EDA284
Parallel Computer Architecture

Lecture :

Interconnection Networks on-chip (NoCs)

Ioannis Sourdis

Outline of Lecture

- NoCs basics
- NoCs design alternatives:
  - Topologies
  - Flow control
  - Routing
  - Router architecture
- Research on NoCs

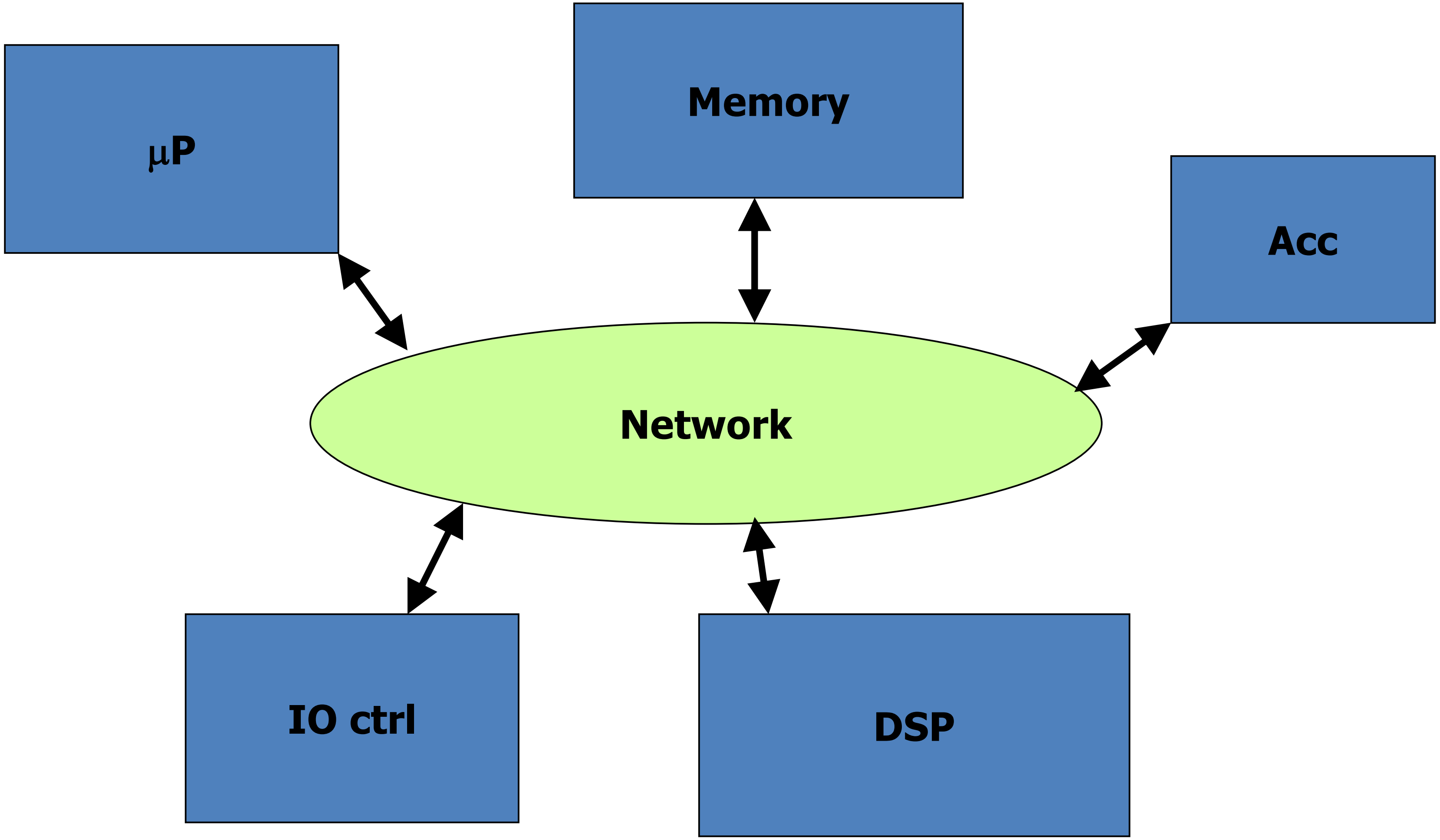
Outline of Lecture

- **NoCs basics**
- NoCs design alternatives:
  - Topologies
  - Flow control
  - Routing
  - Router architecture
- Research on NoCs

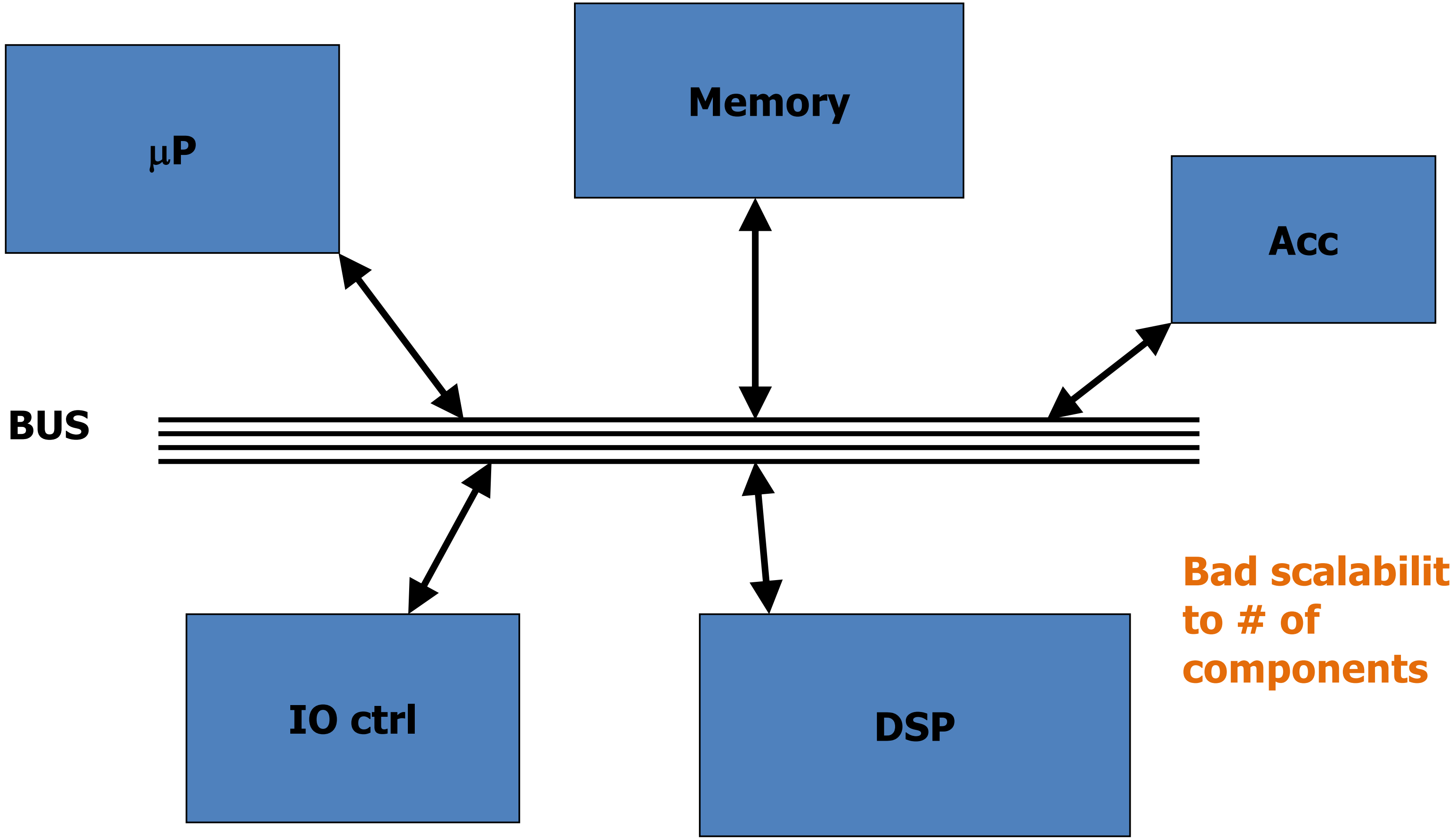
Networks on chip: Why?

- What do they interconnect?
- Why needed?
- What are the design objectives?

Intra chip communication



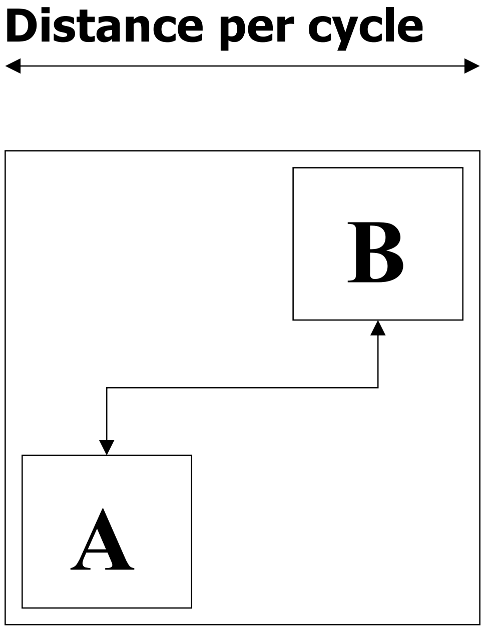
Interconnect with a bus



Bad scalability to # of components

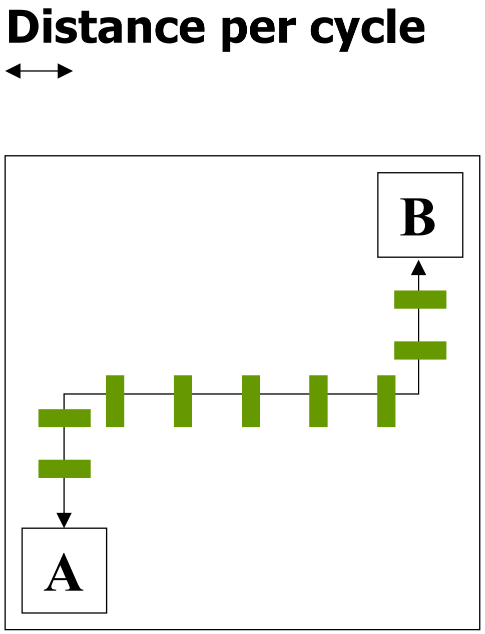
Wire delay/gate delay does not scale

Past



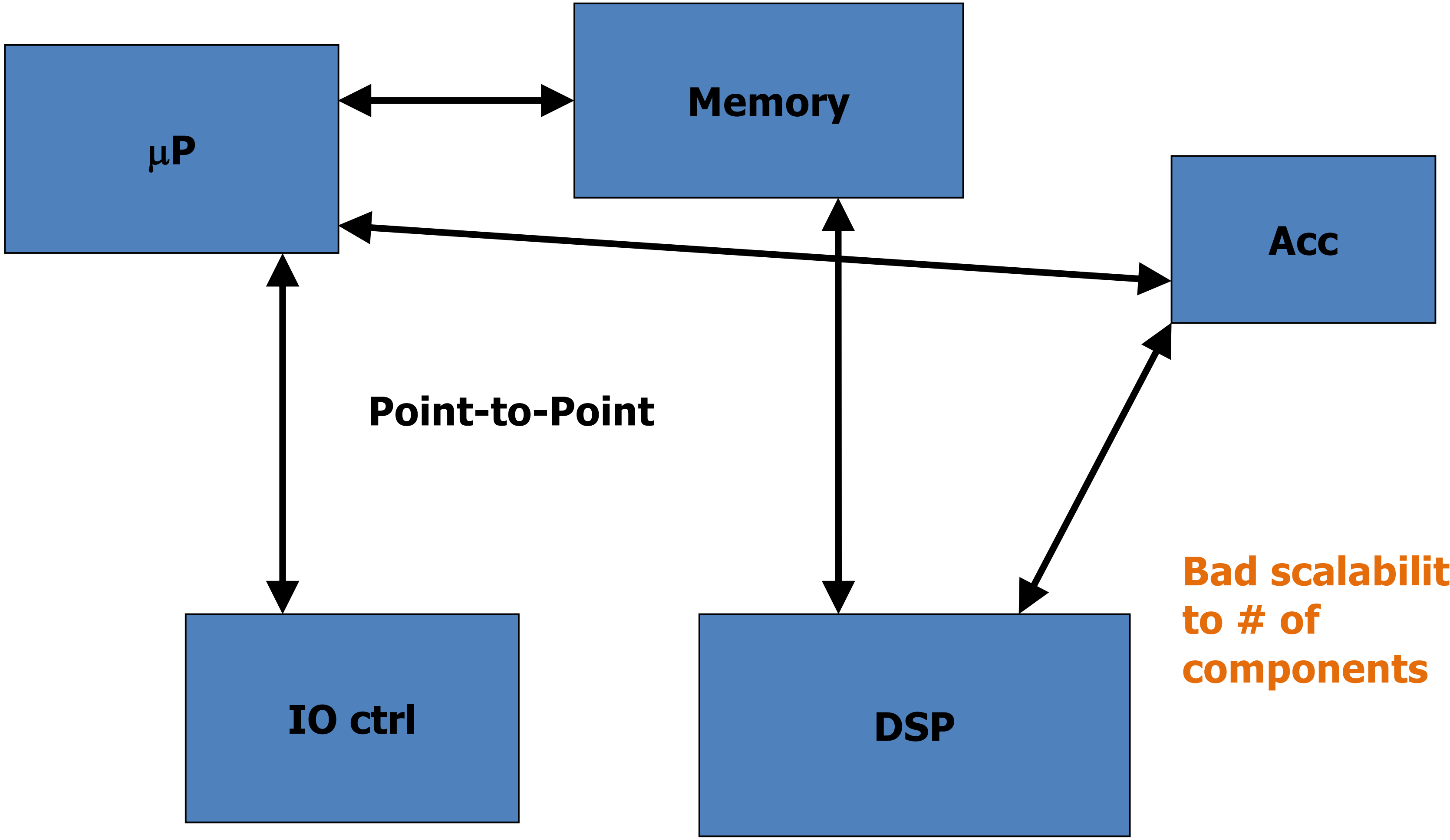
Distance per cycle

Now



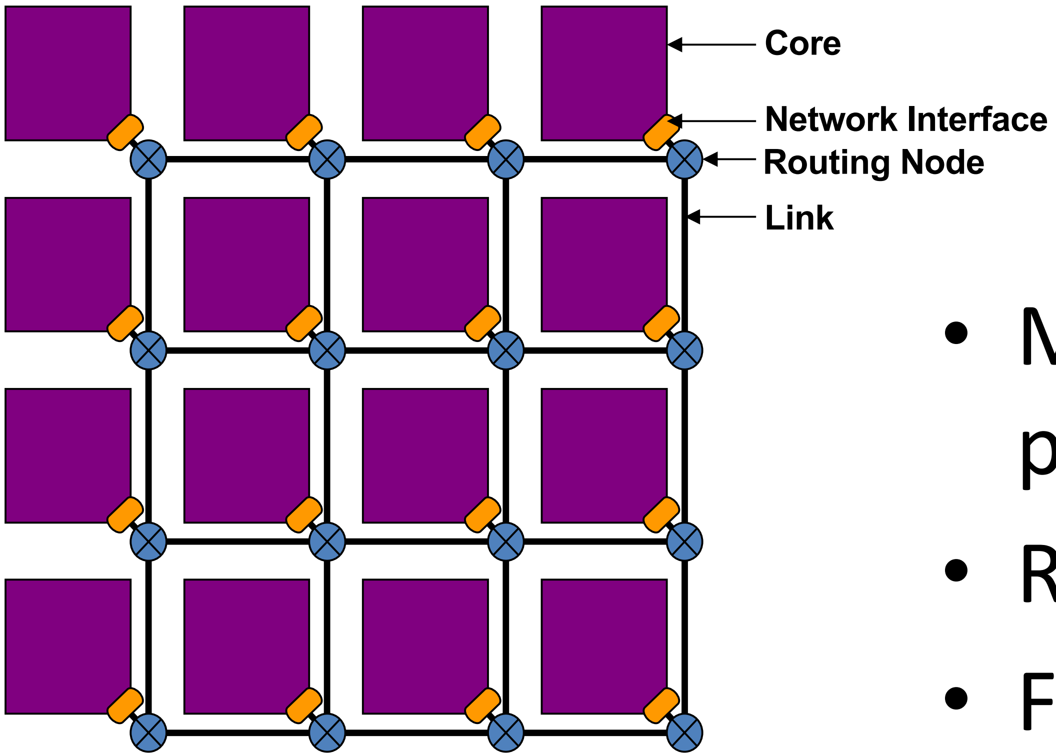
Distance per cycle

Introduction – intra chip communication



Bad scalability to # of components

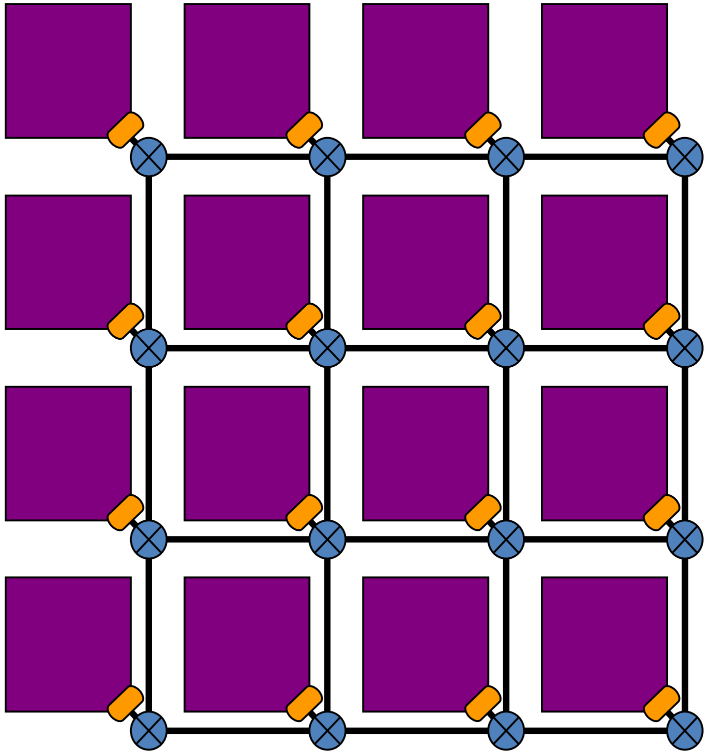
Important Paradigms for NoC



- More scalable performance
- Resource Re-use
- Flexibility

What factors determine which interconnect solution you pick?

- Performance
  - Latency
  - Throughput
- Energy efficiency & Power
  - Energy per transferred bit
- Other metrics:
  - Quality of Service
  - Fault tolerance



On-Chip vs. Off-Chip Differences

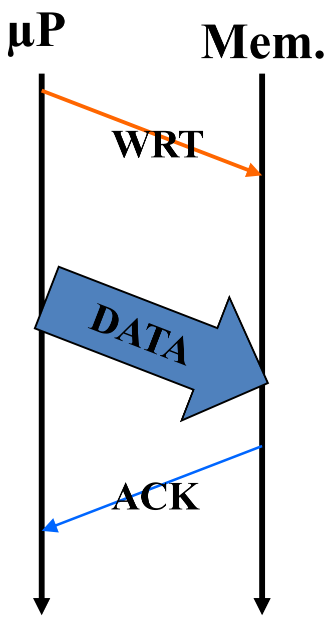
Advantages of on-chip

- **Wires are “free”**
  - Can build highly connected networks with wide buses
- **Low latency**
  - Can cross entire network in few clock cycles
- **High Reliability**
  - Packets are not dropped and links rarely fail

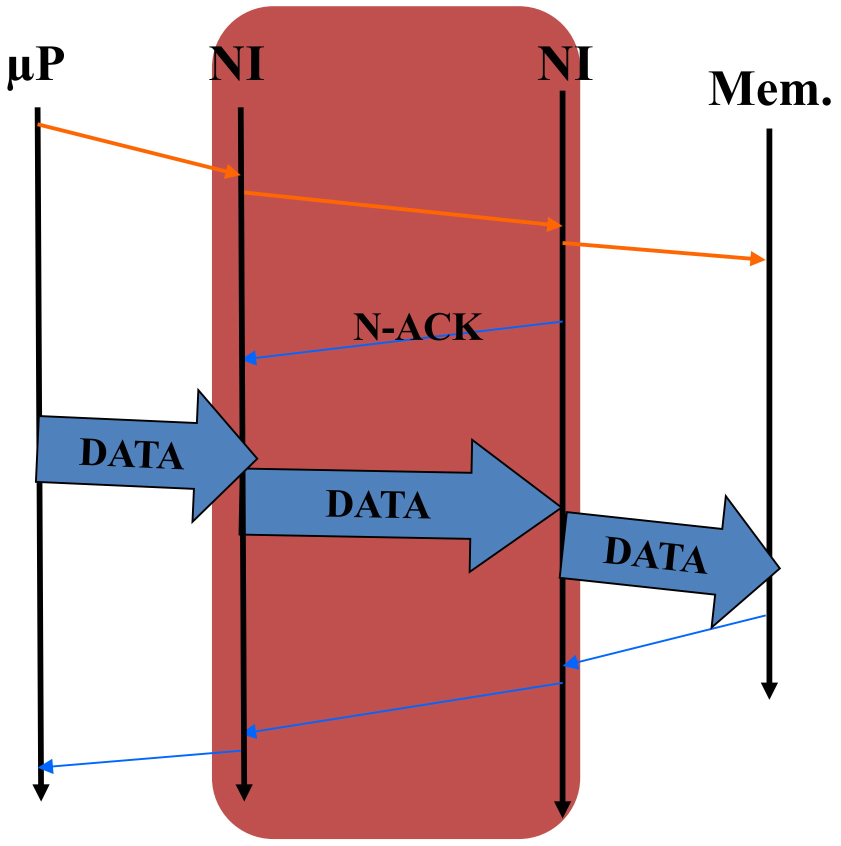
Disadvantages of on-chip

- **Sharing resources with rest of components on chip**
  - Area
  - Power
- **Limited buffering available**
- **Not all topologies map well to 2D plane**

Network Usage Example

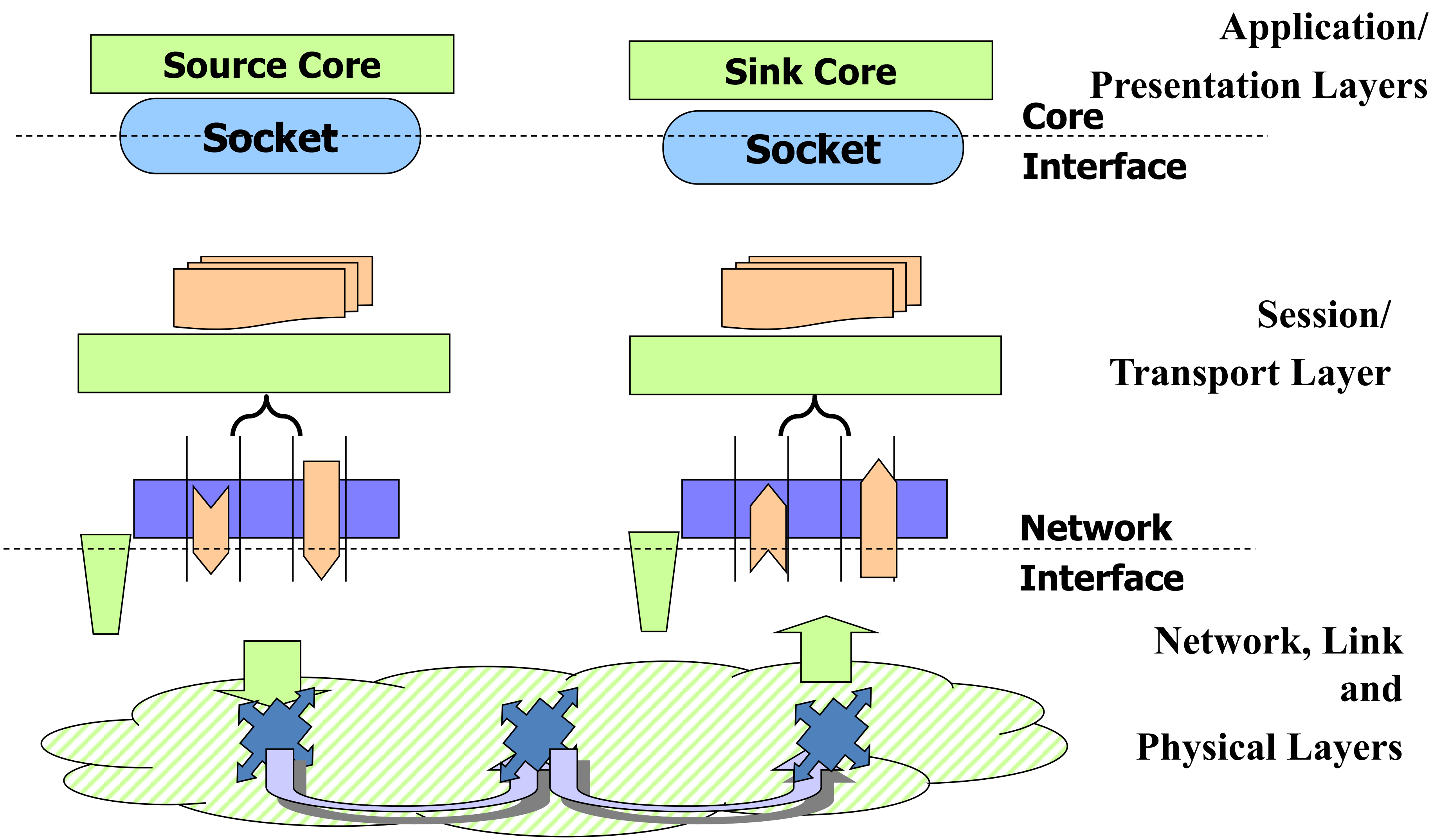


Typical P2P Write Session
(3 comm. events)

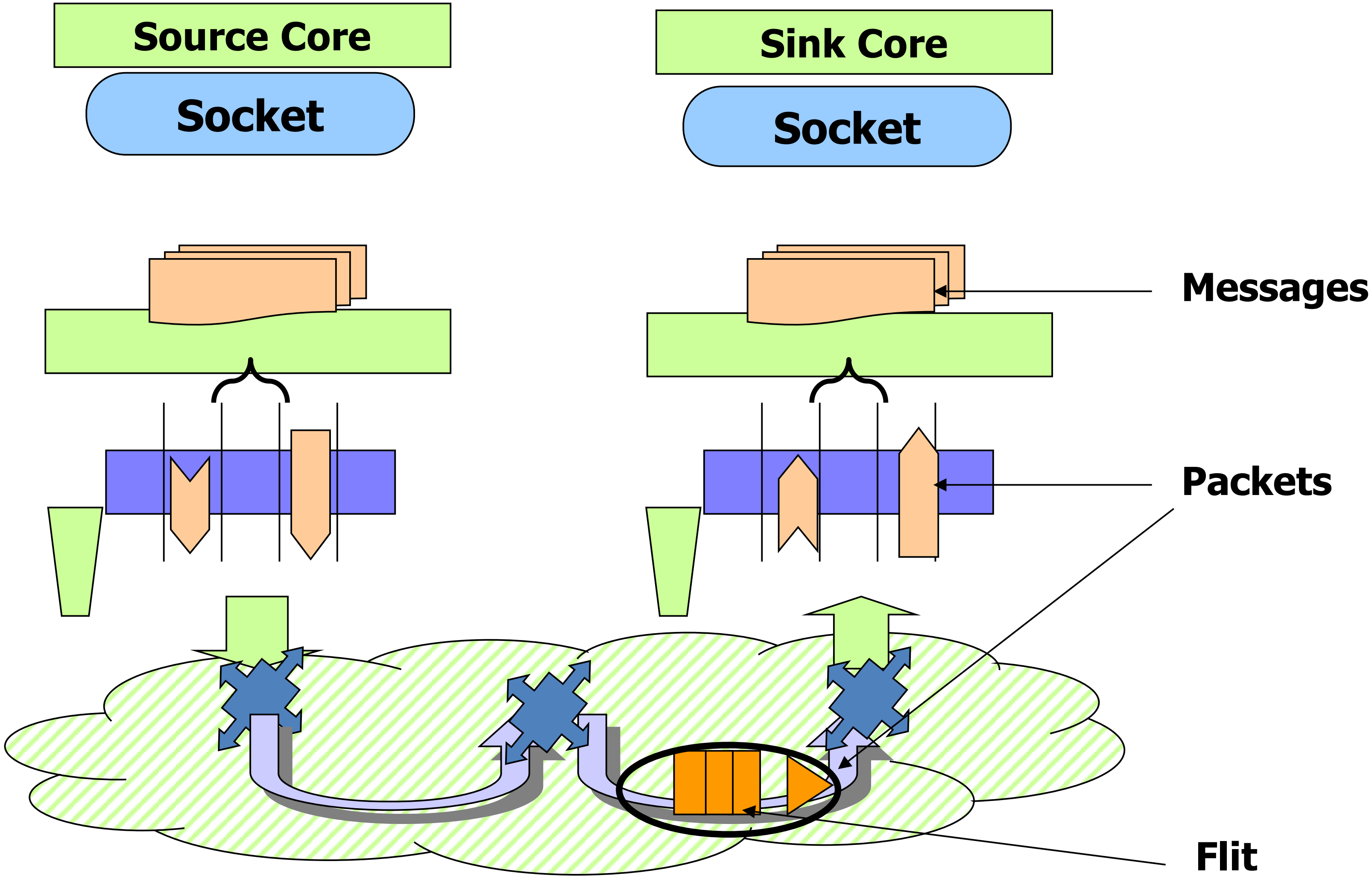


"Networked" Write Session
(4 comm. events)

Network Abstraction



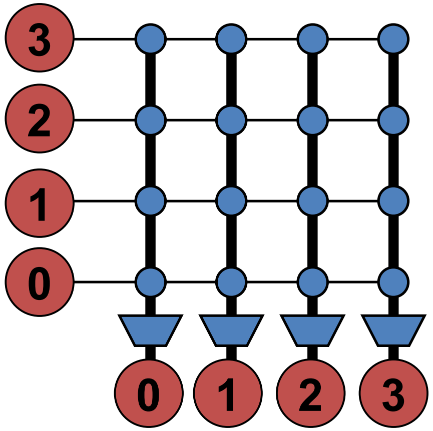
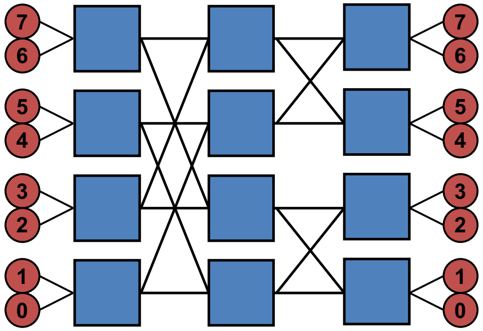
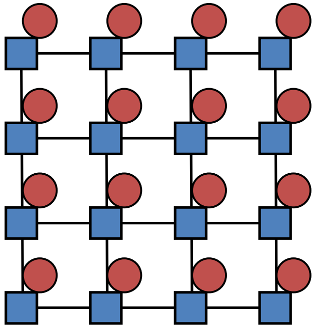
Network Dataflow View



Outline of Lecture

- NoCs basics
- NoCs design alternatives:
  - **Topologies**
  - Flow control
  - Routing
  - Router architecture
- Research on NoCs

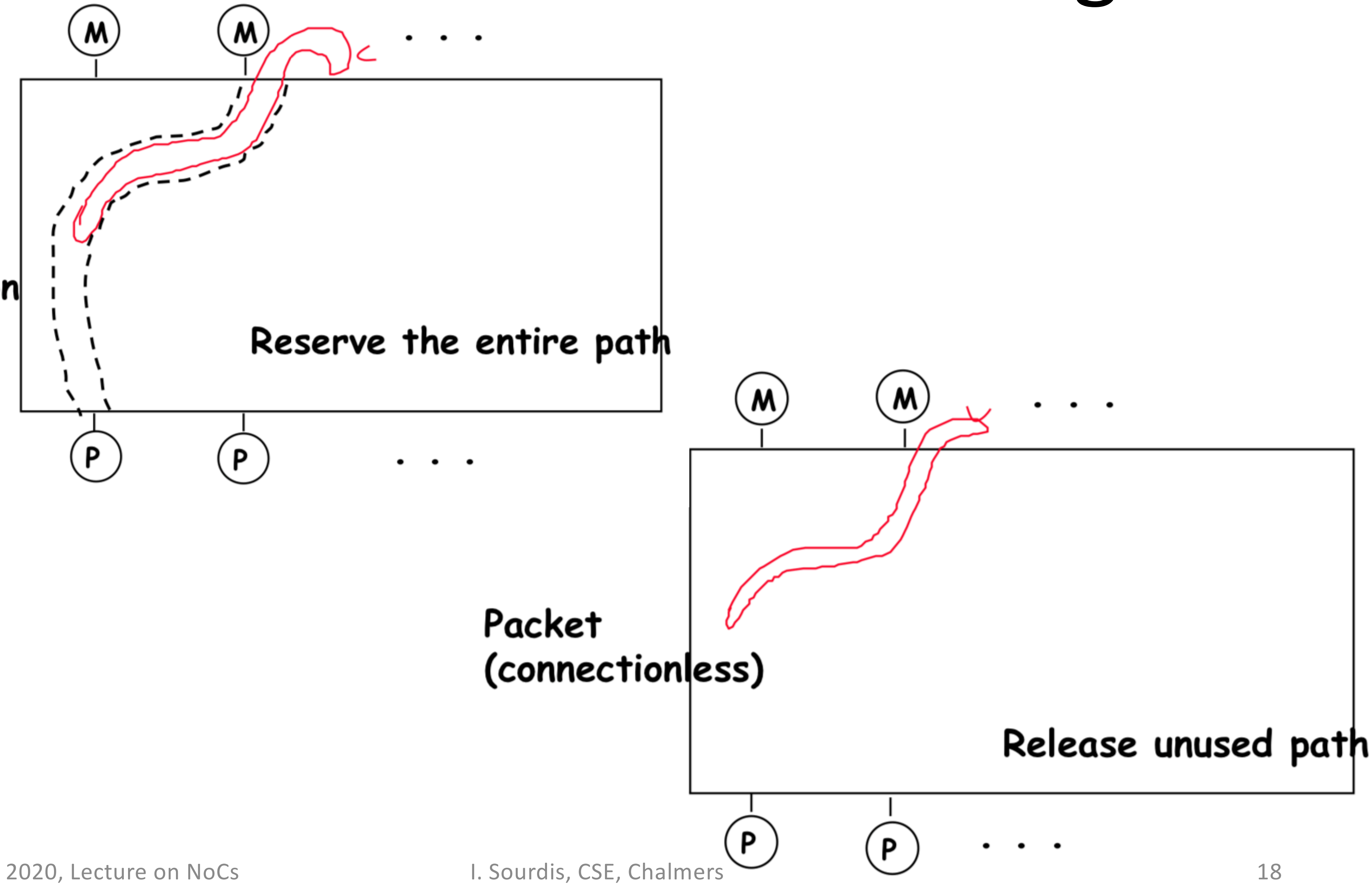
Review: Topologies

		 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0	
Topology	**Crossbar**	**Multistage Logarith.**	**Mesh**
Direct/Indirect	**Indirect**	**Indirect**	**Direct**
Blocking/ Non-blocking	**Non-blocking**	**Blocking**	**Blocking**
Cost	**$O(N^2)$**	**O(NlogN)**	**O(N)**
Latency	**O(1)**	**O(logN)**	**O(sqrt(N))**

Outline of Lecture

- NoCs basics
- NoCs design alternatives:
  - Topologies
  - **Flow control**
  - Routing
  - Router architecture
- Research on NoCs

Flow Control: Circuit vs. Packet switching



Circuit (connection Oriented)

Reserve the entire path

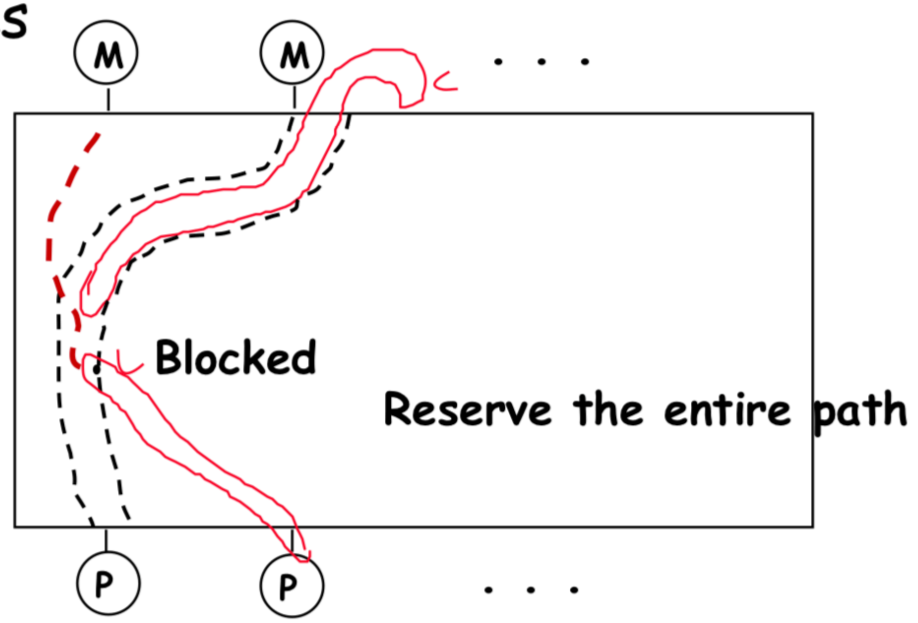
Packet (connectionless)

Release unused path

Flow Control: Circuit vs. Packet switching

Issues

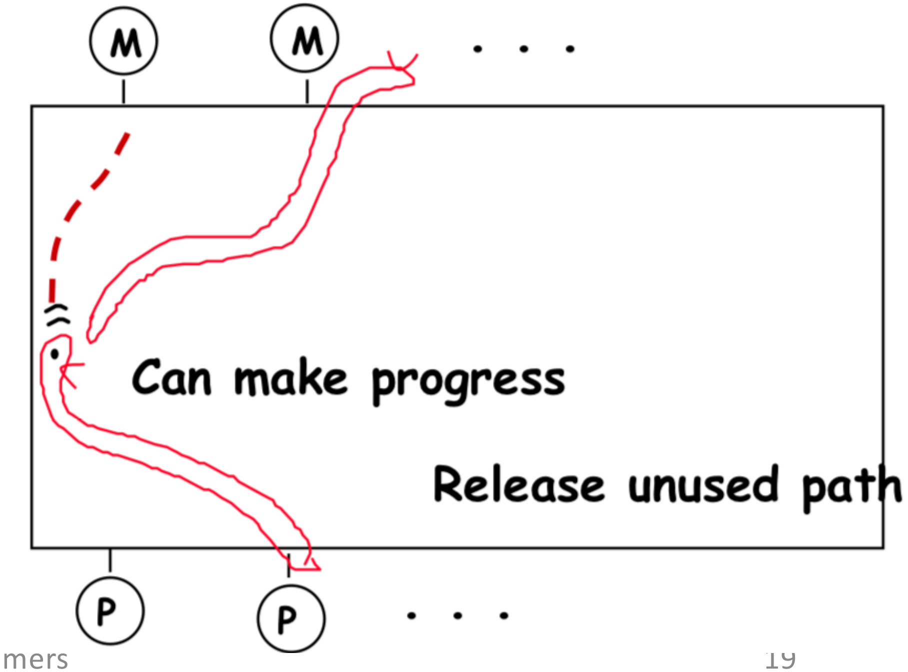
Circuit



Blocked

Reserve the entire path

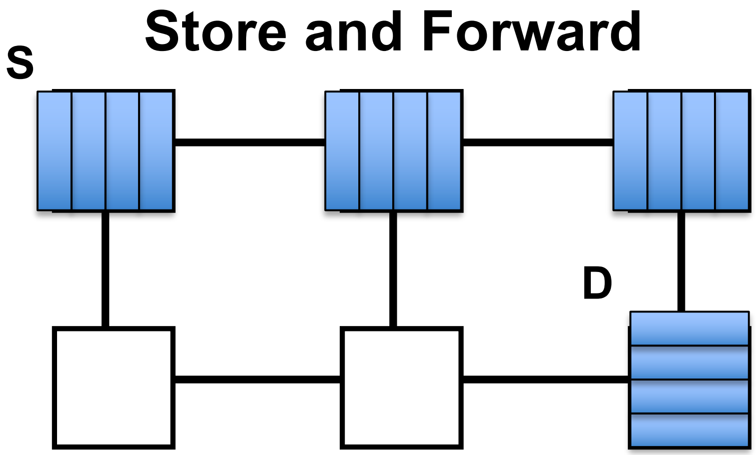
Packet



Can make progress

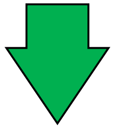
Release unused path

Review: Flow Control



Any other issues?

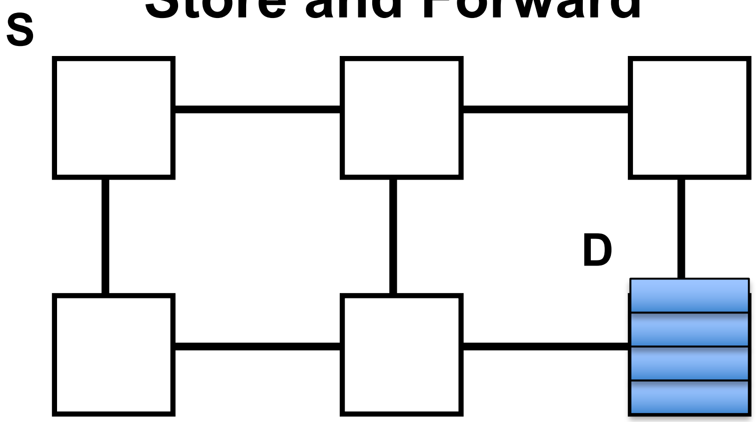
Head-of-Line Blocking



Use Virtual Channels

Review: Flow Control

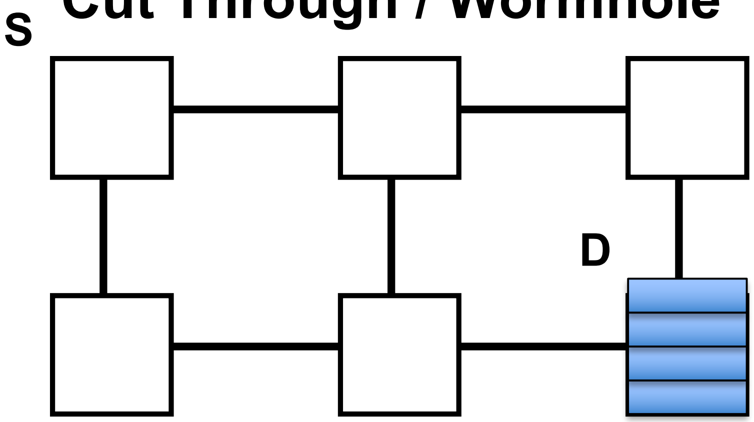
Store and Forward



Shrink Buffers

Reduce latency

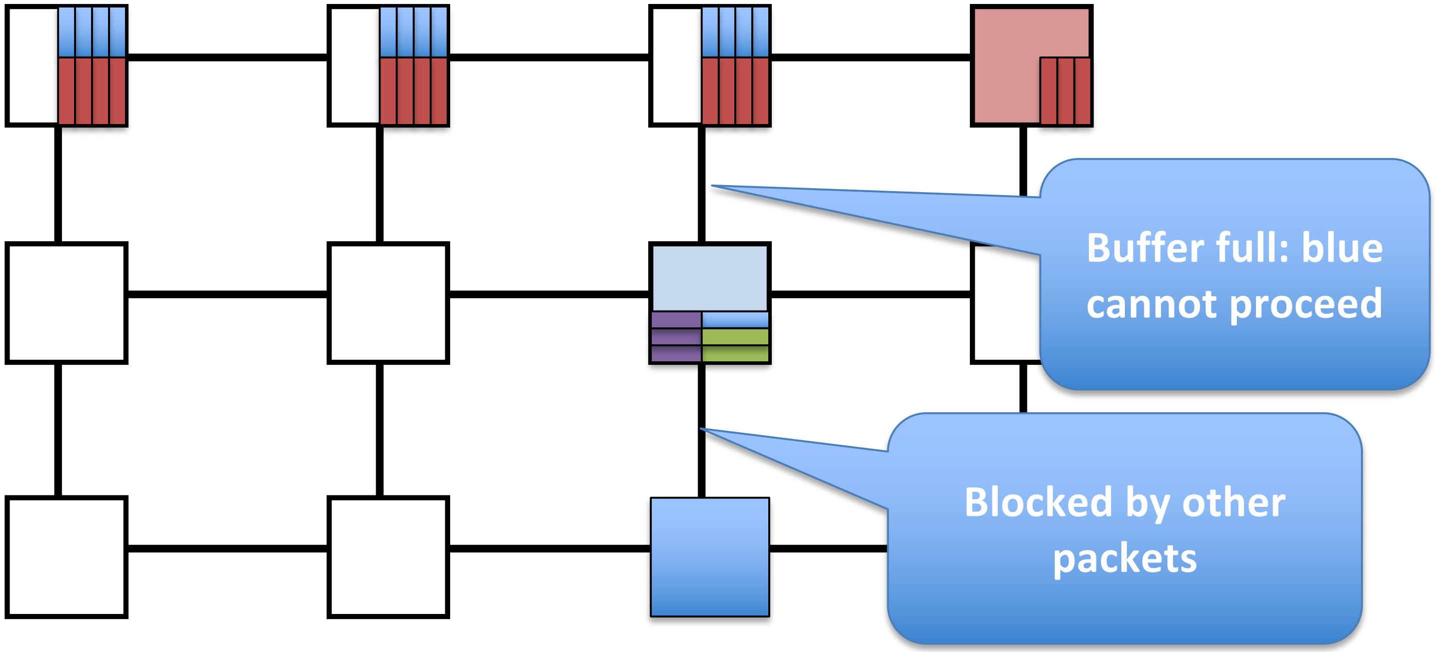
Cut Through / Wormhole



Any other issues?

Head-of-Line Blocking

Use Virtual Channels



Outline of Lecture

- NoCs basics
- NoCs design alternatives:
  - Topologies
  - Flow control
  - **Routing**
  - Router architecture
- Research on NoCs

Routing Mechanism

- Arithmetic
  - Simple arithmetic to determine route in regular topologies
  - Dimension order routing in meshes/tori
- Source Based
  - Source specifies output port for each switch in route

  + Simple switches
    - no control state: strip output port off header

  - Large header
- Table Lookup Based
  - Index into table for output port

  + Small header

  - More complex switches

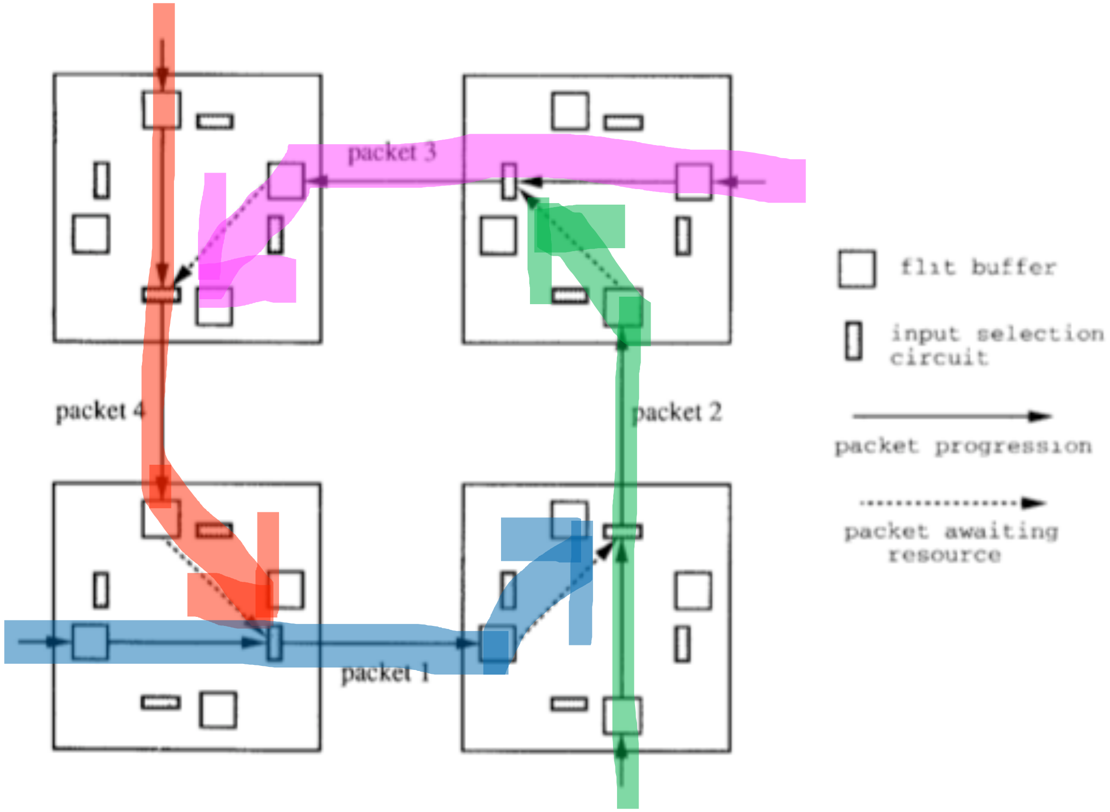
Routing Algorithm

- Types
  - **Oblivious**: do not consider network state (e.g., random)
    - **Deterministic**: always choose the same path
  - **Adaptive**: adapt to state of the network

- How to adapt
  - Local/global feedback
  - Minimal or non-minimal paths

Deadlock

- No forward progress
- Caused by circular dependencies on resources
- Each packet waits for a buffer occupied by another packet downstream



Turn Model to Avoid Deadlock

- Idea
  - Analyze directions in which packets can turn in the network
  - Determine the cycles that such turns can form
  - Prohibit just enough turns to break possible cycles
- Glass and Ni, "The Turn Model for Adaptive Routing," ISCA 1992.

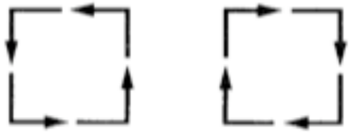


FIG. 2. The possible turns and simple cycles in a two-dimensional mesh.

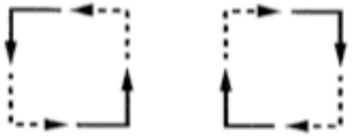


FIG. 3. The four turns allowed by the *xy* routing algorithm.

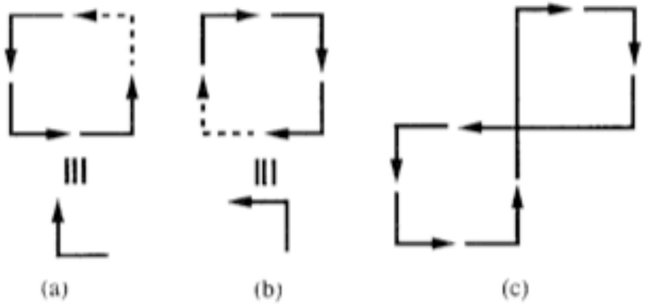
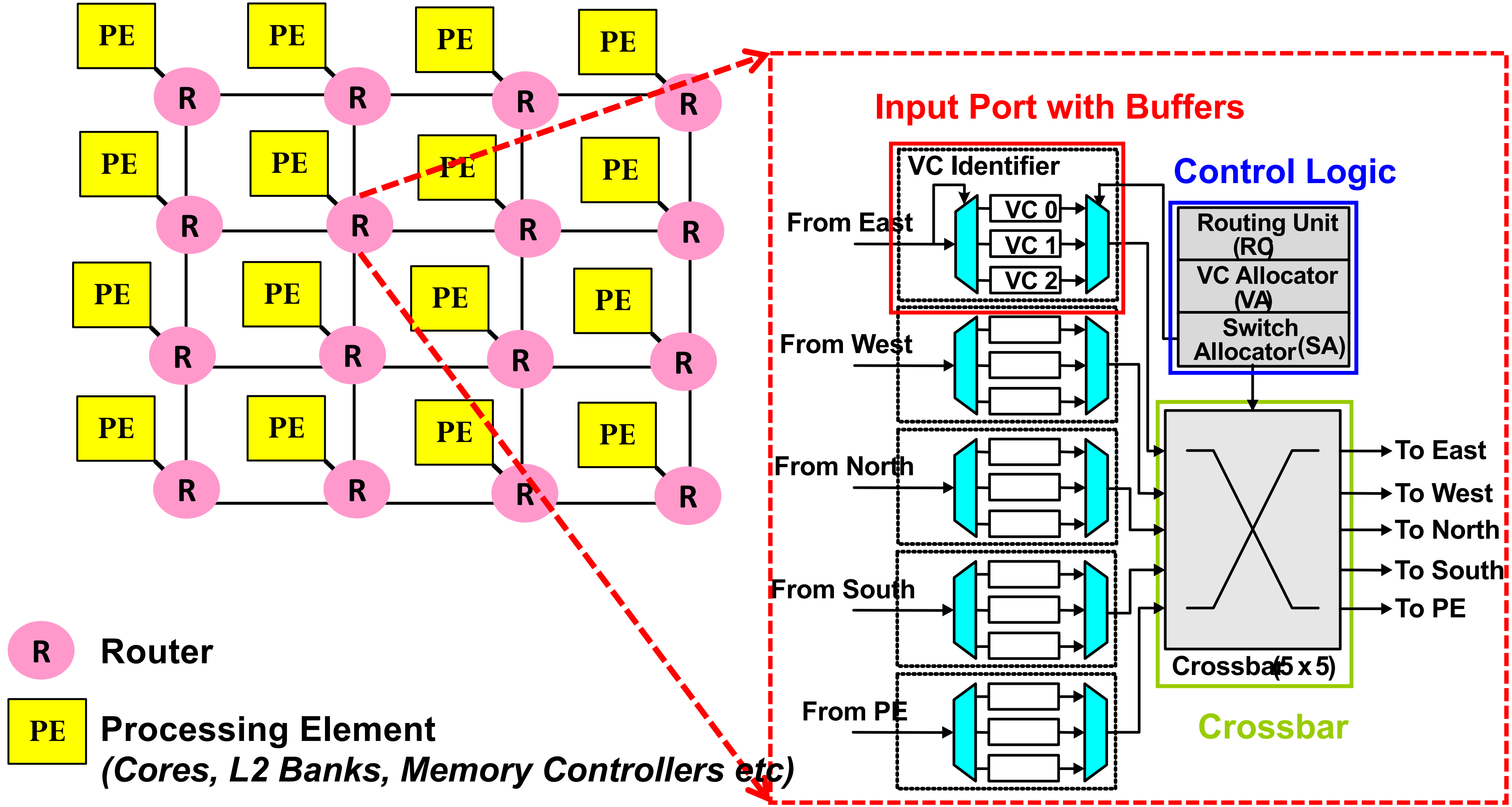


FIG. 4. Six turns that complete the cycles and allow deadlock.

Outline of Lecture

- NoCs basics
- NoCs design alternatives:
  - Topologies
  - Flow control
  - Routing
  - **Router architecture**
- Research on NoCs

On-chip Networks



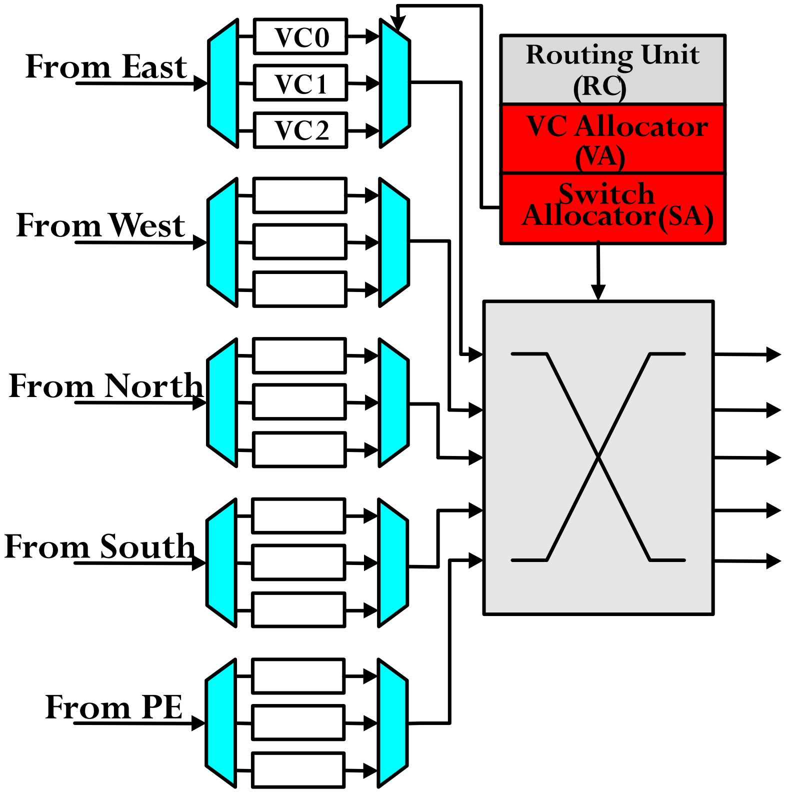
R Router

PE Processing Element

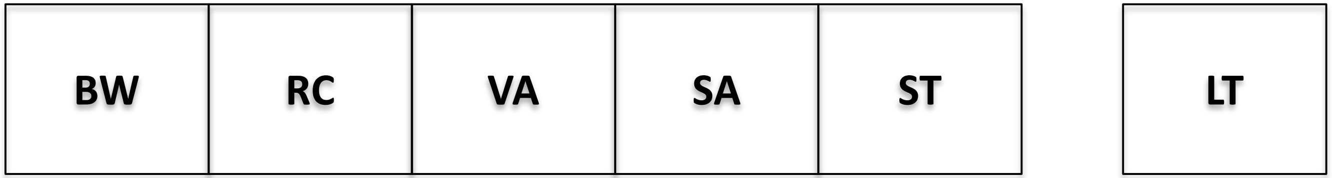
(Cores, L2 Banks, Memory Controllers etc)

Router Design: Functions of a Router

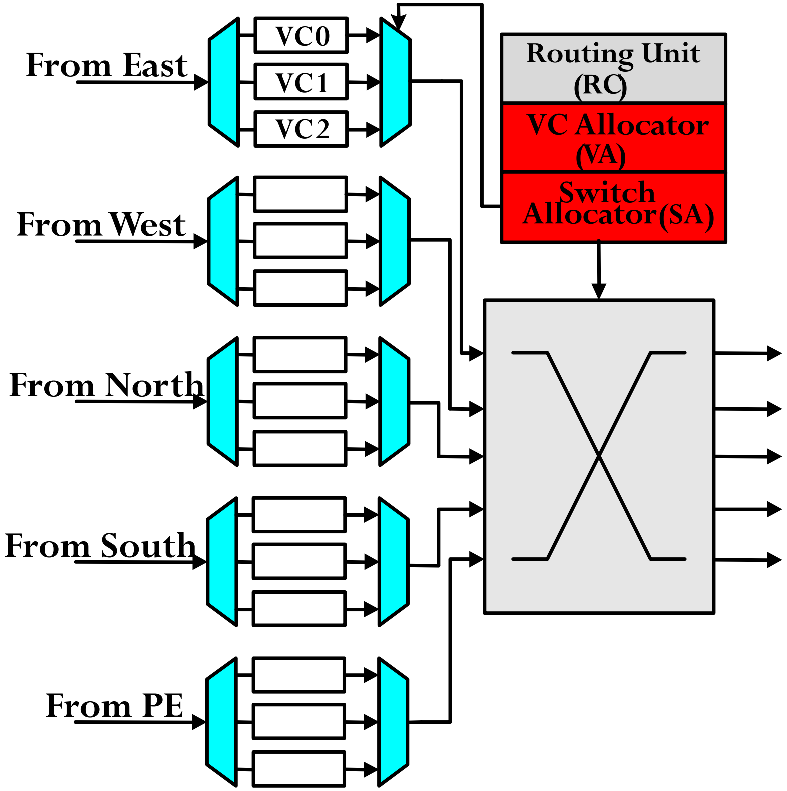
- Buffering (of flits)
- Route computation
- Arbitration of flits (i.e. prioritization) when contention
  - Called packet scheduling
- Switching
  - From input port to output port



Router Pipeline



- Five logical stages
  - BW: Buffer Write
  - RC: Route computation
  - VA: Virtual Channel Allocation
  - SA: Switch Allocation
  - ST: Switch Traversal
  - LT: Link Traversal

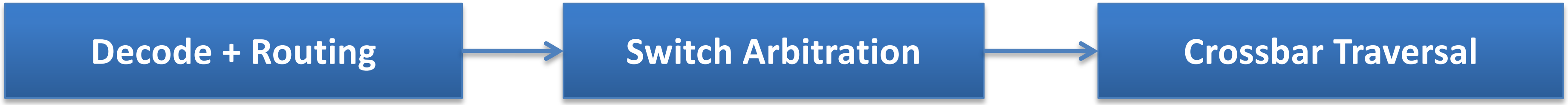


Wormhole Router Timeline

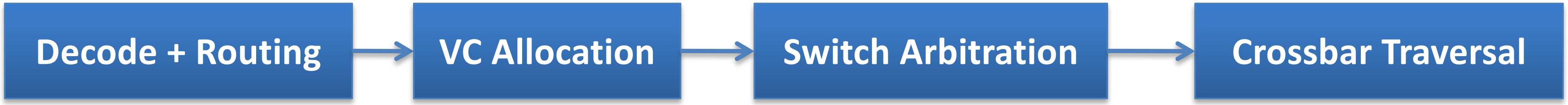
Head	BW	RC	VA	SA	ST	LT			
Body 1		BW	☒	☒	SA	ST	LT		
Body 2			BW	☒	☒	SA	ST	LT	
Tail				BW	☒	☒	SA	ST	LT

- Route computation performed once per packet
- Virtual channel allocated once per packet
- Body and tail flits inherit this information from head flit

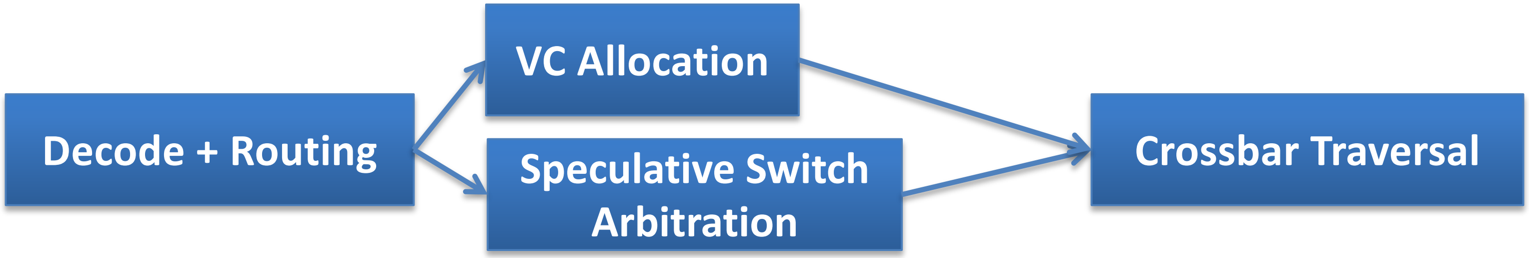
Dependencies in a Router


Decode + Routing
Switch Arbitration
Crossbar Traversal

Wormhole Router


Decode + Routing
VC Allocation
Switch Arbitration
Crossbar Traversal

Virtual Channel Router


VC Allocation
Decode + Routing
Speculative Switch Arbitration
Crossbar Traversal

Speculative Virtual Channel Router

Dependence between output of one module and input of another

- Determine critical path through router
- Cannot bid for switch port until routing performed

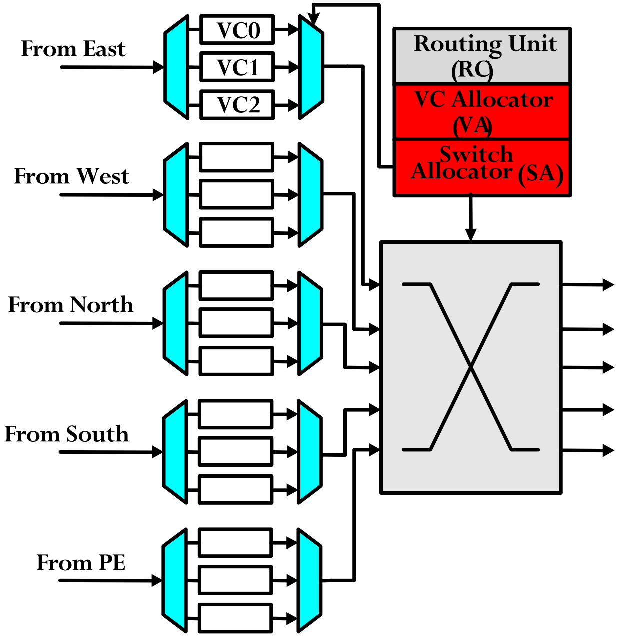
Pipeline Optimizations: Lookahead Routing

BW	RC	VA	SA	ST	LT

- At current router perform routing computation for next router
  - Overlap with BW

BW RC	VA	SA	ST	LT

  - Precomputing route allows flits to compete for VCs immediately after BW
  - RC decodes route header
  - Routing computation needed at next hop
    - Can be computed in parallel with VA



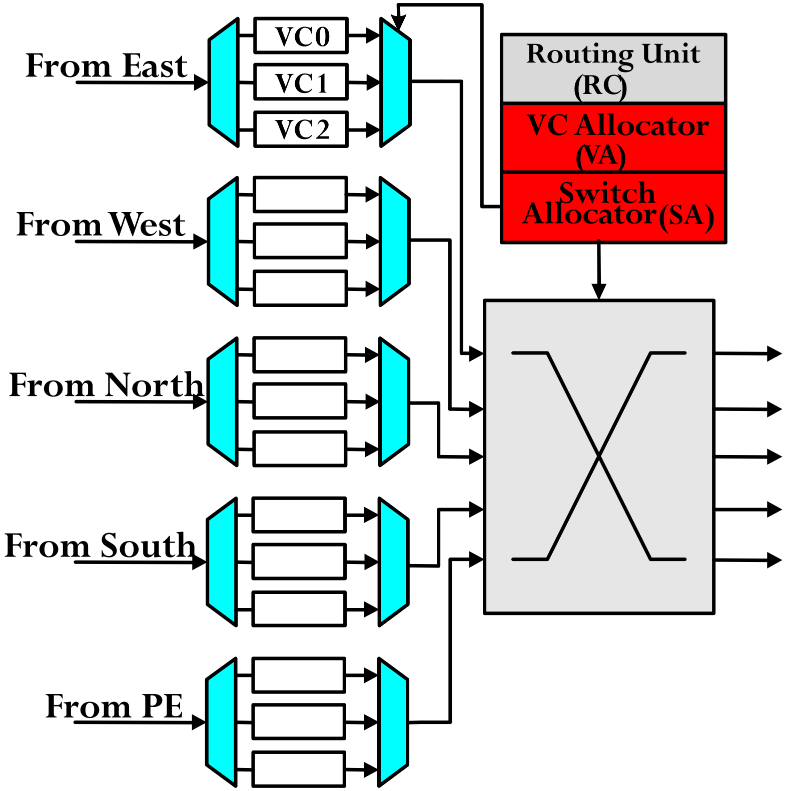
- Galles, “Spider: A High-Speed Network Interconnect,” IEEE Micro 1997.

Pipeline Optimizations: Speculation

- Assume that Virtual Channel Allocation stage will be successful
  - Valid under low to moderate loads
- Entire VA and SA in parallel

BW **RC**	**VA** **SA**	**ST**	**LT**

- If VA unsuccessful (no virtual channel returned)
  - Must repeat VA/SA in next cycle
- Prioritize non-speculative requests

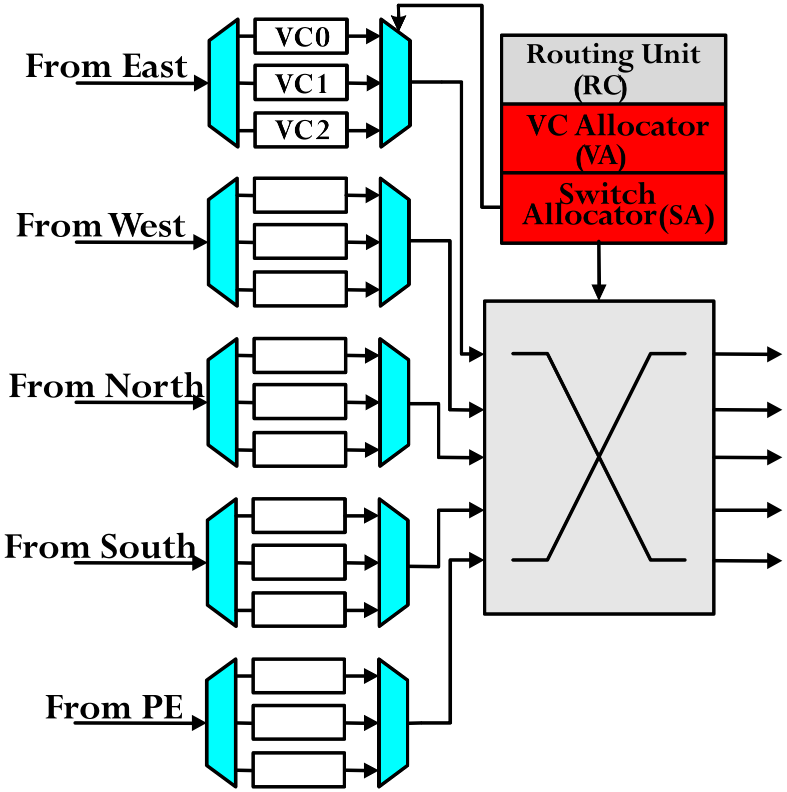


Pipeline Optimizations: Bypassing

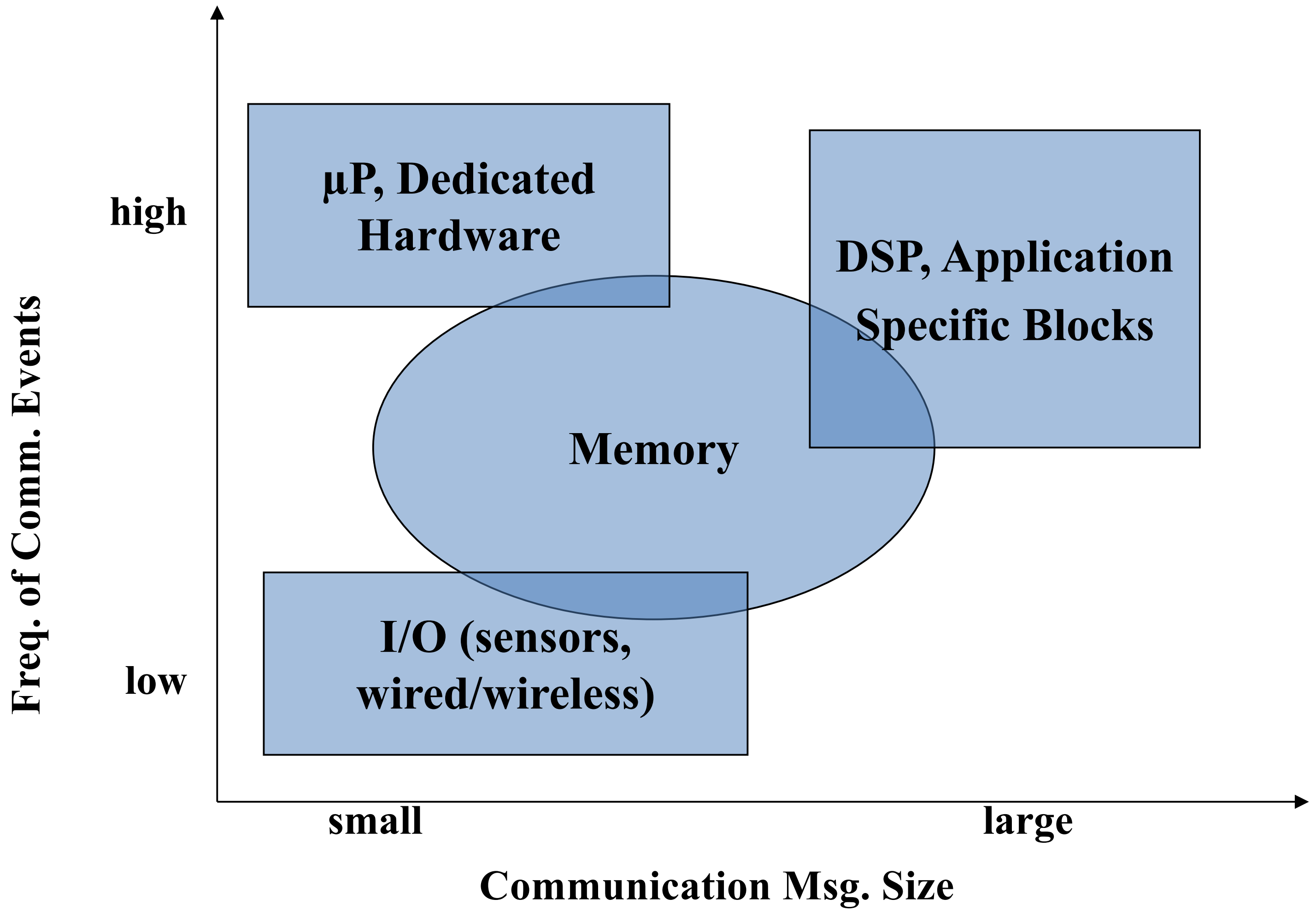
- When no flits in input buffer
  - Speculatively enter ST
  - On port conflict, speculation aborted

VA **RC** **Setup**	**ST**	**LT**

  - In the first stage, a free VC is allocated, next routing is performed and the crossbar is setup



Application Layer Traffic Characterization



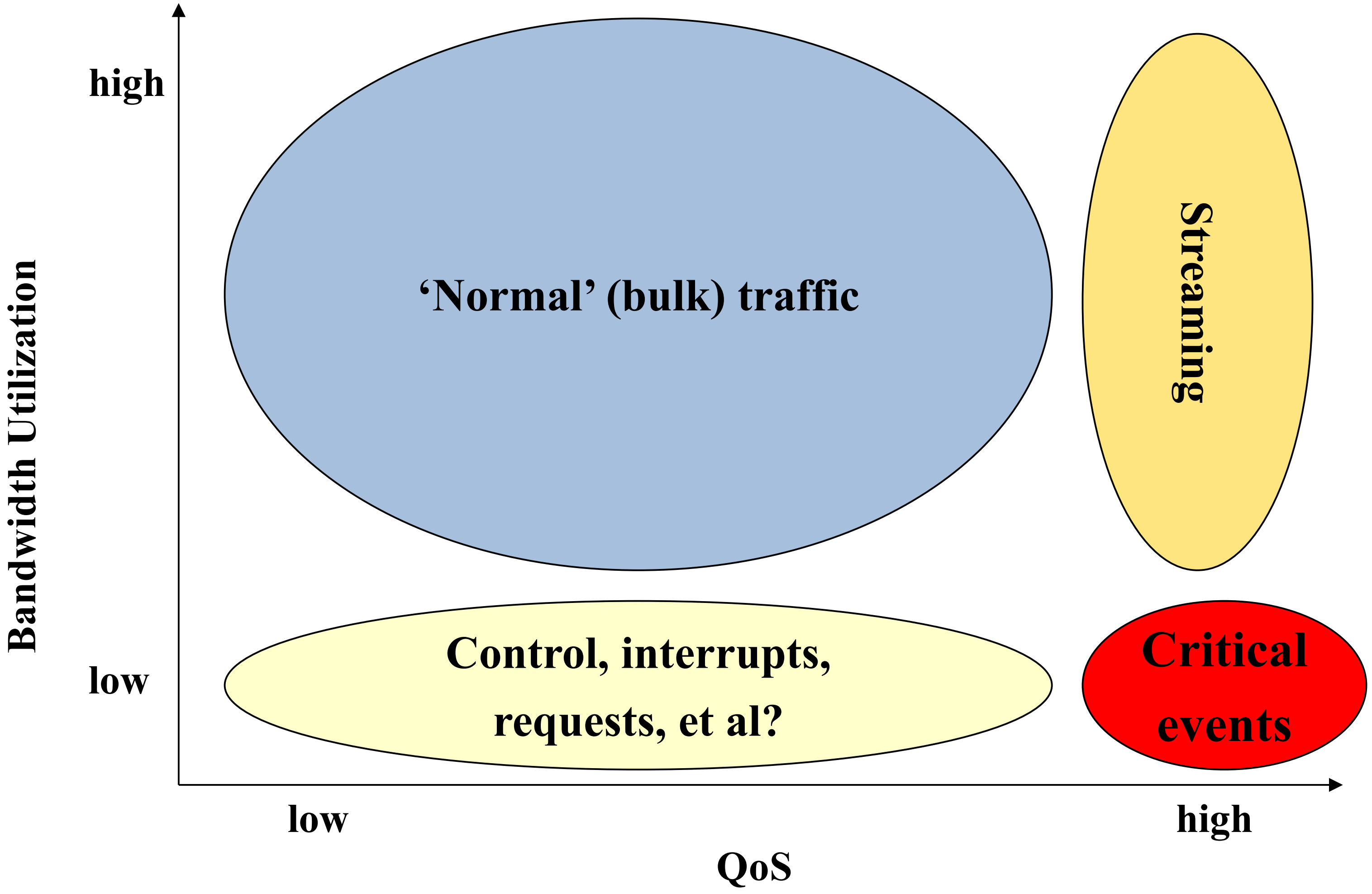
µP, Dedicated Hardware

DSP, Application Specific Blocks

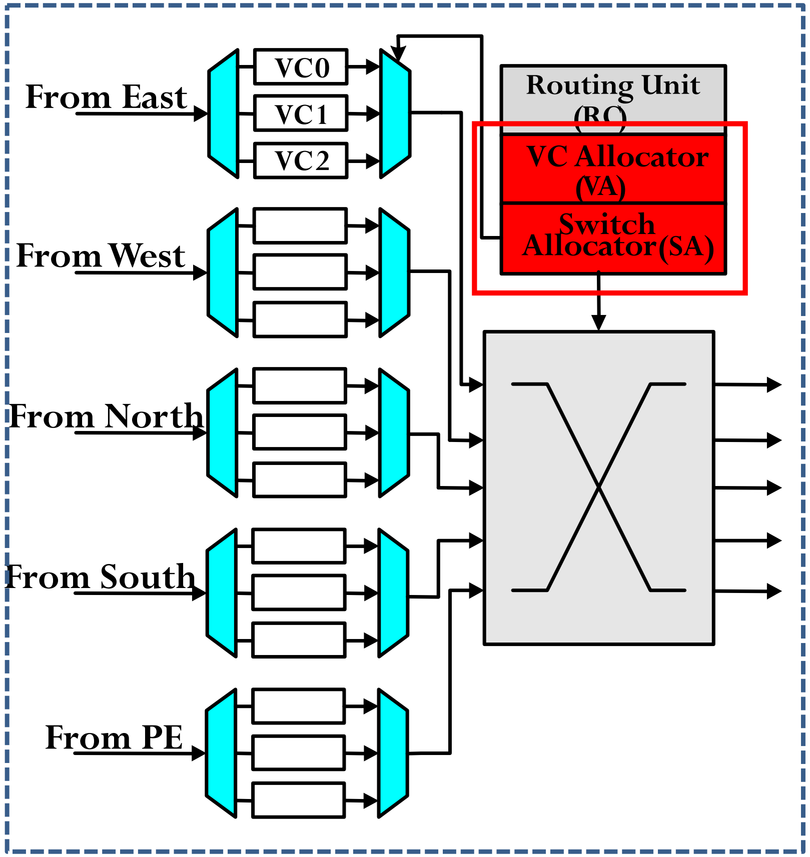
Memory

I/O (sensors, wired/wireless)

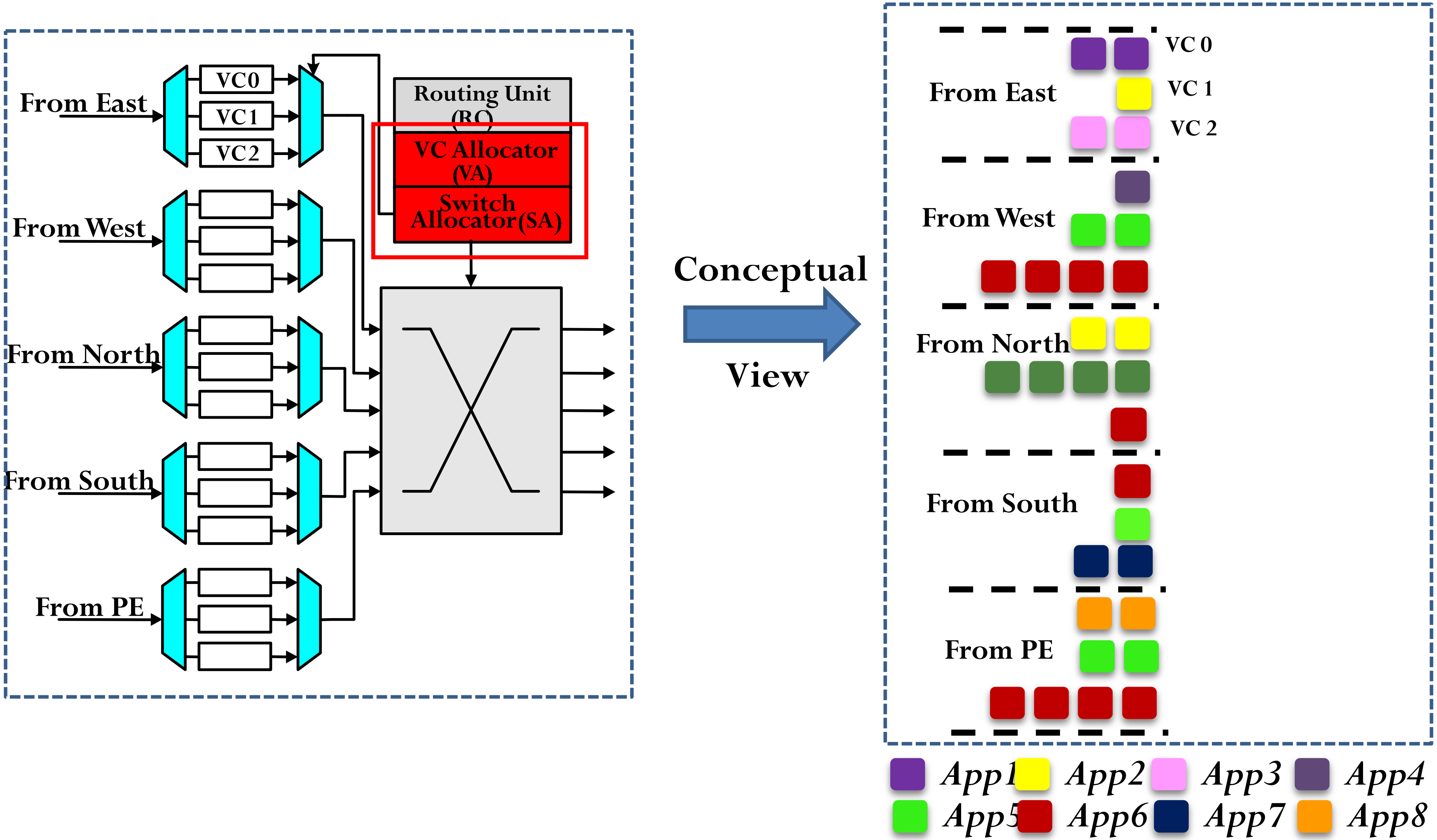
Traffic Distribution



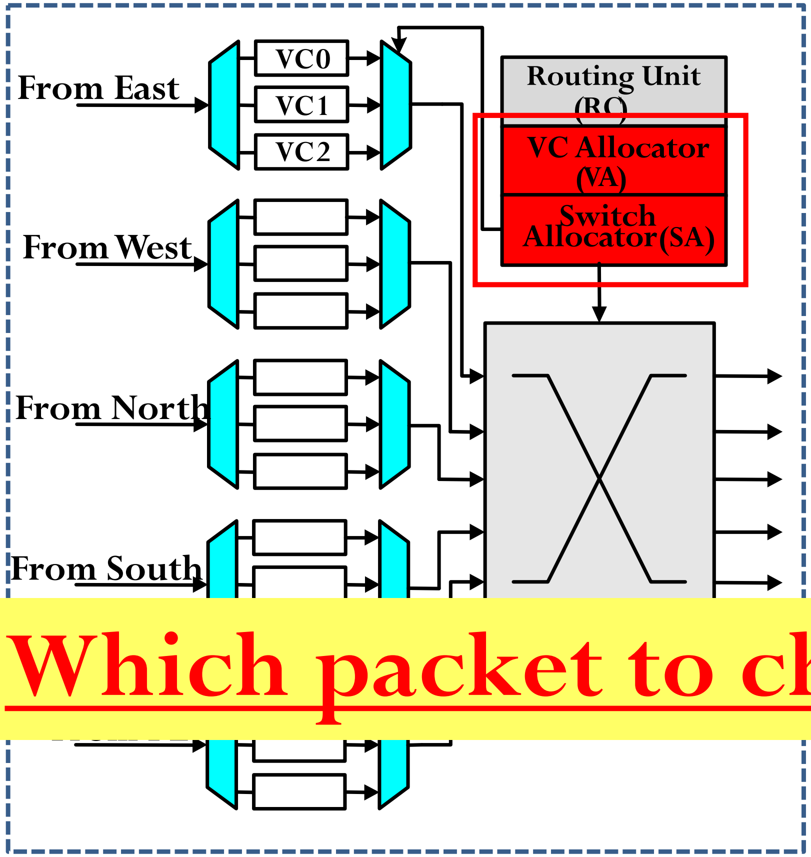
The Problem: Packet Scheduling



The Problem: Packet Scheduling

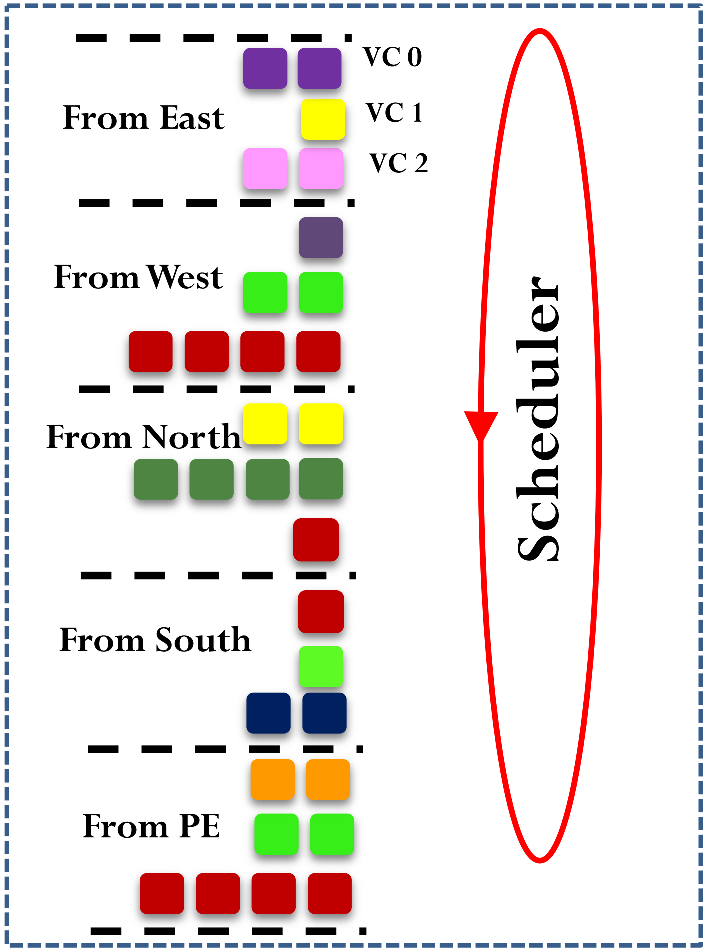


The Problem: Packet Scheduling



Which packet to choose?

Conceptual View

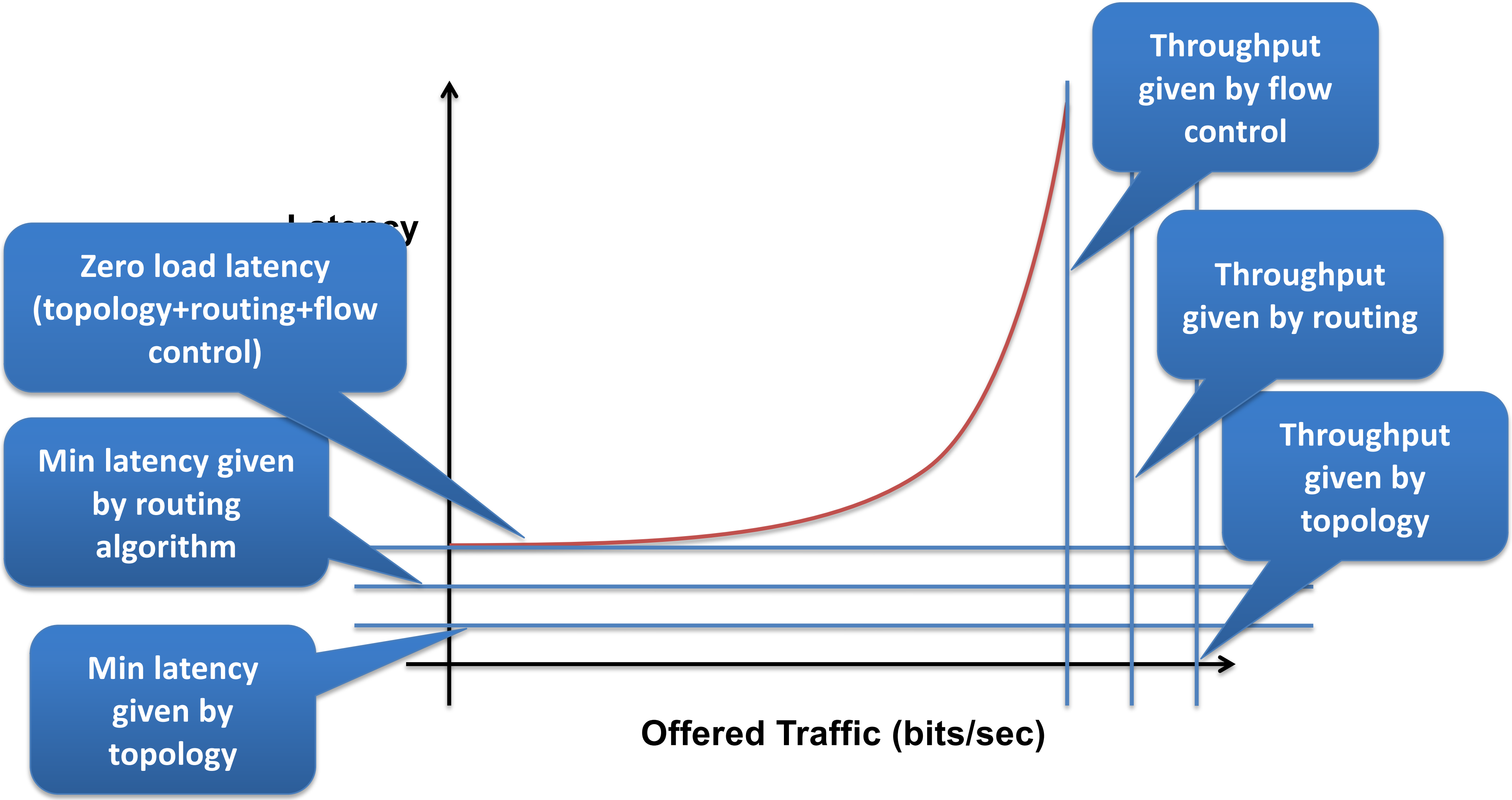


🟪 *App1* 🟨 *App2* ■ *App3* ■ *App4*
🟩 *App5* 🟥 *App6* 🟦 *App7* 🟧 *App8*

Outline of Lecture

- NoCs basics
- **NoCs design alternatives:**
  - Topologies
  - Flow control
  - Routing
  - Router architecture
  - Packet scheduling
- Research on NoCs

Interconnection Network Performance



Tilera Networks

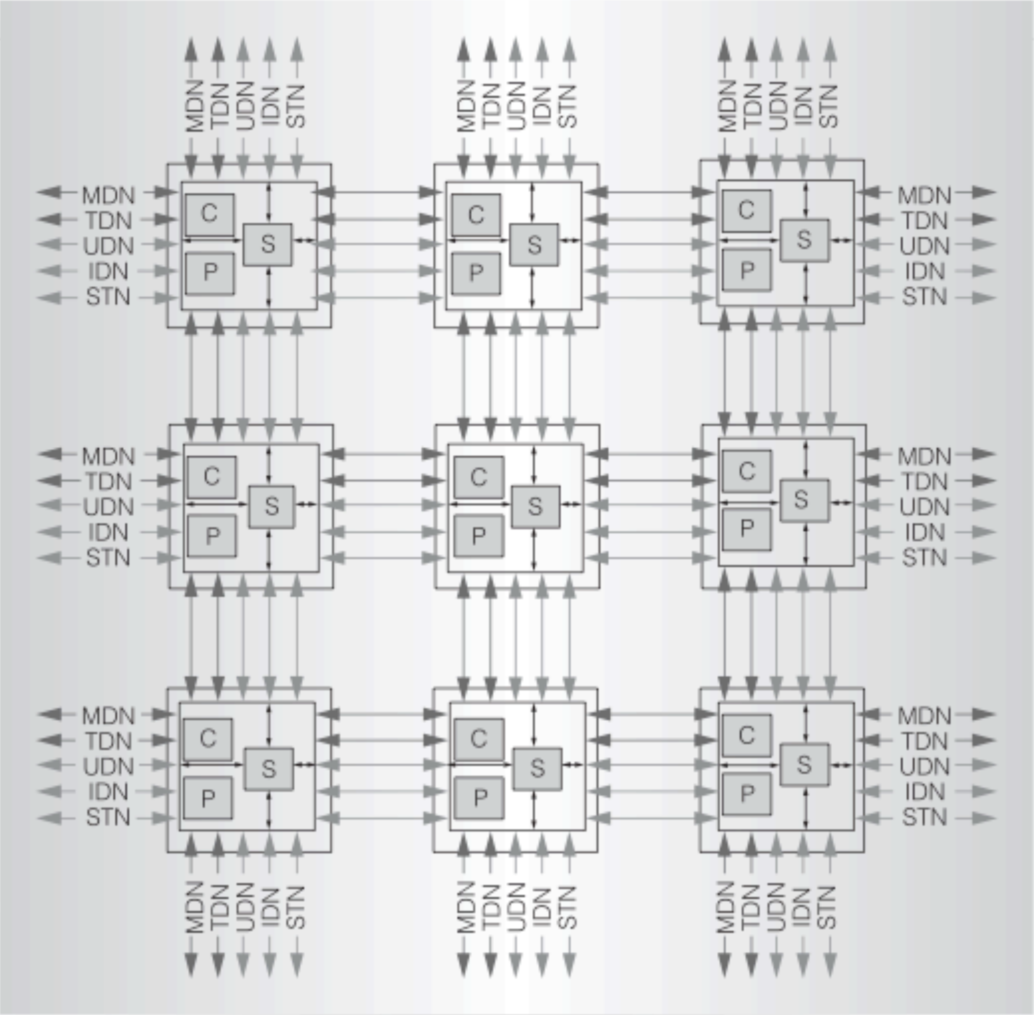


Figure 3. A 3 × 3 array of tiles connected by networks. (MDN: memory dynamic network; TDN: tile dynamic network; UDN: user dynamic network; IDN: I/O dynamic network; STN: static network.)

- 2D Mesh
- Five networks
- Four packet switched
  - Dimension order routing, wormhole flow control
  - TDN: Cache request packets
  - MDN: Response packets
  - IDN: I/O packets
  - UDN: Core to core messaging
- One circuit switched
  - STN: Low-latency, high-bandwidth static network
  - Streaming data

Outline of Lecture

- NoCs basics
- NoCs design alternatives:
  - Topologies
  - Flow control
  - Routing
  - Router architecture
  - Packet scheduling
- **Research on NoCs**

Research Topics in NoCs

Plenty of topics in on-chip networks. Examples:

- Performance:
  - Reduce packet latency
  - Improve Throughput
- Energy/power efficient/proportional design
- Adaptivity: Ability to adapt to different access patterns
- QoS, performance isolation, prioritization
  - Reducing and controlling interference, admission control
  - Request prioritization, priority inversion, coherence, …
- Co-design of NoCs with other shared resources
  - End-to-end performance, QoS, power/energy optimization
- Scalable topologies to many cores
- Fault tolerance
- New technologies (optical, 3D, …)

NoC research at Chalmers

- Freeway NoC
- RQNoC

HighwayNoC:
Objectives And Key Concepts

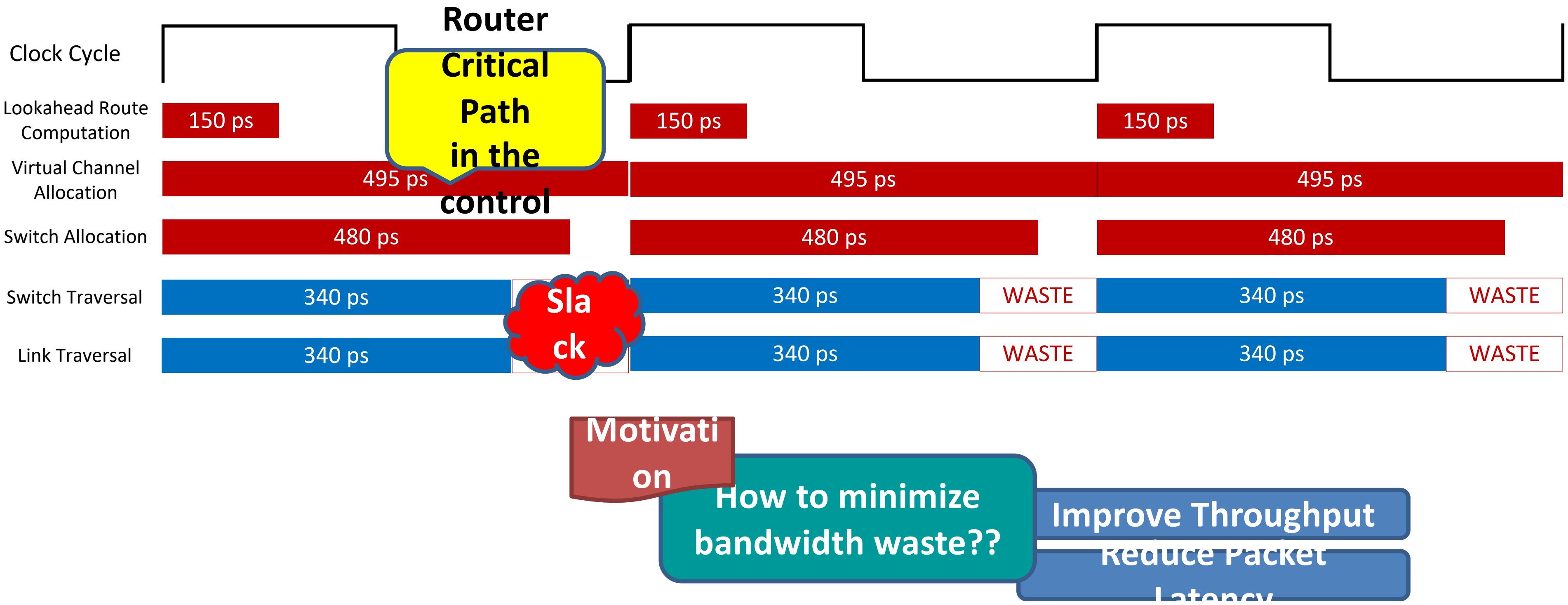
Primary objective is to improve performance:

- Improve network throughput
- Reduce packet latency

FreewayNoC is based on two concepts:

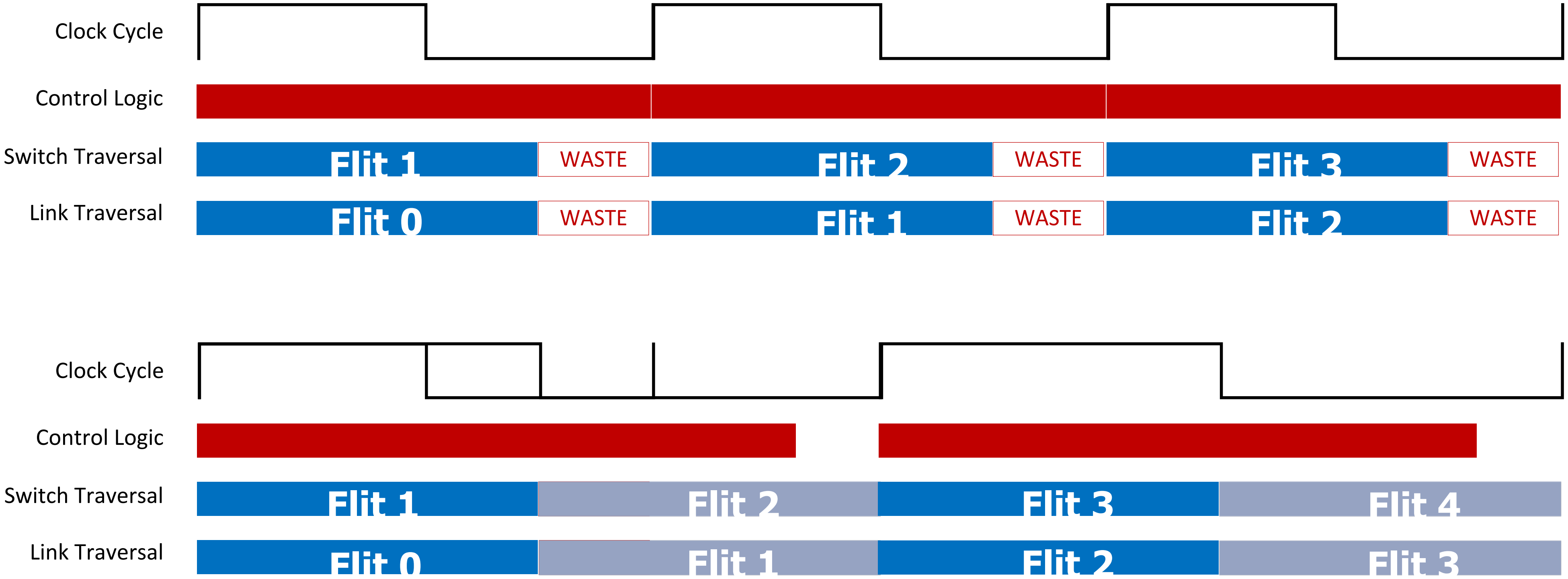
1. **Operate datapath (ST, LT) at DDR to maximize its utilization**
2. **Provide a simplified pipeline stage bypassing to reduce latency**

NoCs Datapath Is Underutilized



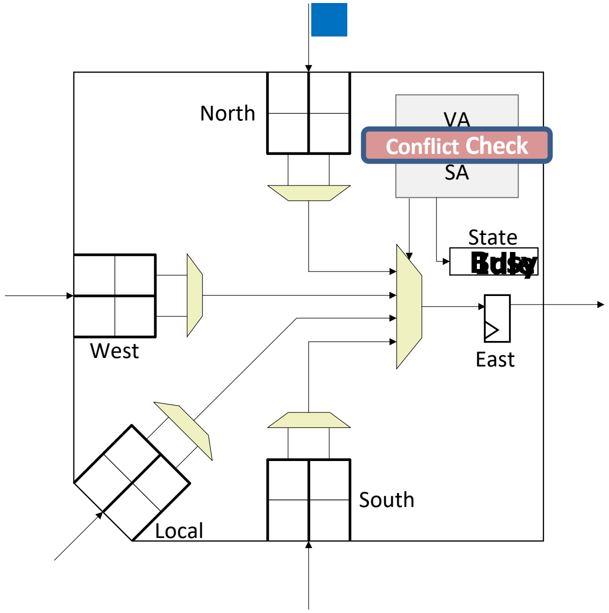
DAT093, 2018, Lecture on NoCs

Operate Datapath In DDR



HighwayNoC: Improve DDRNoC Latency Using Pipeline Bypassing

Flits bypass allocation stage
if router resources free



Router Pipeline

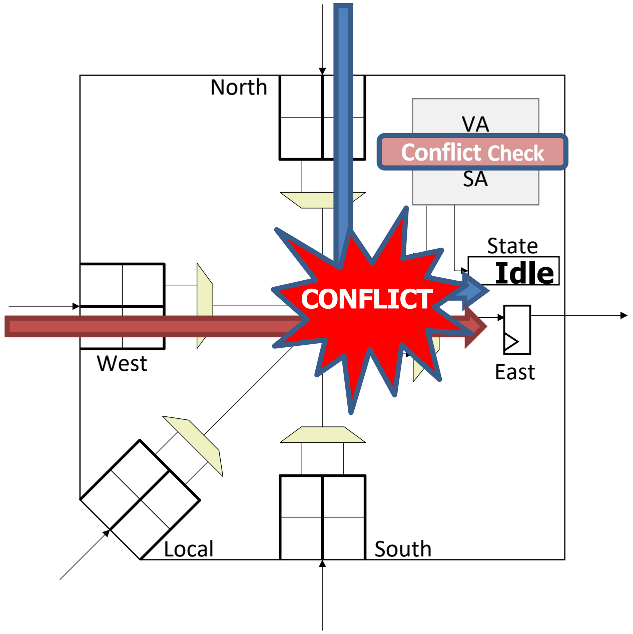
NRC VA SA	ST	LT

DAT093, 2018, Lecture on NoCs

Pipeline Bypassing: Conflict Check

Conflict Check logic needed to deal with concurrent incoming flits

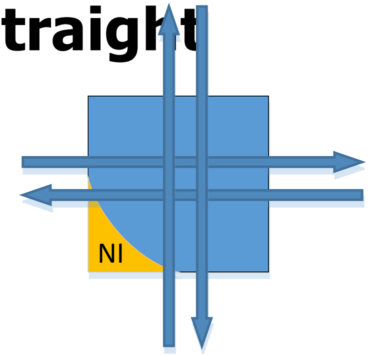
FreewayNoC <span style="color:red">cannot</span> afford the delay of Conflict Check logic:

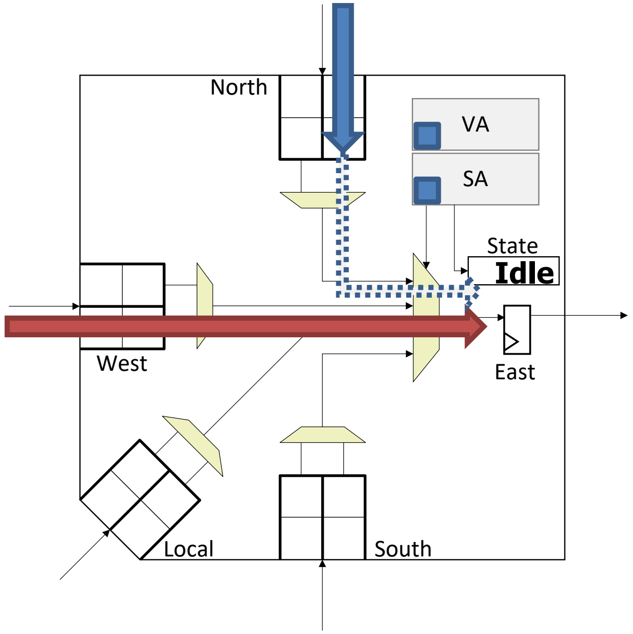


HighwayNoC: Simplified Pipeline Bypassing

Solution:

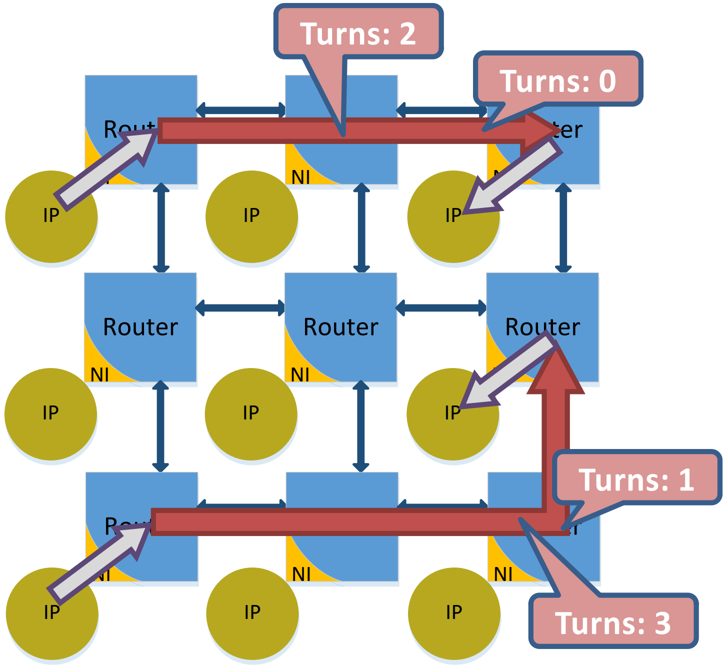
Allow bypassing only when flits go straight





HighwayNoC Simplified Pipeline Bypassing: Performance Implications

- **Slower turns**
- **Fixed overhead**
- **Independent of hop count**



The DDRNoC vs. ShortPath

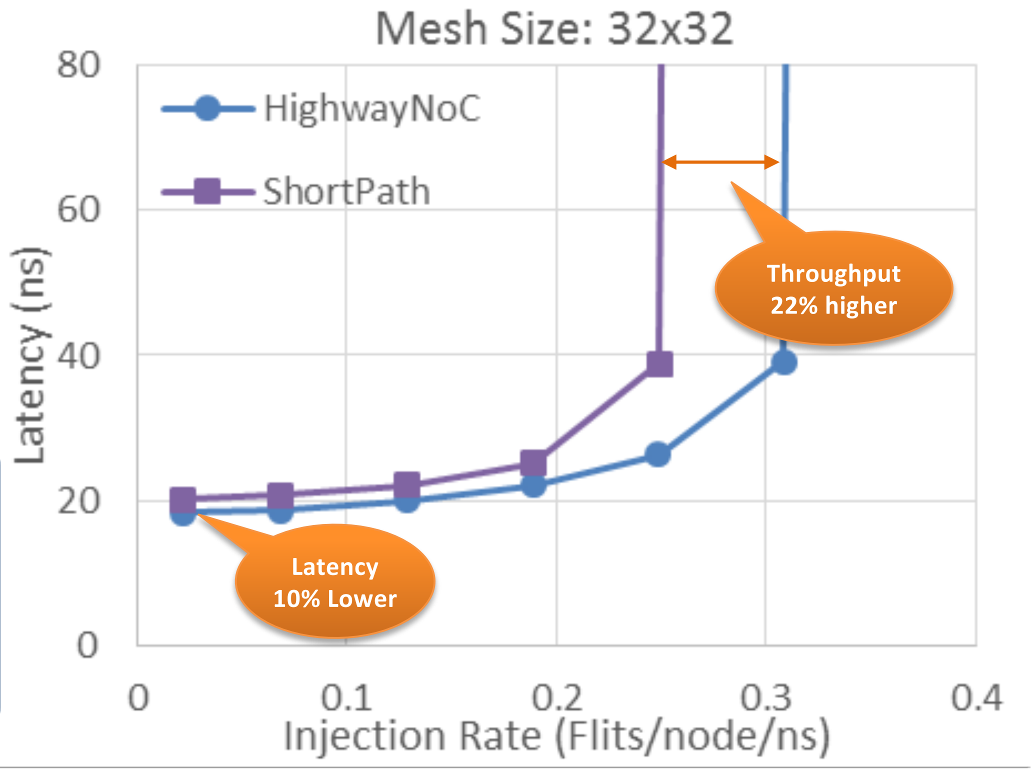
Clock Frequency:

- ShortPath: 2.38 GHz
- HighwayNoC: 1.47 GHz

Traffic Pattern: Uniform Random

HighwayNoC vs ShortPath

Throughput: 22-25% higher
Latency: up to 9% lower

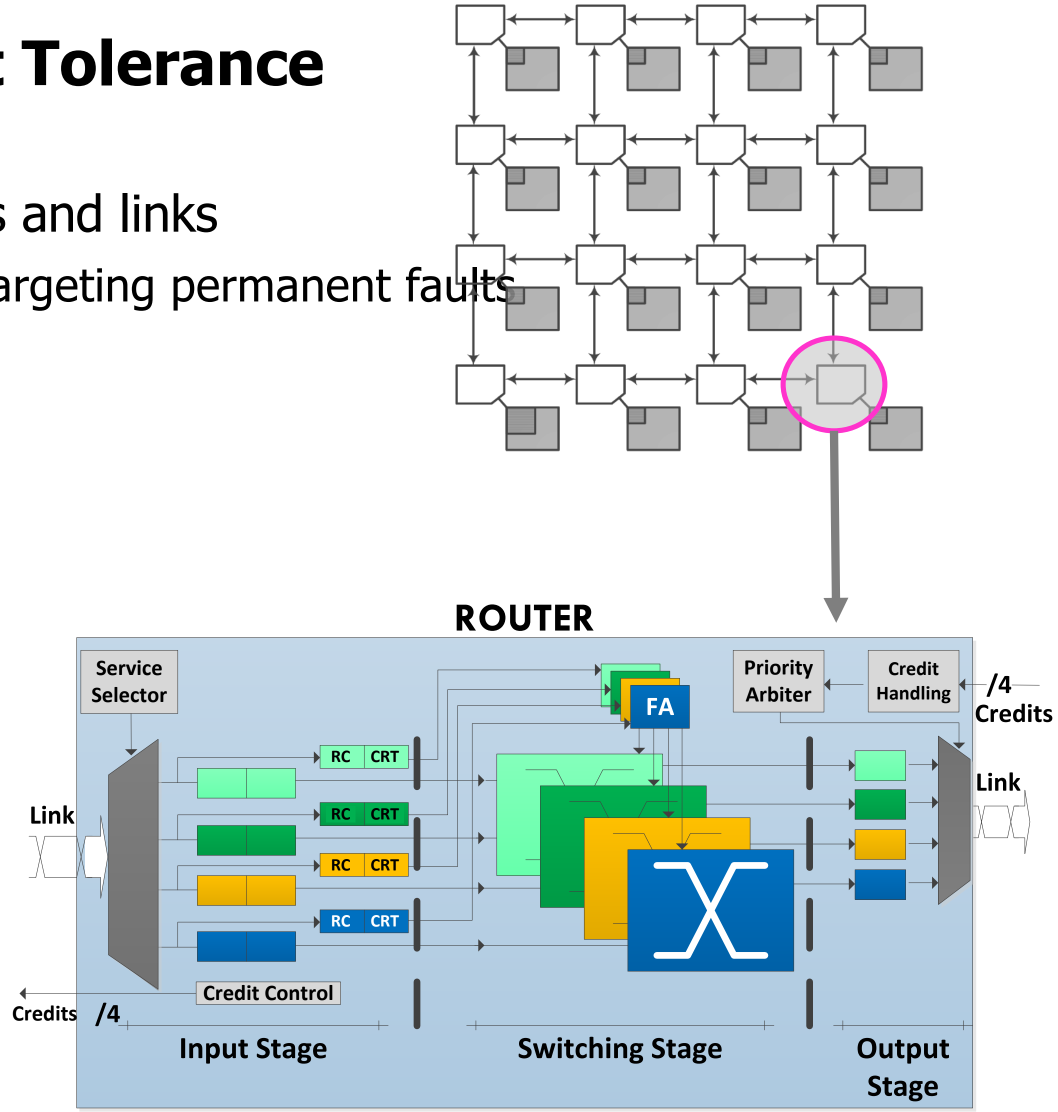


A. Psarras et al., "ShortPath: A Network-on-Chip Router with Fine-Grained Pipeline Bypassing", in *IEEE Transactions on Computers*, 2016

Background: NoC

- ❑ **Microarchitectural Fault Tolerance technique**
    - ❑ Tolerating faults at routers and links
        - We are explicitly targeting permanent faults

- ❑ **Service-Oriented NoC**
    - Supporting multiple traffic classes requirements, e.g. latency and throughout.

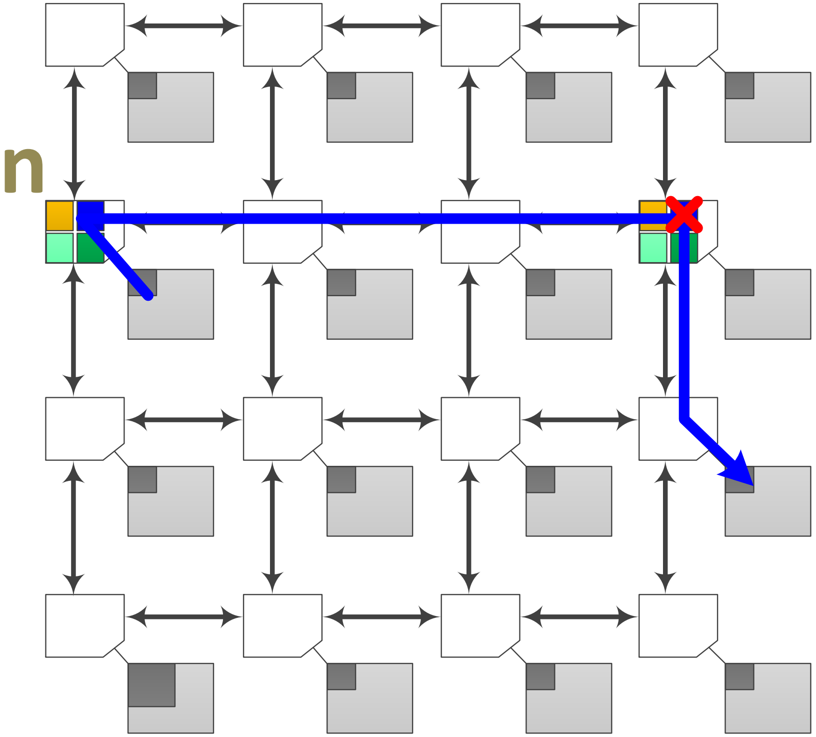


RQNoC: A Resilient Service-Oriented NoC

The Core Idea:

Allowing service redirection

(In presence of a faulty resource on the path)



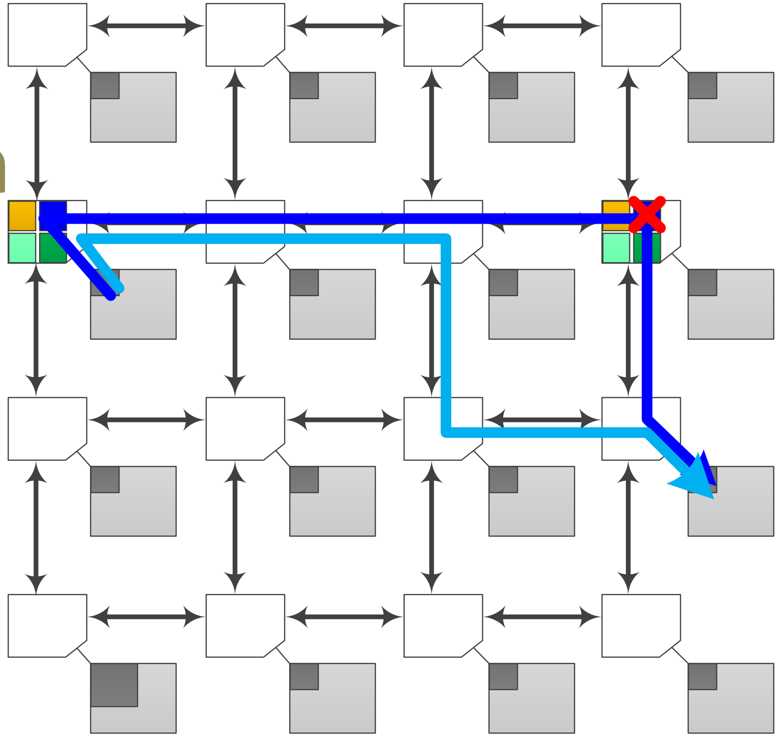
RQNoC: A Resilient Service-Oriented NoC

The Core Idea:

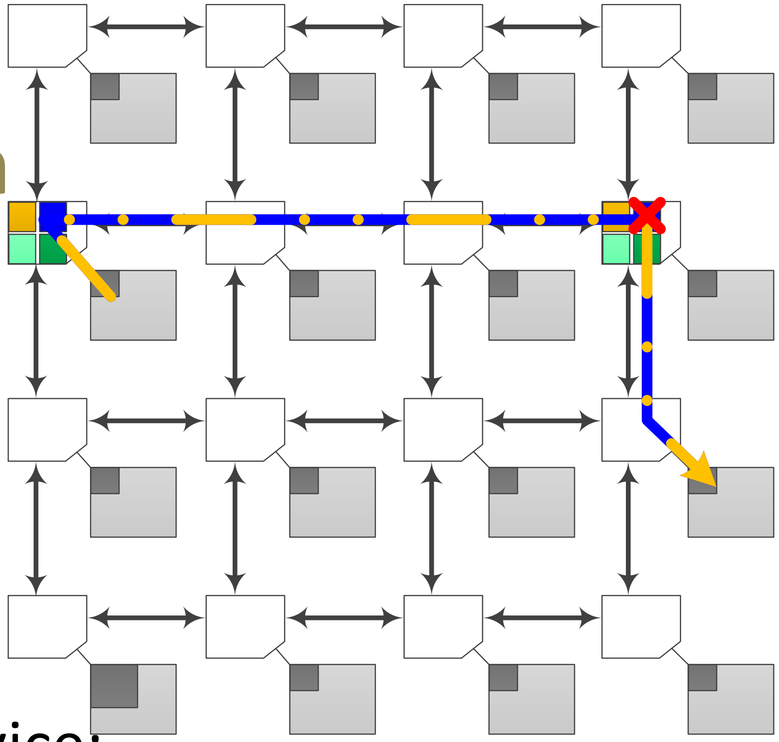
Allowing service redirection

(In presence of a faulty resource on the path)

- ❑ Through alternative path on the same service:
  - **Service Detour (SDetour)**
    - – **Longer alternative path**
    - + **Maintaining service isolation**



RQNoC: A Resilient Service-Oriented NoC



The Core Idea:

Allowing service redirection

(In presence of a faulty resource on the path)

- ❑ Through alternative path on the same service:
  - **Service Detour (SDetour)**
    - – Longer alternative path
    - + Maintaining service isolation
- ❑ Through resources of another service:
  - **Service Merge (SMerge)**
    - + Shorter path
    - – Breaching service isolation

Summary of Part 2

A.Malek et.al., TECS'16

- □ **Objective:**
  To design and evaluate a service-oriented NoC that enables us to ***trade service isolation for fault tolerance***.
- □ RQNoC supports two alternatives for service redirection:
  - □ SDetour: Use alternative resources on the same service
  - □ SMerge: Share resources with another service

- SDetour
  - Requires 9% more resources vs. Baseline
  - Latency increased up to 24% and throughput up to 50% reduced
  - Maintains 41% connectivity in presence of 32 fault
- SMerge
  - Requires 22.4% more resources vs. Baseline
  - Latency increased up to 3.8x and throughput up to 70% reduced
  - Maintains 90% connectivity in presence of 32 faults

Sharing resources between traffic classes imposes considerable latency and throughput penalty but improves the network connectivity to a very high degree

Summary of Lecture

- NoCs basics
- NoCs design alternatives:
  - Topologies
  - Flow control
  - Routing
  - Router architecture
  - Packet scheduling
- Research on NoCs

Reading:

- *Principles and Practices of Interconnection Networks, Book by Bill Dally and Brian Towles*