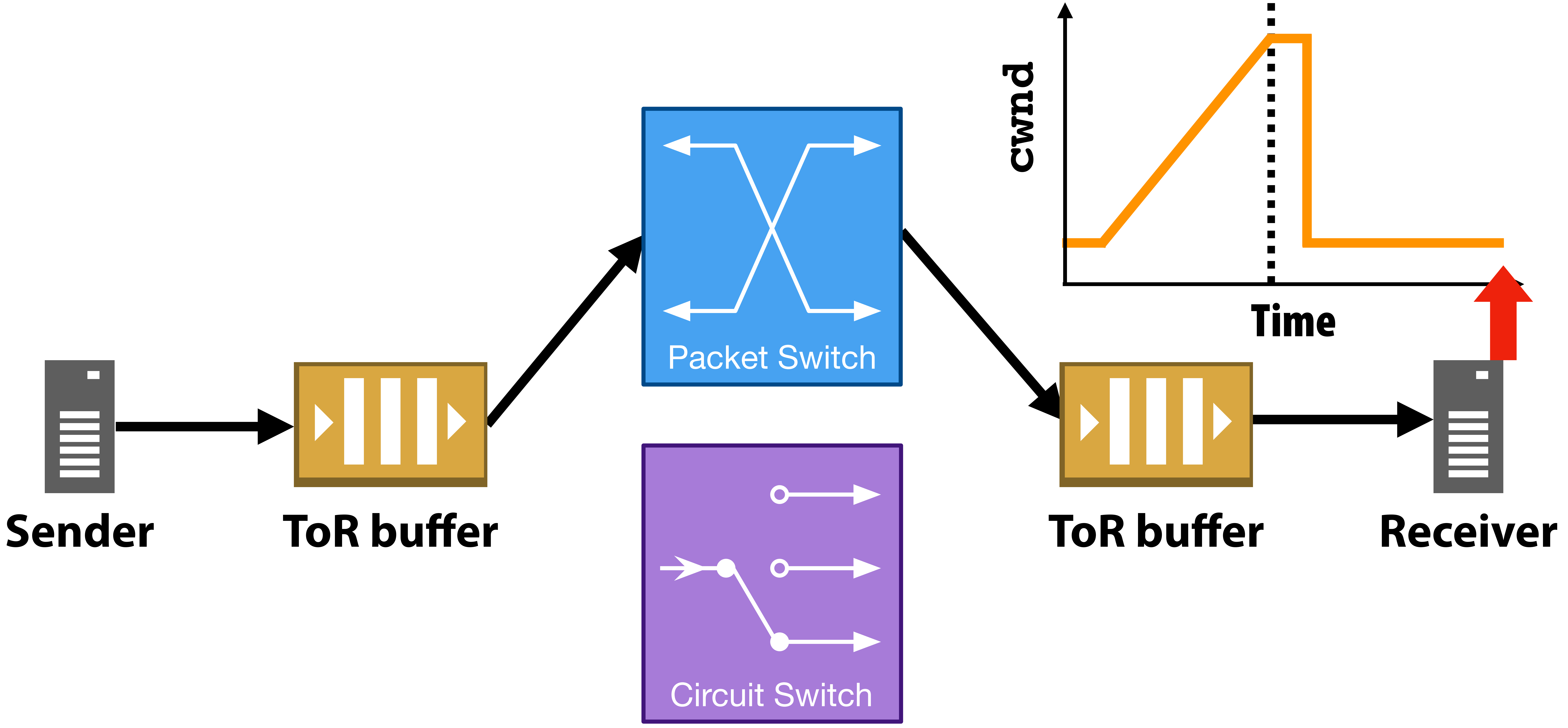
Resize ToR buffers before circuit begins



Resize ToR buffers before circuit begins



Resize ToR buffers before circuit begins



Configuring dynamic buffer resizing

How long in advance should ToR buffers resize (τ)?

- Long enough for TCP to grow `cwnd` to the circuit BDP

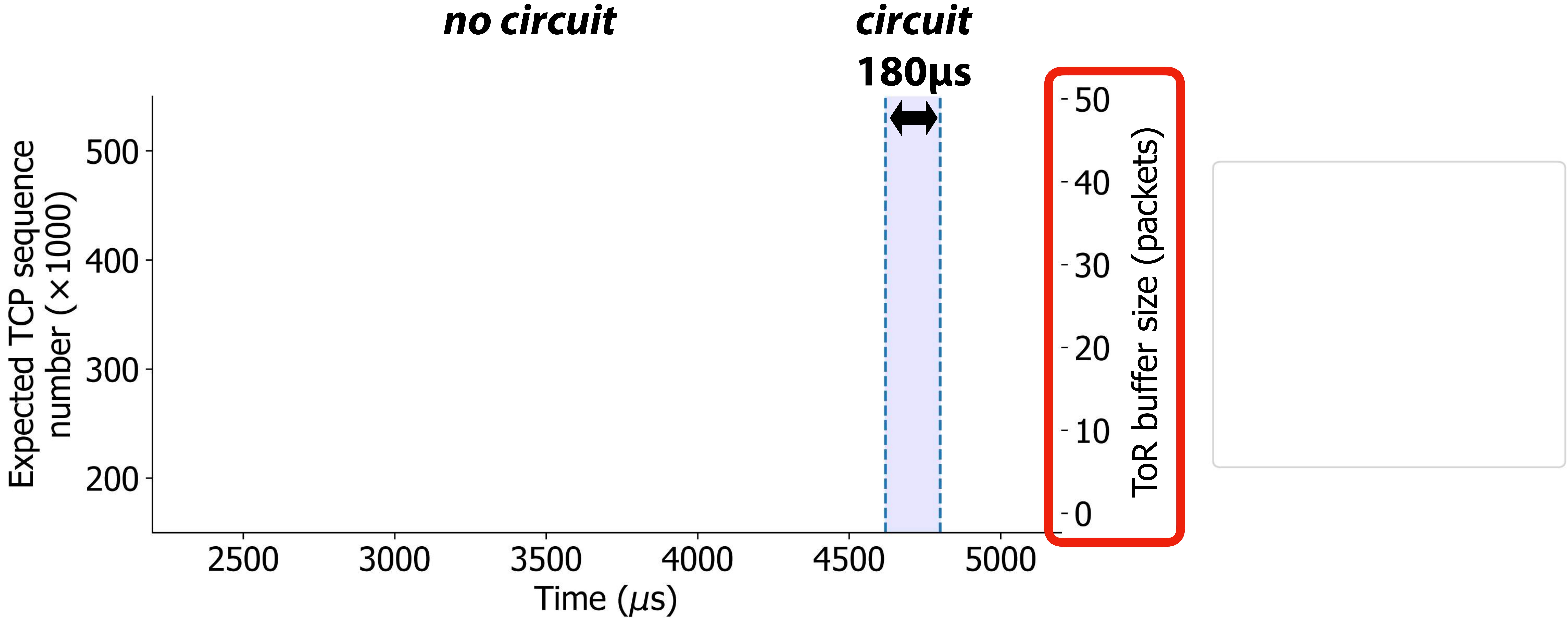
How large should ToR buffers grow to?

- circuit BDP = 80 Gb/s × 40 μs = 45 9000-byte packets

For our configuration, the ToR buffers must hold ~40 packets to achieve 90% utilization, which requires 1800 μs of prebuffering

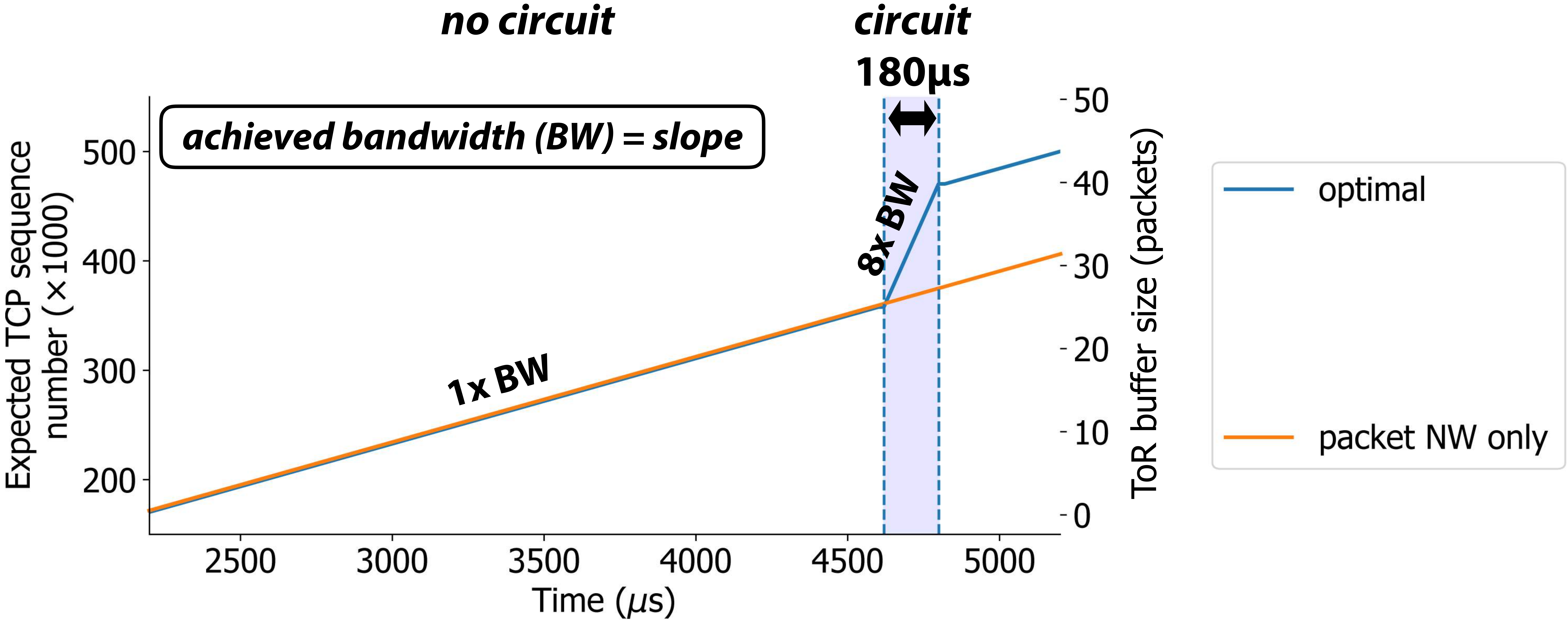
We resize ToR buffers between sizes of 16 and 50 packets

How long in advance to resize, τ?



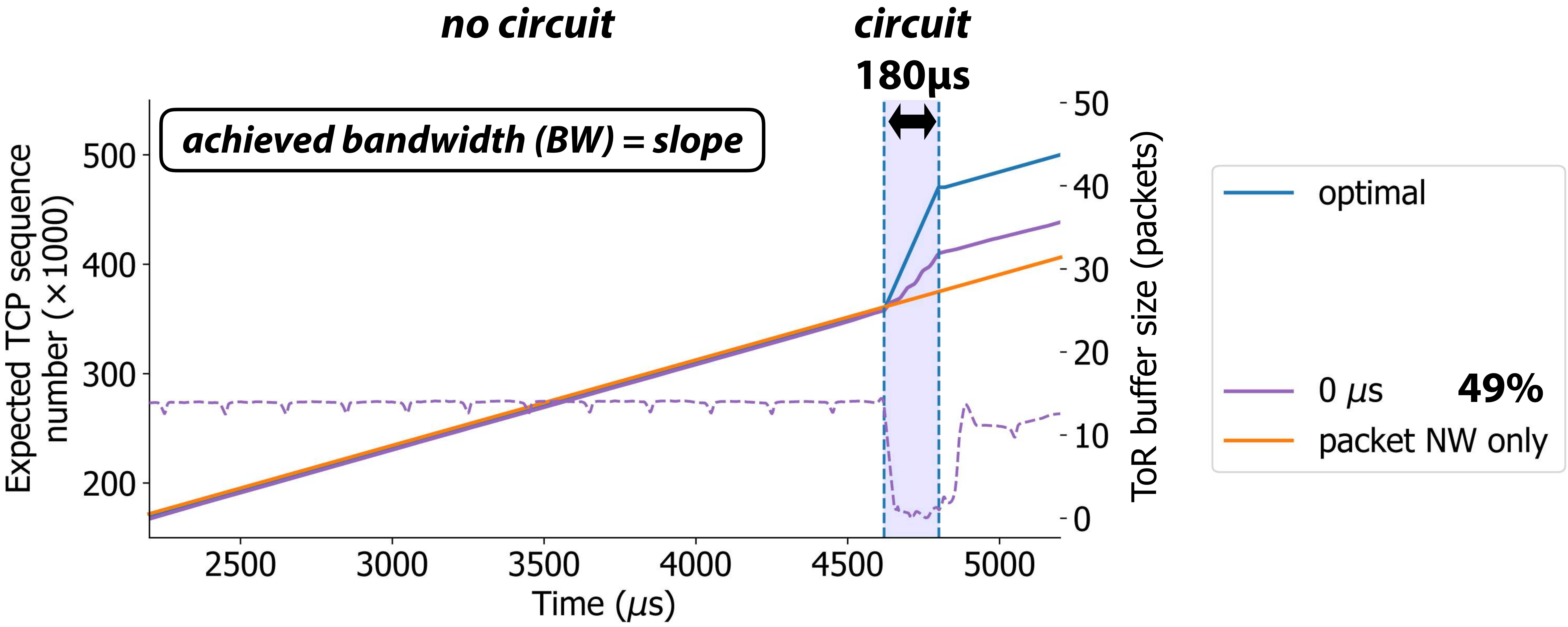
16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s;
small buffers: 16 packets; large buffers: 50 packets

How long in advance to resize, τ?



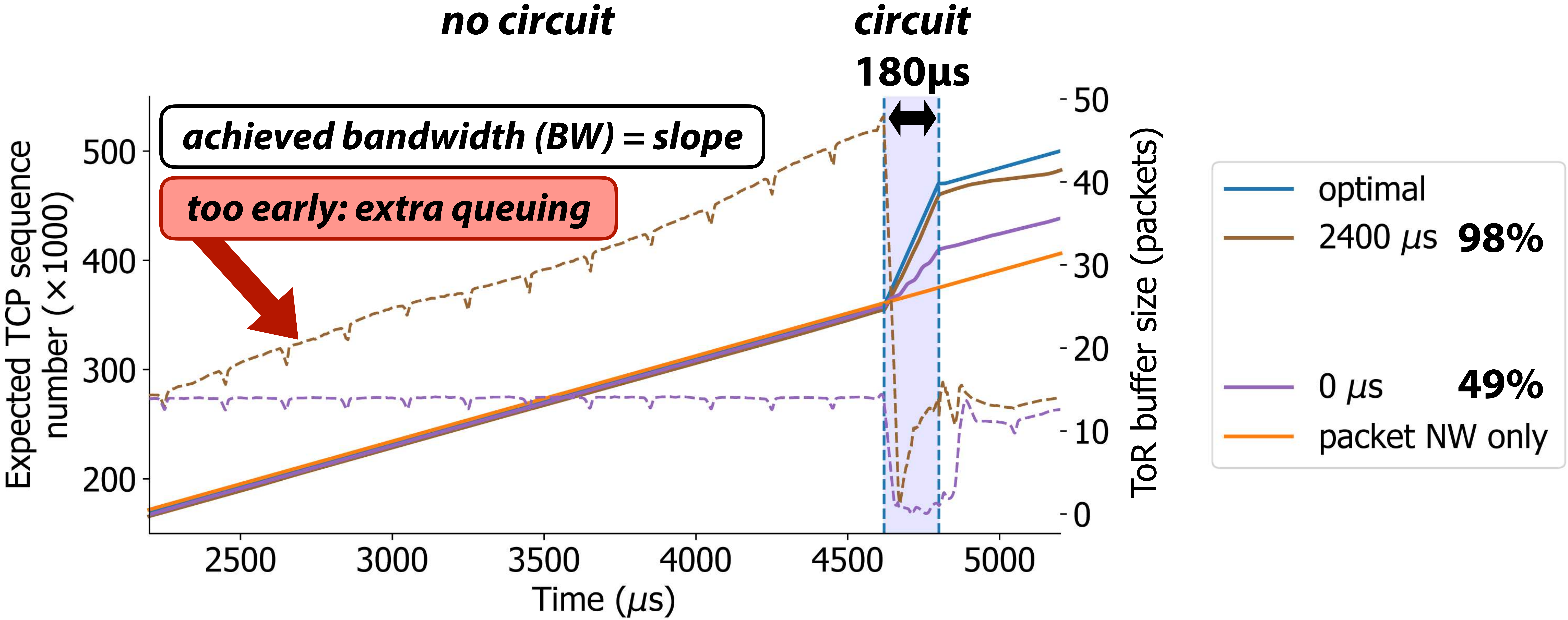
16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s; small buffers: 16 packets; large buffers: 50 packets

How long in advance to resize, τ?



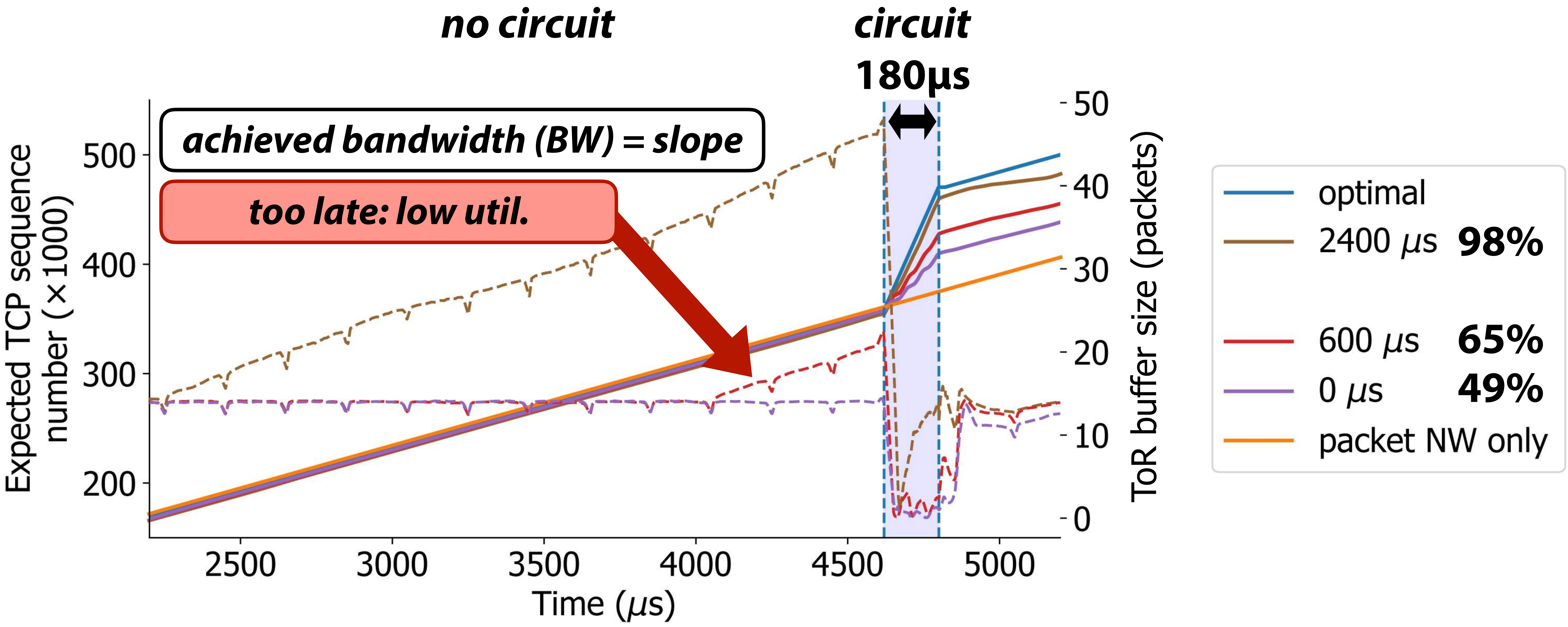
16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s; small buffers: 16 packets; large buffers: 50 packets

How long in advance to resize, $\boldsymbol{\tau}$?



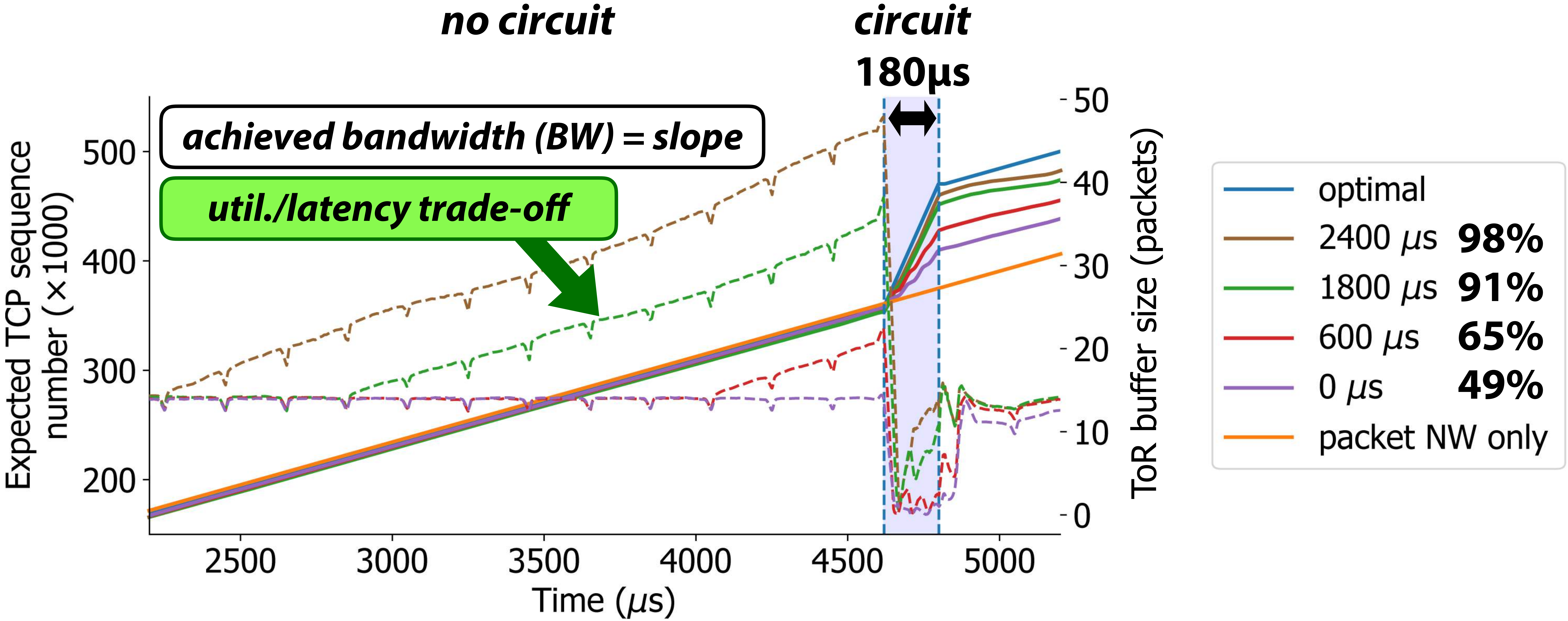
16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s;
small buffers: 16 packets; large buffers: 50 packets

How long in advance to resize, $\boldsymbol{\tau}$?



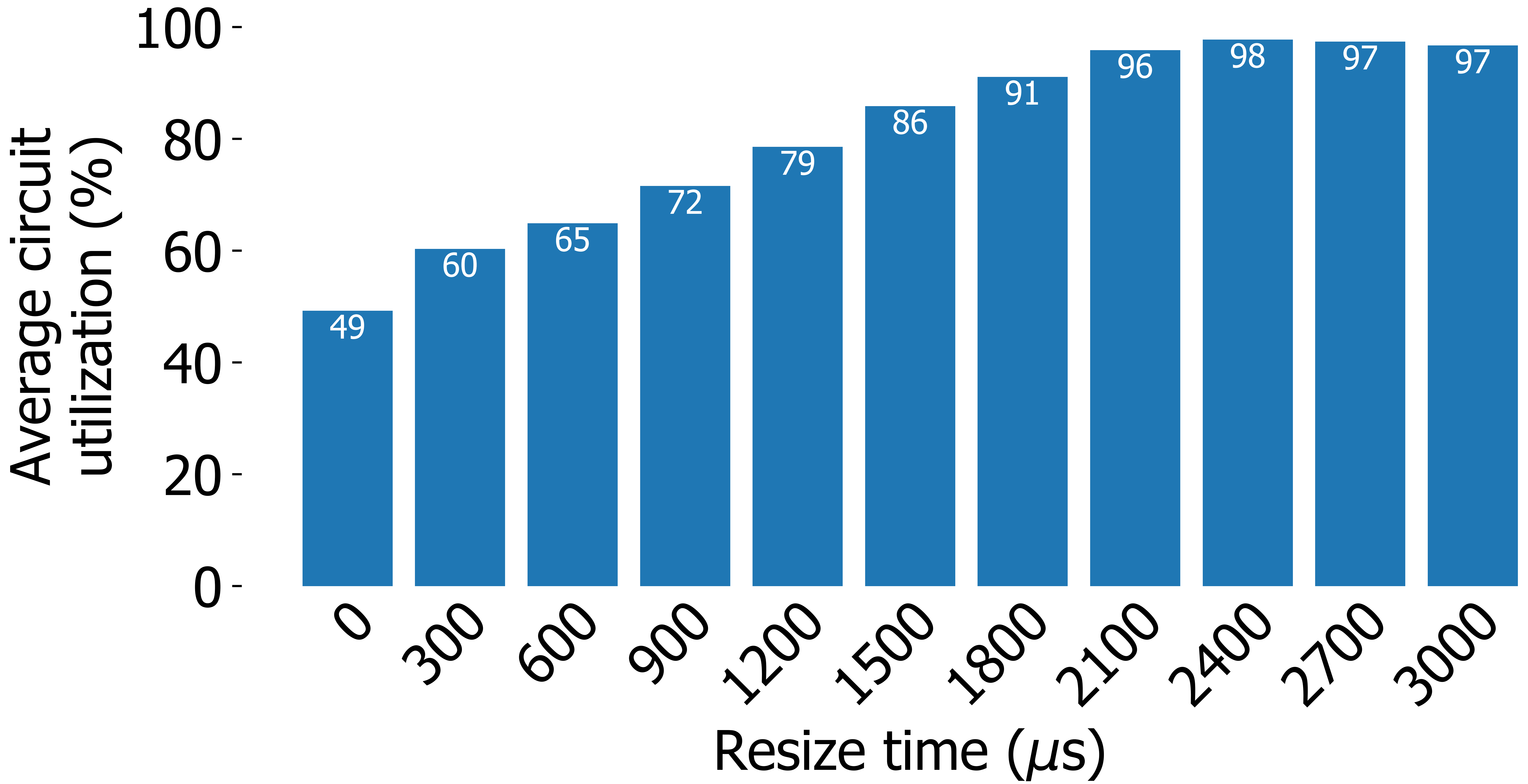
16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s;
small buffers: 16 packets; large buffers: 50 packets

How long in advance to resize, $\boldsymbol{\tau}$?



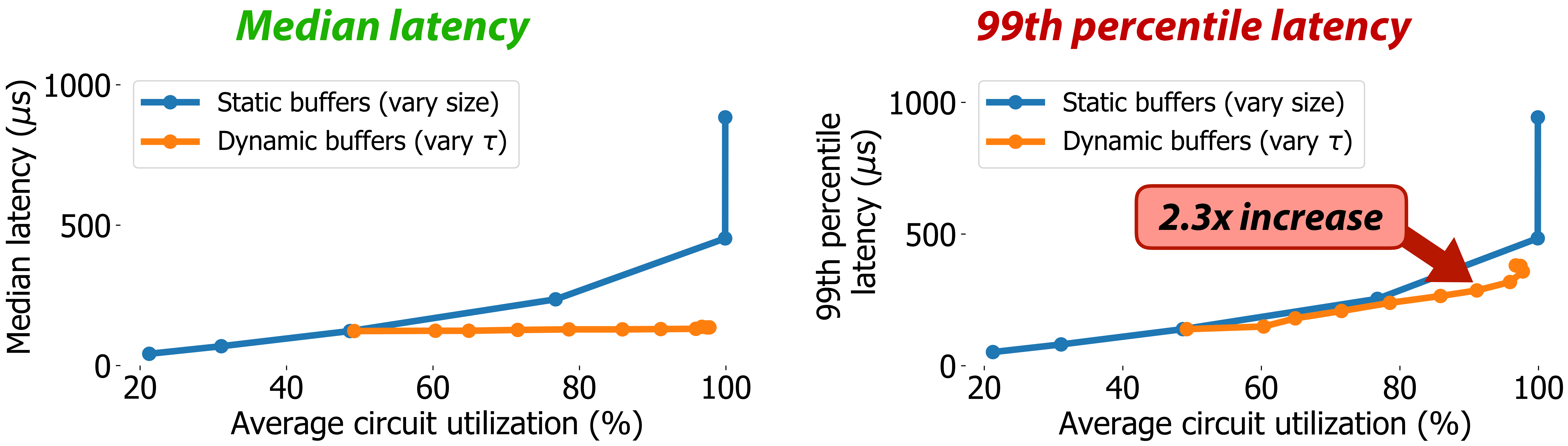
16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s; small buffers: 16 packets; large buffers: 50 packets

1800µs of prebuffering yields 91% util.



16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s; small buffers: 16 packets; large buffers: 50 packets

Latency degradation during ramp-up



We cannot use large queues for so long.
Can we get the same high utilization with shorter prebuffering?

16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s; small buffers: 16 packets; large buffers: 50 packets

This talk: Our 2-part solution

In-network: Use information about upcoming circuits to transparently “trick” TCP into ramping up more aggressively

- High utilization, at the cost of tail latency

At endhosts: New TCP variant, ***reTCP***, that explicitly reacts to circuit state changes

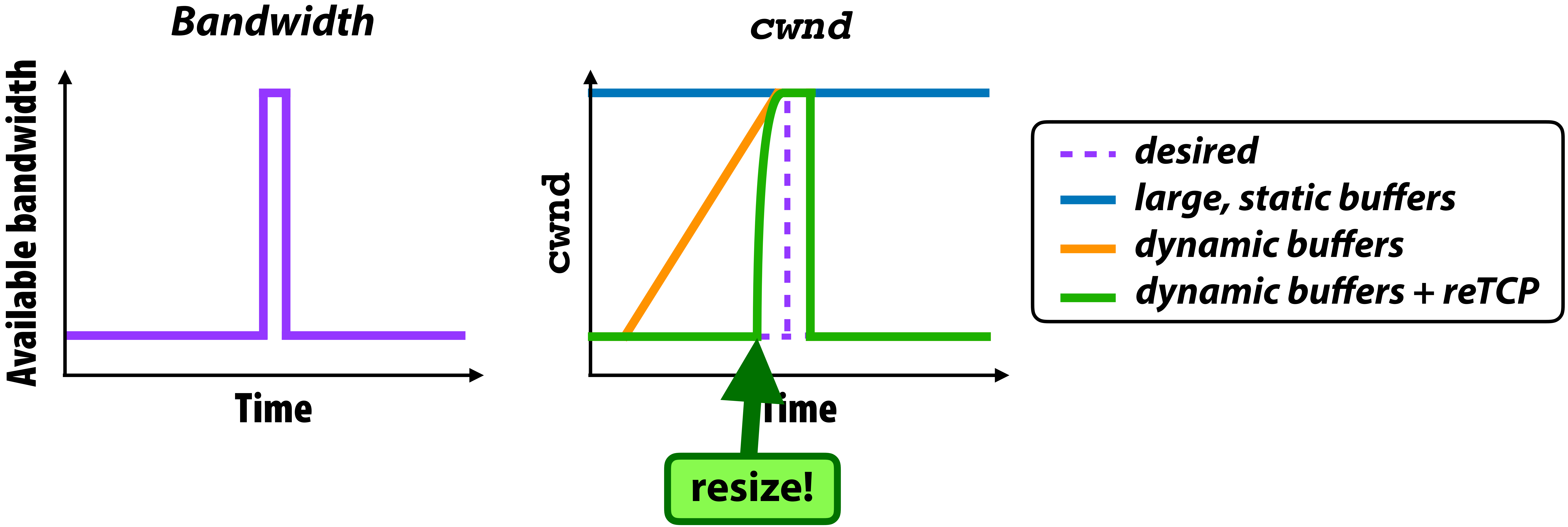
- Mitigates tail latency penalty

The two techniques can be deployed separately, but work best together

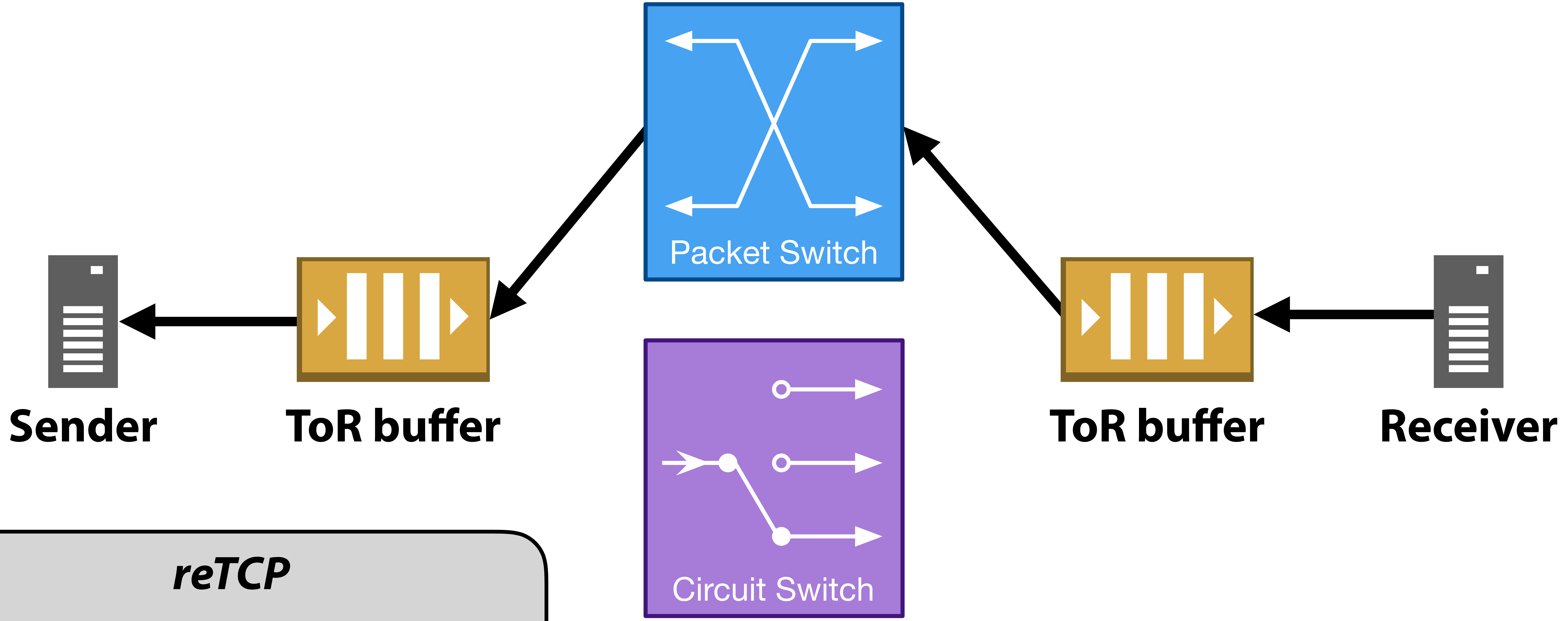
reTCP: Rapidly grow **`cwnd`** before a circuit

1) Communicate circuit state to sender TCP

2) Sender TCP reacts by multiplicatively increasing/decreasing **`cwnd`**



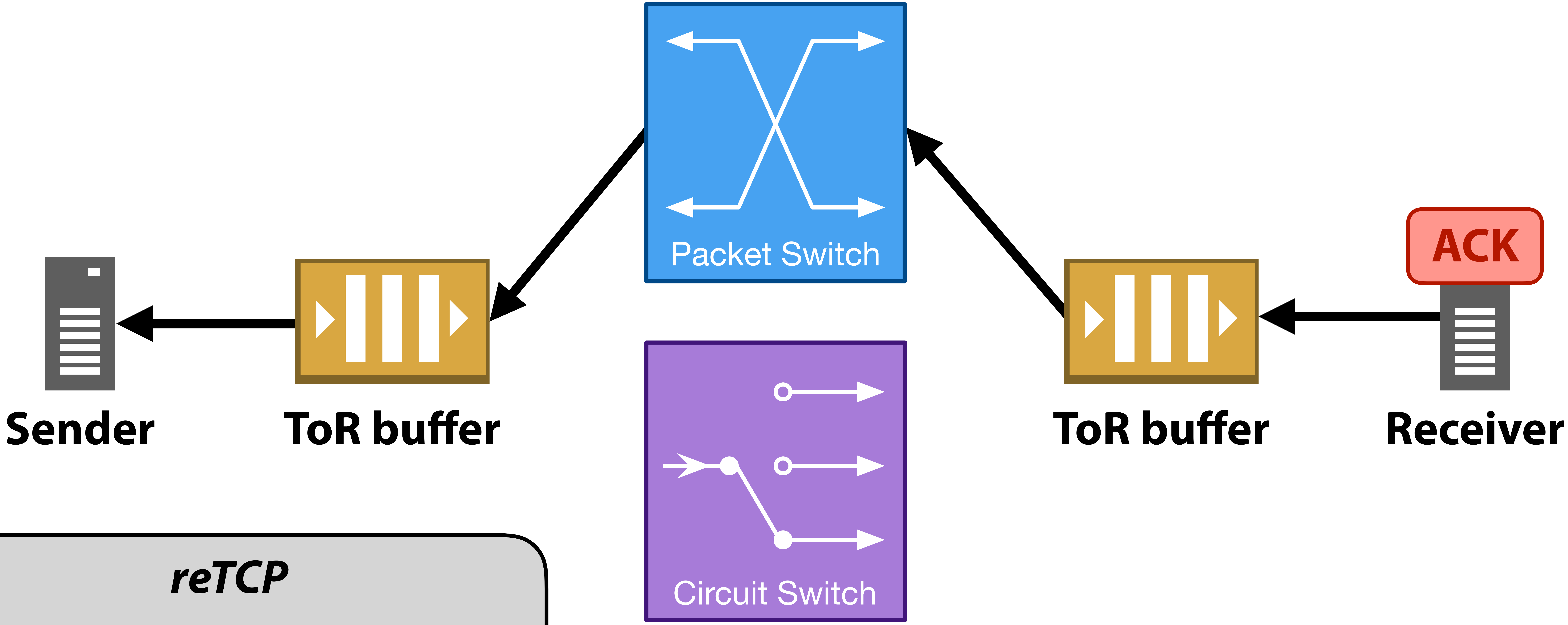
reTCP: Explicit circuit state feedback



reTCP

marks:

reTCP: Explicit circuit state feedback

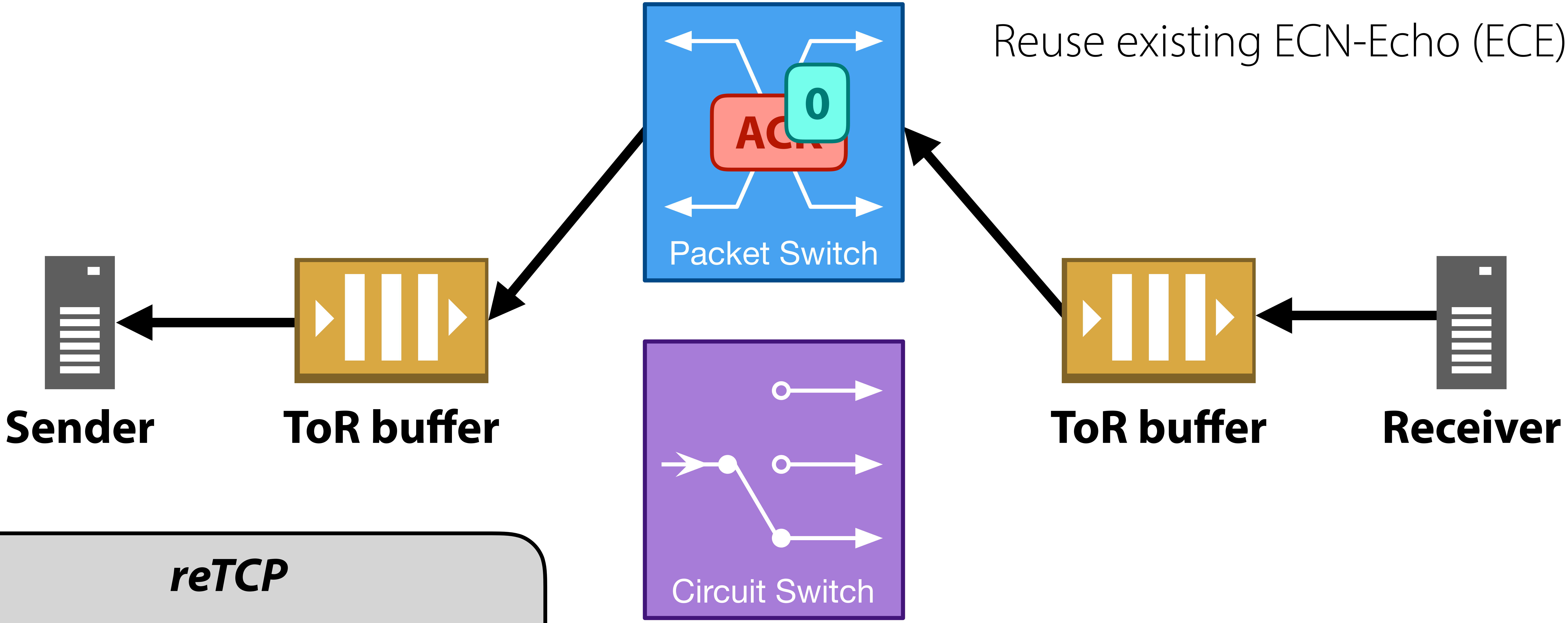


reTCP

marks:

reTCP: Explicit circuit state feedback

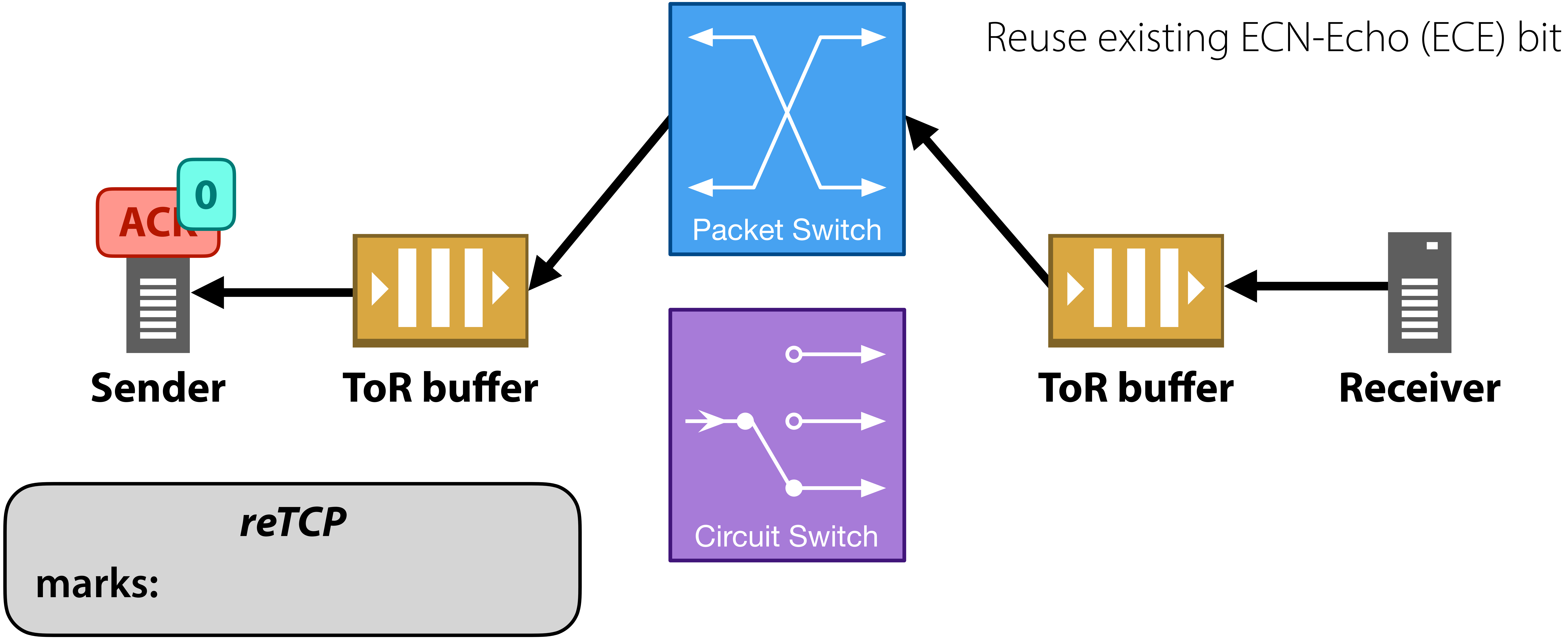
Reuse existing ECN-Echo (ECE) bit



reTCP

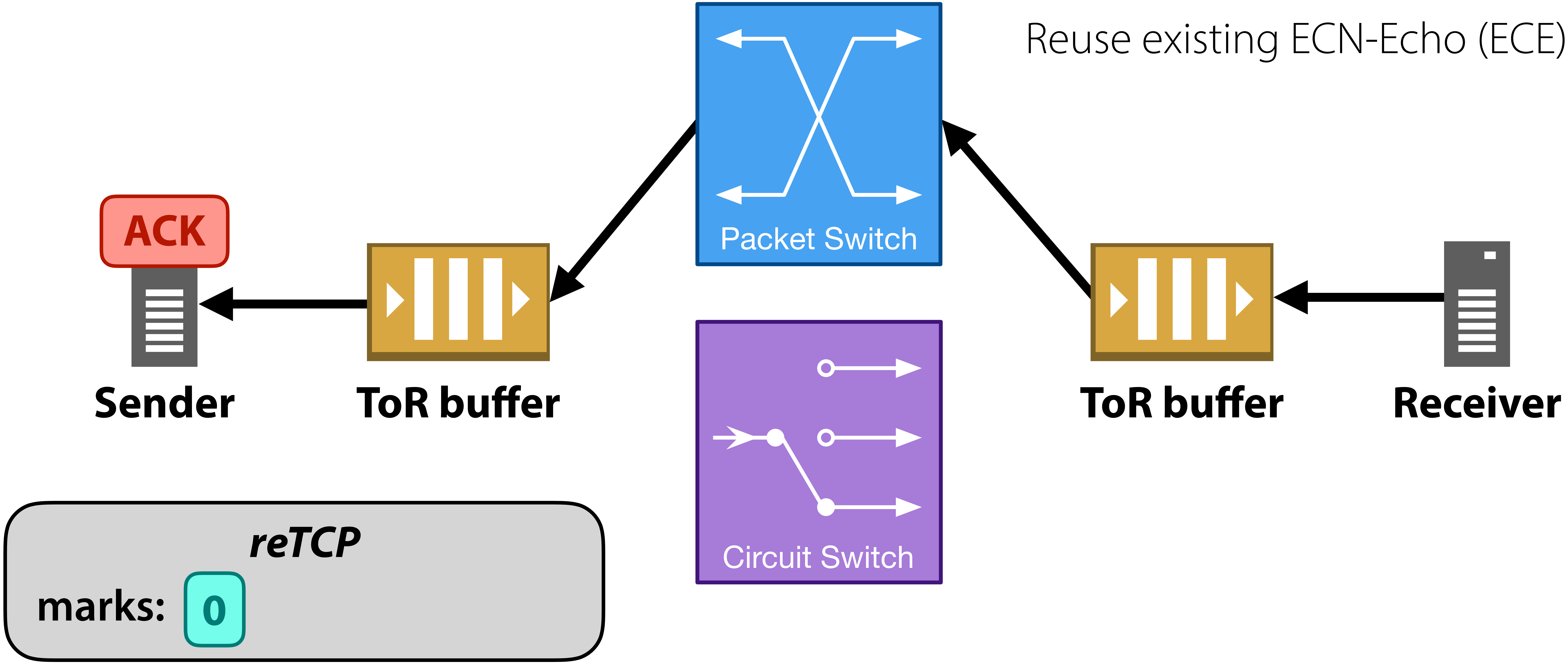
marks:

reTCP: Explicit circuit state feedback

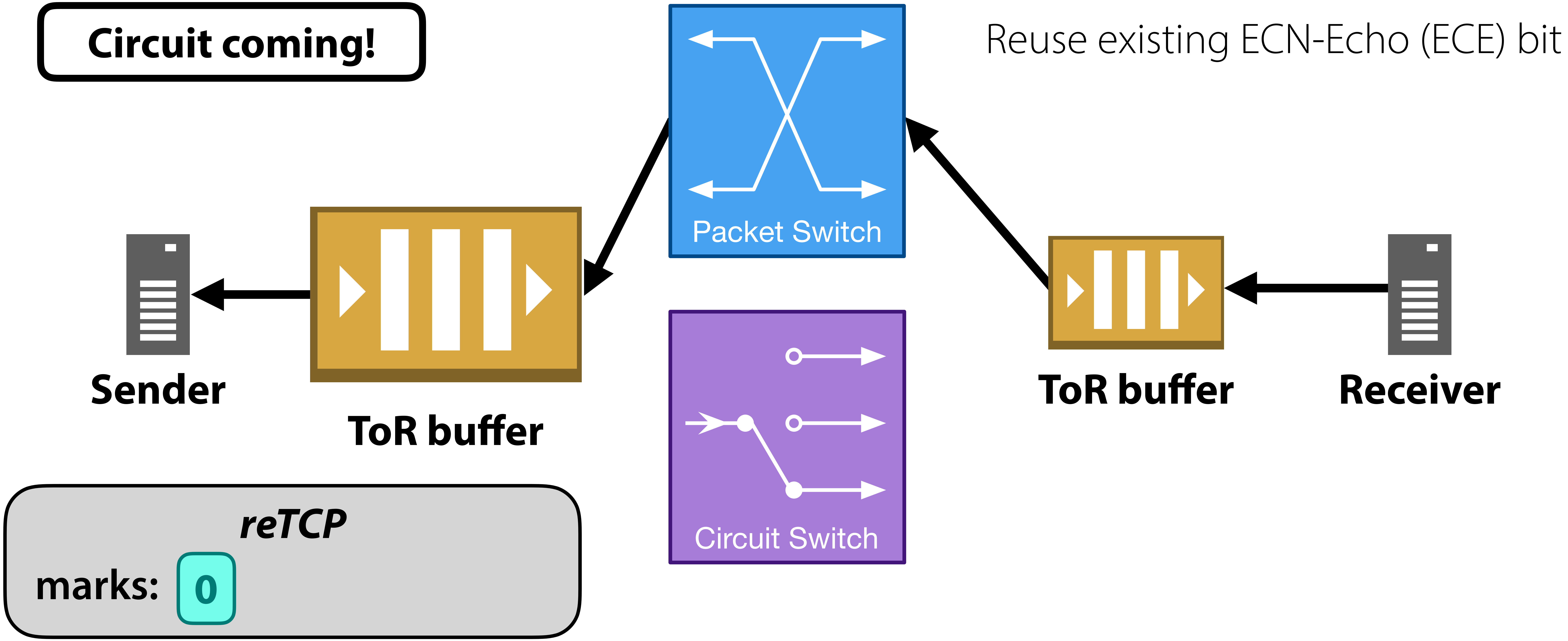


reTCP: Explicit circuit state feedback

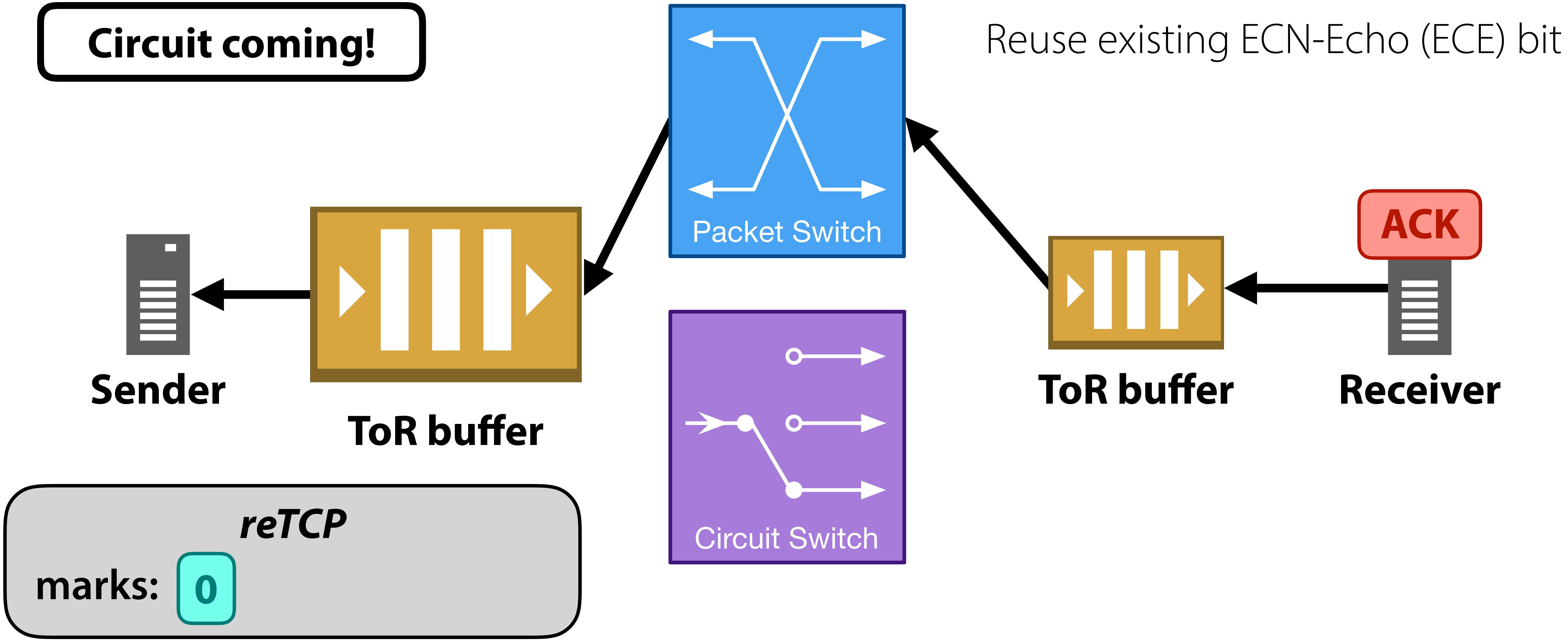
Reuse existing ECN-Echo (ECE) bit



reTCP: Explicit circuit state feedback

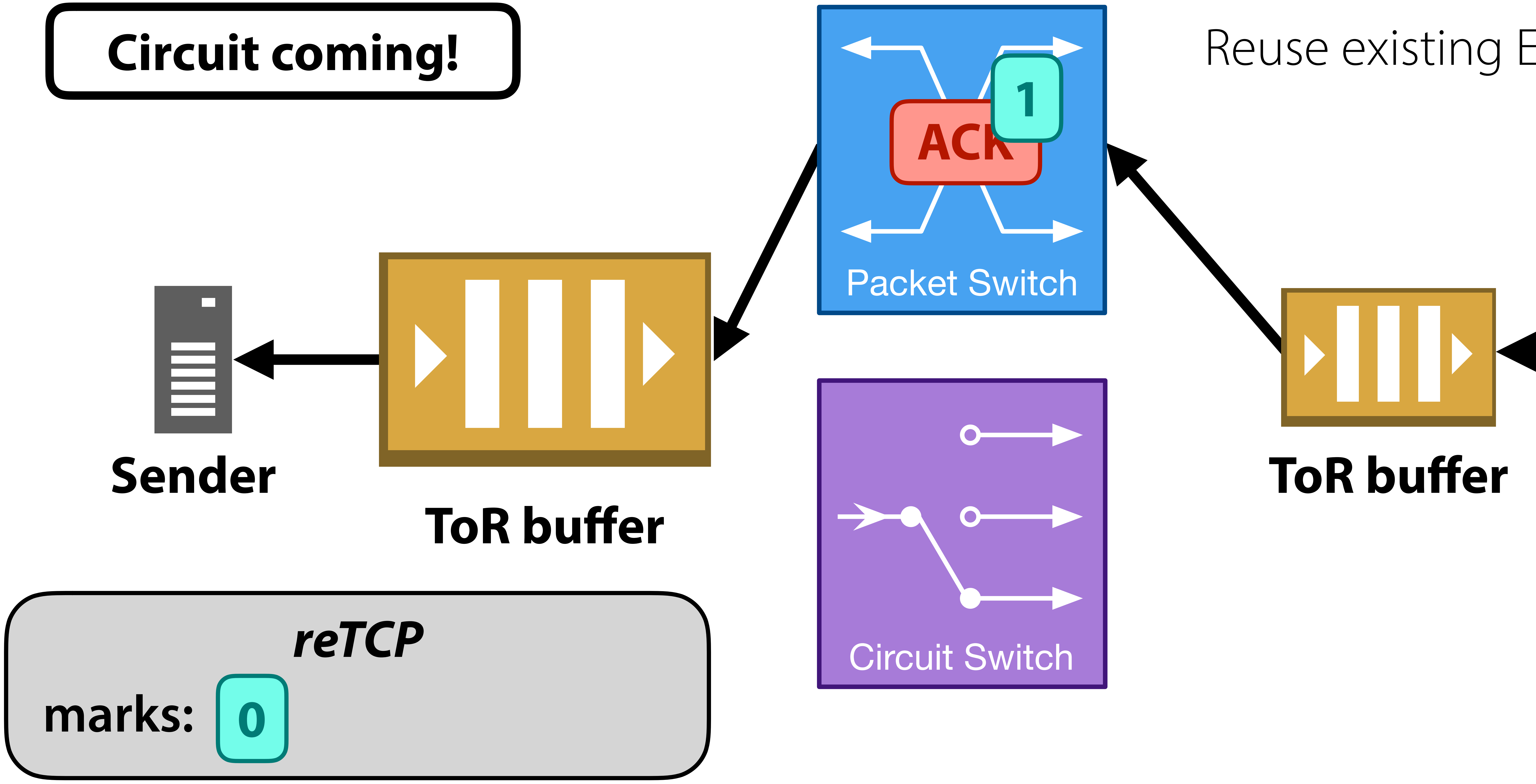


reTCP: Explicit circuit state feedback

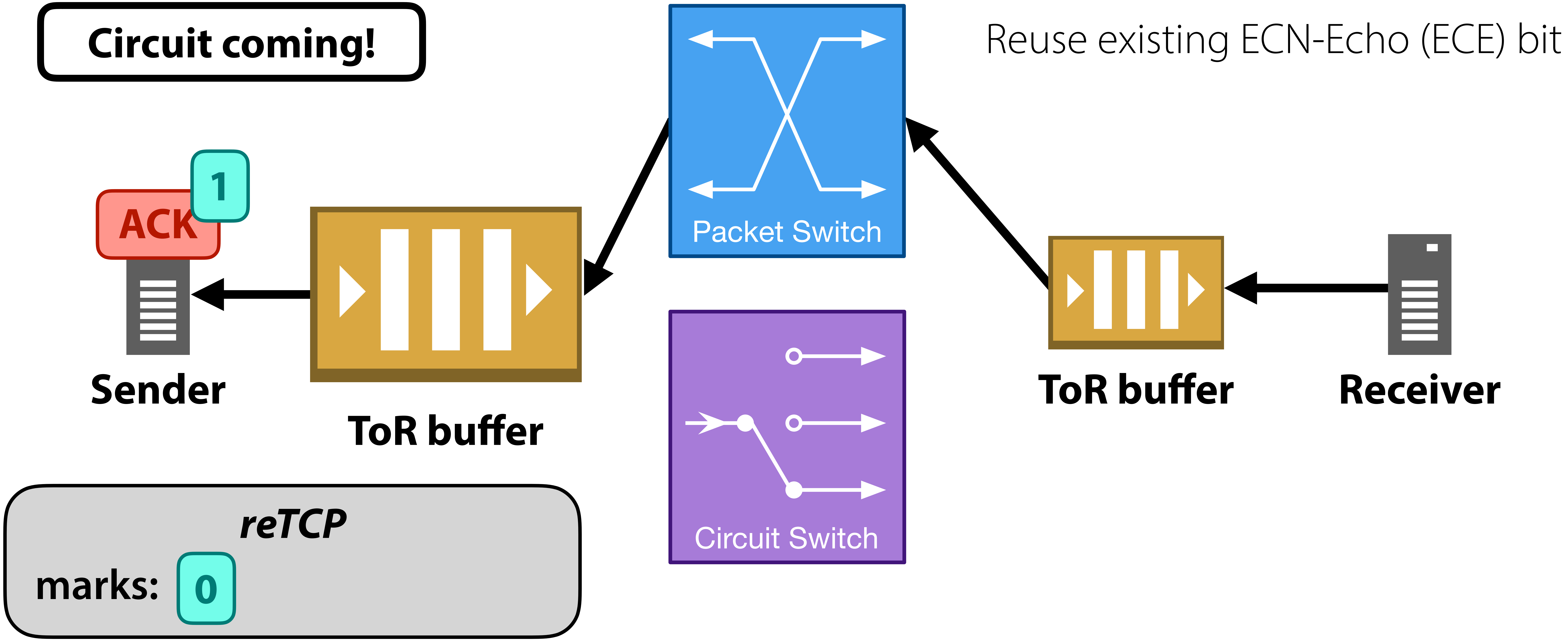


reTCP: Explicit circuit state feedback

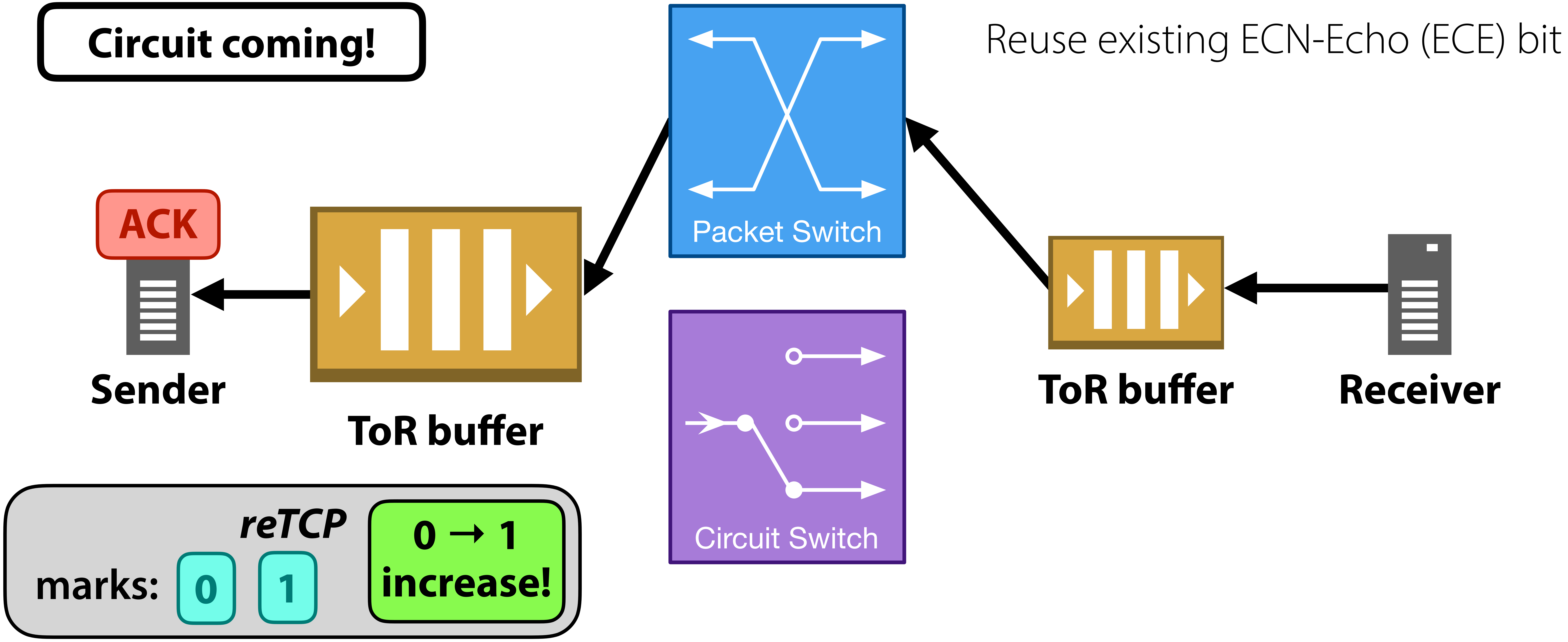
Reuse existing ECN-Echo (ECE) bit



reTCP: Explicit circuit state feedback

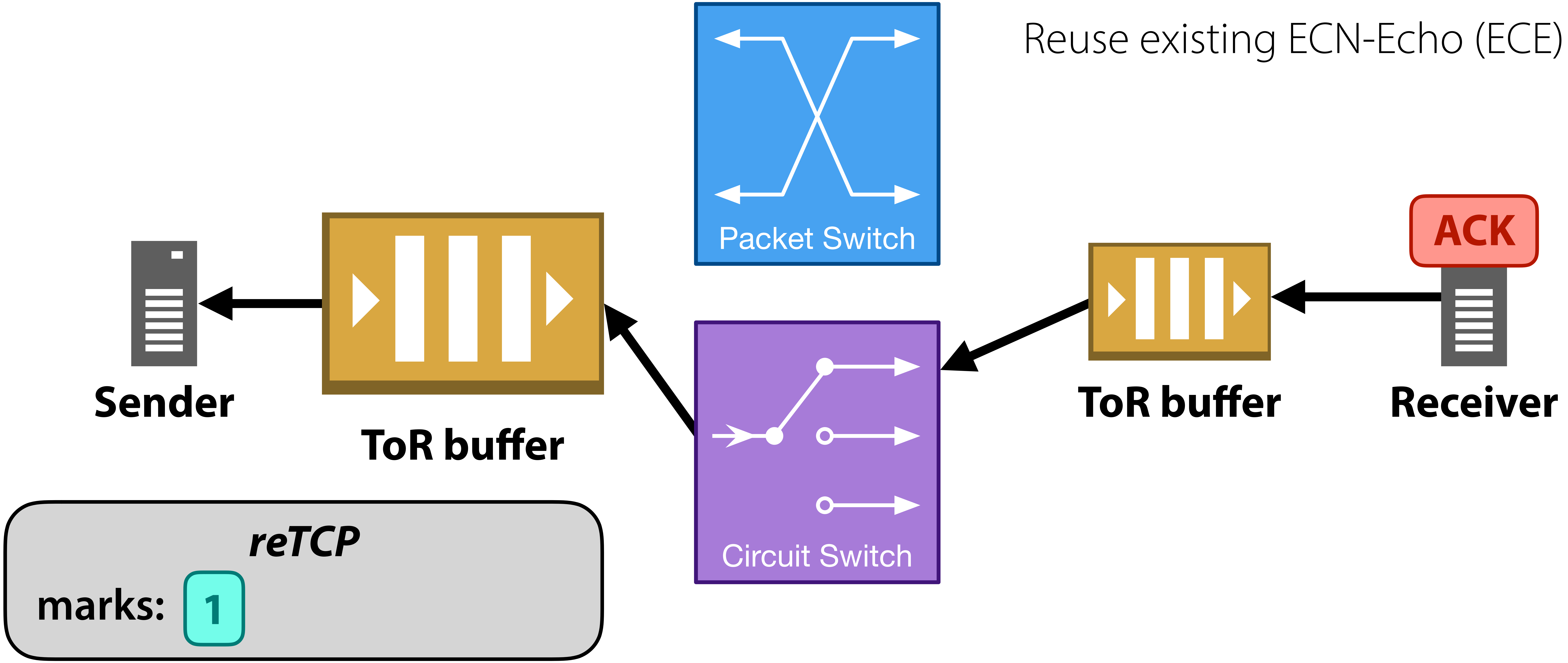


reTCP: Explicit circuit state feedback



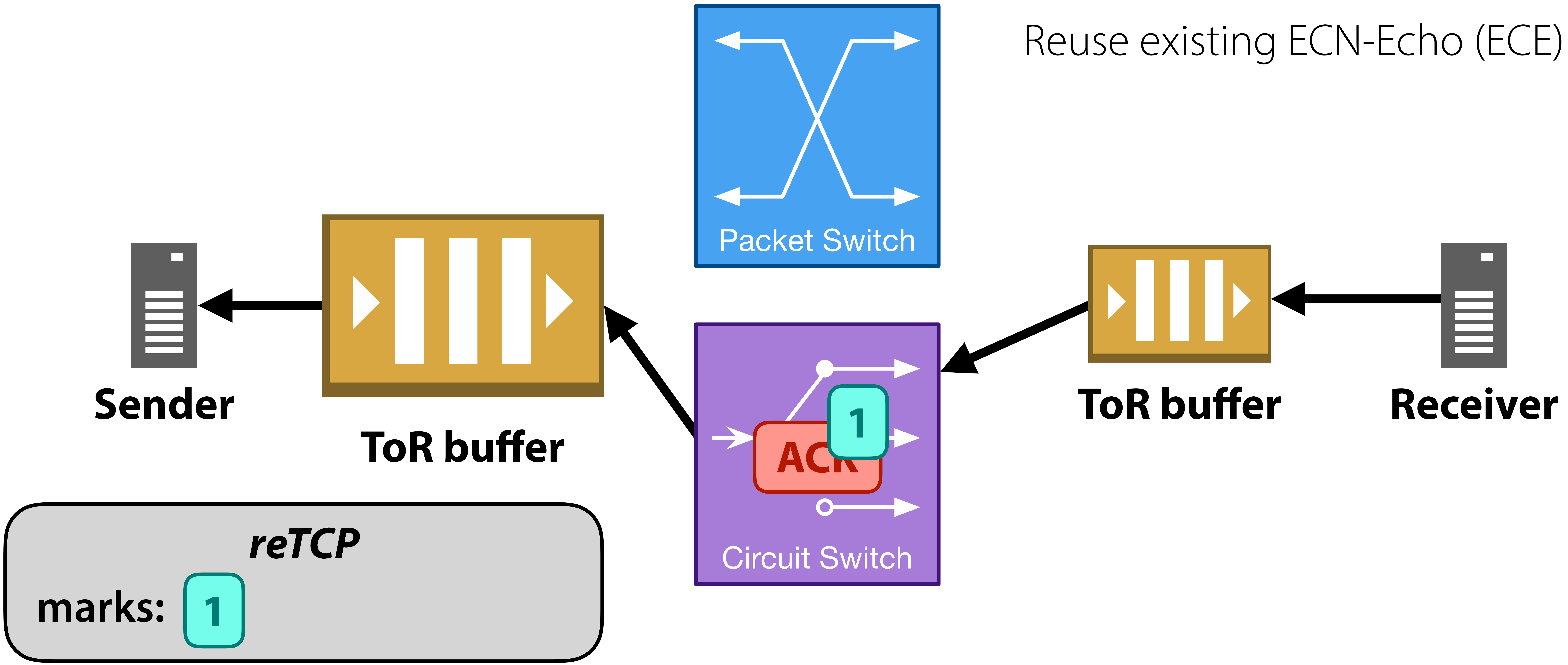
reTCP: Explicit circuit state feedback

Reuse existing ECN-Echo (ECE) bit



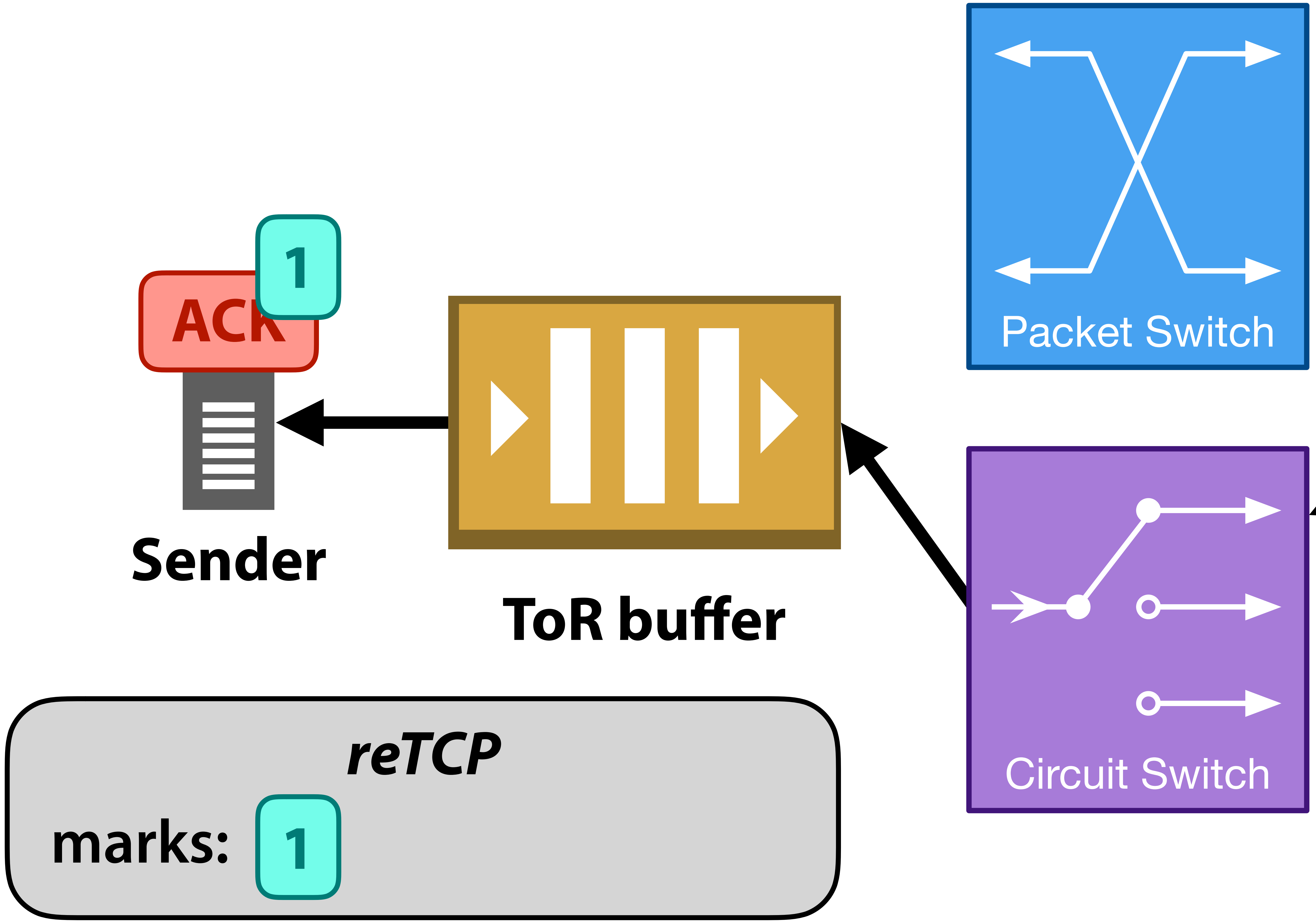
reTCP: Explicit circuit state feedback

Reuse existing ECN-Echo (ECE) bit



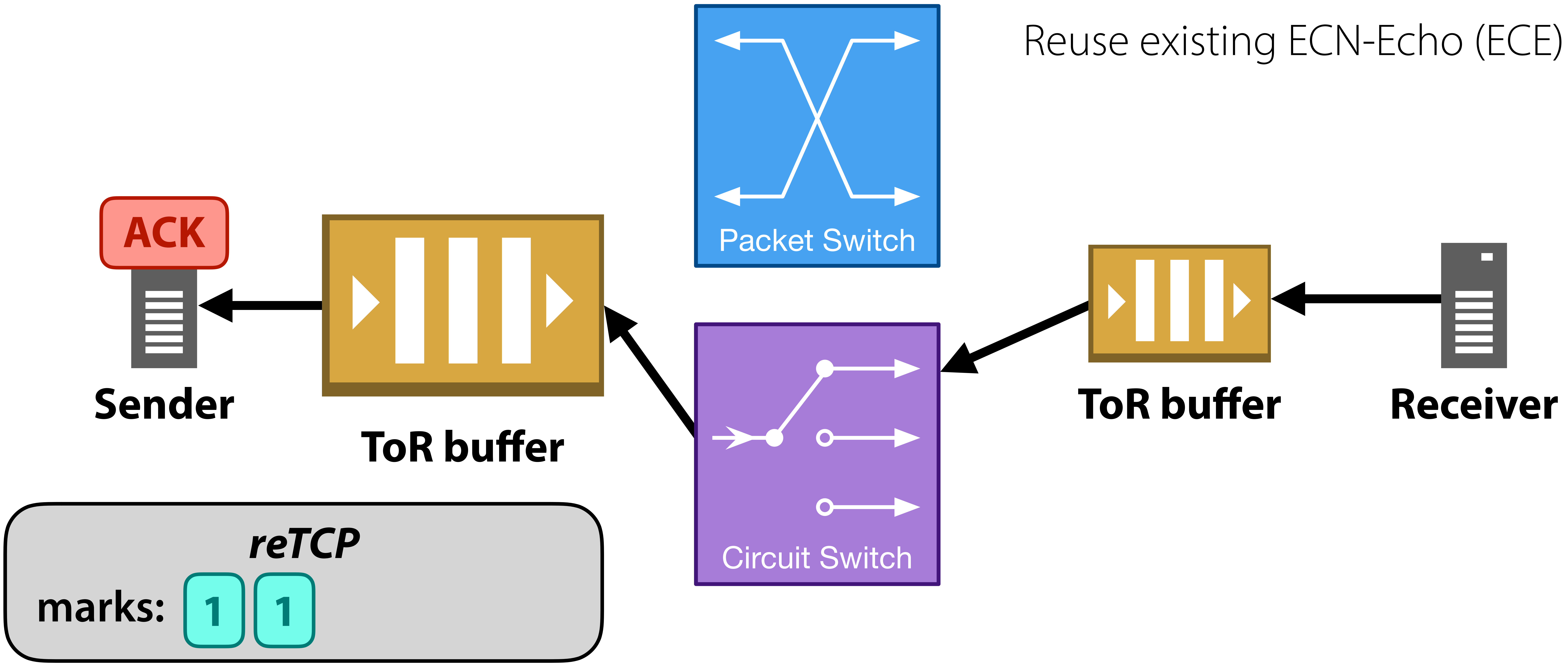
reTCP: Explicit circuit state feedback

Reuse existing ECN-Echo (ECE) bit

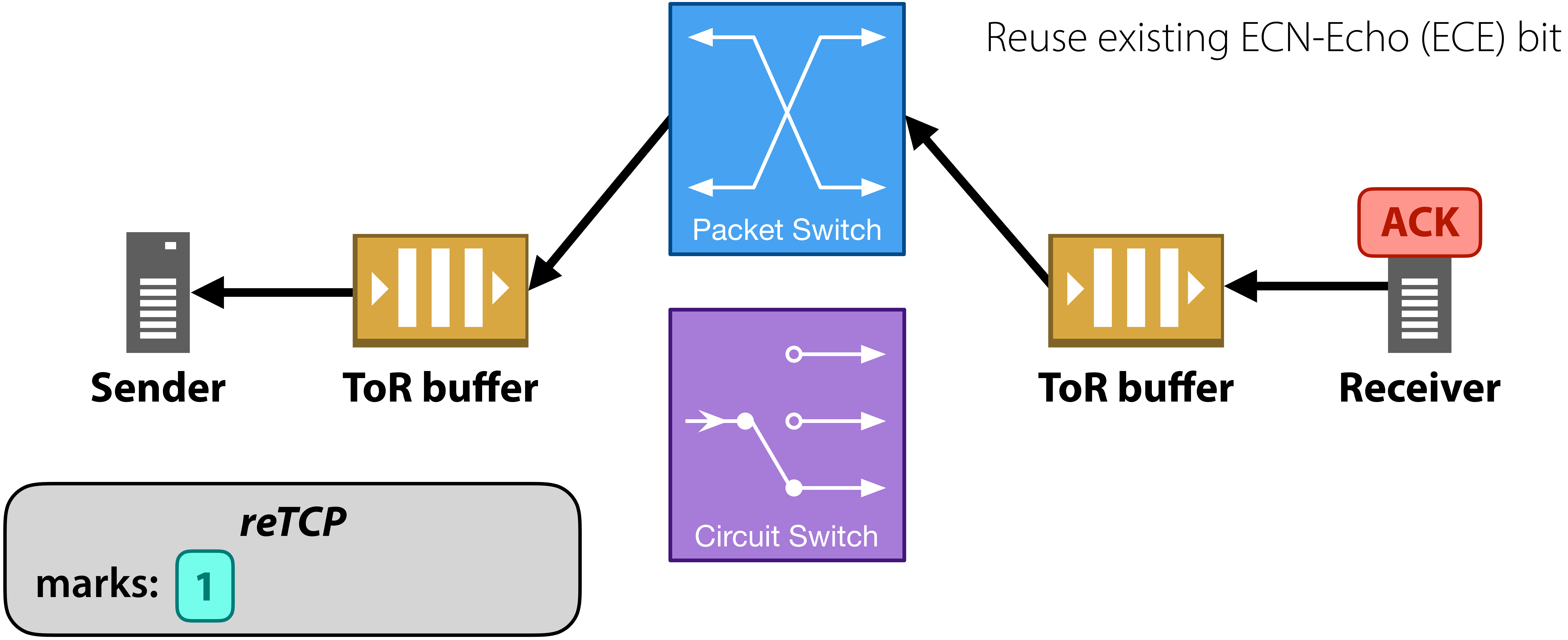


reTCP: Explicit circuit state feedback

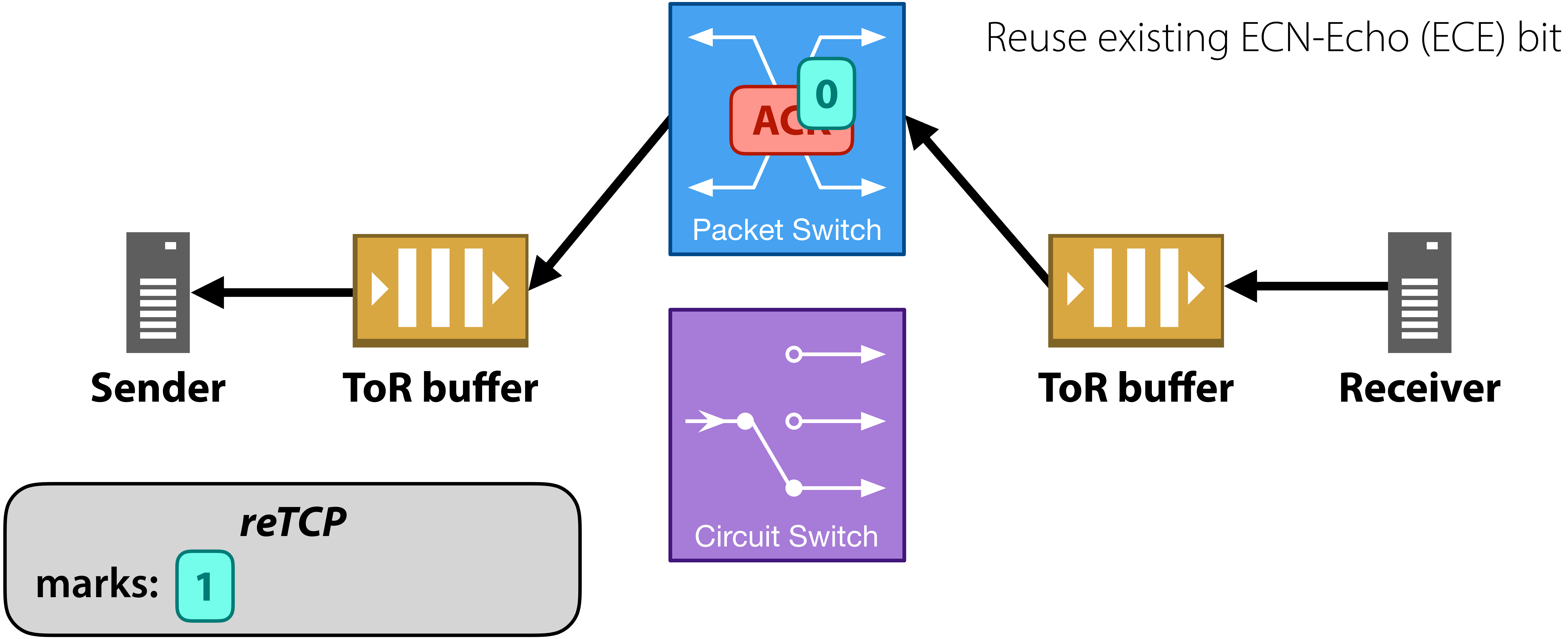
Reuse existing ECN-Echo (ECE) bit



reTCP: Explicit circuit state feedback

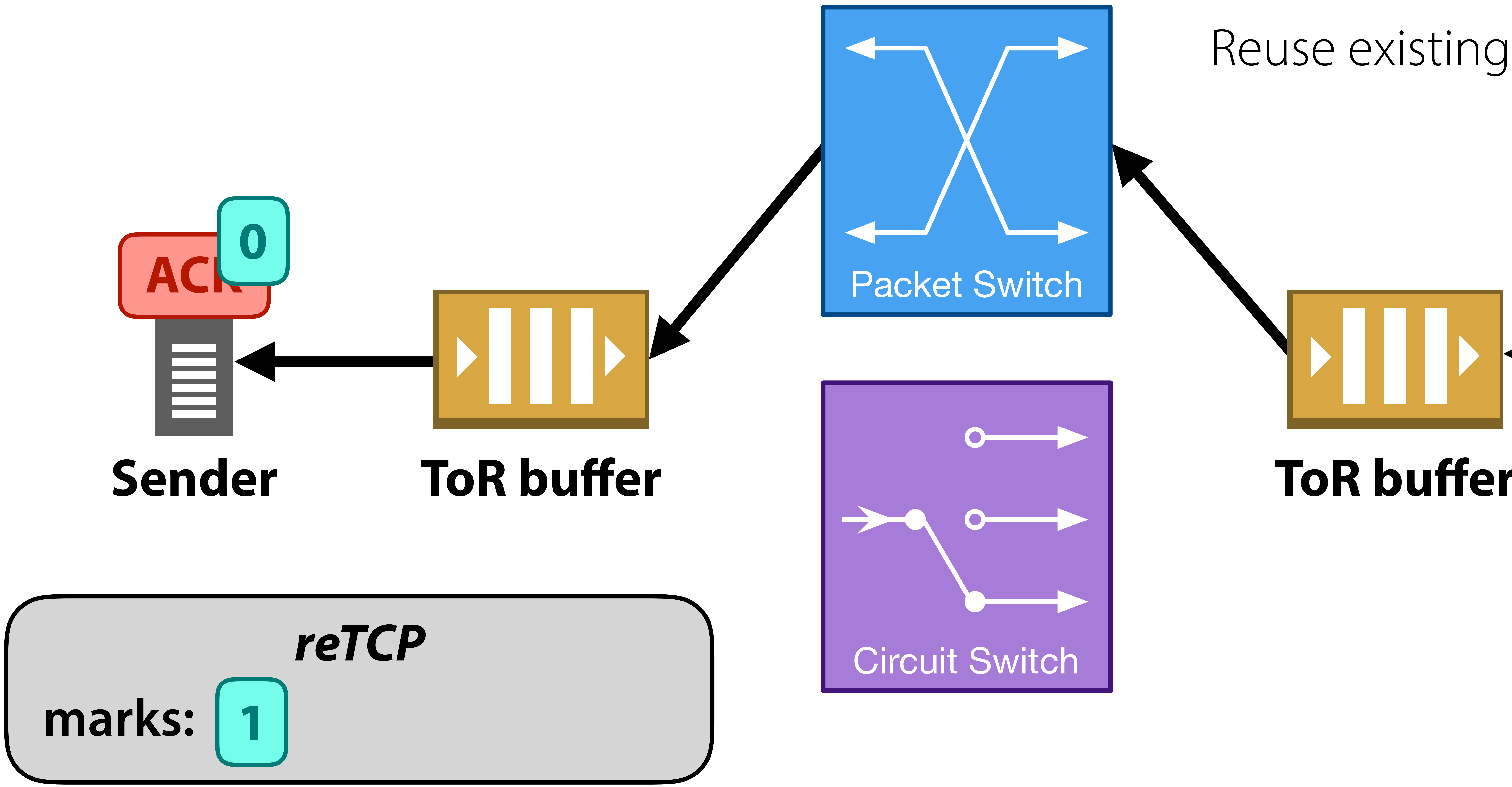


reTCP: Explicit circuit state feedback



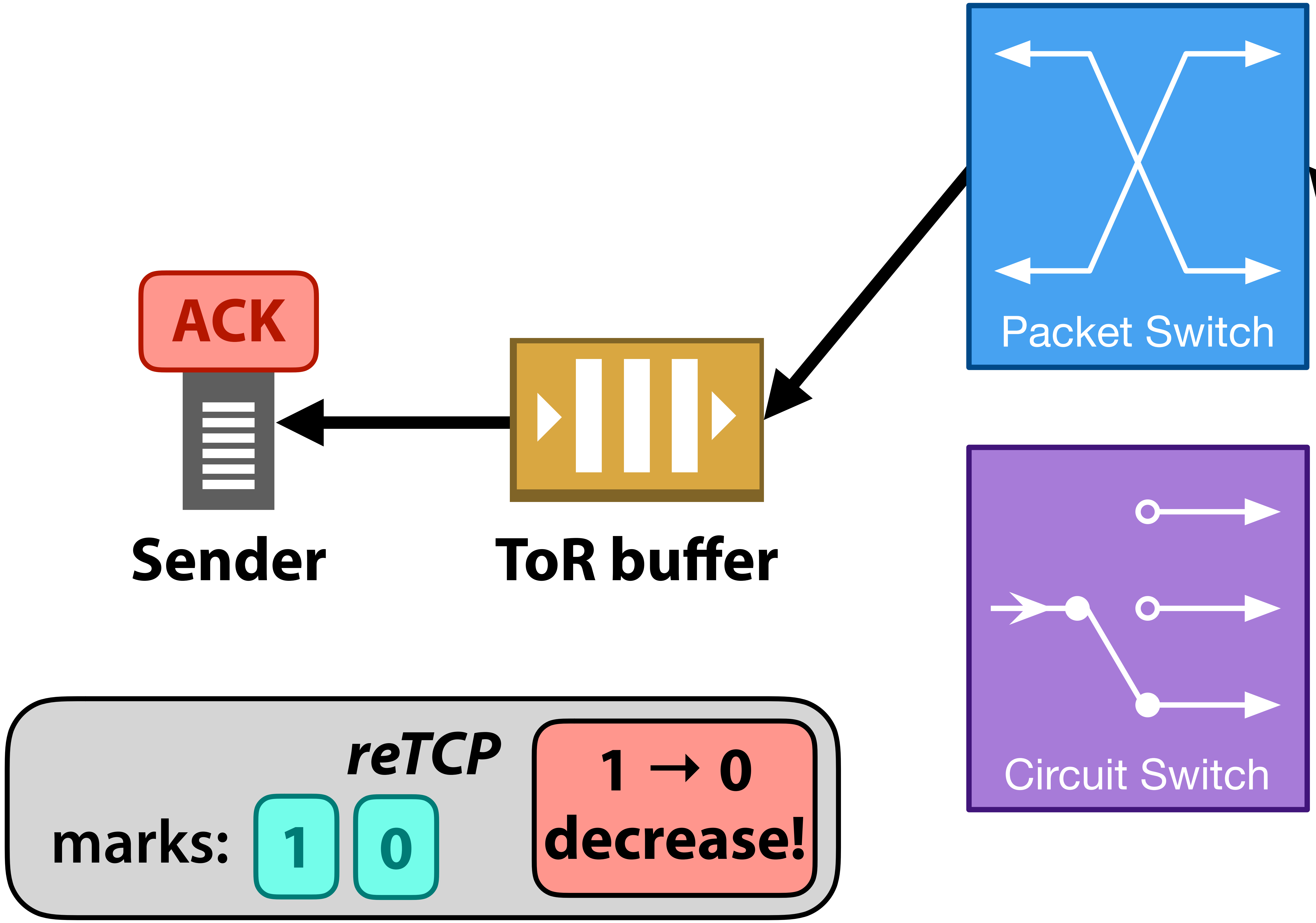
reTCP: Explicit circuit state feedback

Reuse existing ECN-Echo (ECE) bit



reTCP: Explicit circuit state feedback

Reuse existing ECN-Echo (ECE) bit



Single multiplicative increase/decrease

On 0 → 1 transitions:

`cwnd = cwnd` × α

On 1 → 0 transitions:

`cwnd = cwnd /` α

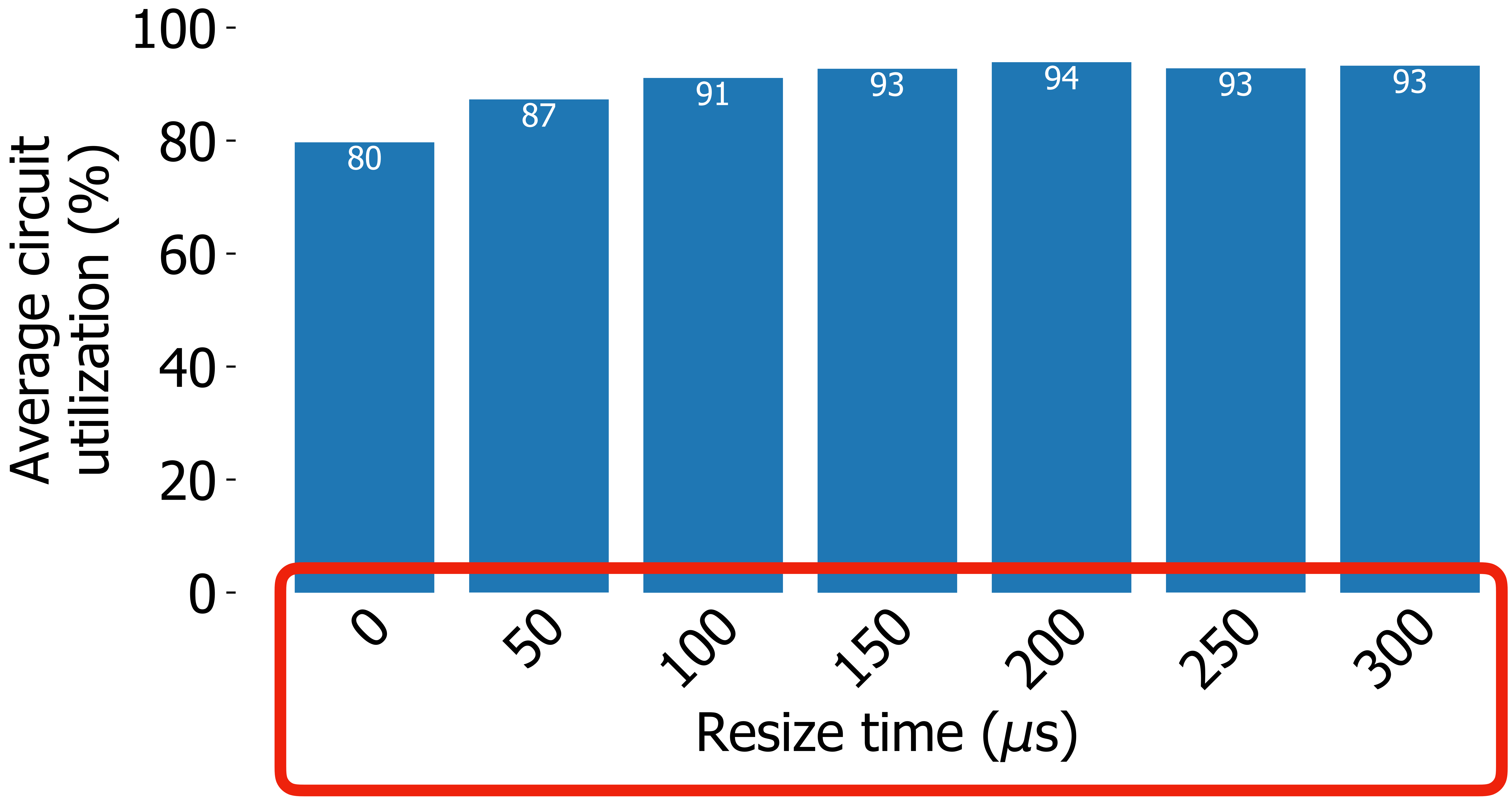
α depends on ratio of circuit BDP to ToR queue capacity:

- Circuit network BDP: 45 packets
- Small ToR queue capacity: 16 packets

We use α `= 2`

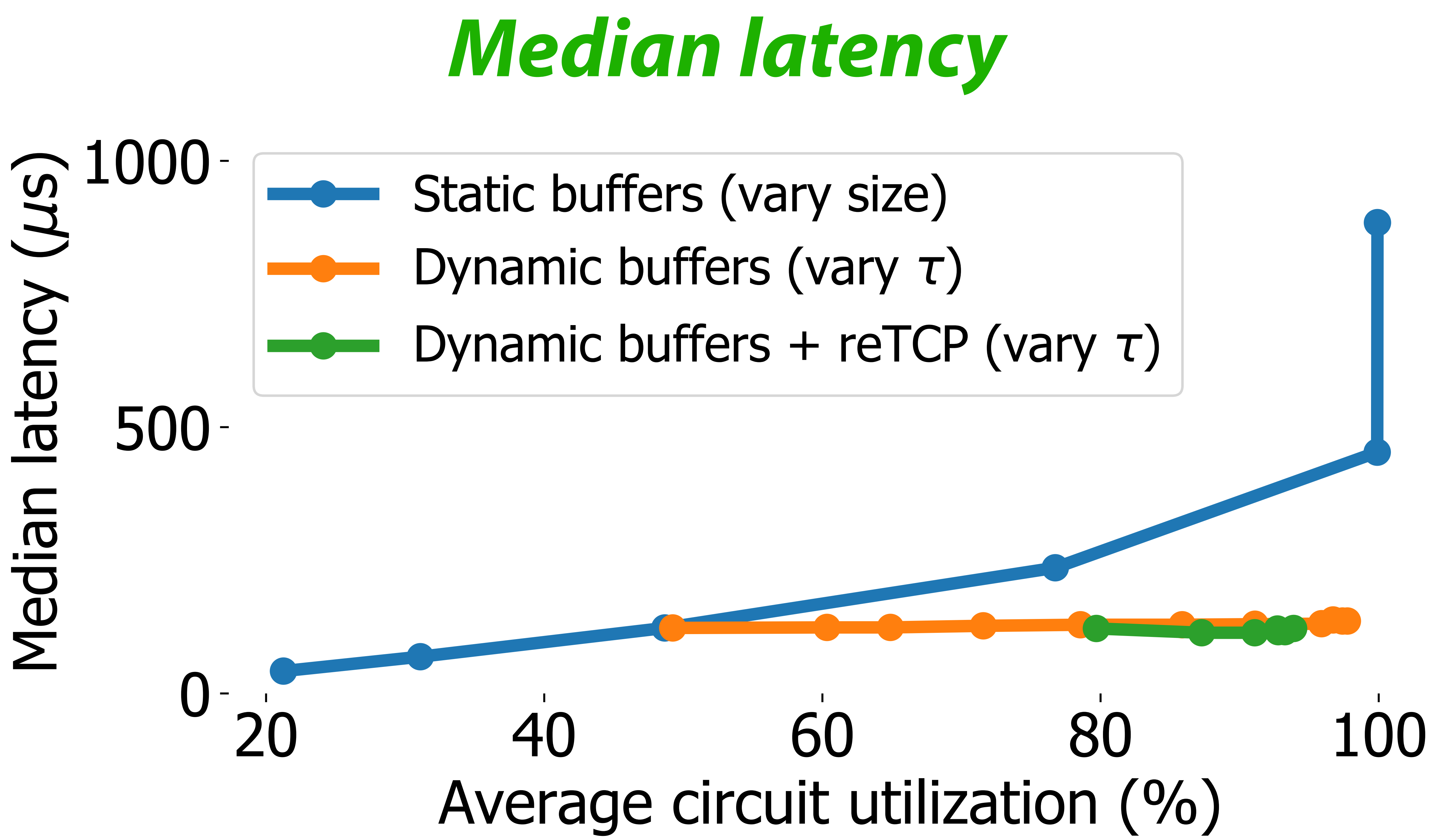
More advanced forms of feedback are possible

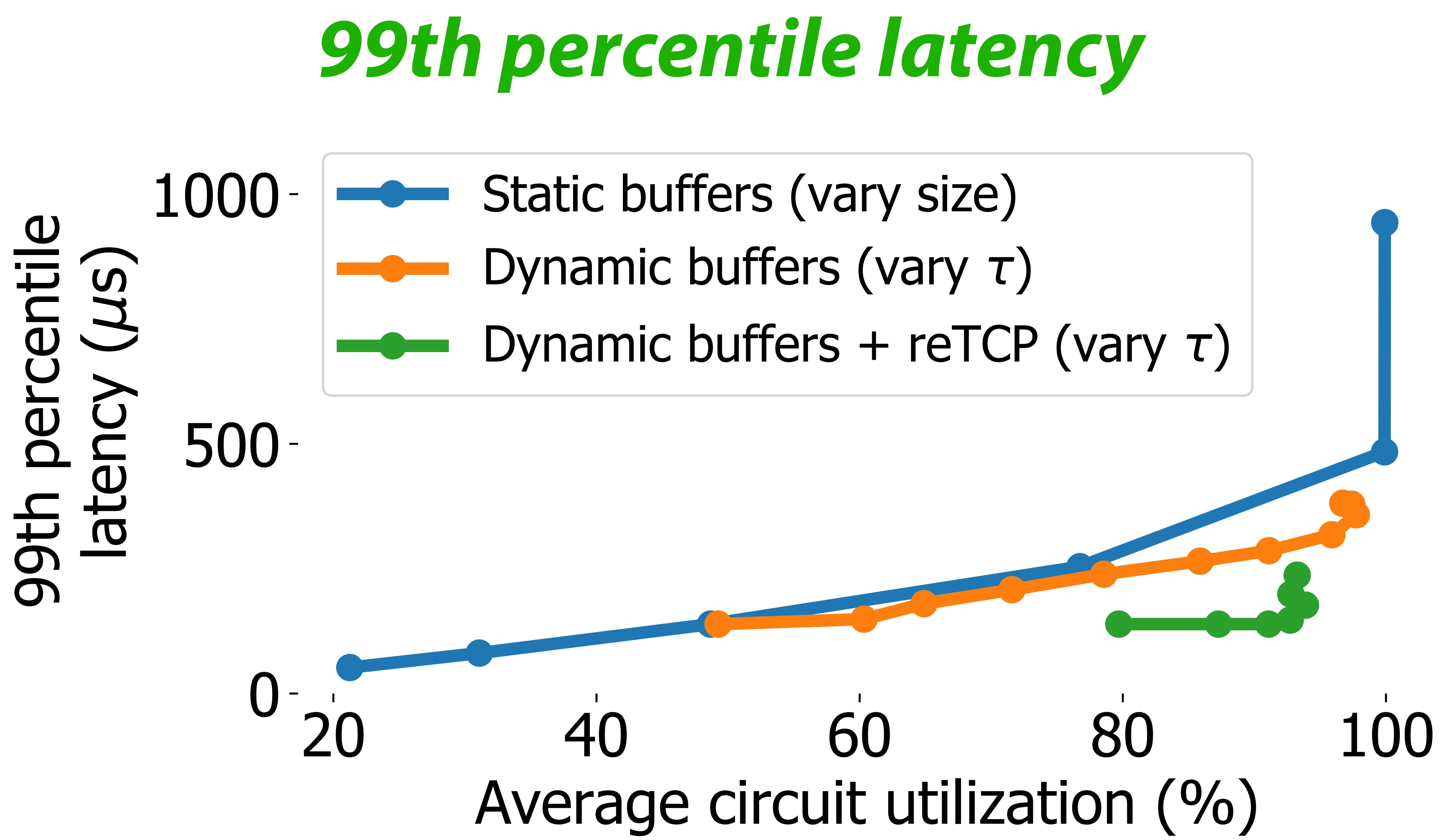
Dynamic buffers + reTCP achieve high utilization



16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s;
small buffers: 16 packets; large buffers: 50 packets

Short prebuffer time means low latency





With 150μs of prebuffering, dynamic buffers + reTCP achieve 93% circuit utilization with an only 1.20x increase in tail latency

16 flows from rack 1 to rack 2; packet network: 10 Gb/s; circuit network: 80 Gb/s; small buffers: 16 packets; large buffers: 50 packets

Limitations and future work

Dynamic buffer resizing and reTCP are designed to be minimally invasive

- Higher performance may be possible by involving the end-host further

Our evaluation used a simple traffic pattern to isolate TCP's behavior

- Important to consider complex workloads as well

Is TCP the right protocol for hybrid networks?

Summary: Adapting TCP for RDCNs

Bandwidth fluctuations in reconfigurable datacenter networks break TCP's implicit assumption of relative network stability

Two techniques to ramp up TCP during short-lived circuits

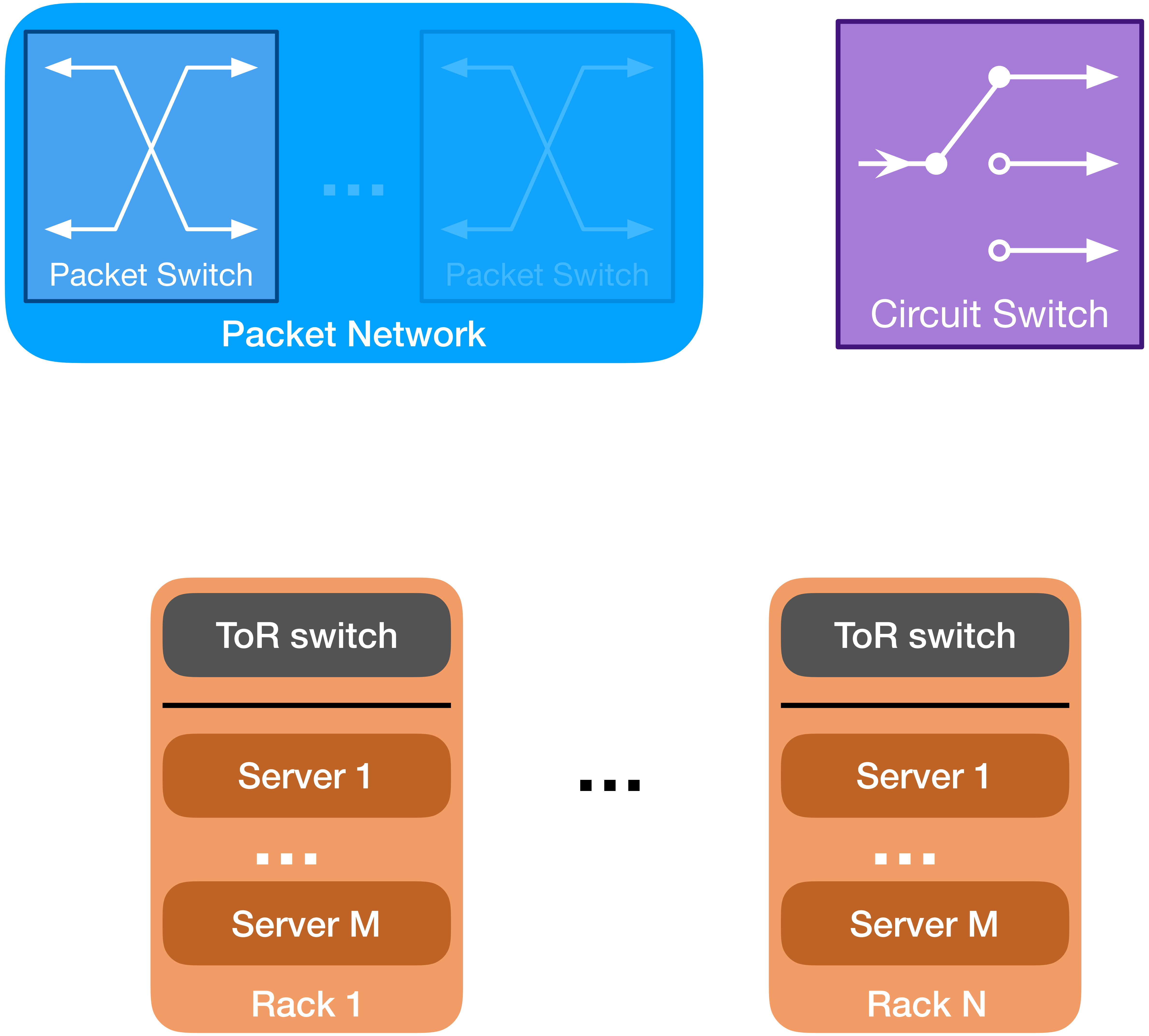
- ***Dynamic buffer resizing:*** Adapt ToR queues to packet or circuit network
- ***reTCP:*** Ramp up aggressively to fill new queue capacity

Etalon emulator open source at: ***github.com/mukerjee/etalon***

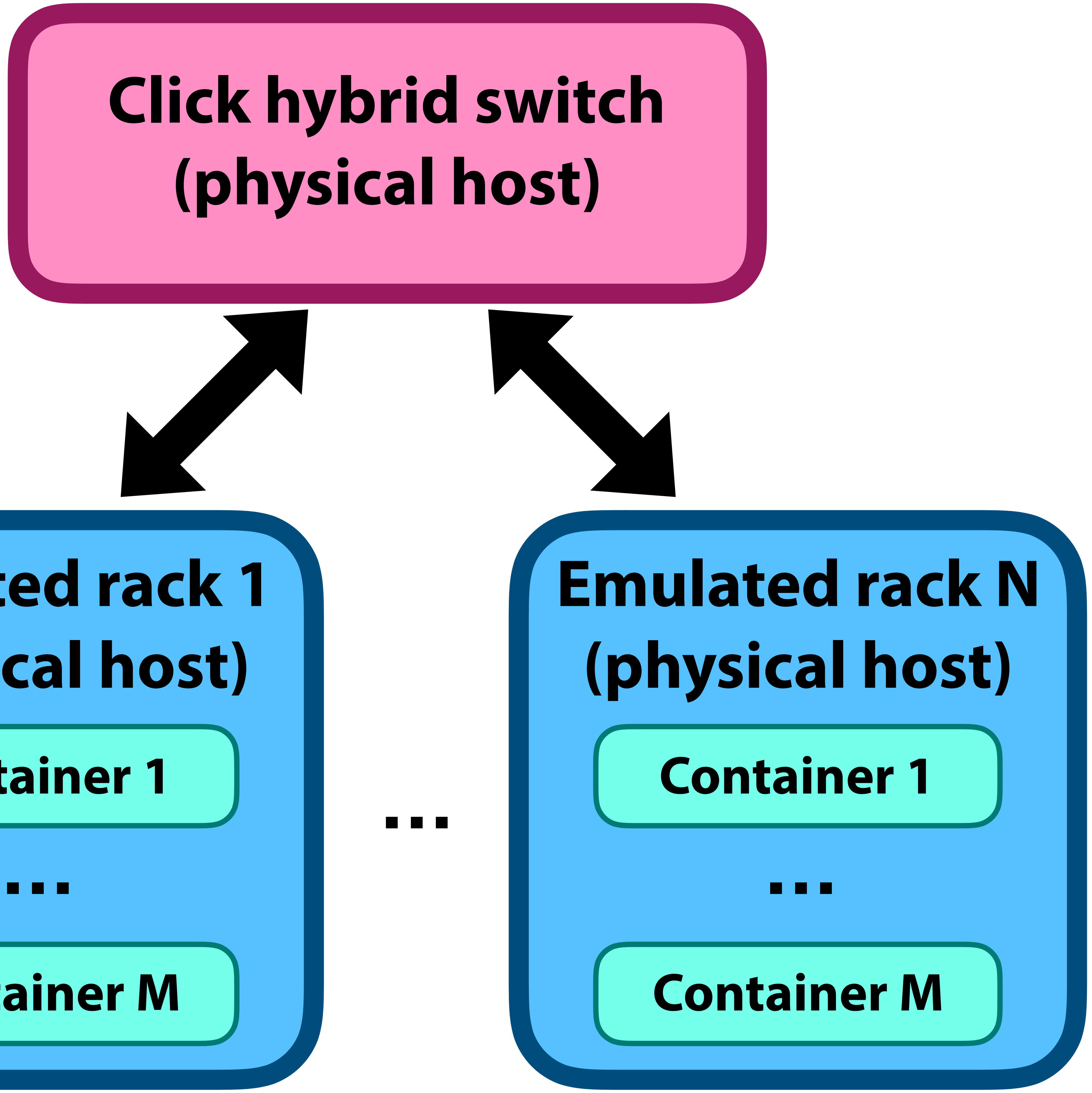
Christopher Canel ~ ccanel@cmu.edu

Thank you!

One more thing: *Etalon* emulator



One more thing: *Etalon* emulator



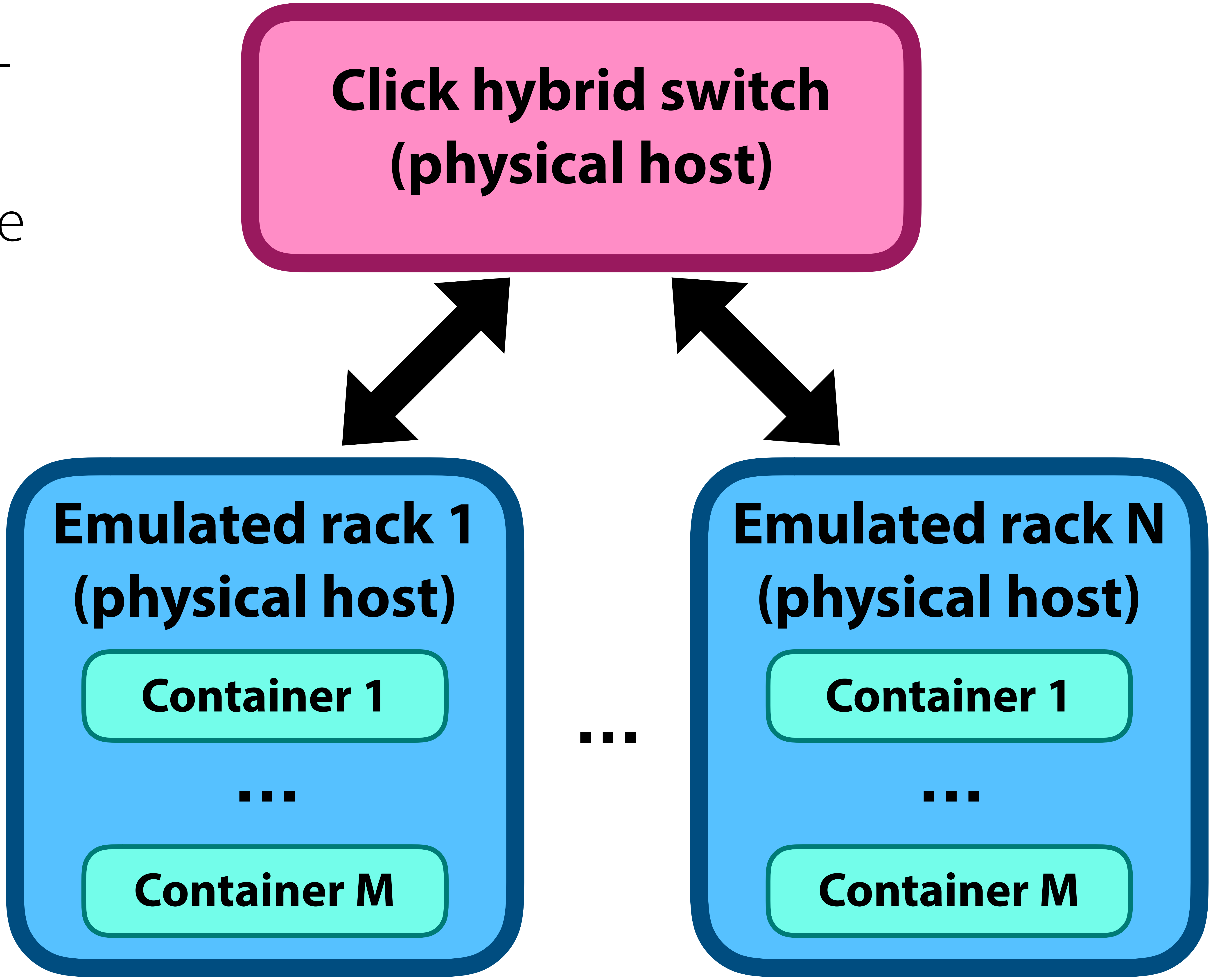
One more thing: *Etalon* emulator

Use ***time dilation*** to emulate high-bandwidth links

- "slows down" rest of the machine
- ***libVT:*** Catches common syscalls

Flowgrind to generate traffic

Strobe schedule: Each rack pair gets a circuit for an equal share



Summary: Adapting TCP for RDCNs

Bandwidth fluctuations in reconfigurable datacenter networks break TCP's implicit assumption of relative network stability

Two techniques to ramp up TCP during short-lived circuits

- ***Dynamic buffer resizing:*** Adapt ToR queues to packet or circuit network
- ***reTCP:*** Ramp up aggressively to fill new queue capacity

Etalon emulator open source at: ***github.com/mukerjee/etalon***

Christopher Canel ~ ccanel@cmu.edu

Thank you!