An Introduction to Smart NICs and their use cases

Mina Tahmasbi Arashloo
Cornell University

Fall 2019

What is a smart NIC?

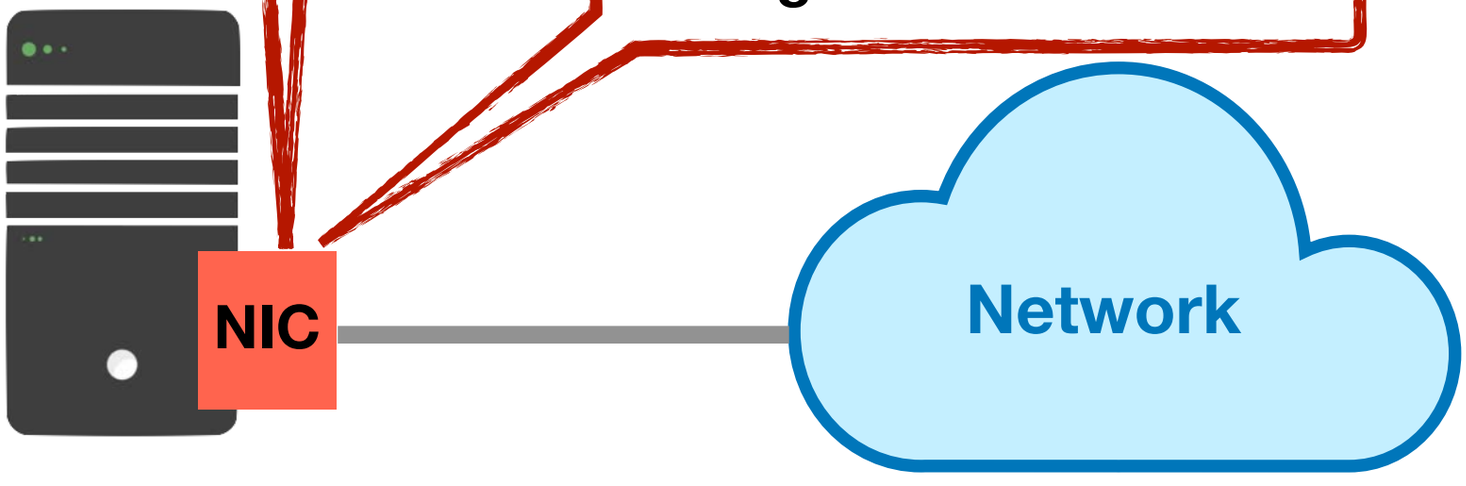
What is a (dumb) NIC?

What is a (dumb) NIC?

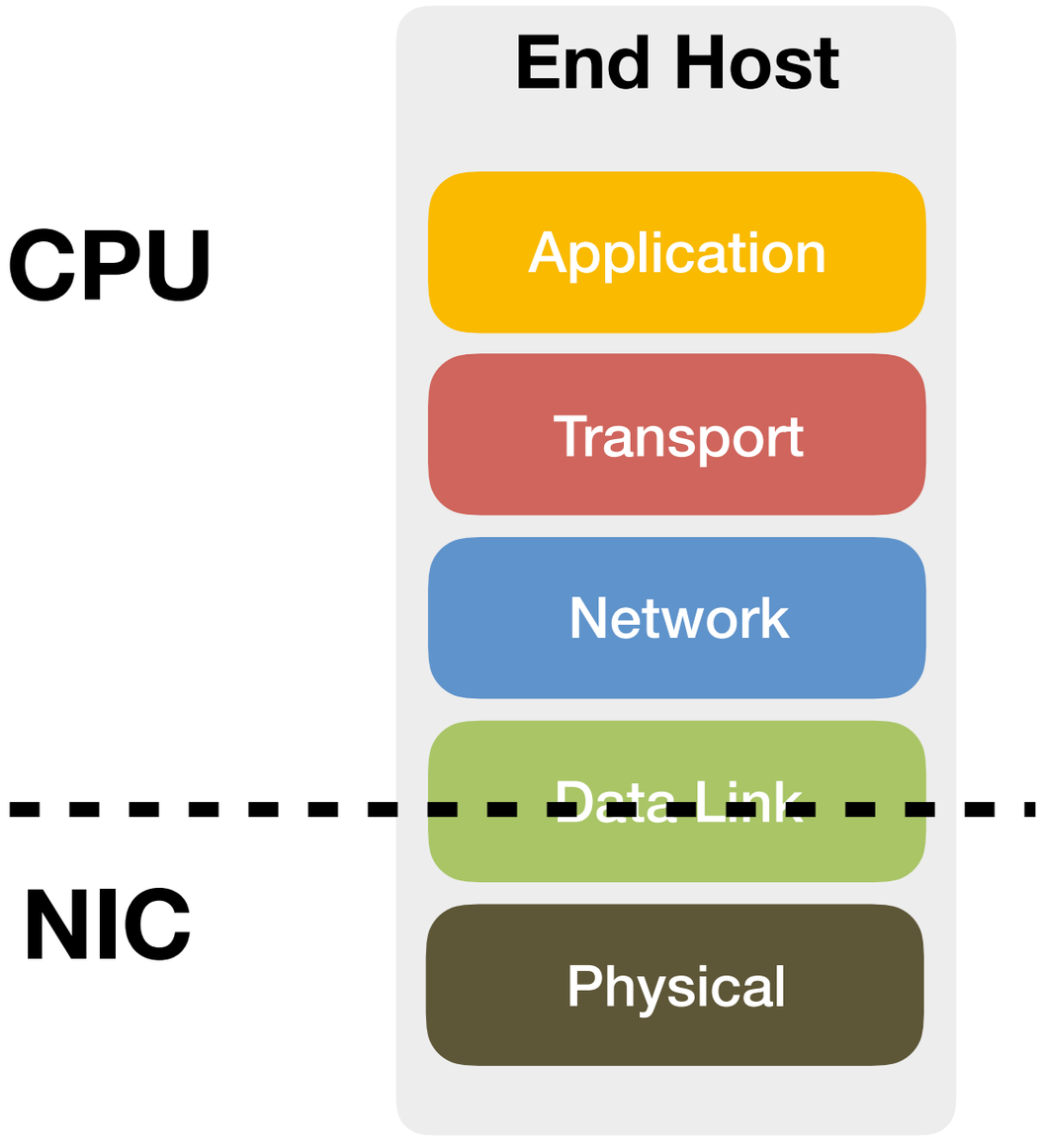
Network Interface Card

Implements:
- the physical layer (L1)
- (part of) the data link layer (L2)

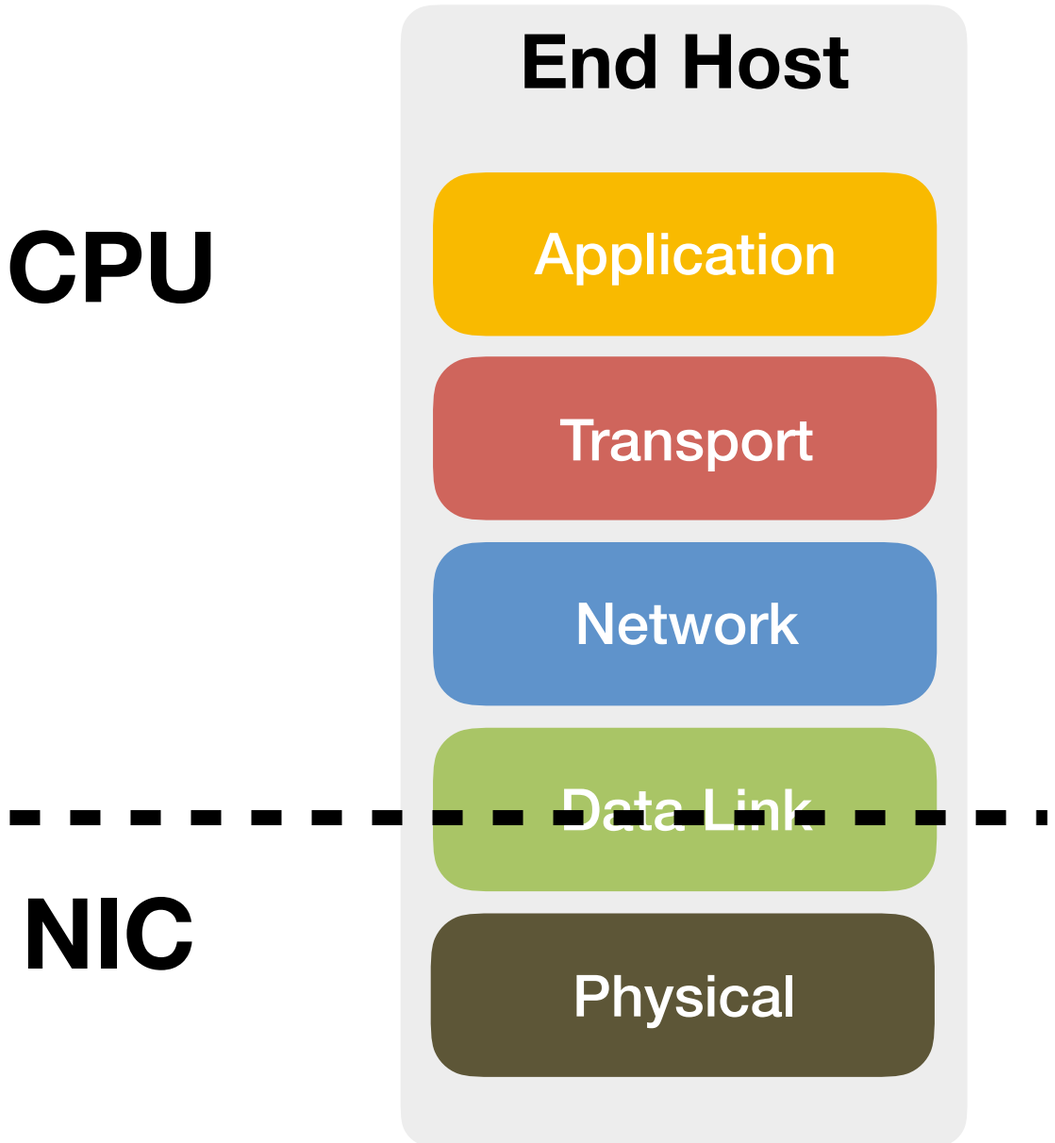
Any packet from the end host to the network and vice versa goes through the NIC



What is a (dumb) NIC?



What is a (dumb) NIC?

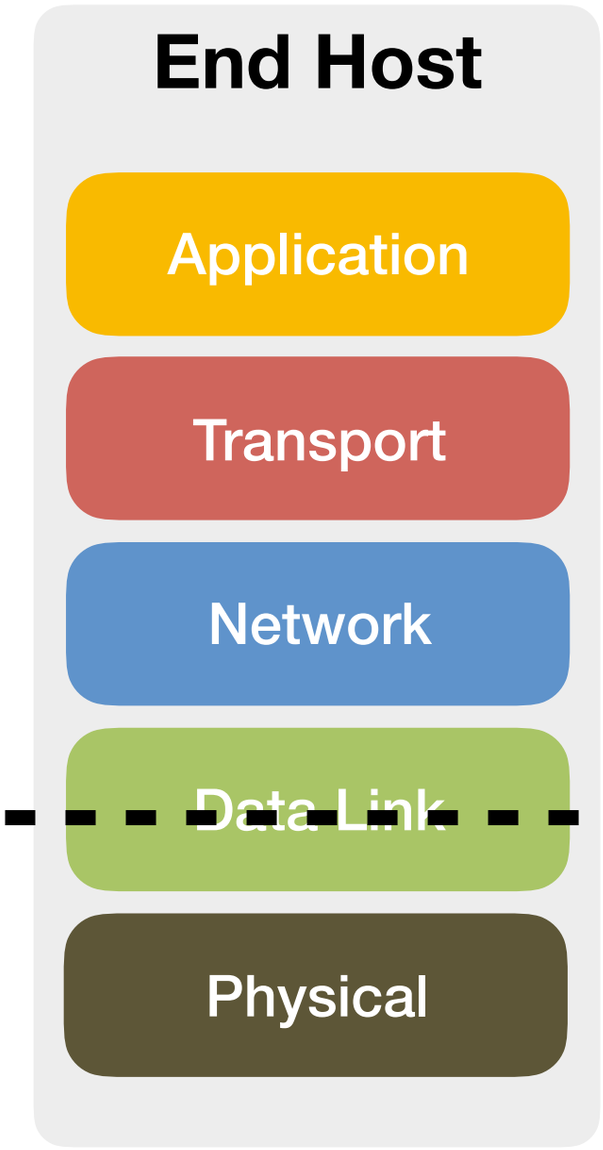


On transmit (egress)

- The host CPU generates packets on application request
- Packets are sent to the NIC over PCIe
- The NIC transforms packets to bits and sends them over the link

What is a (dumb) NIC?

CPU



NIC

On receive (ingress)

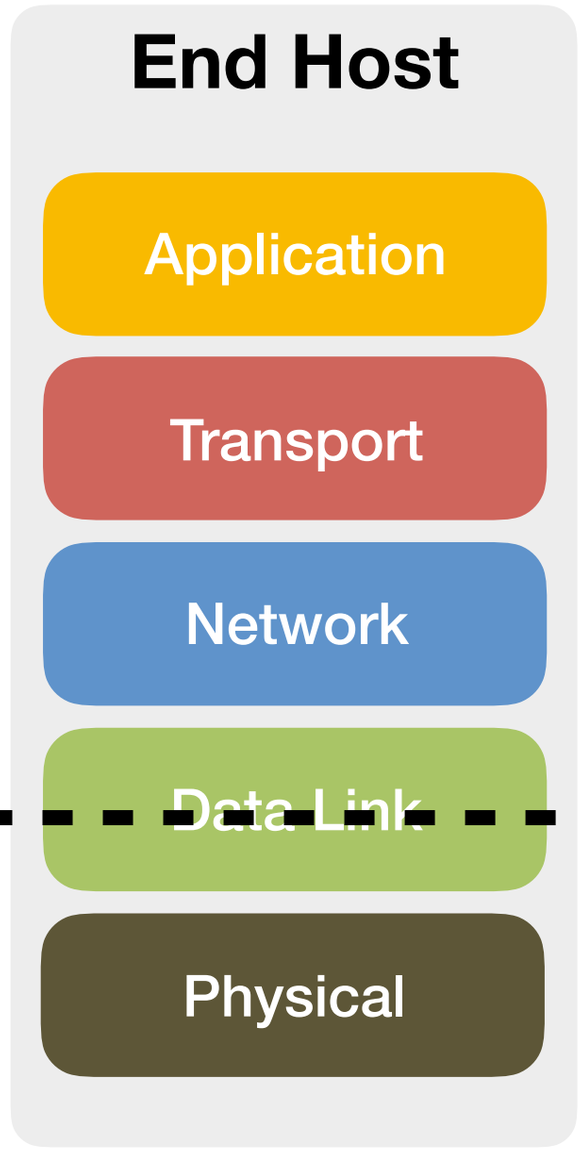
- The NIC turns bits into packets
- Packets are sent to the host CPU over PCIe
- The host CPU processes packets and delivers them to applications

What is a (dumb) NIC?

Great division of labor!

CPU

general-purpose processor running software



NIC

fixed-function hardware

2

- Complicated
- Changes frequently

3

CPU could keep up with the NIC, i.e, process packets at line rate in reasonable #cycles

1

- Simple
- Does not change often

What is a (dumb) NIC?

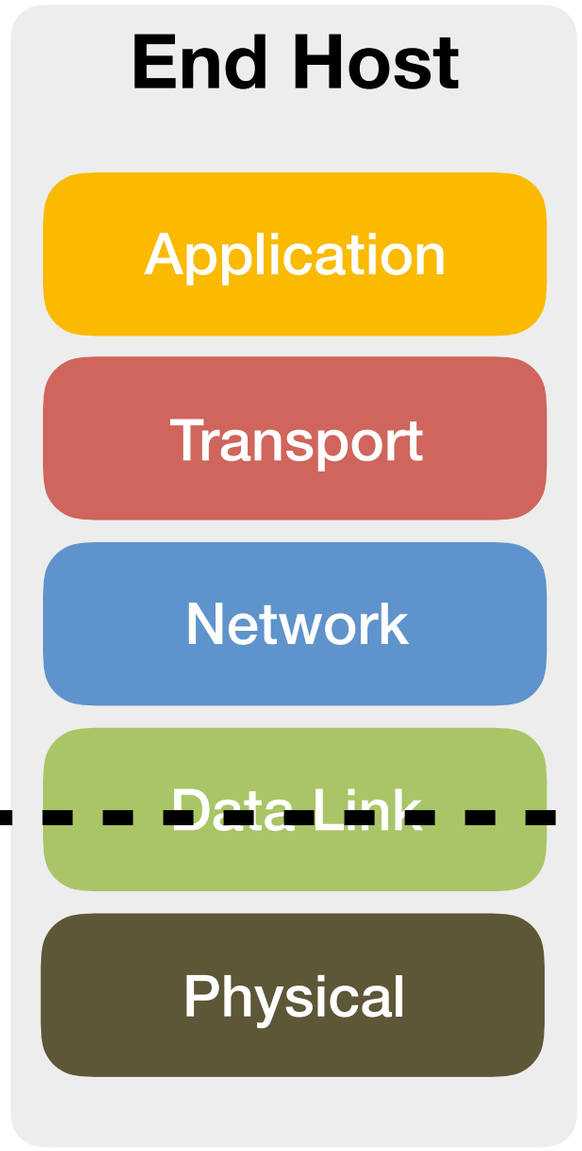
Not so great anymore

CPU

general-purpose processor running software

NIC

fixed-function hardware



2

- Complicated
- Changes frequently

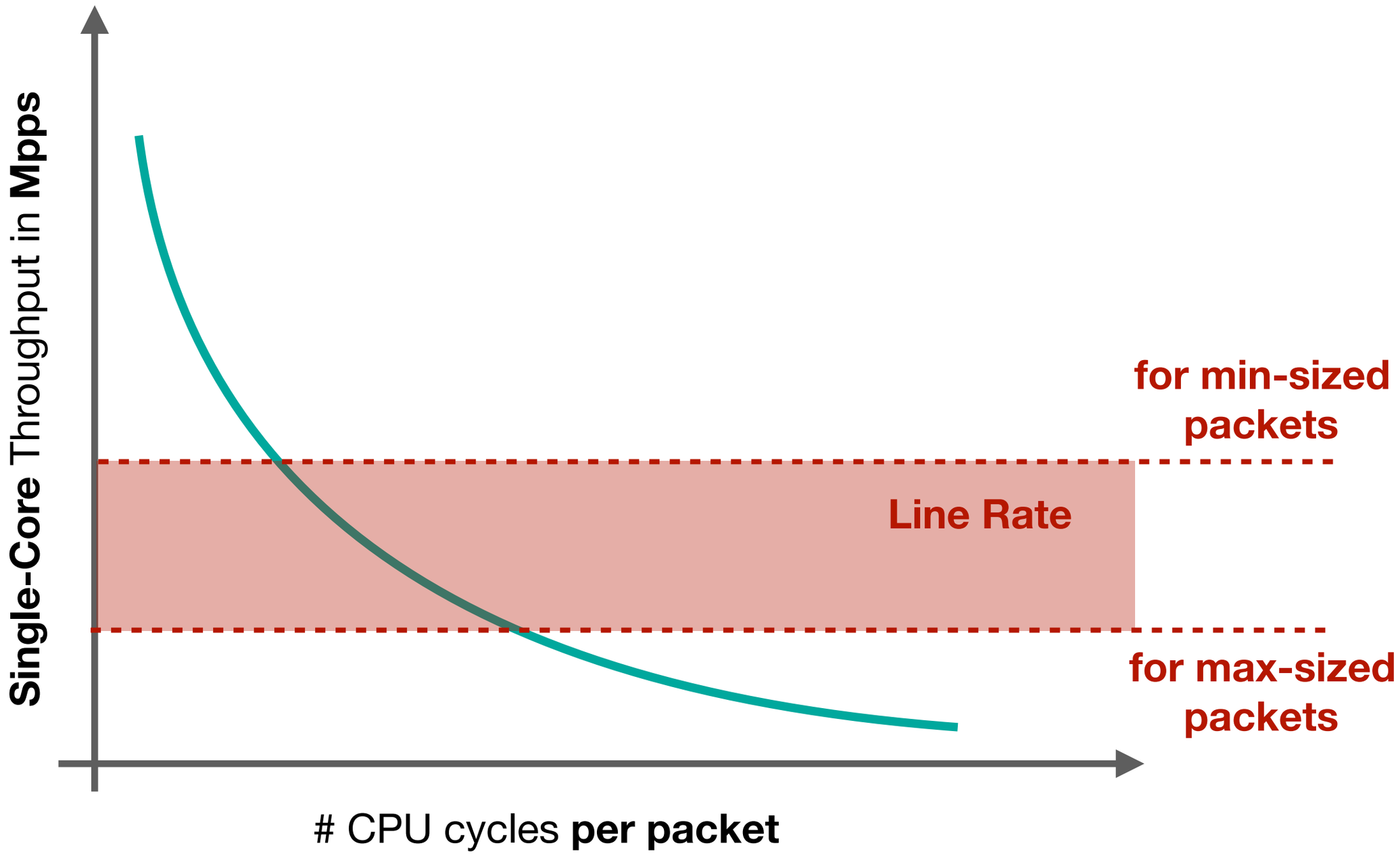
3

CPU could ke[illegible] with the NIC, i.e, process pa[illegible]s at line rate in reasonable [illegible]

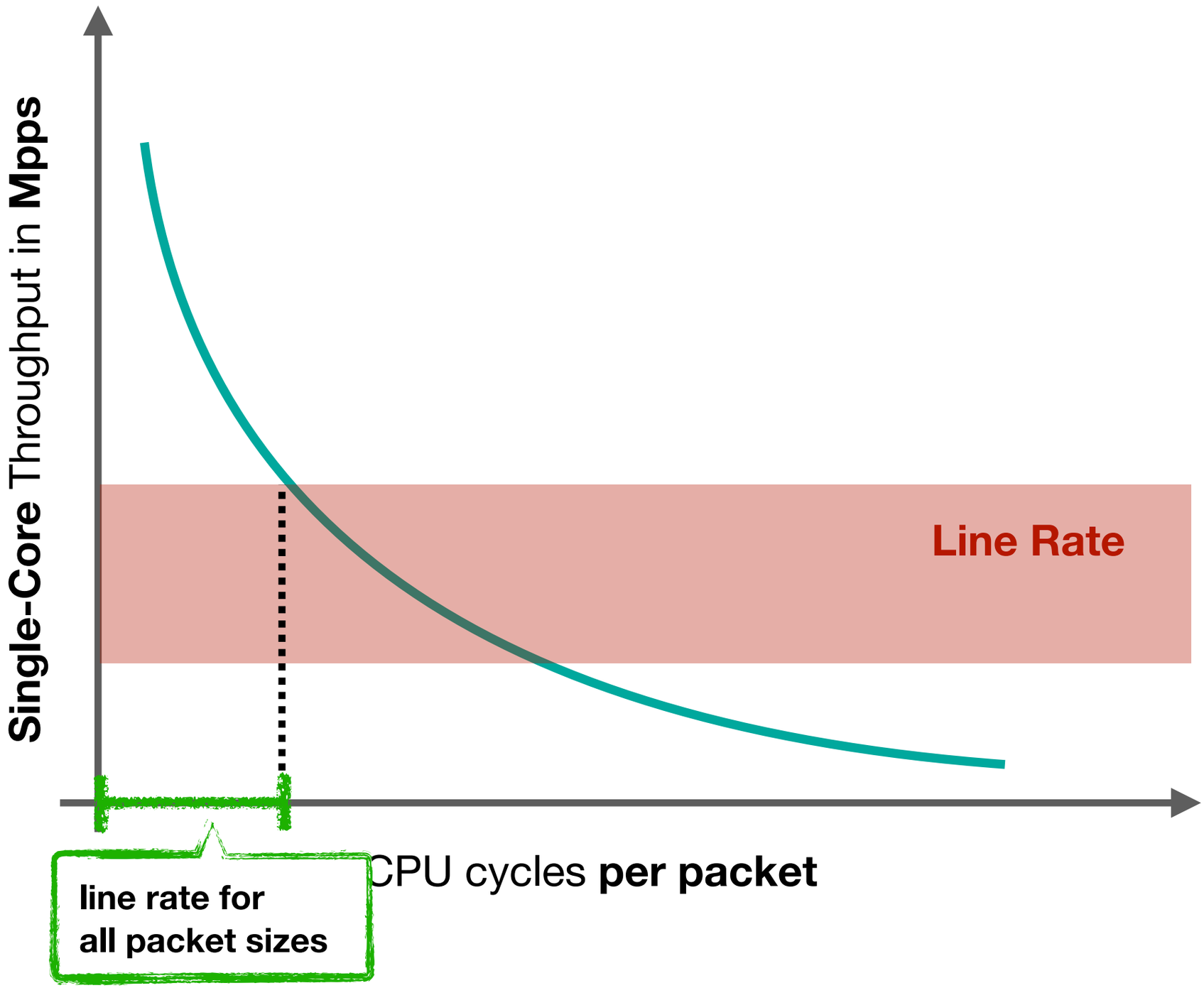
1

- Simple
- Does not change often

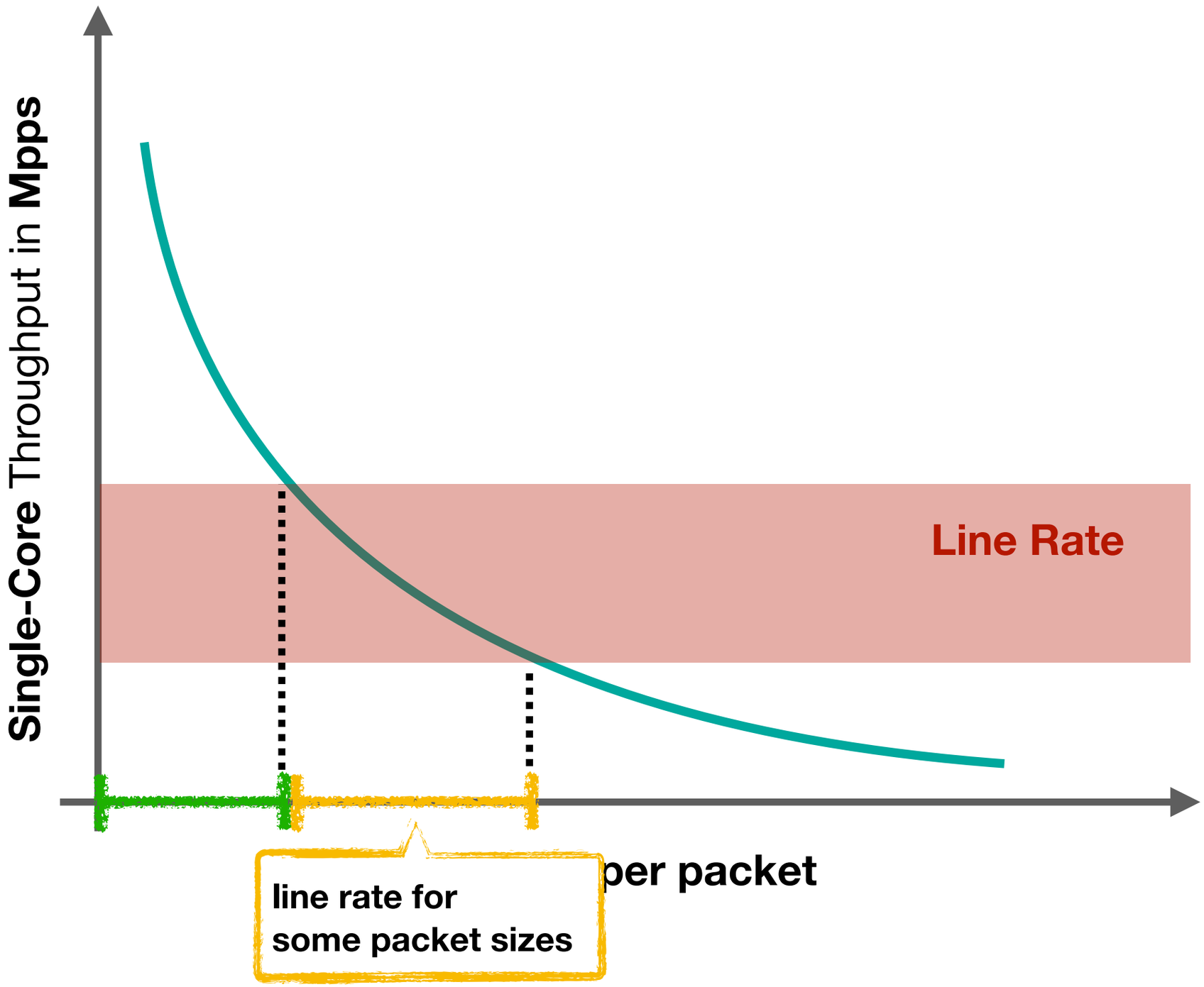
Limits of Software Packet Processing



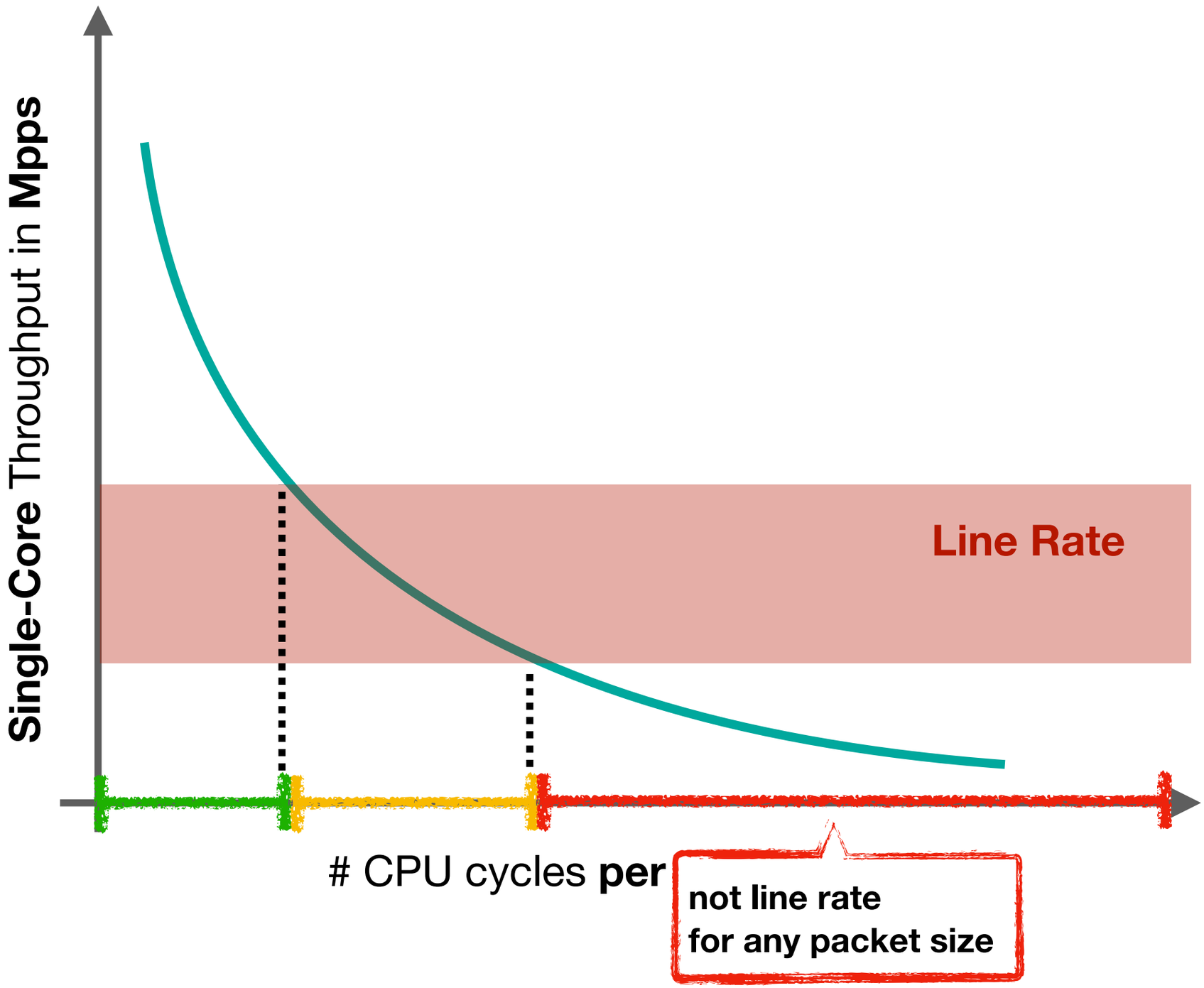
Limits of Software Packet Processing



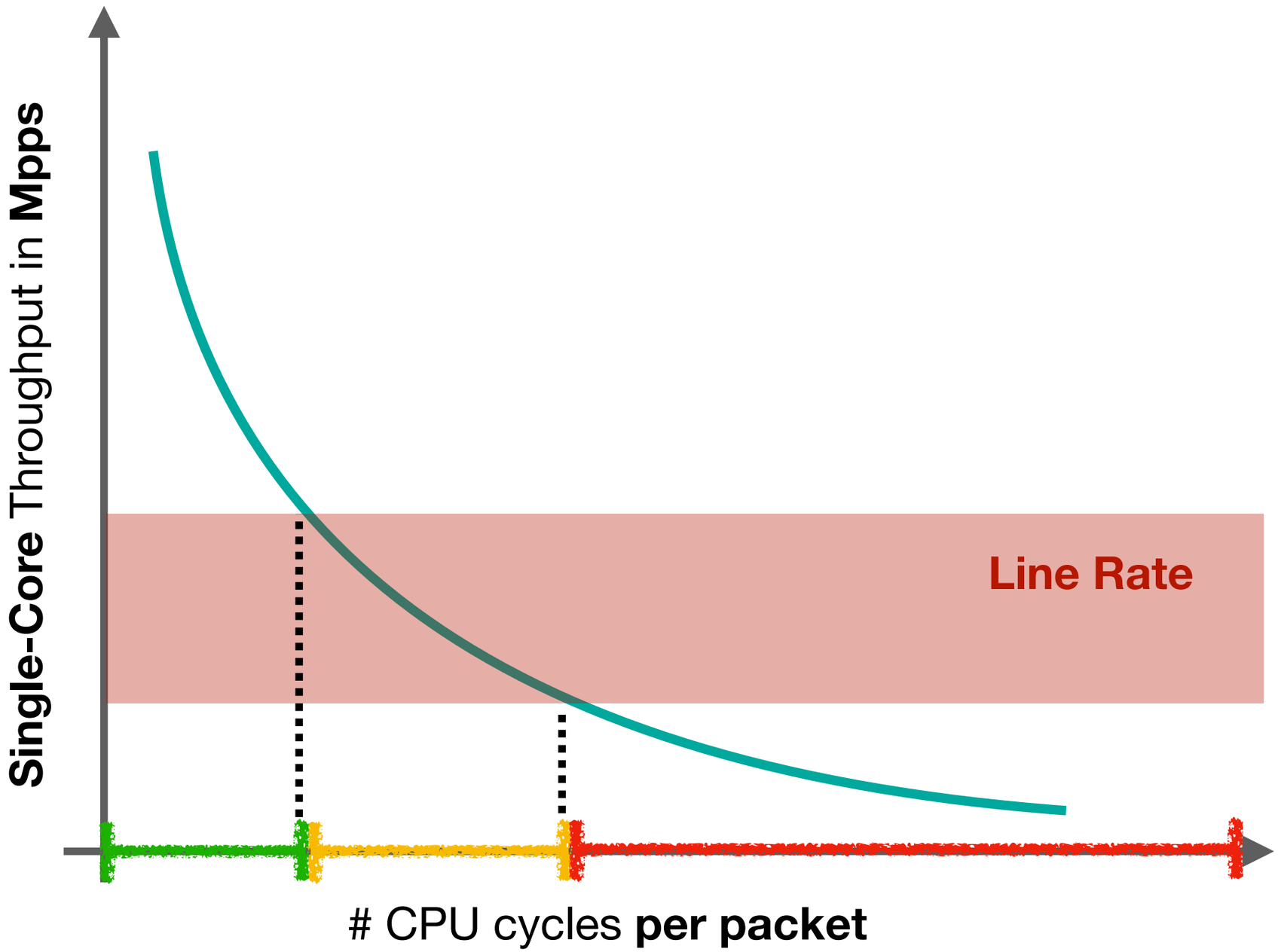
Limits of Software Packet Processing



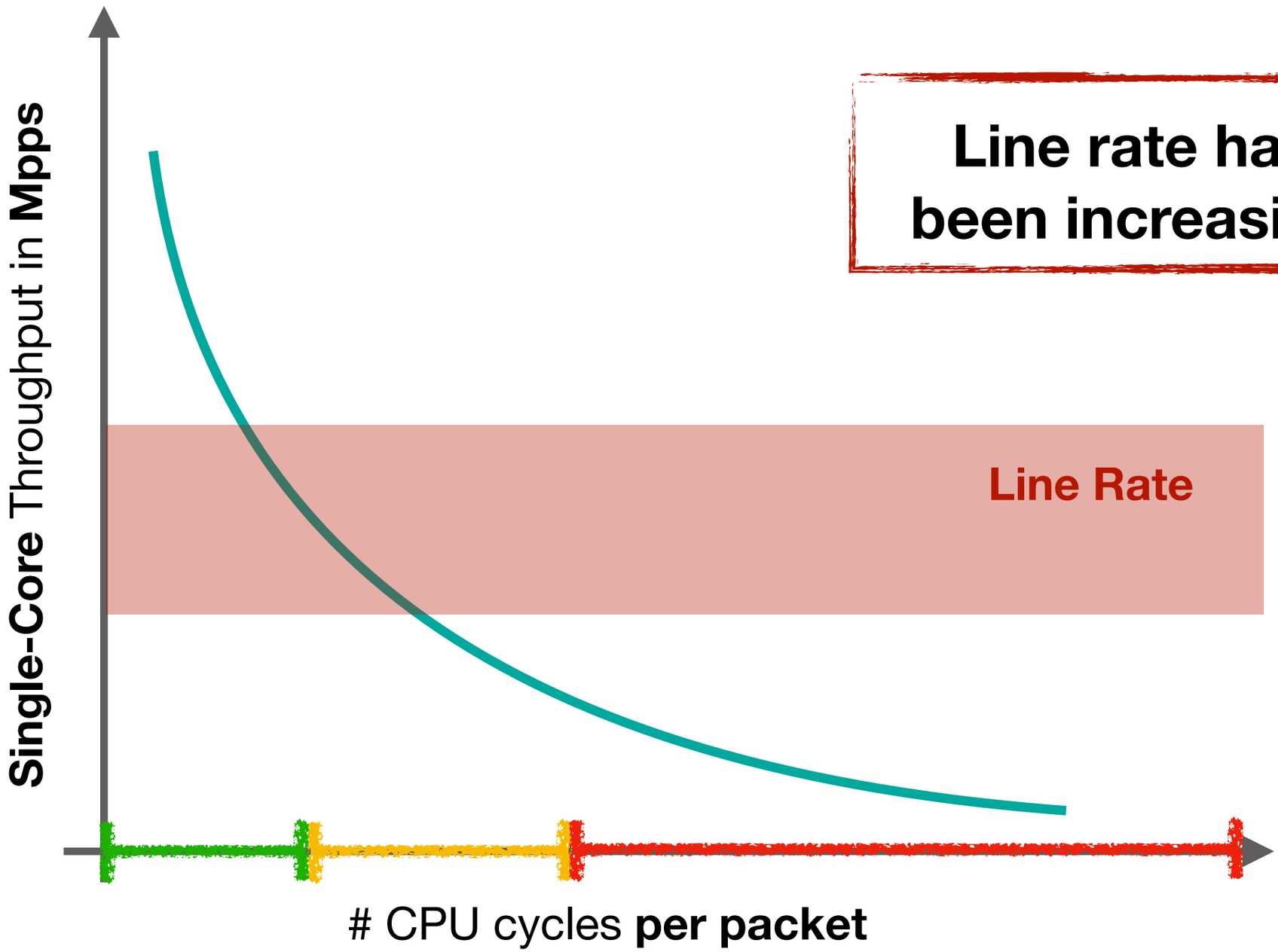
Limits of Software Packet Processing



Limits of Software Packet Processing

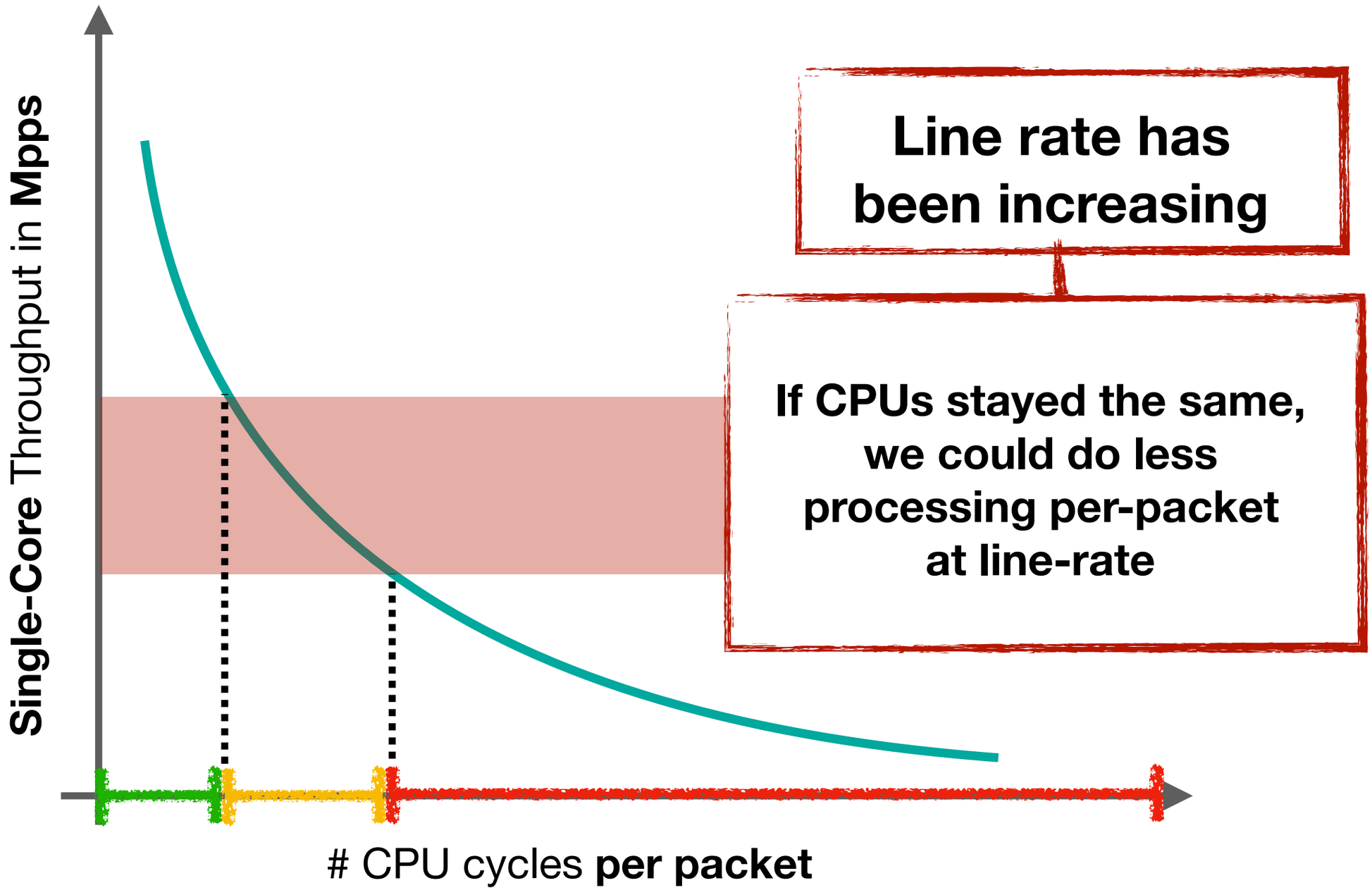


Limits of Software Packet Processing

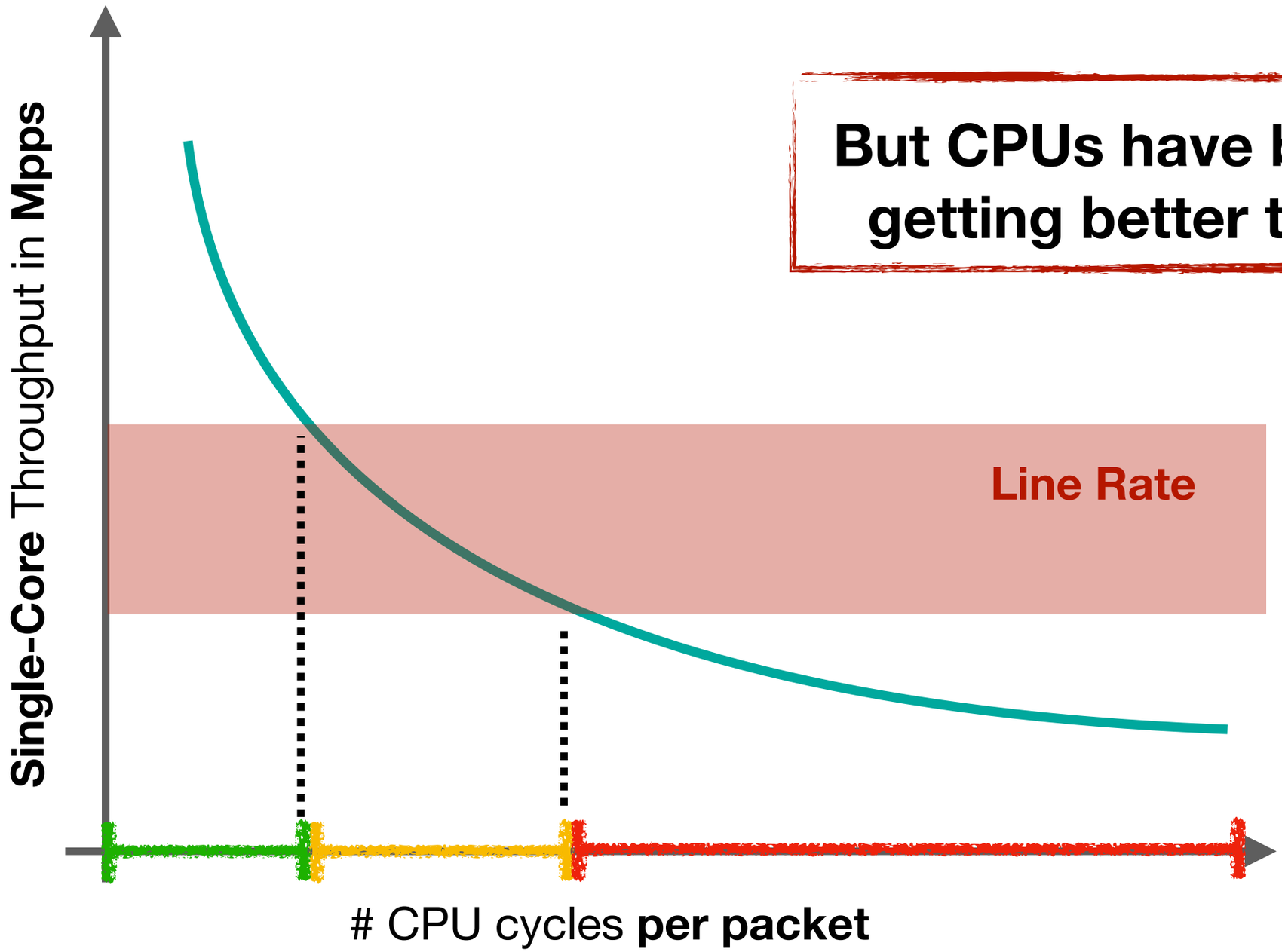


Line rate has been increasing

Limits of Software Packet Processing

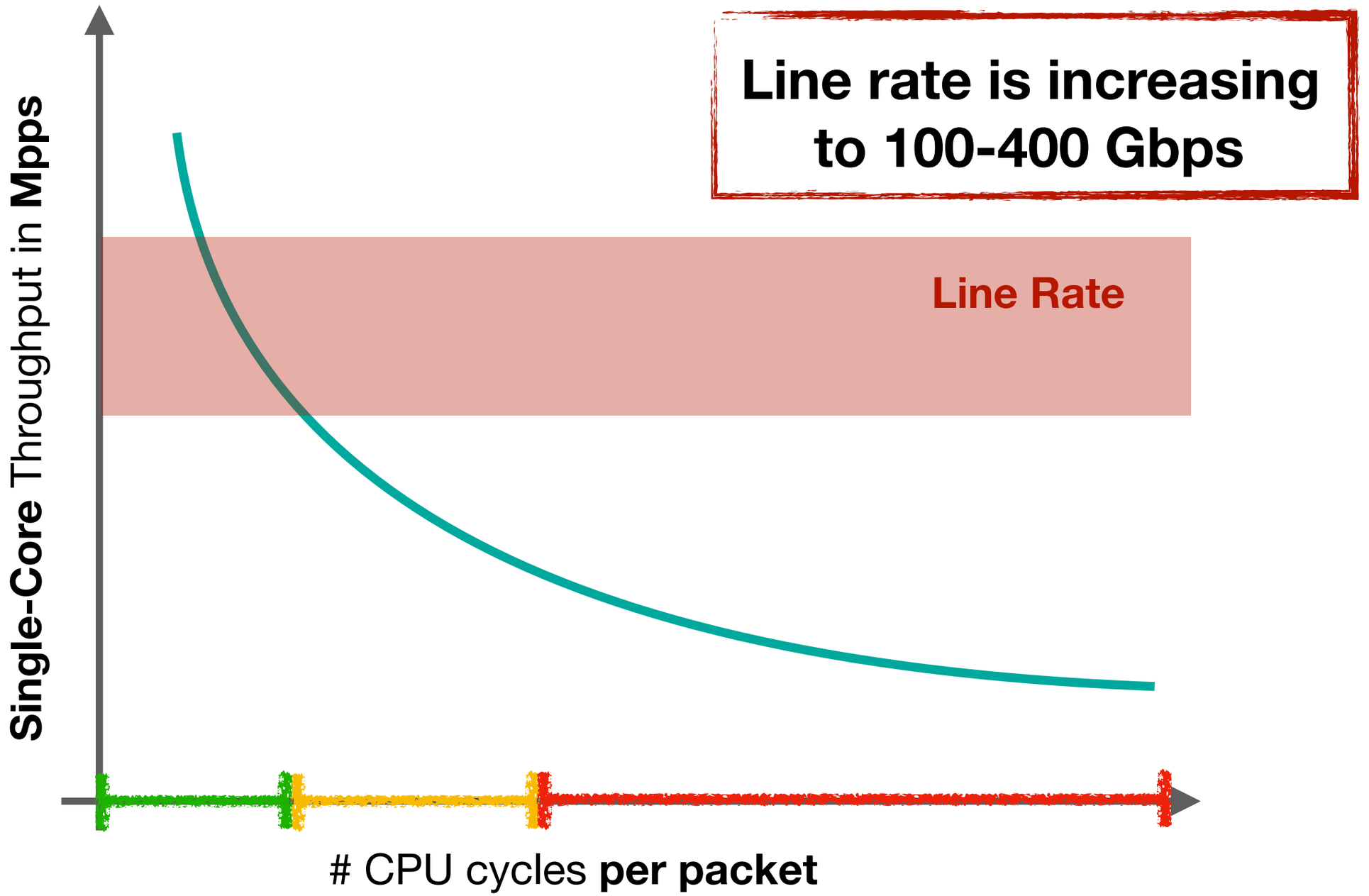


Limits of Software Packet Processing

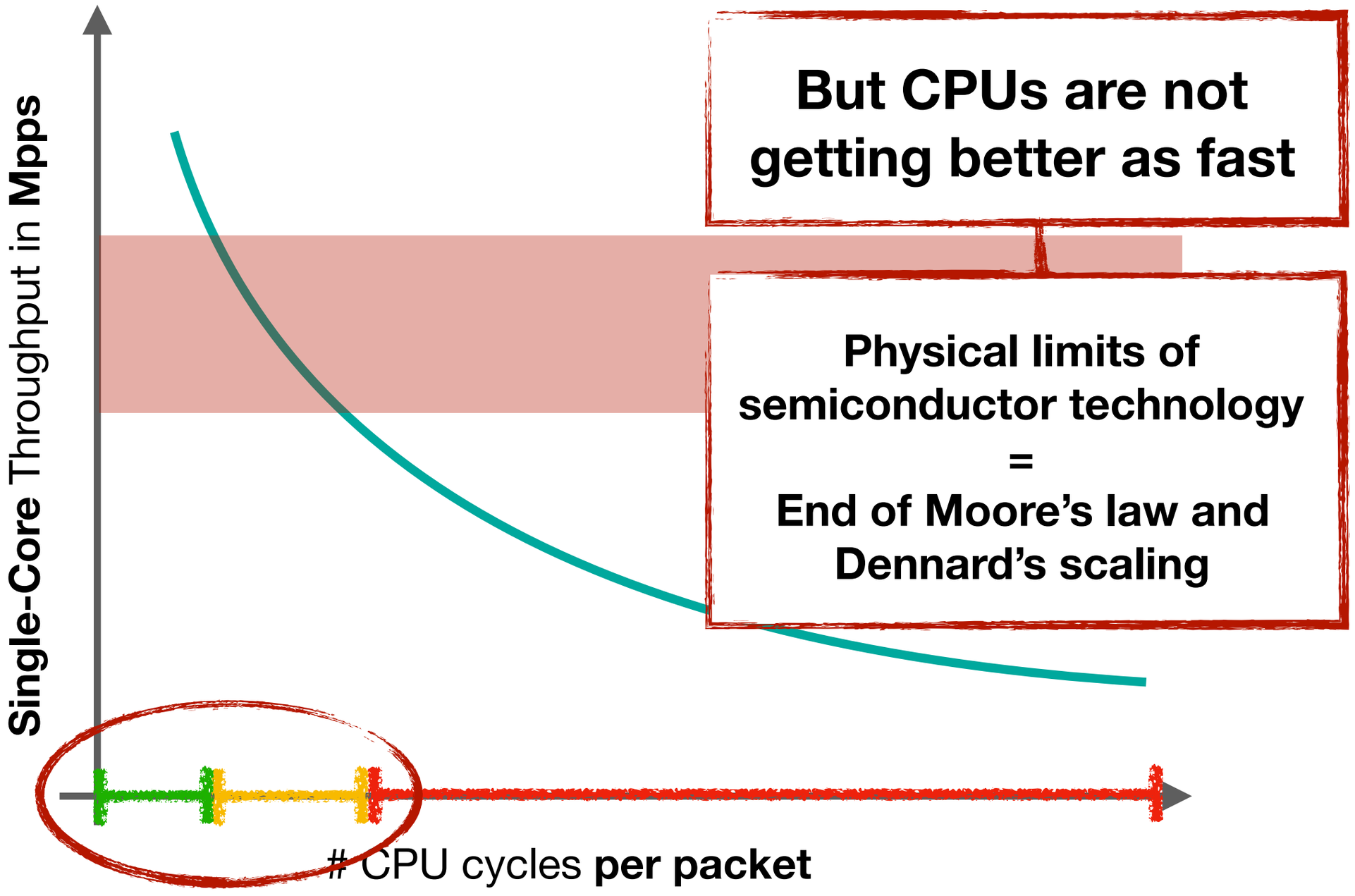


But CPUs have been getting better too!

Limits of Software Packet Processing



Limits of Software Packet Processing



Solution?

A

Programmable

Domain-Specific

Hardware Processor

As opposed to general-purpose CPUs

can be optimized for network processing

operators can decide what part of packet processing is offloaded to hardware and how

Takes over part (or even all) of packet processing that is currently done by CPUs

Solution?

A

Programmable

Domain-Specific

Hardware Processor

Co-location with the NIC provides extra benefits!

On the NIC!

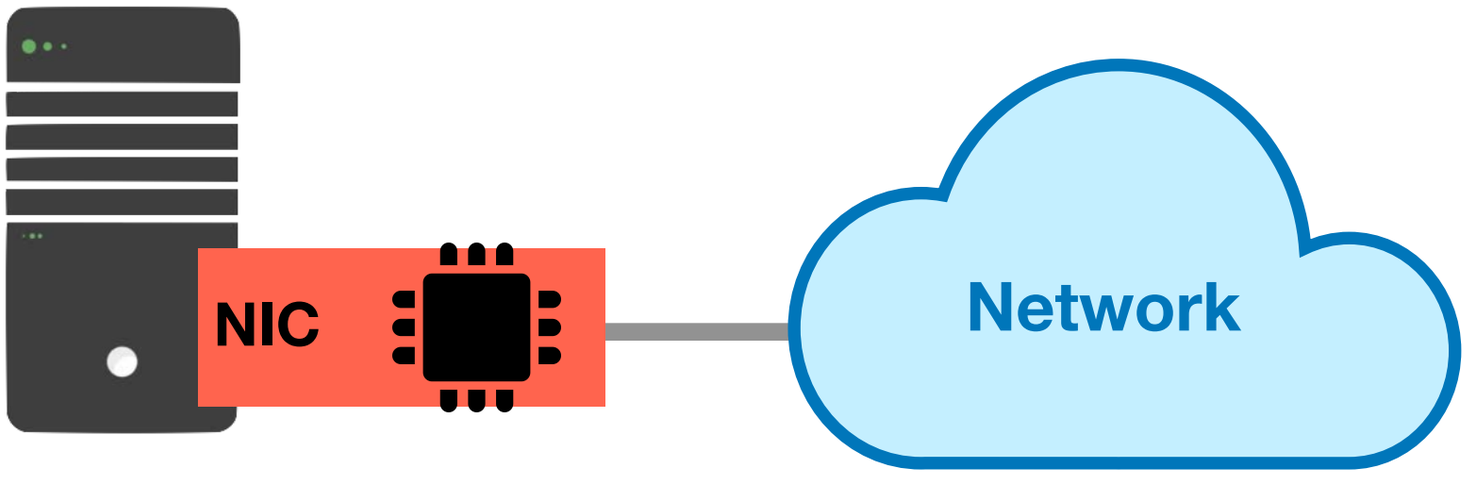
So, what *is* a smart NIC?

You can think of it as

a dumb NIC

+

a programmable domain-specific hardware



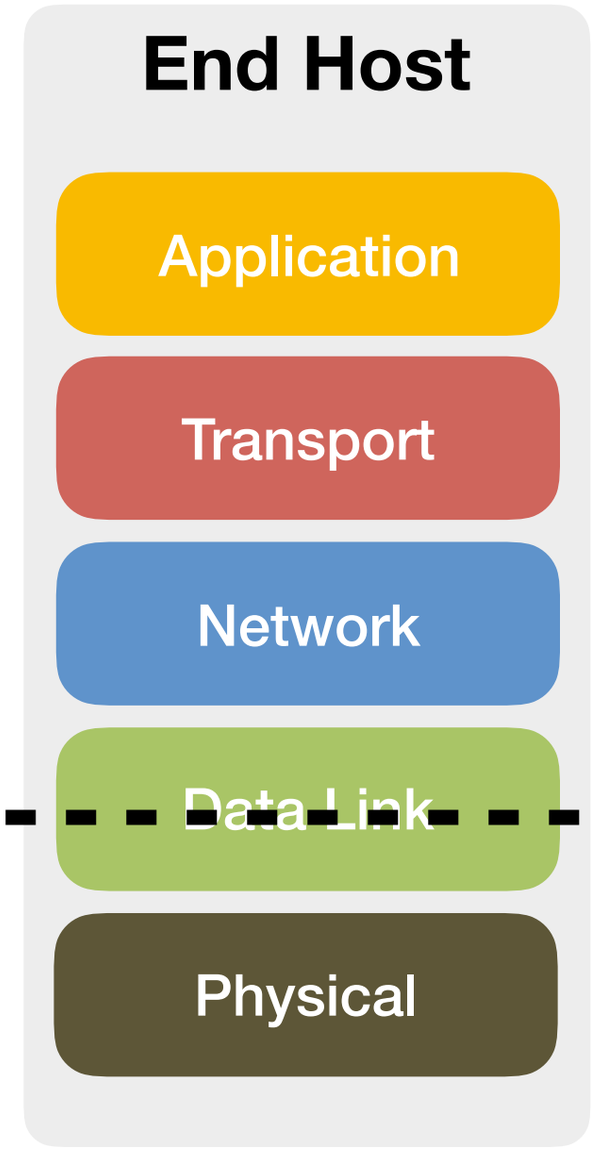
So, what *is* a smart NIC?

CPU

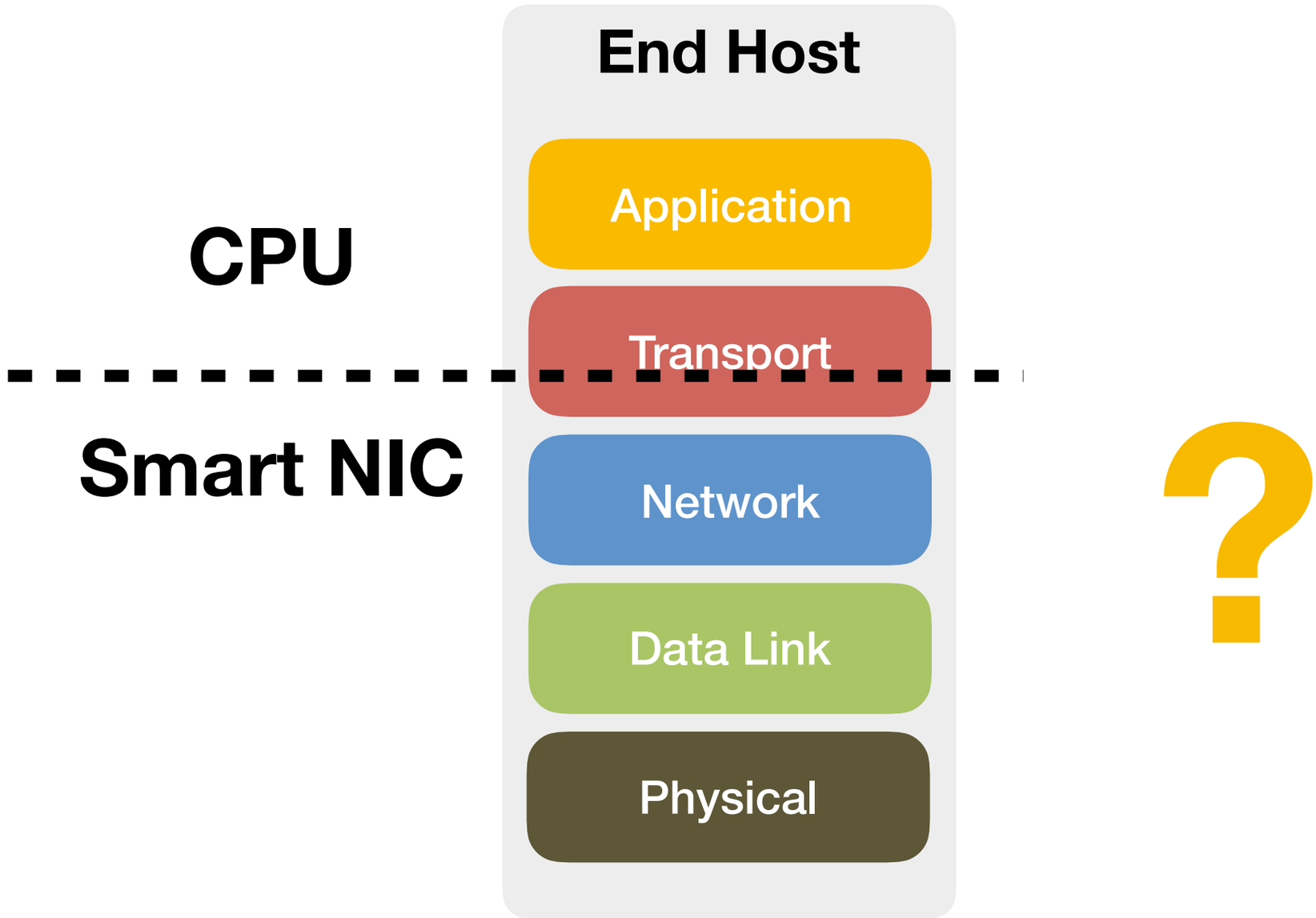
general-purpose
processor
running software

dumb NIC

fixed-function
hardware



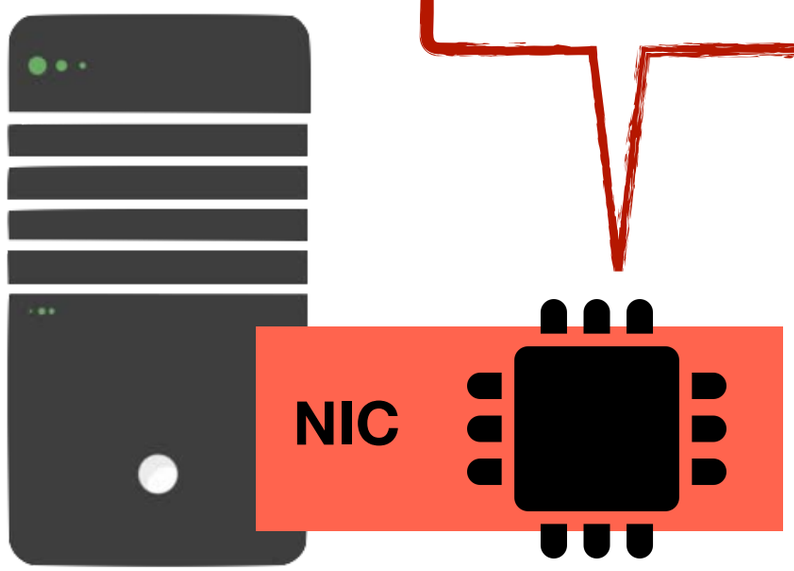
So, what *is* a smart NIC?



A Closer Look at the Hardware

Current popular options:

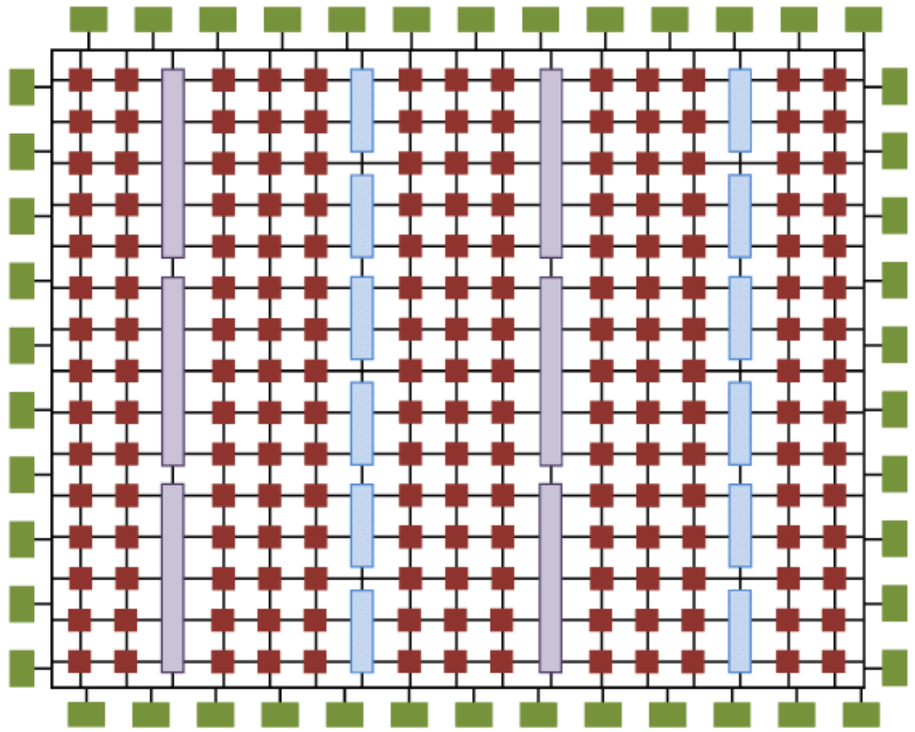
1. Field Programmable gate array (FPGA)
2. Multi-core systems on chip (SoC)



Field Programmable Gate Arrays

- An FPGA is a collection of small configurable logic and memory blocks

- Programmers can write code to assemble these blocks to perform their desired processing

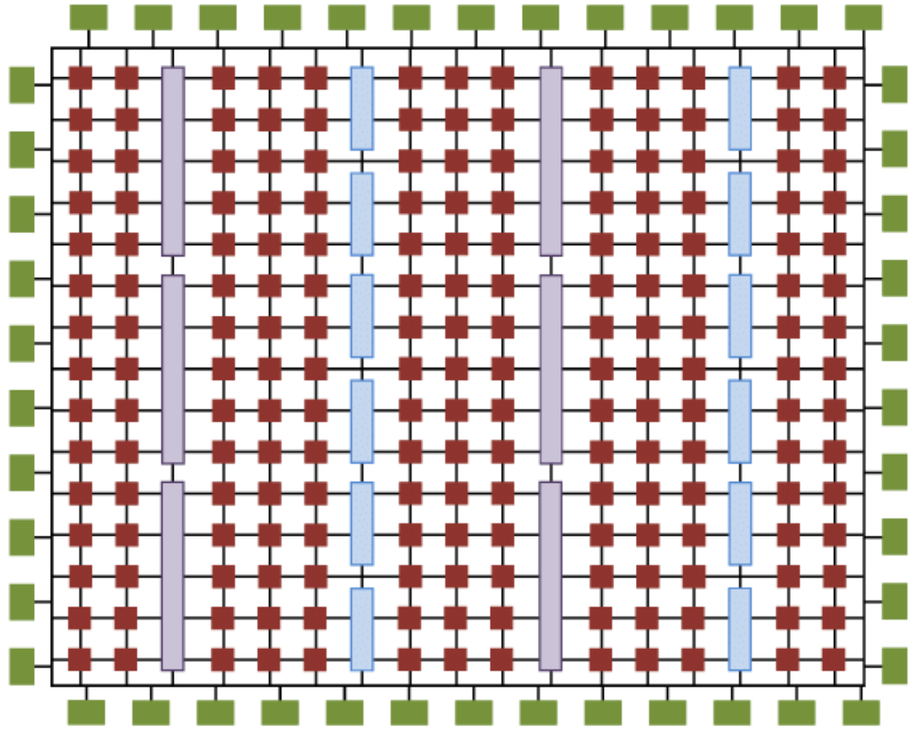


Field Programmable Gate Arrays

Why is an FPGA a popular hardware choice for smart NICs?

- FPGA hardware resources (logic and memory) can be highly customized for the intended computation

- Great fit for highly-parallelizable computation



Multi-Core Systems on Chip

- A “small” computer on a single chip
- Includes (light-weight) processing cores and a memory hierarchy
- Why is it a popular hardware choice for smart NICs?
  - Programming model is close to software
  - Cores (and the architecture) can be specialized for network processing

FPGAs vs Multi-Core SoCs for Network Processing

	FPGAs	Multi-Core SoCs
Hardware Architecture	**Reconfigurable hardware** and therefore can be highly customized for the intended packet processing	The cores' instruction set and memory architecture is fixed and is therefore less customizable
Programming Model	Hardware description languages (e.g., Verilog) ↓ Harder to program	C-like languages ↓ Easier to program
Performance	Higher throughput lower latency *	Lower throughput higher latency *

* For most kinds of network processing

What are smart NICs used for?

- Acceleration across the stack
  - Hypervisor vSwitch: AccelNet (NSDI'18)
  - Scheduling: PIEO (SIGCOMM'19), Loom (NSDI'19)
  - Network functions: ClickNP (SIGCOMM'16), FlowBlaze (NSDI'19)
  - Transport: Tonic (NSDI'20)
  - Even applications: iPipe (SIGCOMM'19), KV-Direct (SOSP'17), Bing web search ranking (ISCA'14)
- Optimizing network I/O
  - e.g., smart steering of packets to cores (FlexNIC, ASPLOS'16)

What are smart NICs used for?

- Acceleration across the stack
  - H
  - S
  - N
  - T
  - E
    s
- Opt
  - e.g., smart steering of packets to cores (FlexNIC, ASPLOS'16)

g web

The Catch?

Resource constraints!

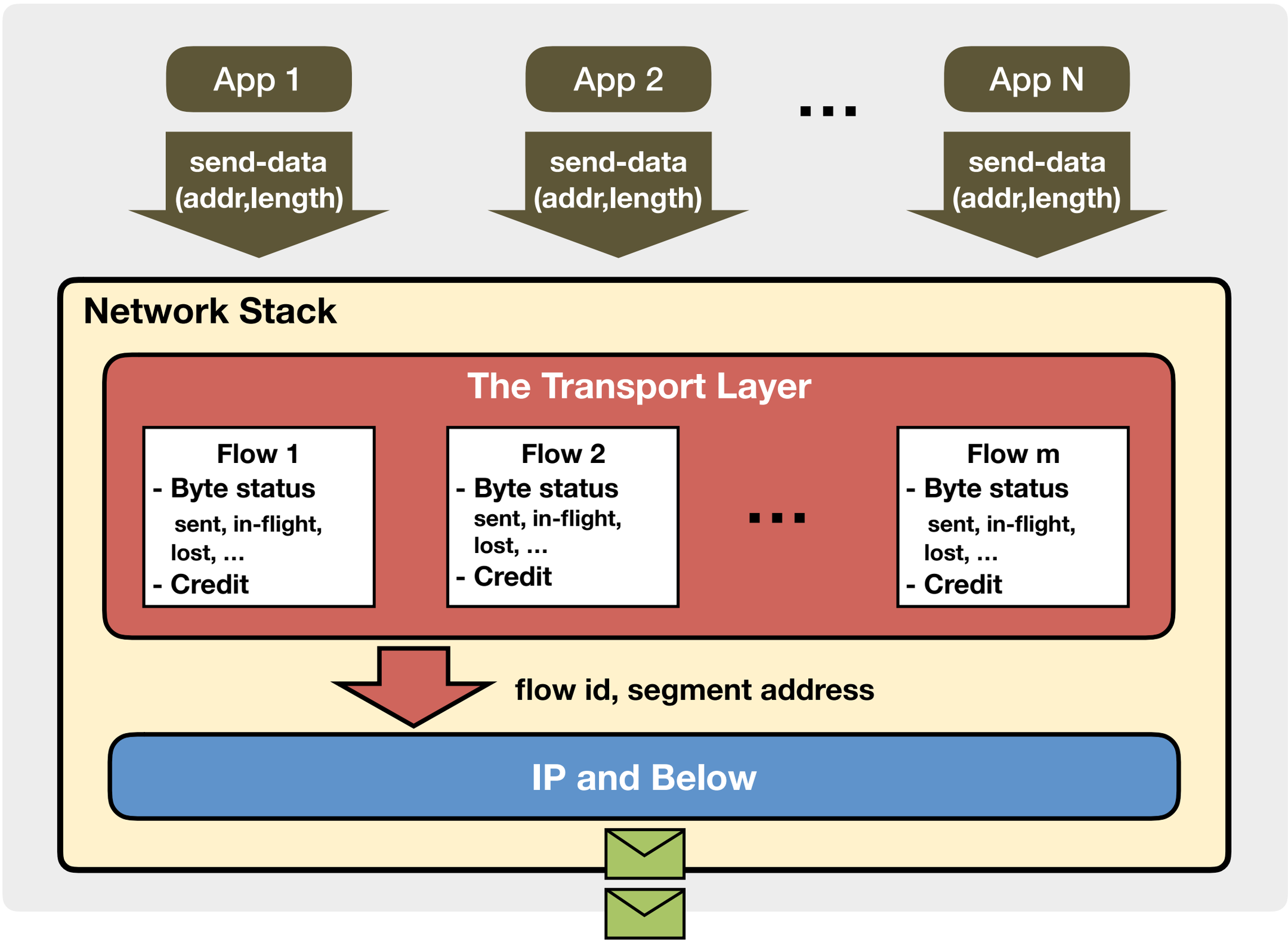
both for computation and memory

Enabling Programmable Transport Protocols on High-Speed NICs

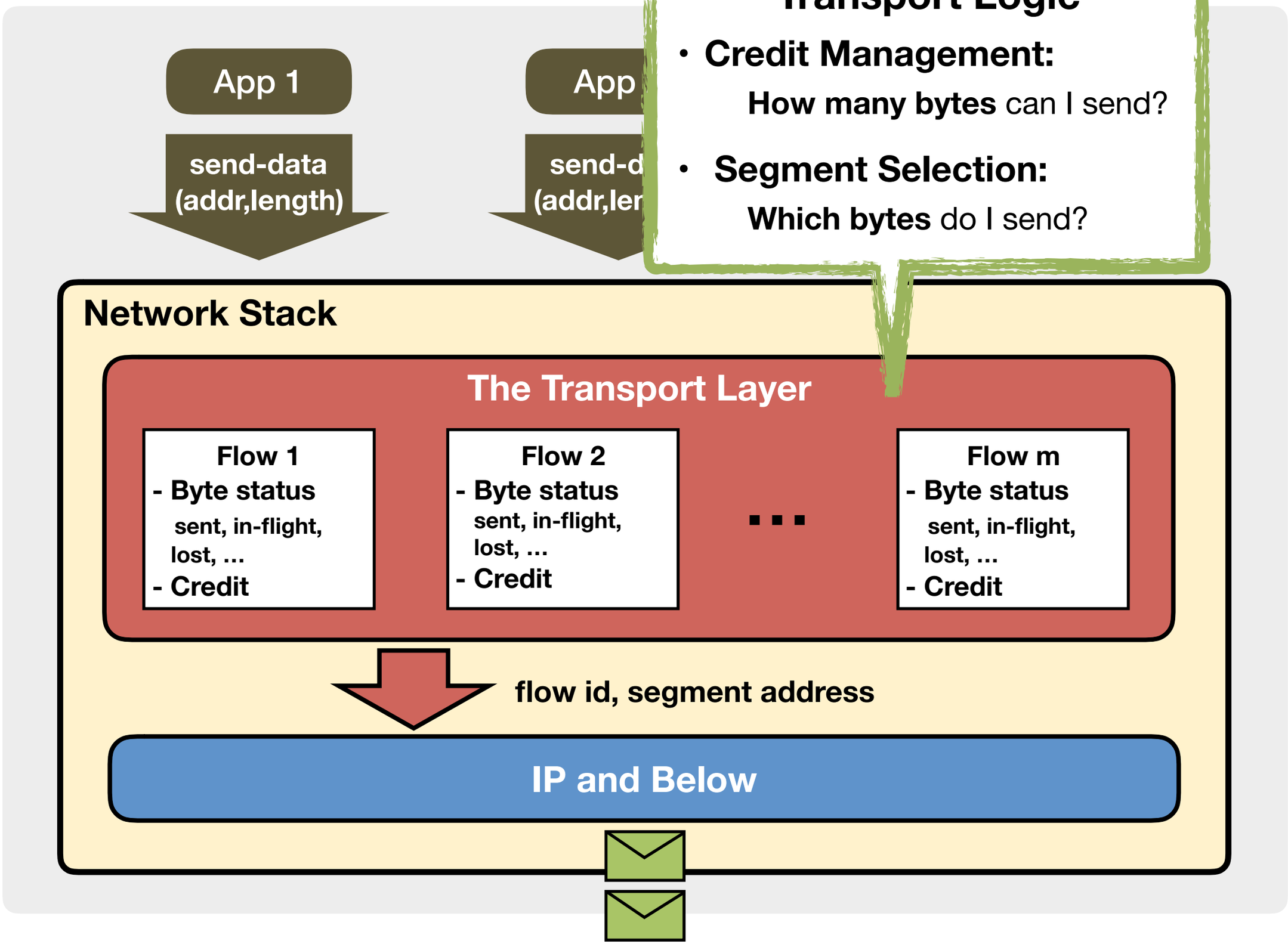
Mina Tahmasbi Arashloo[1], Alexey Lavrov[1],
Manya Ghobadi[2], Jennifer Rexford[1],
David Walker[1], and David Wentzlaff[1]

[1] Princeton University, [2] MIT

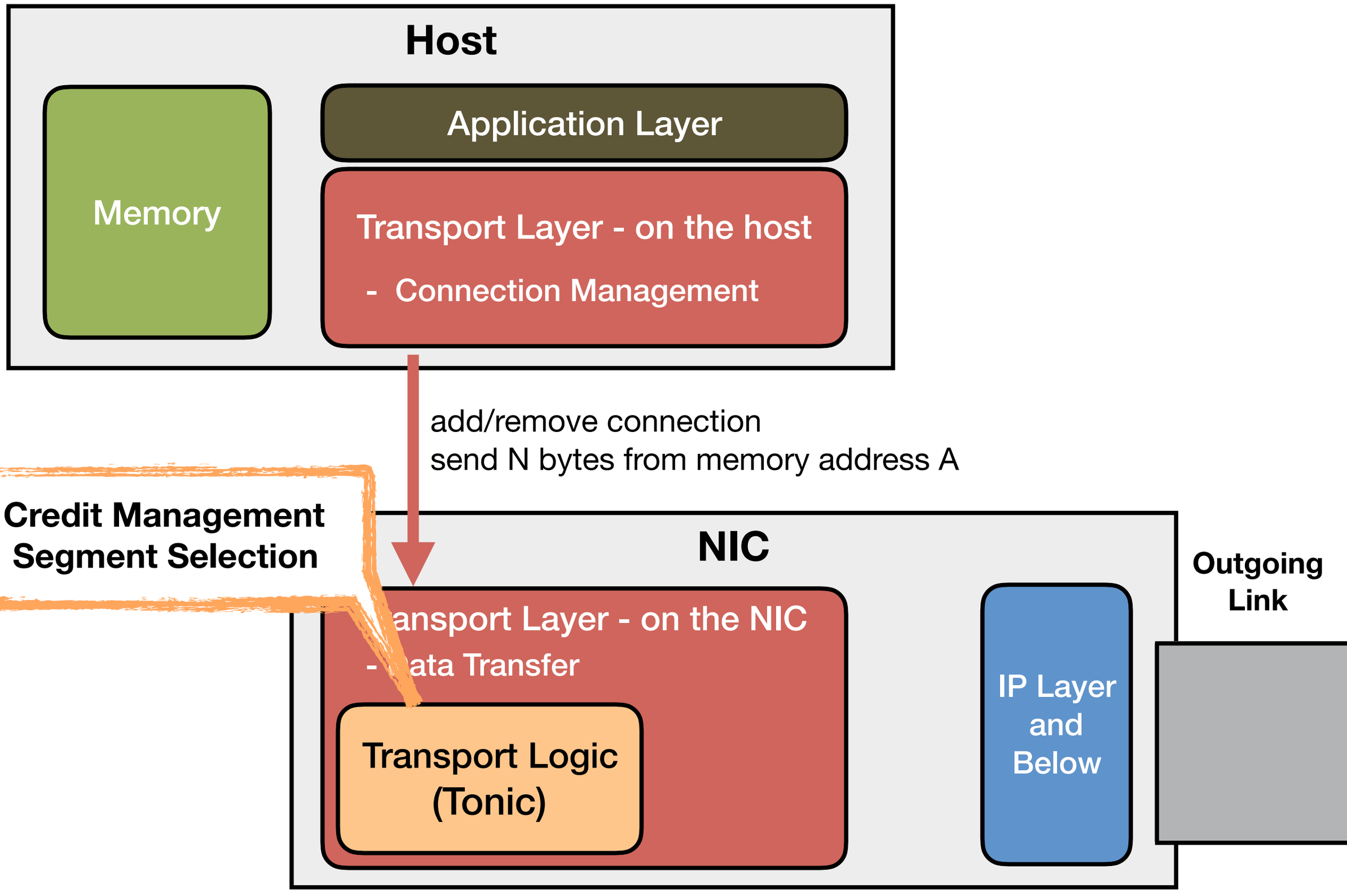
The Transport Layer



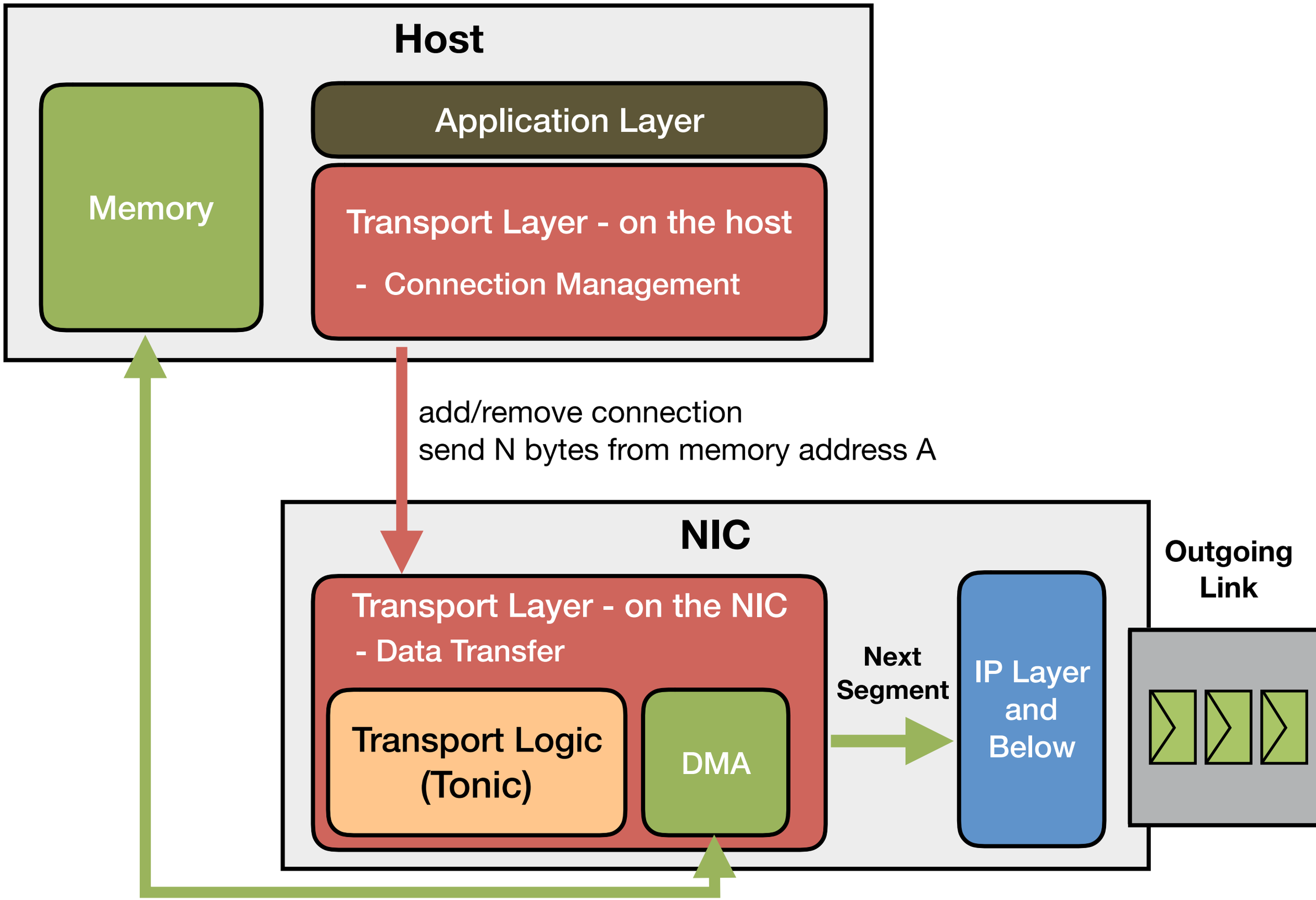
The Transport Layer



Overview


Host
Memory
Application Layer
Transport Layer - on the host
- Connection Management
add/remove connection
send N bytes from memory address A
• Credit Management
• Segment Selection
NIC
ansport Layer - on the NIC
- ata Transfer
Transport Logic (Tonic)
IP Layer and Below
Outgoing Link

Overview


Host
Memory
Application Layer
Transport Layer - on the host
- Connection Management
add/remove connection
send N bytes from memory address A
NIC
Transport Layer - on the NIC
- Data Transfer
Transport Logic (Tonic)
DMA
Next Segment
IP Layer and Below
Outgoing Link

Challenges of Implementing Transport Logic on High-Speed NICs

- **Timing Constraints**
  - Median packet size in data centers is 200 bytes
  - At 100 Gbps, one 128-byte packet every ~10 ns
    - Back-to-back stateful event processing
- **Memory Constraints**
  - A few megabytes of high-speed memory
  - More than a thousand active flows
  - A few kilobits of per-flow state

Challenges of Implementing Transport Logic on High-Speed NICs

- **Timing Constraints**
  - N
  - A
  - .
- **Me**
  - A
  - M
  - A few kilobits of per-flow state

Tonic

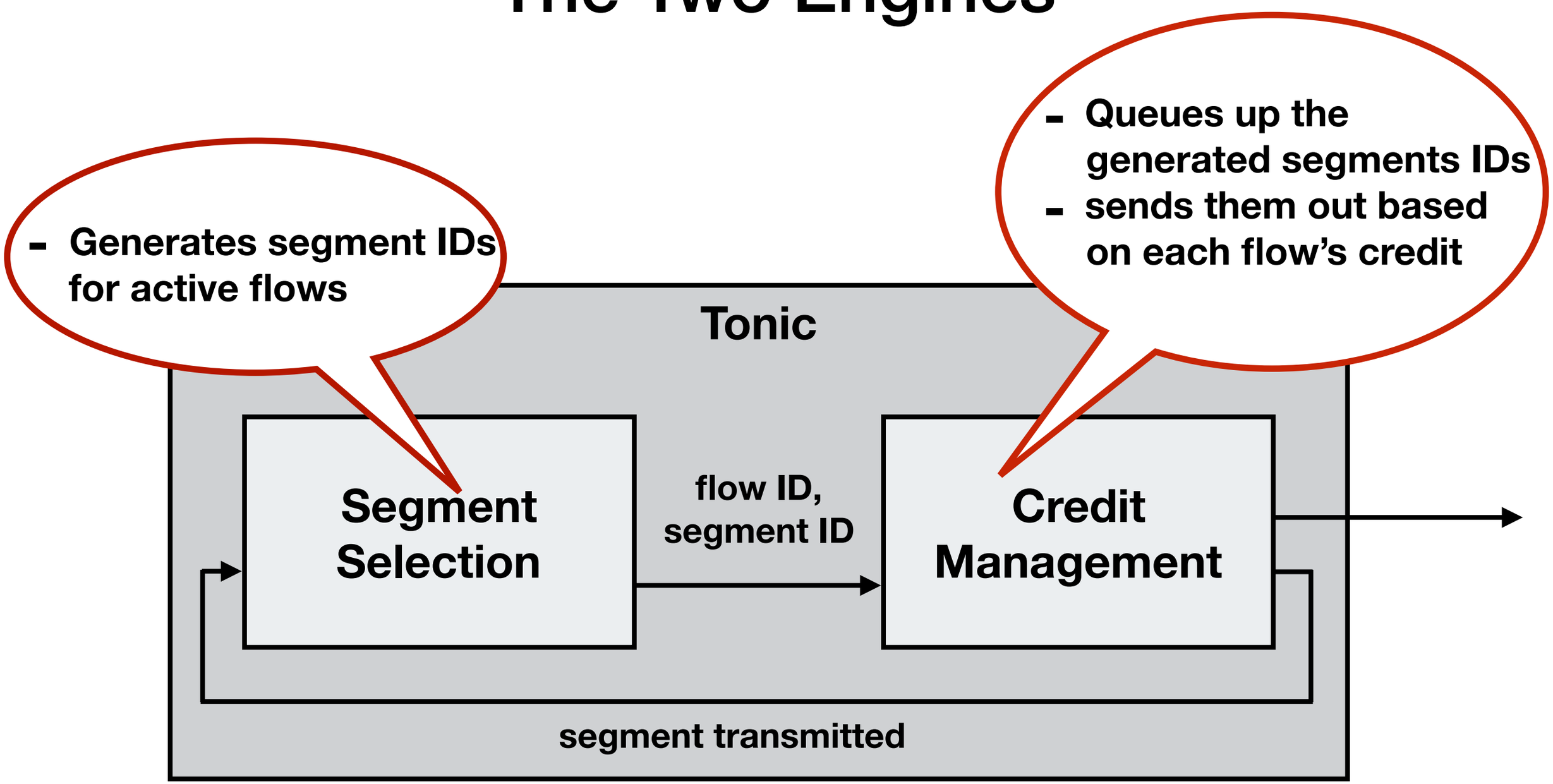
- A **programmable** hardware architecture
  - running at **100** Gbps
  - within **memory limits of commodity NICs**
- to implement transport logic
  - with **modest development effort**

Main Observation

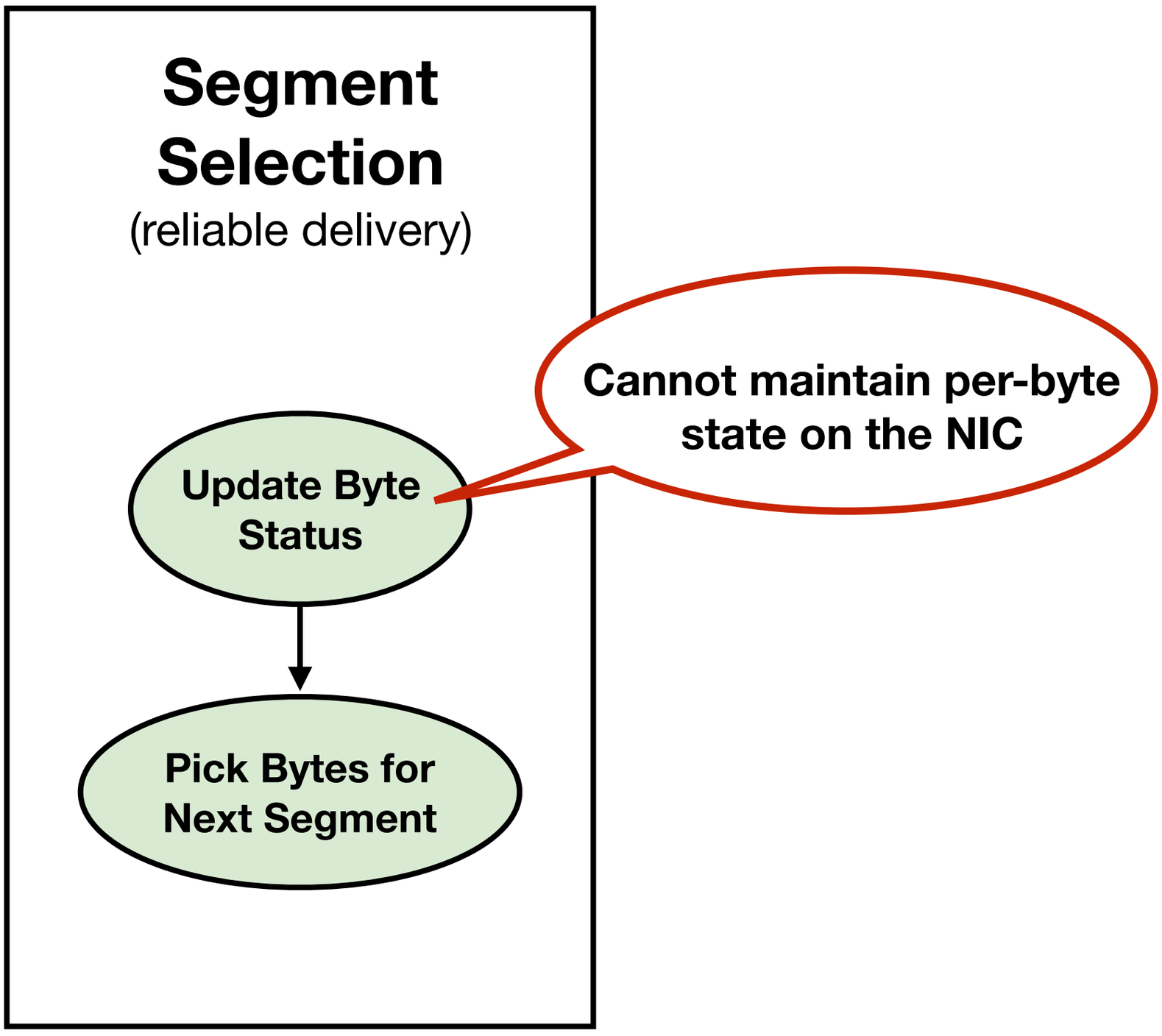
Common transport patterns as reusable components

- drive the design of an efficient hardware “template” for transport logic
- reduce the functionality users must specify

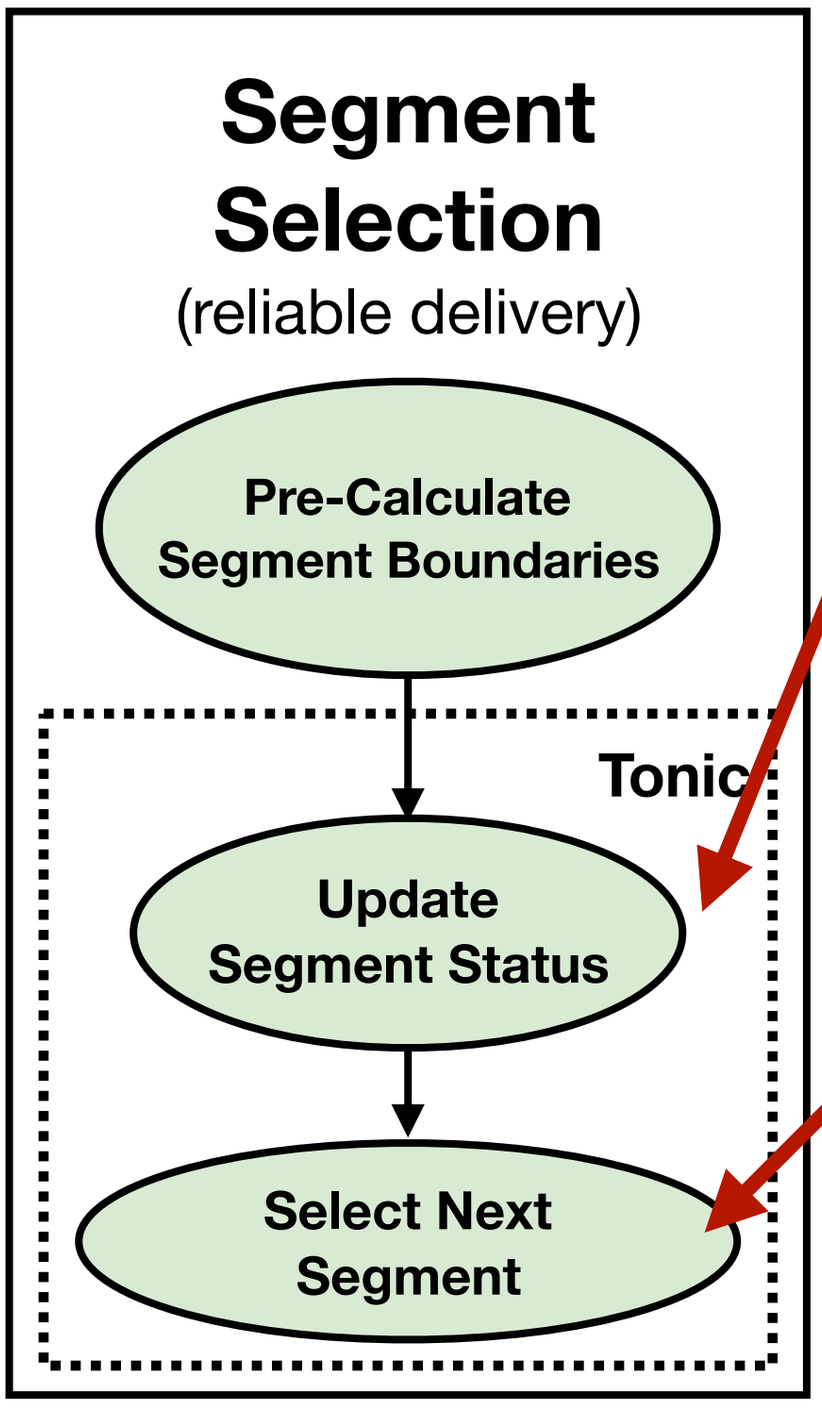
The Two Engines



Segment Selection Patterns

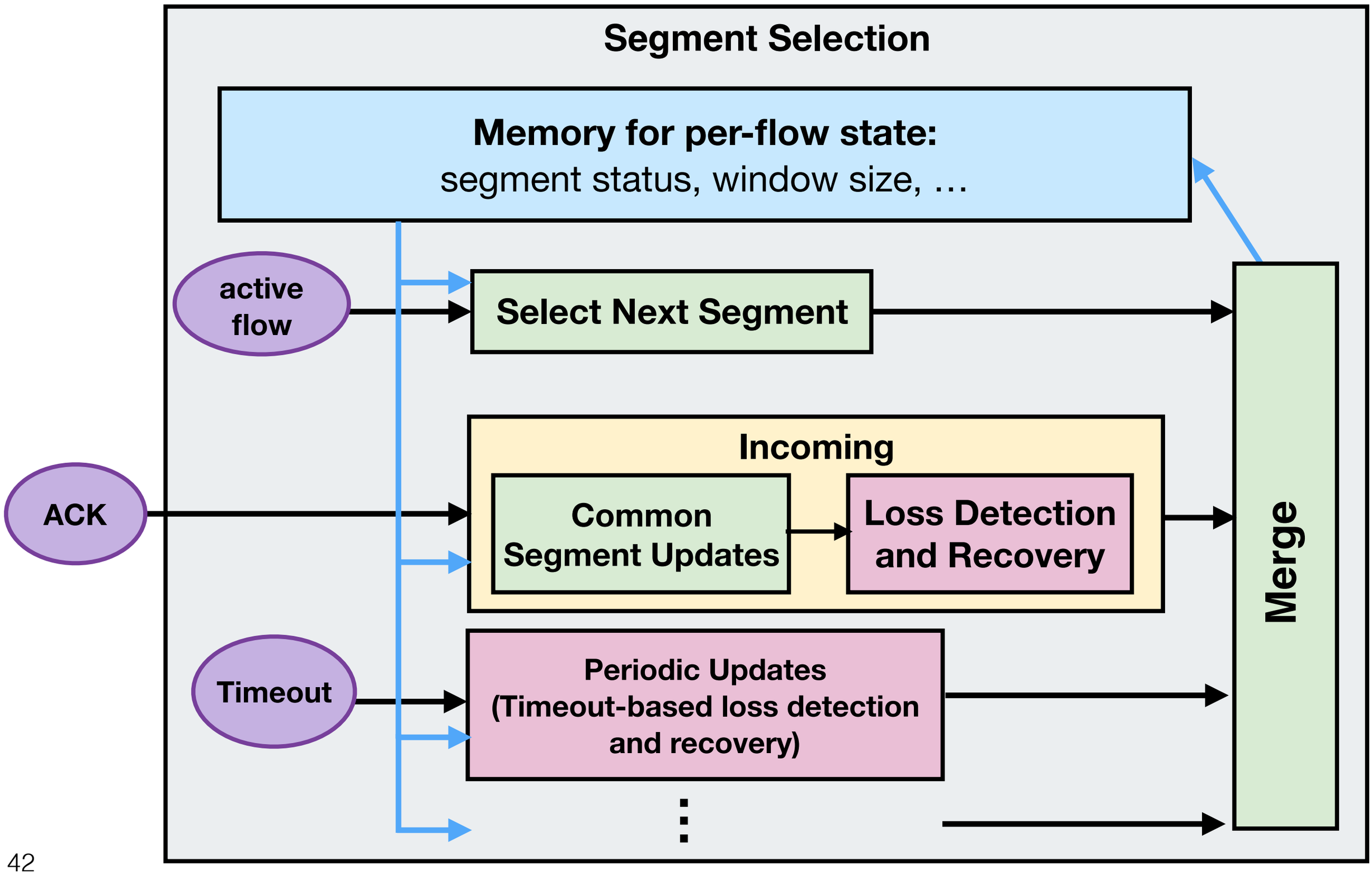


Segment Selection Patterns

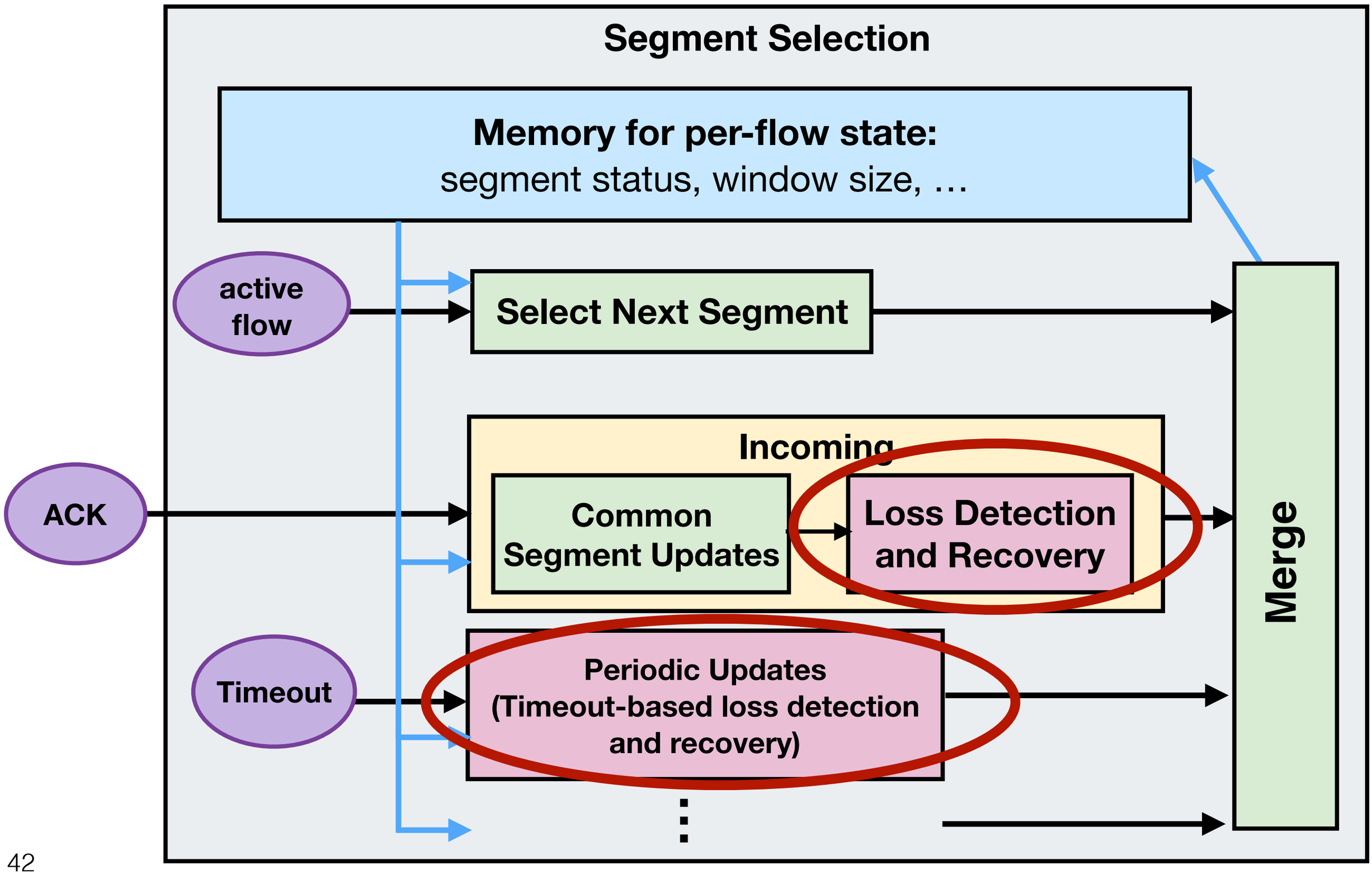


1. **Only a few bits of state per segment**
   - acked, rtxed, lost
   - fixed function modules for common state updates
   - programmable modules only for loss detection
2. **Loss detection: acks and timeouts**
   - only two programmable modules
   - mutually exclusive → fewer concurrent state updates
3. **Lost segments first, new segments next**
   - fixed-function module for segment generation

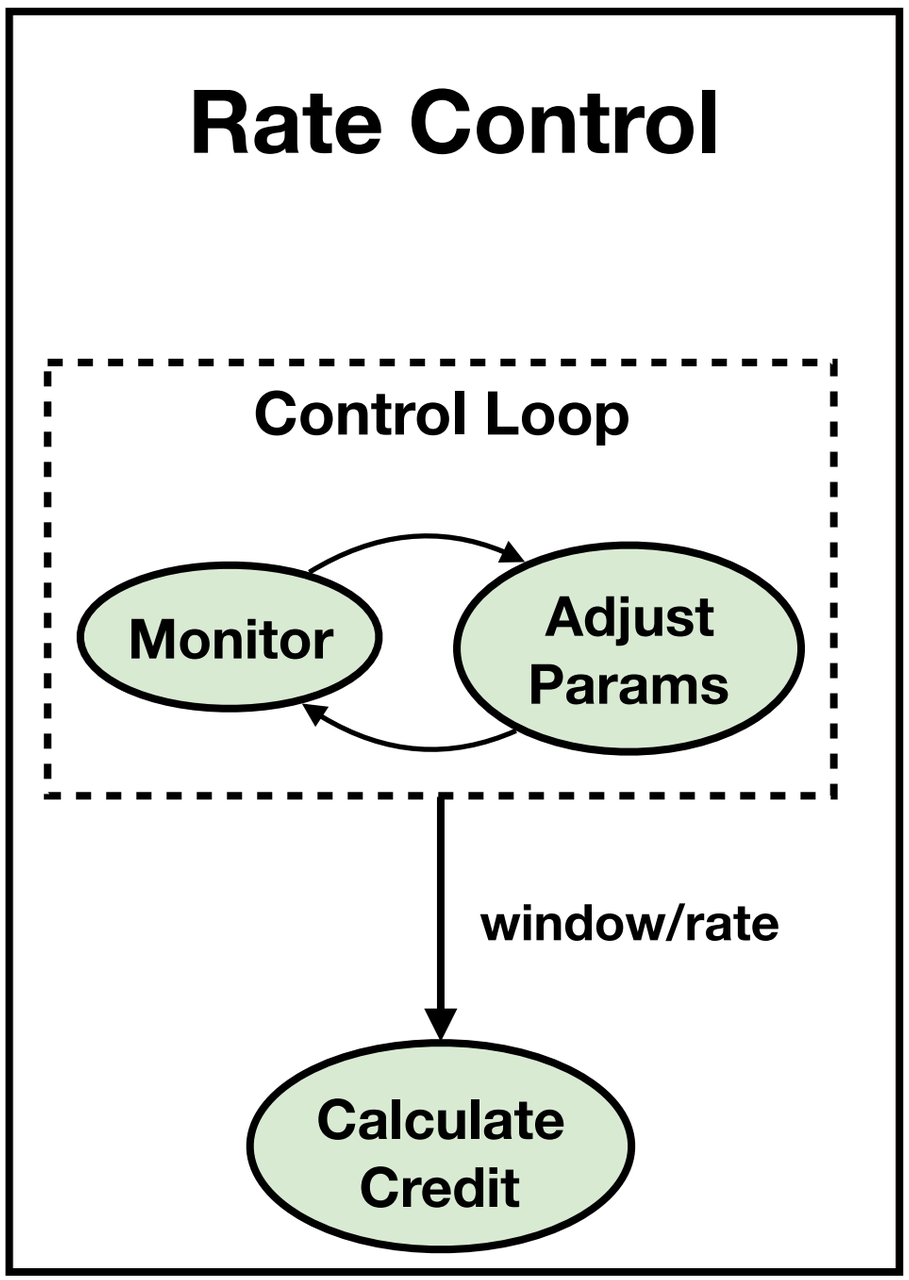
Tonic's Segment Selection Engine



Tonic's Segment Selection Engine

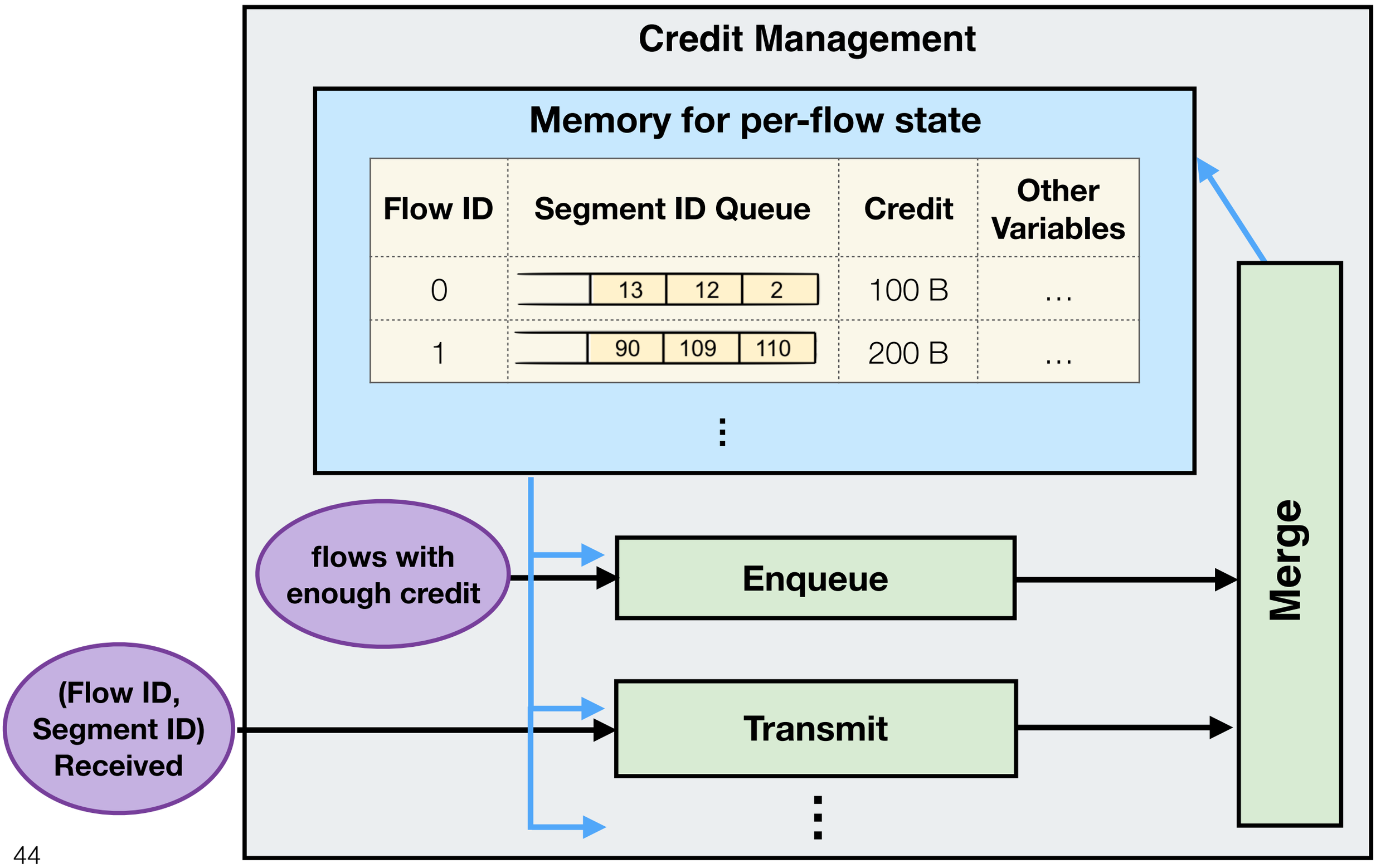


Credit Management Patterns



1. **Common credit management schemes**
   - Rate control: congestion window, data rate
   - Admission control: grant tokens
2. **Two main parameter adjustment signals**
   - external signals, e.g., acks and CNPs
   - periodic internal signals, .e.g., counters
   - aligns with existing programmable modules for segment selection

Tonic's Credit Management Engine



Hardware Implementation Challenges

- Consistent stateful operations
- Bitmap Operations
- Per-flow rate limiting
- More details in the paper

Evaluation - Programmability

- Implemented six representative protocols
  - Reno, New Reno
  - SACK (Selective ACK)
  - NDP (Receiver-driven data-center transport)
  - DCQCN, IRN (Improved RoCE NIC)

- All meet timing for 100 Gpbs (10-ns clock)

- Implemented within 200 lines of *Verilog* code
  - uses 0.5% of total logic resources

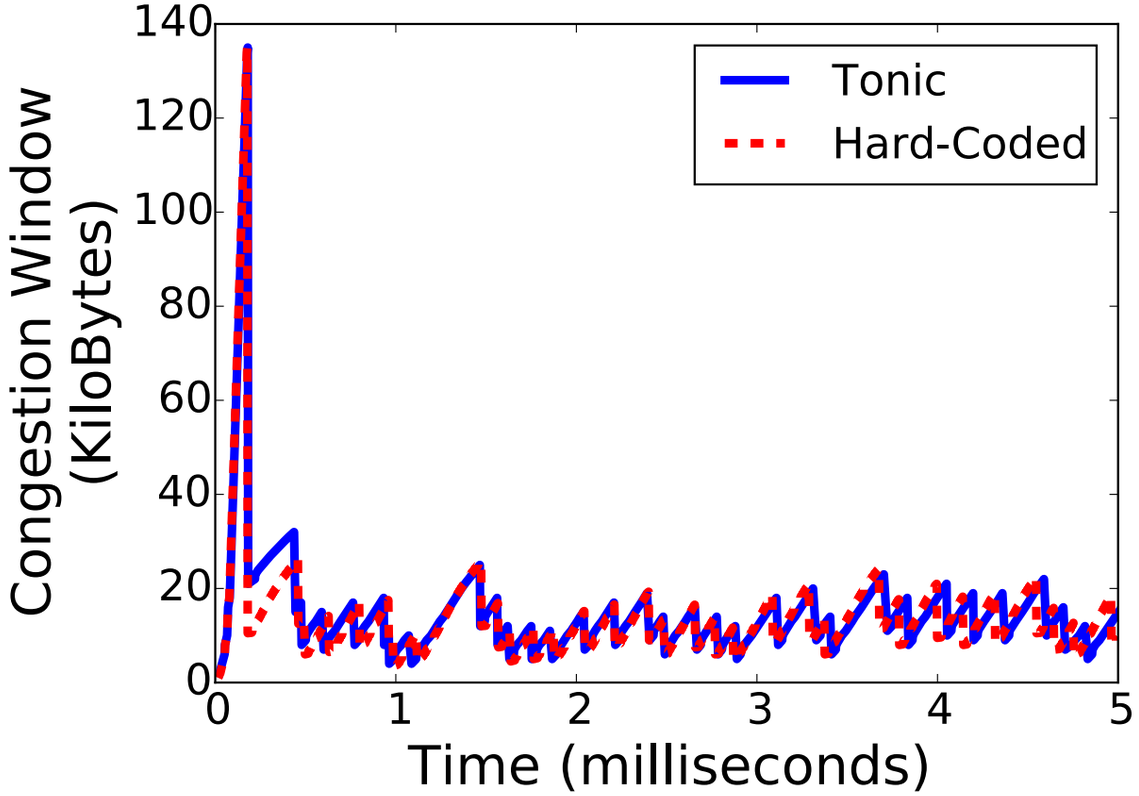
- Re-usable modules are 8K lines of *Verilog* code
  - uses 35% of total logic resources

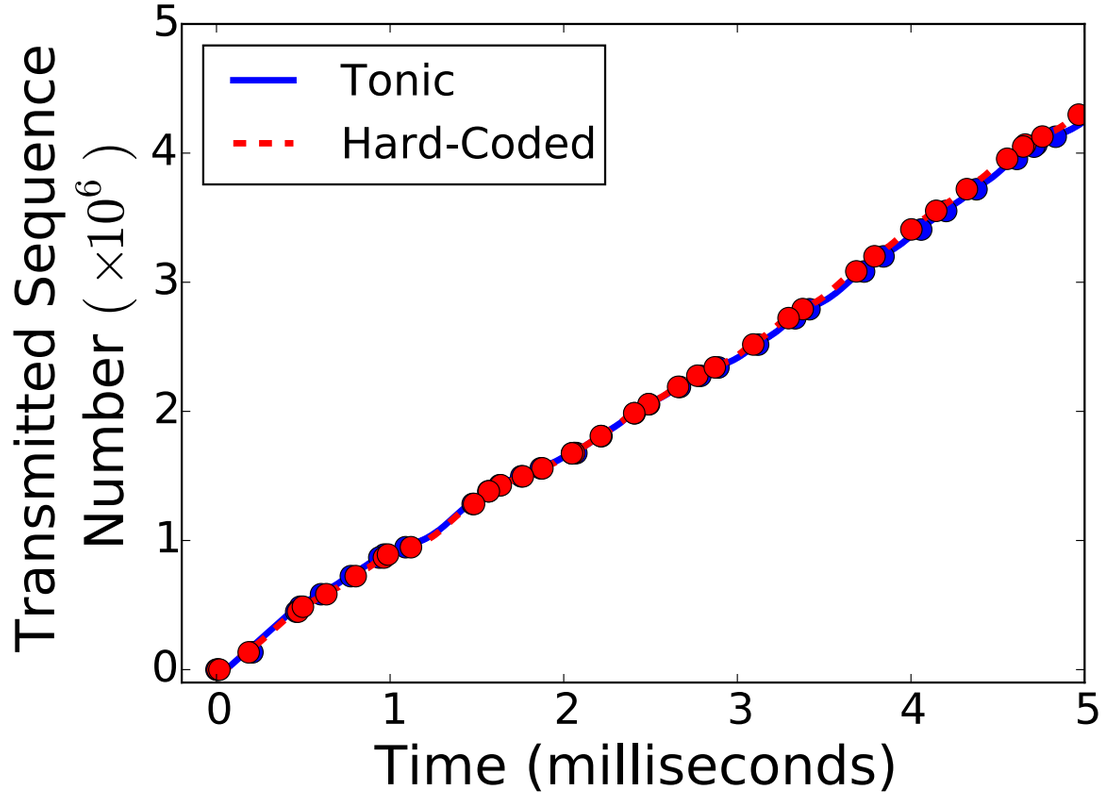
Evaluation - Scalability

	Metric	Results	
Complexity of User-Defined Logic	logic levels	$(\ 0\ , 31]$	meets timing
		$(31, 42]$	depends on operations
		$(42, 65]$	violates timing
User-Defined State	bytes	256	grant token
		340	rate
		448	congestion window
Window Size	segments	256	
Concurrent Flows	count	2048	

Evaluation - End-to-End Simulations

- Cycle-accurate hardware simulator for Tonic within NS3
- Compared existing protocols with Tonic implementations
  - TCP New Reno (plots shown below) and DCQCN





What’s Next for Smart NICs?

- More acceleration in each layer of the stack
  - application acceleration
  - Hardware-efficient transport

What's Next for Smart NICs?

- Generalizing network processing over heterogenous hardware
  - Given a network function, and
  - a server with a CPU, and some accelerators on the NIC (FPGA, SoC, maybe even a GPU?)
  - what is the best offloading strategy?
  - iPipe (SIGCOMM'19): distributed applications over CPU and SoC-baed NICs
  - Can we add programmable switches into the picture?