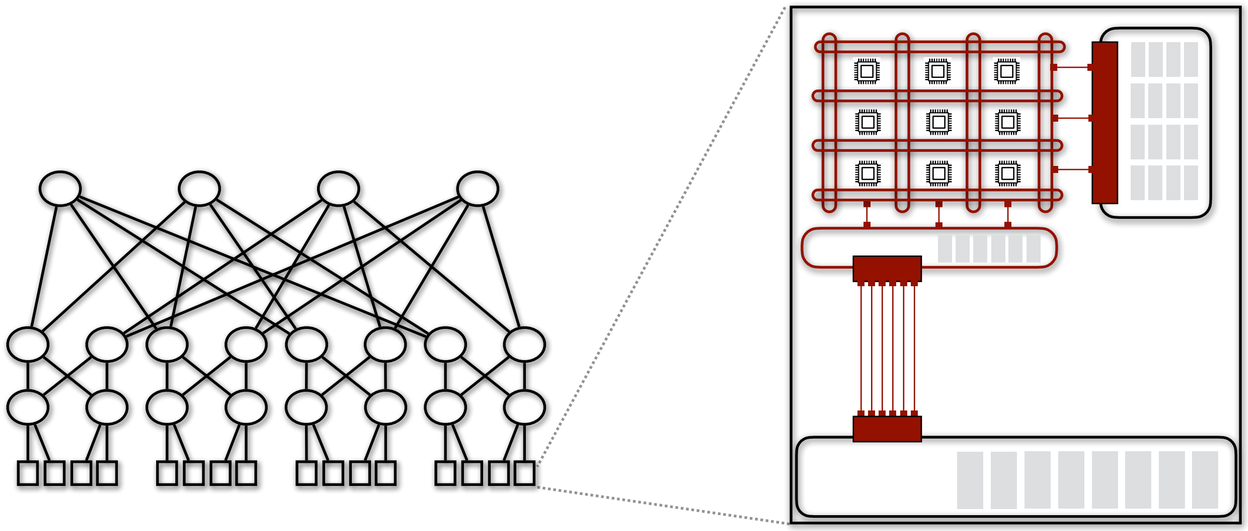
Host Congestion Control



Saksham Agarwal
UIUC

Arvind Krishnamurthy
UW & Google

Rachit Agarwal
Cornell

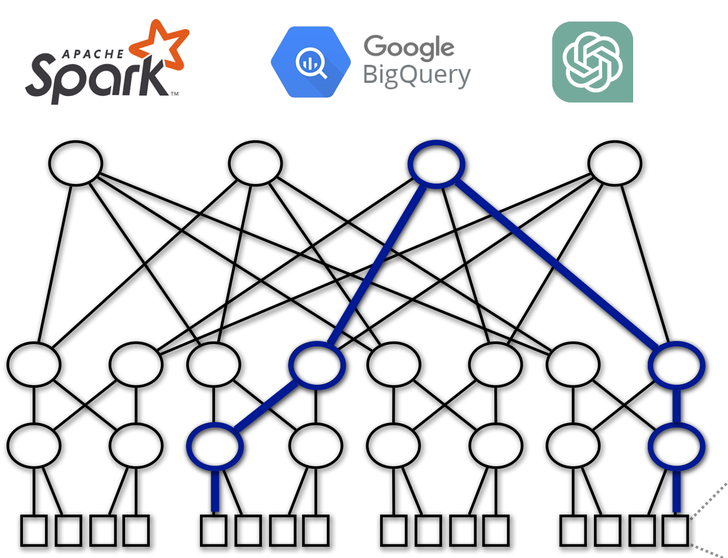
Datacenter Applications Require Fast Data Transfers

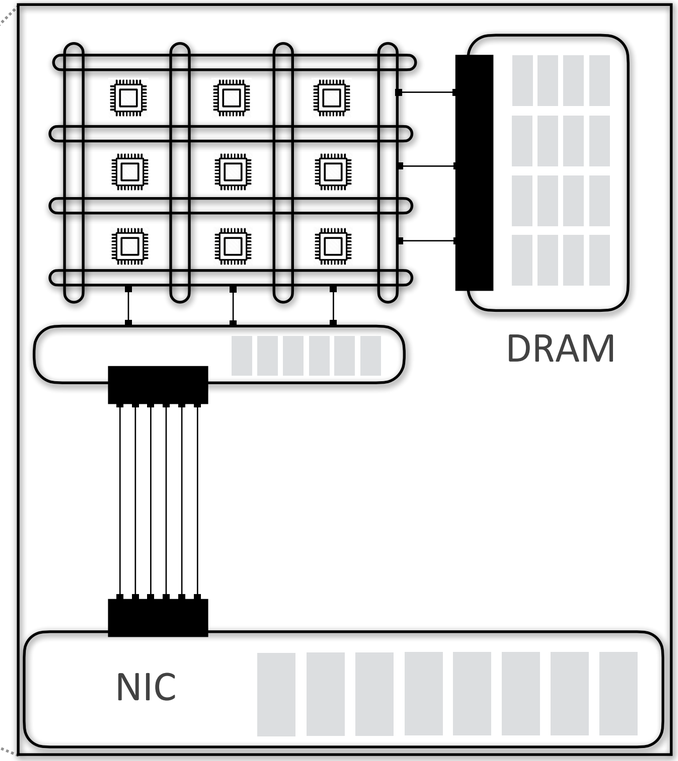
Datacenter apps: distributed and data-intensive

- Data analytics, ML training, etc
- Fast data transfers across multiple hosts

Additionally, fast data transfers <u>within</u> hosts

- Between processors, memory and peripheral devices
- NIC (network interface card): a networking peripheral

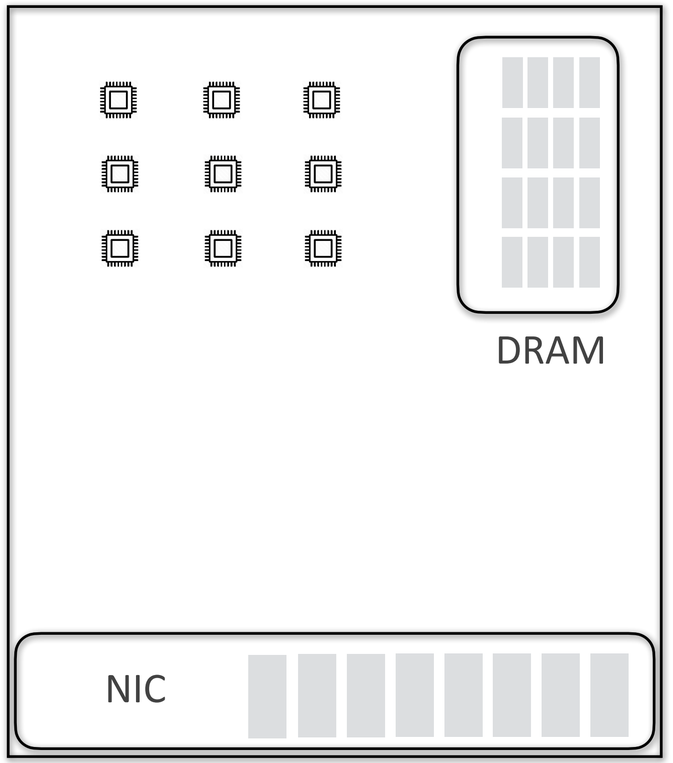




Host Network: The Network Within Every Host

Connects processors, memory and peripheral devices

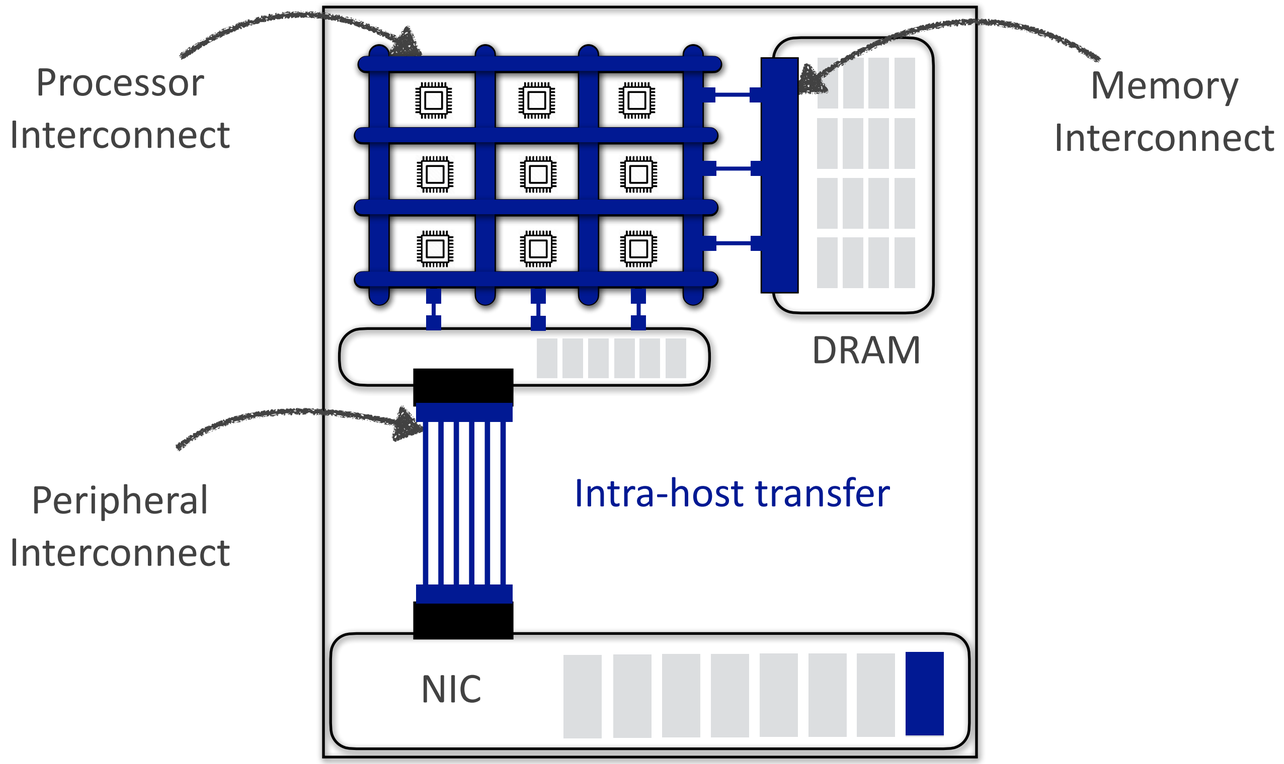
- Using processor, memory and peripheral interconnects



Host Network: The Network Within Every Host

Connects processors, memory and peripheral devices

- Using processor, memory and peripheral interconnects



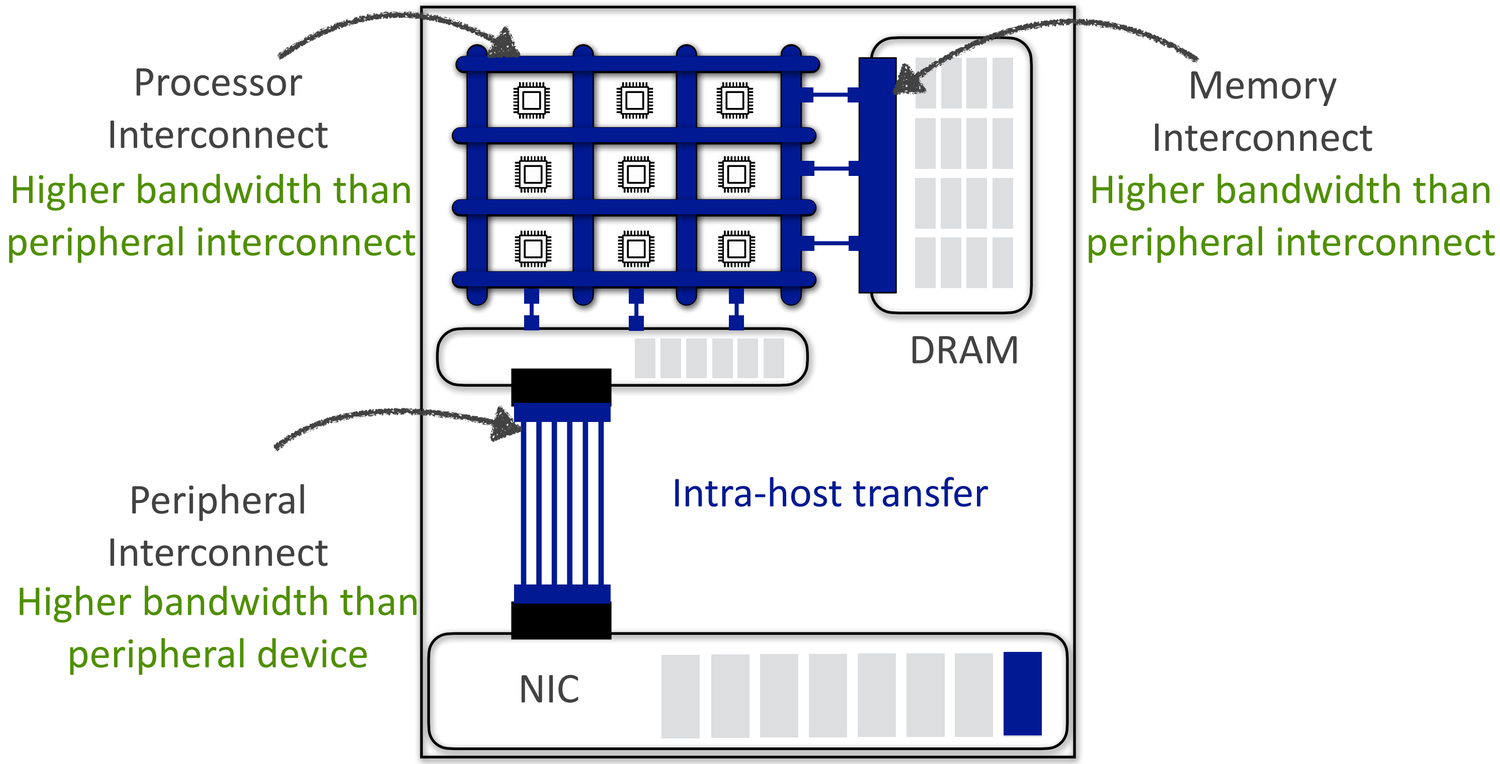
Host Network: The Network Within Every Host

Connects processors, memory and peripheral devices

- Using processor, memory and peripheral interconnects

Provides many desirable properties

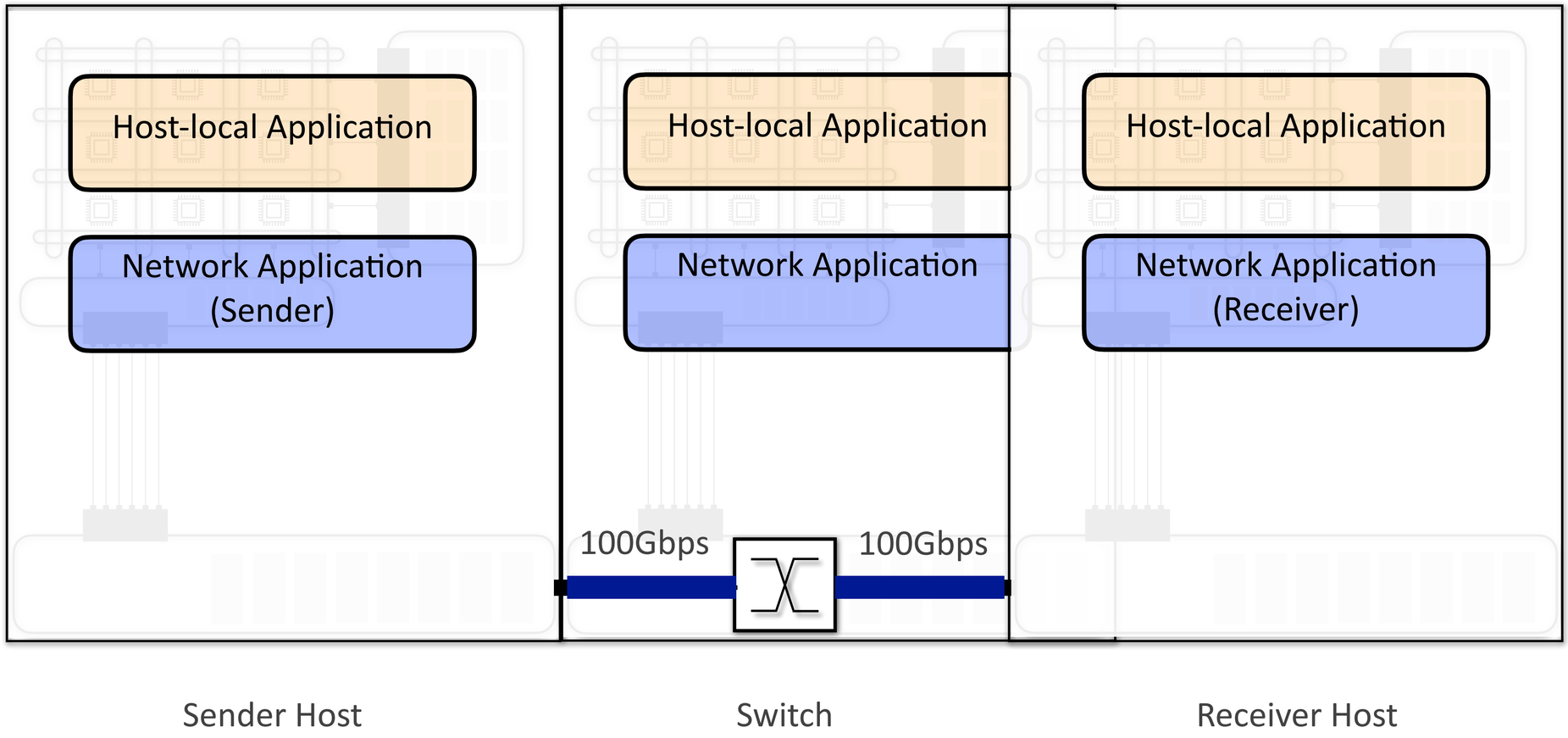
- ns-scale latencies, ample bandwidth, losslessness



Host Network: An Ideal Network…?

Typical datacenter workload

- Network application (sender writing data to receiver)
- + Host-local application (CPUs doing memory reads/writes)



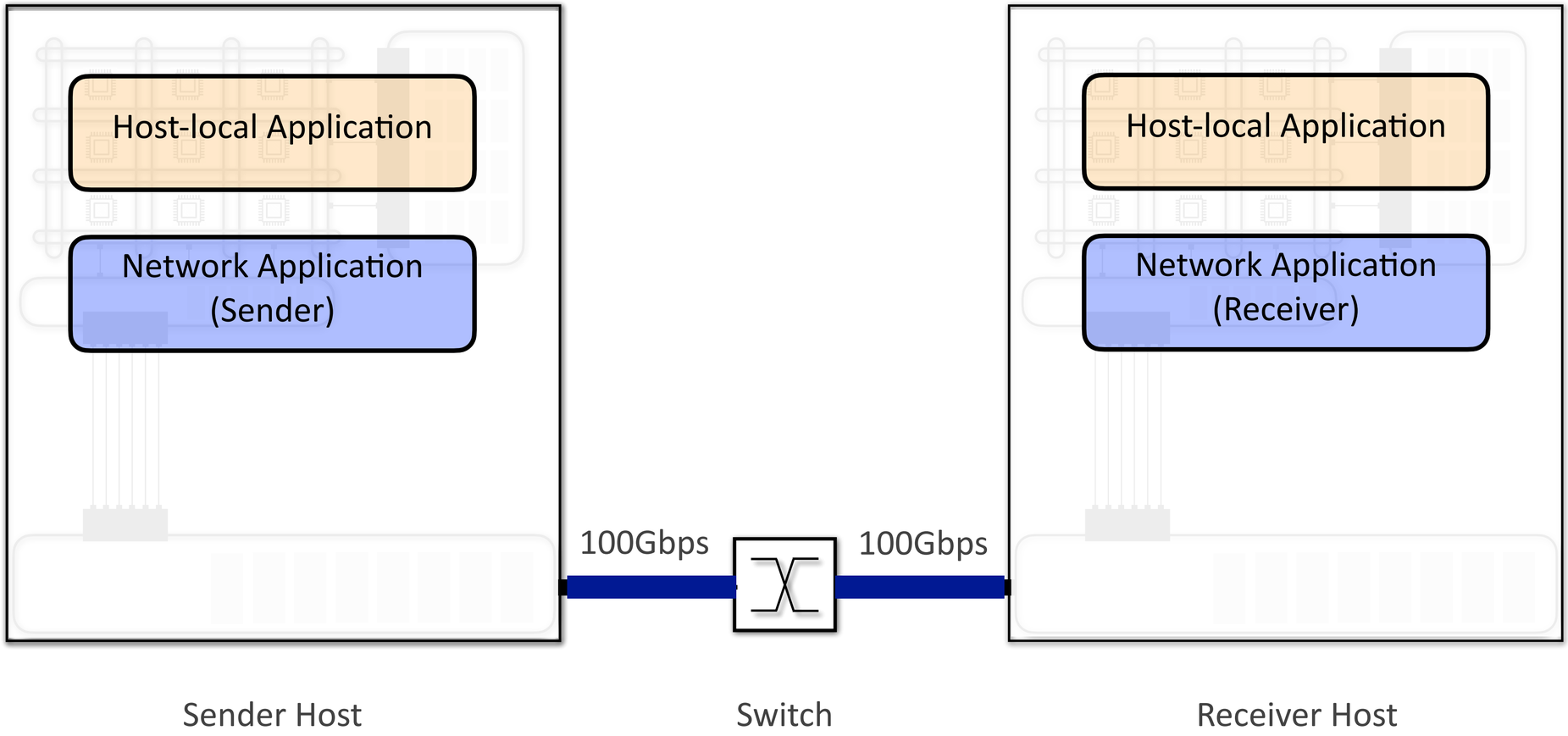
Host Network: An Ideal Network…?

Typical datacenter workload

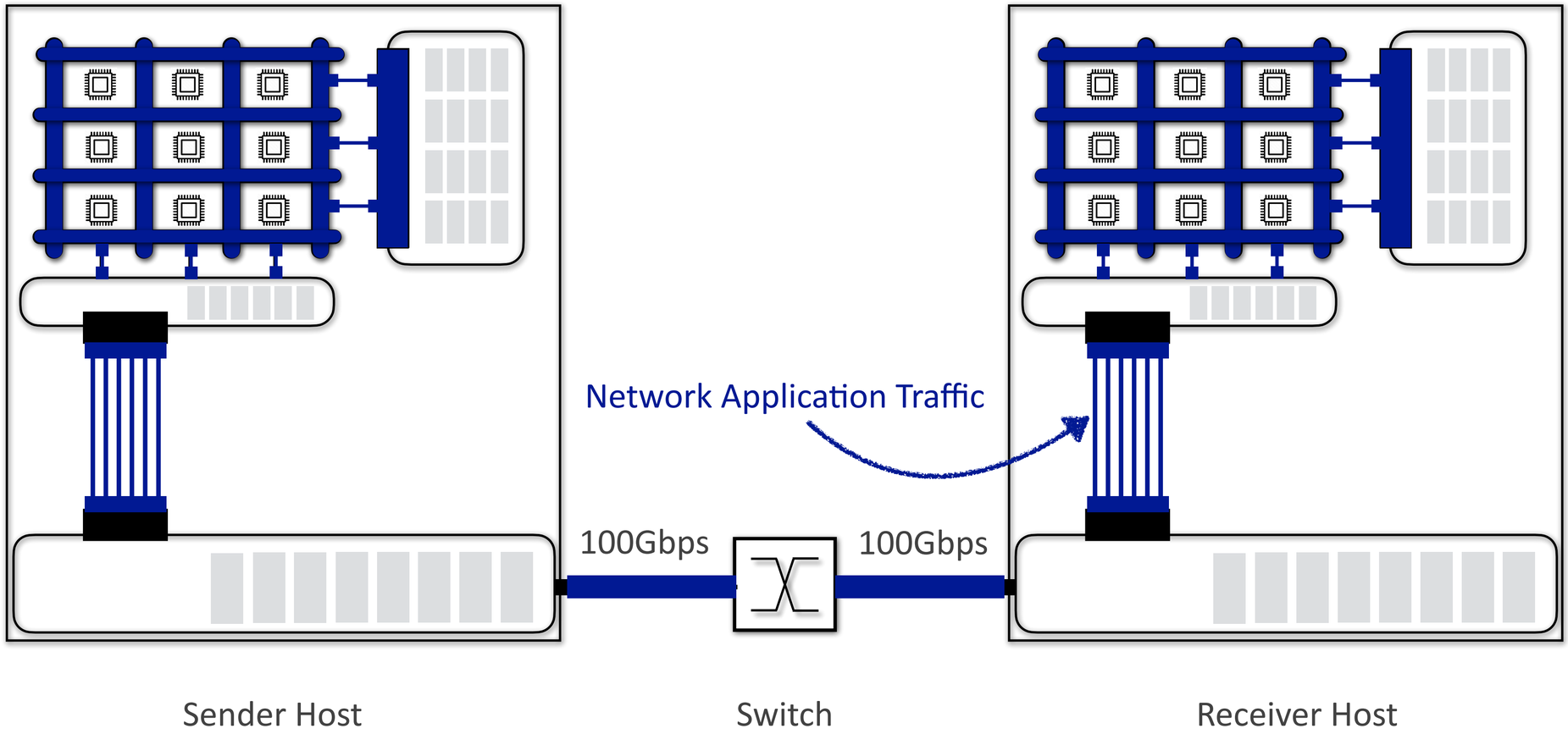
- Network application (sender writing data to receiver)
- + Host-local application (CPUs doing memory reads/writes)

Distributed apps: run at µs-scale; and NIC bandwidth

- Host network: run at ns-scale and higher bandwidth
- Intuition: should have minimal impact on apps



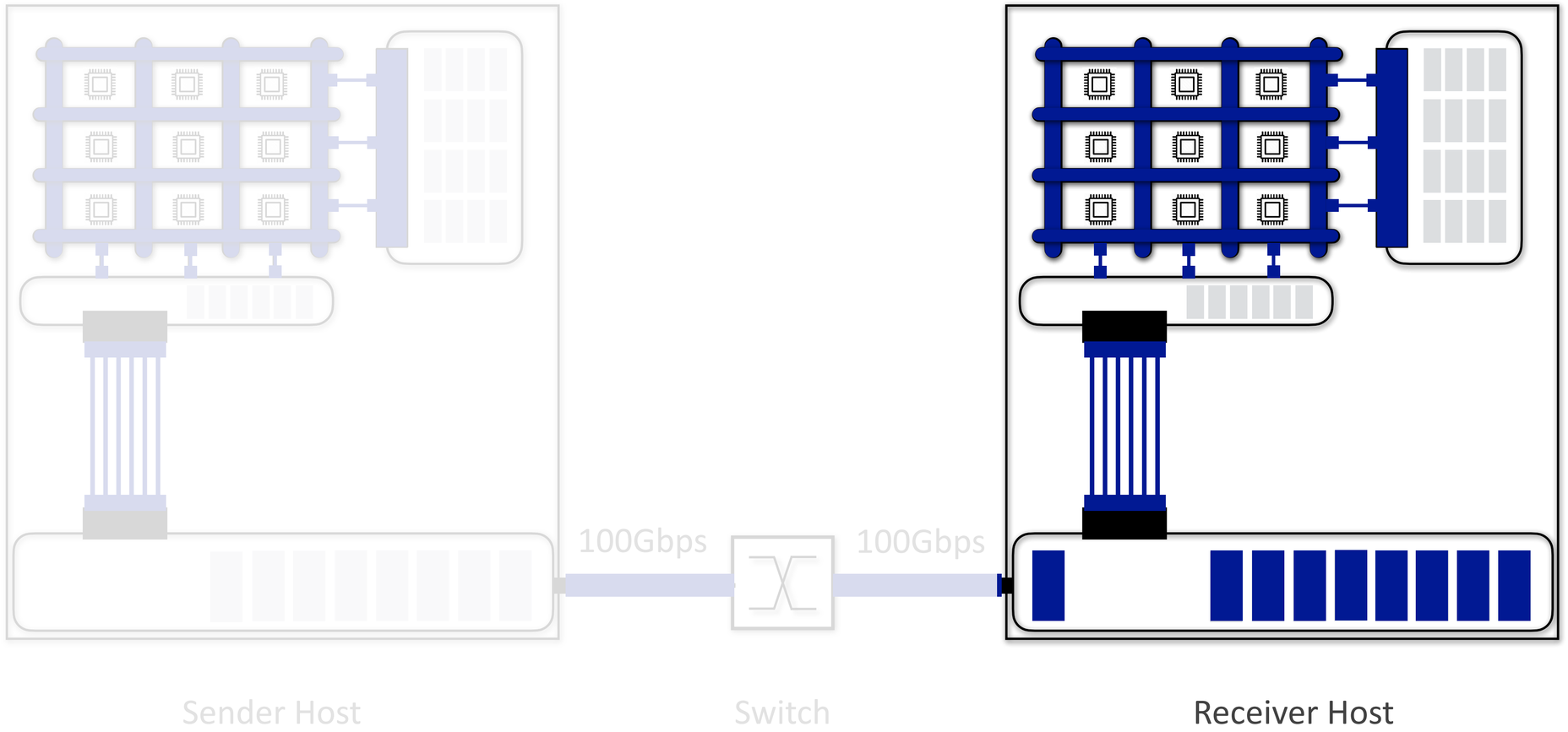
Host Network is Not an Ideal Network



Host Network is Not an Ideal Network

NIC unable to drain packets at arrival rate

- Data queued, and dropped at NIC buffers



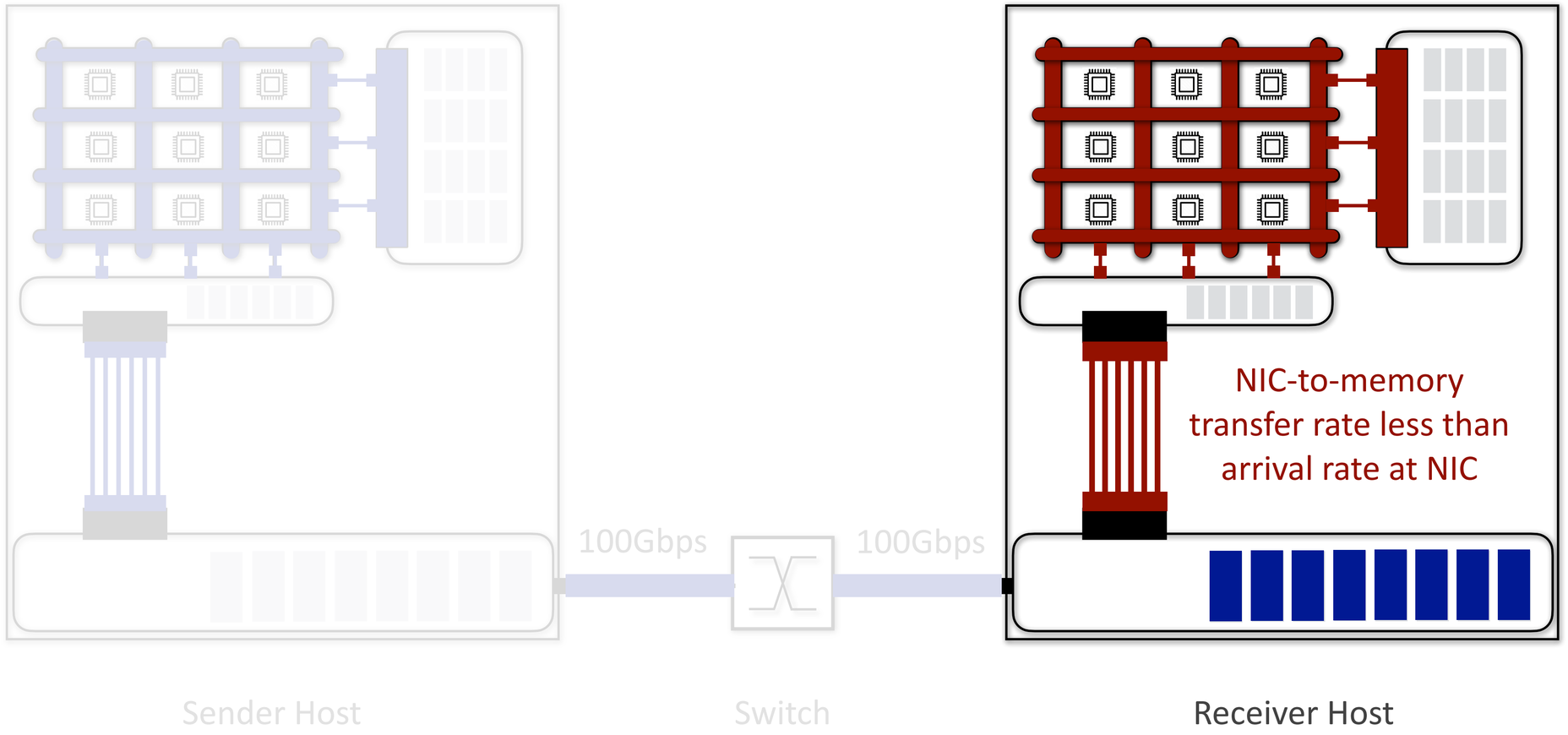
Host Network is Not an Ideal Network

NIC unable to drain packets at arrival rate

- Data queued, and dropped at NIC buffers

Inefficient NIC-to-memory transfers

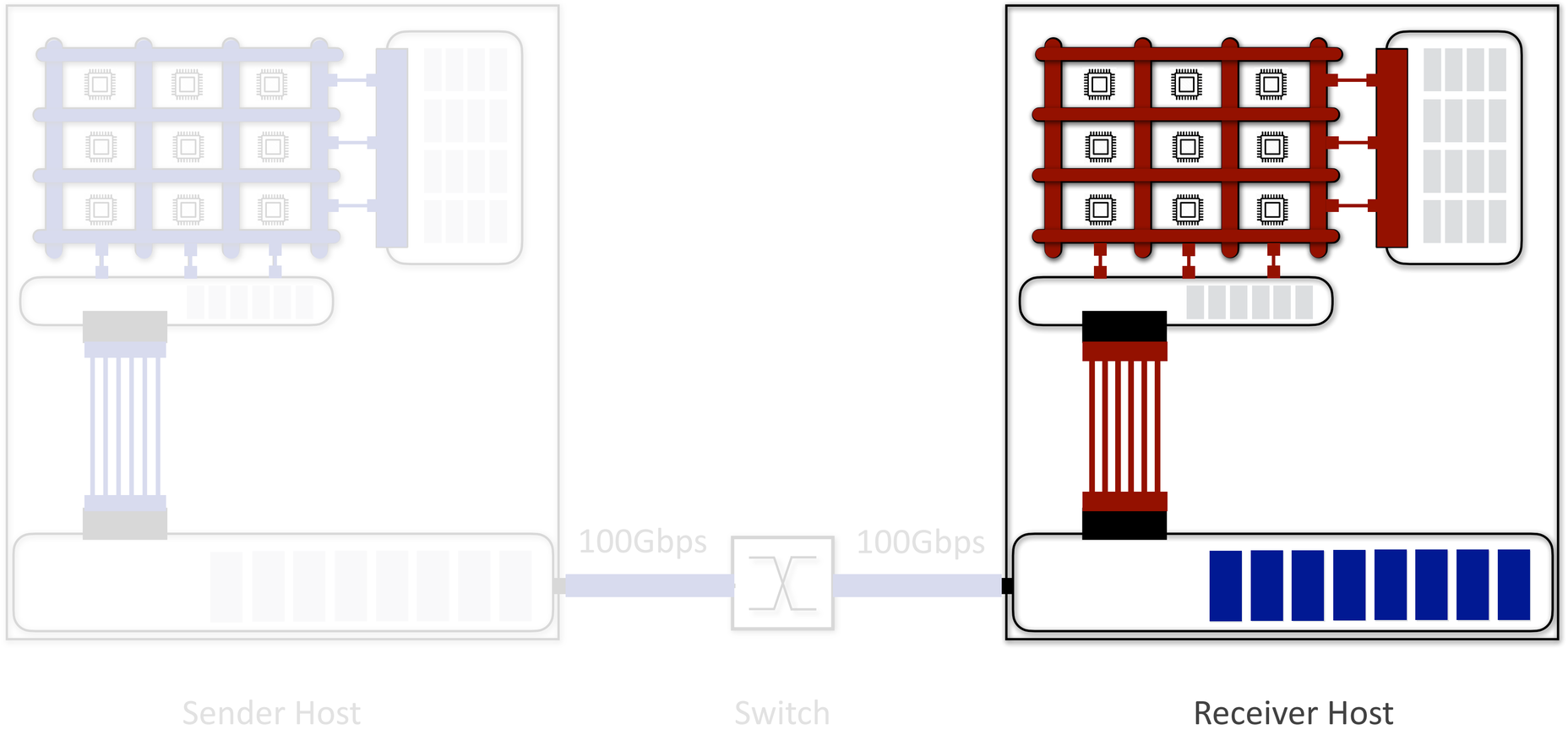
- Host network bandwidth underutilization



Emergence of Host Congestion: Contention Within the Host Network

Contention at memory interconnect

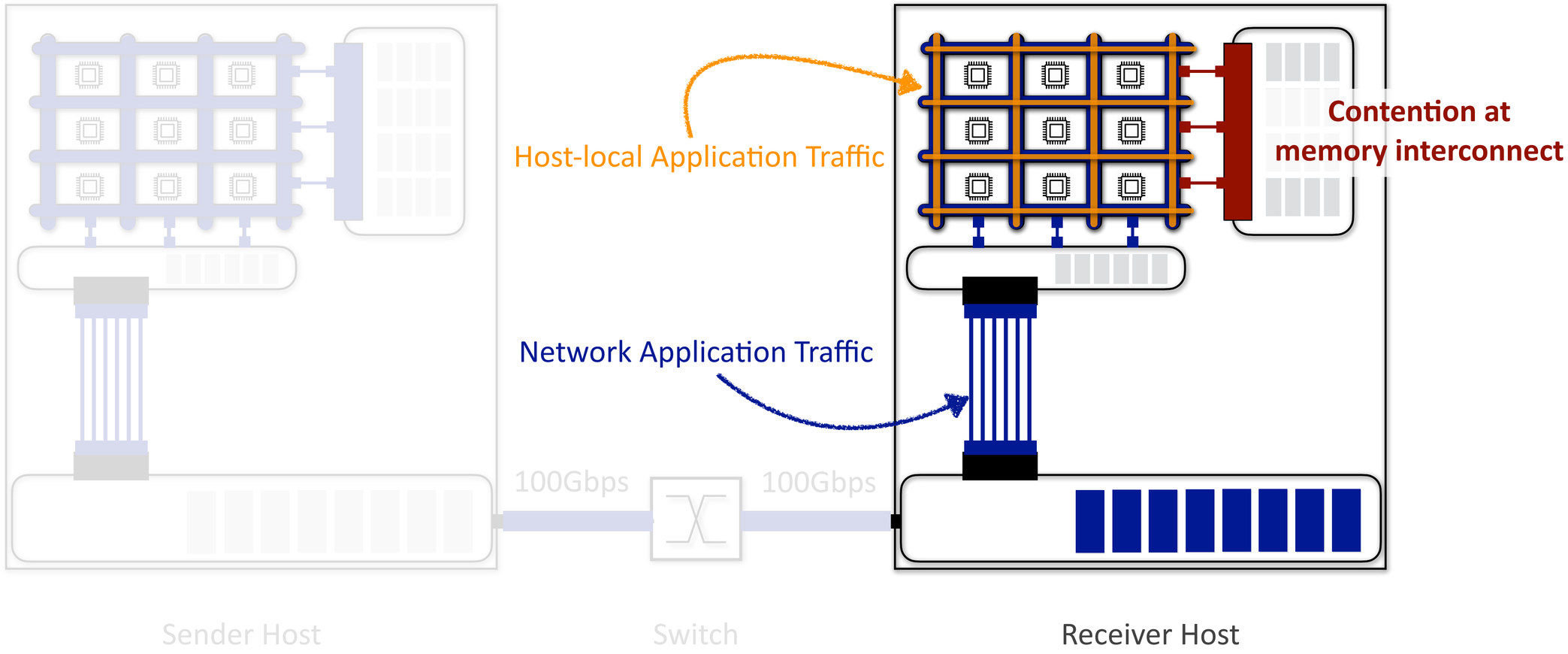
- Eg., competing traffic from peripheral devices and CPUs



Emergence of Host Congestion: Contention Within the Host Network

Contention at memory interconnect

- Eg., competing traffic from peripheral devices and CPUs



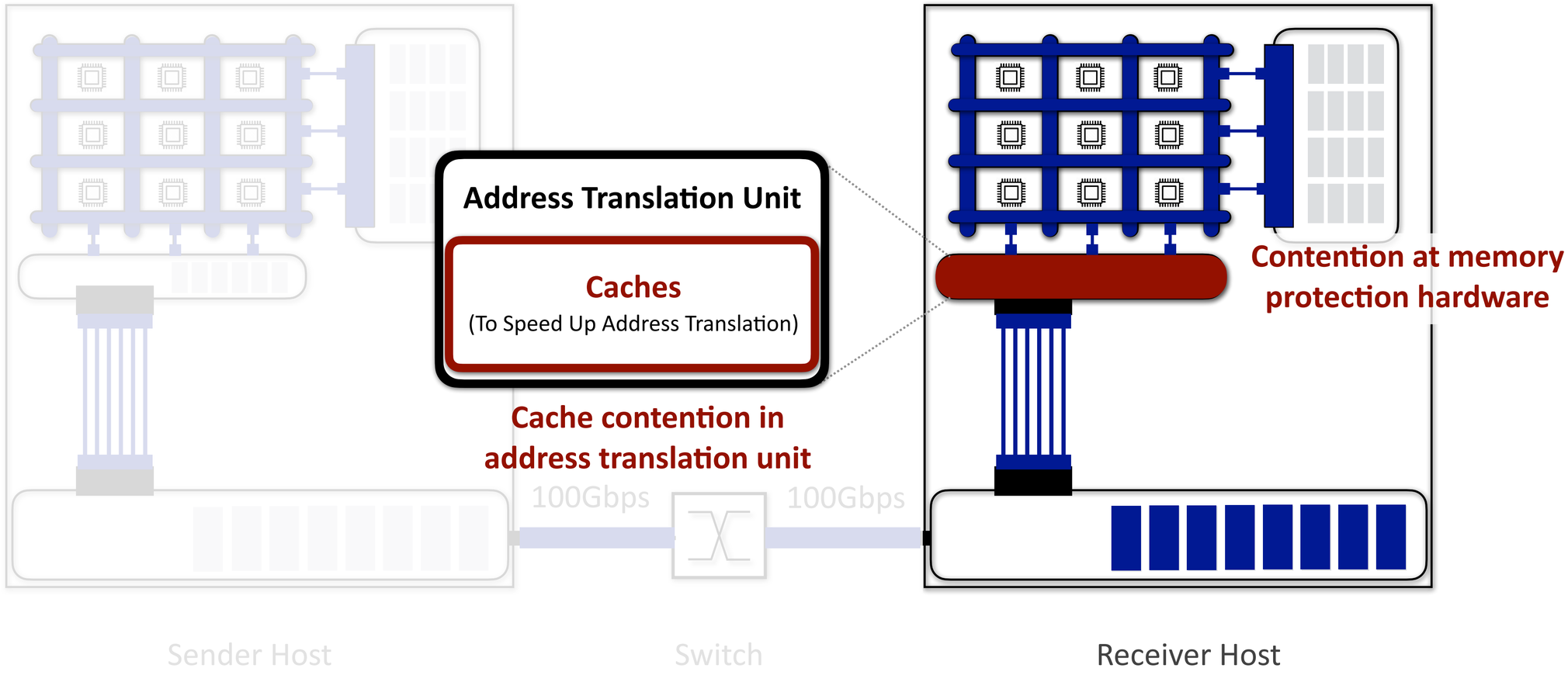
Emergence of Host Congestion: Contention Within the Host Network

Contention at memory interconnect

- Eg., competing traffic from peripheral devices and CPUs

Contention at memory protection hardware

- Eg., competing NIC-to-memory transfers



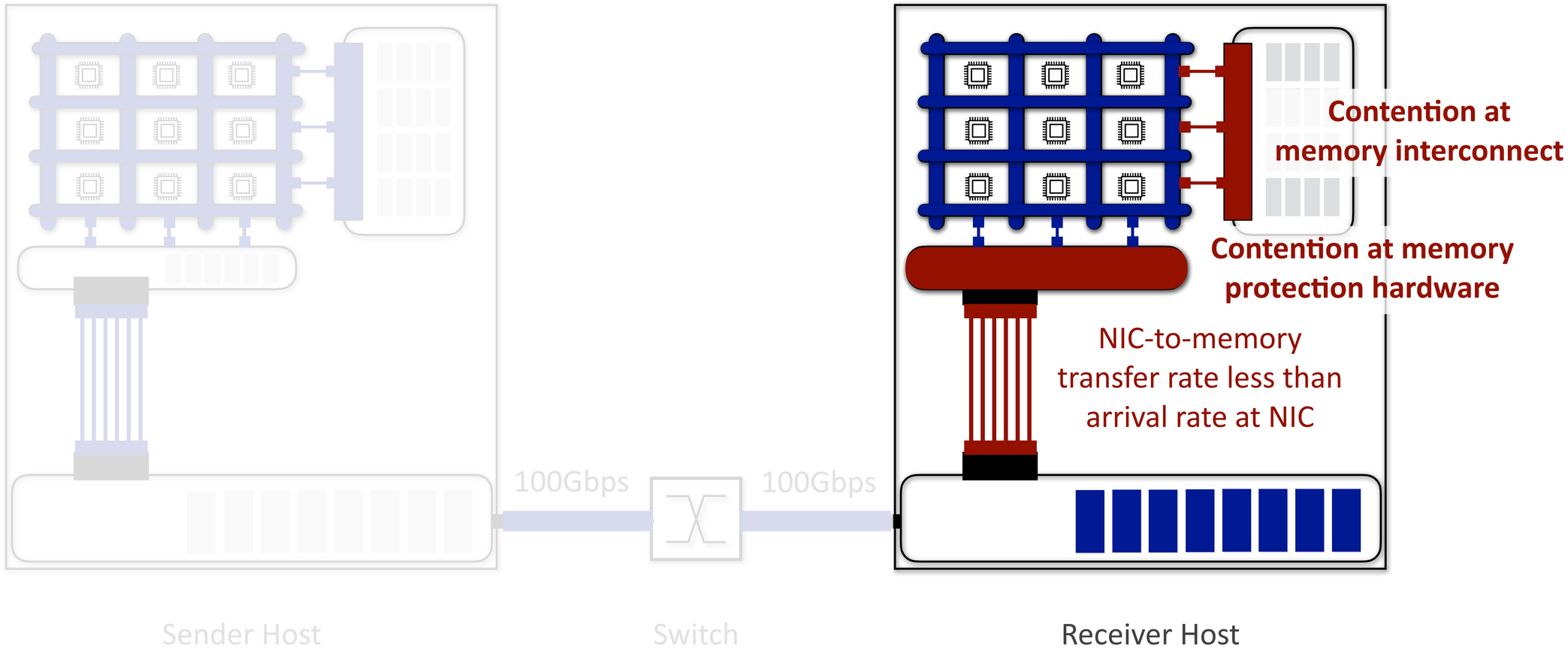
Emergence of Host Congestion: Contention Within the Host Network

Contention at memory interconnect

- Eg., competing traffic from peripheral devices and CPUs

Contention at memory protection hardware

- Eg., competing NIC-to-memory transfers



Emergence of Host Congestion: Contention Within the Host Network

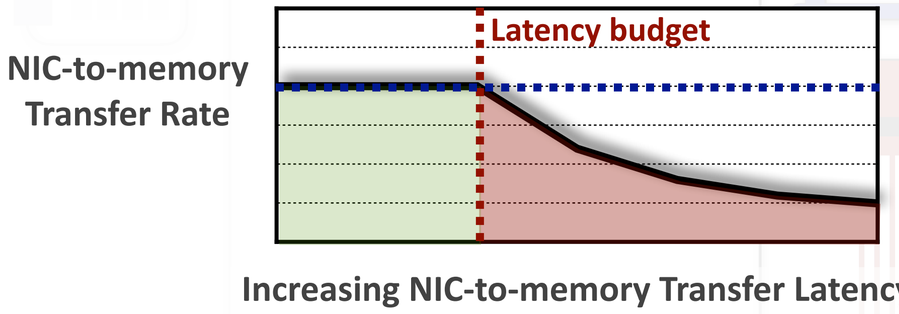
Contention at memory interconnect

- Eg., competing traffic from peripheral devices and CPUs

Contention at memory protection hardware

- Eg., competing NIC-to-memory transfers

$$\text{Latency Budget} \propto \frac{\text{Hardware Parallelism}}{\text{NIC Bandwidth}}$$



Nanosecond-scale latency inflation within the host network ⇒ host congestion

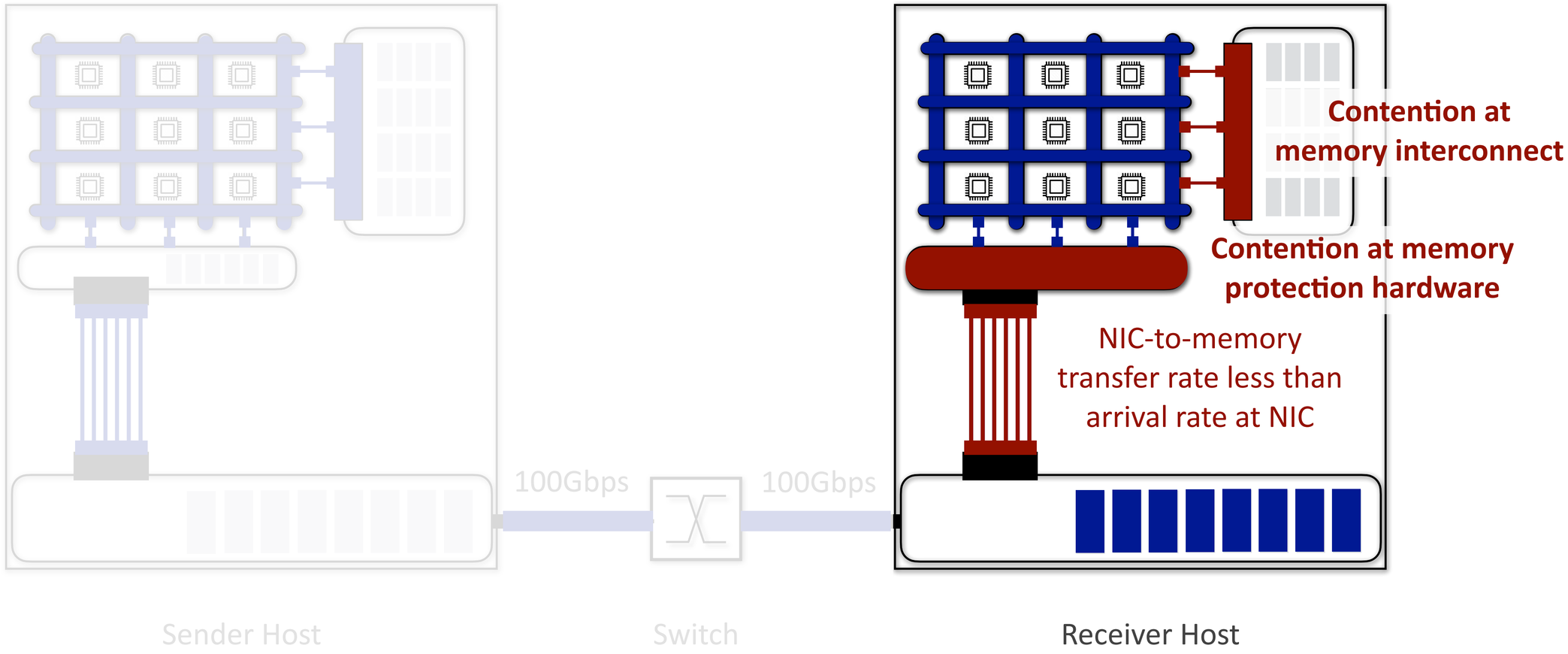
Emergence of Host Congestion: Contention Within the Host Network

Contention at memory interconnect

- Eg., competing traffic from peripheral devices and CPUs

Contention at memory protection hardware

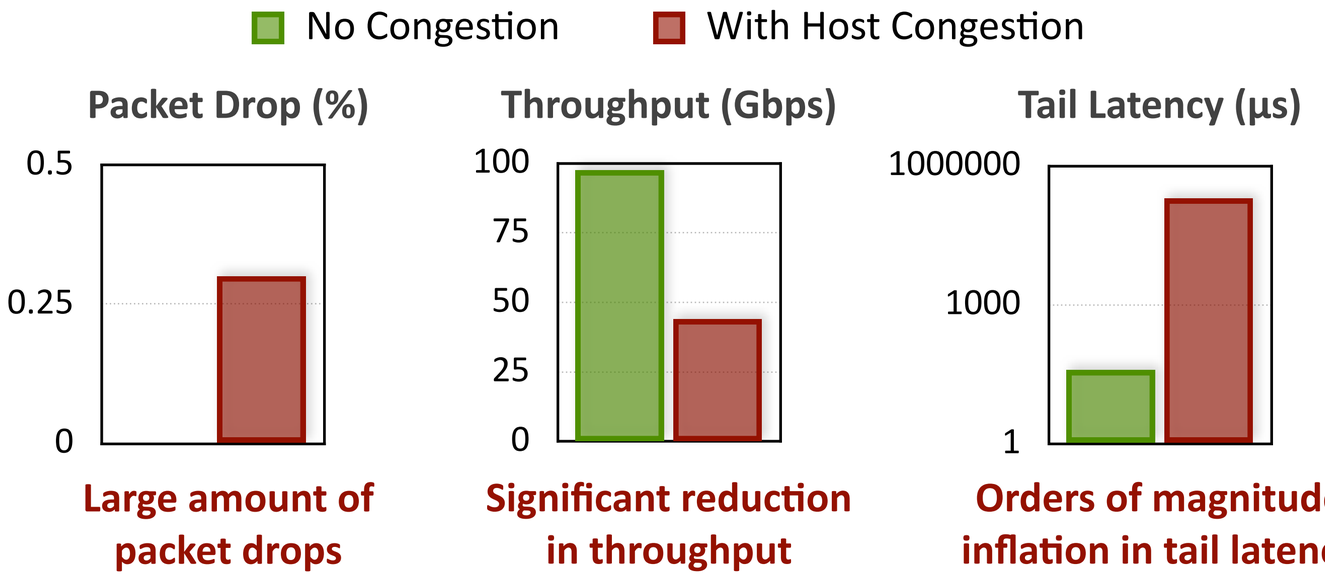
- Eg., competing NIC-to-memory transfers



Impact of Host Congestion

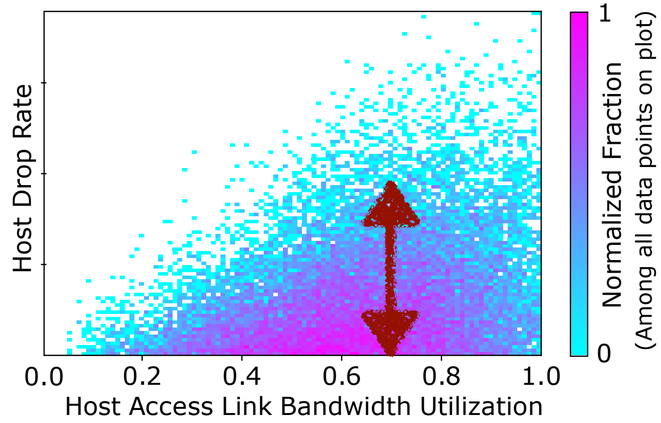
Significant application-level performance degradation

- Using Linux network stack and Datacenter TCP (DCTCP)



Real-world impact: evidence from production cluster

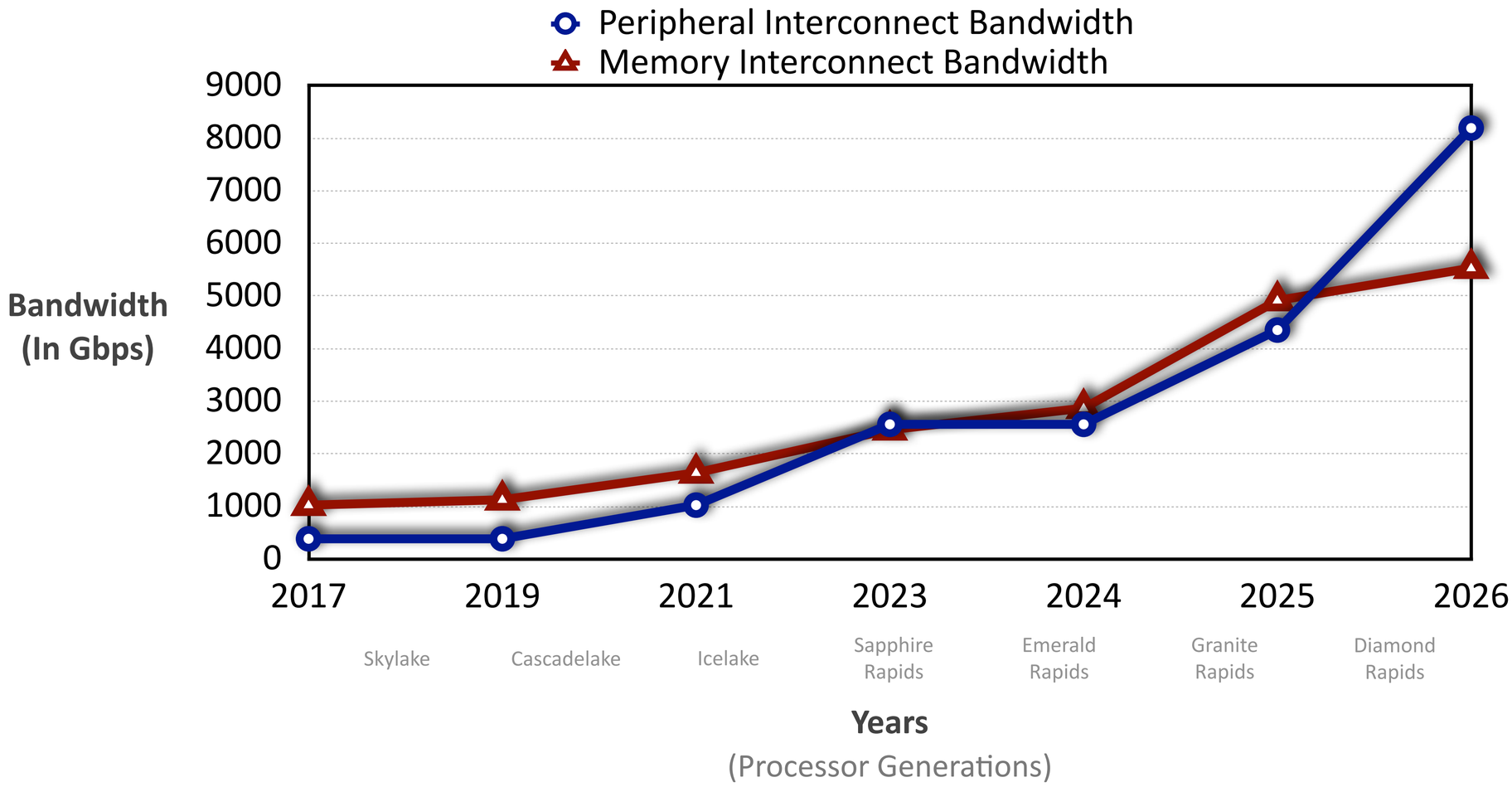
- Even using state-of-the-art network stacks & protocols



Drops due to host congestion in Google's production cluster

Reproduced similar or even worse results in hardware offloaded network stacks
(eg., using SmartNICs and RDMA over Ethernet)

Recent Technology Trend: Diminishing Gap Between Peripheral and Memory Bandwidth

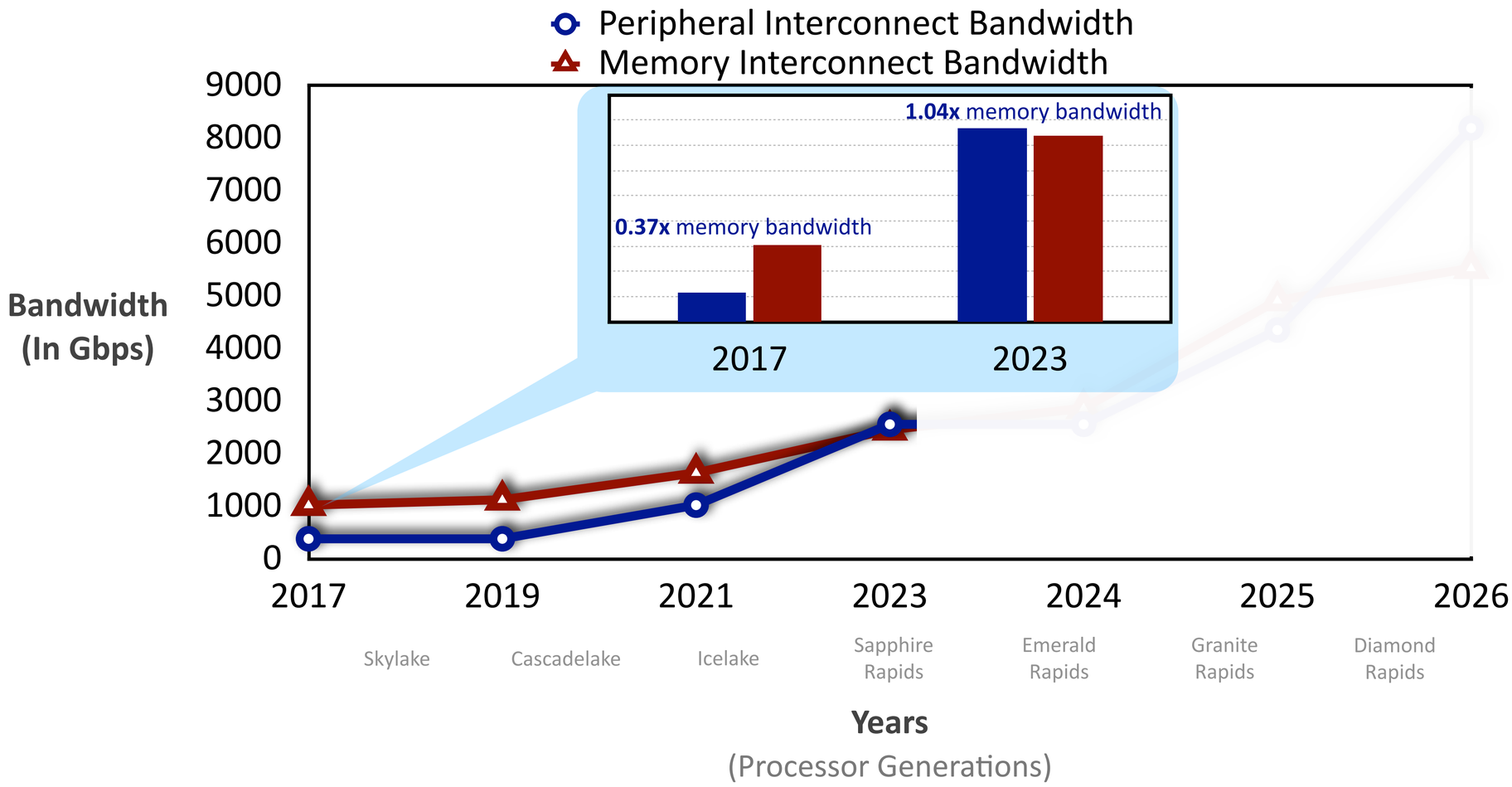


https://wccftech.com/intel-emerald-rapids-5th-gen-xeon-cpu-family-specs-leaked-up-to-64-cores-ddr5-5600-80-gen-5-lanes-350w/

Recent Technology Trend: Diminishing Gap Between Peripheral and Memory Bandwidth

PCIe has quickly approached memory bandwidth

- Easier to approach host congestion



https://wccftech.com/intel-emerald-rapids-5th-gen-xeon-cpu-family-specs-leaked-up-to-64-cores-ddr5-5600-80-gen-5-lanes-350w/

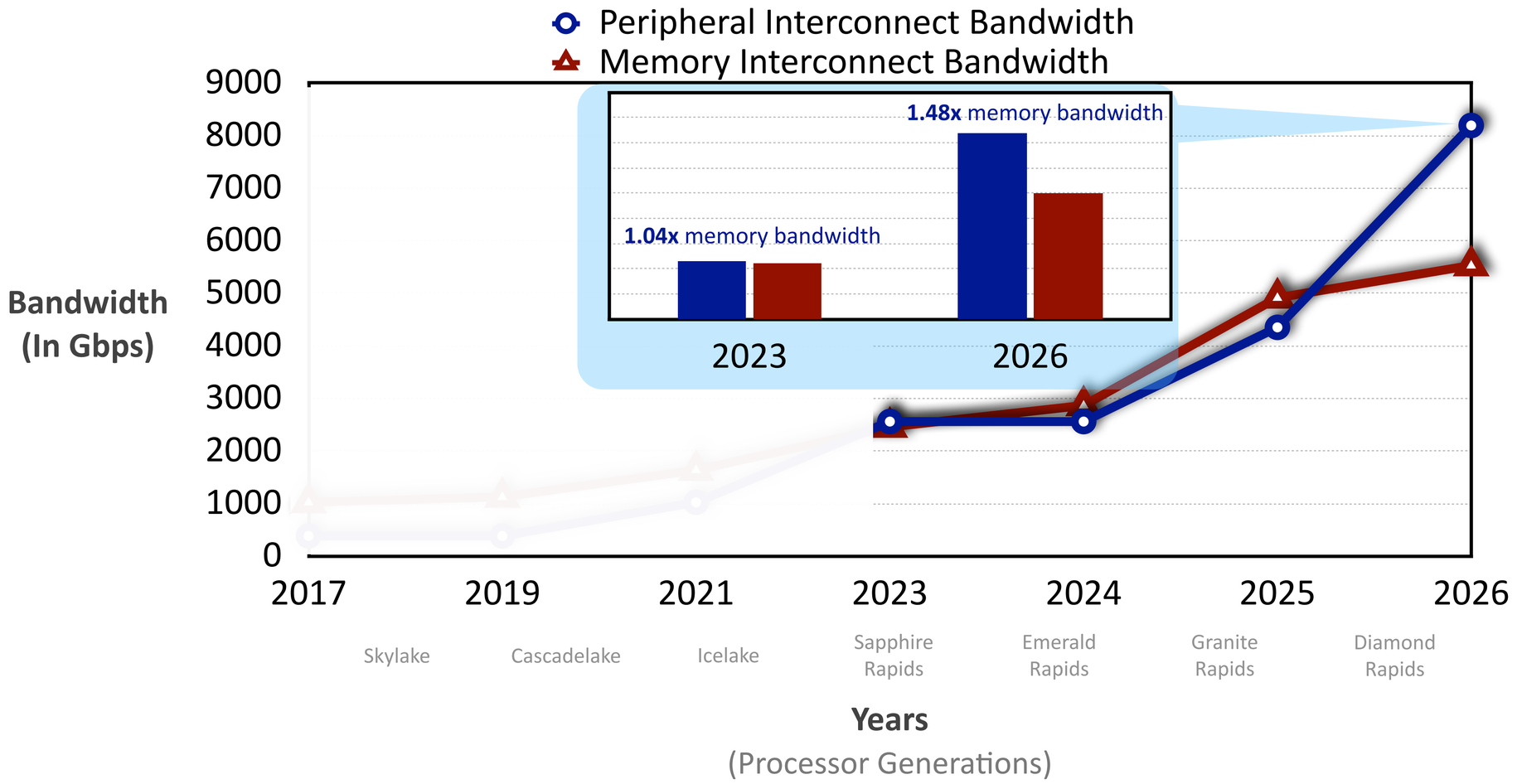
Recent Technology Trend: Diminishing Gap Between Peripheral and Memory Bandwidth

PCIe has quickly approached memory bandwidth

- Easier to approach host congestion

Expected to further outgrow memory bandwidth

- Problem going to worsen over time



https://wccftech.com/intel-emerald-rapids-5th-gen-xeon-cpu-family-specs-leaked-up-to-64-cores-ddr5-5600-80-gen-5-lanes-350w/

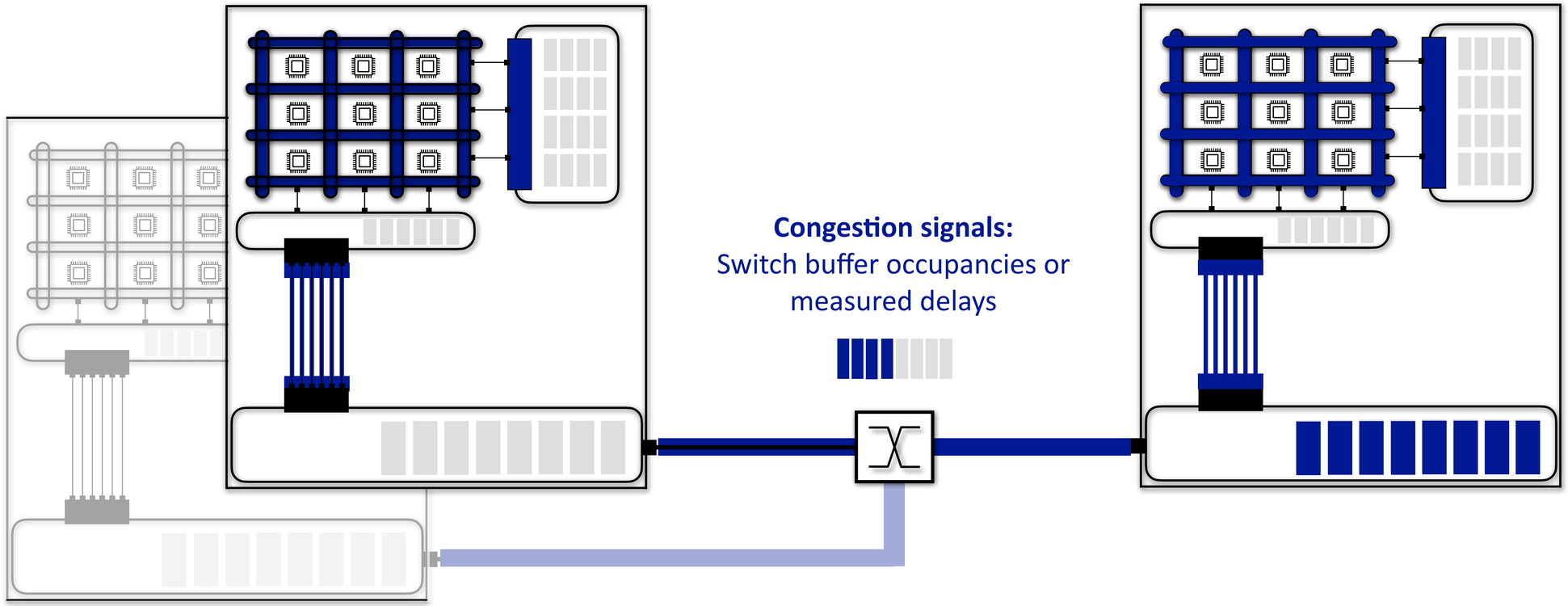
Congestion Control (CC): A Brief Primer

Congestion Signals

- Capture resource contention within the network
- Switch buffer occupancy or measured delay

Congestion Response

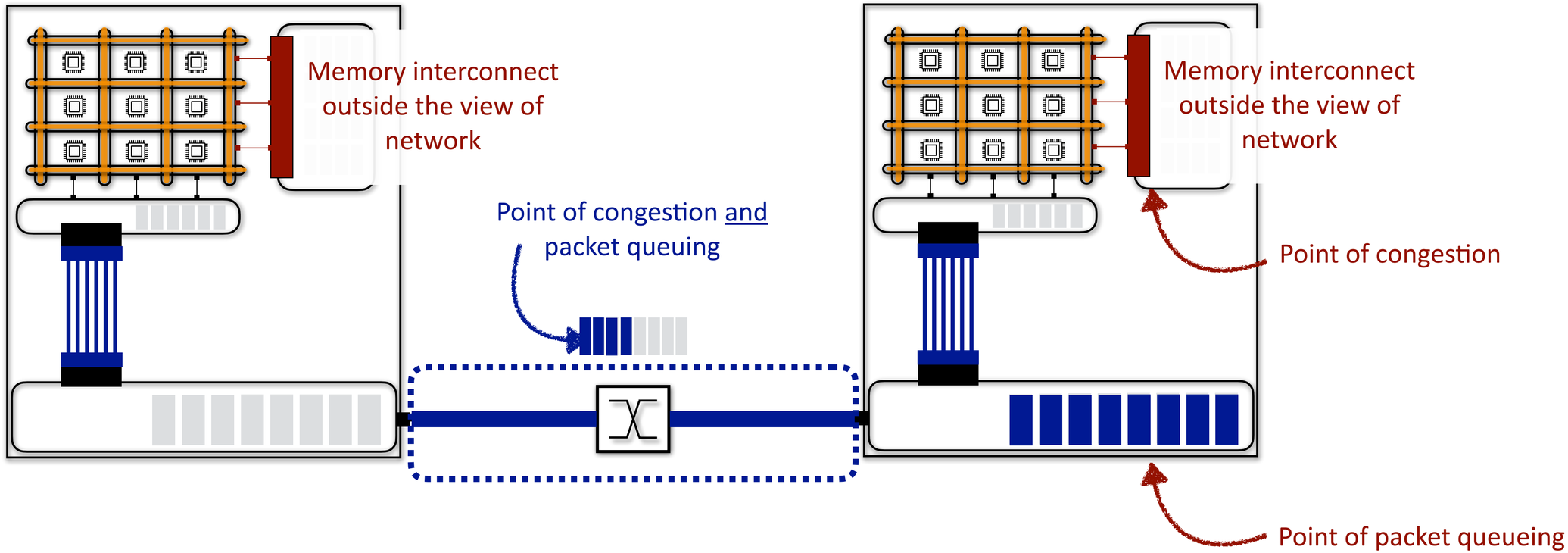
- If no congestion, increase traffic injection rate
- If congestion, reduce traffic injection rate



We Need to Rethink CC Architecture

Rethinking Congestion Signals

- Traditional signals don’t precisely capture host congestion
- Congestion happening “outside” the network



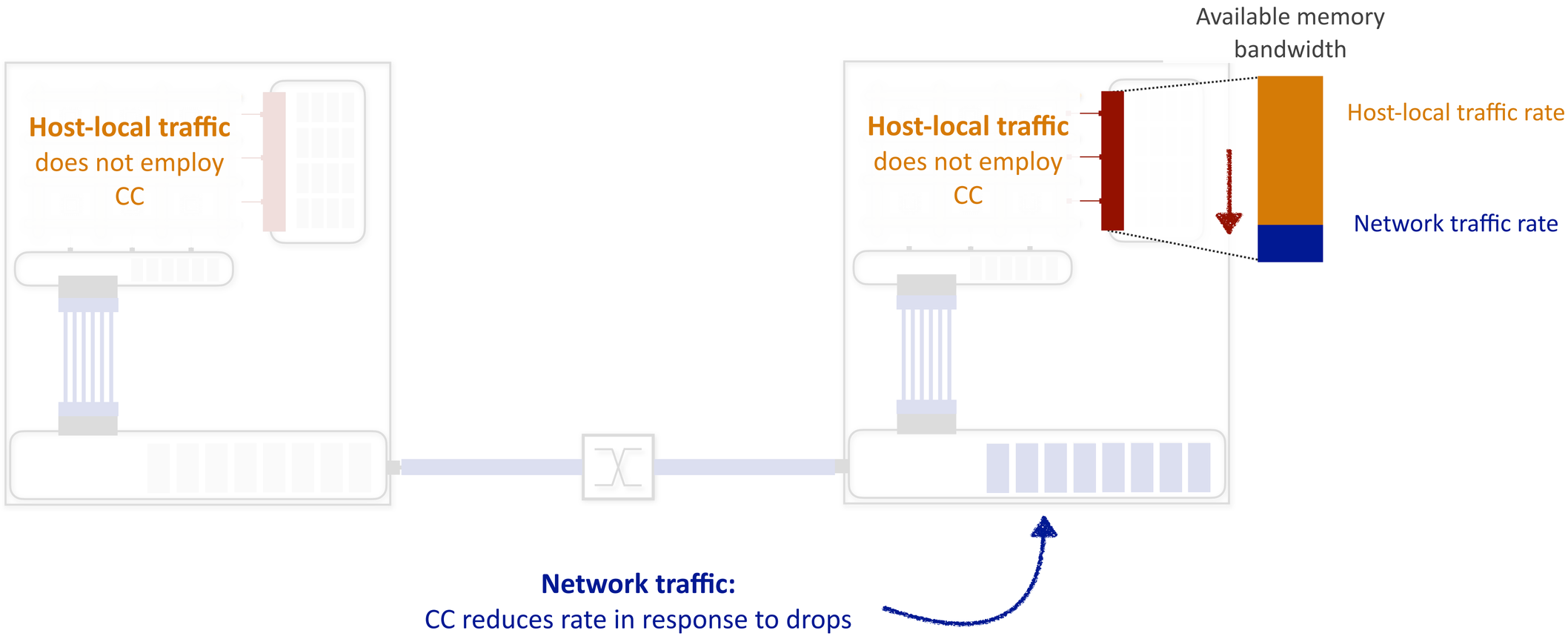
We Need to Rethink CC Architecture

Rethinking Congestion Signals

- Traditional signals don't precisely capture host congestion
- Congestion happening "outside" the network

Rethinking Congestion Response

- Host-local traffic does not employ CC



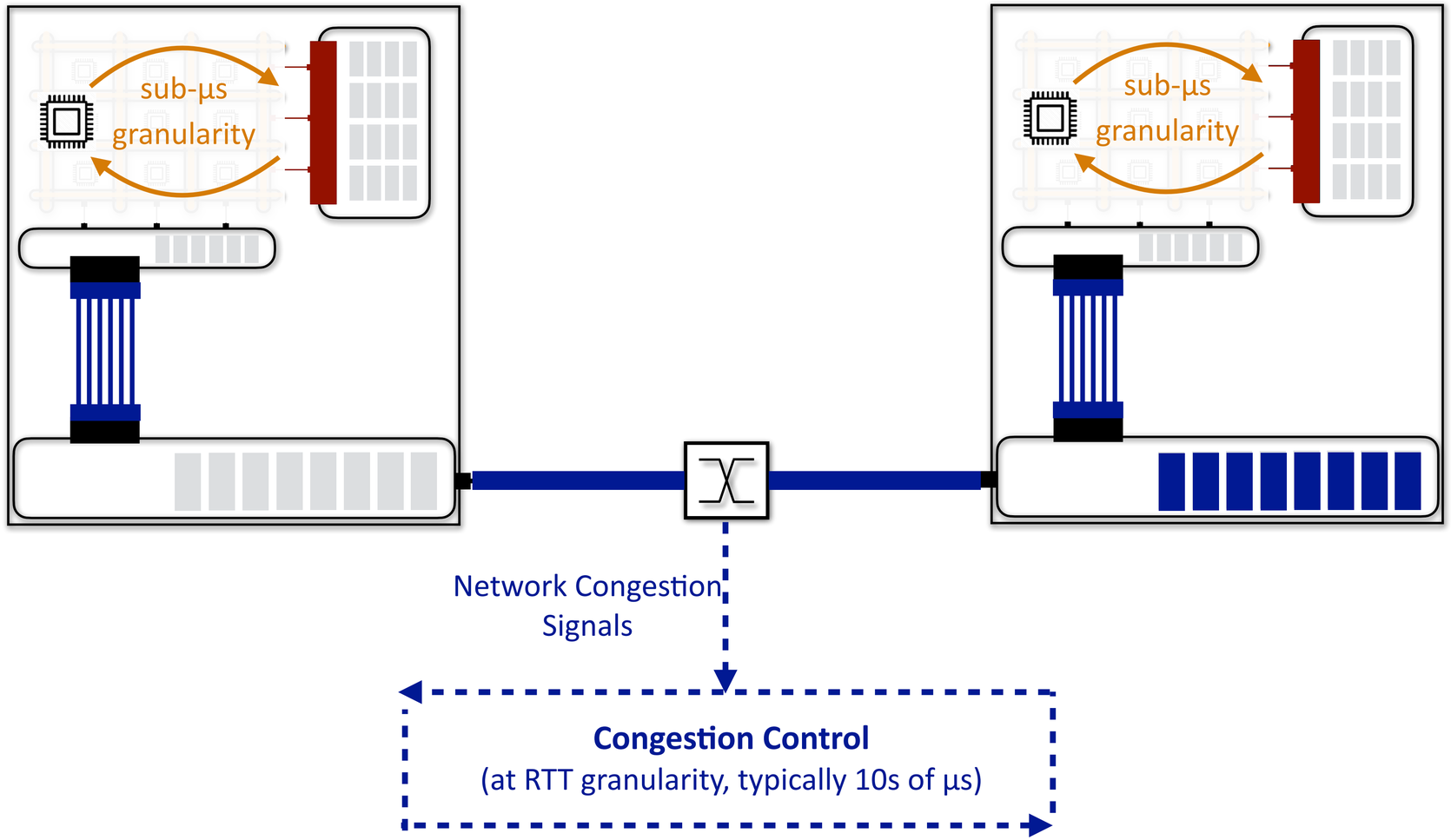
We Need to Rethink CC Architecture

Rethinking Congestion Signals

- Traditional signals don't precisely capture host congestion
- Congestion happening "outside" the network

Rethinking Congestion Response

- Host-local traffic does not employ CC
- Network traffic performs CC at RTT* granularity

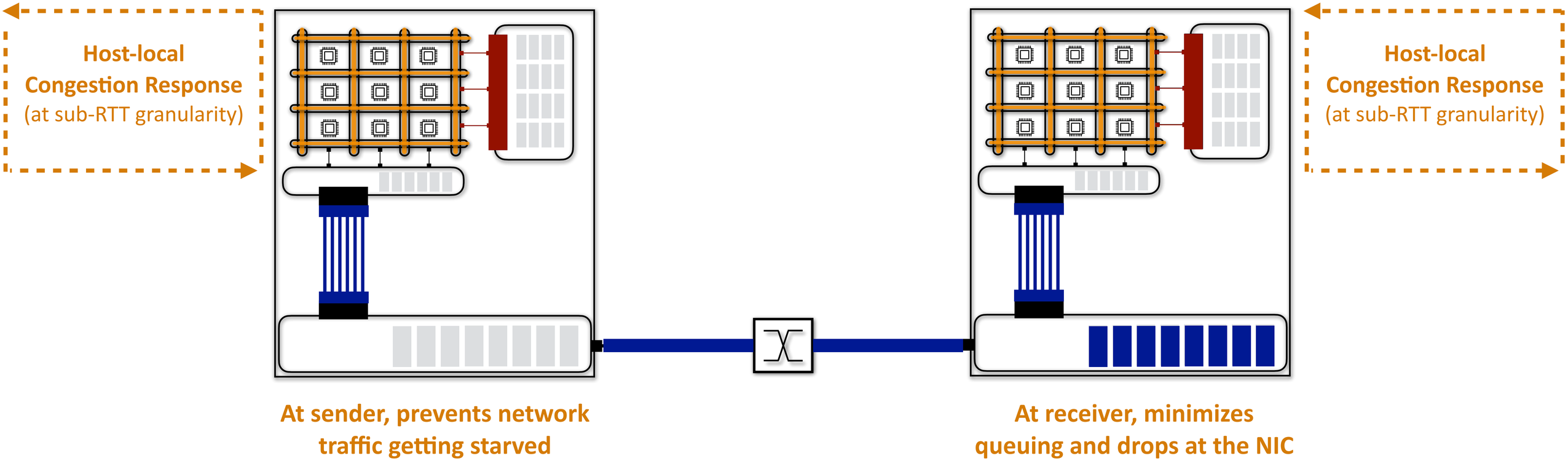


*RTT: Round trip time for a network, typically a few 10s to 100s of microseconds for large-scale datacenter networks

hostCC: A CC Architecture for Host and Network Fabric Congestion

Key Idea: Host-local congestion response, at sub-RTT granularity

- Host network resources allocated between network traffic and host-local traffic
- Coordinated effort between network protocols and OS

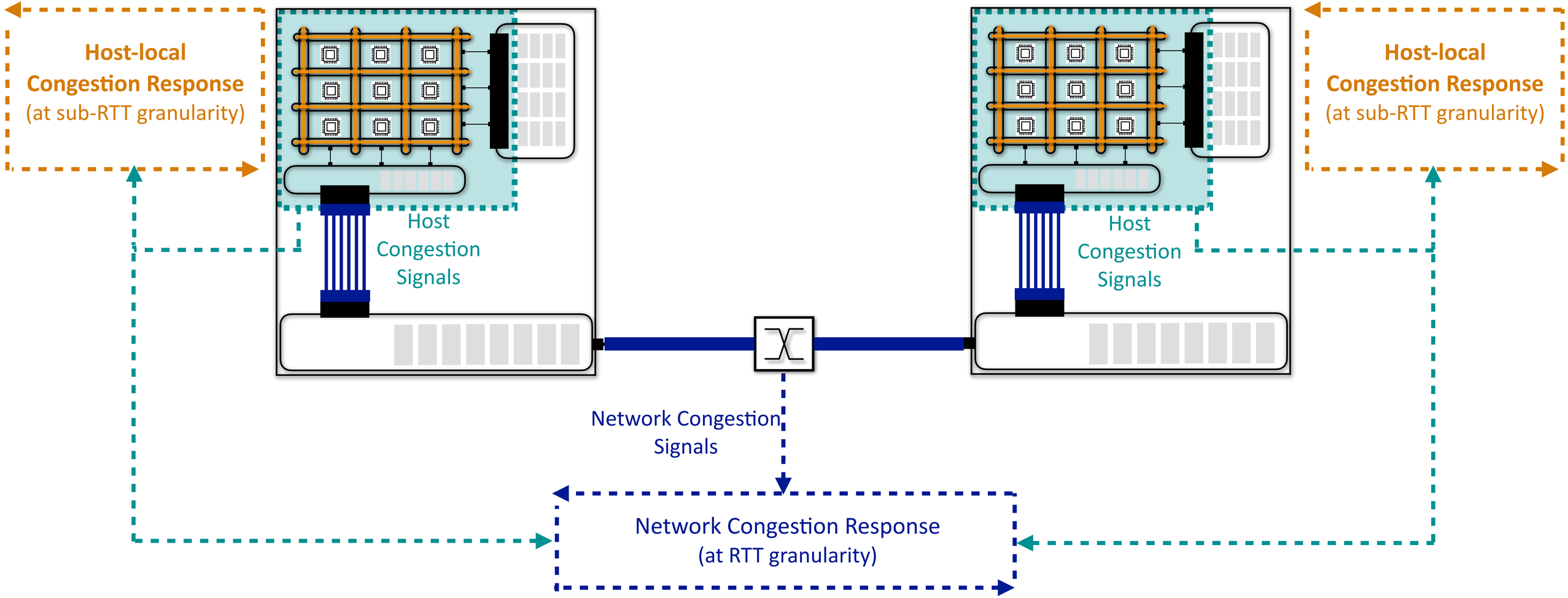


hostCC: End-to-End Overview

1. Host Congestion Signals
Precisely capture host congestion

2. Host-local Congestion Response
Efficient host resource allocation

3. Network Congestion Response
Efficient network resource allocation

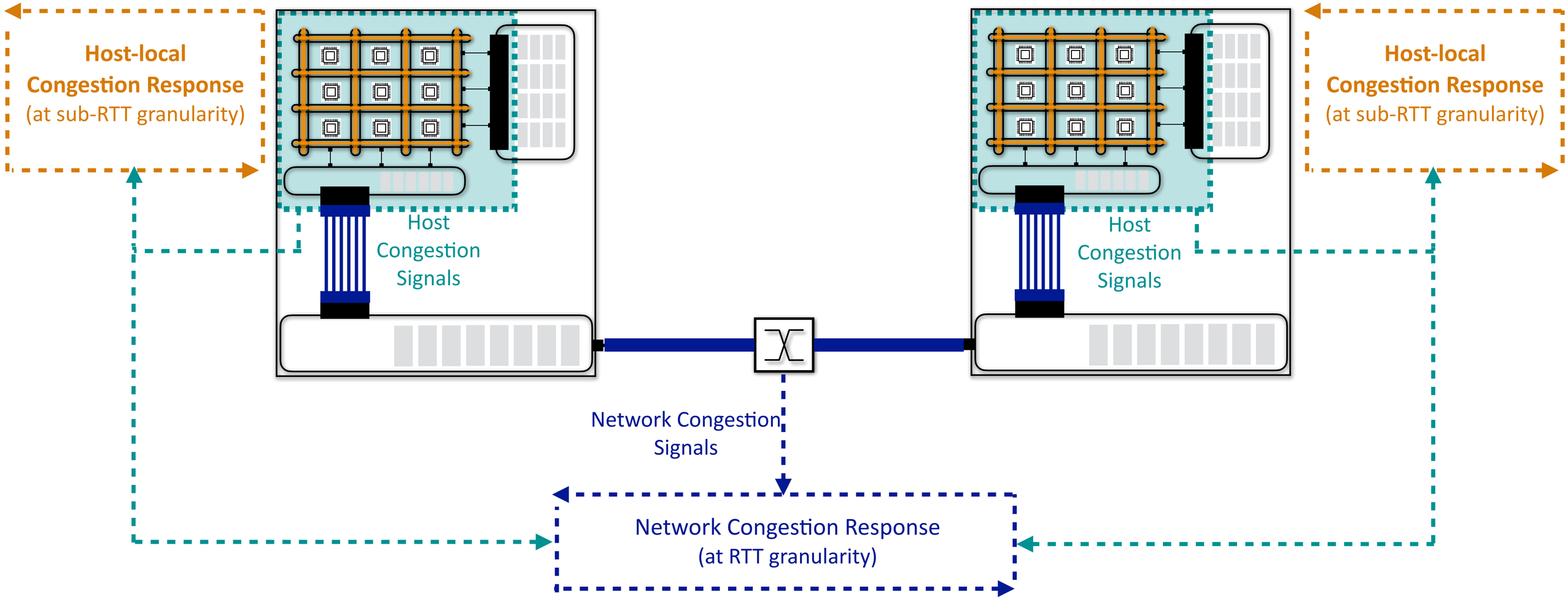


hostCC Details [1/3]: Host Congestion Signals

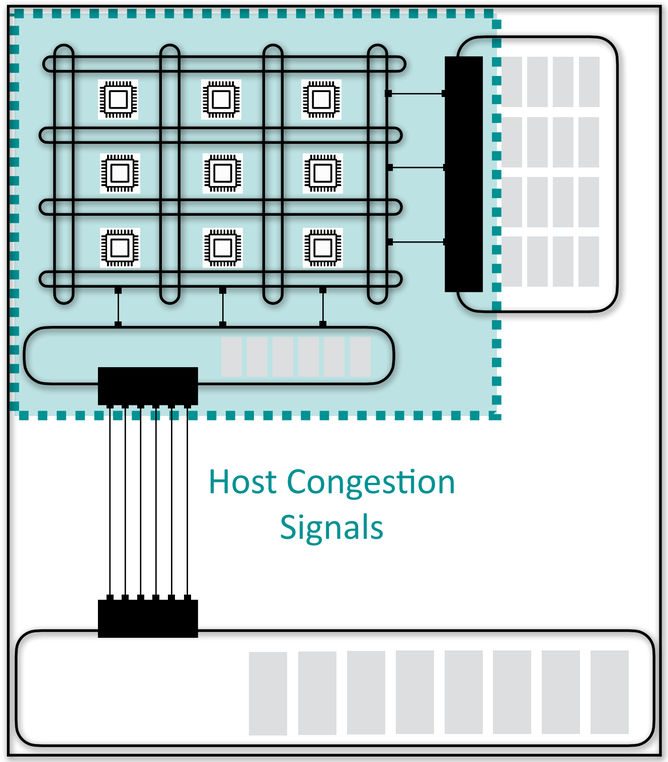
1. Host Congestion Signals
Precisely capture host congestion

2. Host-local Congestion Response
Efficient host resource allocation

3. Network Congestion Response
Efficient network resource allocation



hostCC Details [1/3]: Host Congestion Signals



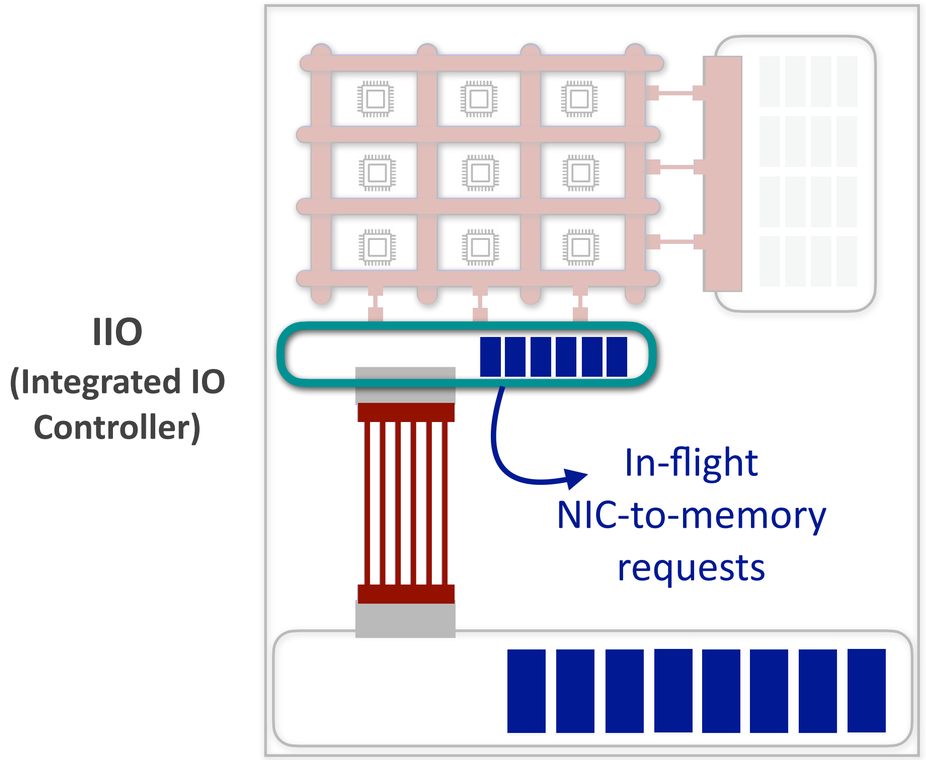
hostCC Details [1/3]: Host Congestion Signals

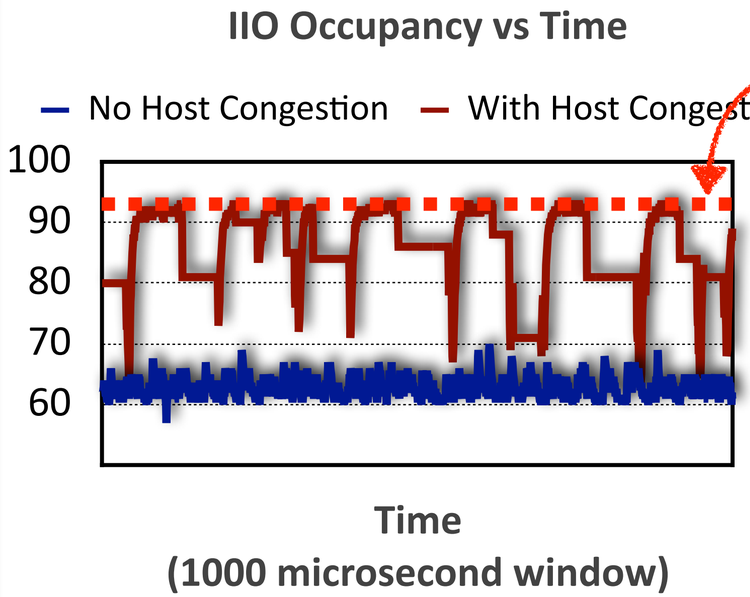
Precisely capturing host congestion using IIO occupancy

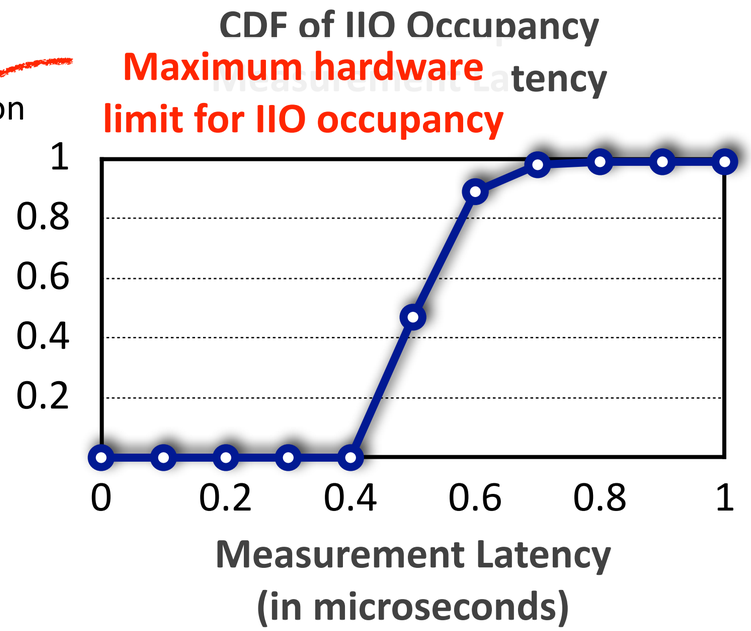
- IIO occupancy: in-flight peripheral-to-memory requests
- Host congestion ⇔ larger IIO Occupancy

Signals host congestion at sub-microsecond scale

- IIO occupancy measured using single register read*
- Measurement latency ~600 nanoseconds







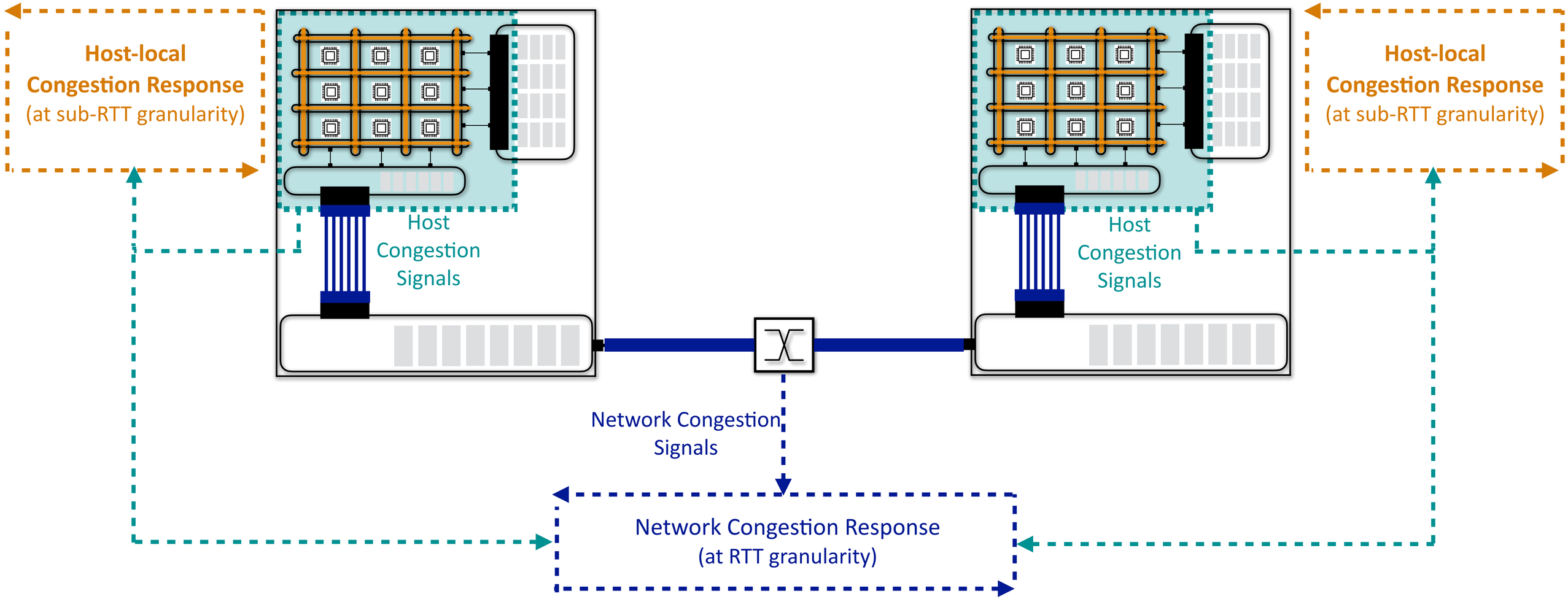
*Using MSRs specified in Intel Performance Monitoring Manual (https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html#uncore)

hostCC Details [2/3]: Host-local Congestion Response at Sub-RTT Granularity

1. Host Congestion Signals
Precisely capture host congestion

2. Host-local Congestion Response
Efficient host resource allocation

3. Network Congestion Response
Efficient network resource allocation



hostCC Details [2/3]: Host-local Congestion Response at Sub-RTT Granularity

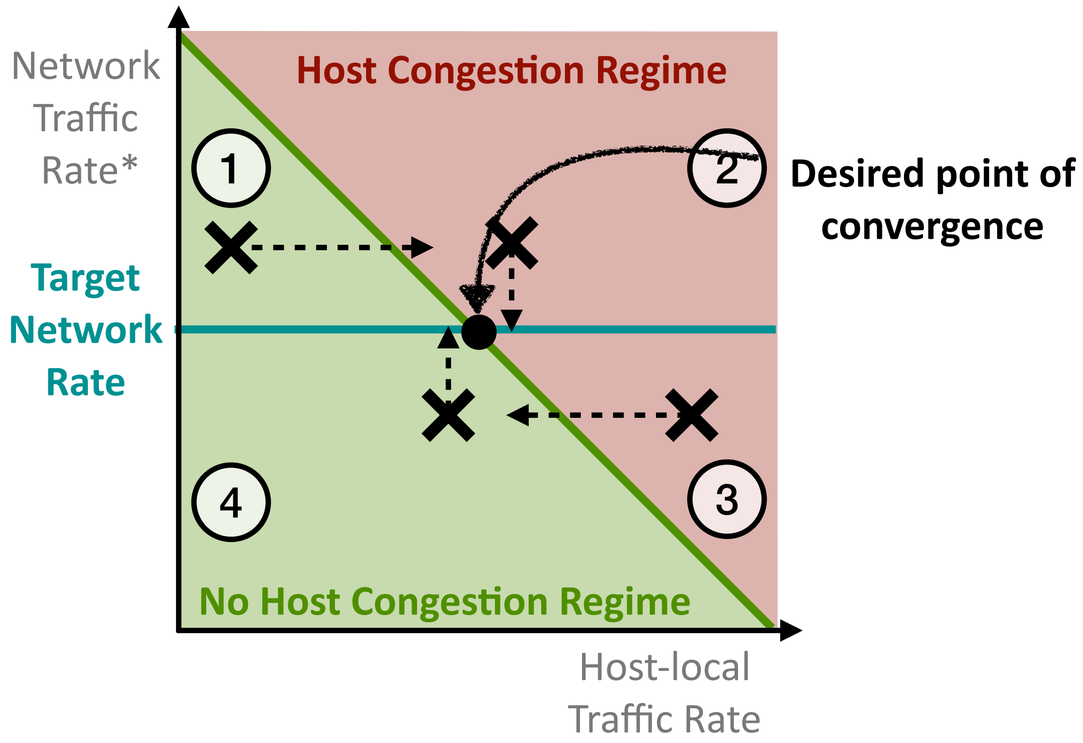
Maximize rate of traffic + minimize host congestion

- Enable user-specified resource allocation policy
- Eg., max-min fairness, prioritization, min guarantee, etc

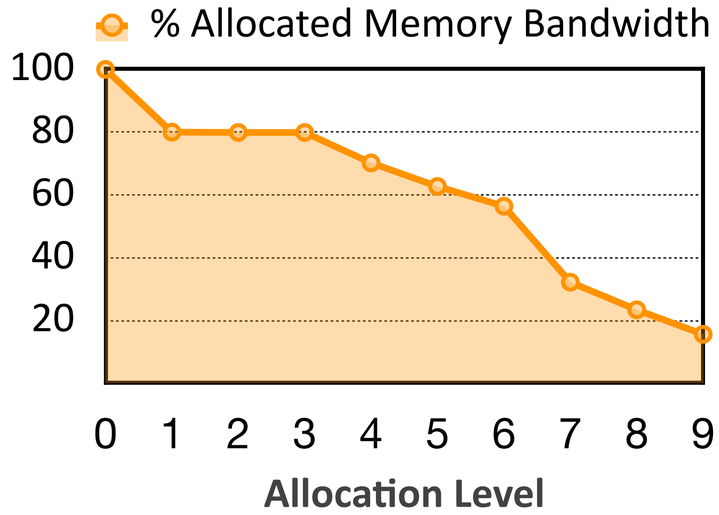
Host resource allocation at sub-RTT granularity

- Using available backpressure-based mechanisms
- Vary host resource allocation via single register writes^

Host-local Congestion Response (at sub-RTT granularity)



Increasing/decreasing host-local traffic rate using Intel Memory Bandwidth Allocation (MBA) tool



Example allocation on Intel Cascade Lake Architecture

MBA performs allocation transparently to applications

Allocation perfomed on a per-Class-of-Service (COS) basis,
COS can be dynamically mapped to one/more cores running arbitrary applications

(1) Increase host-local traffic rate (3) Decrease host-local traffic rate

*PCIe utilization, measured using single a MSR read, similar to IIO occupancy

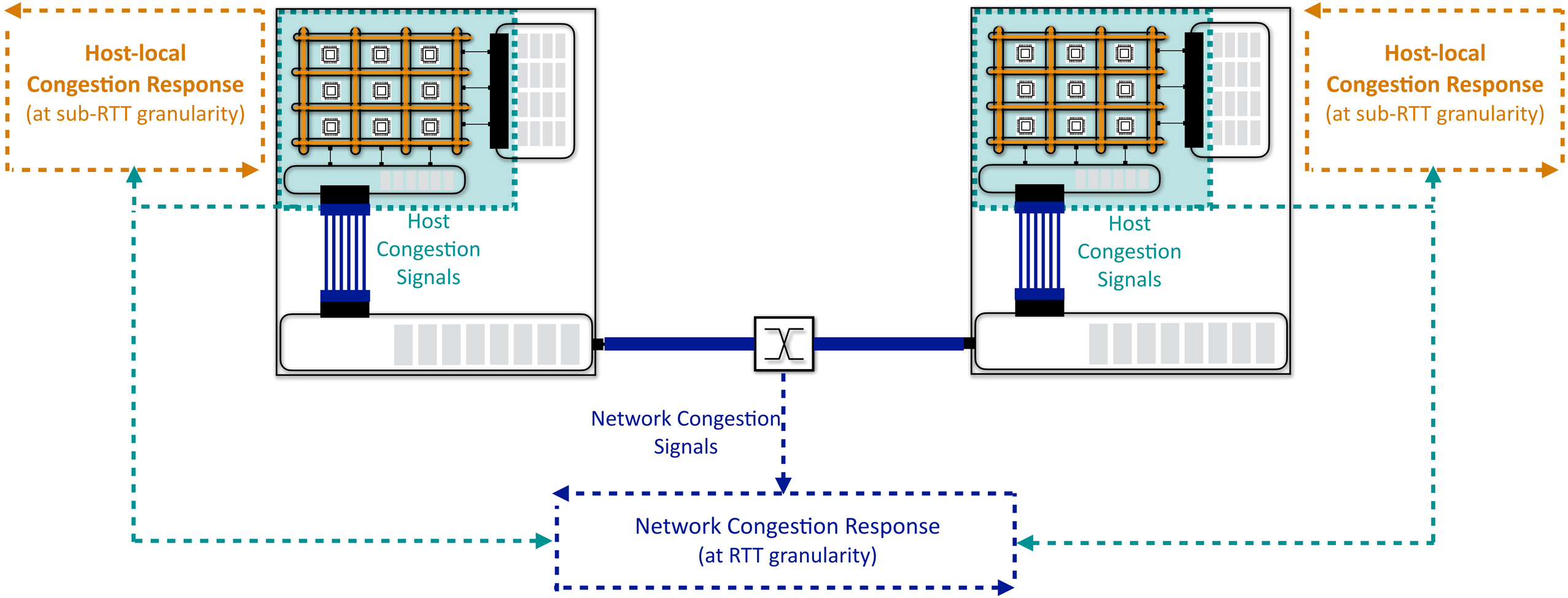
^Using MSRs specified in Intel RDT package (https://github.com/intel/intel-cmt-cat)

hostCC Details [3/3]: Network Congestion Response at RTT Granularity

1. Host Congestion Signals
Precisely capture host congestion

2. Host-local Congestion Response
Efficient host resource allocation

3. Network Congestion Response
Efficient network resource allocation

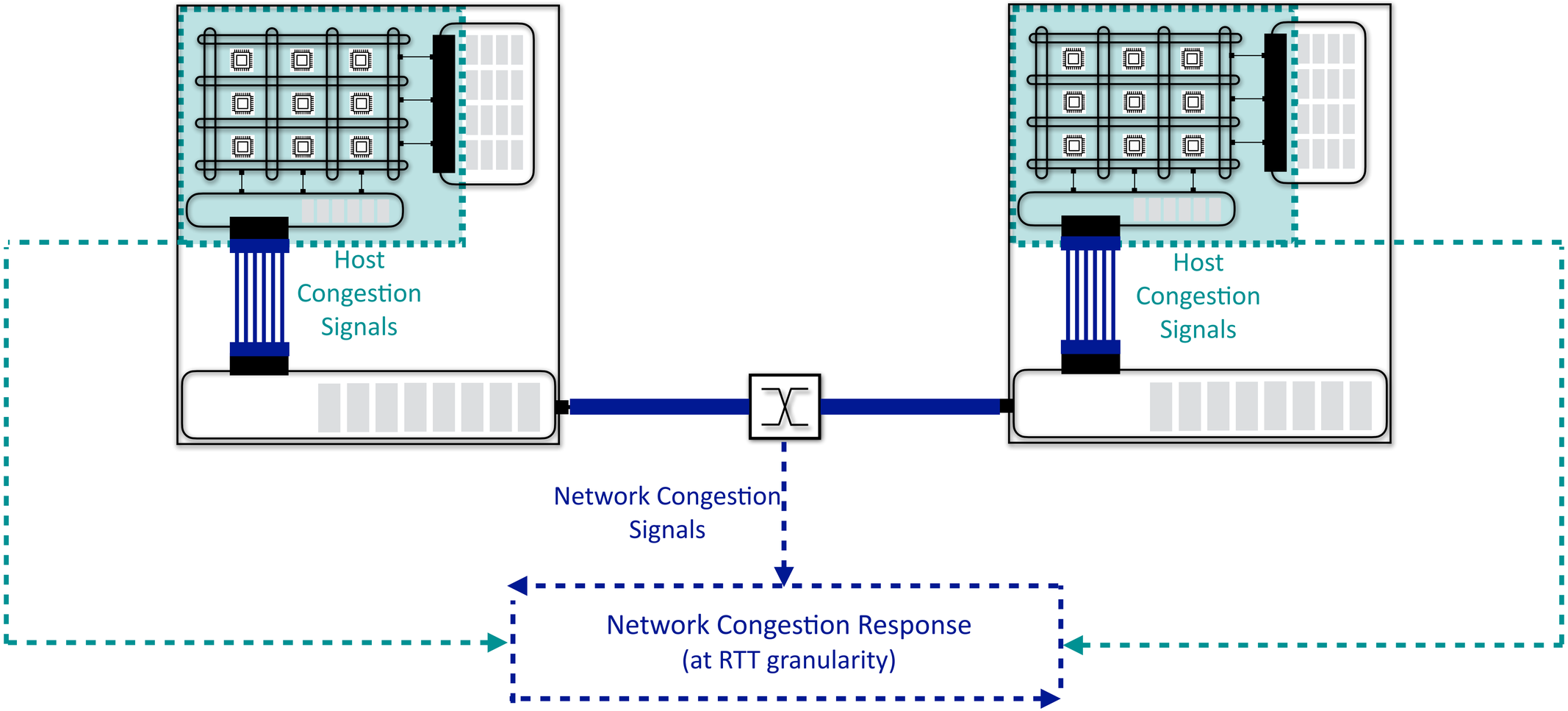


hostCC Details [3/3]: Network Congestion Response at RTT Granularity

Network rate: set using bottlenecks in network & host

- CC Protocols: use network & host congestion signals
- No modifications to existing network CC protocols

Example: ECN (explicit congestion notification) based CC protocols

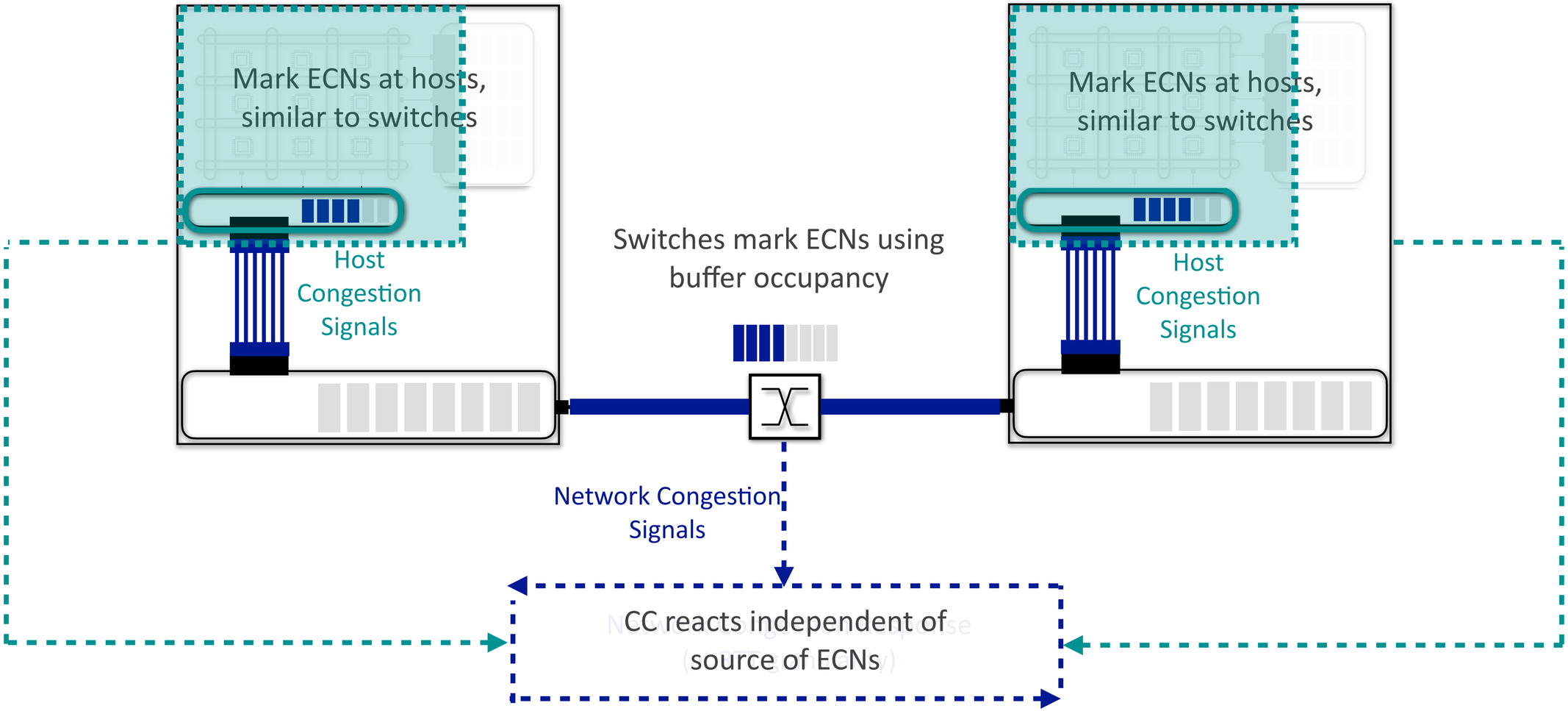


hostCC Details [3/3]: Network Congestion Response at RTT Granularity

Network rate: set using bottlenecks in network & host

- CC Protocols: use network & host congestion signals
- No modifications to existing network CC protocols

Example: ECN (explicit congestion notification) based CC protocols



hostCC Details [3/3]: Network Congestion Response at RTT Granularity

Network rate: set using bottlenecks in network & host

- CC Protocols: use network & host congestion signals
- No modifications to existing network CC protocols

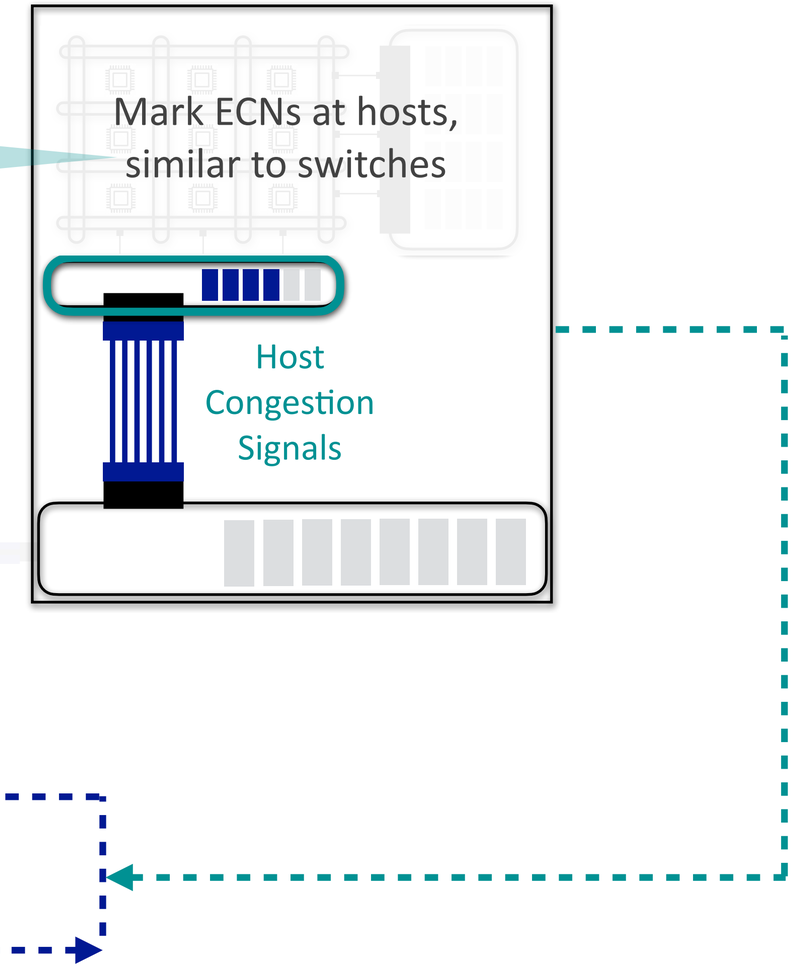
Example: ECN (explicit congestion notification) based CC protocols

Utilizing Linux NetFilter* subsystem to mark ECNs

NetFilter hooks triggerred during IP layer processing

Network stack runs following extra logic when processing each IP header

```
If (IIO Occupancy > Threshold)          // host congestion detected
 (ip_hdr->tos = ip_hdr->tos | 0x03) // Mark ECN bits in IP header
```



For transport layer, indistinguishable whether ECN marked at switch or host

CC reacts independent of source of ECNs

*Typically used for IP-based packet filtering, mangling, network address translation, etc

hostCC Details [3/3]: Network Congestion Response at RTT Granularity

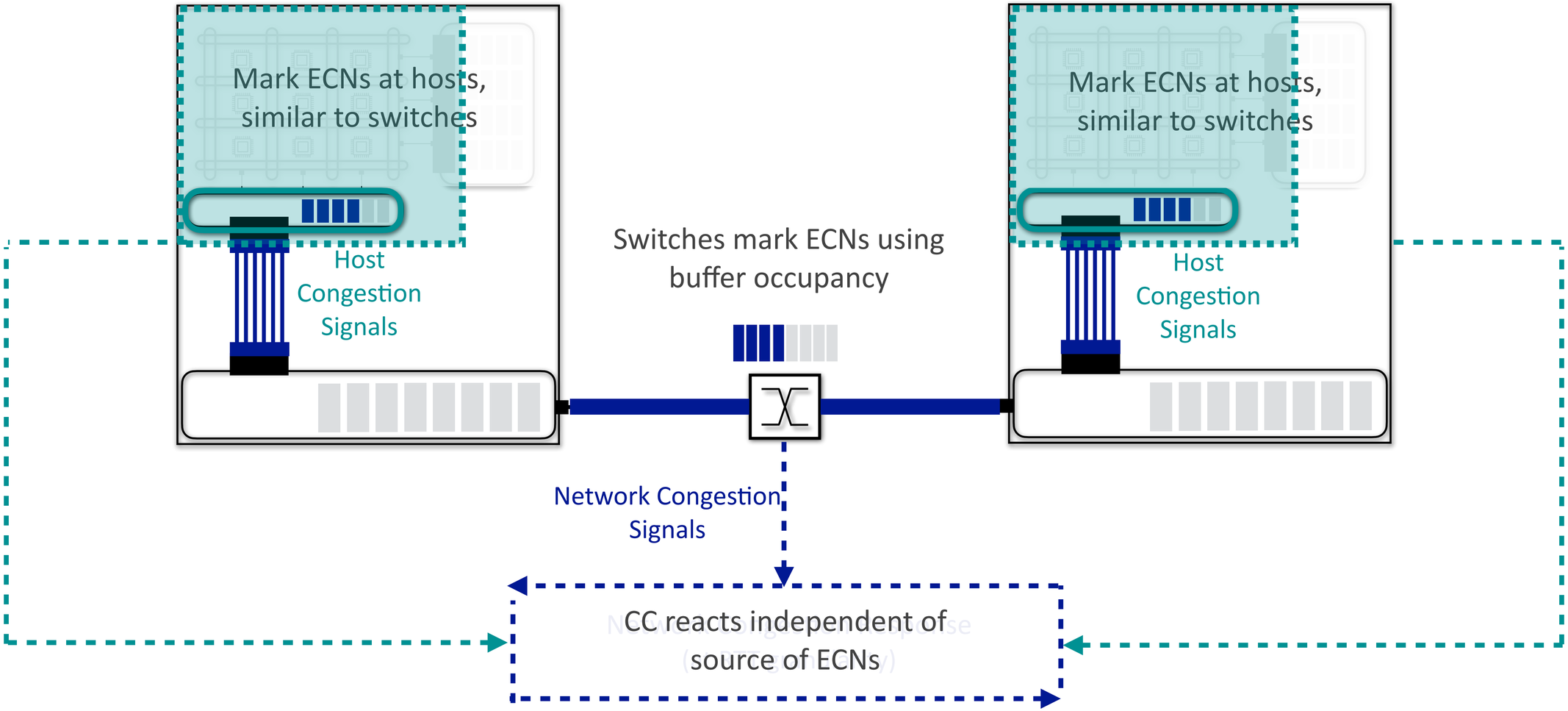
Network rate: set using bottlenecks in network & host

- CC Protocols: use network & host congestion signals
- No modifications to existing network CC protocols

RTT granularity network response sufficient

- Host-local response handles sub-RTT level changes
- + Gives network CC time to converge without drops

Example: ECN (explicit congestion notification) based CC protocols

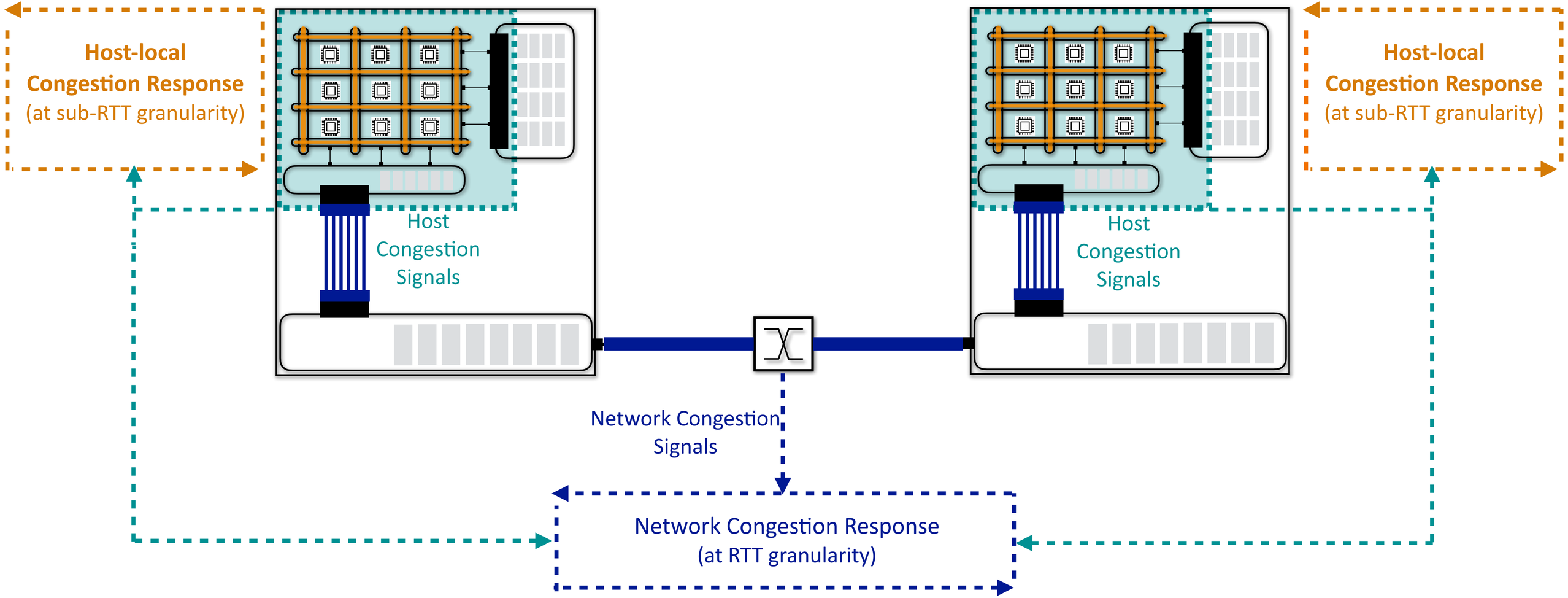


hostCC: A New CC Architecture to Handle Host and Network Fabric Congestion

1. Host Congestion Signals
Precisely capture host congestion

2. Host-local Congestion Response
User-specified host resource allocation

3. Network Congestion Response
Efficient network resource allocation



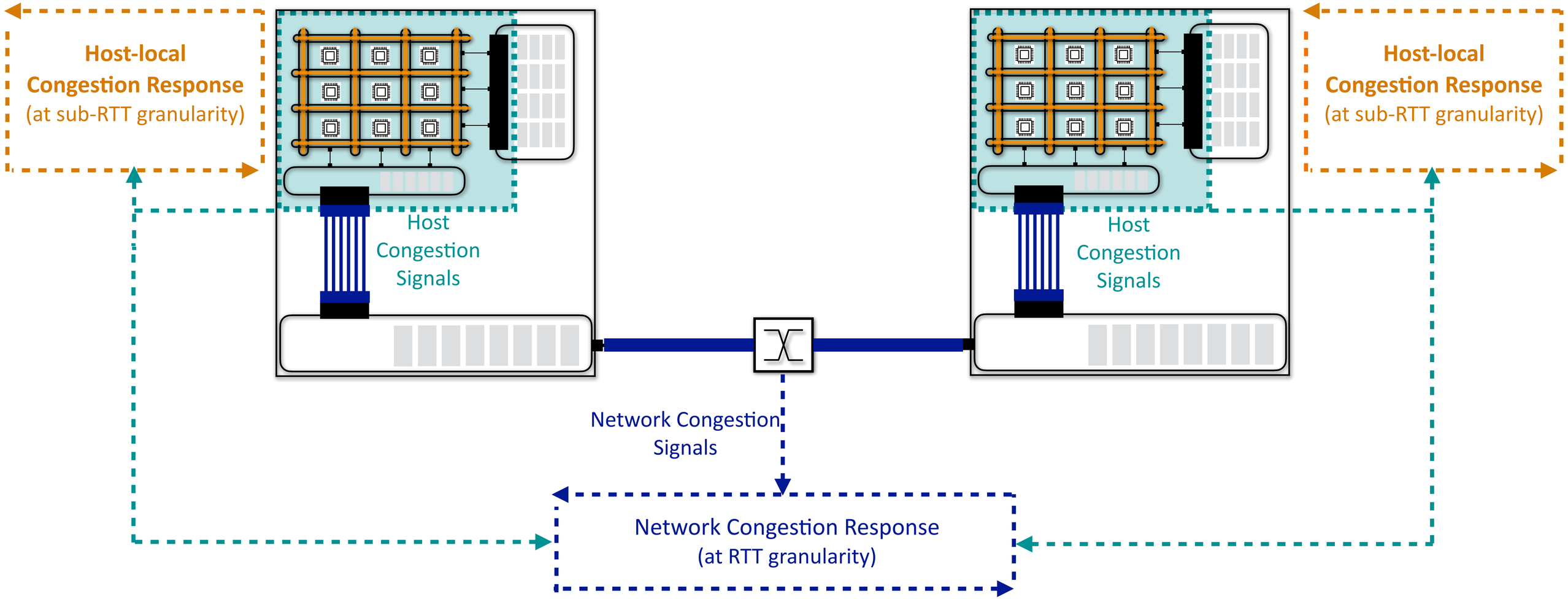
hostCC: End-to-End Implementation

Implemented within Linux kernel

- Prototyped as a loadable module; <~800LOC
- Makes minimal modifications to the network stack

Requires no changes to apps, host HW & network HW

- Supports Intel Cascadelake (2019) and newer servers
- + Supports kernel optimizations like TSO, GRO & aRFS



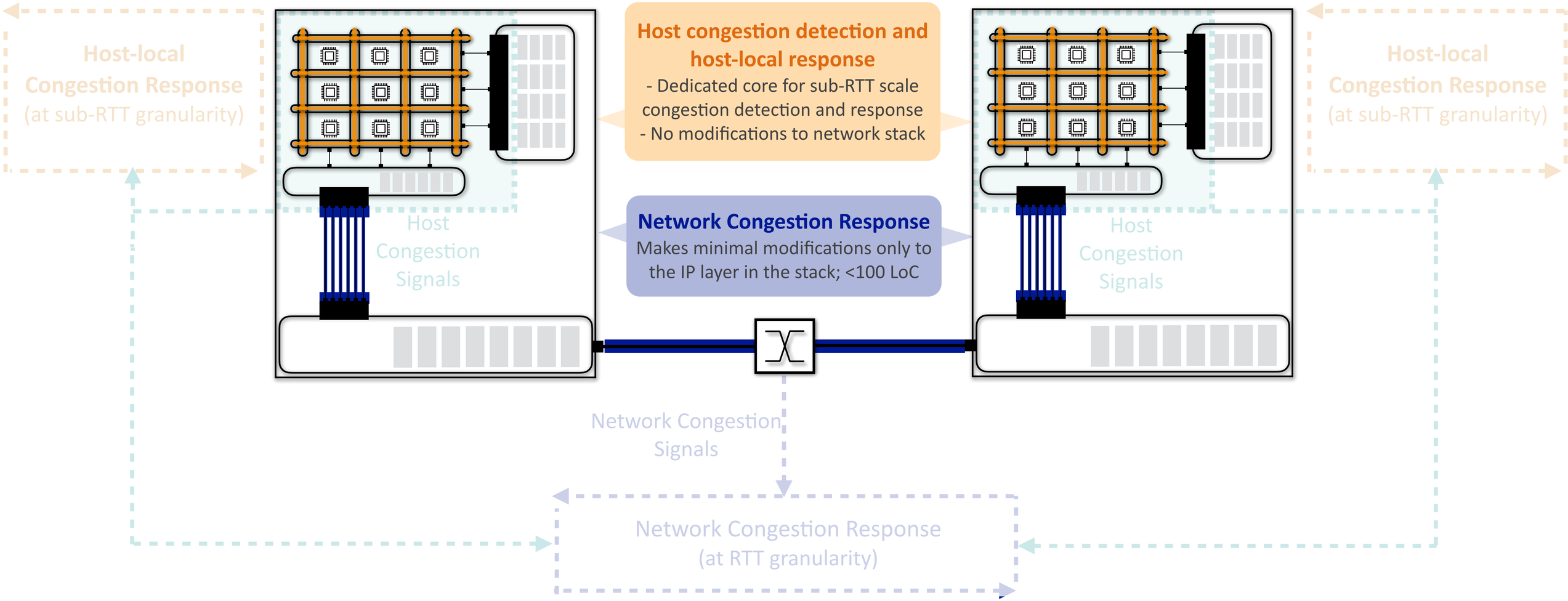
hostCC: End-to-End Implementation

Implemented within Linux kernel

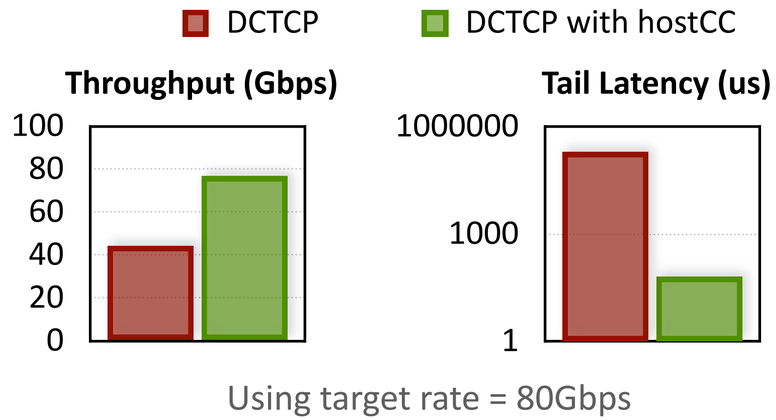
- Prototyped as a loadable module; <~800LOC
- Makes minimal modifications to the network stack

Requires no changes to apps, host HW & network HW

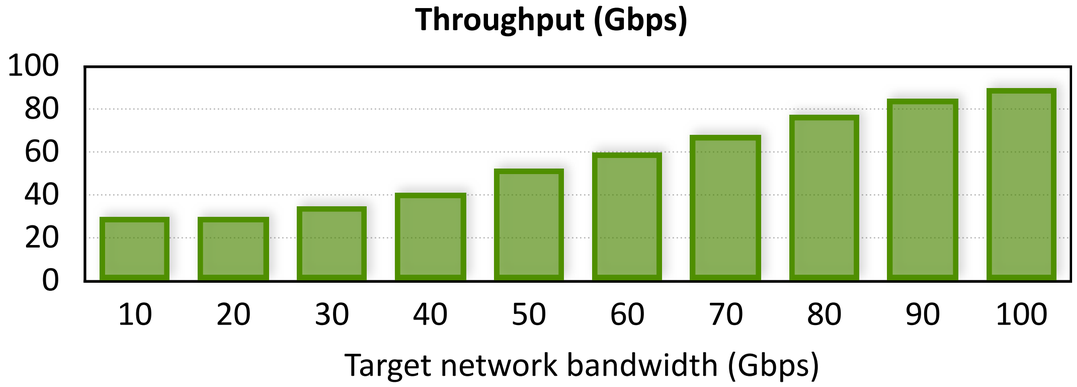
- Supports Intel Cascadelake (2019) and newer servers
- + Supports kernel optimizations like TSO, GRO & aRFS



hostCC Benefits Under Host Congestion



Using target rate = 80Gbps



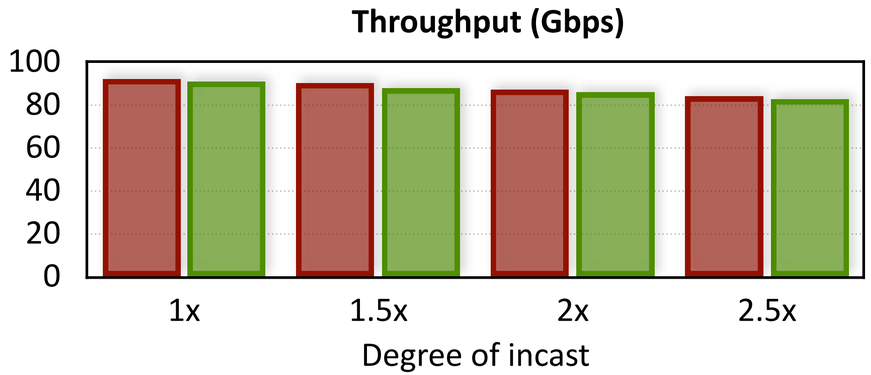
Near-optimal throughput and latency

- Reduces queueing and/or drops to a bare minimum

Enables enforcing desired allocation policy

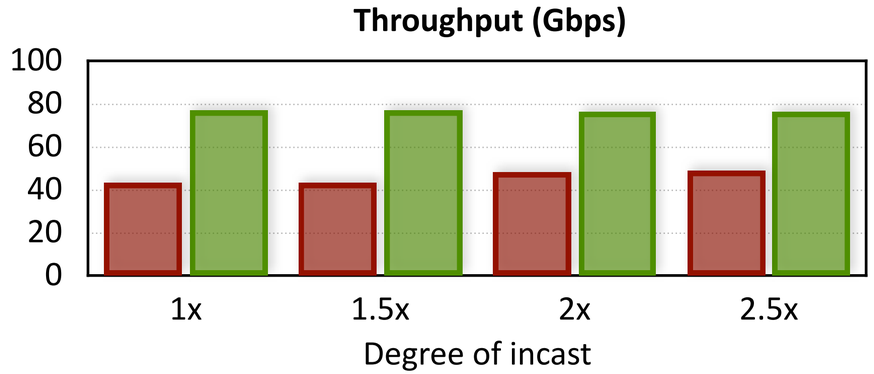
- Achieves close to desired user-specified target rates

hostCC Benefits Under Host Congestion + Network Fabric Congestion



Performance similar to network CC, in presence of only network fabric congestion

- Minimal overheads of using hostCC

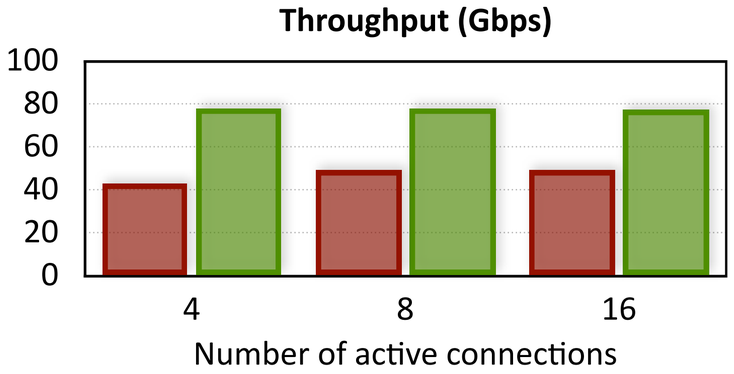


Maintains benefits even in presence of both host and network fabric congestion

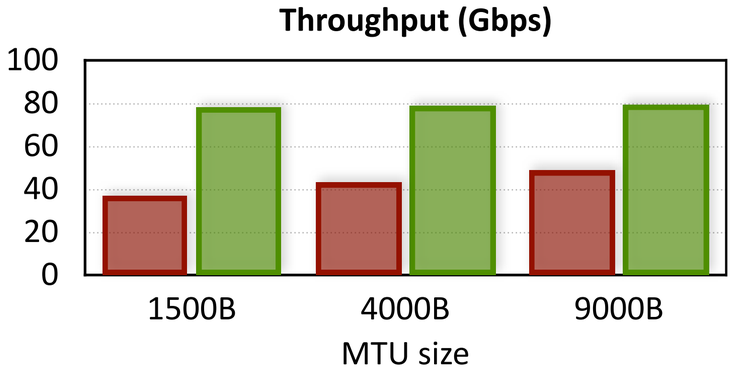
- Interpolates well with network CC

Setup: used **2 sender and 1 receiver machines** to induce network fabric congestion
(Larger degree of congestion created by increasing #concurrent connections)

hostCC Maintains its Benefits with Varying Workload and Network Settings



Varying number of active connections



Varying MTU sizes

Lessons Learnt, and Potential Design Opportunities

Designing precise, timely & CPU-efficient host congestion signals

- Perhaps, hardware redesign to provide interrupts to signal host congestion events?
- (Instead of busy polling hardware counters in current prototype)

Designing mechanisms for finer-grained host-local response

- Existing mechanisms like MBA only provide few, discrete, bandwidth allocation levels
- Can be quite coarse-grained (several 10s of Gbps), especially with today's high bandwidth servers

Extending support to delay-based CC protocols

- Like BBR, TCP Vegas, etc
- Potentially using IIO-to-DRAM latency using available hardware counters (see hostCC paper for details)

Extending support to hardware-offloaded network stacks

- Eg., RDMA
- New challenge: larger latency b/w point of host congestion (near CPU memory) and CC implementation (RDMA NIC)

Handling host congestion in complex, scale-up network topologies

- Eg., contention at PCIe switches/ NVswitches, in addition to the memory interconnect
- Might need hardware support to signal congestion (like ECNs) at these switches

Check out our papers for more details…

Understanding Host Interconnect Congestion, HotNets'22

- Characterization of host congestion phenomenon in Google's production clusters

Host Congestion Control, SIGCOMM'23

- A new CC architecture to handle host congestion, along with network fabric congestion

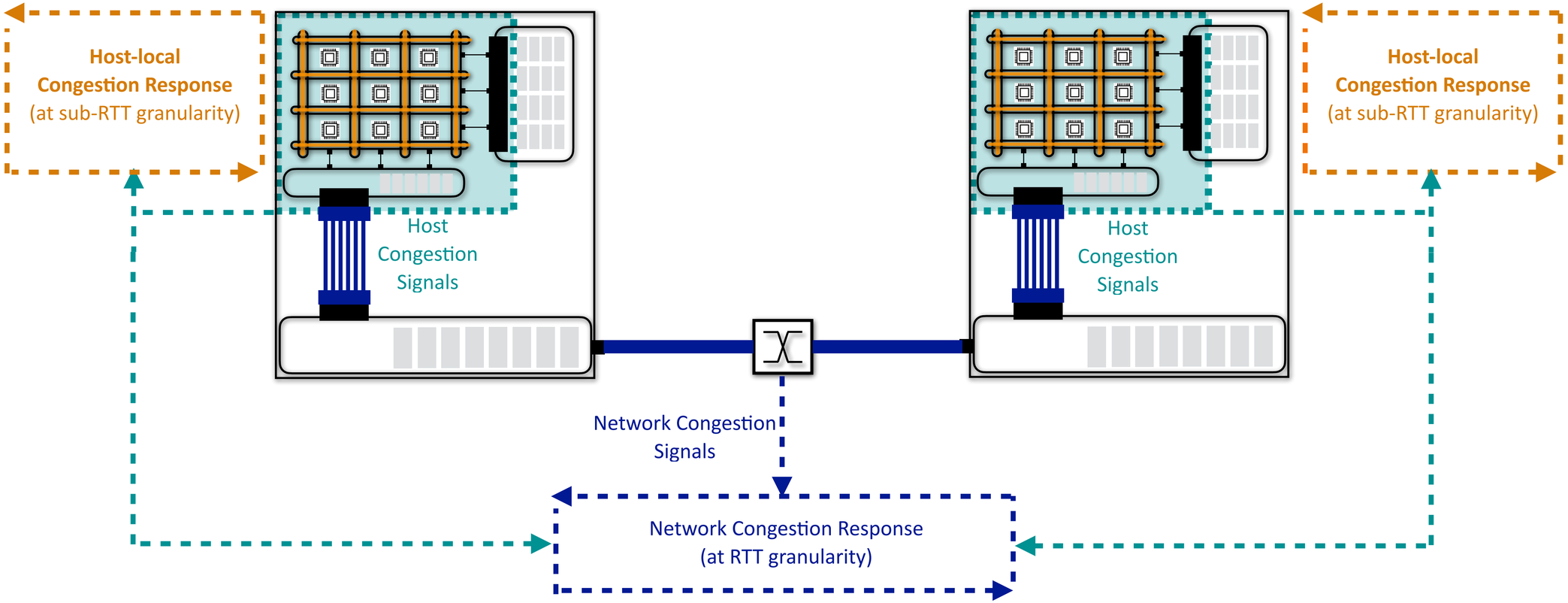
Understanding the Host Network, SIGCOMM'24 [Best Student Paper Award]

- New abstractions to capture intricate interactions within host network interconnects

Fast & Safe IO Memory Protection, SOSP'24

- Rethinking IO memory protection mechanisms to handle host congestion

hostCC: A New CC Architecture to Handle Host and Network Fabric Congestion



hostCC Linux implementation & workloads to reproduce our results are available at **www.github.com/Terabit-Ethernet/hostCC**

hostCC project webpage: **saksham.web.illinois.edu/hostcc**