UNIVERSITY OF CAMBRIDGE

P51(bis): High Performance Networked-Systems

Prof. Andrew W. Moore

Lecture 5/6

A **huge** thank you to Eben Upton, Raspberry Pi Foundation, and PiHut people for enabling this incarnation of the module at incredibly short notice.

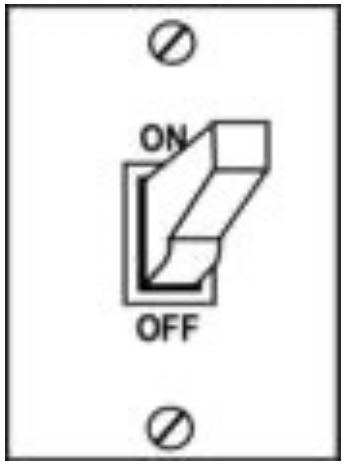
With great appreciation to Dick Sites for sharing wisdom, patience, and teaching materials.

With ongoing gratitude to Dr Noa Zilberman

General architecture of high performance network devices

What Is a Switch?

We use switches all the time!



ON / OFF

Left / Right

What Is a Network Switch?

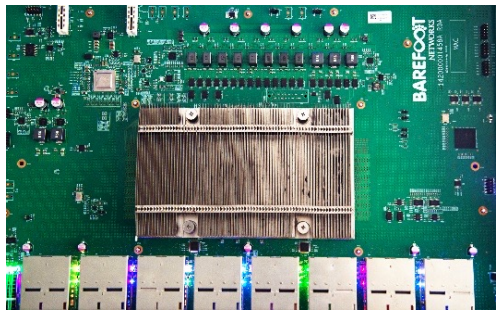
Conceptually, a left / right switch…

- Receives a packet through port <N>
- Decides through which port to send it
  - A *forwarding* decision

+ Some “real world” considerations

Real World Switches

- High Throughput Switch Silicon: 6.4Tbps (64x100G) – 12.8Tbps (32x400G) Top of Rack Switches
  - E.g. Broadcom Tomahawk III, Barefoot Tofino, Mellanox spectrum II
- High Throughput Core Switch System: >100Tbps
  - E.g. Arista 7500R series, Huawei NE5000E, Cisco CRS Multishelf



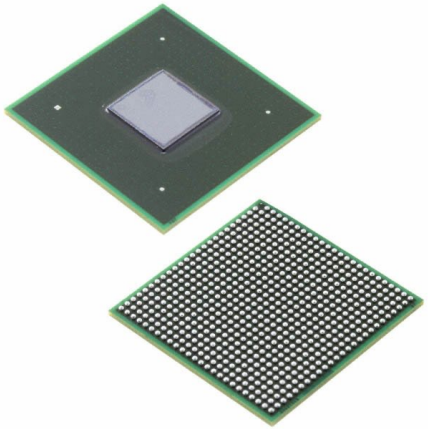
Real World Switches

- Low latency switch (Layer 1): ~5ns fan-out, ~55ns aggregation
- Low latency switch (Layer 2): 95ns - 300ns
  - Examples: g. Mellanox spectrum II, Exablaze Fusion
- Low latency NIC: <1us (loopback)
  - E.g. Mellanox Connect-X, Solarflare 8000, Chelsio T6, Exablaze ExaNIC

- Low latency switches don’t always support full line rate!

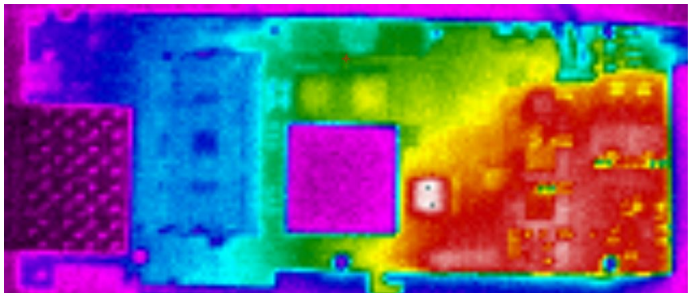
Real World Switch Silicon in Numbers

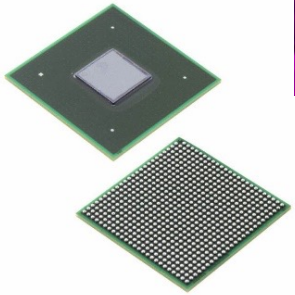
- Over 7 Billion Transistors
- Silicon size: 400 to 600 square mm
- Clock Rate: ~1GHz (typical)
- Packet Rate: ~10 Billion packets per second
- Buffer Memory: ~16MB-30MB on-chip
- Ports: Up to 256
- Power: ~100W-300W
- 2017 Numbers



What Drives The Architecture of a Switch?

- Cost 
- Manufacturing limitations (e.g. maximum silicon size)
- Power consumption
- General purpose or user specific?
- I/O on the package
- Number of ports:
  - Front panel size (24,32,48 ports in 19inch rack)
  - MAC area





Packet Rate as a Performance Metric

- Bandwidth is misleading
  - For example: full line rate for 1024B packets
    but not for 64B packets…
- Packet Rate: how many packets can be processed every second?
- Unit: packets per second (PPS)

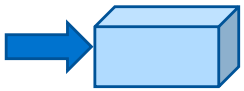
- An easy way to calculate the packet rate:

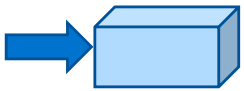
  (Clock Frequency) / (Number of Clock Cycles per Packet)

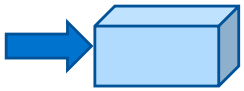
Switch Internals 101

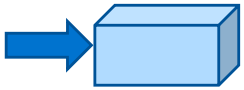
What defines the architecture of a switch?

Input Ports



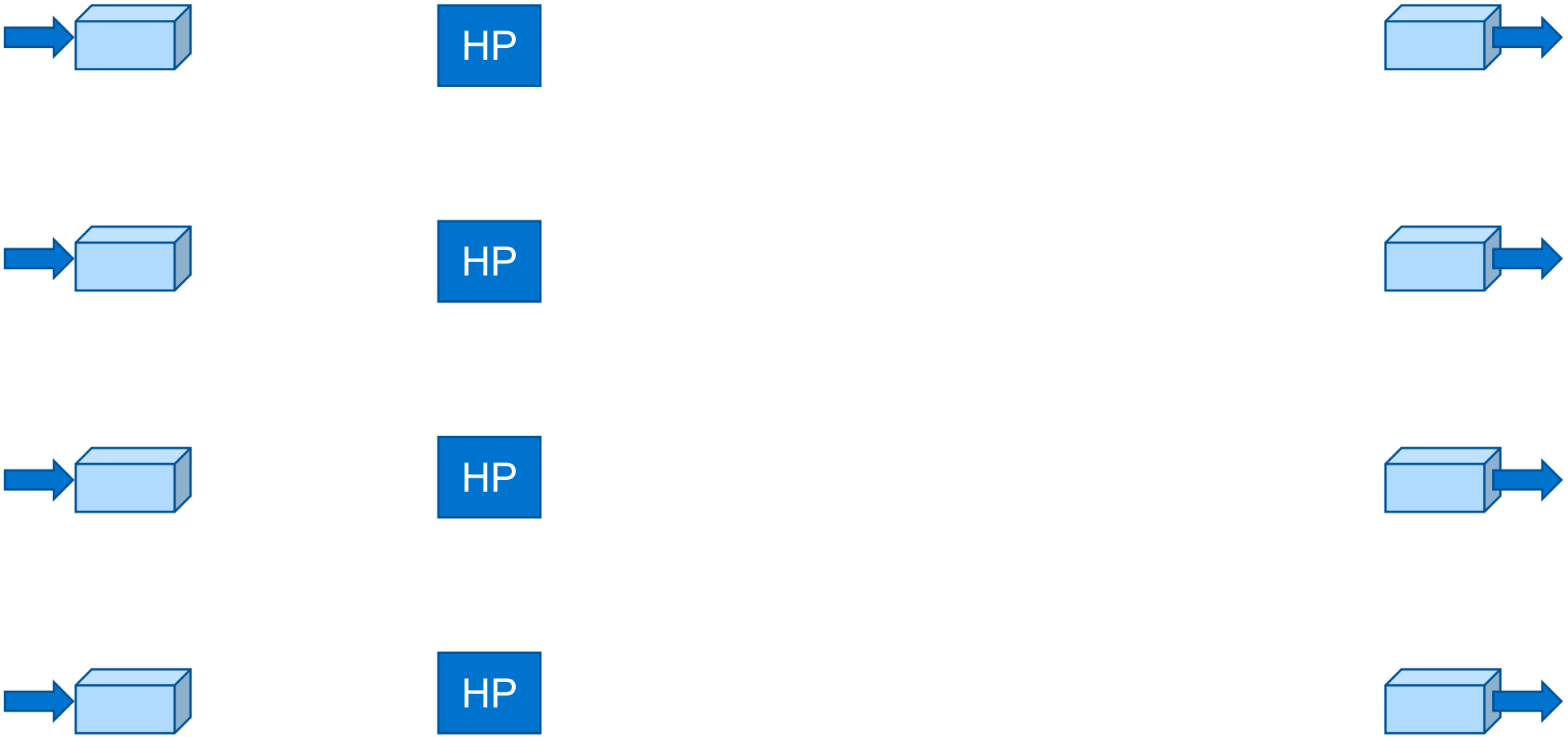




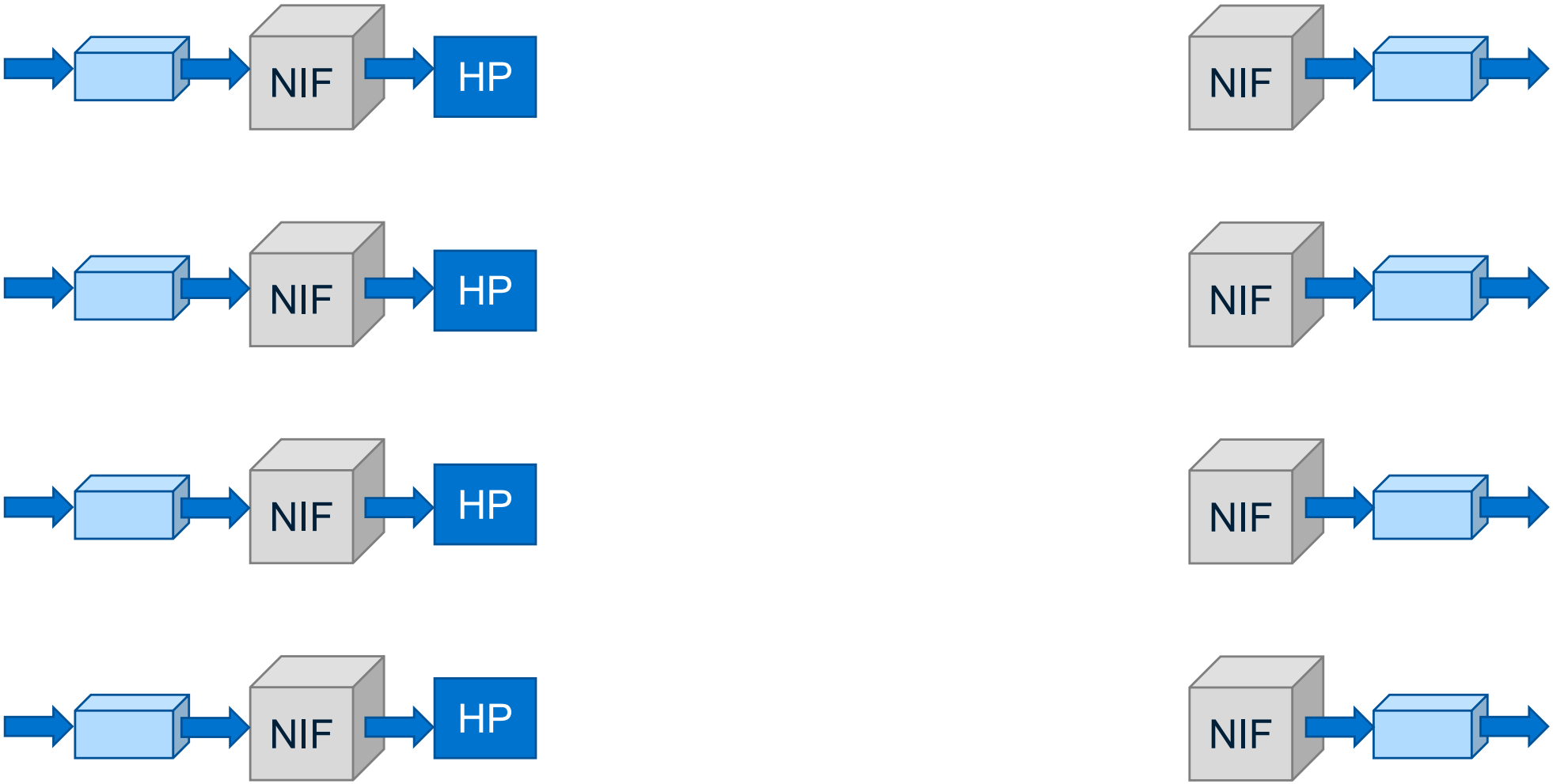


Output Ports

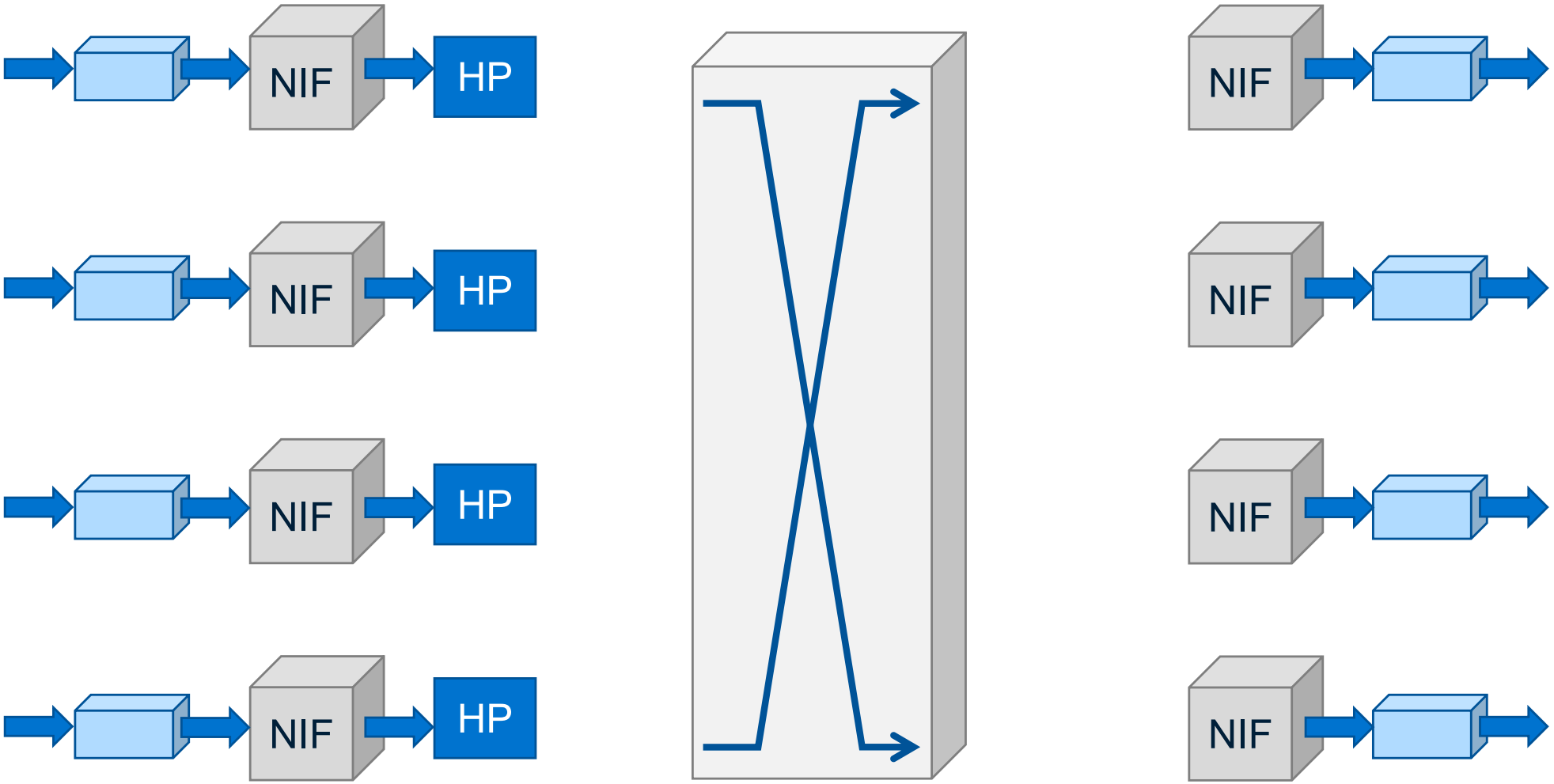
Header Processing



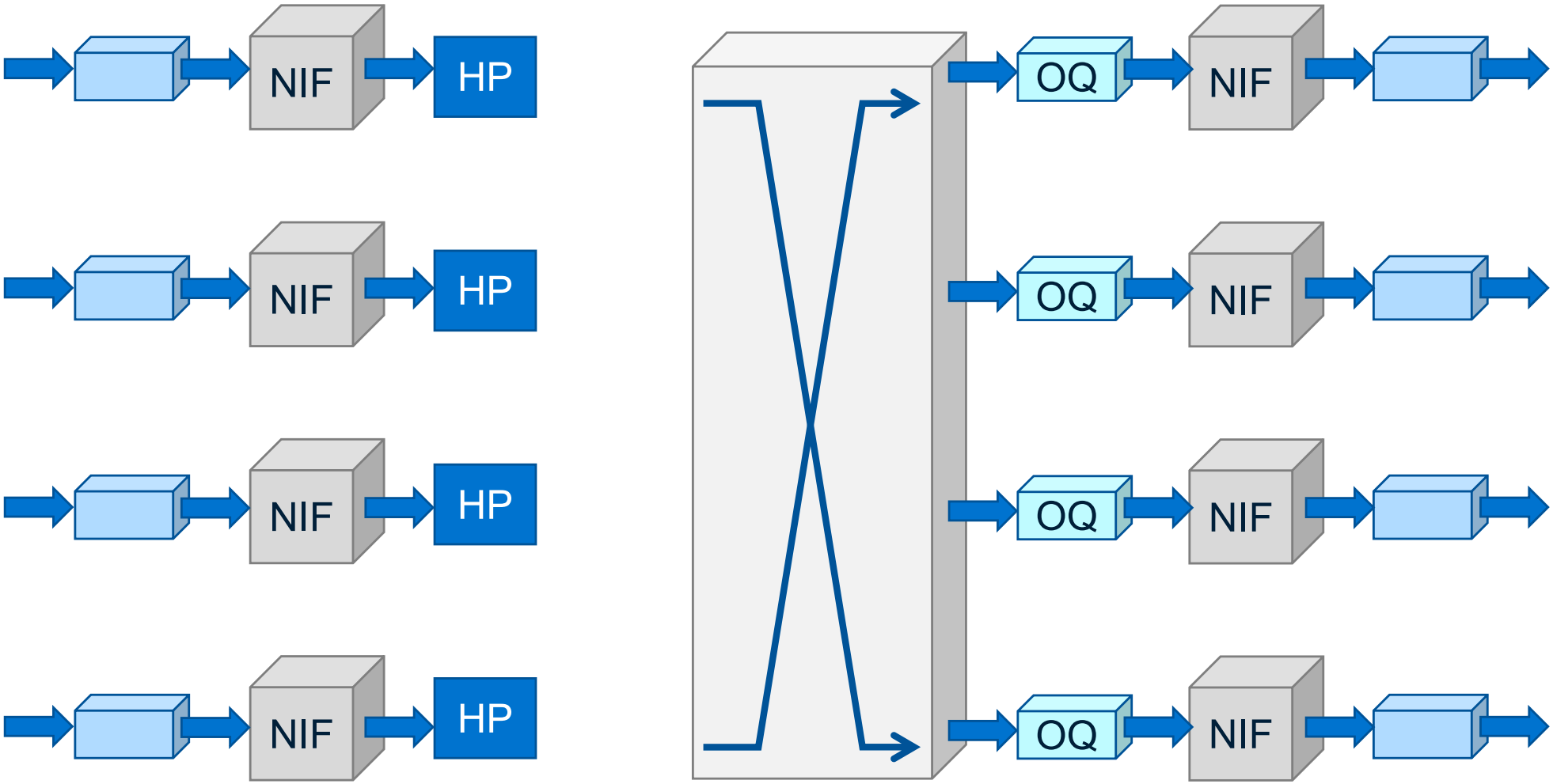
Network Interfaces



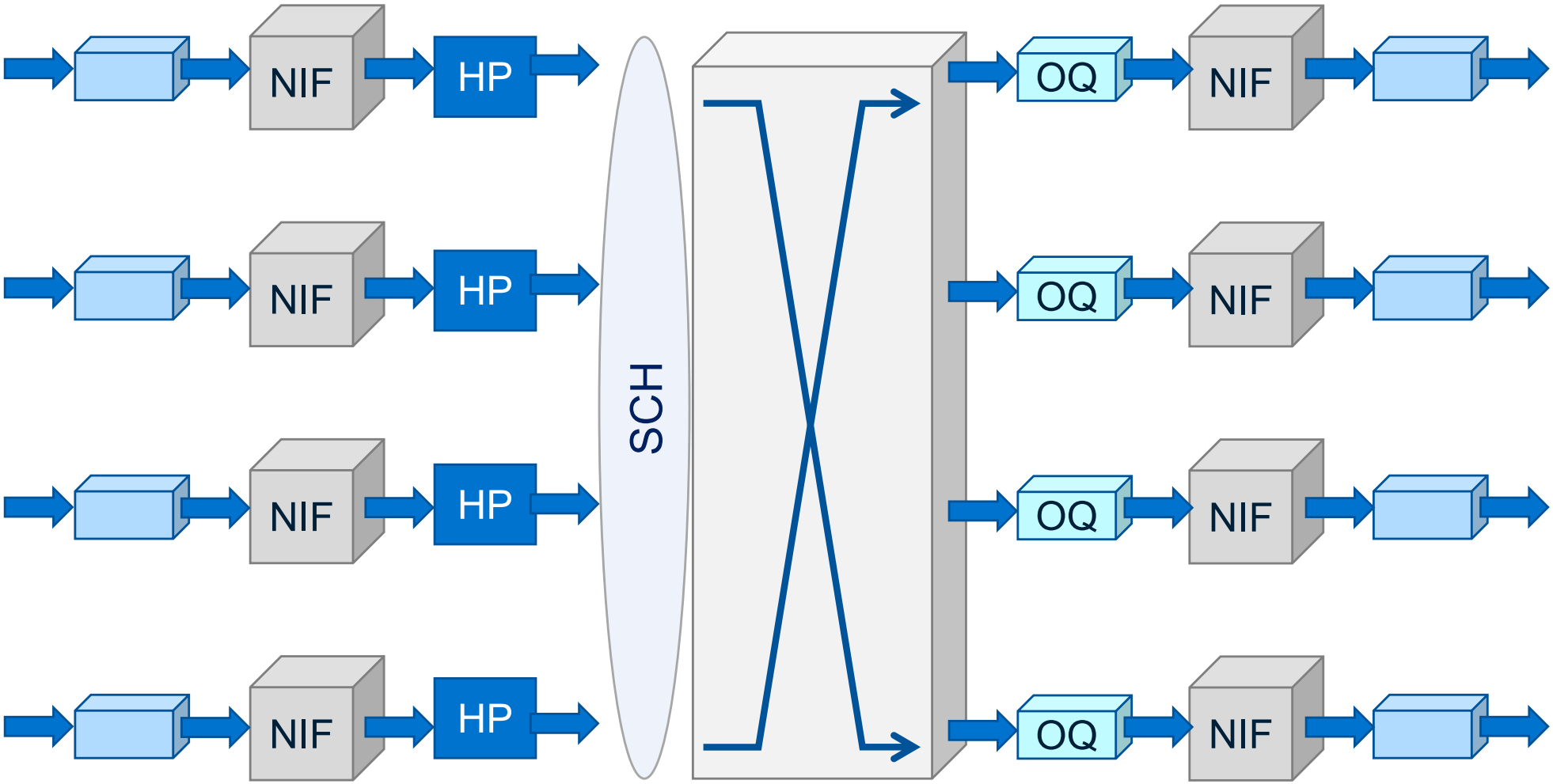
Switching



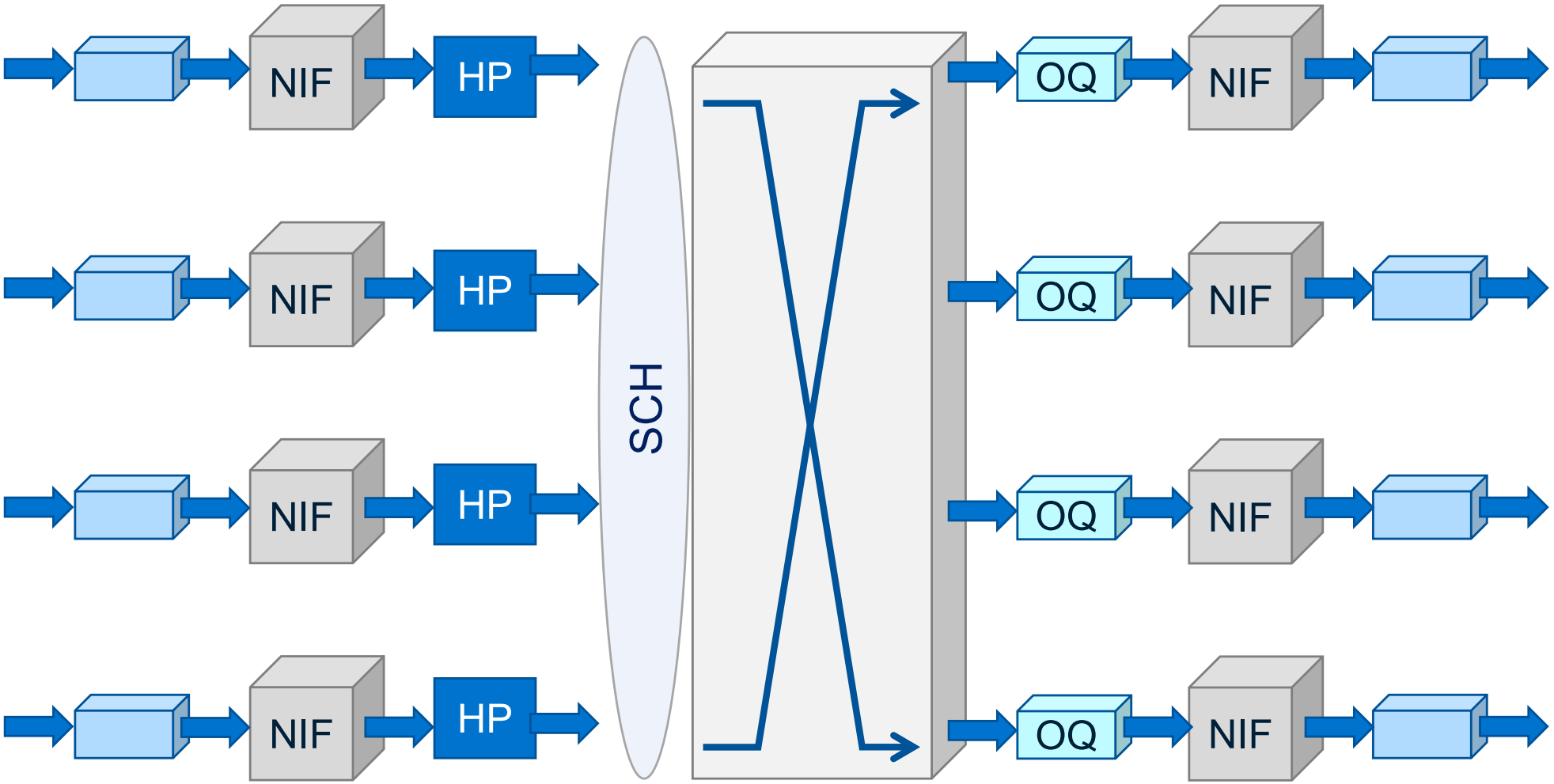
Output Queues



Scheduling



Is This A Real Switch?



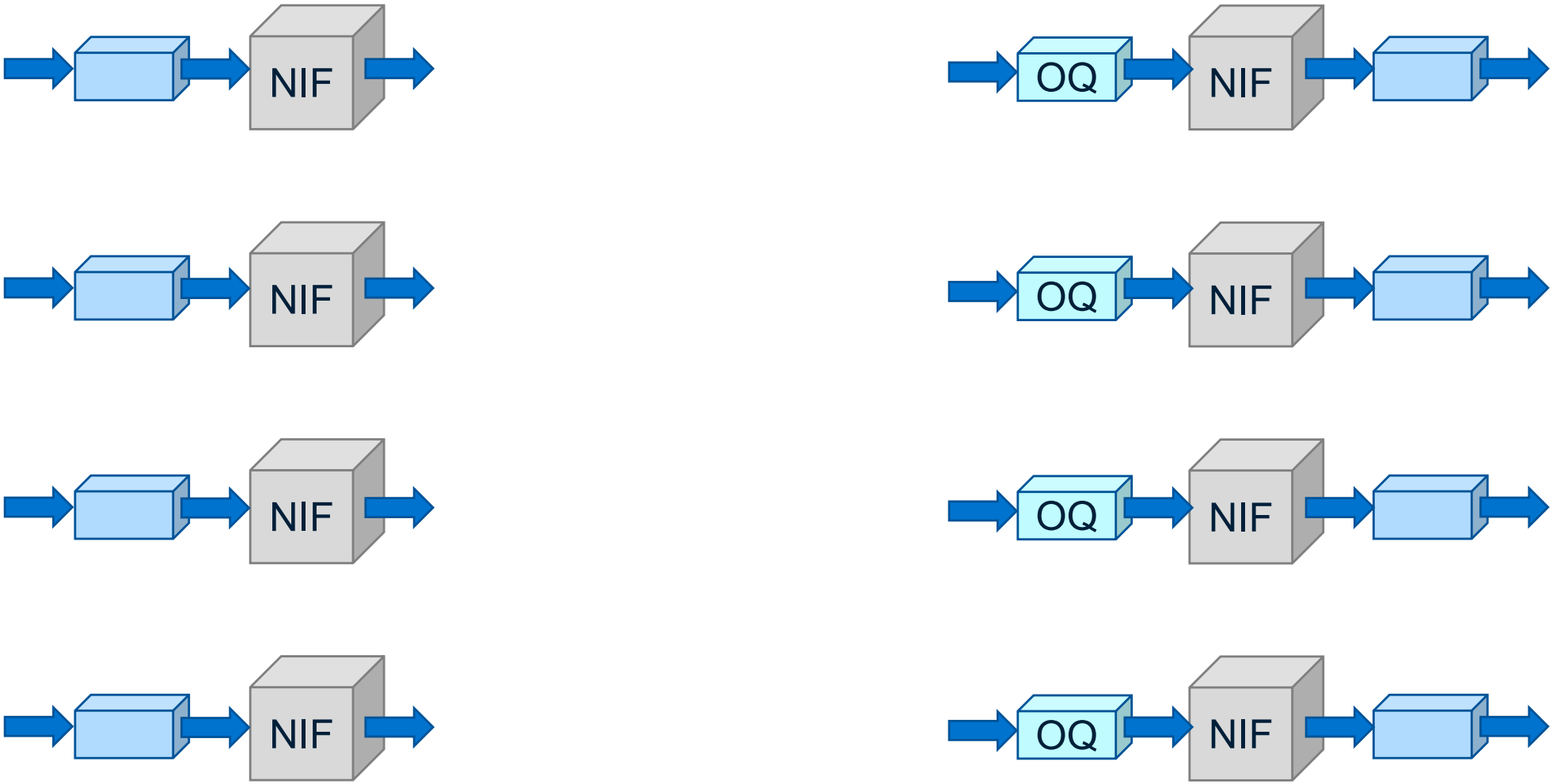
Recall What Drives Real World Switches

- Cost
- Power
- Area

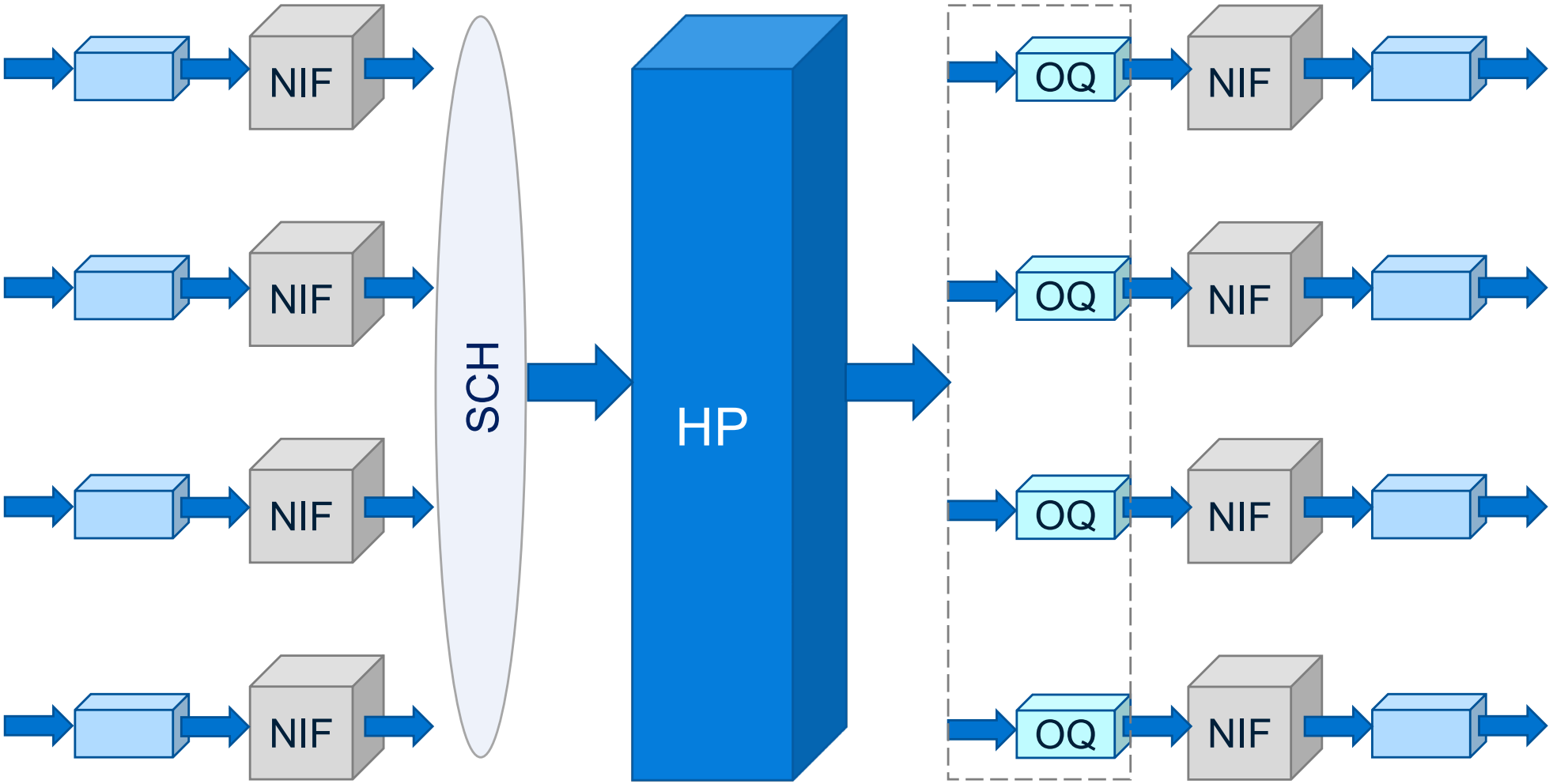
Sharing Resources Is Good!

- Single header processor (if possible)
- Shared memories
- No concurrency problems
  - Also no need to synchronise tables, no need to send updates, ….

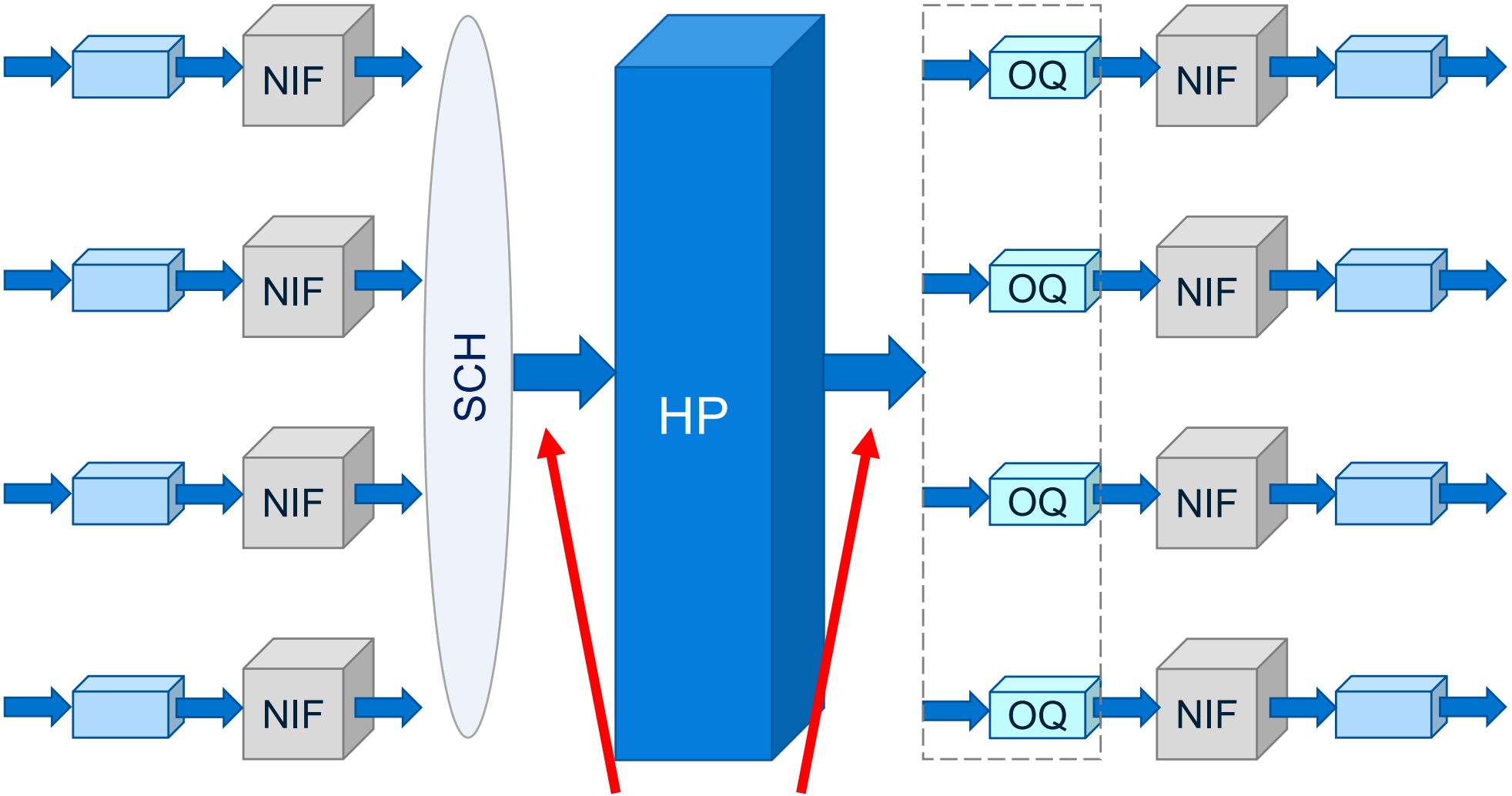
Rethinking The Switch Architecture



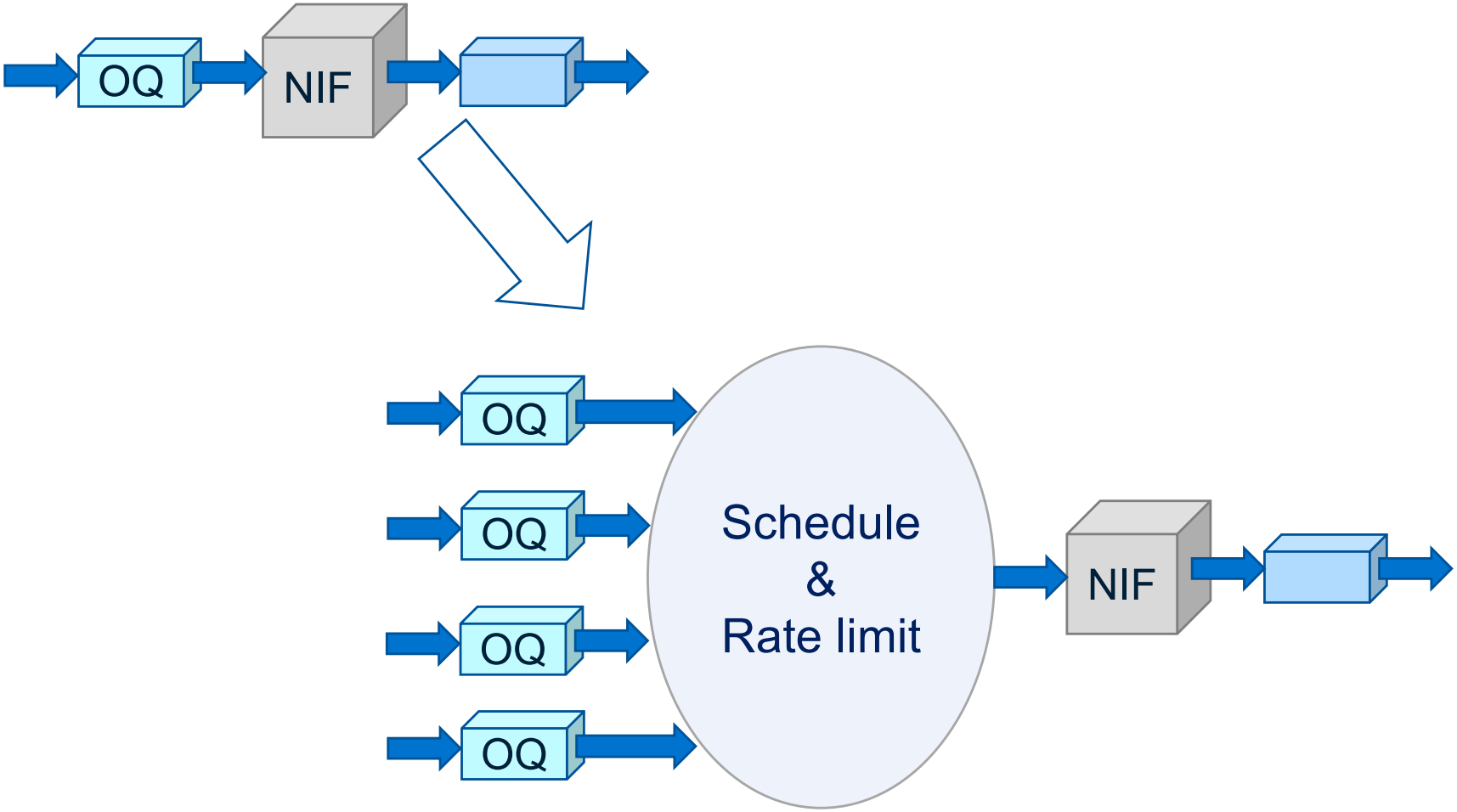
Rethinking The Switch Architecture



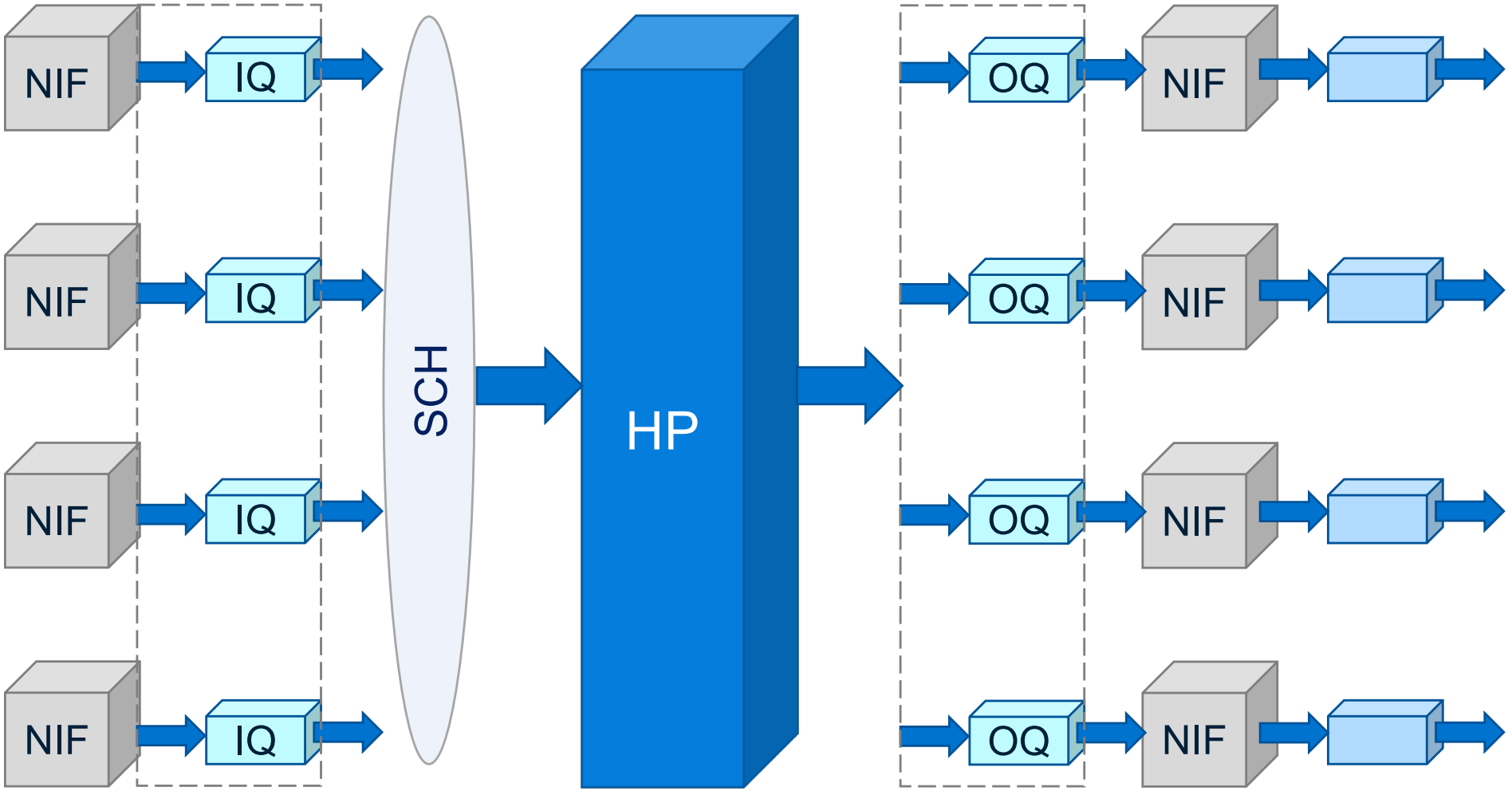
Where Is The Switching?



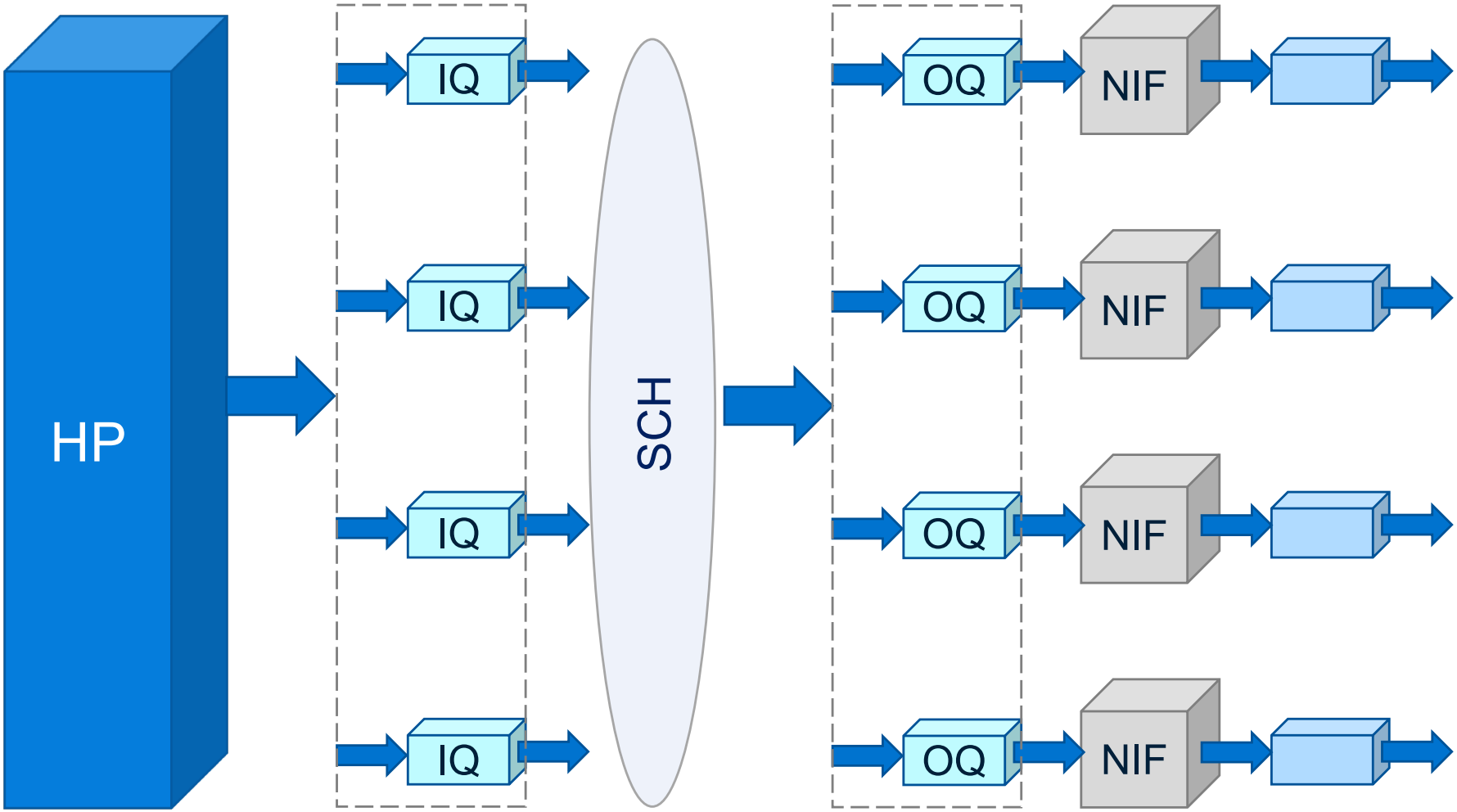
Output Queueing



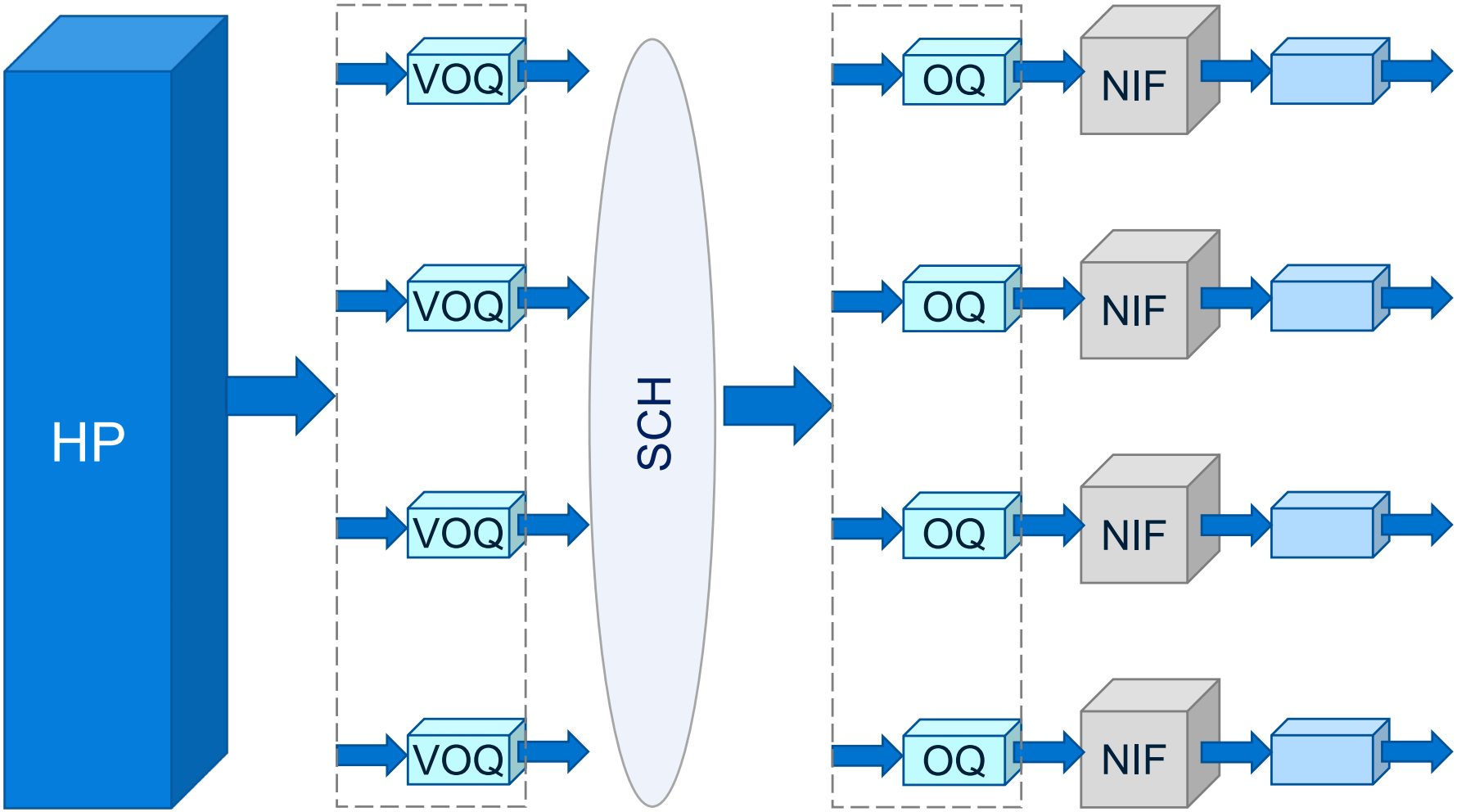
Input Queueing



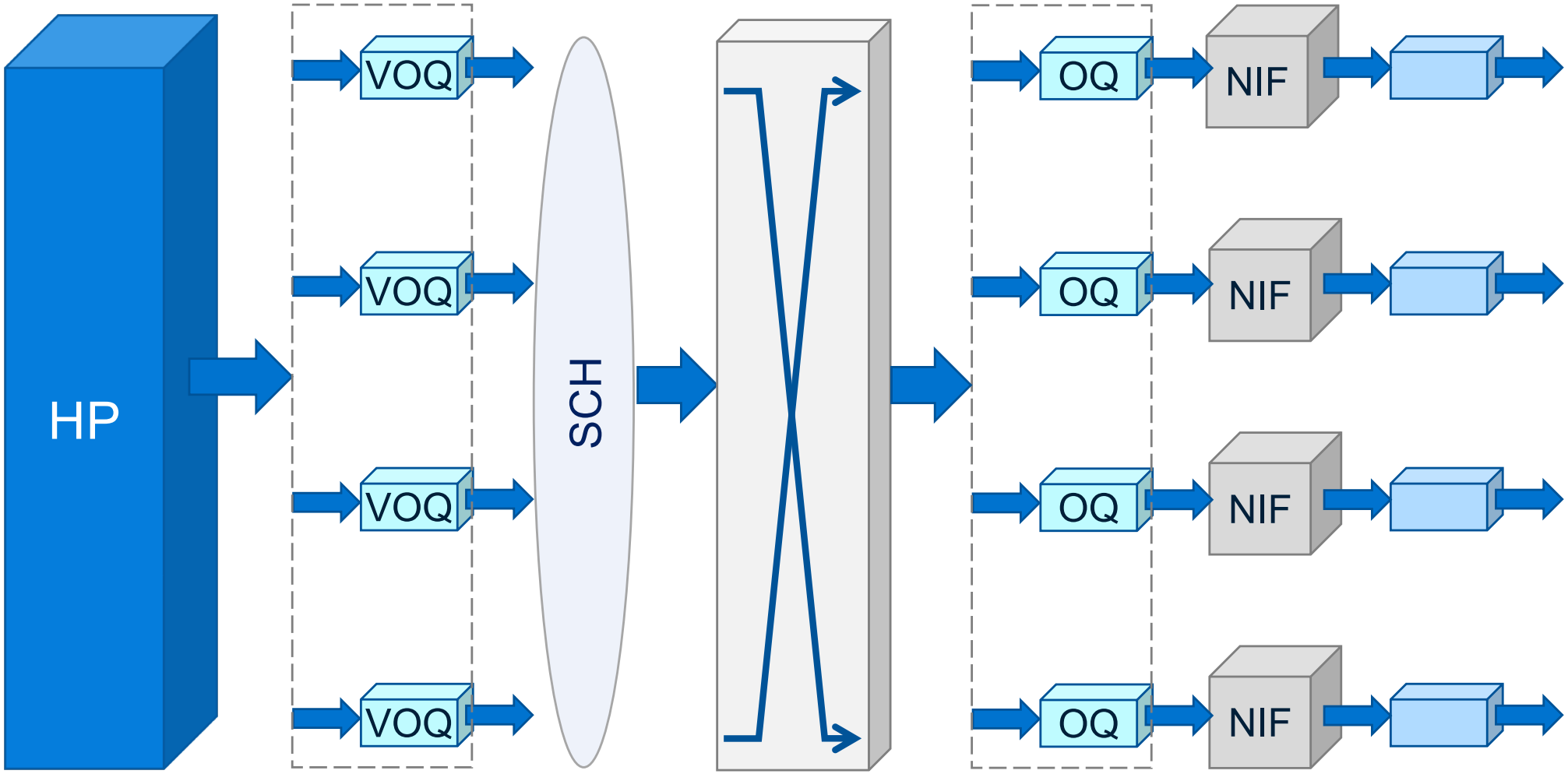
Virtual Output Queueing



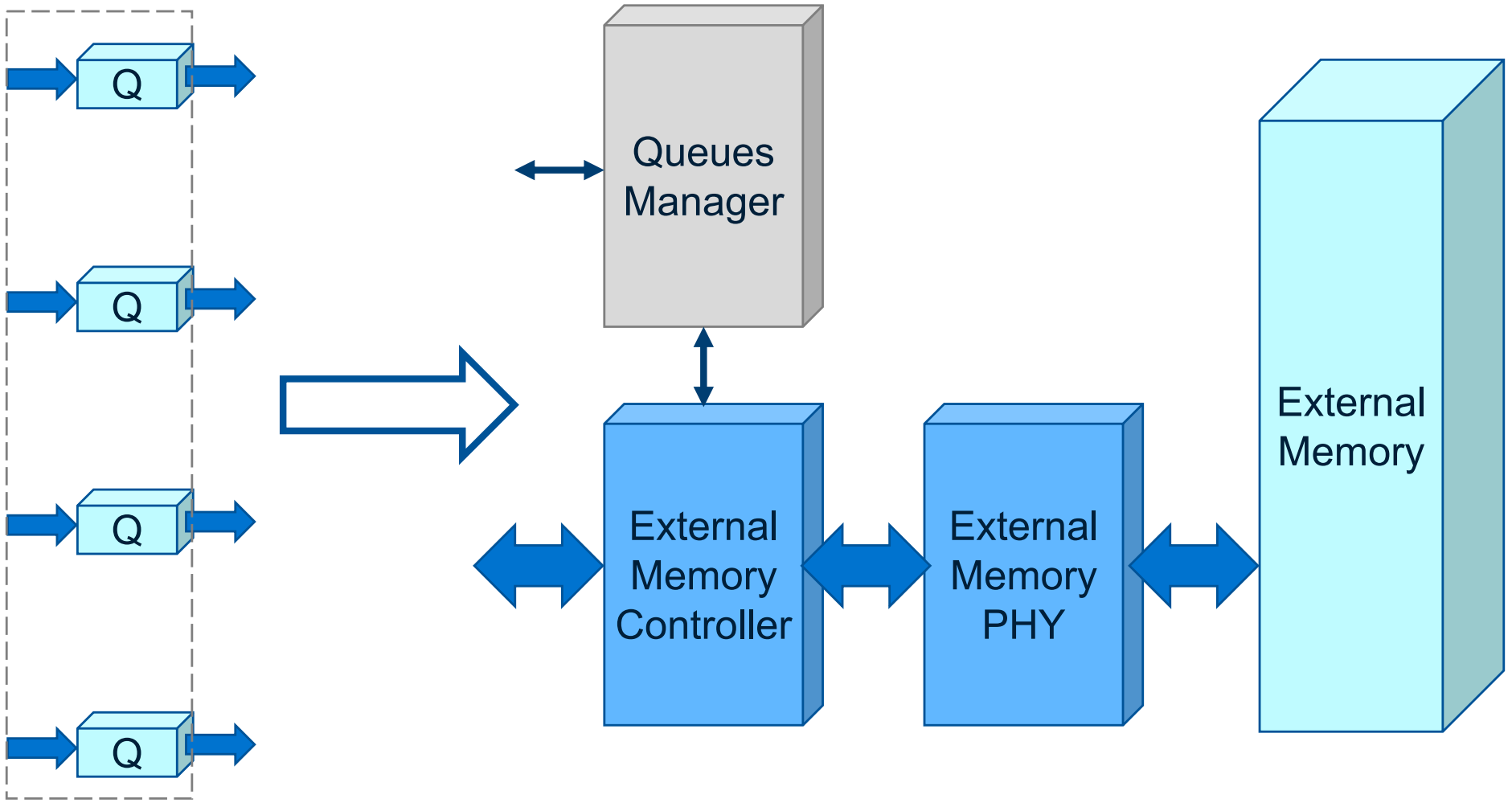
Virtual Output Queueing



Virtual Output Queueing



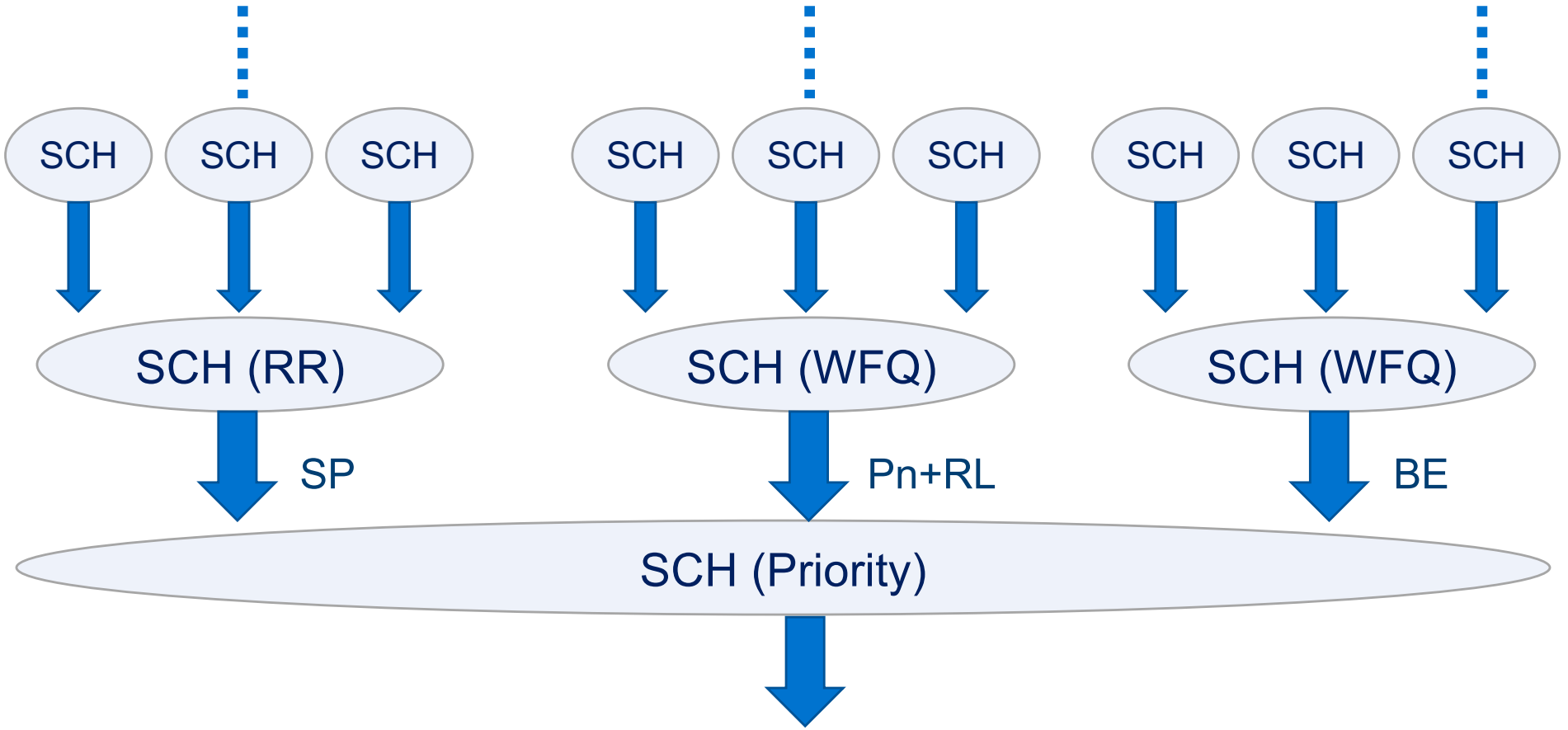
Deep Buffers



Scheduling

- Different operations within the switch:
  - Arbitration
  - Scheduling
  - Rate limiting
  - Shaping
  - Policing
- Many different scheduling algorithms
  - Strict priority, Round robin, weighted round robin, deficit round robin, weighted fair queueing…

Scheduling Hierarchies



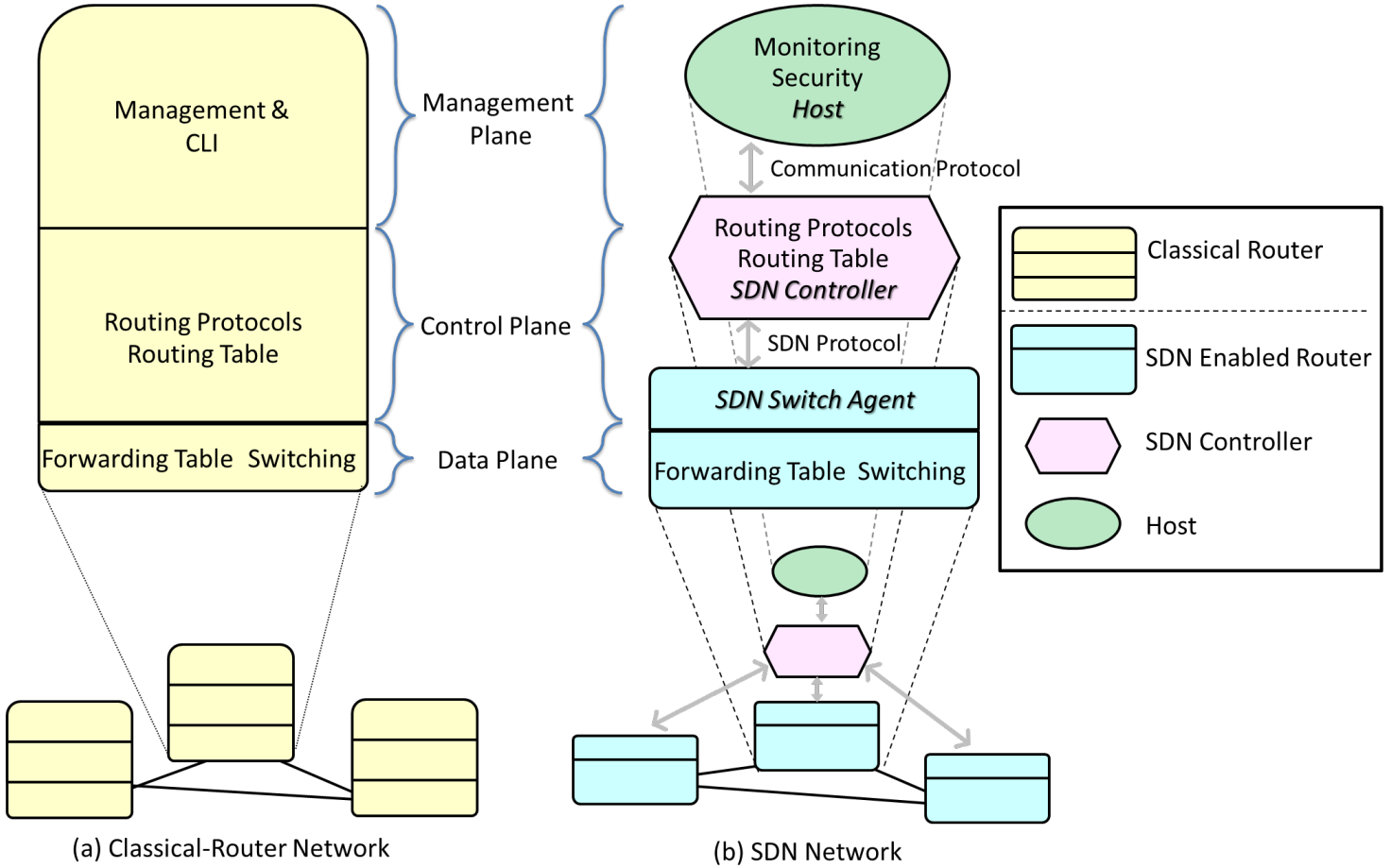
SP – Strict Priority
Pn – Priority <n>

BE – Best Effort
RL – Rate Limiting

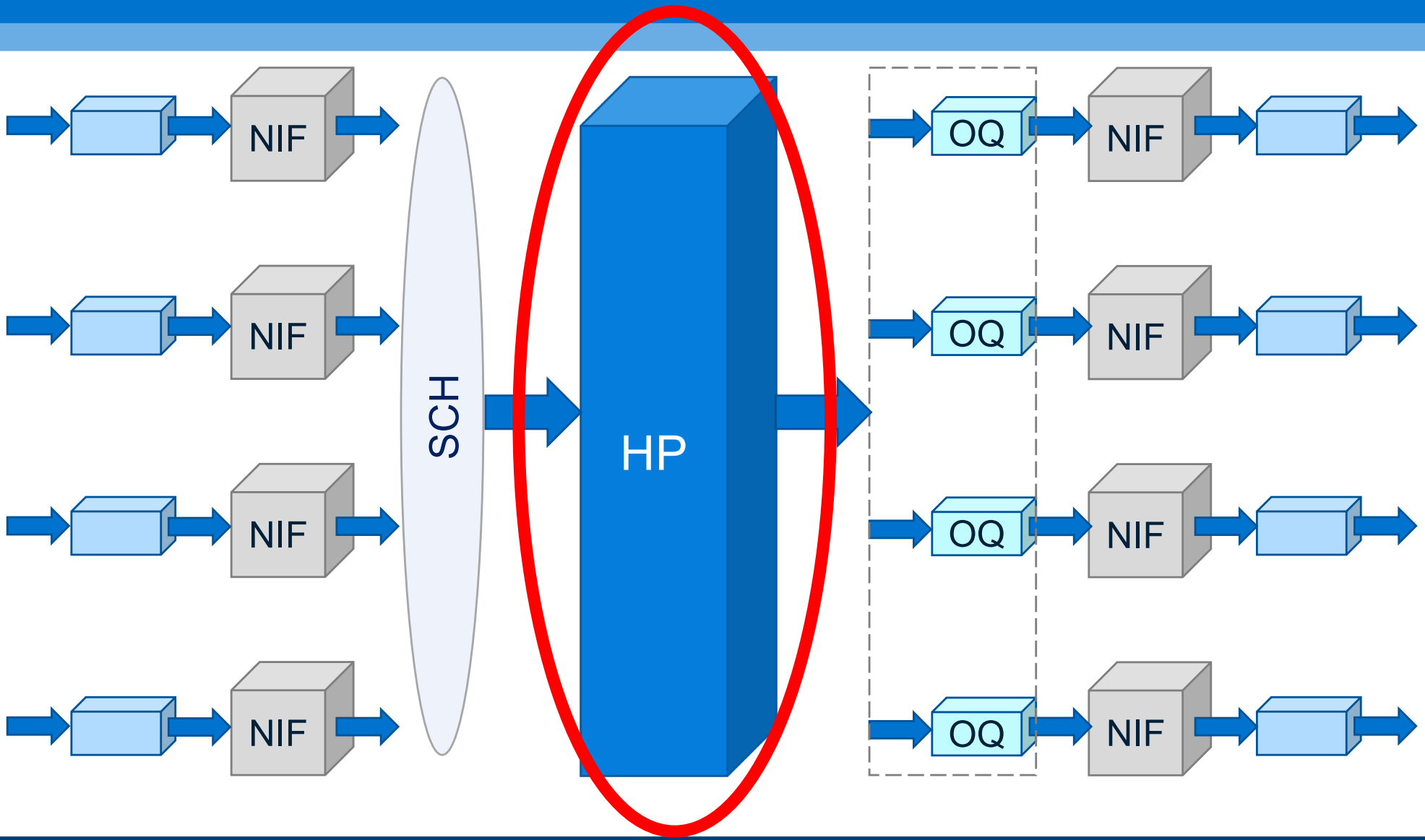
WFQ – Weighted Fair Queueing
RR – Round Robin

Software Defined Networking (SDN)

Key Idea: Separation of Data and Control Planes



Switch Architecture and SDN



High Throughput Switching

- High throughput metrics
- Types of high performance switches
  - cut through vs. store and forward
  - ToR vs. Core switch
  - General purpose vs. proprietary ASIC
- High throughput switch architectures (including silicon vs system vs network)

Bandwidth, Throughput and Goodput

- Bandwidth – how much data can pass through a channel.
- Throughput – how much data actually travels through a channel.
- Goodput is often referred to as application level throughput.

But bandwidth can be limited below link's capacity and vary over time, throughput can be measured differently from bandwidth etc…..

Speed and Bandwidth

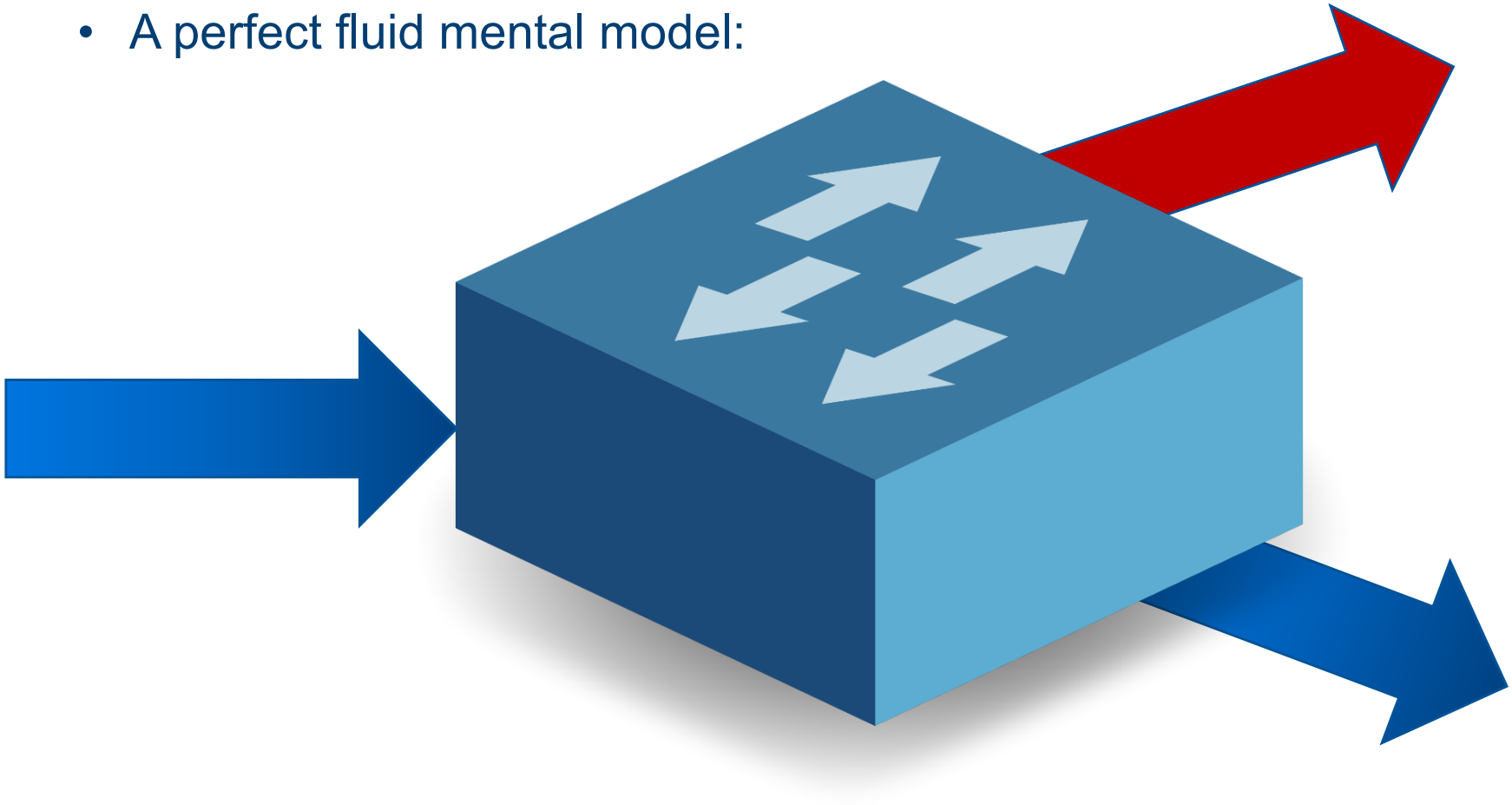
- Higher bandwidth does not necessarily mean higher speed
- E.g. can mean the aggregation of links
  - 100G = 2x50G or 4x25G or 10x10G
  - A very common practice in interconnects

Packet Rate

- Throughput may change under different conditions, e.g. packet size
- Packet Rate: how many packets can be processed in a given amount of time
  - Also changes under different conditions
  - But often provides better insights

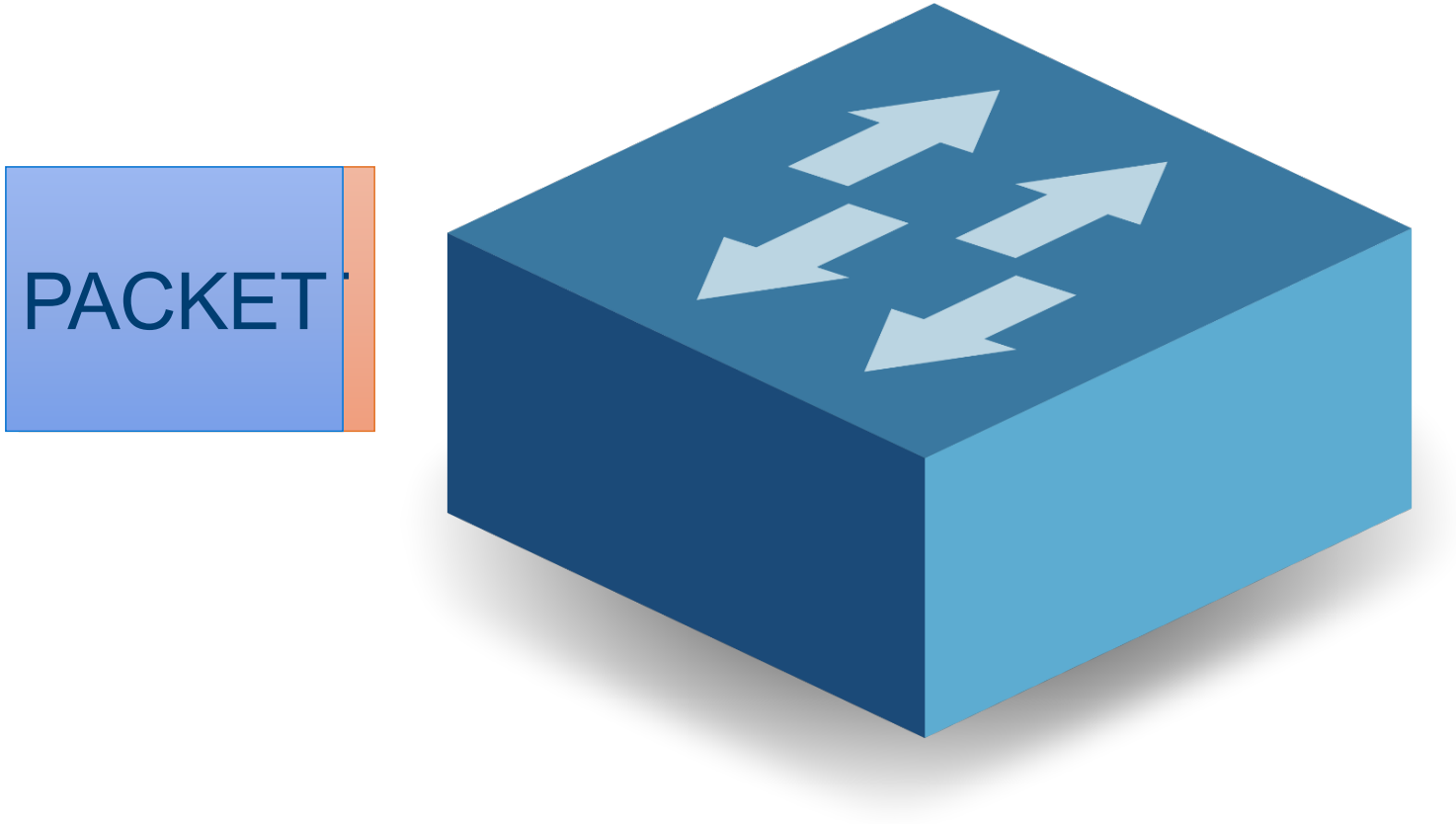
Switch Models

- A perfect fluid mental model:



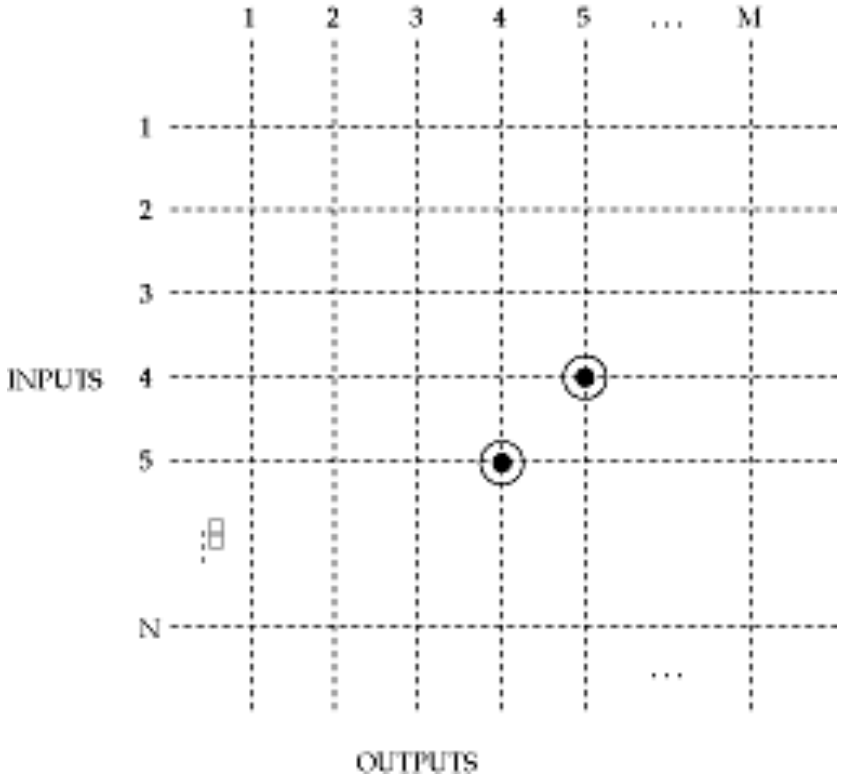
Switch Models

- A single packet mental model:



Circuit Switches

- Input A is connected to output X
- Example: a crossbar
  - Not the only option
- Used mostly in optical switching
  - No header processing!
- But also in electrical switching
  - E.g. high frequency trading (HFT)
- Scheduling is a limiting factor



Packet Switches

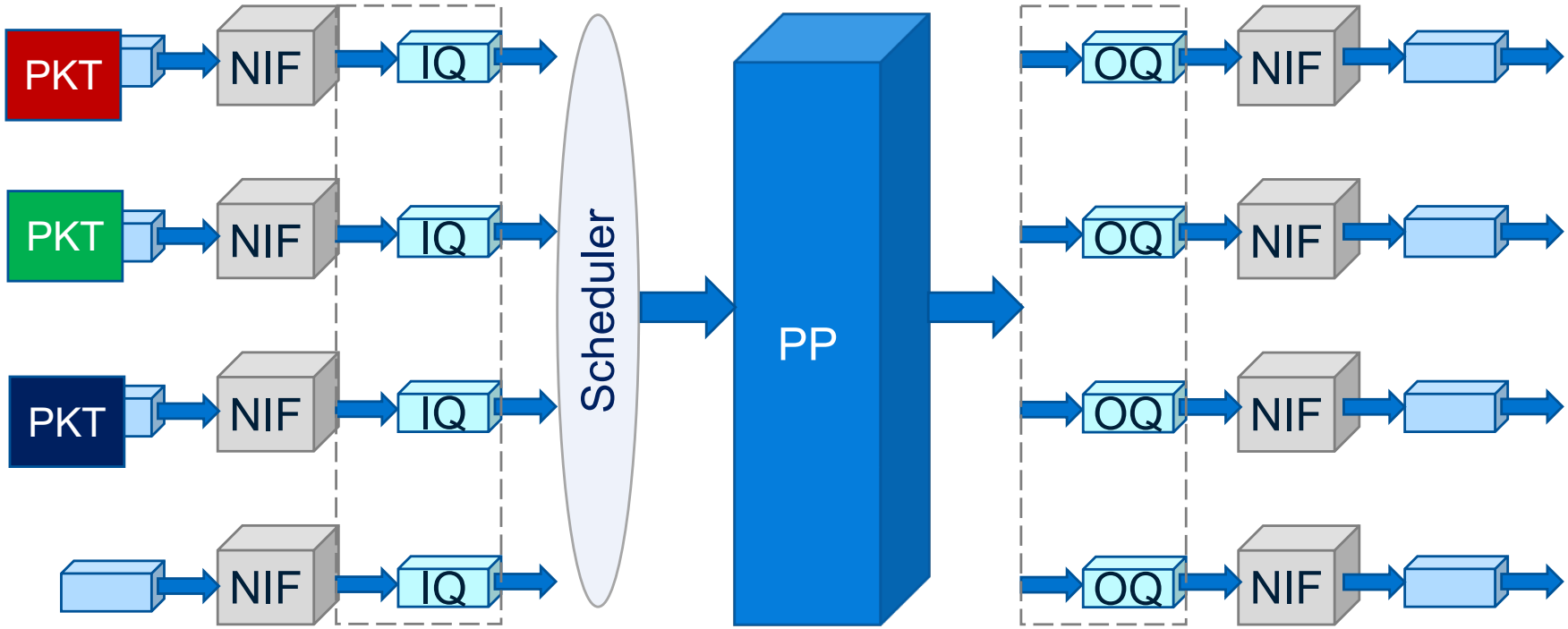
- In a circuit switch:

  The path of a sample is determined at time of connection establishment

- In a packet switch, packets carry a destination field
- Need to look up destination port on-the-fly
  - Two sequential packets may head to a different destination

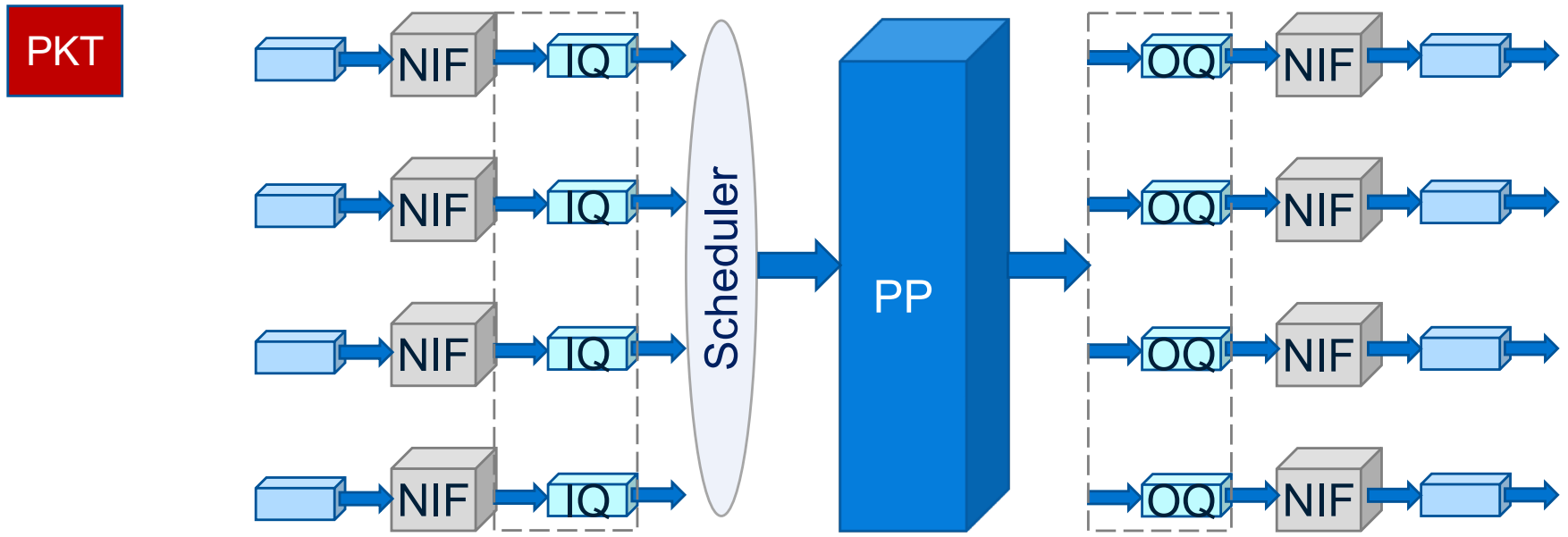
Pipelining

To achieve high throughput, packet switches are pipelined:



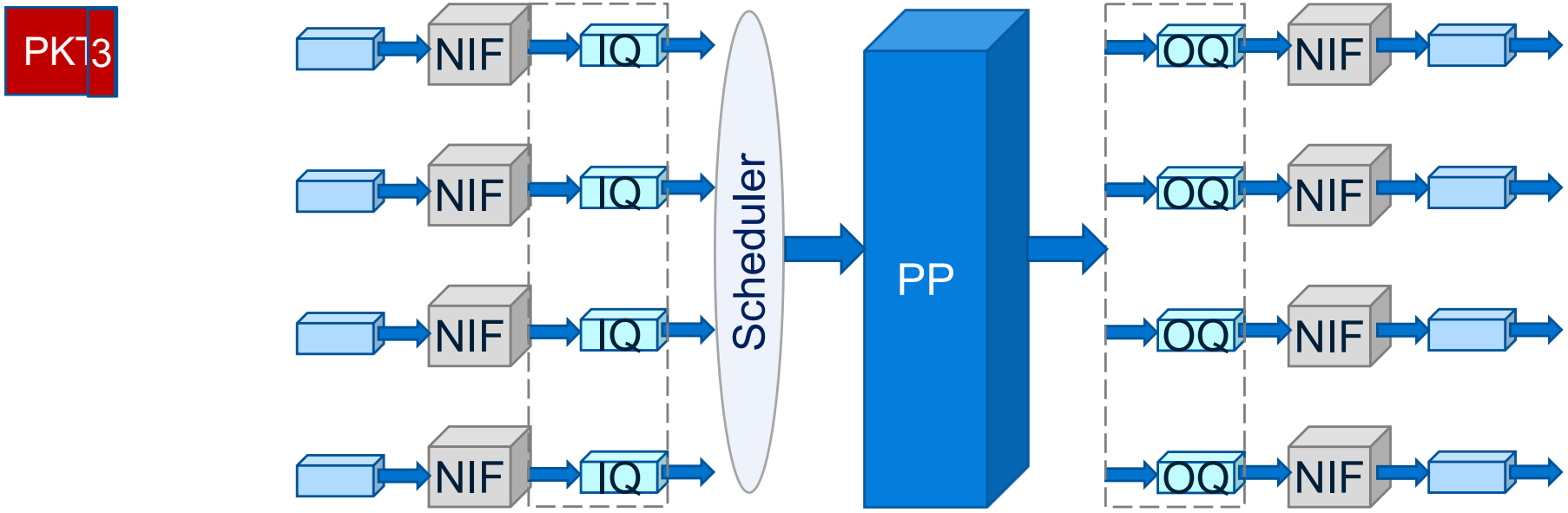
Store and Forward

- Wait for the entire packet to arrive
- Check the FCS, then start processing
  - FCS – frame check sequence, terminates the packet
- Once the packet is checked, it starts propagating through the pipeline
  - Not necessarily the entire packet



Cut Through

- Start processing the packet as soon as the first chunk arrives
  - Do not wait for the FCS
- If FCS error is detected, the packet is dropped somewhere along the pipeline

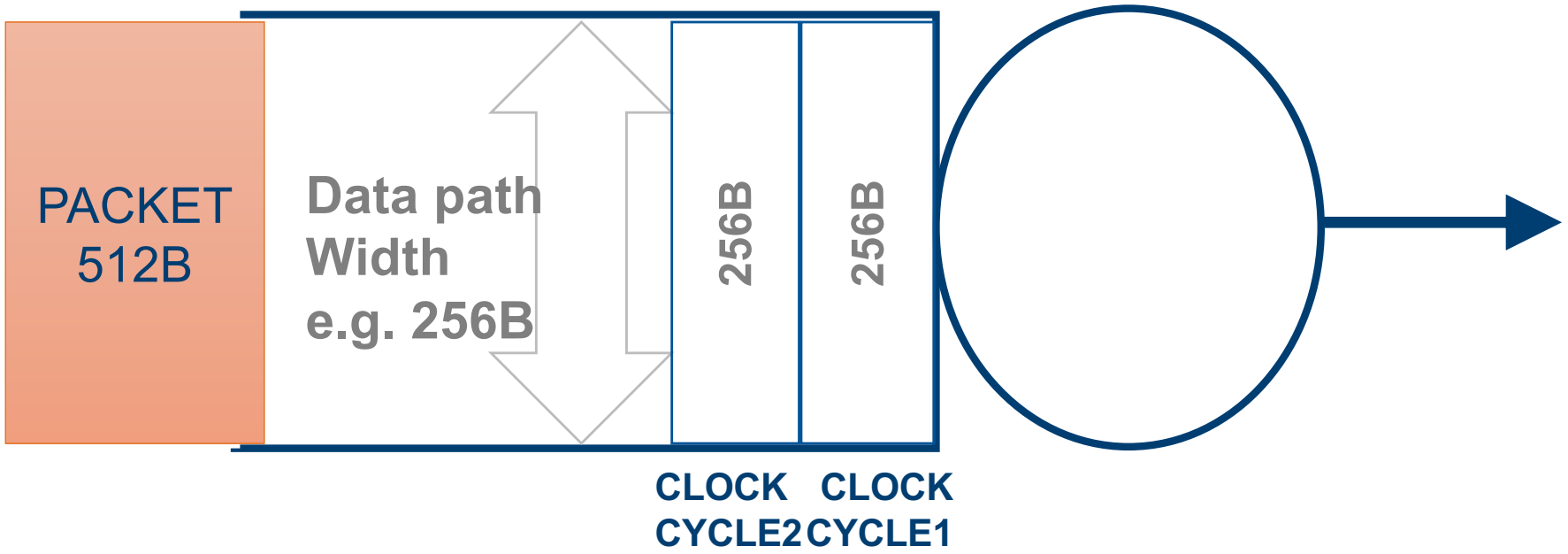


Measuring Performance

- Bandwidth: number of bits (or bytes) through the channel every unit of time
  - One way to calculate: *bus width* $\times$ *clock frequency*

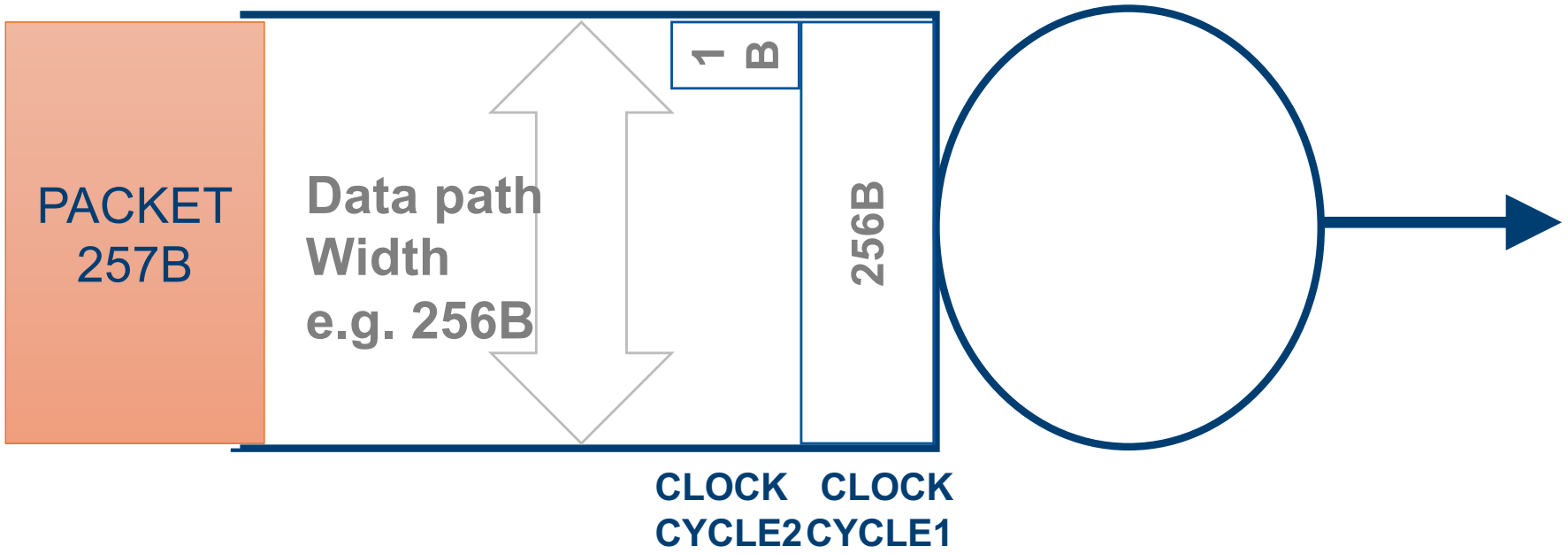
Measuring Performance

Throughput = clock frequency x bus width ?



The Truth About Switch Silicon Design

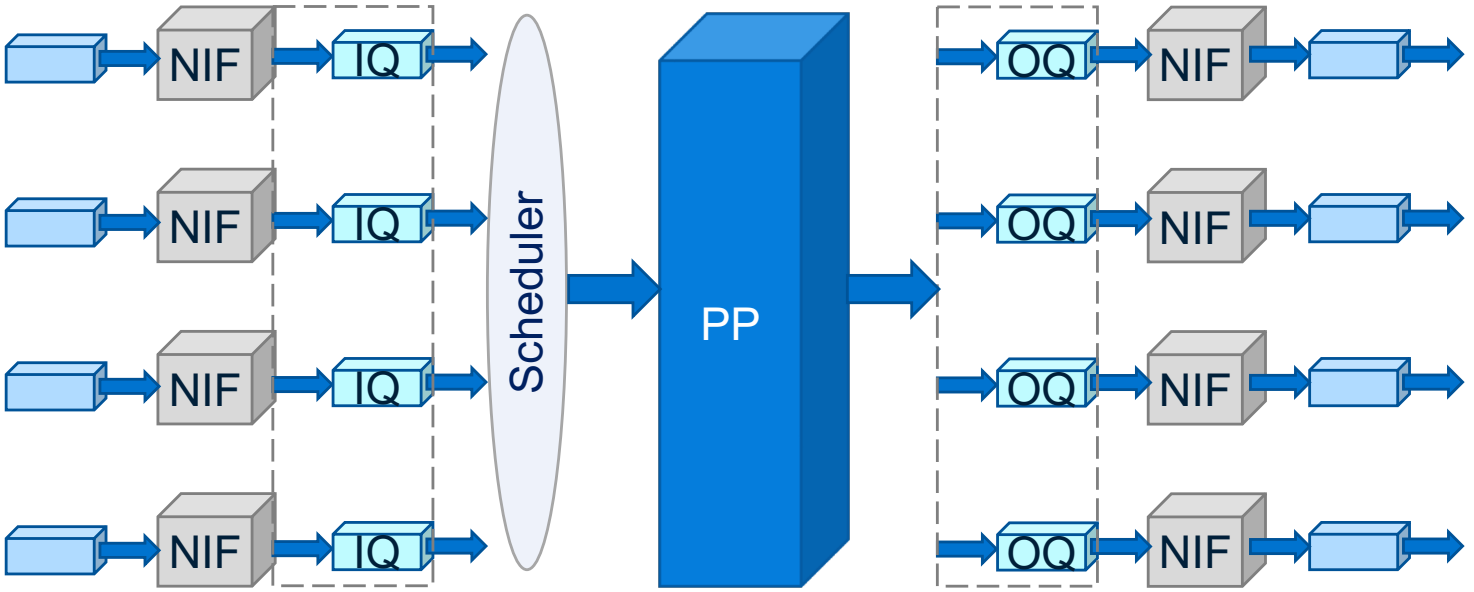
Throughput ≠ clock frequency x bus width !



Low Latency Switches

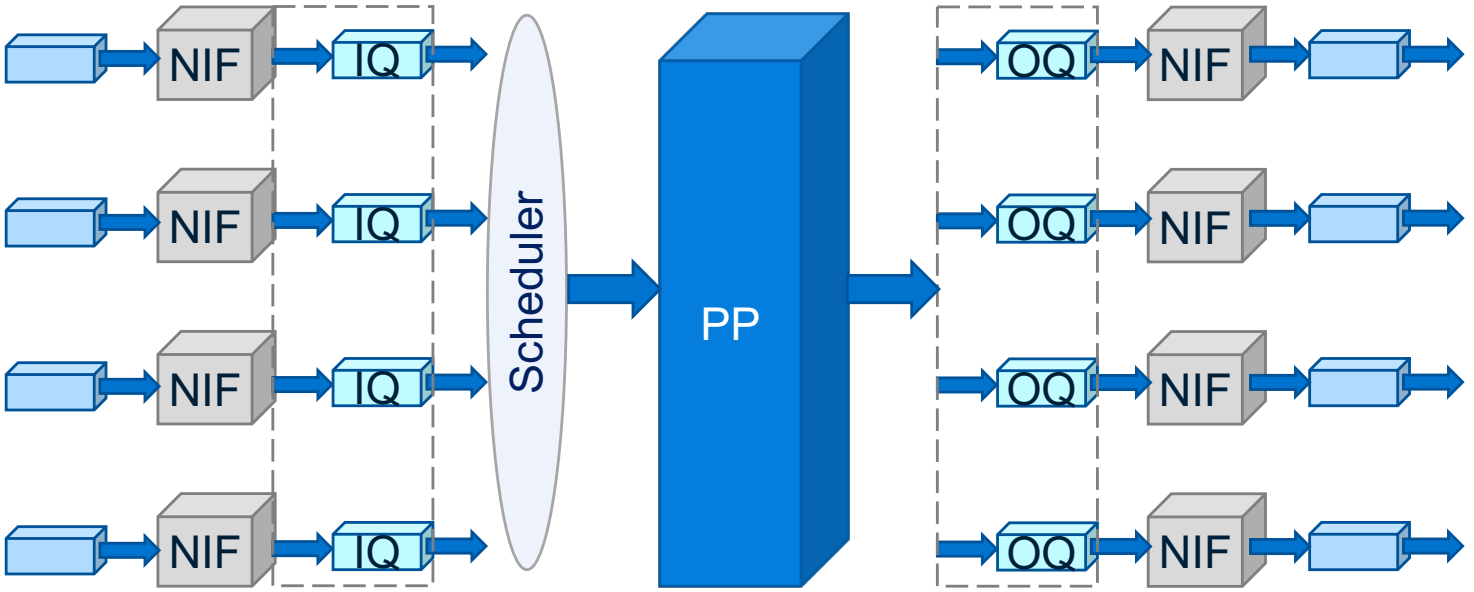
How to lower the latency of a switch?

- Obvious option 1: Increase clock frequency
  - E.g. change core clock frequency from 100MHz to 200MHz
  - Half the time through the pipeline



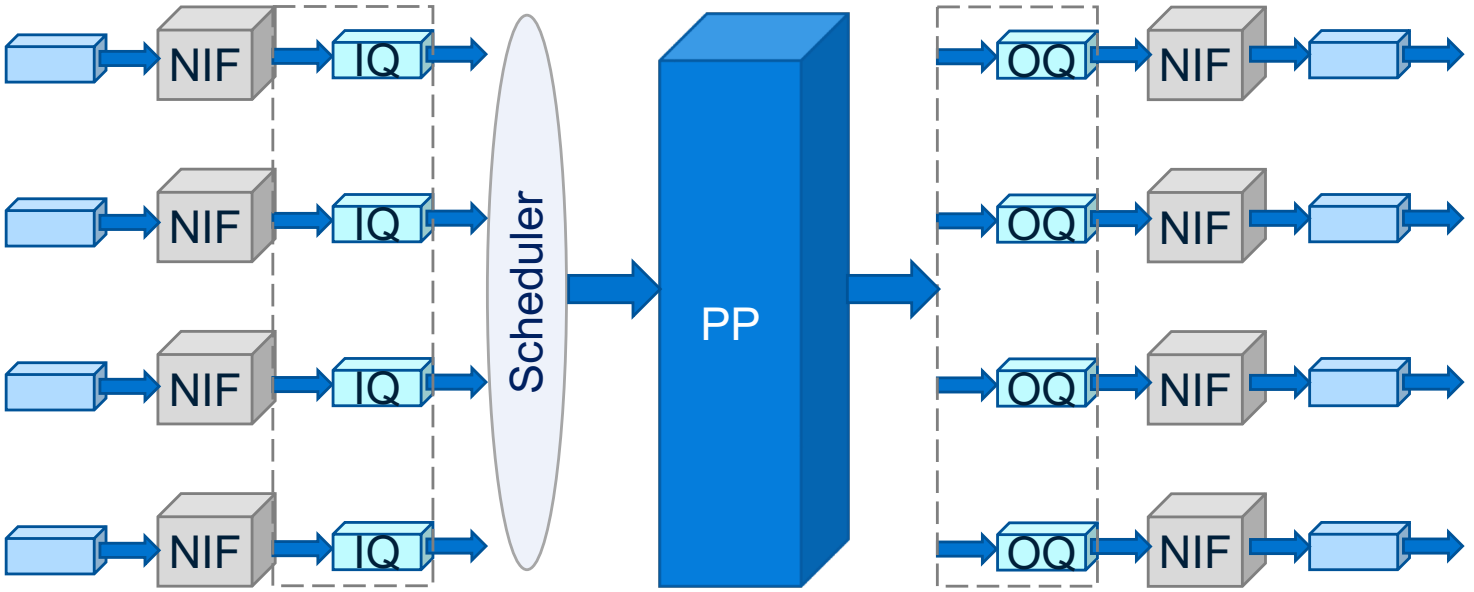
How to lower the latency of a switch?

- Obvious option 1: Increase clock frequency
- Limitations:
  - Frequency is often a property of manufacturing process
  - Some modules (e.g. PCS) must work at a specific frequency (multiplications)



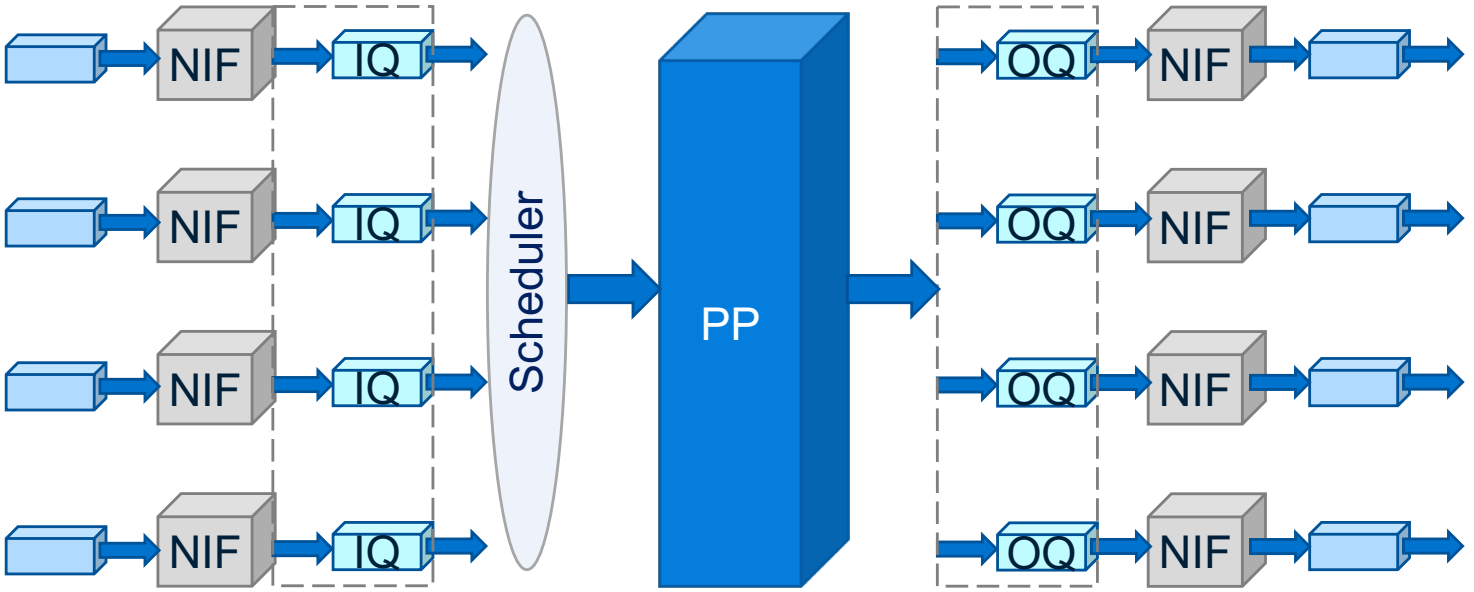
How to lower the latency of a switch?

- Obvious option 2: Reduce the number of pipeline stages
  - Can you do the same in 150 pipeline stages instead of 200?
  - Limitation: hard to achieve.



How to lower the latency of a switch?

- Can we achieve ~0 latency switch?
  - Is there a lower bound on switch latency?



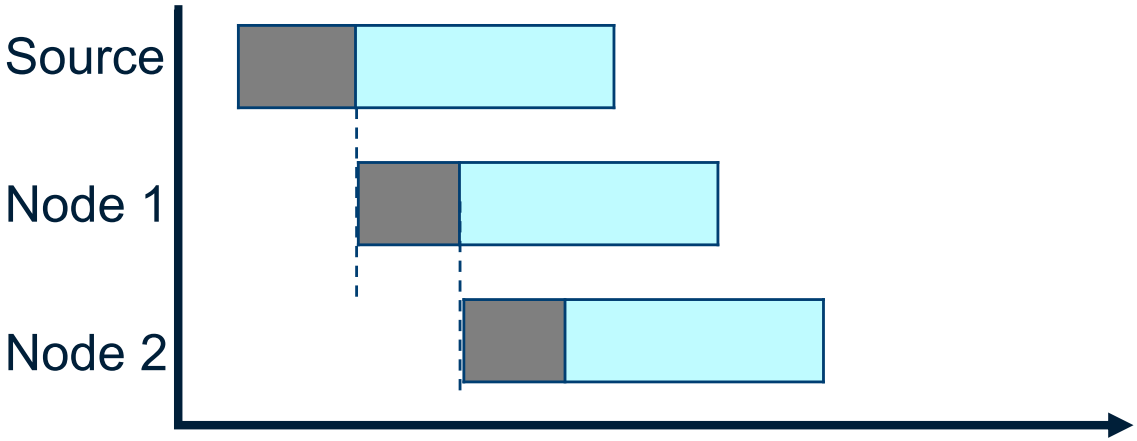
Cut Through Switching

Cut Through Switch

- Cut through switch ≠ Low latency switch
  - A cut through switch can implement a very long pipeline…
- But:
  - For the smallest packet, the latency is ~same
  - As packet size grows, latency saving grows

What is a cut-through switch?

- Kermani & Kleinrock, “Virtual cut-through: A new computer communication switching technique”, 1976
- “when a message arrives in an intermediate node and its selected outgoing channel is free (just after the reception of the header), then, in contrast to message switching, the message is sent out to the adjacent node towards its destination before it is received completely at the node; only if the message is blocked due to a busy output channel is a message buffered in an intermediate node.”

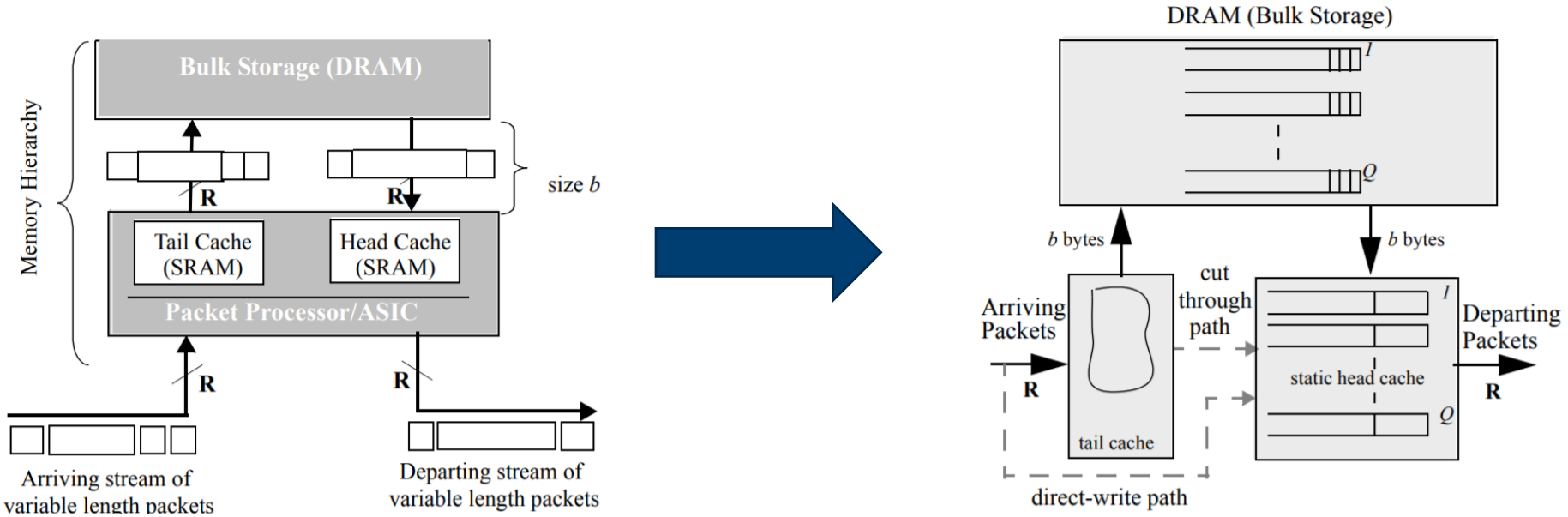


What is a cut-through switch?

- Past (far back):
  - Networks were slow
  - Memory was fast
  - Writing packets to the DRAM took “negligible” time
- With time:
  - Networks become faster
  - Memory access time is no longer “negligible”

What is a cut-through switch?

- Sundar, Kompella, and McKeown. "Designing packet buffers for router line cards." *2002.*



What is a cut-through switch?

- But what does a REAL silicon implementation looks like?
- Tip 1: search for patents on Google Scholar
- Tip 2: read *carefully* performance evaluation reports
  - We'll discuss some examples next (time permitting).

Latency considerations within modules

Network Interfaces

- Data arrives at (up to) ~50Gbps per link.
- Let us ignore clock recovery, signal detection etc.
- Feasible clock rate is ~1GHz
- But if data rate is $\times$50 times faster…

- Observation: data bus width will be no less than incoming data rate and feasible clock rate

Network Interfaces

- Line coding often directs the bus widths:
  - E.g., 8b/10b coding led to bus widths of 16b (20b) or 64b (80b)
- A port is commonly an aggregation of multiple serial links
  - 10G XAUI = 4 × 3.125Gbps
  - 100G CAUI4 = 4 × 25Gbps
  - 400G PSM4 = 8 × 50Gbps
  - Need to take care of aligning the data arriving from multiple links on the same port.

Network Interfaces

- Role: check the validity of the packet (e.g., FCS)
- What to do if an error is detected?
  - Forward an error using a “fast path”
  - Mark the last cycle of the packet
    - E.g., to cause drop in the next hop
- Other roles need to be maintained too
  - Frame delimiting and recognition, flow control, enforcing IFG, …

Packet Processing

- A likely flow:

- Possible implementations:
  - The entire packet goes through the header processing unit
  - Just the header goes through the header processing unit
  - “Better” depends on your performance profile (what are the bottlenecks? Resource limitations?)

Packet Processing

- A likely flow:

- Challenges:
  - A field may arrive over multiple clock cycles (e.g. 32b field, 16b on clock 2 and 16b on clock 3)
  - Memory access taking more than 1 clock cycle
    - E.g. request on clock 1, reply on clock 3
    - Some memories allow multiple concurrent accesses, some don't
    - The bigger the memory, the more time it takes

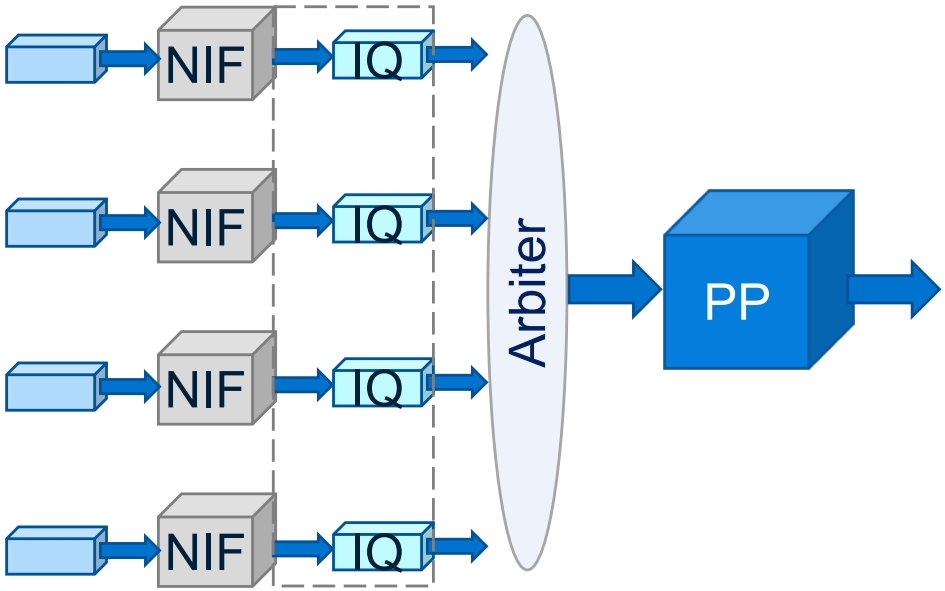
Packet Processing

- A likely flow:

- Solutions:
  - Pipelining!
    Don't stall, add NOP stages in your pipe.
  - Reorder operations (where possible)
    - E.g. Lookup 1 → Action 1 → Lookup 2 → Action 2 turns:
      Lookup 1 → Lookup 2 → Action 1 → Action 2
    - Don't create hazards!

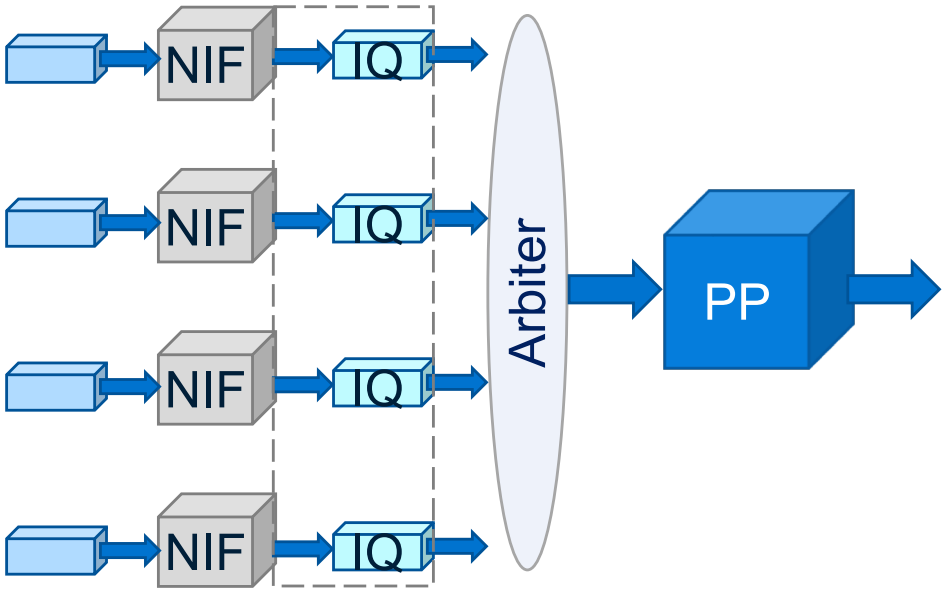
Arbitration

- Simple example:
  - Packets arriving from 4 ports
  - (approximately) same arrival time
  - Arbiter uses Round Robin
- Problem: arbitration on packet boundaries?
  - No: interleaved packets within the pipeline
    Need to track which cycle belongs to which packet
    May require multiple concurrent header lookups
    Order is not guaranteed (e.g. P1-P2-P3-P1-P2-P2-…), due to NIF timing



Arbitration

- Simple example:
  - Packets arriving from 4 ports
  - (approximately) same arrival time
  - Arbiter uses Round Robin
- Problem: arbitration on packet boundaries?
  - Yes: packets need to wait for previous packets to be handled before being admitted.
    Worst case waiting with <N> inputs is <N-1>×Packet time



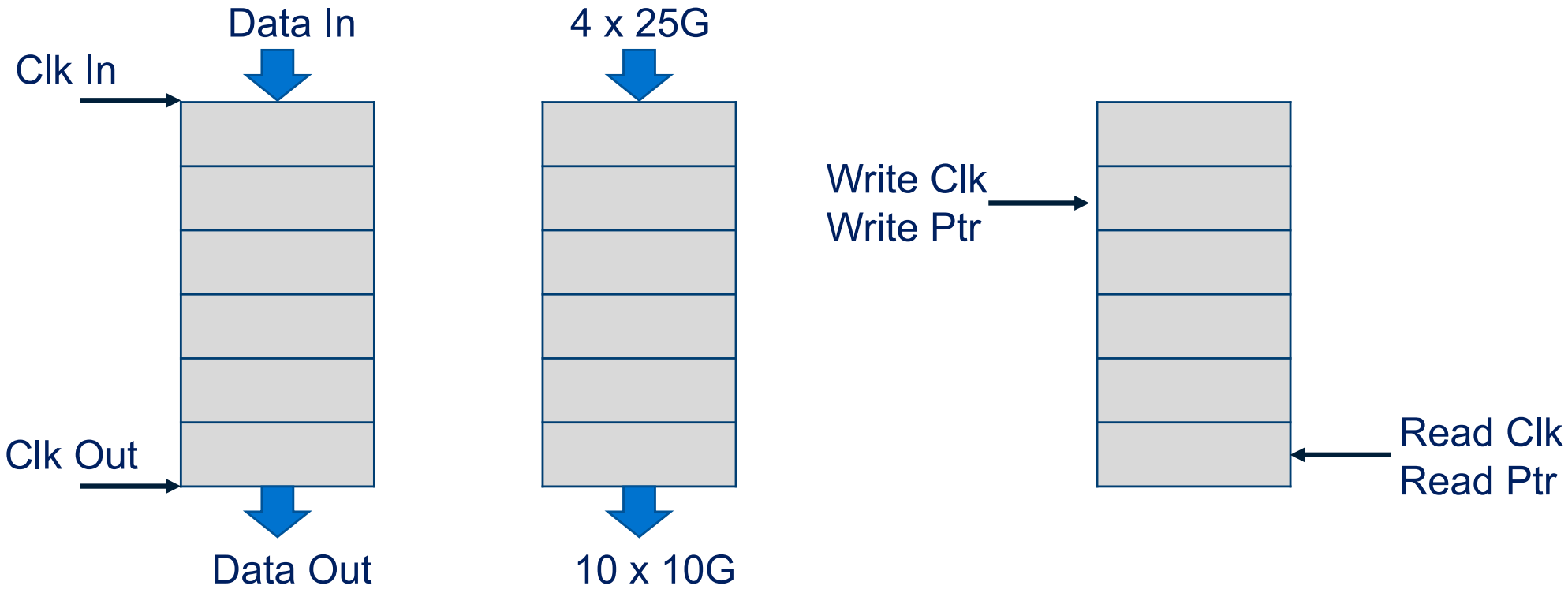
Arbitration

- Solutions to the previous problem:
  - Scheduled (or slotted) traffic
  - Multiple pipelines
  - …

Low Latency Devices

Crossing Clock Domains

- We have discussed the need for differing clock frequencies required in different places in the design.
- Crossing clock domains requires careful handling



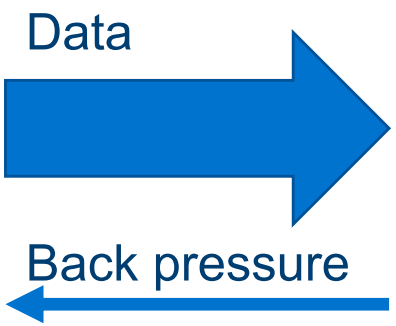
Crossing Clock Domains

Why do we care about clock domain crossing?

- Adds latency
- The latency is not deterministic
  - But bounded
- Crossing clock domains multiple times increases the jitter
- Using a single clock is often not an option:
  - Insufficient packet processing rate
  - Multiple interface clocks
  - Need speed up (e.g., to handle control events)

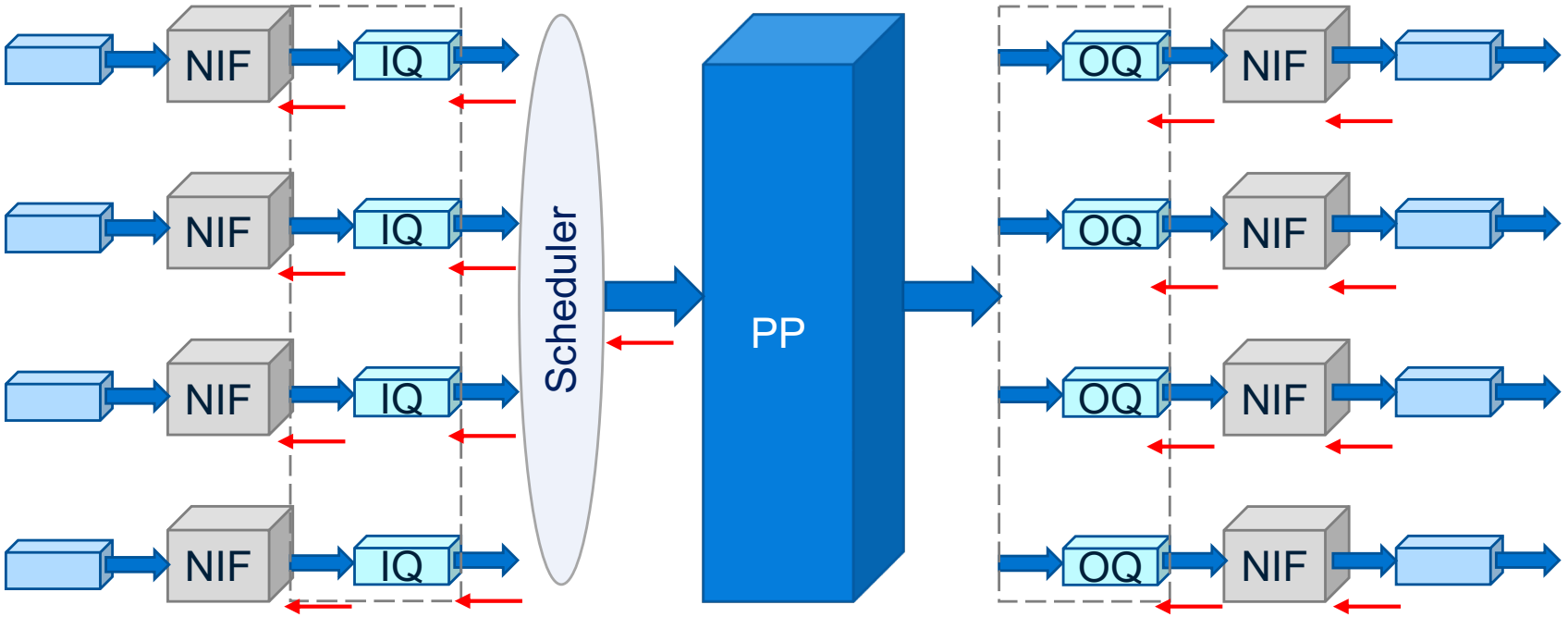
Flow Control

- The flow of the data through the device (the network) needs to be regulated
- Different events may lead to stopping the data:
  - An indication from the destination to stop
  - Congestion (e.g. 2 ports sending to 1 port)
  - Crossing clock domains
  - Rate control
  - …



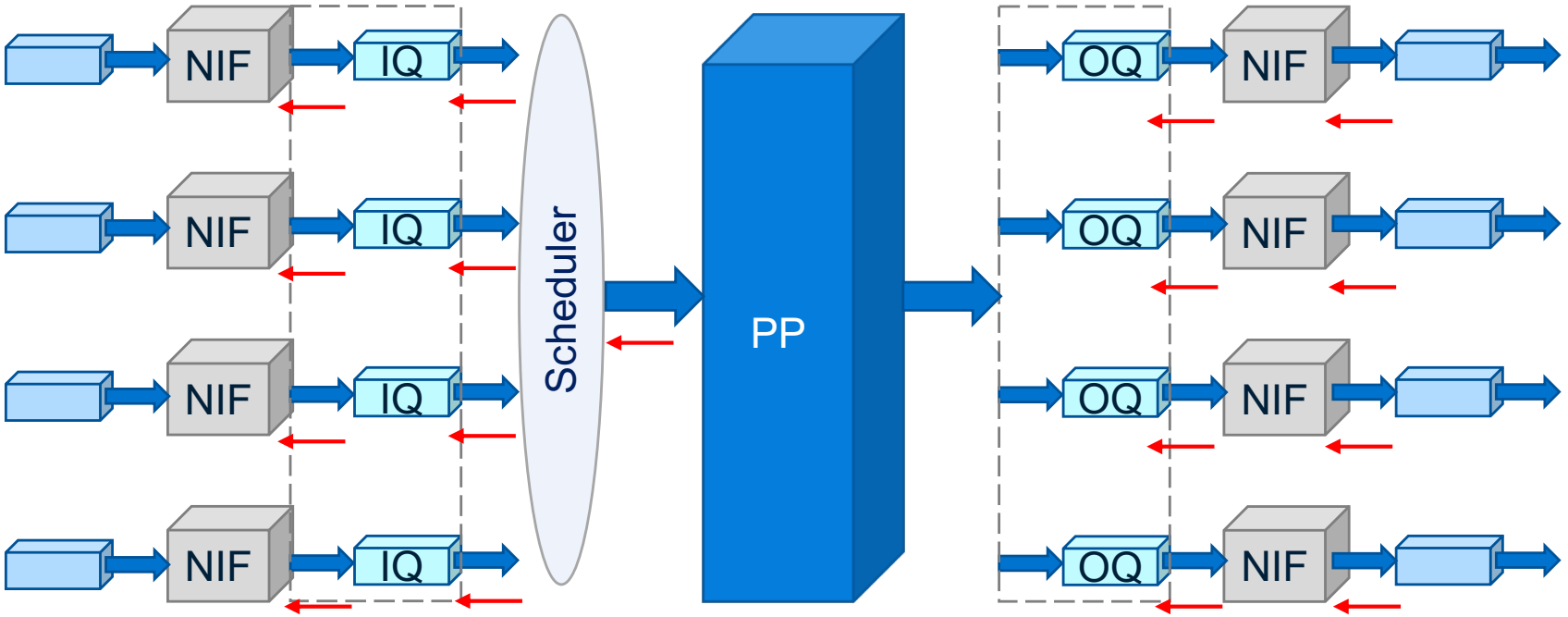
Flow Control

- Providing back pressure is not always allowed
- In such cases, need to make amendments in the design



Flow Control

- What to do if an output queue is congested?



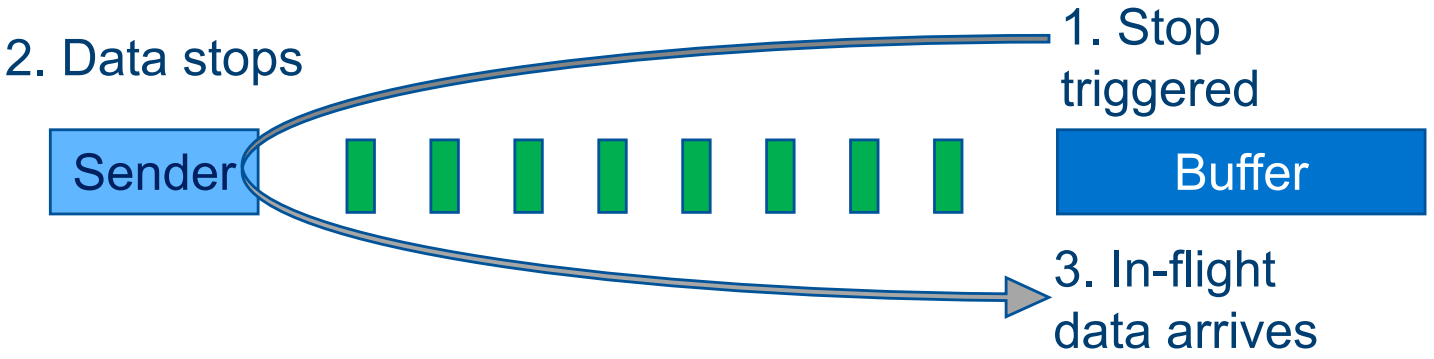
Flow Control and Buffering

- Back pressure may take *time*

- Need to either:
  - Assert back pressure sufficient time before traffic needs to stop OR
  - Provide sufficient buffering

Flow Control and Buffering

Calculating buffer size:



Intuitively:

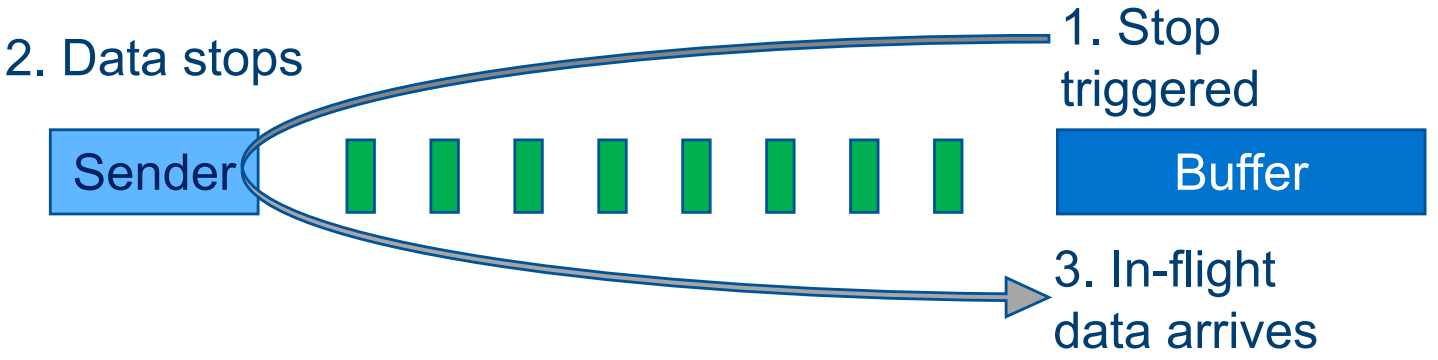
Nearby sender: Buffer size ≥ Reaction time × Data rate

Remote sender: Buffer size ≥ RTT × Data rate

Buffer size ≥ (RTT + Reaction time) × Data rate

Flow Control and Buffering

Calculating buffer size:



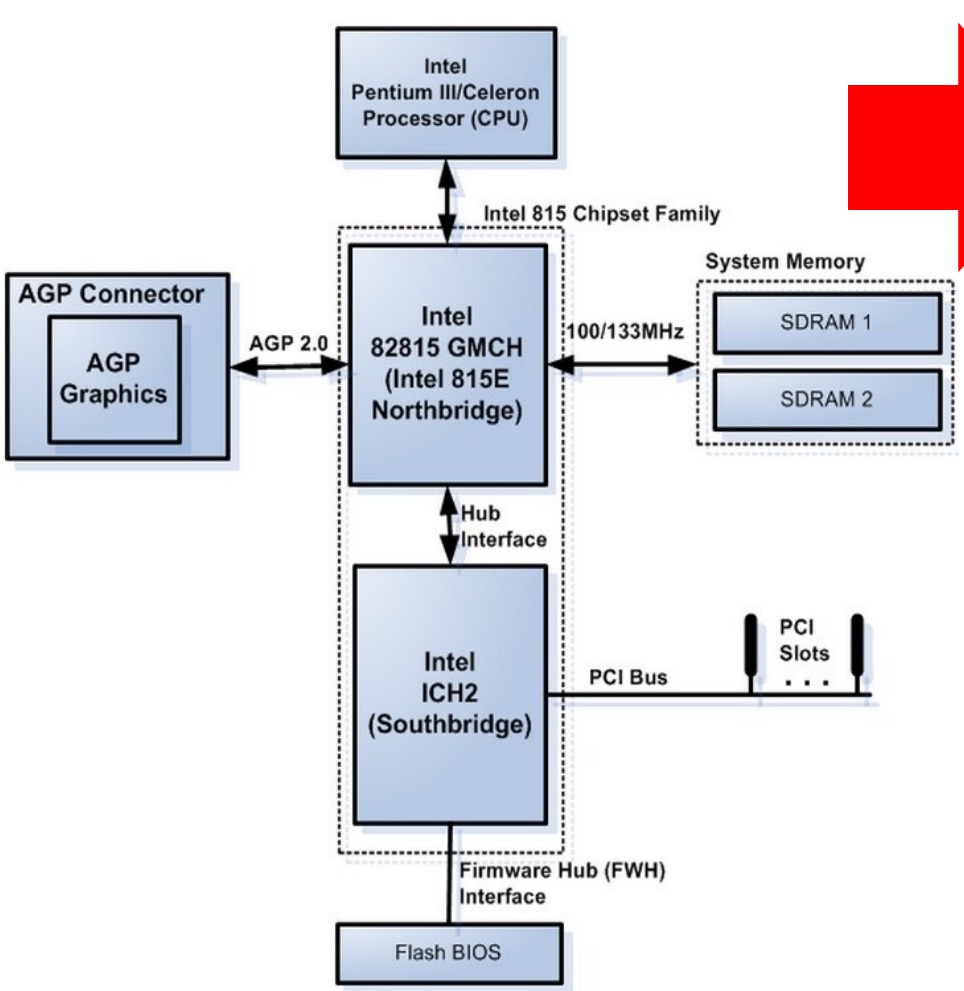
2 switches, connected using 100m fibre, 10G port, instantaneous response time:

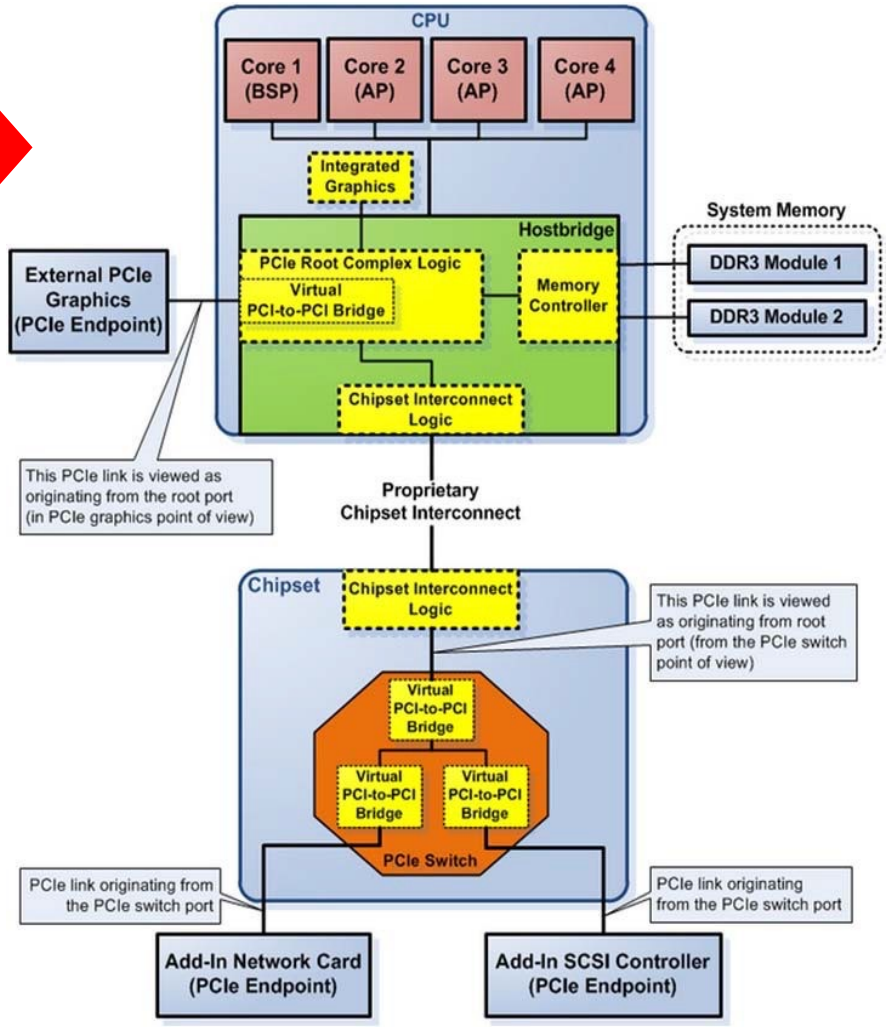
Propagation delay in a fibre is 5ns/m

Buffer size $\geq$ 1us $\times$ 10Gbps = ~1.25KB

DMA

Host architecture

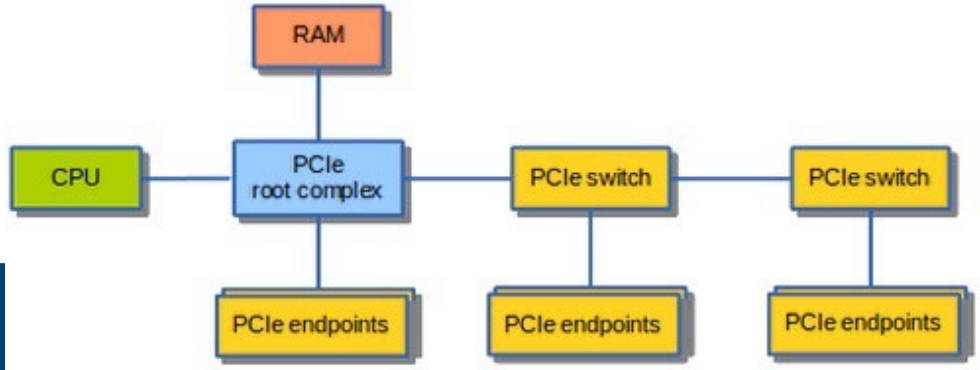




Legacy vs. Recent (courtesy of Intel)

Interconnecting components

- **Need interconnections between**
  - CPU, memory, storage, network, I/O controllers
- **Shared Bus: shared communication channel**
  - A set of parallel wires for data and synchronization of data transfer
  - Can become a bottleneck
- **Performance limited by physical factors**
  - Wire length, number of connections
- **More recent alternative: high-speed serial connections with switches**
  - Like networks



I/O System Characteristics

- **Performance measures**
  - Latency (response time)
  - Throughput (bandwidth)
  - Desktops & embedded systems
    - Mainly interested in response time & diversity of devices
  - Servers
    - Mainly interested in throughput & expandability of devices
- **Reliability**
  - Particularly for storage devices (fault avoidance, fault tolerance, fault forecasting)

I/O Management and strategies

- **I/O is mediated by the OS**
  - Multiple programs share I/O resources
    - Need protection and scheduling
  - I/O causes asynchronous interrupts
    - Same mechanism as exceptions
  - I/O programming is fiddly
    - OS provides abstractions to programs

Strategies characterize the *amount of work* done by the CPU in the I/O operation:

- **Polling**
- **Interrupt Driven**
- **Direct Memory Access**

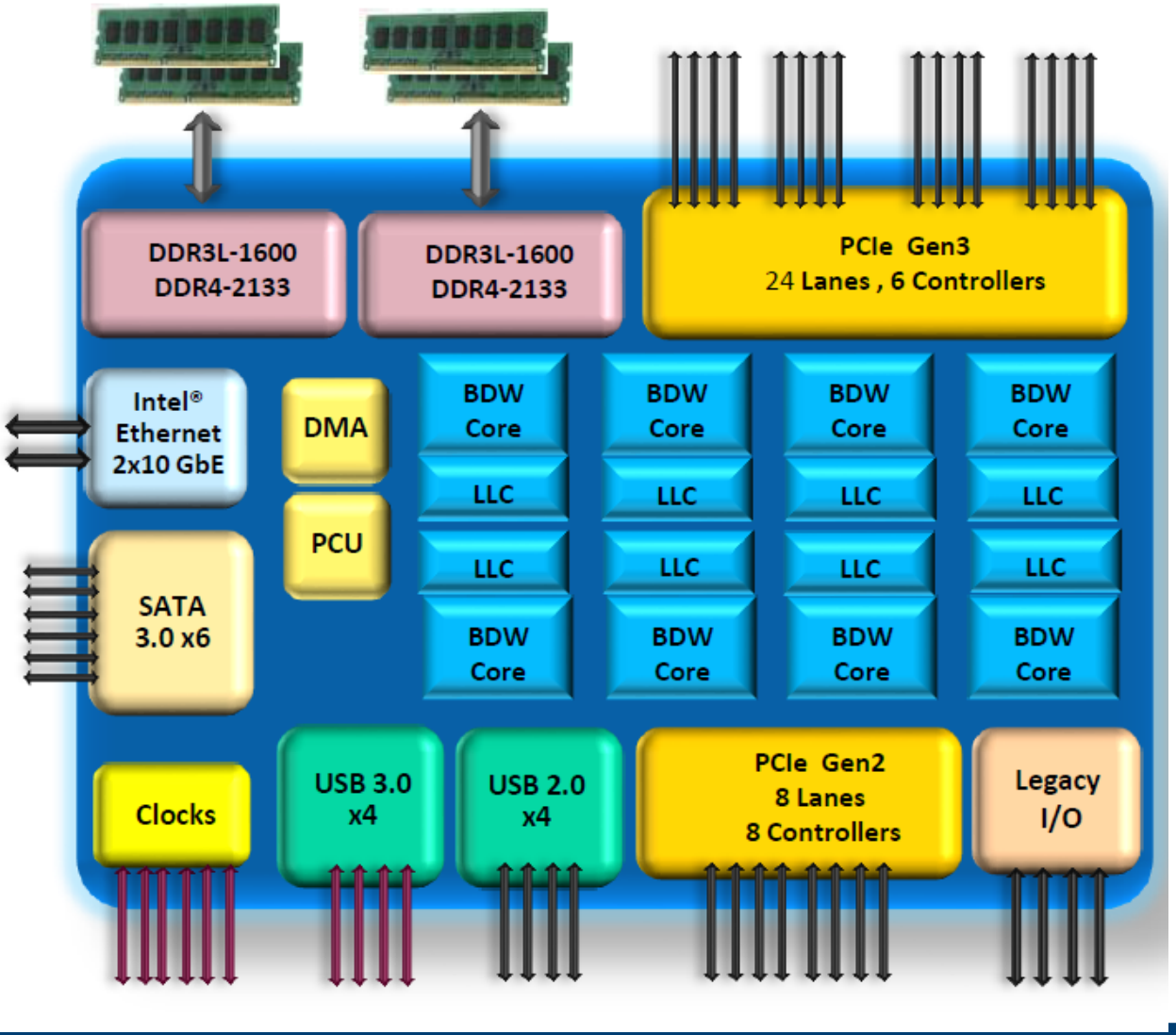
The I/O Access Problem

- Question: how to transfer data from I/O devices to memory (RAM)?
- Trivial solution:
  - Processor individually reads or writes every word
  - Transferred to/from I/O through an internal register to memory
- Problems:
  - Extremely inefficient – can occupy a processor for 1000's of cycles
  - Pollute cache

DMA

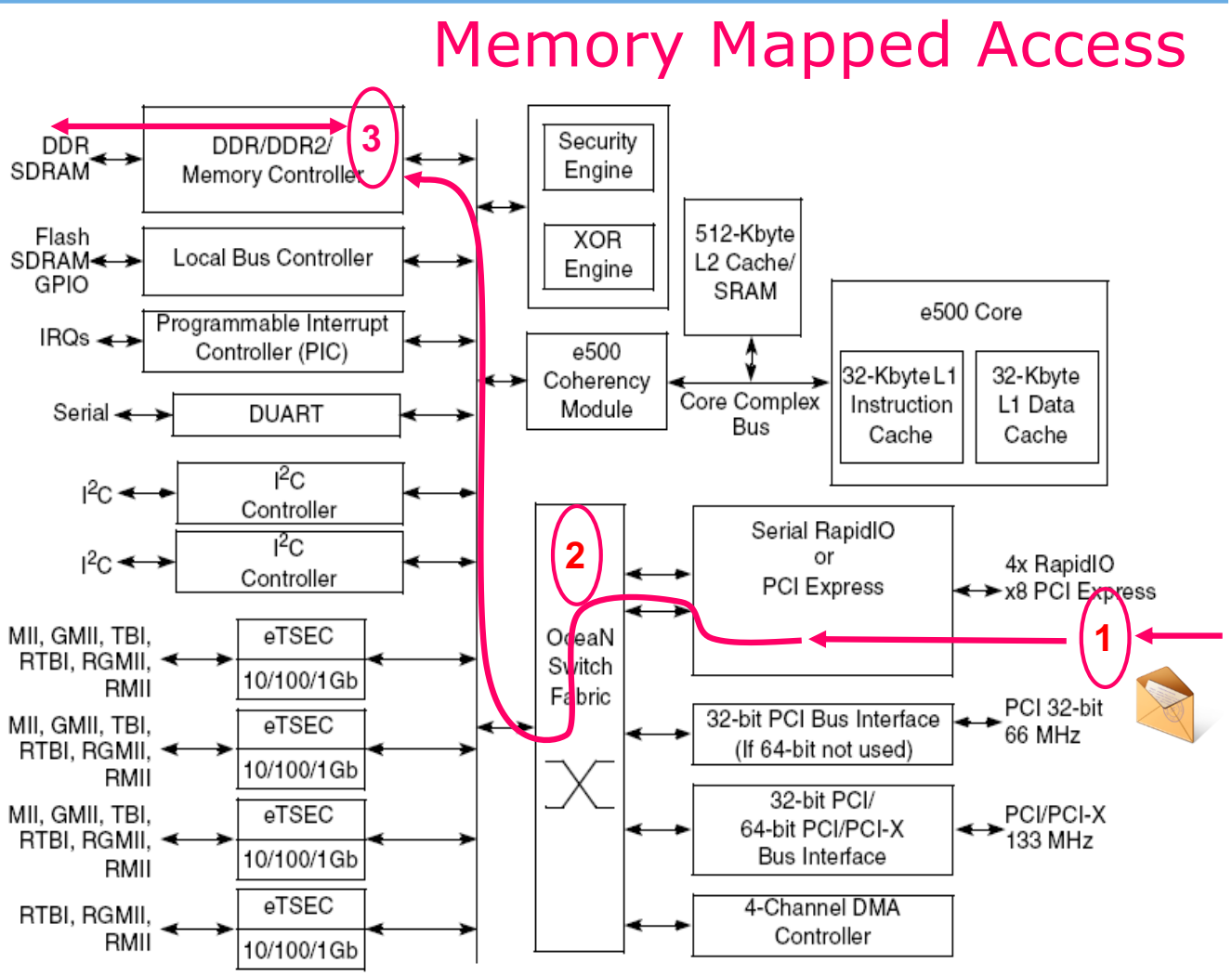
- DMA – Direct Memory Access
- A modern solution to the I/O access problem
- The peripheral I/O can issue read/write commands directly to the memory
  - Through the main memory controller
  - The processor does not need to execute any operation
- Write: The processor is notified when a transaction is completed (interrupt)
- Read: The processor issues a signal to the I/O when the data is ready in memory

Example – Intel Xeon D



Example (Embedded Processor)

1. Message arrives on I/O interface.
   Message is decoded to Mem read/write.
   Address is converted to internal address.
2. Mem Read/Write command goes through the switch to the internal bus and memory controller.
3. Memory controller executes the command to the DRAM.
   Returns data if required in the same manner.



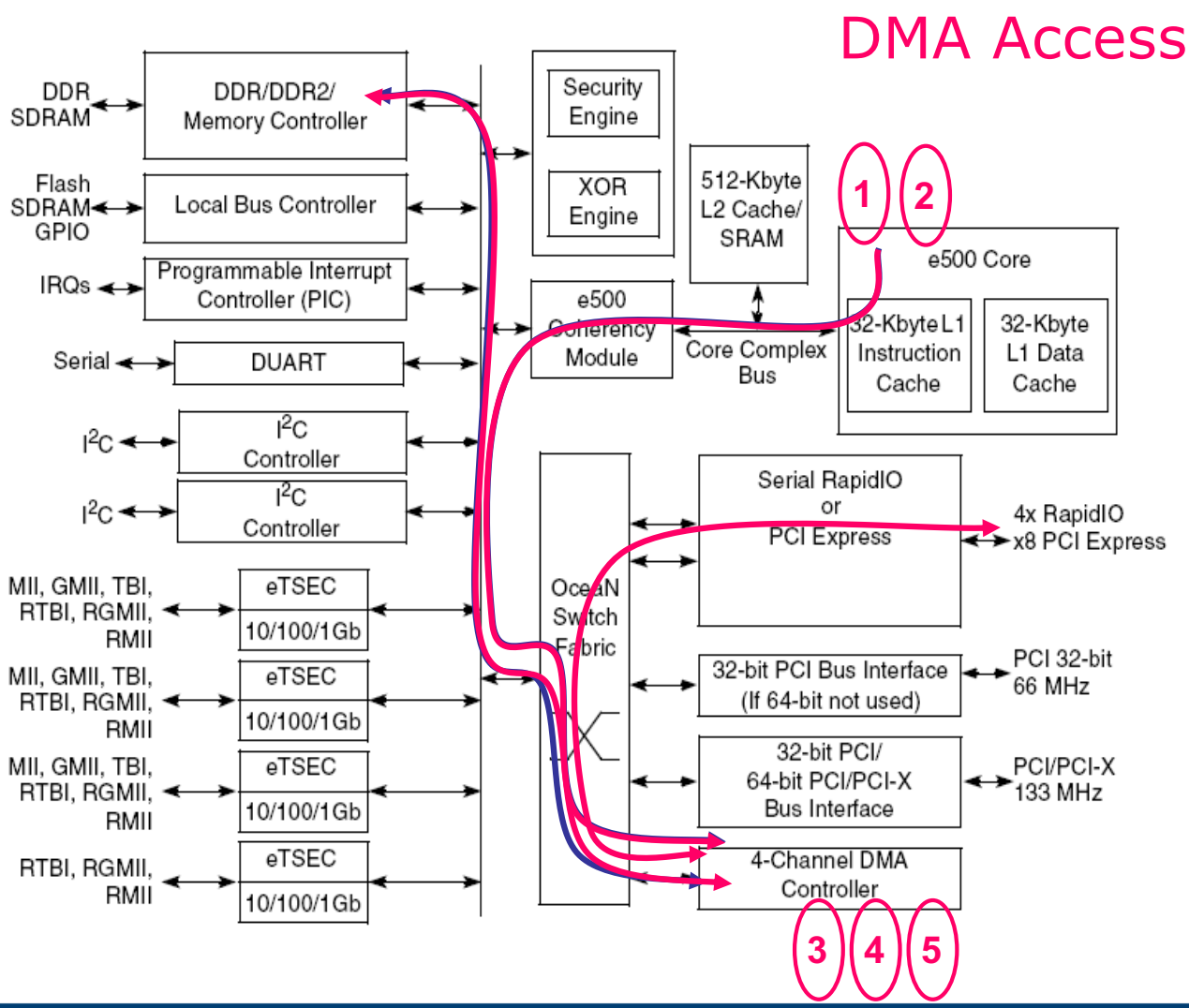
DMA

- DMA accesses are usually handled in *buffers*
  - Single word/block is typically inefficient
- The processors assigns the peripheral unit the buffers in advance
- The buffers are typically handled by *buffer descriptors*
  - Pointer to the buffer in the memory
  - May point to the next buffer as well
  - Indicates buffer status: owner, valid etc.
  - May include additional buffer properties as well

Example (Embedded Processor)

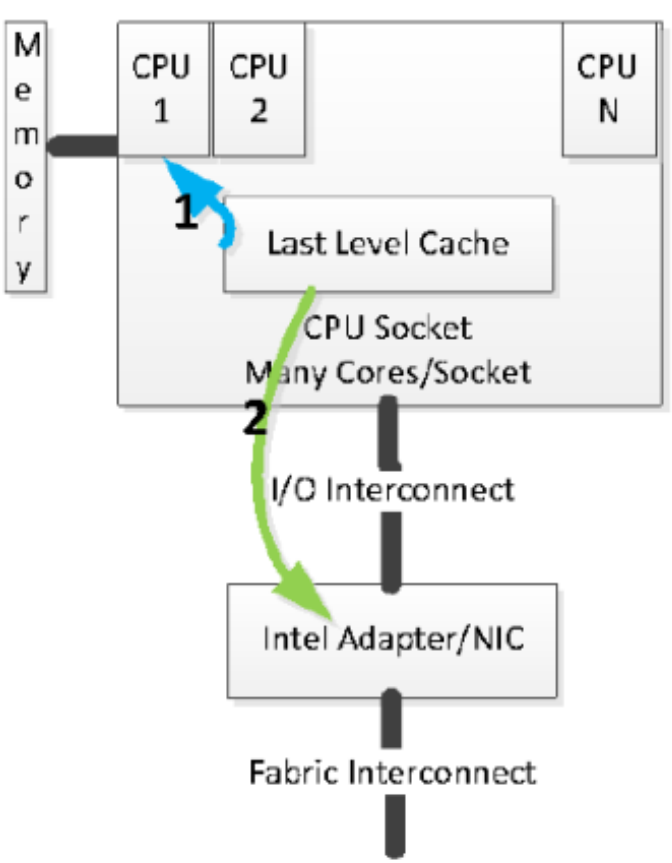
Transfers blocks of data between external interfaces and local address space

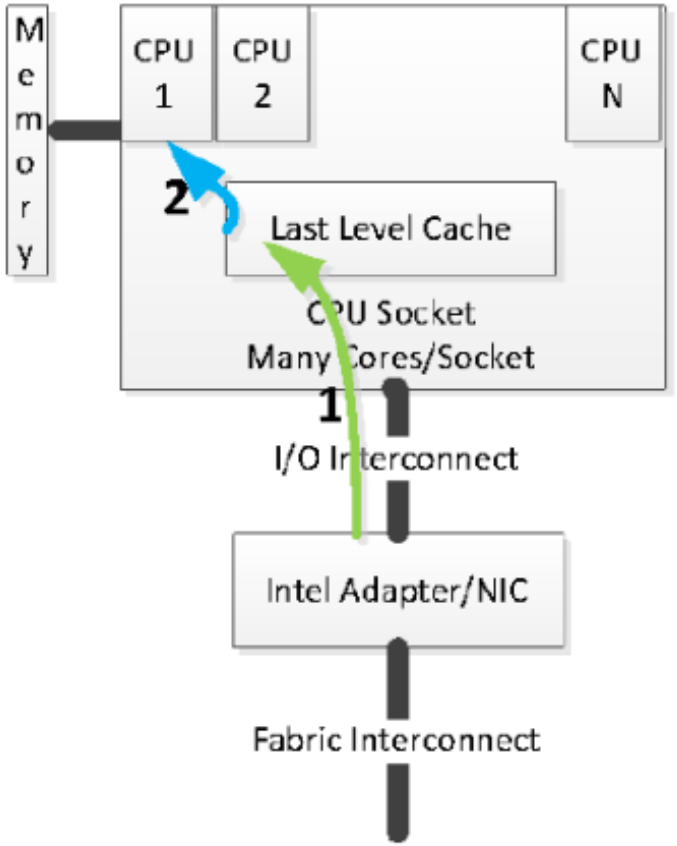
1. A transfer is started by SW writing to DMA engine configuration registers
2. SW Polls DMA channel state to idle and sets trigger
3. DMA engine fetches a descriptor from memory
4. DMA engine reads block of data from source
5. DMA engine writes data to destination



Intel Data Direct I/O (DDIO)

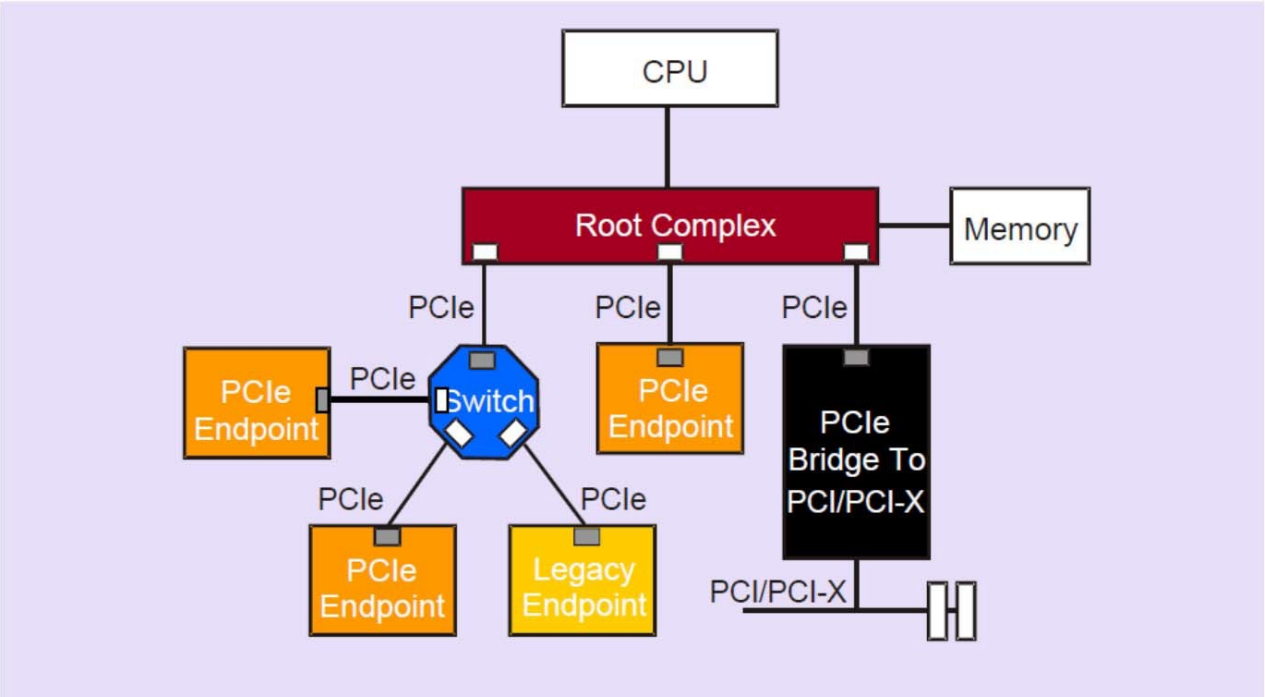
- Data is written and read directly to/from the last level cache





PCIe introduction

- PCIe is a *serial point-to-point interconnect* between two devices
- Implements *packet based protocol (TLPs)* for information transfer
- *Scalable performance* based on # of signal Lanes implemented on the PCIe interconnect
- Supports *credit-based* point-to-point flow control (not end-to-end)



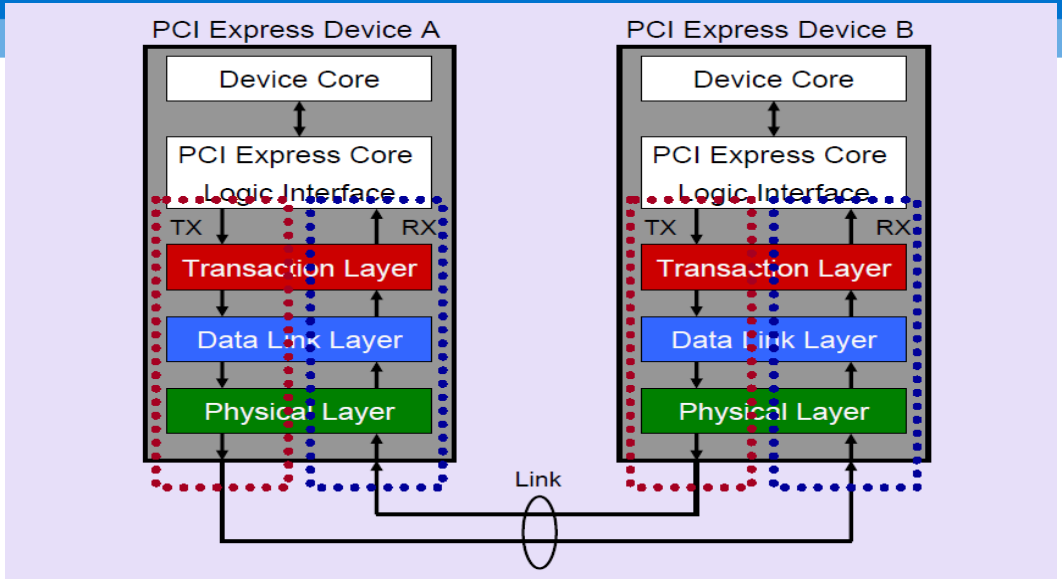
Provides:

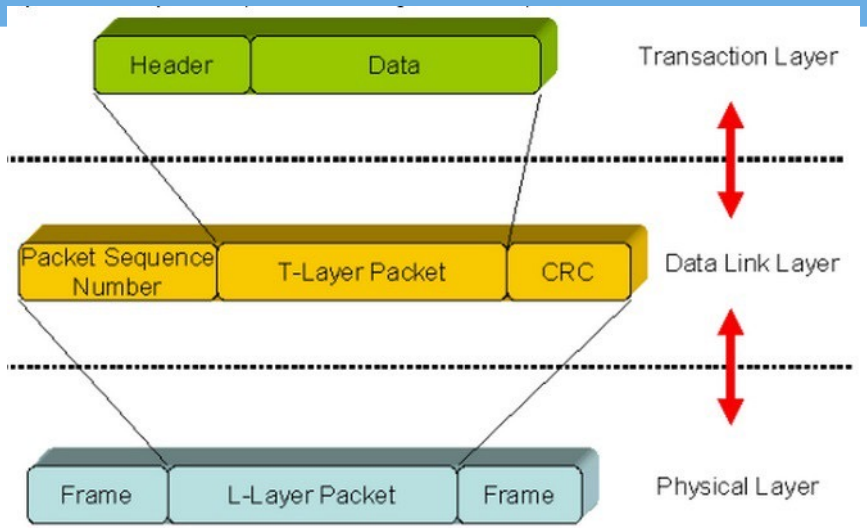
- Processor independence & buffered isolation
- Bus mastering
- Plug and Play operation

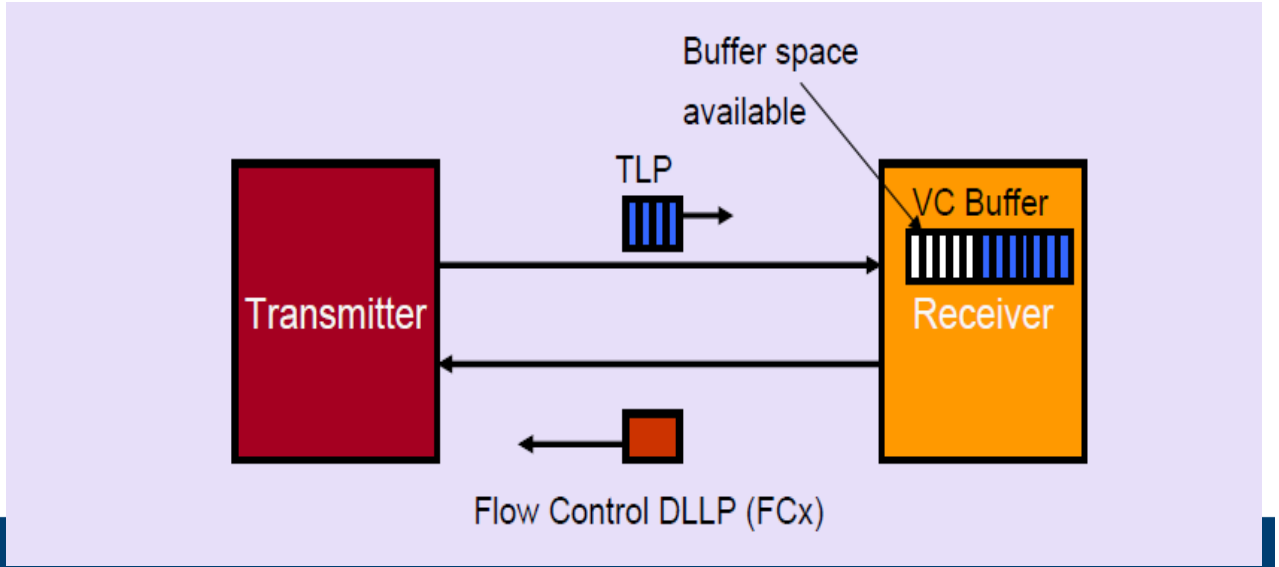
PCIe transaction types

- Memory Read or Memory Write. Used to transfer data from or to a memory mapped location
- I/O Read or I/O Write. Used to transfer data from or to an I/O location
- Configuration Read or Configuration Write. Used to discover device capabilities, program features, and check status in the 4KB PCI Express configuration space.
- Messages. Handled like posted writes. Used for event signaling and general purpose messaging.

PCIe architecture







Interrupt Model

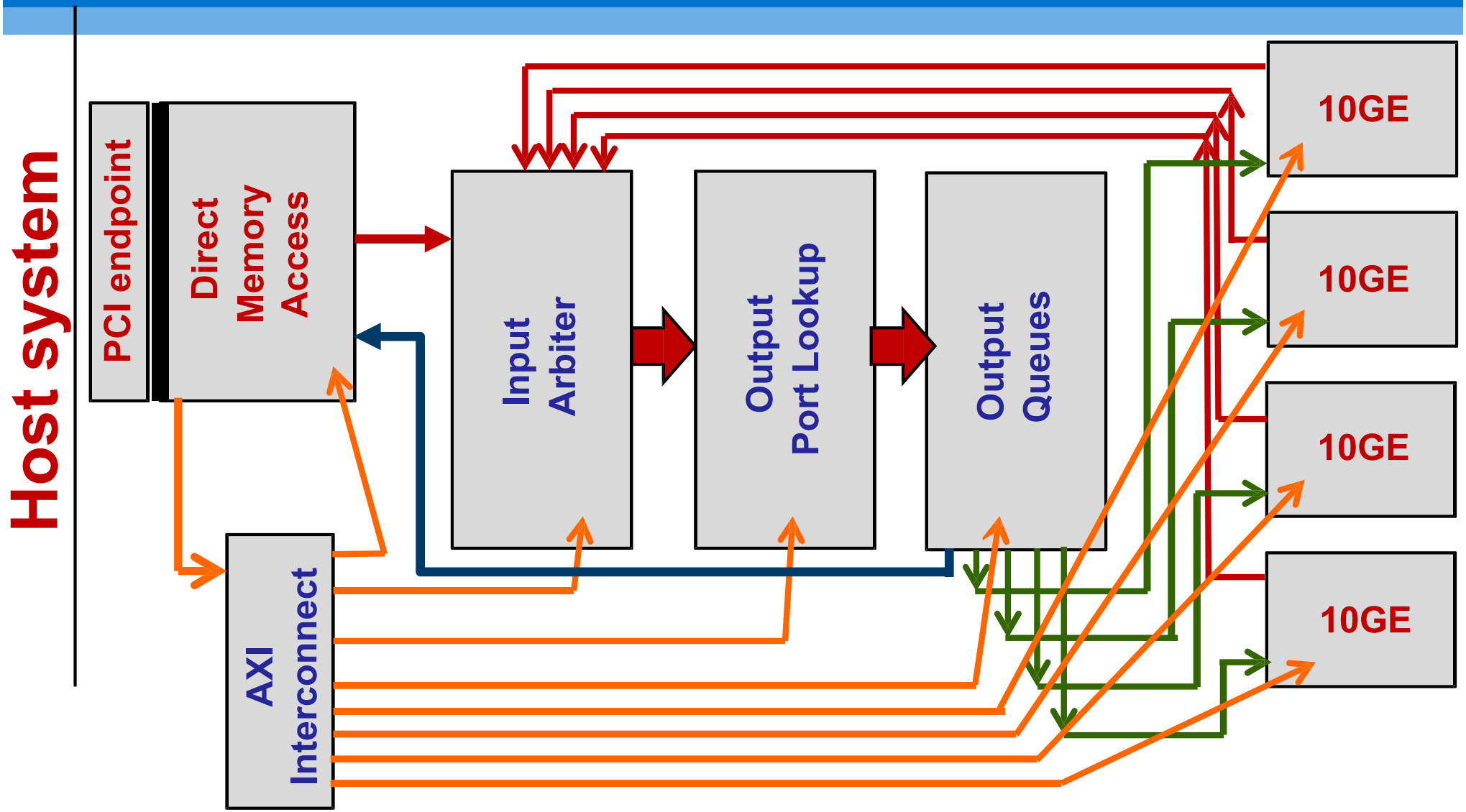
PCI Express supports three interrupt reporting mechanisms:

1. Message Signaled Interrupts (MSI)
   - interrupt the CPU by writing to a specific address in memory with a payload of 1 DW

2. Message Signaled Interrupts - X (MSI-X)
   - MSI-X is an extension to MSI, allows targeting individual interrupts to different processors

3. INTx Emulation
   four physical interrupt signals INTA-INTD are messages upstream
   - ultimately be routed to the system interrupt controller

NetFPGA Reference Projects



Processing Overheads

- Processing in the kernel takes a lot of time…

Component	Time [us]
Driver RX	0.60
Ethernet & IPv4 RX	0.19
TCP RX	0.53
Socket Enqueue	0.06
TCP TX	0.70
IPv4 & Ethernet TX	0.06
Driver TX	0.43

Source: Yasukata *et al*. “StackMap: Low-Latency Networking with the OS Stack and Dedicated NICs”, Usenix ATC 2016

Processing Overheads

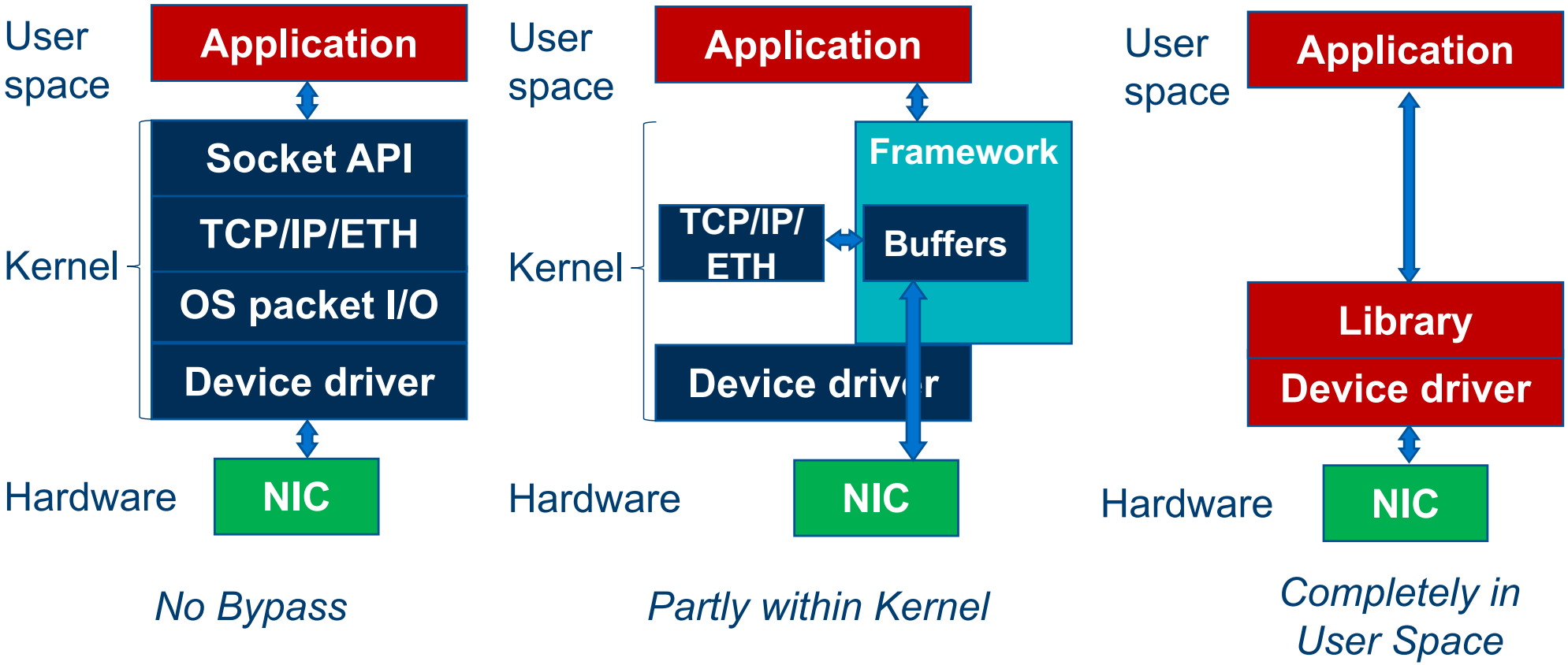
- Processing in the kernel takes a lot of time…
- Order of microseconds (~2-4us on Xeon E5-v4)
- $\times$10 the time through a switch

- Solution: don't go through the kernel!

Kernel Bypass

- The Kernel is slow – lets bypass the Kernel!
- There are many ways to achieve kernel bypass
- Some examples:
  - Device drivers:
    - Customized kernel device driver. E.g. Netmap forks standard Intel drivers with extensions to map I/O memory into userspace.
    - Custom hardware and use bespoke device drivers for the specialized hardware.
  - Userspace library: anything from basic I/O to the entire TCP/IP stack

Kernel Bypass - Examples



DPDK

- DPDK is a popular set of **libraries and drivers** for fast packet processing.
- Originally designed for Intel processors
  - Now running also on ARM and Power CPUs
- Runs mostly in Linux User space.
- Main libraries: multicore framework, huge page memory, ring buffers, poll-mode drivers (networking, crypto etc)
- It is ***not*** a networking stack

DPDK

- Usage examples:
  - Send and receive packets within minimum number of CPU cycles
    - E.g. less than 80 cycles
  - Fast packet capture algorithms
  - Running third-party stacks
- Some projects demonstrated 100’s of millions packets per seconds
  - But with limited functionality
  - E.g. as a software switch / router

High Throughput Switches

The Truth About Switch Silicon Design

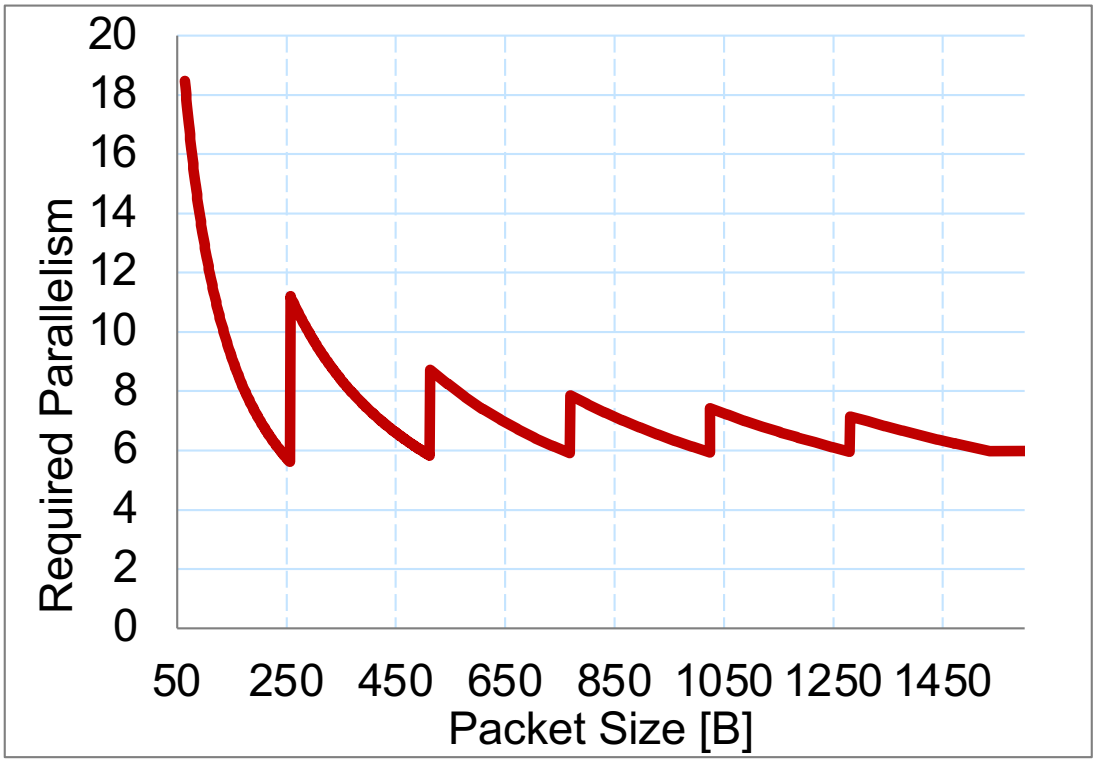
12.8Tbps Switches!

Lets convert this to packet rate requirements:

5.8 Gpps @ 256B

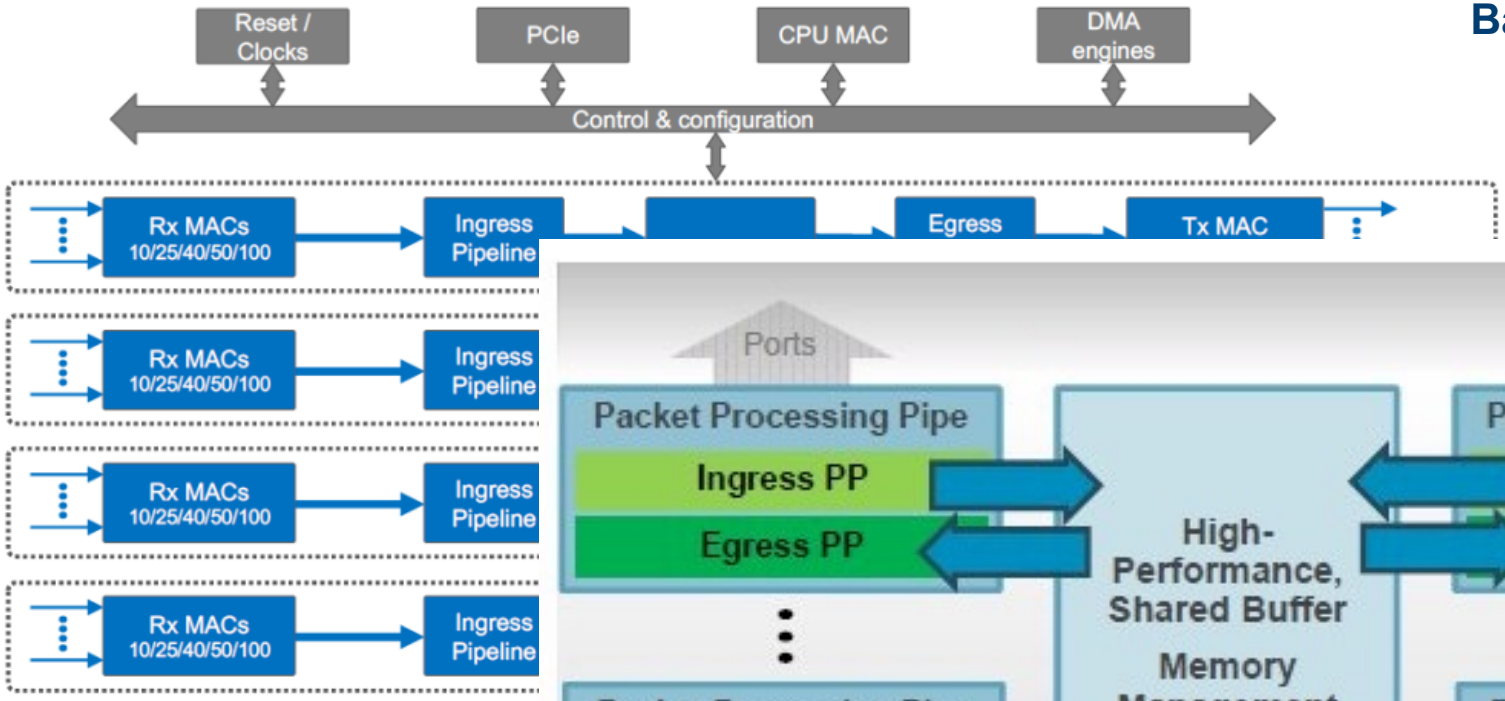
19.2 Gpps @ 64B

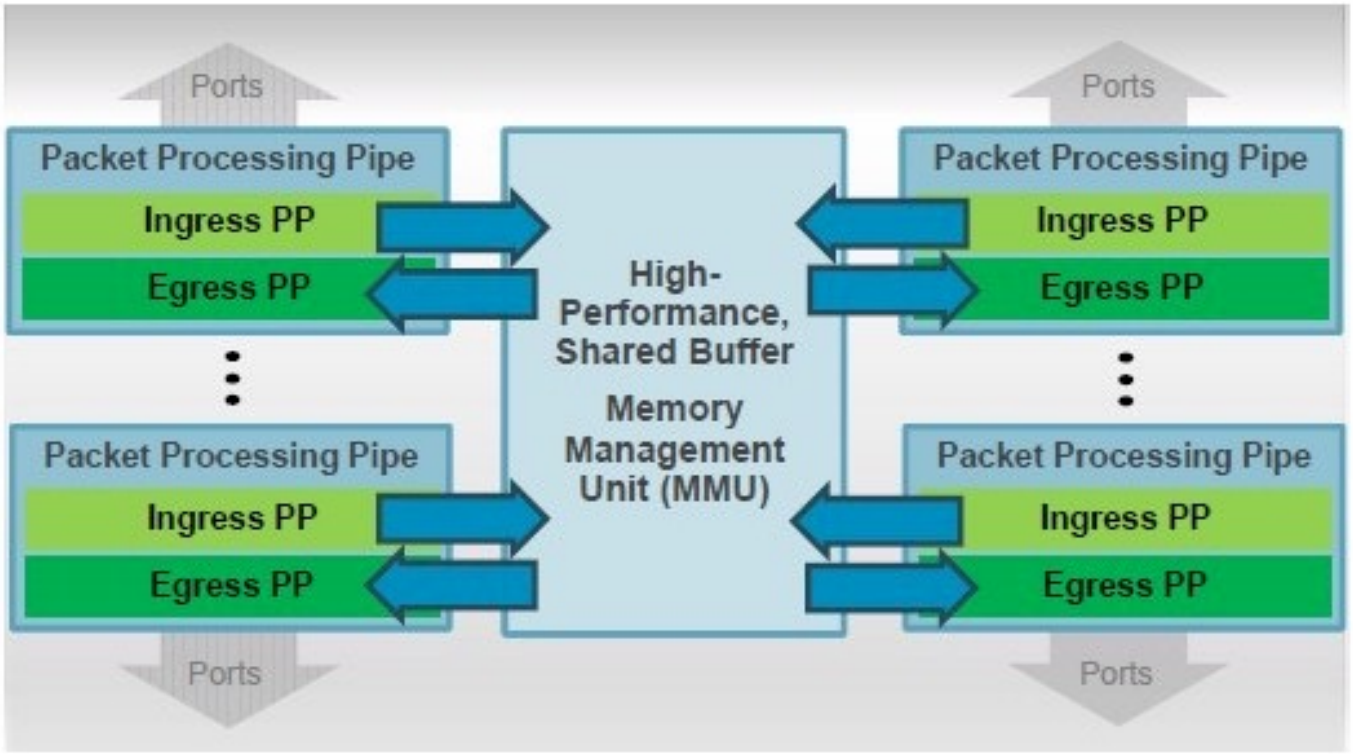
But clock rate is only ~1GHz….



Multi-Core Switch Design

Barefoot Tofino





Broadcom Tomahawk 3

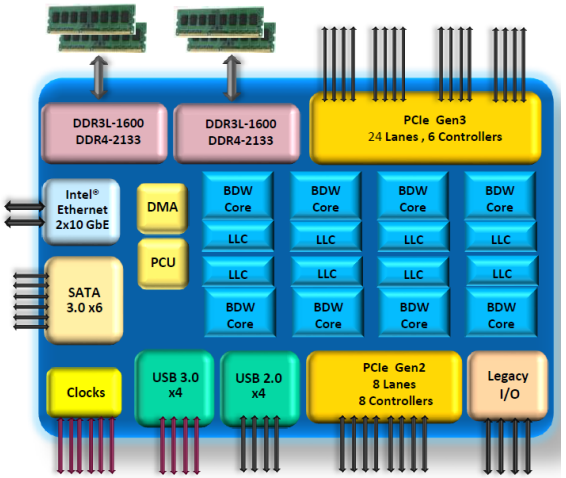
Image sources: https://p4.org/assets/p4_d2_2017_programmable_data_plane_at_terabit_speeds.pdf
https://www.nextplatform.com/2018/01/20/flattening-networks-budgets-400g-ethernet/

Multi Core Switch Design

- So what? Multi-core in CPUs for over a decade
- Network devices are not like CPUs:
  - CPU: Pipeline - instructions, memory – data
  - Switch: pipeline – data, memory – control
- Network devices have a strong notion of *time*
  - *Must* process the header on cycle X
  - Headers are split across clock cycles
  - Pipelining is the way to achieve performance

Multi Core Switch Design

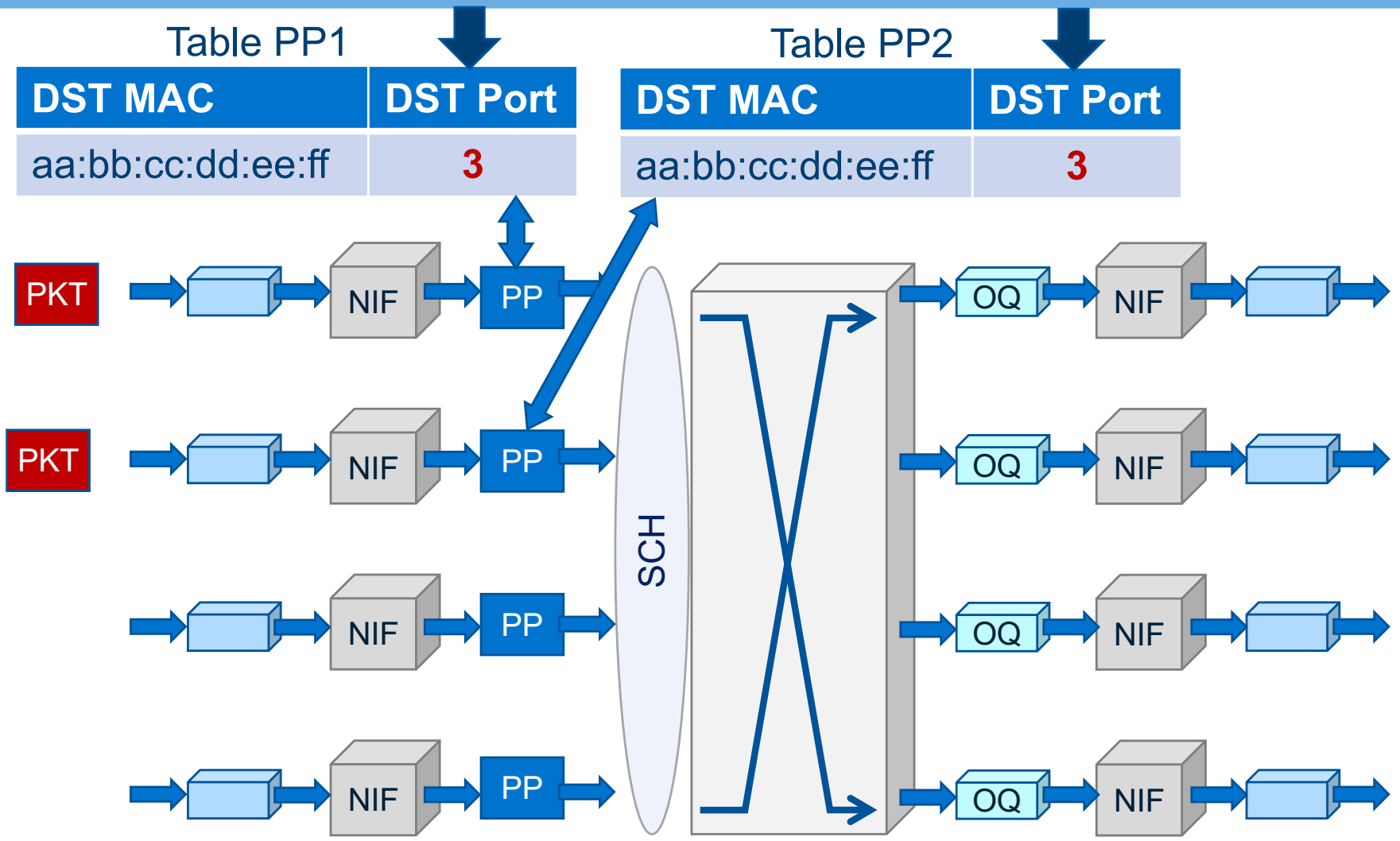
- The limitations of processing packets in the host:
- DPDK: can process a packet in 80 clock cycles
  - Lets assume 4GHz clock (0.25ns/cycle)
  - Can process $4\times10^9 \div 80 = 50\times10^6$
  - 50Mpps is not sufficient for 40GE. 30% of 64B packets at 100GE.
  - Can dedicate multiple cores…
  - And this is just sending / receiving, not operating on the packet!



Multi Core Switch Design

- The problem with multi-core switch design: look up tables.
  - Shared tables:
    - need to allow access from multiple pipelines
    - need to support query rate at packet rate
  - Separate tables:
    - wastes resources
    - need to maintain consistency
      - Not everyone agree with this assumption

Multi Core Switch Design

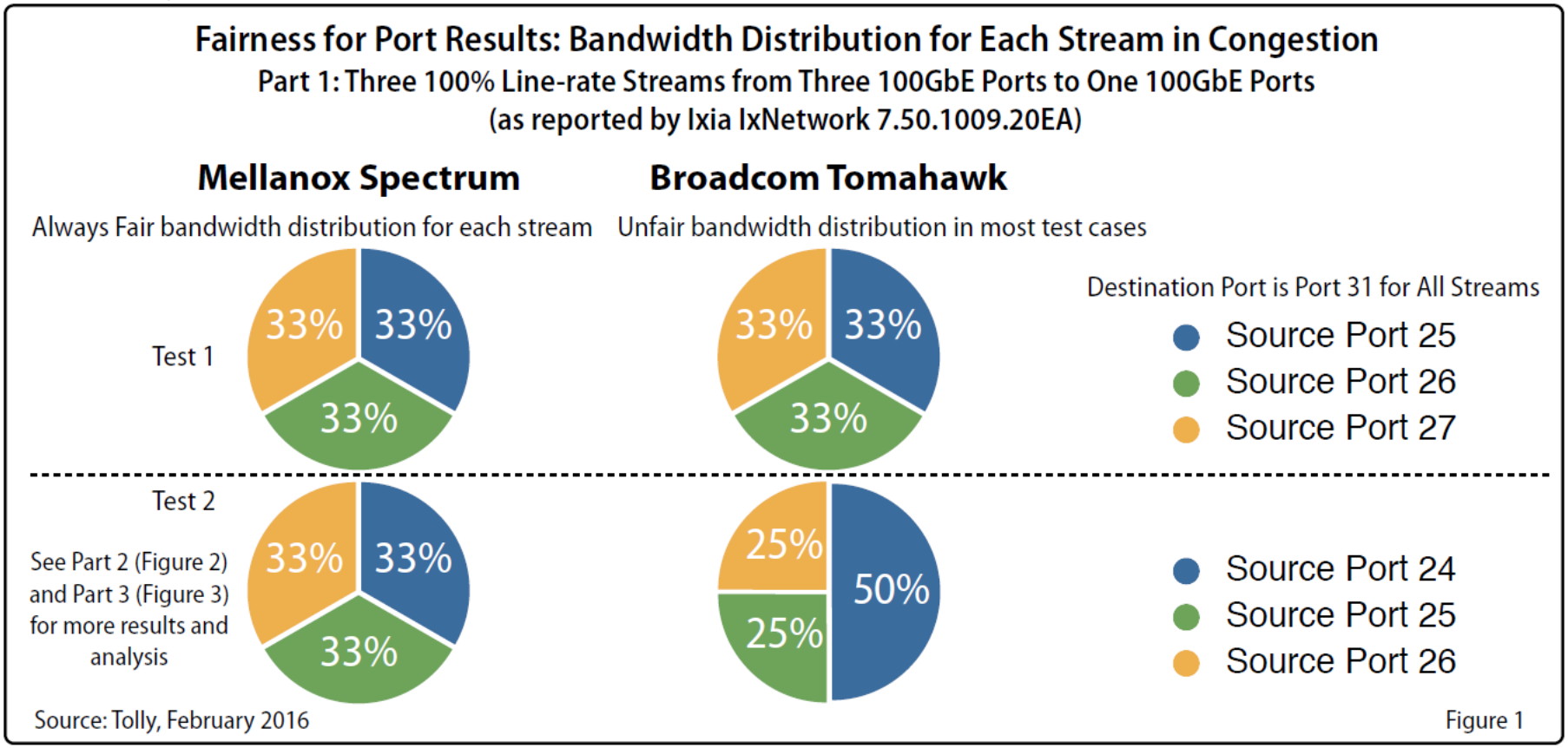


Inferring Switch Architecture

refer to:
http://www.mellanox.com/tolly/

All interpretations in the following slides are a guess, and not based on internal information

What is wrong with Broadcom Tomahawk?



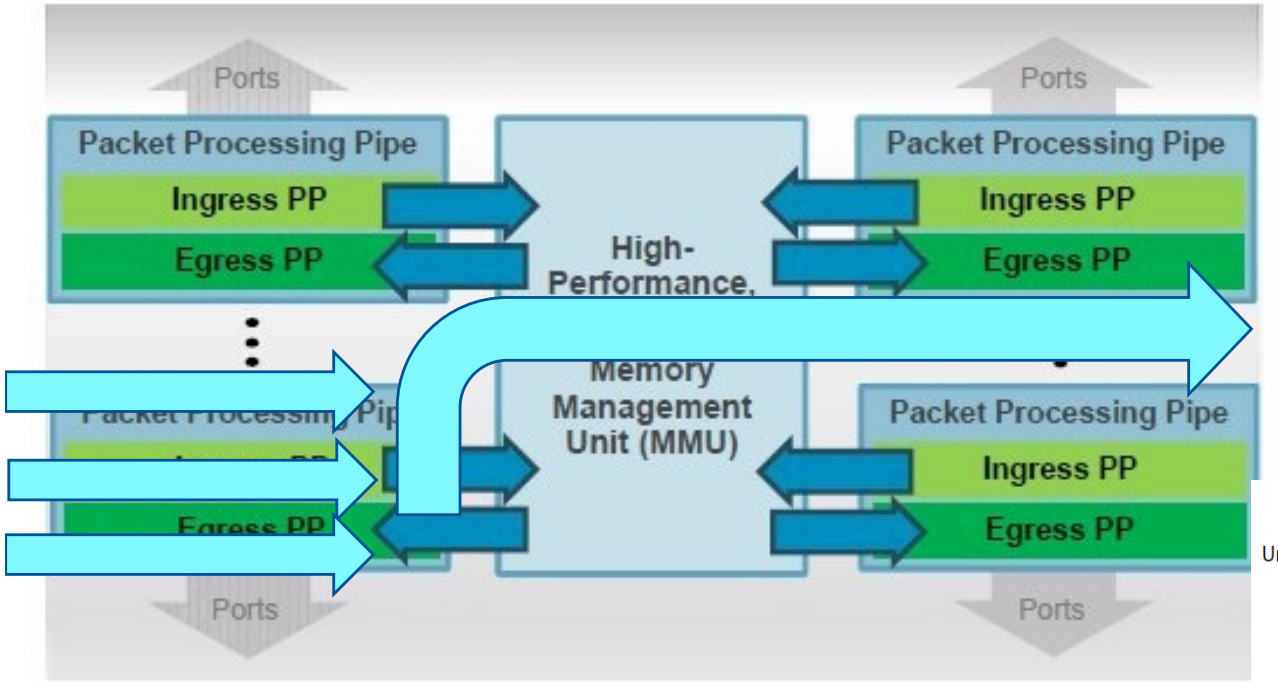
Broadcom Tomahawk

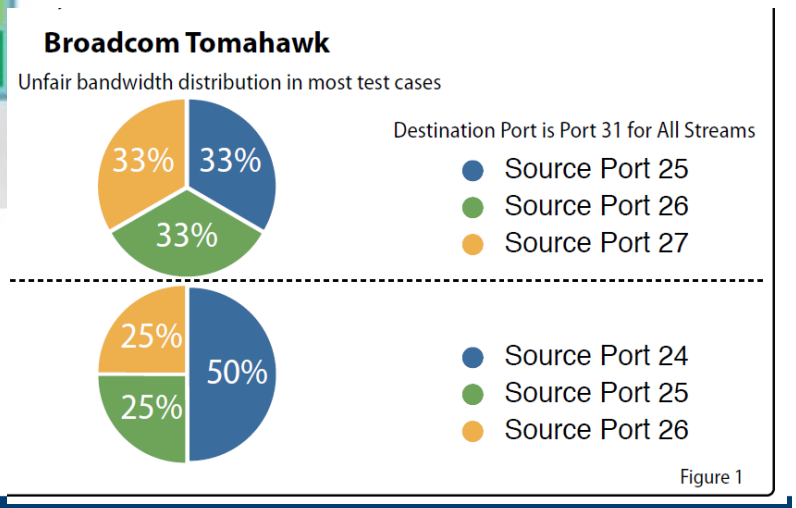
- 32 x 100GE
- In packet rate: 32 x 150Mpps = 4800 Mpps
- Manufacturing process: 28nm
  - Therefore clock frequency likely <1GHz
- More than 7 billion transistor
  - Reference: Intel debut around the same time 18-core Xeon E5-2600 v3 with 5.57 billion transistors

- … now lets think of these experimental results in a multi core switch…

What is wrong with Broadcom Tomahawk?

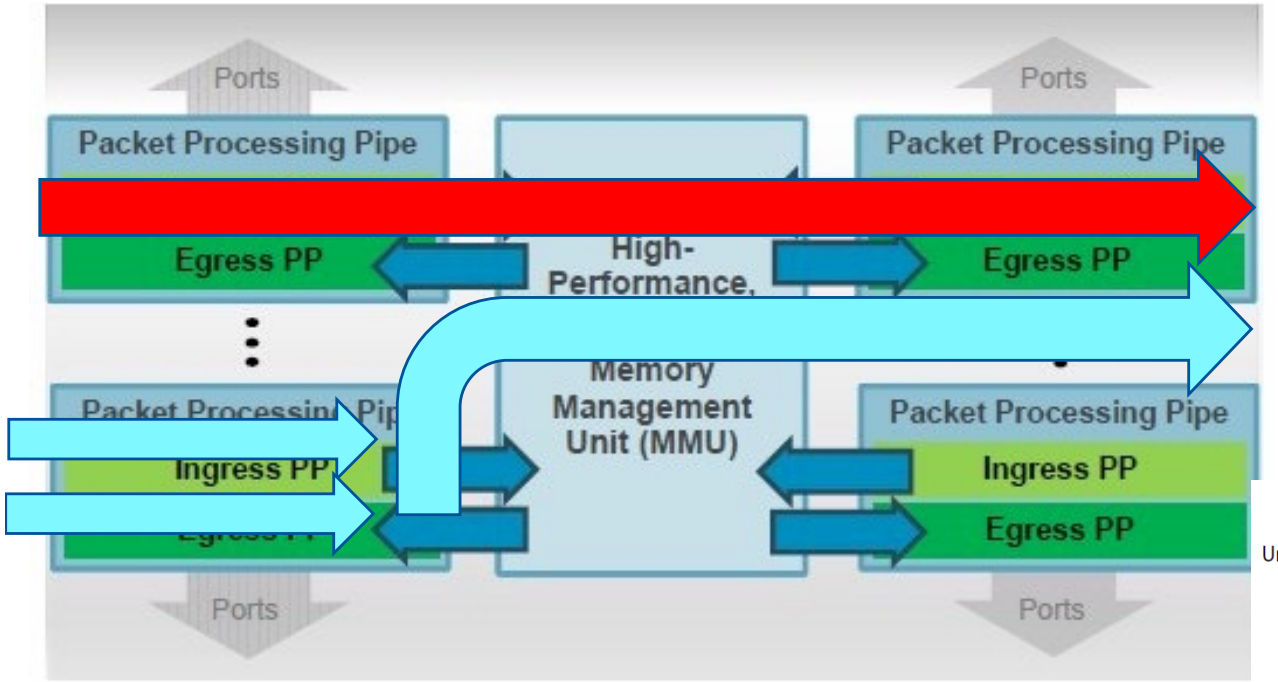
- Let us assume the same architecture as used by Tomahawk 3:

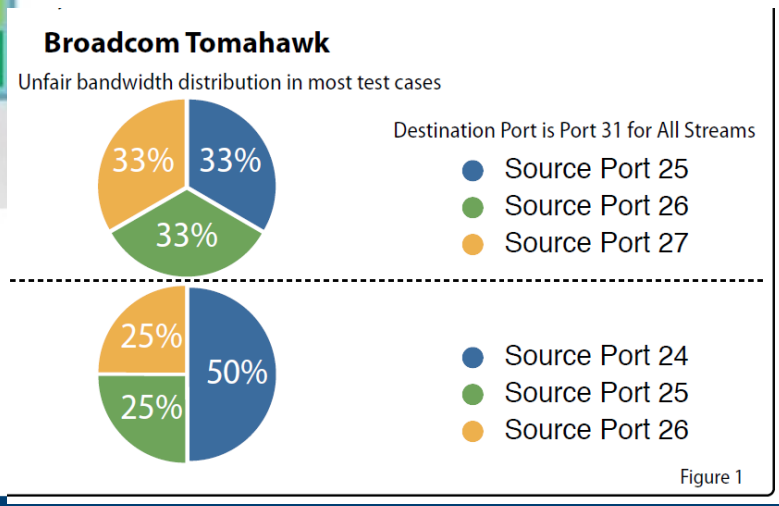




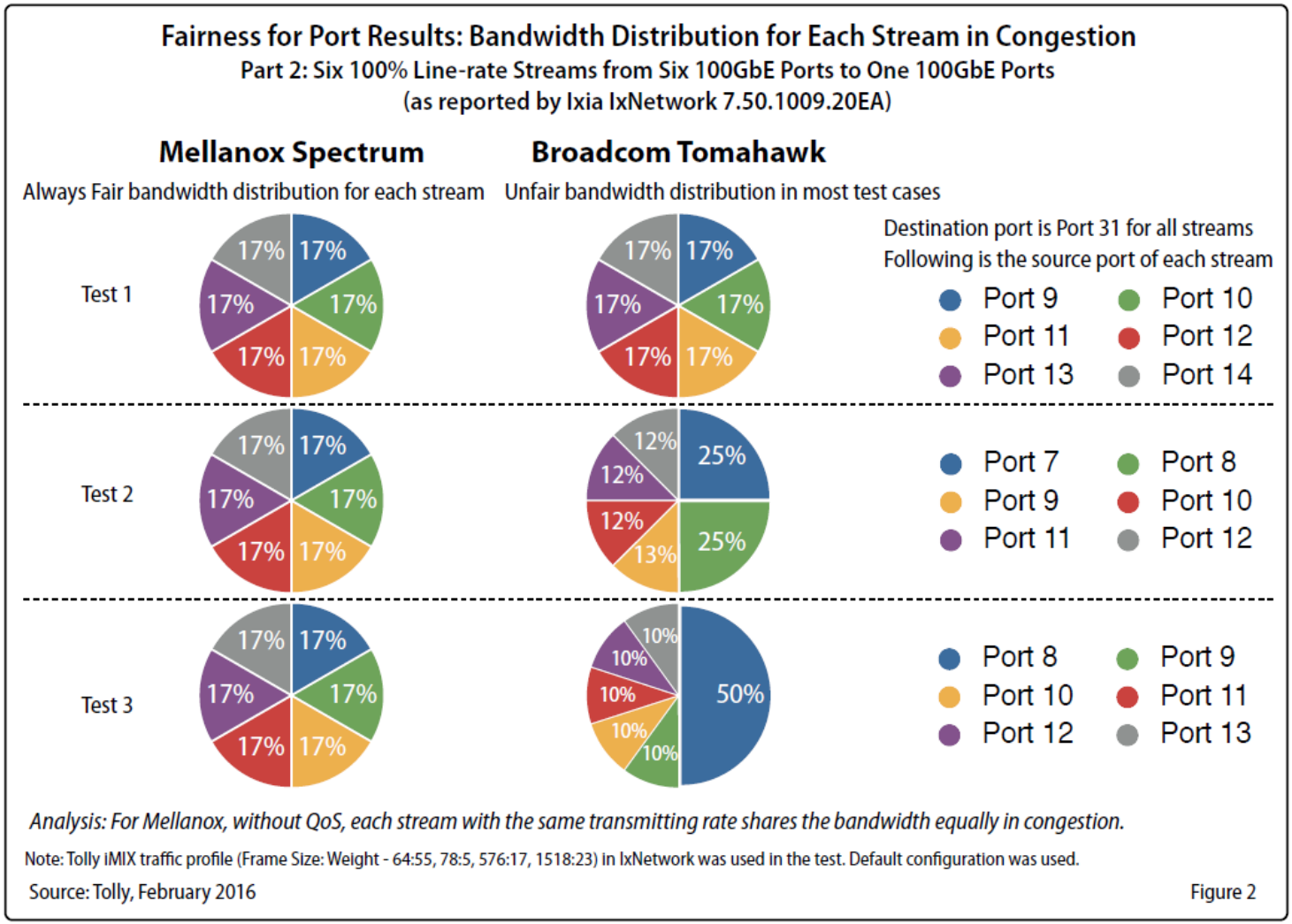
What is wrong with Broadcom Tomahawk?

- Let us assume the same architecture as used by Tomahawk 3:





What is wrong with Broadcom Tomahawk?



High Throughput Interfaces

Performance Limitations

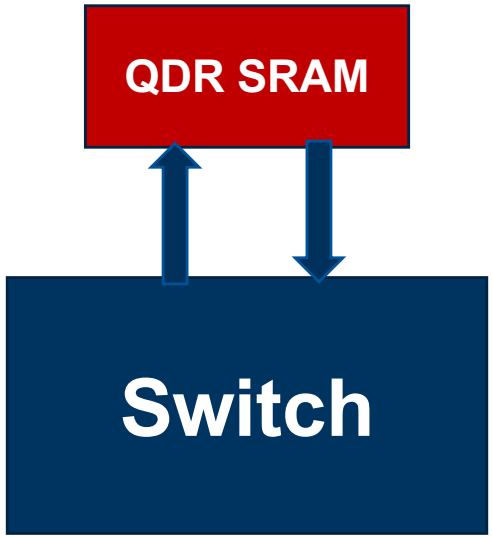
- So far we discussed performance limitations due to:
  - Data path
  - Network Interfaces
- Other common critical paths include:
  - Memory interfaces
    - Lookup tables, packet buffers
  - Host interfaces
    - PCIe, DMA engine

Memory Interfaces

- On chip memories
  - Advantage: fast access time
  - Disadvantage: limited size (10’s of MB)
- Off chip memory:
  - Advantage: large size (up to many GB)
  - Disadvantage: access time, cost, area, power
- New technologies
  - Offer mid-way solutions

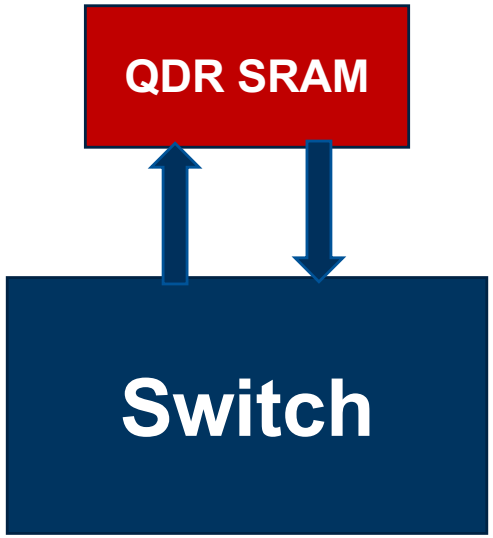
Example: QDR-IV SRAM

- Does 4 operations every clock: 2 READs, 2 WRITEs
- Constant latency
- Maximum random transaction rate: 2132 MT/s
- Maximum bandwidth: 153.3Gbps
- Maximum density: 144Mb
- Example applications: Statistics, head-tail cache, descriptors lists



Example: QDR-IV SRAM

- Does 4 operations every clock: 2 READs, 2 WRITEs
  - *DDR4 DRAM: 2 operations every clock*
- Constant latency
  - *DDR4 DRAM: variable latency*
- Maximum random transaction rate: 2132 MT/s
  - DDR4 DRAM: 20MT/s (worst case! t_{RC}~50ns)
    - DDR4 theoretical best case 3200MT/s
- Maximum bandwidth: 153.3Gbps
  - DDR4 DRAM maximum bandwidth: 102.4Gbps (for 32b (2x16) bus)
- Maximum density: 144Mb
  - **DDR4 maximum density: 16Gb**
- Example applications: Statistics, head-tail cache, descriptors lists
  - No longer applicable: packet buffer

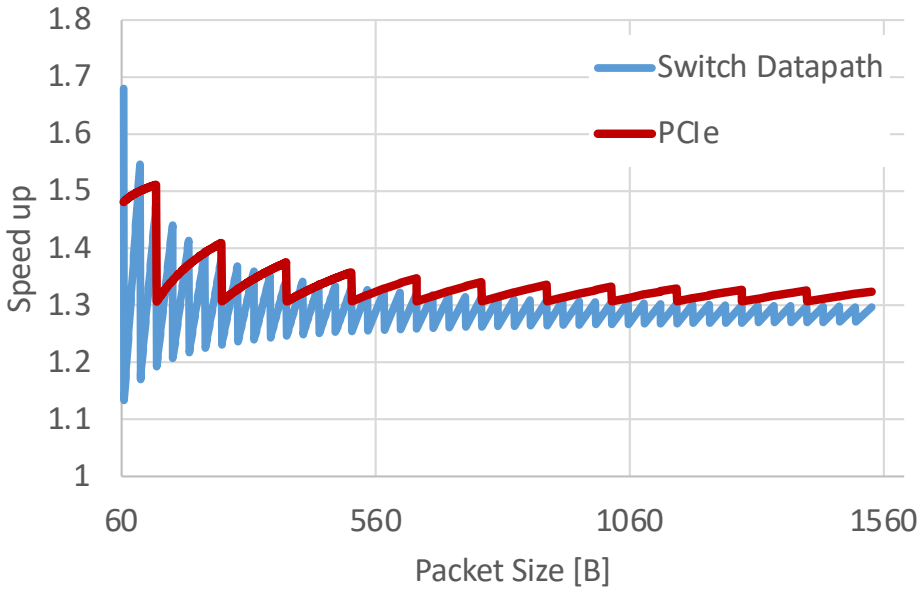


Random Memory Access

- Random access is a “killer” when accessing DRAM based memories
  - Due to strong timing constraints
- Examples: rules access, packet buffer access
- DRAMs perform well (better) when there is strong locality or when accessing large chunks of data
  - E.g. large cache lines, files etc.
  - Large enough to hide timing constraints
    - E.g. for 3200MT/s, 64b bus: 50ns~ 1KB

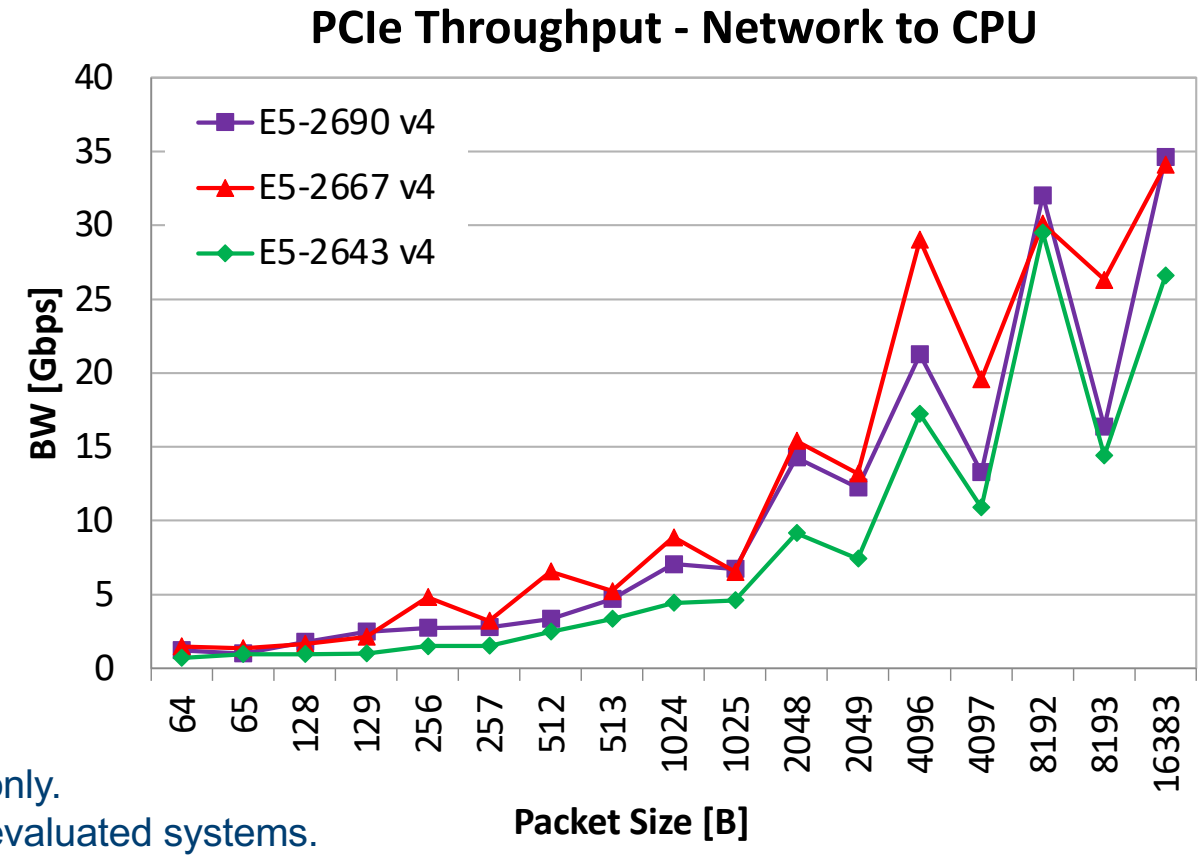
Example: PCI Express Gen 3, x8

- The theoretical performance profile:
- PCIe Gen 3 – each lane runs at 8Gbps
- ~97% link utilization (128/130 coding, scrambling)
- Data overhead – 24B-28B (including headers and CRC)
- Configurable MTU (e.g., 128B, 256B, …)



Example: PCI Express Gen 3, x8

- Actual throughput on VC709, using Xilinx reference project: (same FPGA as NetFPGA SUME)
- This is so far for the performance profile…
- Why?



Note: the graph is for illustration purposes only.
There were slight differences between the evaluated systems.

Thank you