# ECE/CS 757: Advanced Computer Architecture II

Instructor:Mikko H Lipasti

Spring 2015

University of Wisconsin-Madison

Lecture notes based on slides created by John Shen, Mark Hill, David Wood, Guri Sohi, Jim Smith, Natalie Enright Jerger, Michel Dubois, Murali Annavaram, Per Stenström and probably others

# Lecture Outline

- Introduction to Networks
- Network Topologies
- Network Routing
- Network Flow Control
- Router Microarchitecture
- Technology example: On-chip Nanophotonics

# Introduction

- How to connect individual devices into a group of communicating devices?
  - A device can be:
    - Component within a chip
    - Component within a computer
    - Computer
    - System of computers
  - Network consists of:
    - End point devices with interface to network
    - Links
    - Interconnect hardware
- Goal: transfer maximum amount of information with the least cost (minimum time, power)

# Types of Interconnection Networks

- Interconnection networks can be grouped into four domains
  - Depending on number and proximity of devices to be connected
- On-Chip networks (OCNs or NoCs)
  - Devices include microarchitectural elements (functional units, register files), caches, directories, processors
  - Current designs: small number of devices
    - Ex: IBM Cell, Sun's Niagara
  - Projected systems: dozens, hundreds of devices
    - Ex: Intel Teraflops research prototypes, 80 cores
  - Proximity: millimeters

# Types of Interconnection Networks (2)

- System/Storage Area Network (SANs)
  - Multiprocessor and multicomputer systems
    - Interprocessor and processor-memory interconnections
  - Server and data center environments
    - Storage and I/O components
  - Hundreds to thousands of devices interconnected
    - IBM Blue Gene/L supercomputer (64K nodes, each with 2 processors)
  - Maximum interconnect distance typically on the order of tens of meters, but some with as high as a few hundred meters
    - InfiniBand: 120 Gbps over a distance of 300 m
  - Examples (standards and proprietary)
    - InfiniBand, Myrinet, Quadrics, Advanced Switching Interconnect

# Types of Interconnection Networks (3)

- Local Area Network (LANs)
  - Interconnect autonomous computer systems
  - Machine room or throughout a building or campus
  - Hundreds of devices interconnected (1,000s with bridging)
  - Maximum interconnect distance on the order of few kilometers, but some with distance spans of a few tens of kilometers
  - Example (most popular): 1-10Gbit Ethernet

# Types of Interconnection Networks (4)

- Wide Area Networks (WANs)
  - Interconnect systems distributed across the globe
  - Internetworking support is required
  - Many millions of devices interconnected
  - Maximum interconnect distance of many thousands of kilometers
  - Example: ATM

# Organization

- Here we focus on On-chip networks
- Concepts applicable to all types of networks
  - Focus on trade-offs and constraints as applicable to NoCs

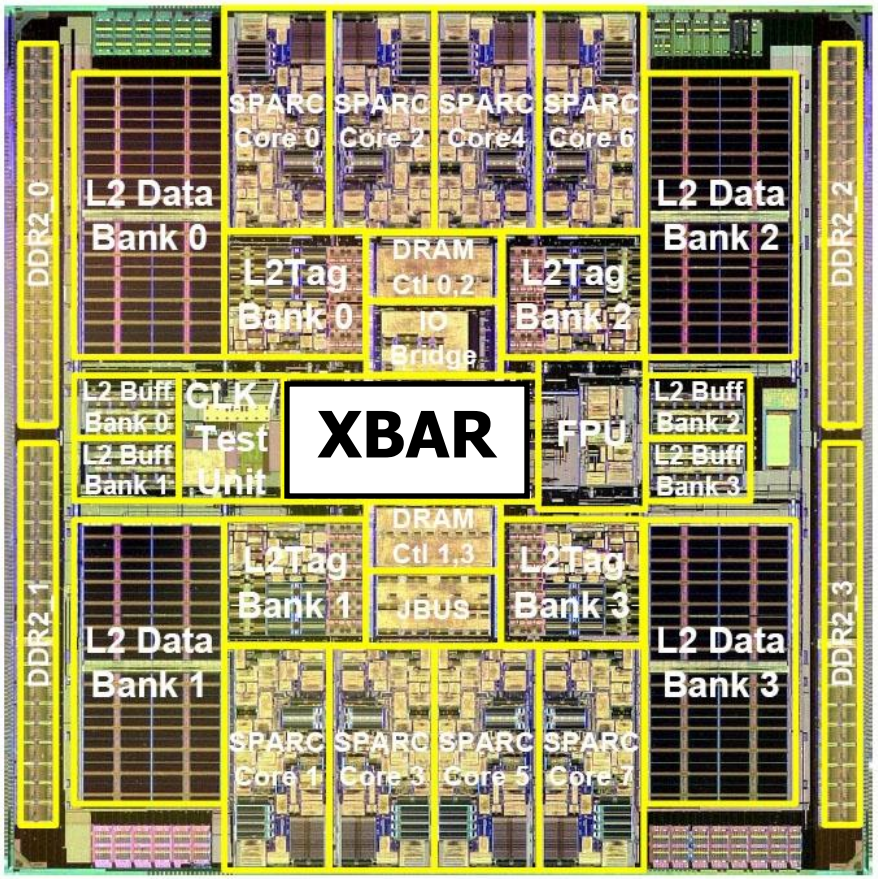
# On-Chip Networks (NoCs)

- Why Network on Chip?
  - Ad-hoc wiring does not scale beyond a small number of cores
    - Prohibitive area
    - Long latency
- OCN offers
  - scalability
  - efficient multiplexing of communication
  - often modular in nature (ease verification)

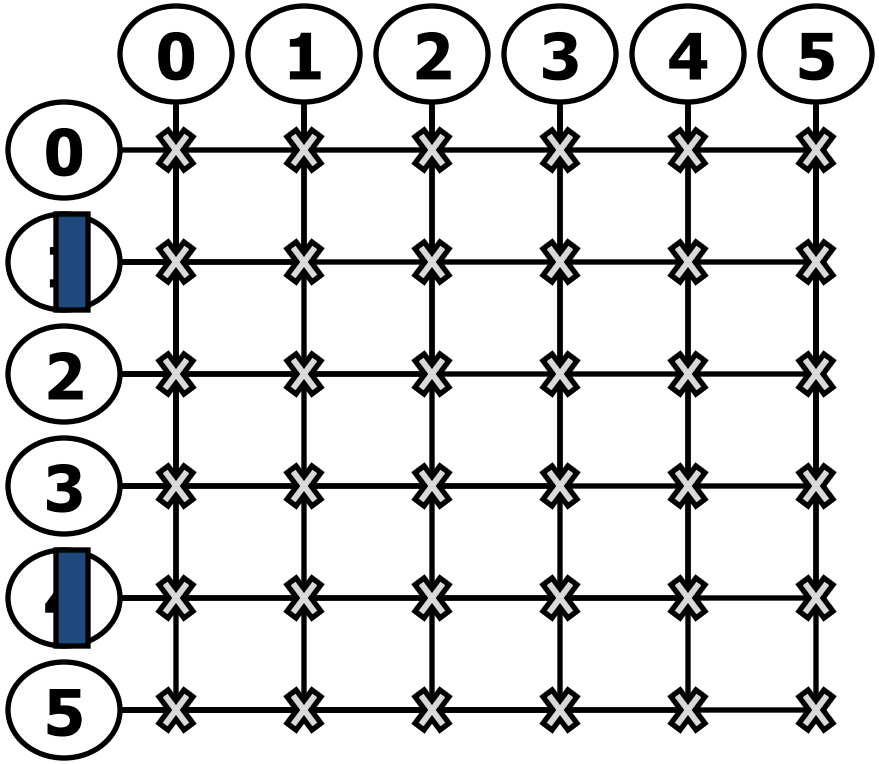
# Differences between on-chip and off-chip networks

- Off-chip: I/O bottlenecks
  - Pin-limited bandwidth
  - Inherent overheads of off-chip I/O transmission
- On-chip
  - Tight area and power budgets
  - Ultra-low on-chip latencies

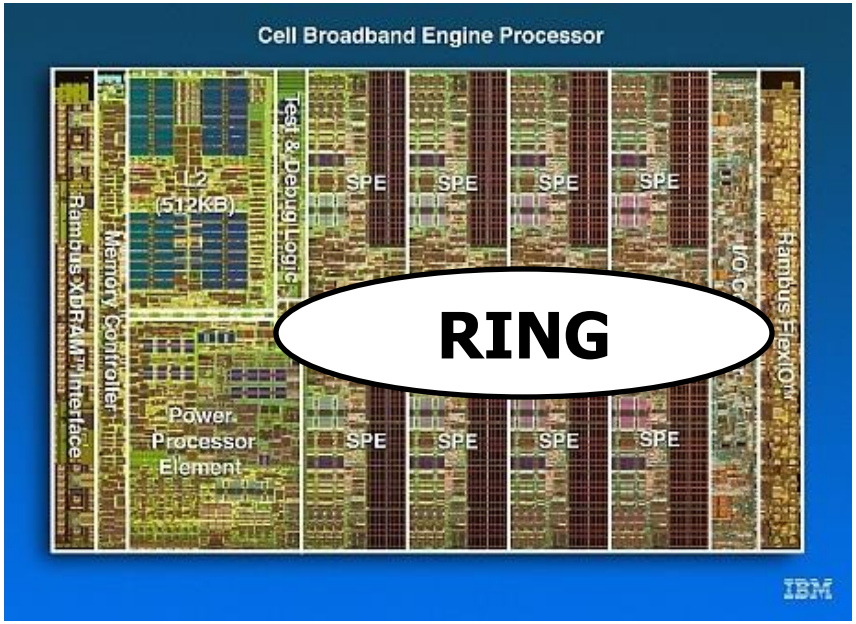
# MulticoreExamples (1)




Sun Niagara



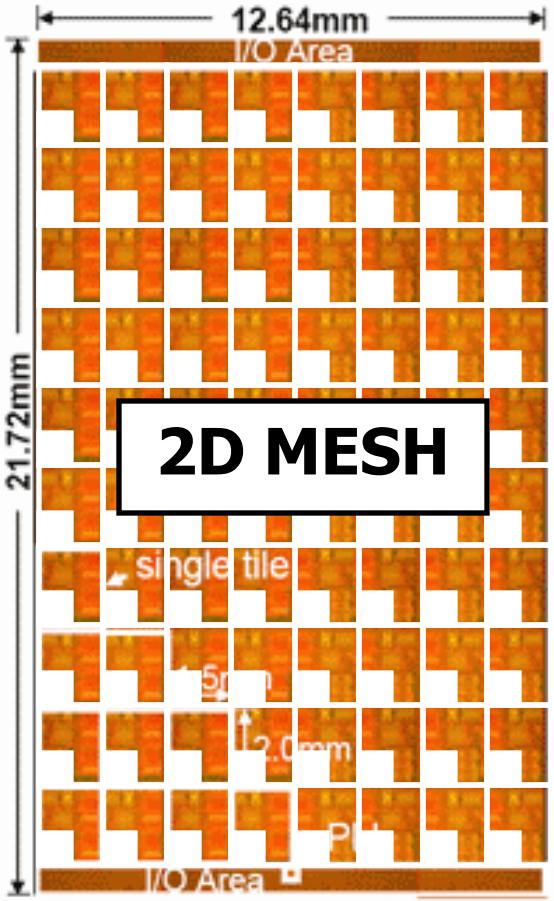
# Multicore Examples (2)

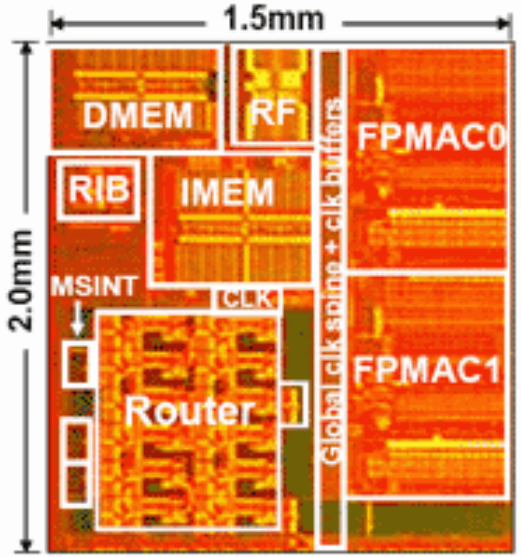



IBM Cell

- Element Interconnect Bus
  - 4 rings
  - Packet size: 16B-128B
  - Credit-based flow control
  - Up to 64 outstanding requests
  - Latency: 1 cycle/hop

# Many Core Example




| | |
|---|---|
| Technology | 65nm CMOS Process |
| Interconnect | 1 poly, 8 metal (Cu) |
| Transistors | 100 Million |
| Die Area | 275mm² |
| Tile area | 3mm² |
| Package | 1248 pin LGA, 14 layers, 343 signal pins |

- Intel Polaris
  - 80 core prototype
- Academic Research ex:
  - MIT Raw, TRIPs
    - 2-D Mesh Topology
    - Scalar Operand Networks

# References


- William Dally and Brian Towles. Principles and Practices of Interconnection Networks. Morgan Kaufmann Pub., San Francisco, CA, 2003.
- William Dally and Brian Towles, “Route packets not wires: On-chip interconnection networks,” in Proceedings of the 38th Annual Design Automation Conference (DAC-38), 2001, pp. 684–689.
- David Wentzlaff, Patrick Griffin, Henry Hoffman, LieweiBao, Bruce Edwards, Carl Ramey, Matthew Mattina, Chi-Chang Miao, John Brown I I I, and AnantAgarwal. On-chip interconnection architecture of the tile processor. IEEE Micro, pages 15–31, 2007.
- Michael Bedford Taylor, Walter Lee, SamanAmarasinghe, and AnantAgarwal. Scalar operand networks: On-chip interconnect for ILP in partitioned architectures. In Proceedings of the International Symposium on High Performance Computer Architecture, February 2003.
- S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote, and N. Borkar. An 80-tile 1.28tflops network-on-chip in 65nm cmos. *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International, pages 98–589, Feb. 2007.*

# Lecture Outline

- Introduction to Networks
- Network Topologies
- Network Routing
- Network Flow Control
- Router Microarchitecture
- Technology example: On-chip Nanophotonics

# Topology Overview

- Definition: determines arrangement of channels and nodes in network
- Analogous to road map
- Often first step in network design
- Routing and flow control build on properties of topology

# Abstract Metrics

- Use metrics to evaluate performance and cost of topology
- Also influenced by routing/flow control
  - At this stage
    - Assume ideal routing (perfect load balancing)
    - Assume ideal flow control (no idle cycles on any channel)
- Switch Degree: number of links at a node
  - Proxy for estimating cost
    - Higher degree requires more links and port counts at each router

# Latency

- Time for packet to traverse network
  - Start: head arrives at input port
  - End: tail departs output port
- Latency = Head latency + serialization latency
  - Serialization latency: time for packet with Length L to cross channel with bandwidth b (L/b)
- Hop Count: the number of links traversed between source and destination
  - Proxy for network latency
  - Per hop latency with zero load

# Impact of Topology on Latency

- Impacts average minimum hop count
- Impact average distance between routers
- Bandwidth

# Throughput

- Data rate (bits/sec) that the network accepts per input port
- Max throughput occurs when one channel saturates
  - Network cannot accept any more traffic
- Channel Load
  - Amount of traffic through channel c if each input node injects 1 packet in the network

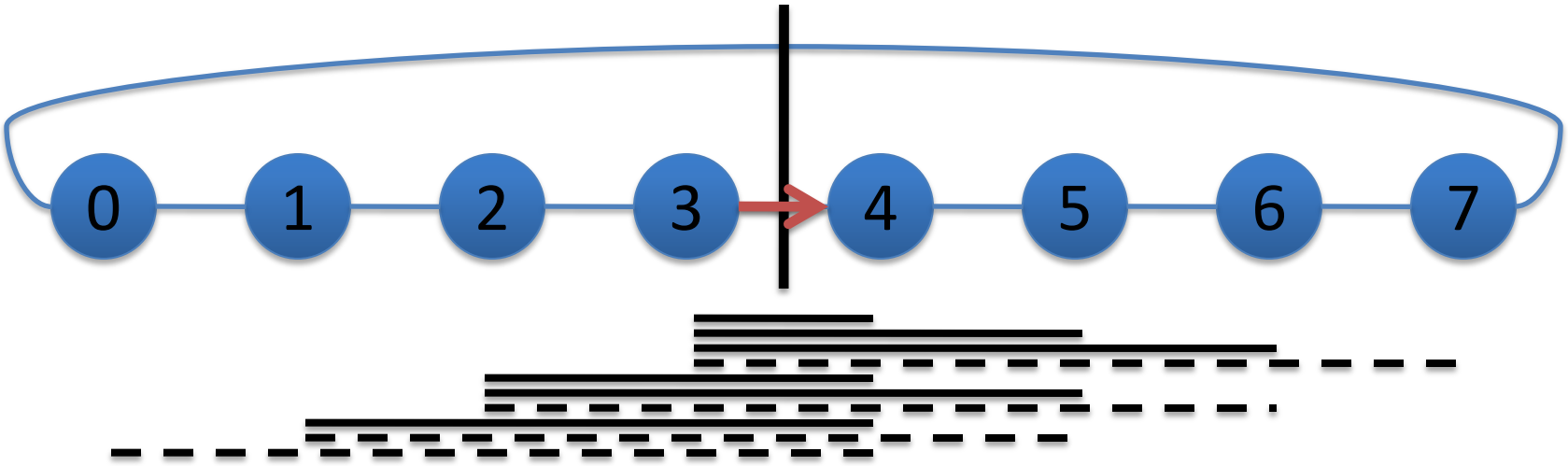
# Maximum channel load

- Channel with largest fraction of traffic
- Max throughput for network occurs when channel saturates
  - Bottleneck channel

# Bisection Bandwidth

- Cuts partition all the nodes into two disjoint sets
  - Bandwidth of a cut
- Bisection
  - A cut which divides all nodes into nearly half
  - Channel bisection → min. channel count over all bisections
  - Bisection bandwidth →min. bandwidth over all bisections
- With uniform traffic
  - ½ of traffic cross bisection

# Throughput Example




- Bisection = 4 (2 in each direction)
- With uniform random traffic
  - 3 sends 1/8 of its traffic to 4,5,6
  - 3 sends 1/16 of its traffic to 7 (2 possible shortest paths)
  - 2 sends 1/8 of its traffic to 4,5
  - Etc
- Channel load = 1

# Path Diversity

- Multiple minimum length paths between source and destination pair
- Fault tolerance
- Better load balancing in network
- Routing algorithm should be able to exploit path diversity
- We'll see shortly
  - Butterfly has no path diversity
  - Torus can exploit path diversity

# Path Diversity (2)

- Edge disjoint paths: no links in common
- Node disjoint paths:  no nodes in common except source and destination
- If j = minimum number of edge/node disjoint paths between any source-destination pair
  - Network can tolerate j link/node failures

- Path diversity does not provide pt-to-pt order
  - Implications on coherence protocol design!

# Symmetry

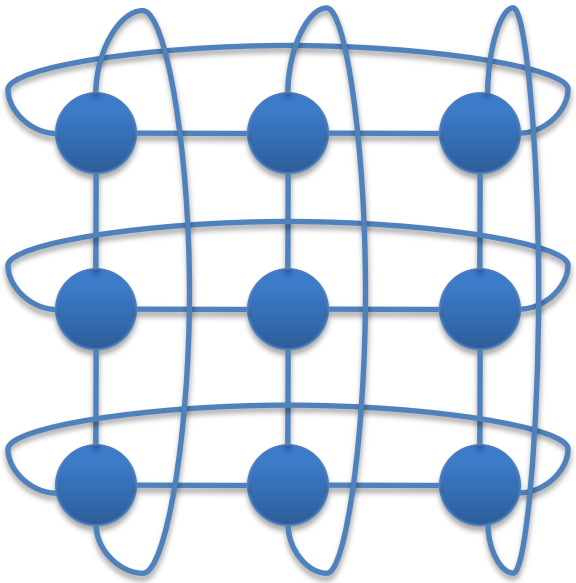
- Vertex symmetric:
  - An automorphism exists that maps any node a onto another node b
  - Topology same from point of view of all nodes
- Edge symmetric:
  - An automorphism exists that maps any channel a onto another channel b

# Direct & Indirect Networks

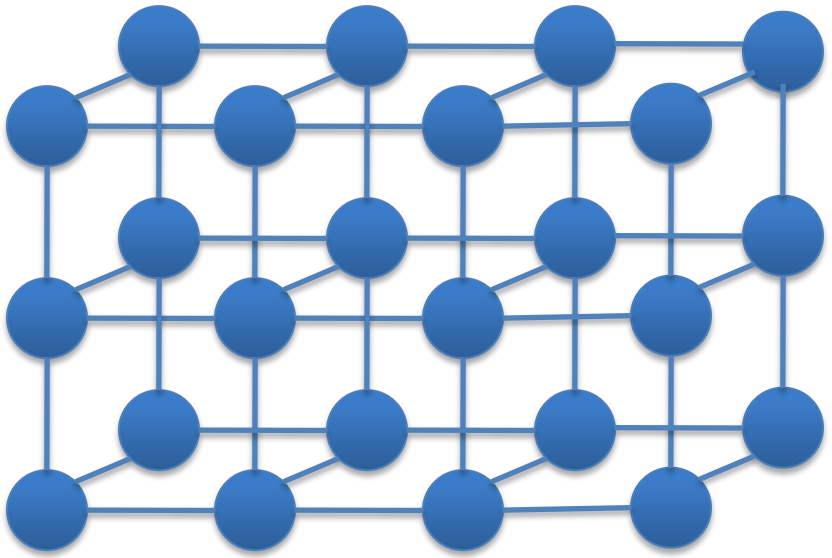
- Direct: Every switch also network end point
  - Ex: Torus
- Indirect: Not all switches are end points
  - Ex: Butterfly

# Torus (1)

- K-ary n-cube: $k^n$ network nodes
- n-dimensional grid with k nodes in each dimension



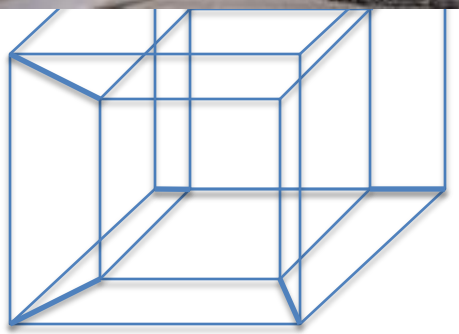
3-ary 2-cube



2,3,4-ary 3-mesh

# Torus (2)





- Topologies in Torus Family
  - Ring k-ary 1-cube
  - Hypercubes 2-ary n-cube
- Edge Symmetric
  - Good for load balancing
  - Removing wrap-around links for mesh loses edge symmetry
    - More traffic concentrated on center channels
- Good path diversity
- Exploit locality for near-neighbor traffic

# Torus (3)

$$H_{\min} = \begin{cases} \frac{nk}{4} & k \text{ even} \\ n\left(\frac{k}{4} - \frac{1}{4k}\right) & k \text{ odd} \end{cases}$$

- Hop Count:
- Degree = 2n, 2 channels per dimension

# Channel Load for Torus

- Even number of k-ary (n-1)-cubes in outer dimension
- Dividing these k-ary (n-1)-cubes gives 2 sets of $k^{n-1}$ bidirectional channels or $4k^{n-1}$
- ½ Traffic from each node cross bisection

$$channel\,load = \frac{N}{2} \times \frac{k}{4N} = \frac{k}{8}$$

- Mesh has ½ the bisection bandwidth of torus

# Torus Path Diversity

$$\left|R_{xy}\right| = \binom{\Delta x + \Delta y}{\Delta x}$$

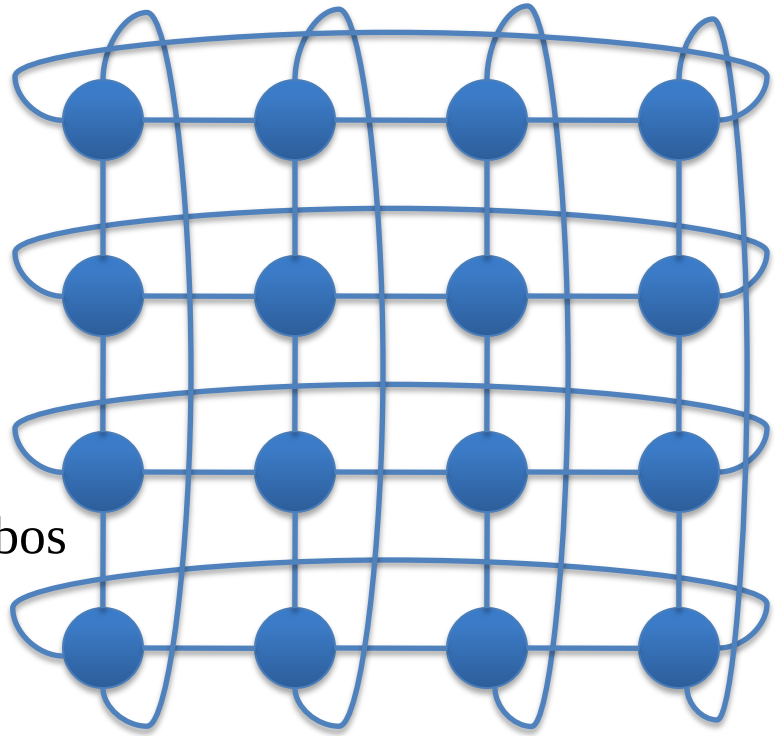
2 dimensions*

$$\Delta x = 2, \Delta y = 2$$

$$\left|R_{xy}\right| = 6$$

$$\left|R_{xy}\right| = 24$$ NW, NE, SW, SE combos

$$\left|R_{xy}\right| = \prod_{i=0}^{n-1} \binom{\sum_{j=i}^{n-1} \Delta j}{\Delta i} = \frac{\left(\sum_{i=0}^{n-1} \Delta i\right)!}{\prod_{i=0}^{n-1} \Delta i!}$$



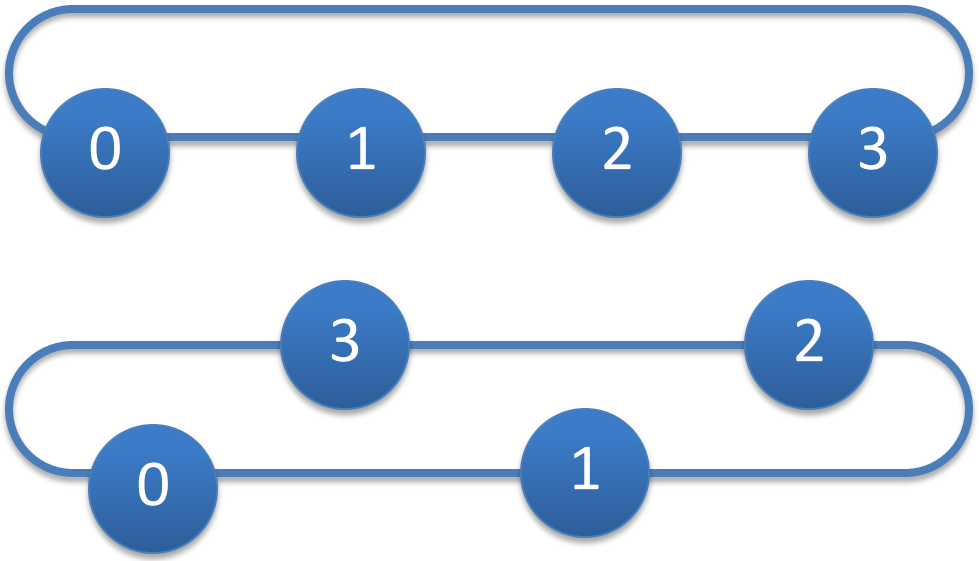
2 edge and node disjoint minimum paths

n dimensions with $\Delta$i hops in i dimension

*assume single direction for x and y

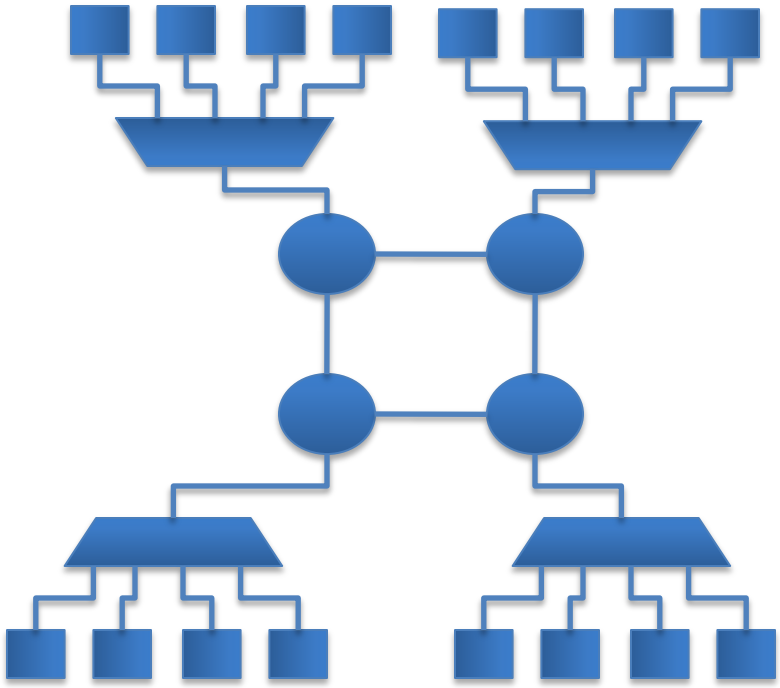
# Implementation

- Folding
  - Equalize path lengths
    - Reduces max link length
    - Increases length of other links



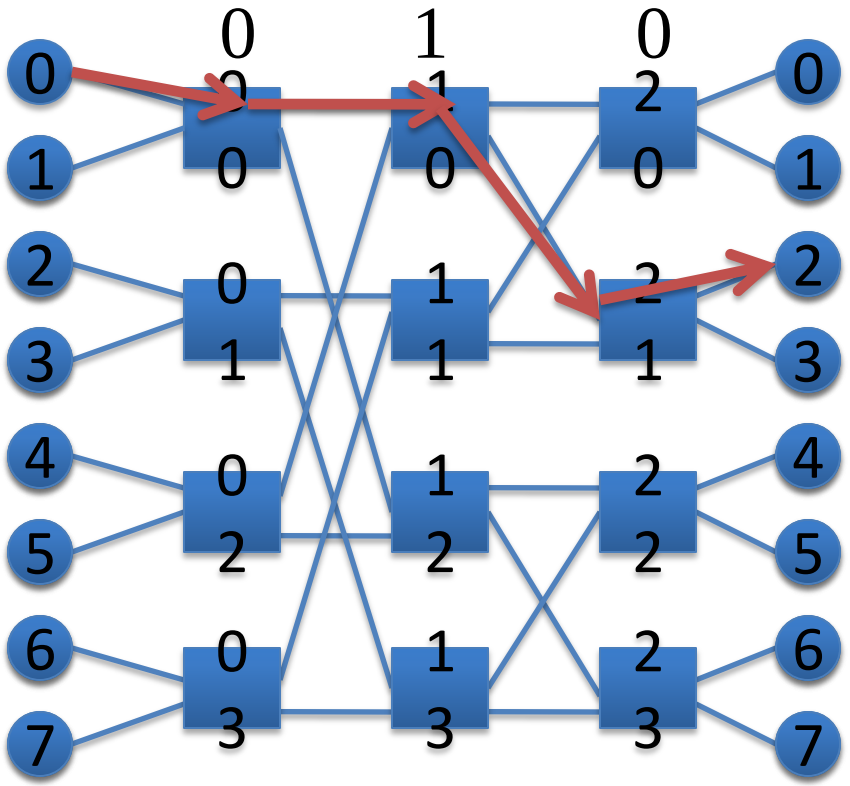
# Concentration

- Don’t need 1:1 ratio of network nodes and cores/memory
- Ex: 4 cores concentrated to 1 router



# Butterfly

- K-ary n-fly: $k^n$ network nodes
- Example: 2-ary 3-fly
- Routing from 000 to 010
  - Dest address used to directly route packet
  - Bit n used to select output port at stage n



# Butterfly (2)

- No path diversity $\left|R_{xy}\right| = 1$
- Hop Count
  - $\text{Log}_k n + 1$
  - Does not exploit locality
    - Hop count same regardless of location
- Switch Degree = 2k
- Channel Load → uniform traffic

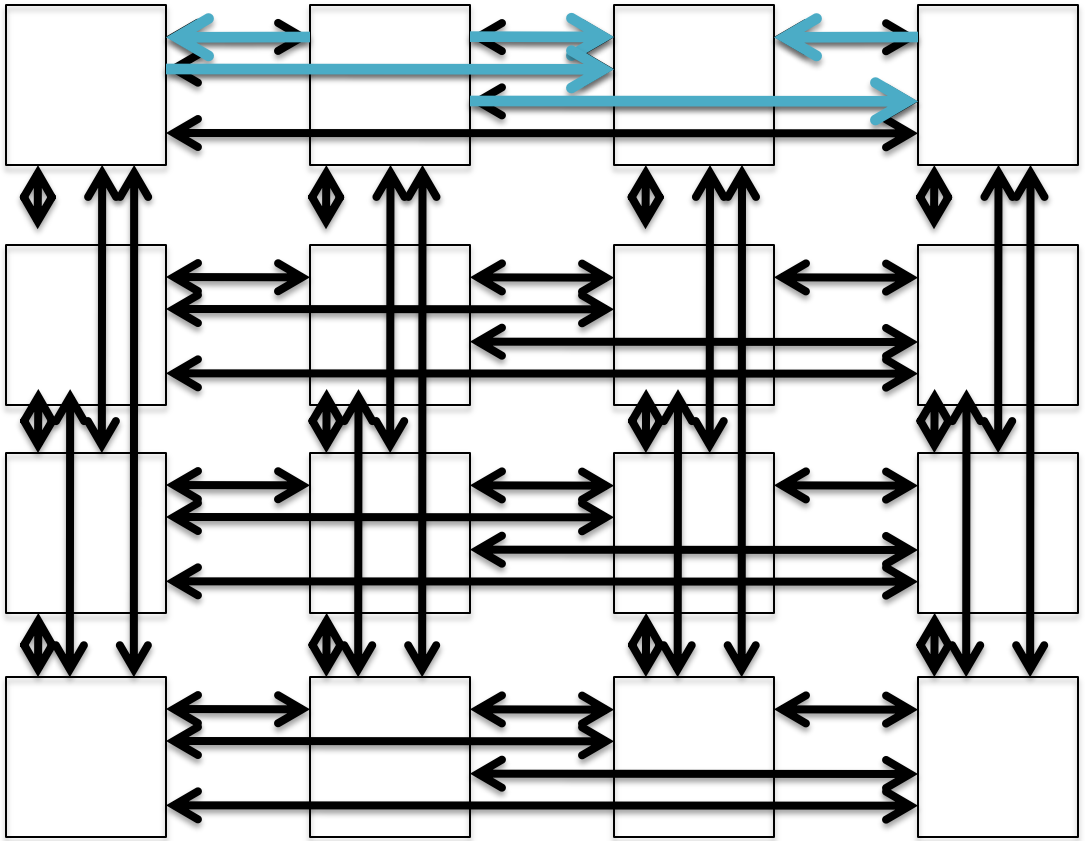
$$\frac{NH_{\min}}{C} = \frac{k^n(n+1)}{k^n(n+1)} = 1$$

  - Increases for adversarial traffic

# Flattened Butterfly

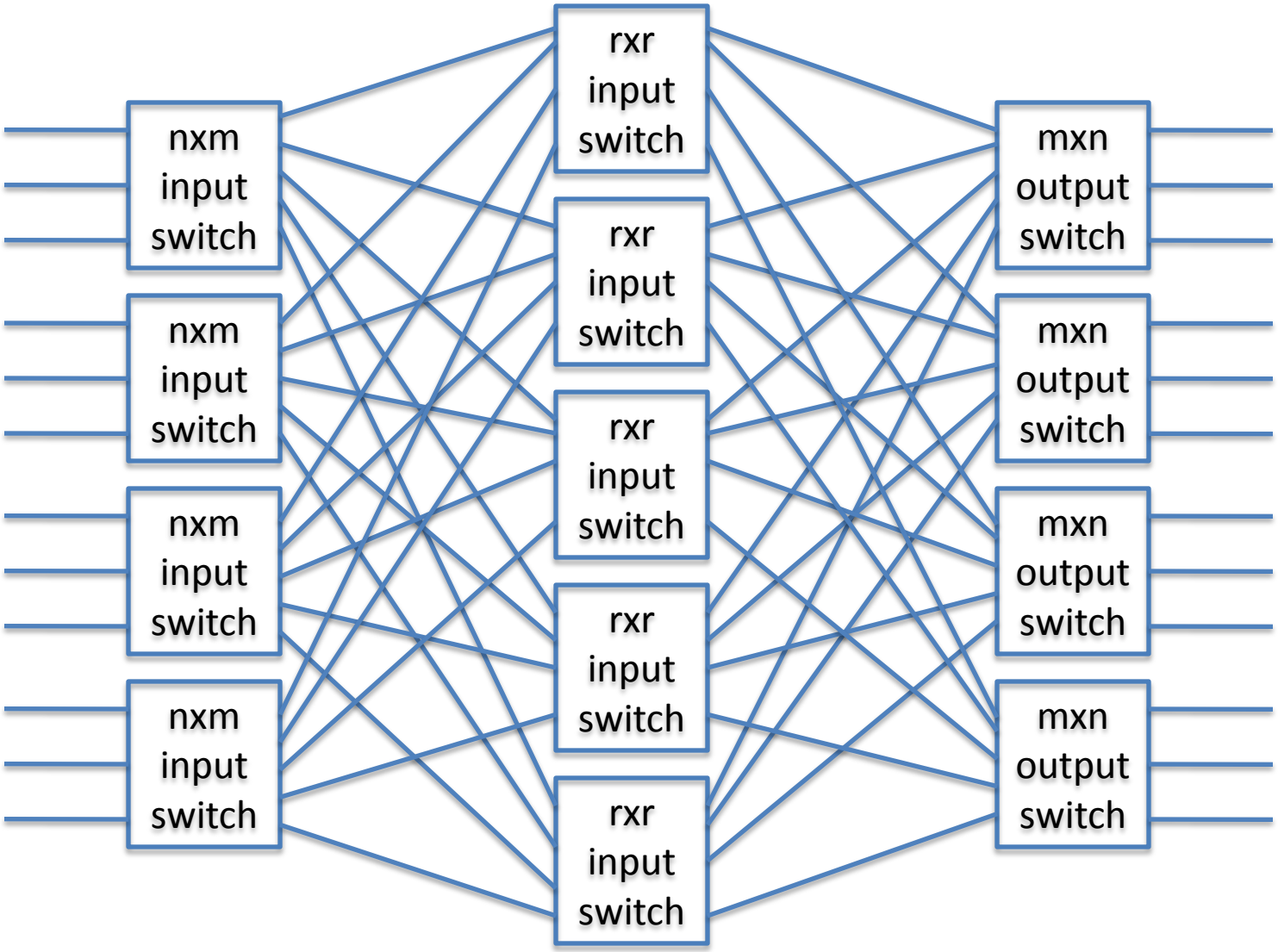
- Proposed by Kim et al (ISCA 2007)
  - Adapted for on-chip (MICRO 2007)
- Advantages
  - Max distance between nodes = 2 hops
  - Lower latency and improved throughput compared to mesh
- Disadvantages
  - Requires higher port count on switches (than mesh, torus)
  - Long global wires
  - Need non-minimal routing to balance load

# Flattened Butterfly



- Path diversity through non-minimal routes

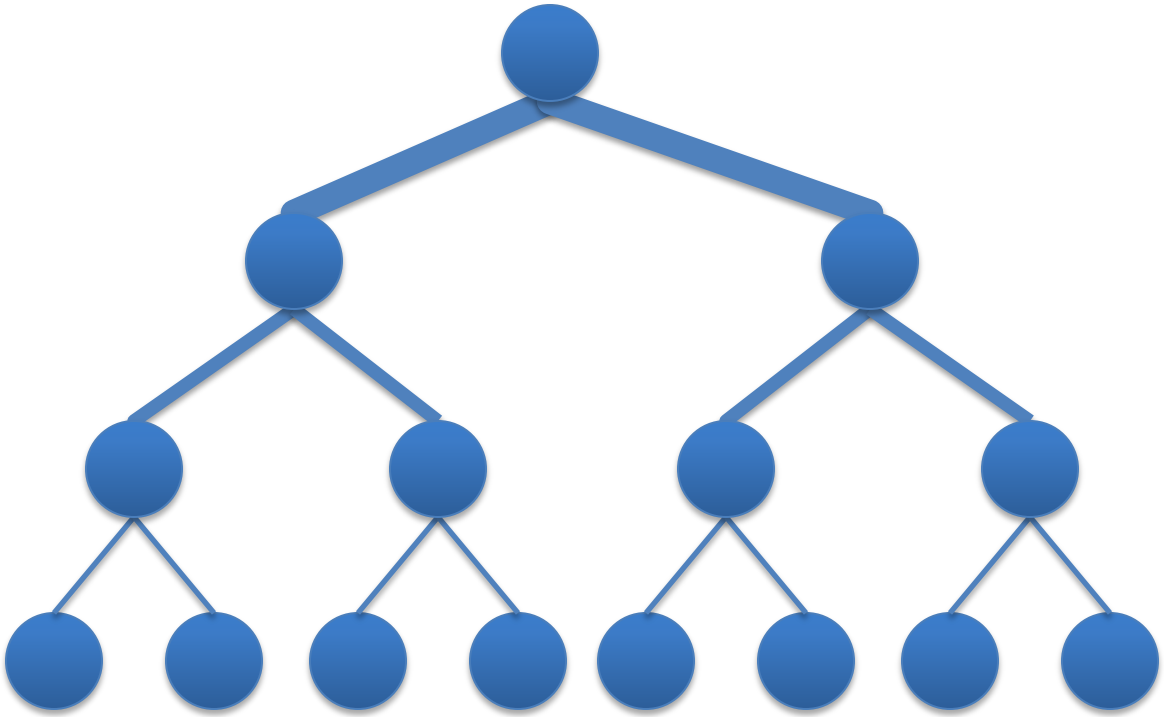
# Clos Network



# Clos Network

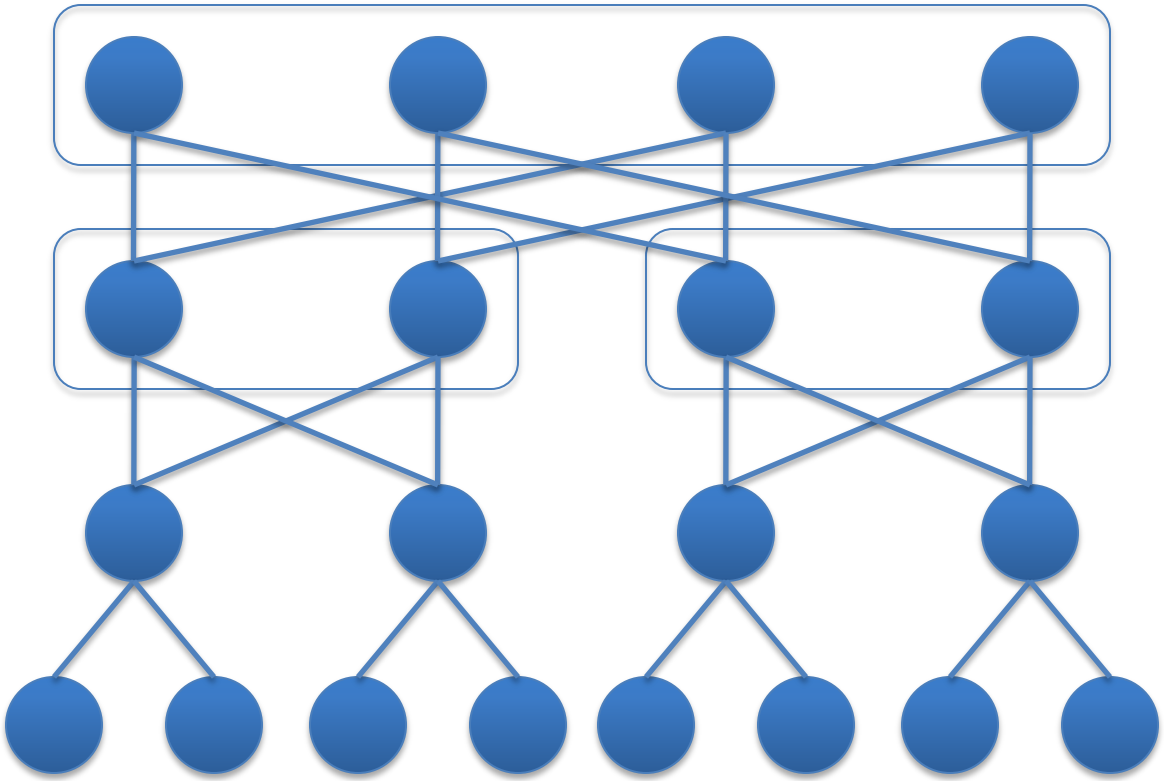
- 3-stage indirect network
- Characterized by triple (m, n, r)
  - M: # of middle stage switches
  - N: # of input/output ports on input/output switches
  - R: # of input/output switching
- Hop Count = 4

# Folded Clos (Fat Tree)



- Bandwidth remains constant at each level
- Regular Tree: Bandwidth decreases closer to root

# Fat Tree (2)



- Provides path diversity

# Common On-Chip Topologies

- Torus family: mesh, concentrated mesh, ring
  - Extending to 3D stacked architectures
  - Favored for low port count switches
- Butterfly family: Flattened butterfly

# Topology Summary

- First network design decision
- Critical impact on network latency and throughput
  - Hop count provides first order approximation of message latency
  - Bottleneck channels determine saturation throughput

# Lecture Outline

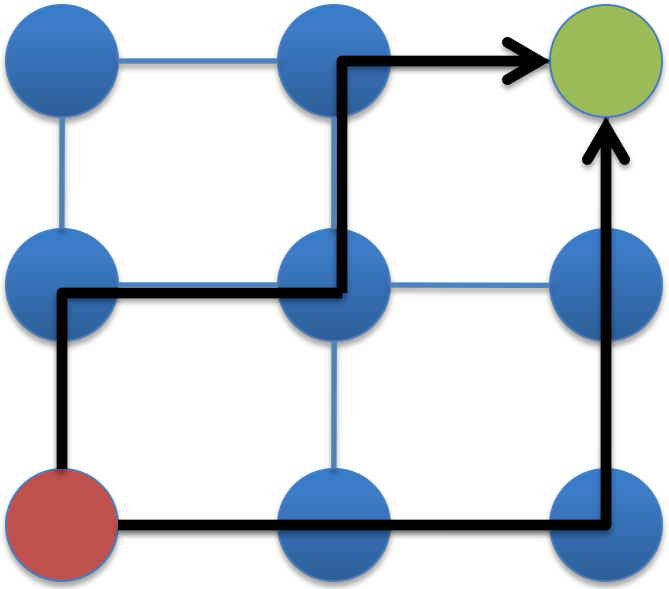
- Introduction to Networks
- Network Topologies
- Network Routing
- Network Flow Control
- Router Microarchitecture
- Technology example: On-chip Nanophotonics

# Routing Overview

- Discussion of topologies assumed ideal routing
- Practically though routing algorithms are not ideal
- Discuss various classes of routing algorithms
  - Deterministic, Oblivious, Adaptive
- Various implementation issues
  - Deadlock

# Routing Basics

- Once topology is fixed
- Routing algorithm determines path(s) from source to destination



# Routing Algorithm Attributes

- Number of destinations
  - Unicast, Multicast, Broadcast?
- Adaptivity
  - Oblivious or Adaptive? Local or Global knowledge?
- Implementation
  - Source or node routing?
  - Table or circuit?

# Oblivious

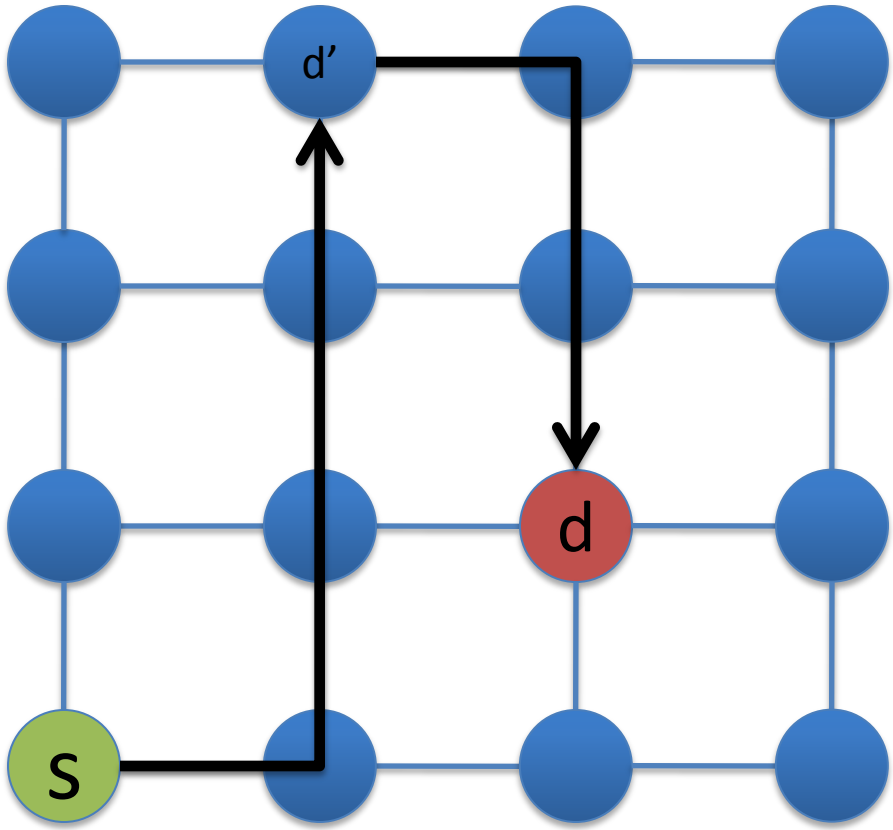
- Routing decisions are made without regard to network state
  - Keeps algorithms simple
  - Unable to adapt
- Deterministic algorithms are a subset of oblivious

# Deterministic

- All messages from Src to Dest will traverse the same path
- Common example: Dimension Order Routing (DOR)
  - Message traverses network dimension by dimension
  - Aka XY routing
- Cons:
  - Eliminates any path diversity provided by topology
  - Poor load balancing
- Pros:
  - Simple and inexpensive to implement
  - Deadlock free

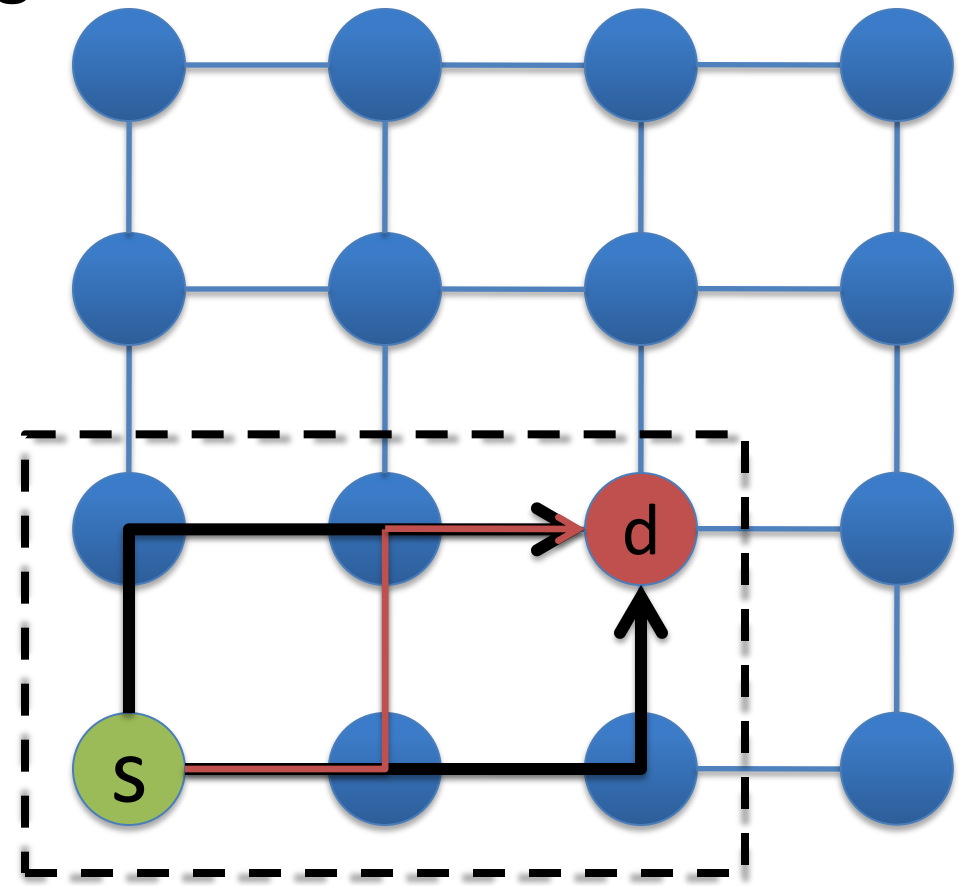
# Valiant's Routing Algorithm

- To route from s to d, randomly choose intermediate node d'
  - Route from s to d' and from d' to d.
- Randomizes any traffic pattern
  - All patterns appear to be uniform random
  - Balances network load
- Non-minimal



# Minimal Oblivious

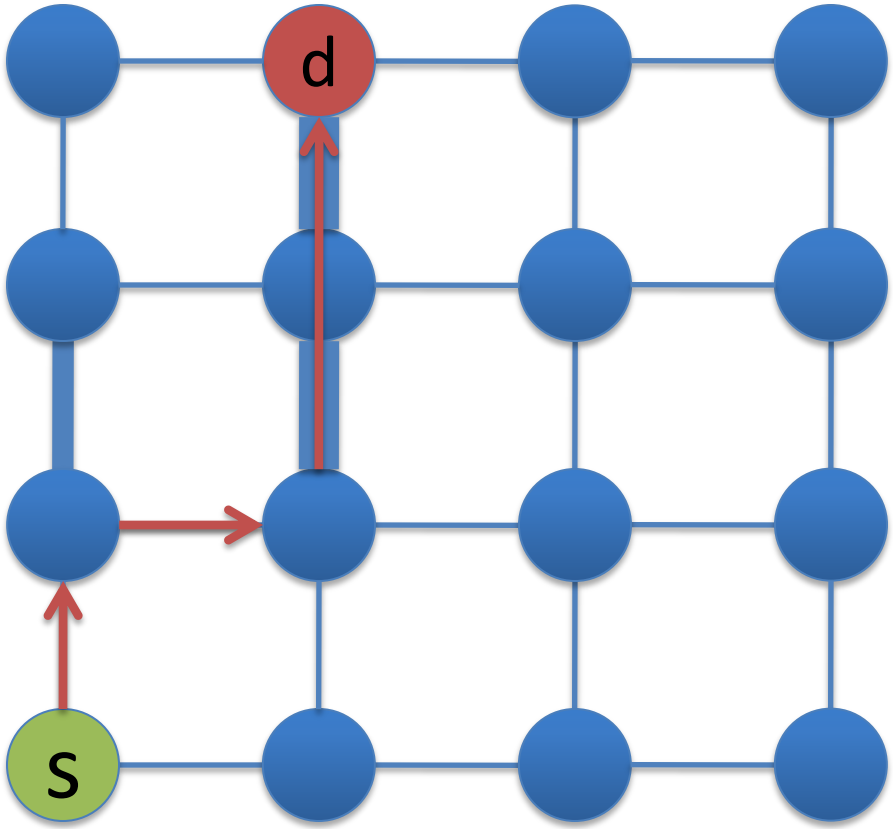
- Valiant's: Load balancing comes at expense of significant hop count increase
  - Destroys locality
- Minimal Oblivious: achieve some load balancing, but use shortest paths
  - d' must lie within minimum quadrant
  - 6 options for d'
  - Only 3 different paths



# Adaptive

- Uses network state to make routing decisions
  - Buffer occupancies often used
  - Couple with flow control mechanism
- Local information readily available
  - Global information more costly to obtain
  - Network state can change rapidly
  - Use of local information can lead to non-optimal choices
- Can be minimal or non-minimal

# Minimal Adaptive Routing

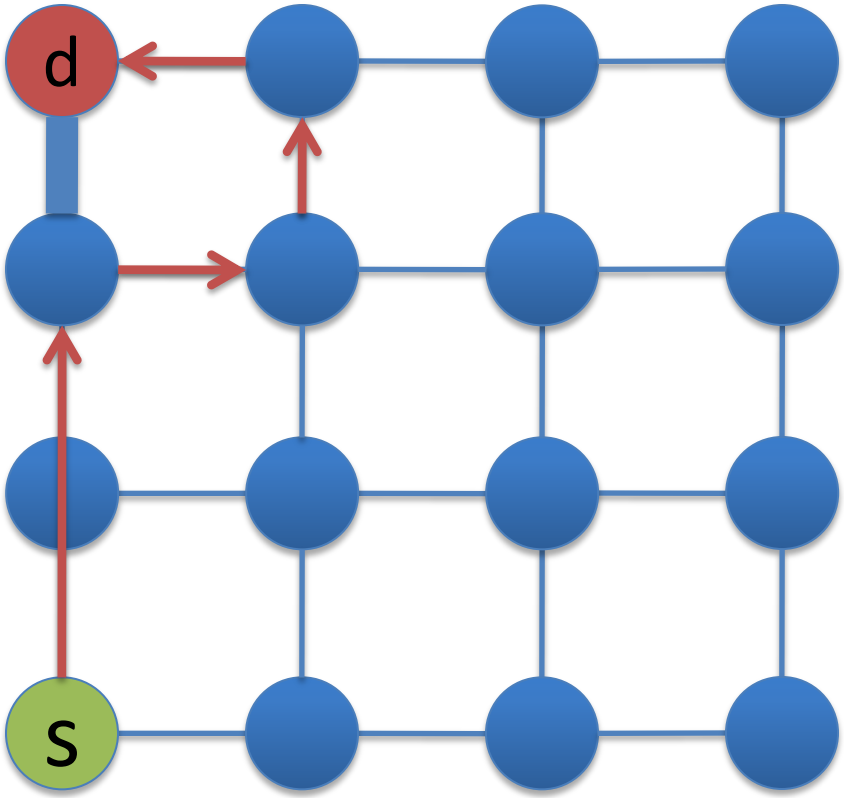


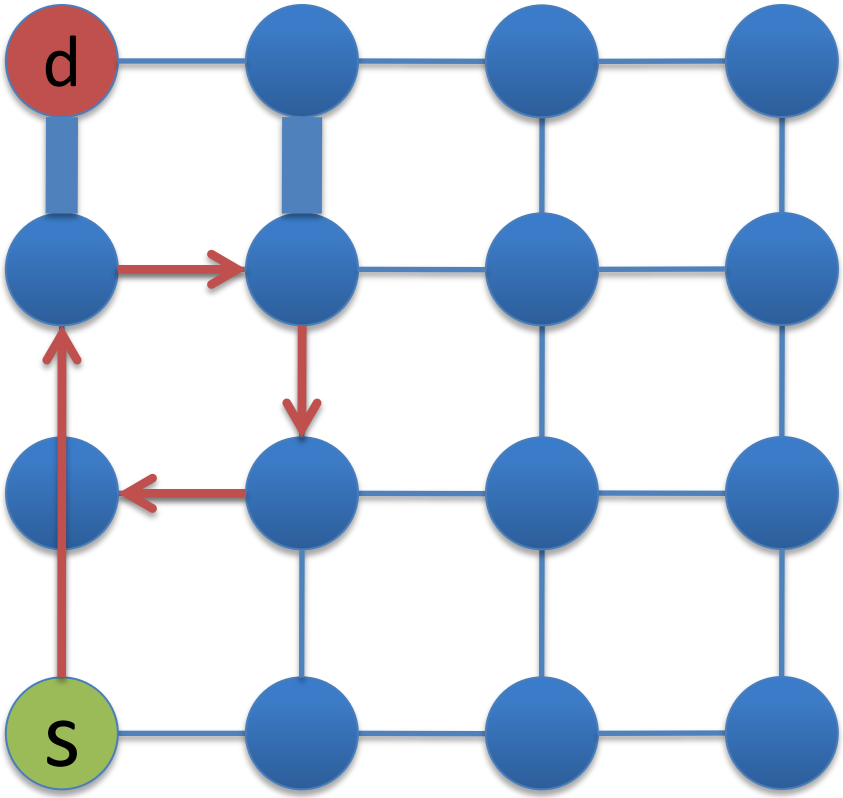

- Local info can result in sub-optimal choices

# Non-minimal adaptive

- Fully adaptive
- Not restricted to take shortest path
  - Example: FBfly
- Misrouting: directing packet along non-productive channel
  - Priority given to productive output
  - Some algorithms forbid U-turns
- Livelock potential: traversing network without ever reaching destination
  - Mechanism to guarantee forward progress
    - Limit number of misroutings

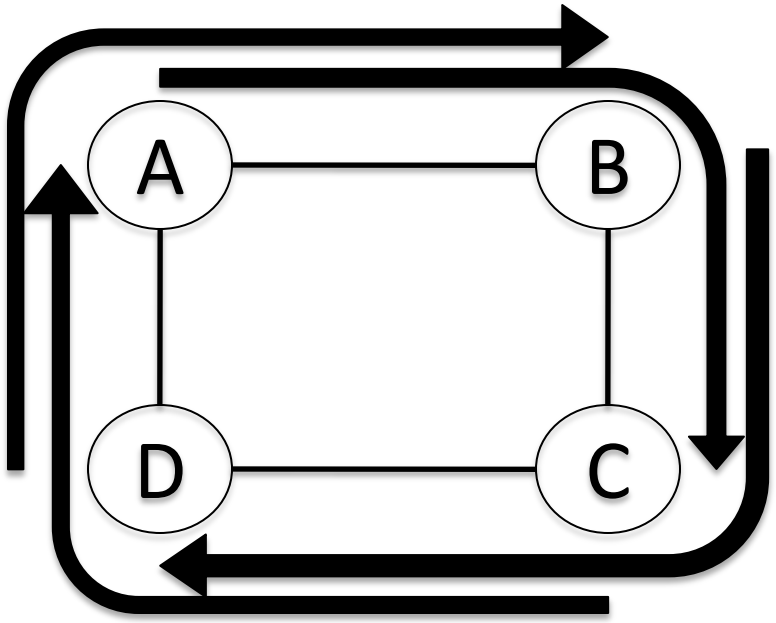
# Non-minimal routing example




- Longer path with potentially lower latency
- Livelock: continue routing in cycle

# Routing Deadlock

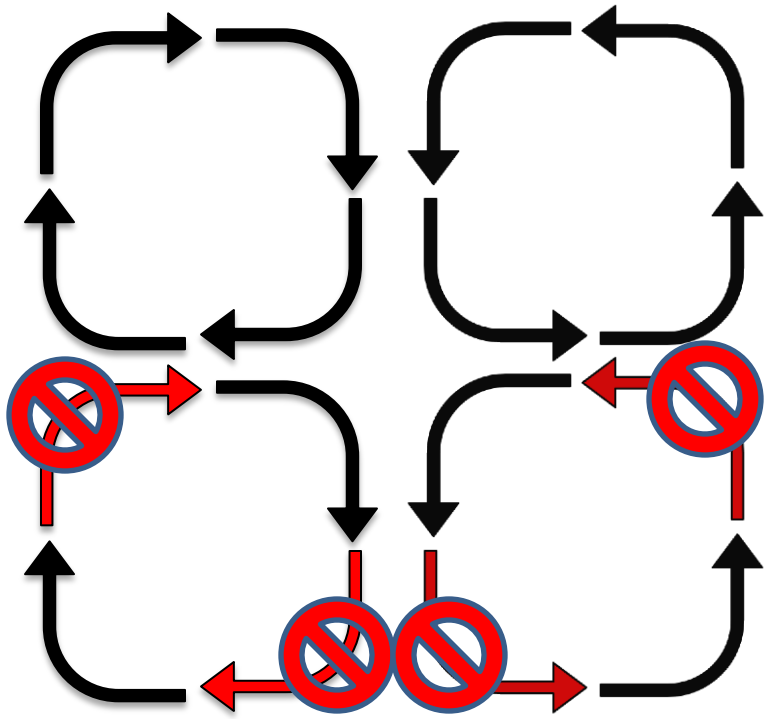



- Without routing restrictions, a resource cycle can occur
  - Leads to deadlock

# Eliminate Cycles by Construction

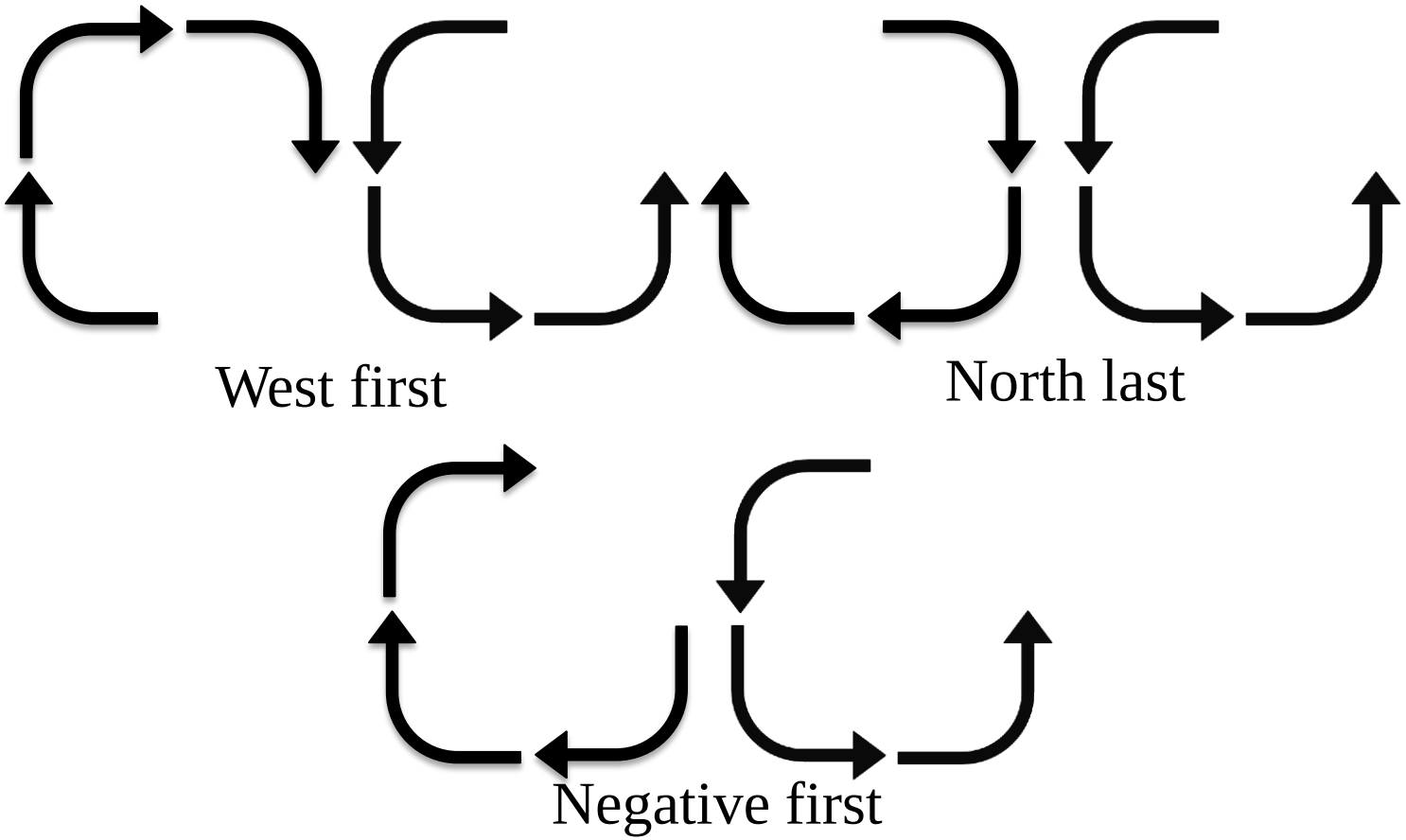
Deadlock Possible

XY Routing



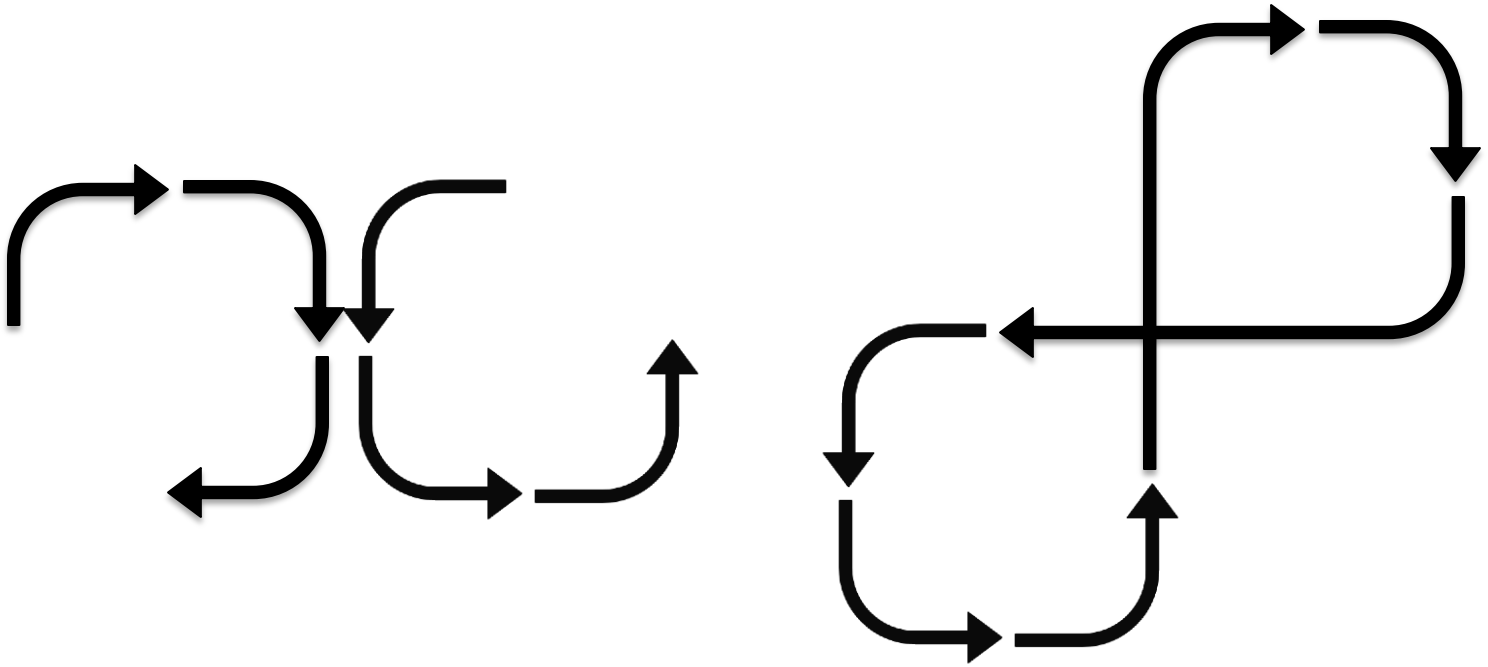
- Don’t allow turns that cause cycles
- In general, acquire resources in fixed priority order

# Turn Model Routing




- Some adaptivity by removing 2 of 8 turns
  - Remains deadlock free (but less restrictive than DOR)

# Turn Model Routing Deadlock



- Not a valid turn elimination
  - Resource cycle results

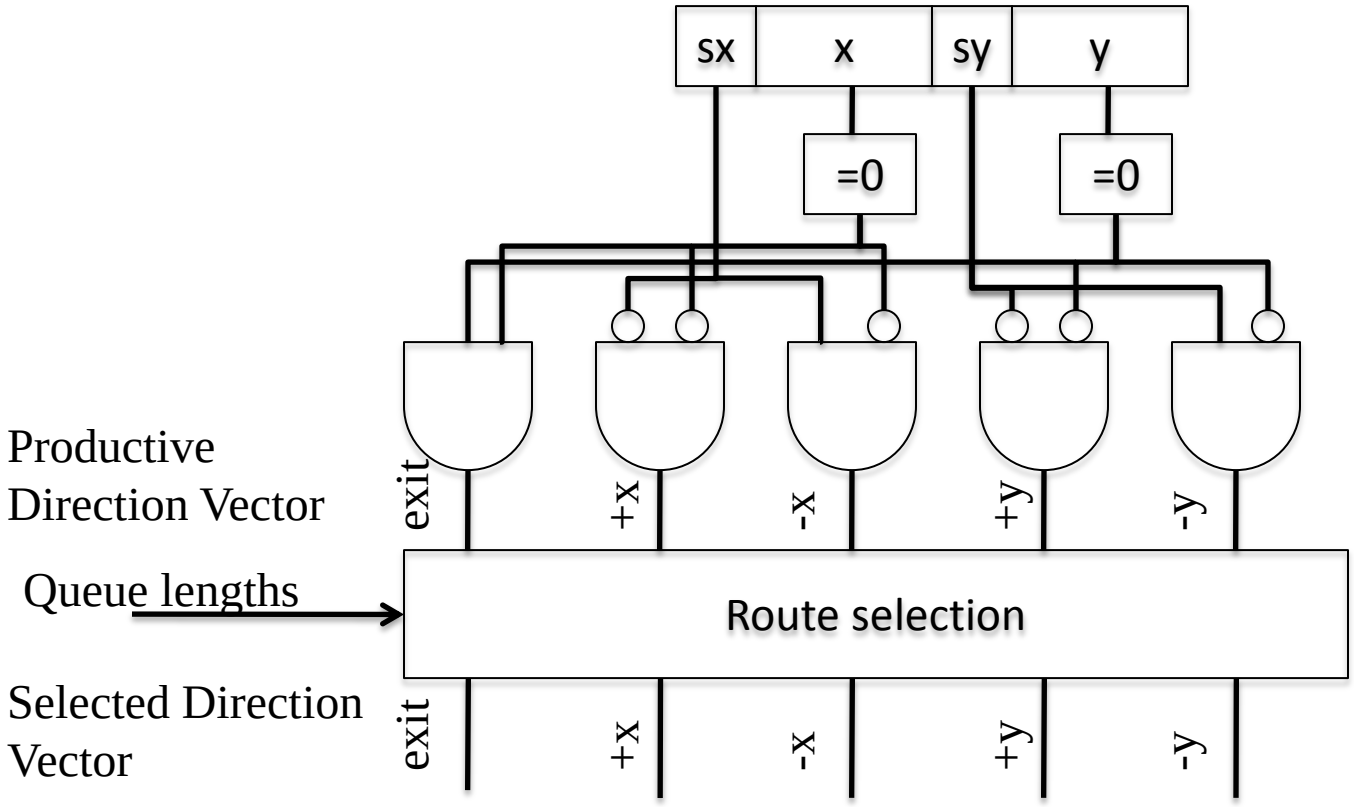
# Routing Implementation

- Source tables
  - Entire route specified at source
  - Avoids per-hop routing latency
  - Unable to adapt to network conditions
  - Can specify multiple routes per destination
- Node tables
  - Store only next routes at each node
  - Smaller tables than source routing
  - Adds per-hop routing latency
  - Can adapt to network conditions
    - Specify multiple possible outputs per destination

# Implementation

- Combinational circuits can be used
  - Simple (e.g. DOR): low router overhead
  - Specific to one topology and one routing algorithm
    - Limits fault tolerance
- Tables can be updated to reflect new configuration, network faults, etc

# Circuit Based



# Routing Summary

- Latency paramount concern
  - Minimal routing most common for NoC
  - Non-minimal can avoid congestion and deliver low latency
- To date: NoC research favors DOR for simplicity and deadlock freedom
  - On-chip networks often lightly loaded
- Only covered unicast routing
  - Recent work on extending on-chip routing to support multicast

# Topology & Routing References


- Topology
  - William J. Dally and C. L Seitz. The torus routing chip. Journal of Distributed Computing, 1(3):187–196, 1986.
  - Charles Leiserson. Fat-trees: Universal networks for hardware efficient supercomputing. IEEE Transactions on Computers, 34(10), October 1985.
  - Boris Grot, Joel Hestness, Stephen W. Keckler, and OnurMutlu. Express cube topologies for on-chip networks. In Proceedings of the International Symposium on High Performance Computer Architecture, February 2009.
  - Flattened butterfly topology for on-chip networks. In Proceedings of the 40th International Symposium on Microarchitecture, December 2007.
  - J. Balfour and W. Dally. Design tradeoffs for tiled cmp on-chip networks. In Proceedings of the International Conference on Supercomputing, 2006.
- Routing
  - L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In Proceedings of the 13th Annual ACM Symposium on Theory of Computing, pages 263–277, 1981.
  - D. Seo, A. Ali, W.-T. Lim, N. Rafique, and M. Thottenhodi. Near-optimal worst- case throughput routing in two dimensional mesh networks. In Proceedings of the 32nd Annual International Symposium on Computer Architecture, June.
  - Christopher J. Glass and Lionel M. Ni. The turn model for adaptive routing. In Proceedings of the International Symposium on Computer Architecture, 1992.
  - P. Gratz, B. Grot, and S. W. Keckler, “Regional congestion awareness for load balance in networks-on-chip,” in Proceedings of the 14th IEEE International Symposium on High-Performance Computer Architecture, February 2008.
  - N. EnrightJerger, L.-S. Peh, and M. H. Lipasti, “Virtual circuit tree multi- casting: A case for on-chip hardware multicast support,” in Proceedings of the International Symposium on Computer Architecture (ISCA-35), Beijing, China, June 2008.

# Lecture Outline

- Introduction to Networks
- Network Topologies
- Network Routing
- Network Flow Control
- Router Microarchitecture
- Technology example: On-chip Nanophotonics

# Flow Control Overview

- Topology: determines connectivity of network
- Routing: determines paths through network
- Flow Control: determine allocation of resources to messages as they traverse network
  - Buffers and links
  - Significant impact on throughput and latency of network

# Packets

- Messages: composed of one or more packets
  - If message size is <= maximum packet size only one packet created
- Packets: composed of one or more flits
- Flit: flow control digit
- Phit: physical digit
  - Subdivides flit into chunks = to link width
  - In on-chip networks, flit size == phit size.
    - Due to very wide on-chip channels

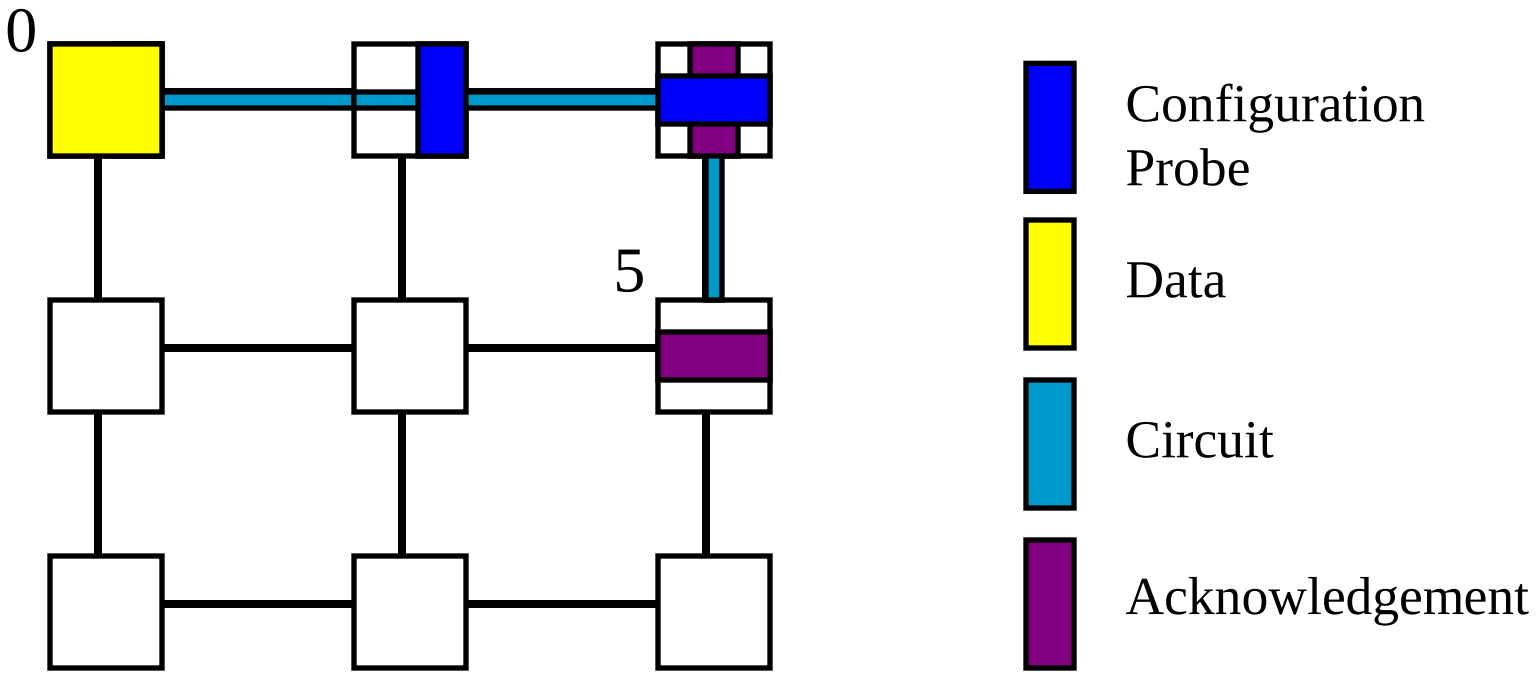
# Switching

- Different flow control techniques based on granularity
- Circuit-switching: operates at the granularity of messages
- Packet-based: allocation made to whole packets
- Flit-based: allocation made on a flit-by-flit basis

# Circuit Switching

- All resources (from source to destination) are allocated to the message prior to transport
  - Probe sent into network to reserve resources
- Once probe sets up circuit
  - Message does not need to perform any routing or allocation at each network hop
  - Good for transferring large amounts of data
    - Can amortize circuit setup cost by sending data with very low per-hop overheads
- No other message can use those resources until transfer is complete
  - Throughput can suffer due setup and hold time for circuits

# Circuit Switching Example

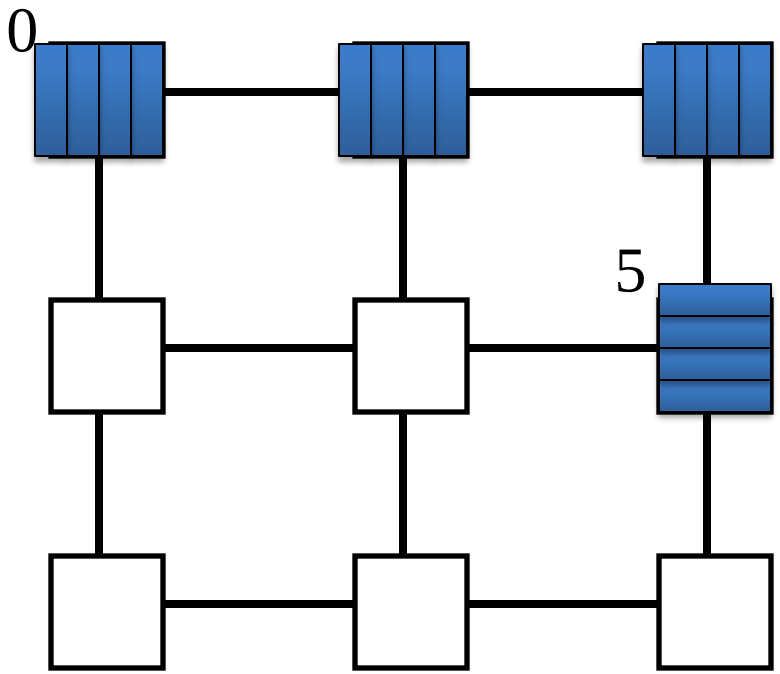



- Significant latency overhead prior to data transfer
- Other requests forced to wait for resources

# Packet-based Flow Control

- Store and forward
- Links and buffers are allocated to entire packet
- Head flit waits at router until entire packet is buffered before being forwarded to the next hop
- Not suitable for on-chip
  - Requires buffering at each router to hold entire packet
  - Incurs high latencies (pays serialization latency at each hop)

# Store and Forward Example

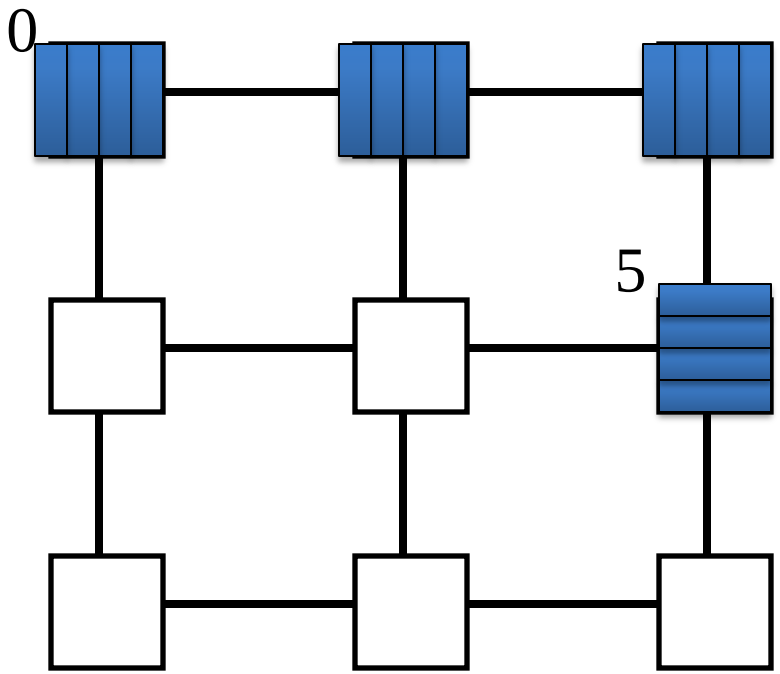



- High per-hop latency
- Larger buffering required

# Virtual Cut Through

- Packet-based: similar to Store and Forward
- Links and Buffers allocated to entire packets
- Flits can proceed to next hop before tail flit has been received by current router
  - But only if next router has enough buffer space for entire packet
- Reduces the latency significantly compared to SAF
- But still requires large buffers
  - Unsuitable for on-chip

# Virtual Cut Through Example

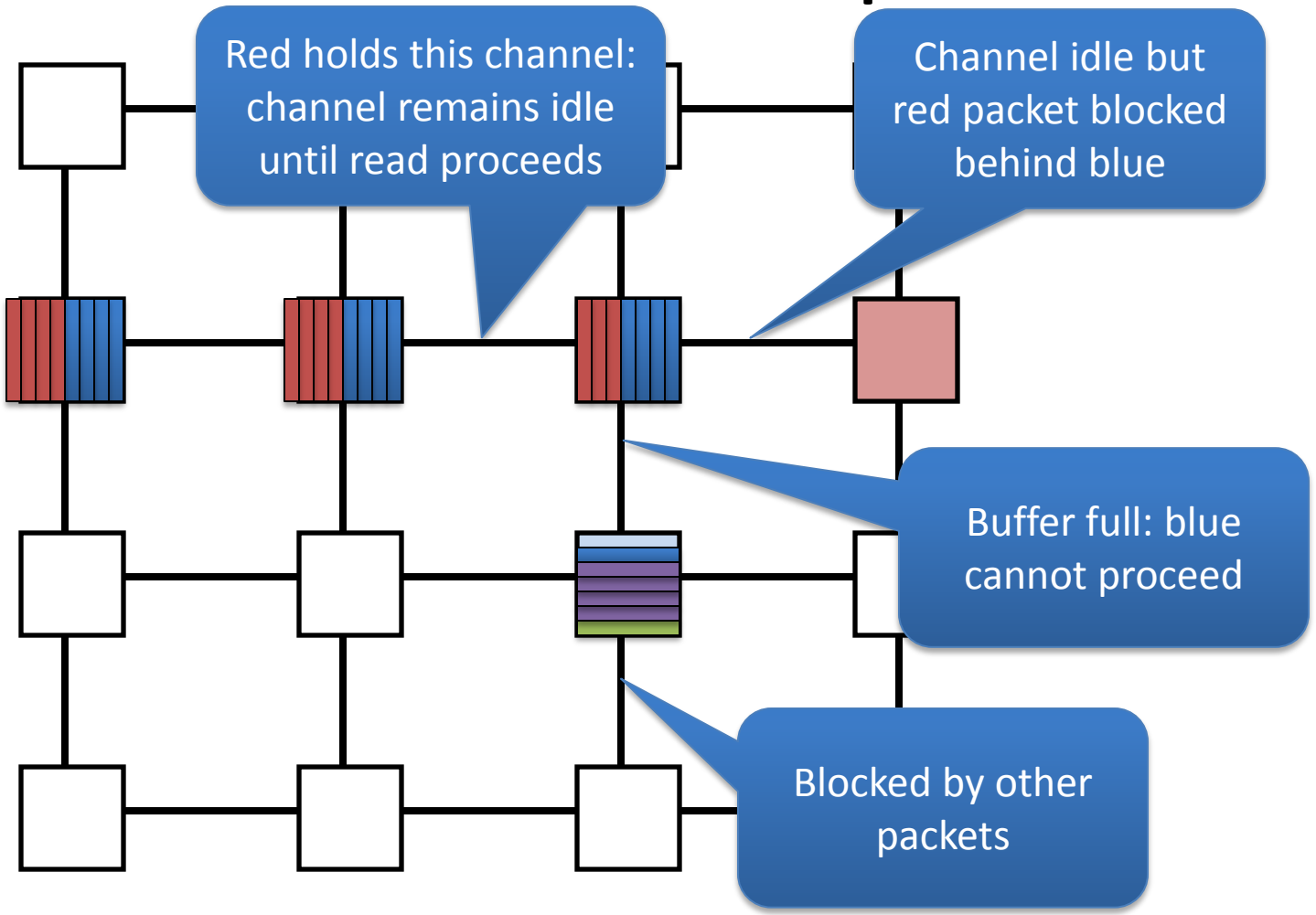



- Lower per-hop latency
- Larger buffering required

# Flit Level Flow Control

- Wormhole flow control
- Flit can proceed to next router when there is buffer space available for that flit
  - Improved over SAF and VCT by allocating buffers on a flit-basis
- Pros
  - More efficient buffer utilization (good for on-chip)
  - Low latency
- Cons
  - Poor link utilization: if head flit becomes blocked, all links spanning length of packet are idle
    - Cannot be re-allocated to different packet
    - Suffers from head of line (HOL) blocking

# Wormhole Example

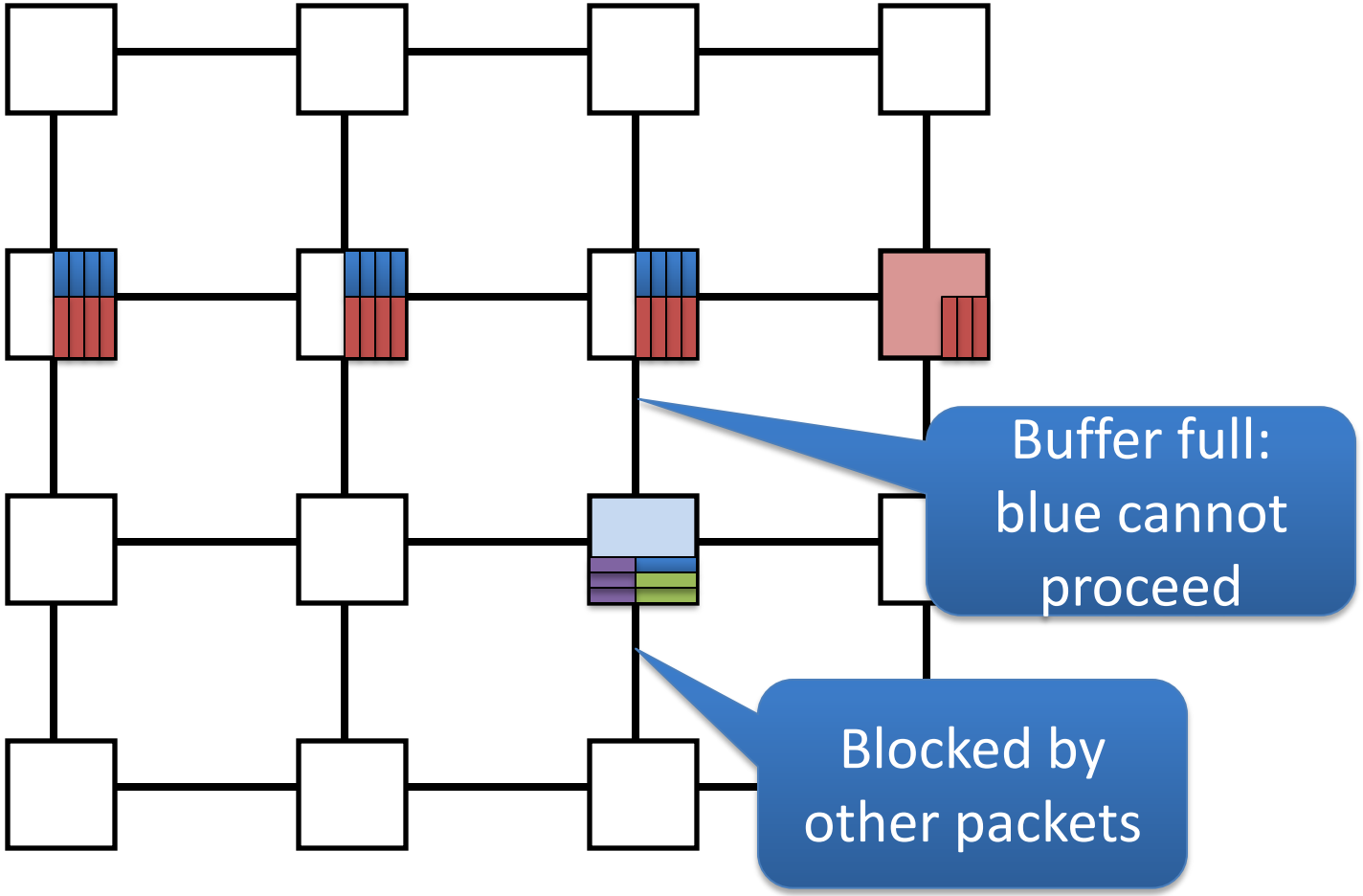



- 6 flit buffers/input port

# Virtual Channel Flow Control

- Virtual channels used to combat HOL block in wormhole
- Virtual channels: multiple flit queues per input port
  - Share same physical link (channel)
- Link utilization improved
  - Flits on different VC can pass blocked packet

# Virtual Channel Example




- 6 flit buffers/input port
- 3 flit buffers/VC

# Deadlock

- Using flow control to guarantee deadlock freedom give more flexible routing
- Escape Virtual Channels
  - If routing algorithm is not deadlock free
  - VCs can break resource cycle
  - Place restriction on VC allocation or require one VC to be DOR
- Assign different message classes to different VCs to prevent protocol level deadlock
  - Prevent req-ack message cycles

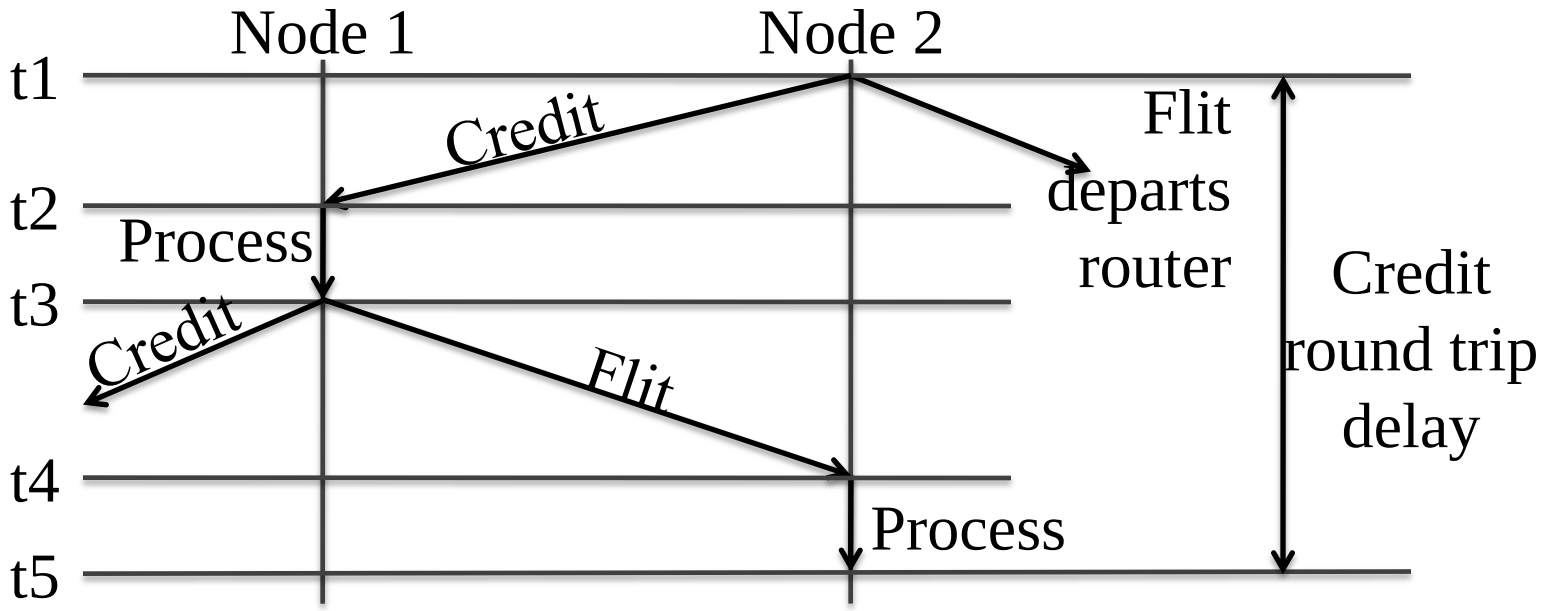
# Buffer Backpressure

- Need mechanism to prevent buffer overflow
  - Avoid dropping packets
  - Upstream nodes need to know buffer availability at downstream routers
- Significant impact on throughput achieved by flow control
- Credits
- On-off

# Credit-Based Flow Control

- Upstream router stores credit counts for each downstream VC
- Upstream router
  - When flit forwarded
    - Decrement credit count
  - Count == 0, buffer full, stop sending
- Downstream router
  - When flit forwarded and buffer freed
    - Send credit to upstream router
    - Upstream increments credit count

# Credit Timeline

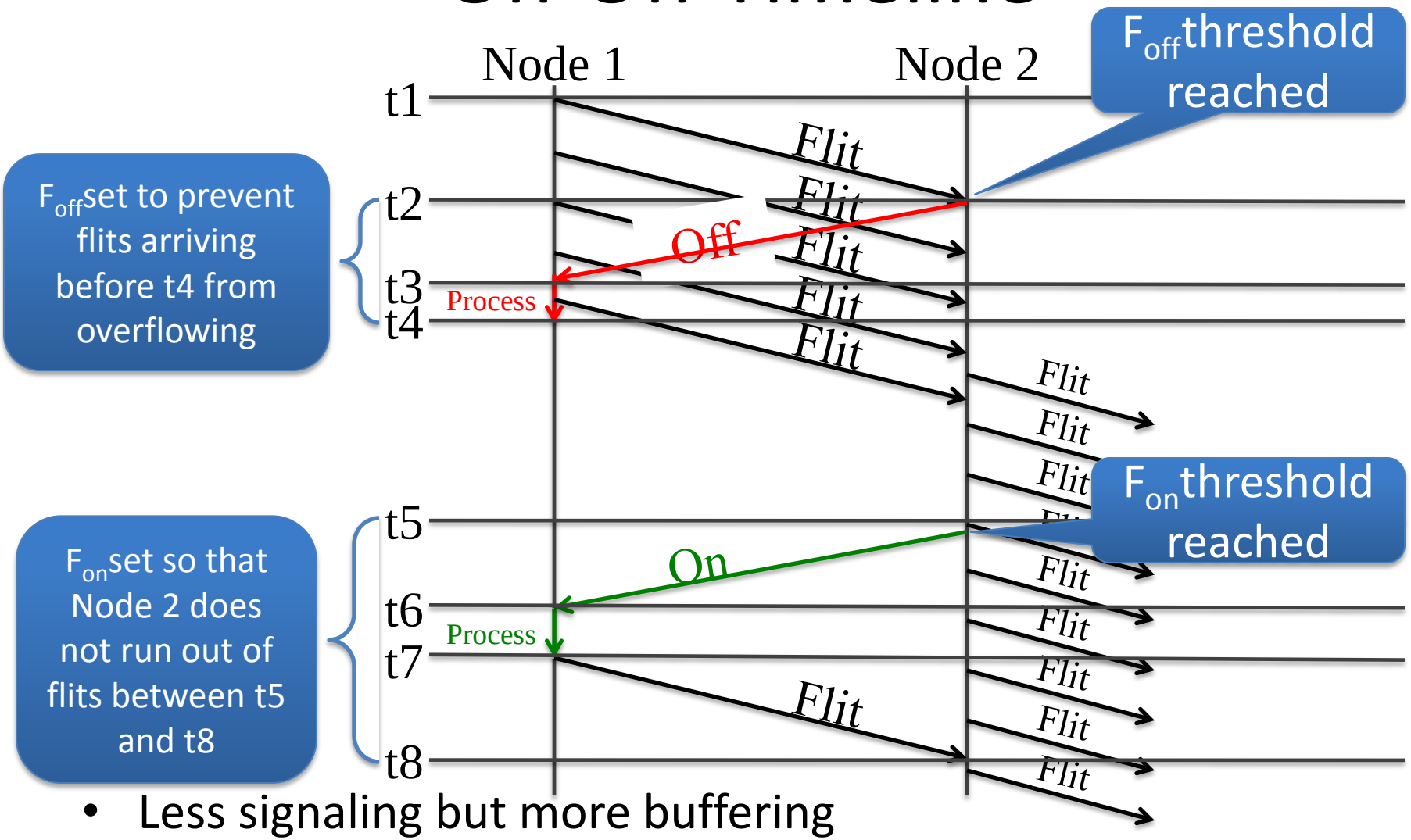



- Round-trip credit delay:
  - Time between when buffer empties and when next flit can be processed from that buffer entry
  - If only single entry buffer, would result in significant throughput degradation
  - Important to size buffers to tolerate credit turn-around

# On-Off Flow Control

- Credit: requires upstream signaling for every flit
- On-off: decreases upstream signaling
- Off signal
  - Sent when number of free buffers falls below threshold $F_{off}$
- On signal
  - Send when number of free buffers rises above threshold $F_{on}$

# On-Off Timeline




- Less signaling but more buffering
  - On-chip buffers more expensive than wires

# Virtual Channels vs. Physical Channels

- Virtual channels share one physical set of links
  - Buffers hold flits that are in flight to free up physical channel
  - Flits from alternate VCs can traverse shared physical link
- This was the right design decision when:
  - Links were expensive (off-chip, backplane traces, or cables)
  - Buffers/router resources were cheap
  - Router latency was a small fraction of link latency
- With modern on-chip networks, this may not be true
  - Links are cheap (just global wires)
  - Buffer/router resources consume dynamic and static power, hence not cheap
  - Router latency is significant relative to link latency (usually just one cycle)
- Tilera design avoids virtual channels, simply provides multiple physical channels to avoid deadlock

# Flow Control Summary

- On-chip networks require techniques with lower buffering requirements
  - Wormhole or Virtual Channel flow control
- Dropping packets unacceptable in on-chip environment
  - Requires buffer backpressure mechanism
- Complexity of flow control impacts router microarchitecture (next)

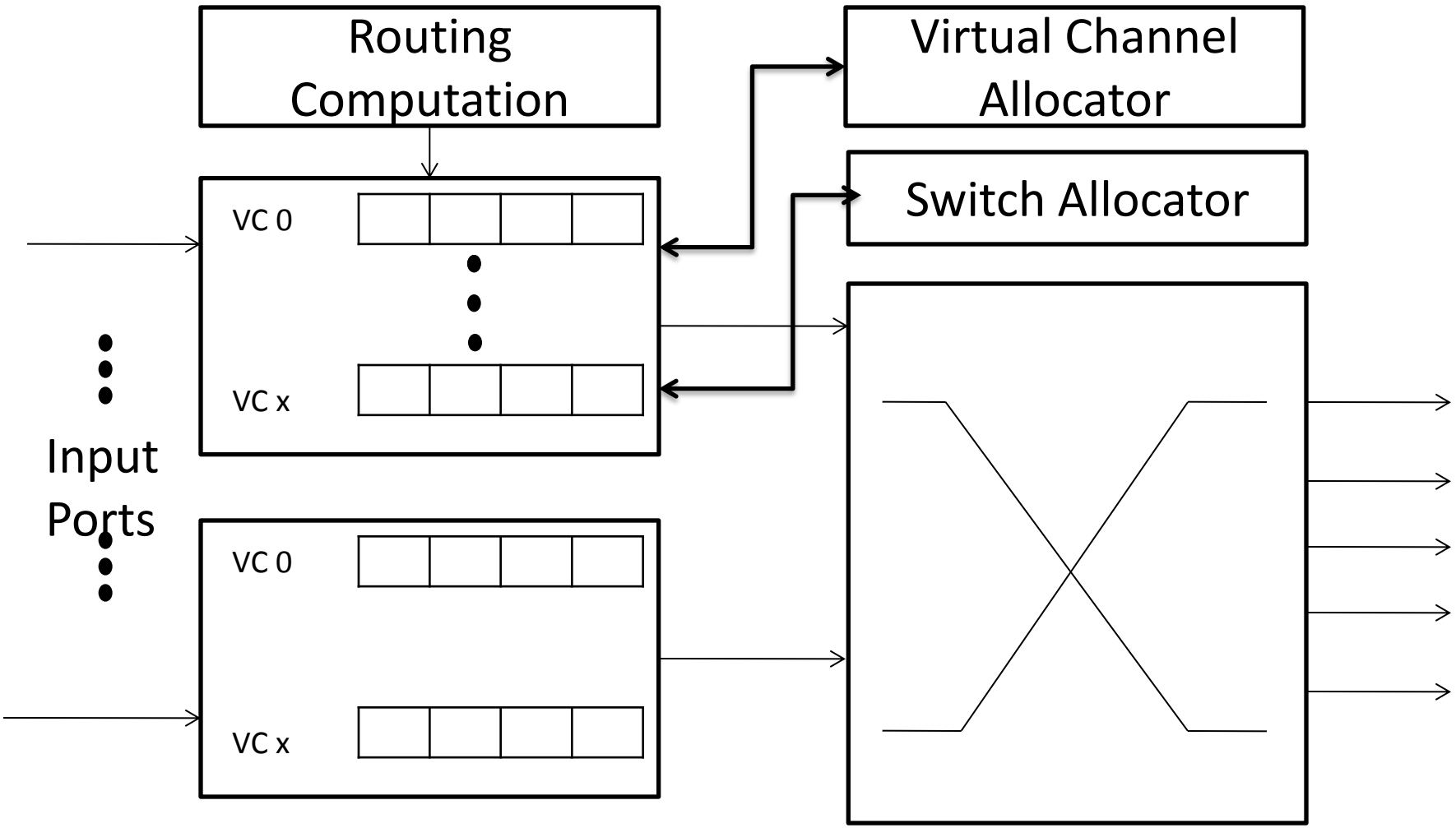
# Lecture Outline

- Introduction to Networks
- Network Topologies
- Network Routing
- Network Flow Control
- Router Microarchitecture
- Technology example: On-chip Nanophotonics

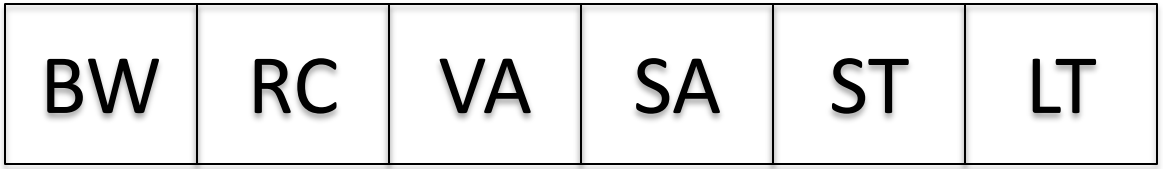
# Router Microarchitecture Overview

- Consist of buffers, switches, functional units, and control logic to implement routing algorithm and flow control
- Focus on microarchitecture of Virtual Channel router
- Router is pipelined to reduce cycle time

# Virtual Channel Router

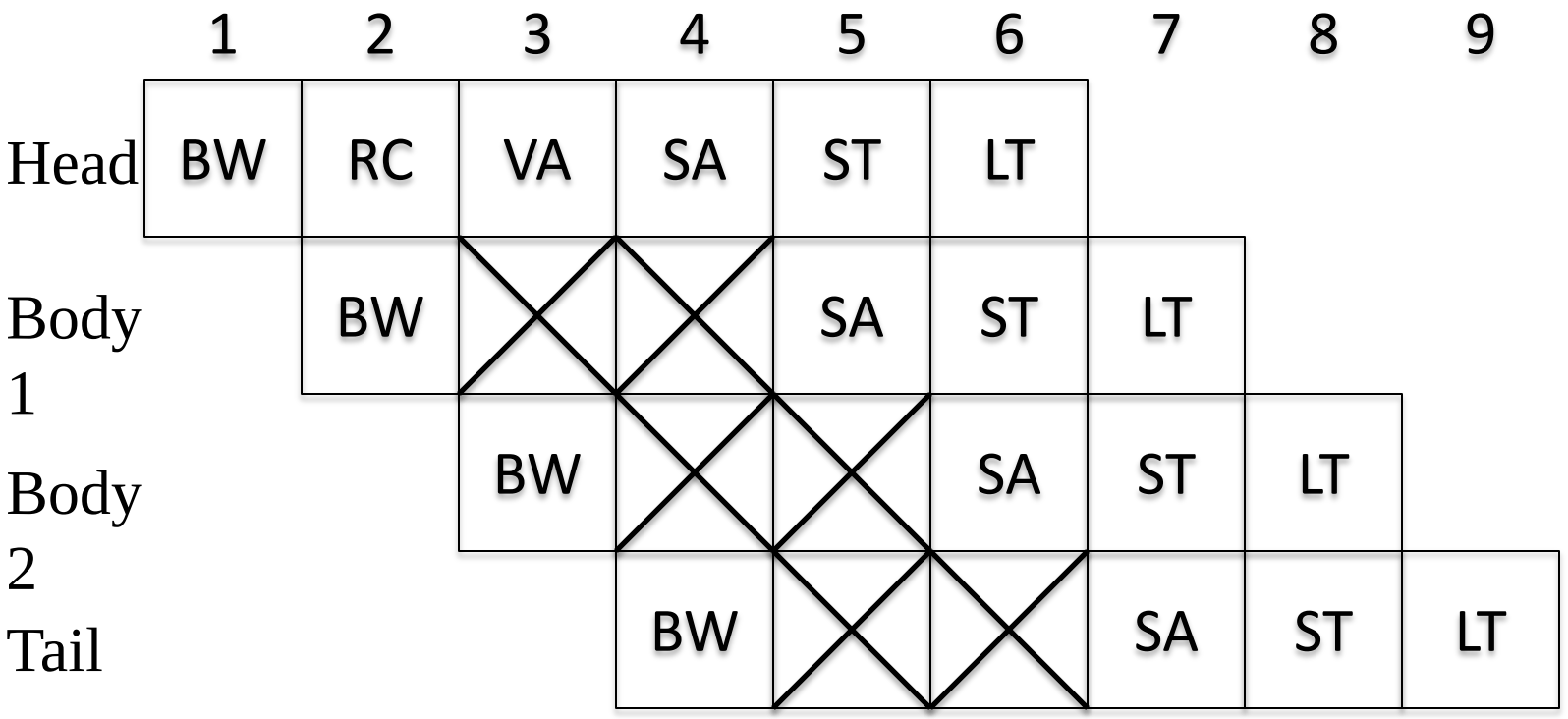


# Baseline Router Pipeline




- Canonical 5-stage (+link) pipeline
  - BW: Buffer Write
  - RC: Routing computation
  - VA: Virtual Channel Allocation
  - SA: Switch Allocation
  - ST: Switch Traversal
  - LT: Link Traversal

# Baseline Router Pipeline (2)



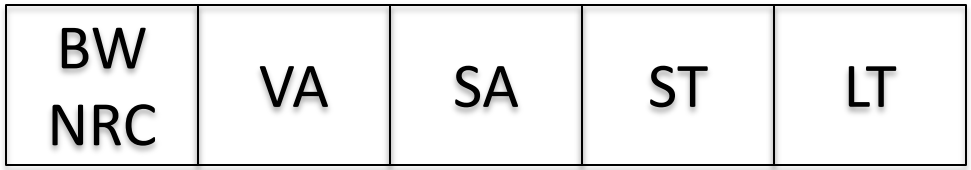

- Routing computation performed once per packet
- Virtual channel allocated once per packet
- body and tail flits inherit this info from head flit

# Router Pipeline Optimizations

- Baseline (no load) delay

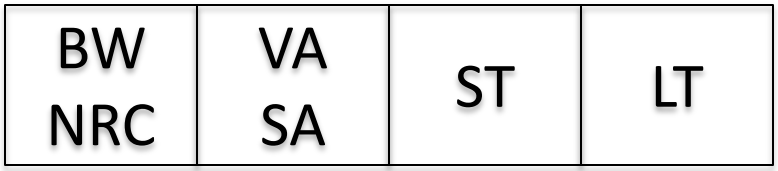
$$= (5 cycles + link\,delay) \times hops + t_{serialization}$$

- Ideally, only pay link delay
- Techniques to reduce pipeline stages
  - Lookahead routing: At current router perform routing computation for next router
    - Overlap with BW



# Router Pipeline Optimizations (2)

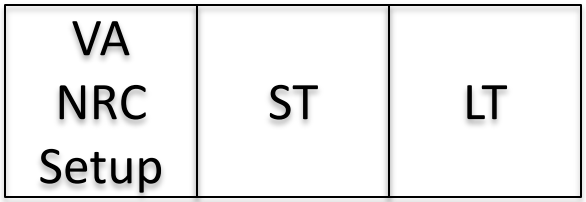
- Speculation
  - Assume that Virtual Channel Allocation stage will be successful
    - Valid under low to moderate loads
  - Entire VA and SA in parallel




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

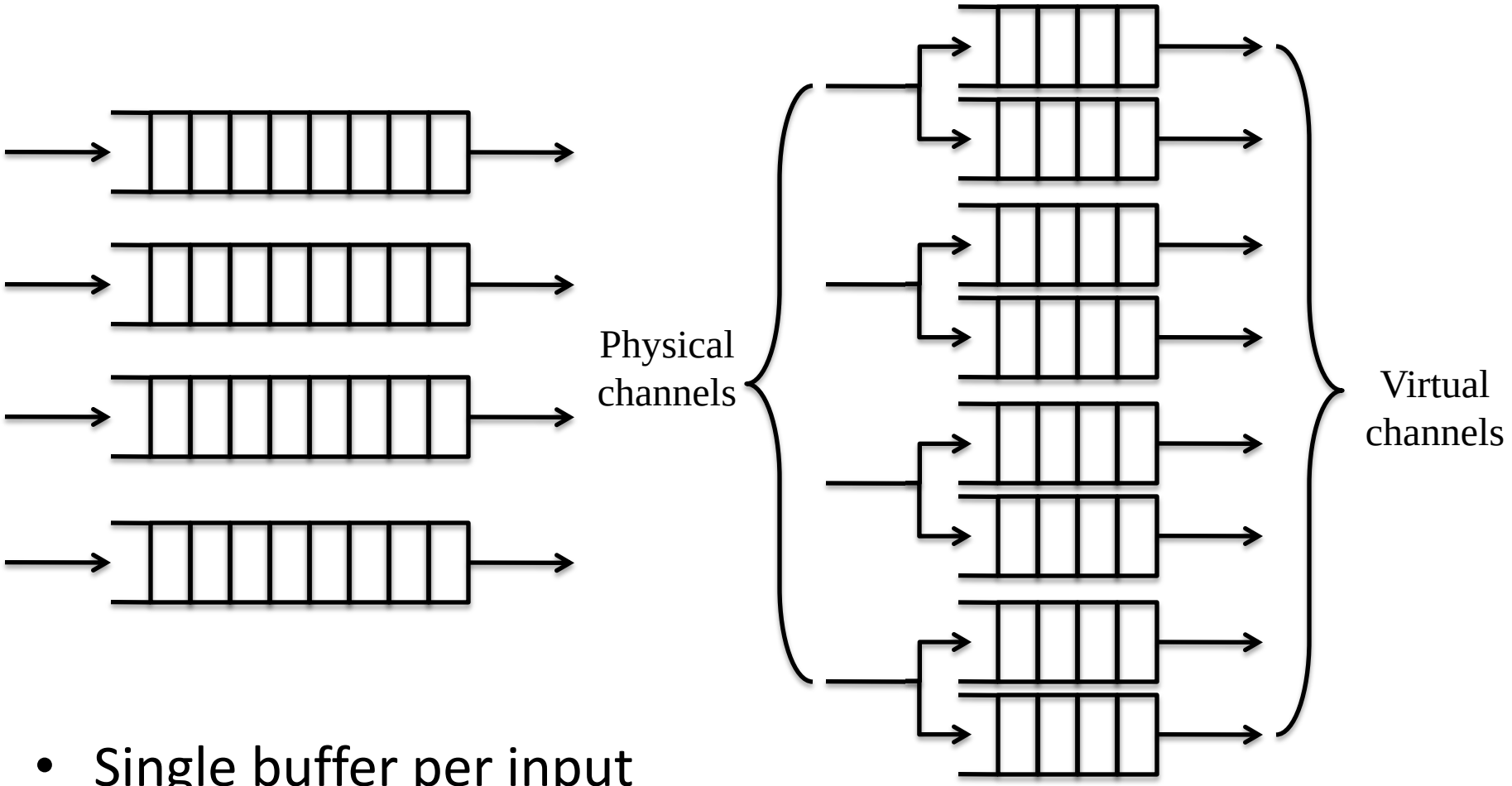
# Router Pipeline Optimizations (3)

- Bypassing: when no flits in input buffer
  - Speculatively enter ST
  - On port conflict, speculation aborted




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

# Buffer Organization




- Single buffer per input
- Multiple fixed length queues per physical channel

# Arbiters and Allocators

- *Allocator* matches N requests to M resources
- *Arbiter* matches N requests to 1 resource
- Resources are VCs (for virtual channel routers) and crossbar switch ports.
- Virtual-channel allocator (VA)
  - Resolves contention for output virtual channels
  - Grants them to input virtual channels
- Switch allocator (SA) that grants crossbar switch ports to input virtual channels
- Allocator/arbiter that delivers high matching probability translates to higher network throughput.
  - Must also be fast and able to be pipelined

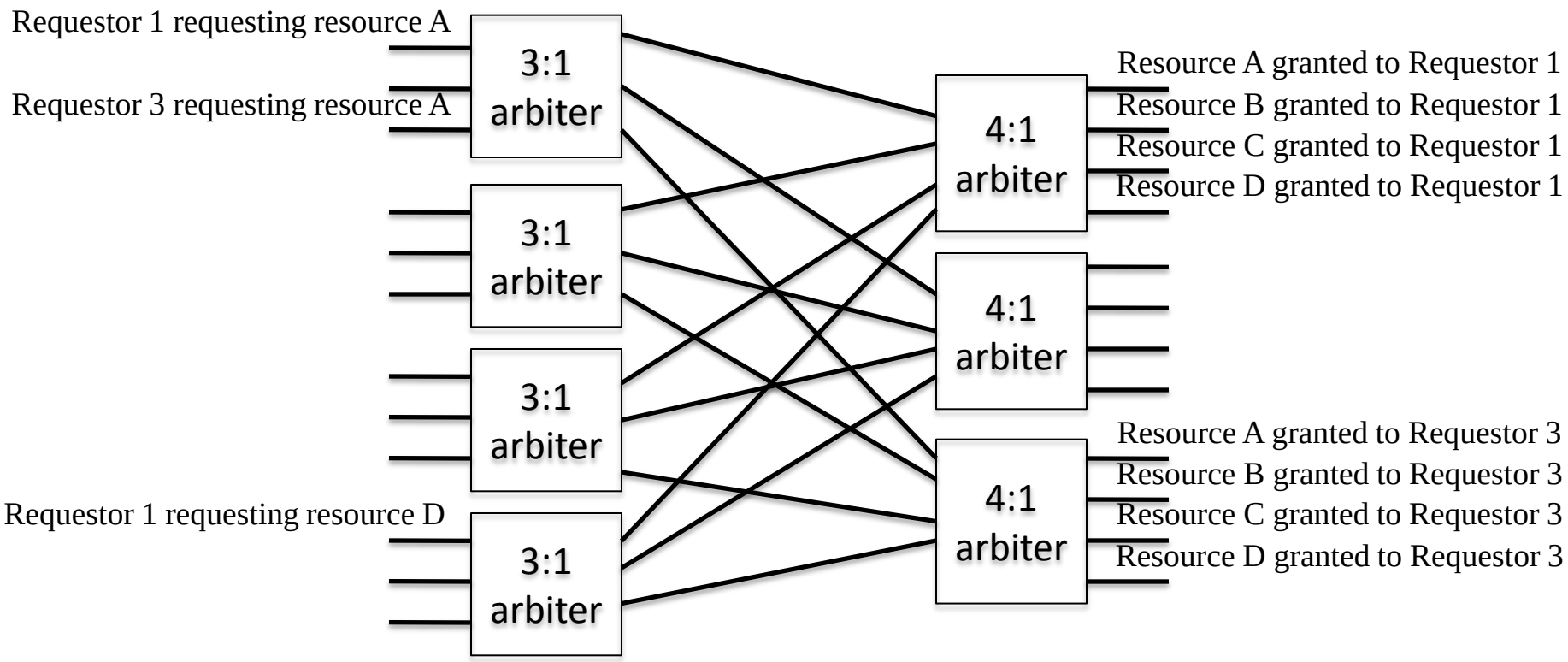
# Round Robin Arbiter

- Last request serviced given lowest priority
- Generate the next priority vector from current grant vector
- Exhibits fairness

# Matrix Arbiter

- Least recently served priority scheme
- Triangular array of state bits $w_{ij}$ for i<j
  - Bit $w_{ij}$ indicates request i takes priority over j
  - Each time request k granted, clears all bits in row k and sets all bits in column k
- Good for small number of inputs
- Fast, inexpensive and provides strong fairness

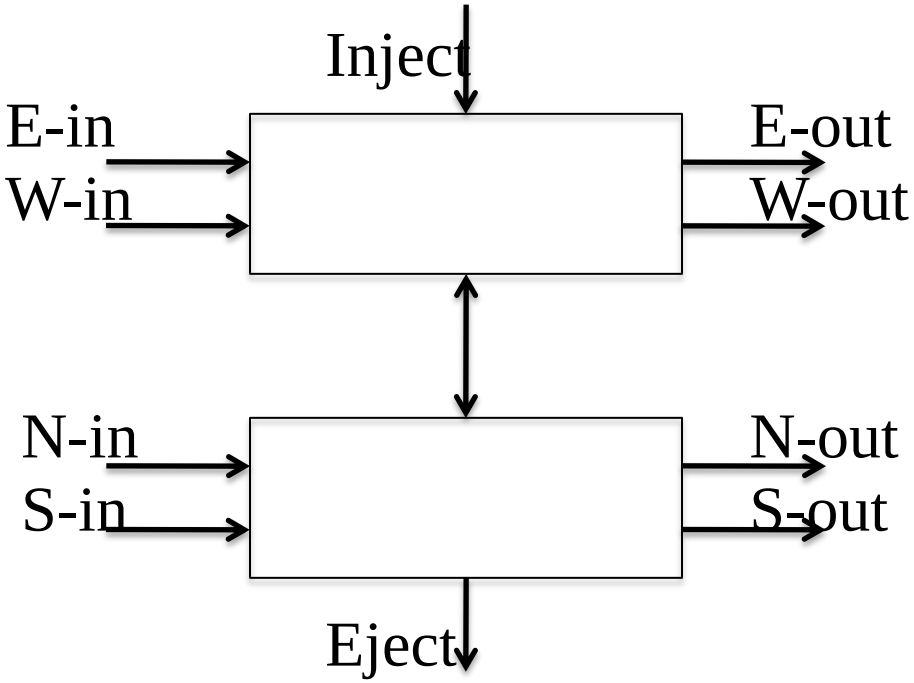
# Separable Allocator




- A 3:4 allocator
- First stage: decides which of 3 requestors wins specific resource
- Second stage: ensures requestor is granted just 1 of 4 resources

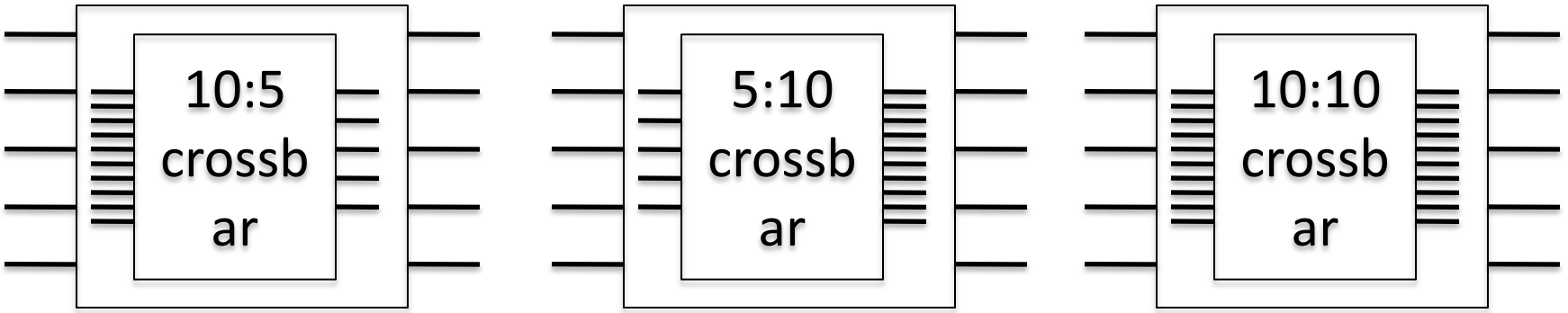
# Crossbar Dimension Slicing

- Crossbar area and power grow with $O((pw)^2)$




- Replace 1 5x5 crossbar with 2 3x3 crossbars

# Crossbar speedup




- Increase internal switch bandwidth
- Simplifies allocation or gives better performance with a simple allocator
- Output speedup requires output buffers
  - Multiplex onto physical link

# Evaluating Interconnection Networks

- Network latency
  - Zero-load latency: average distance * latency per unit distance
- Accepted traffic
  - Measure the max amount of traffic accepted by the network before it reaches saturation
- Cost
  - Power, area, packaging

# Interconnection Network Evaluation

- Trace based
  - Synthetic trace-based
    - Injection process
      - Periodic, Bernoulli, Bursty
  - Workload traces
- Full system simulation

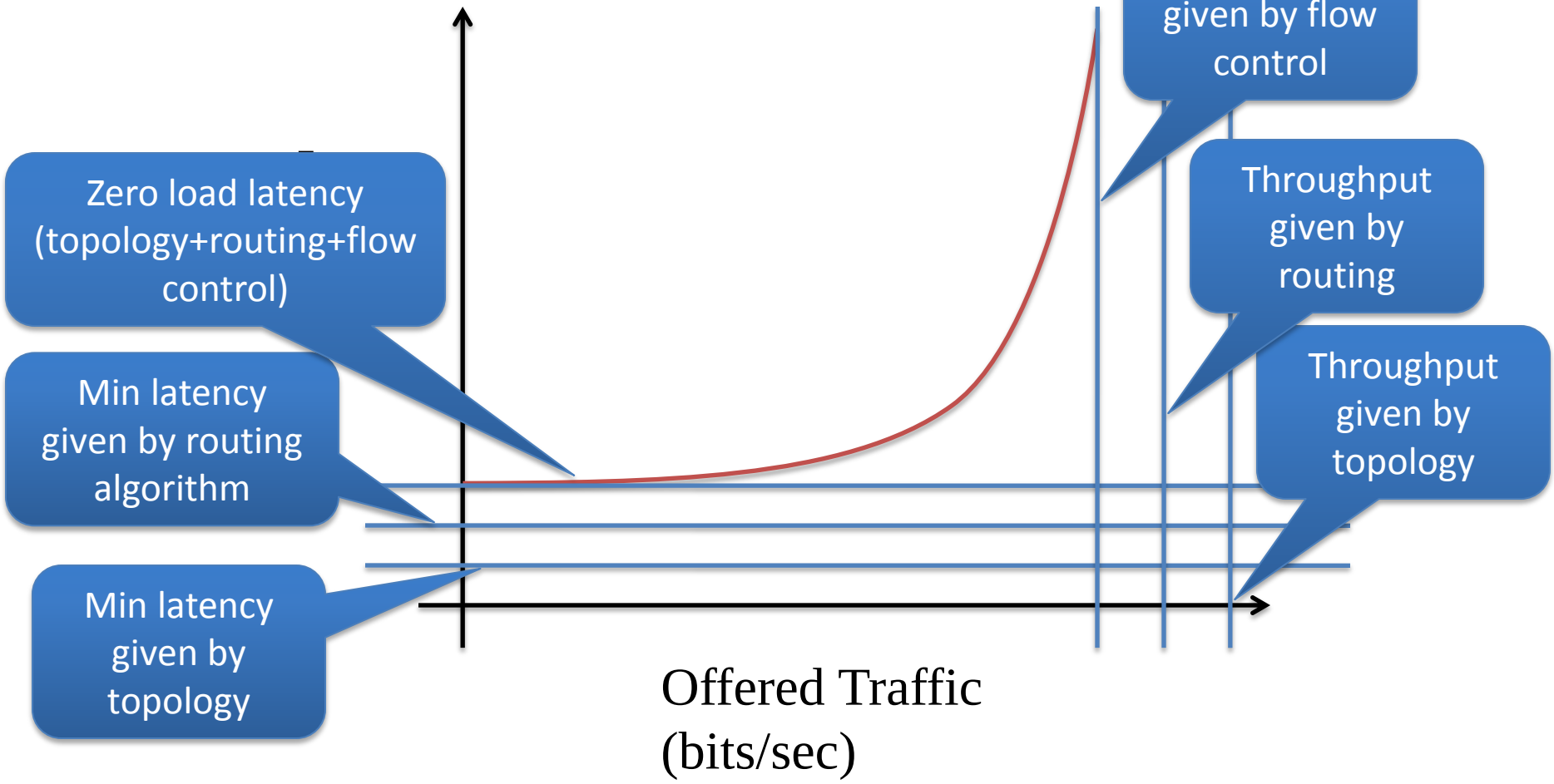
# Traffic Patterns

- Uniform Random
  - Each source equally likely to send to each destination
  - Does not do a good job of identifying load imbalances in design
- Permutation (several variations)
  - Each source sends to one destination
- Hot-spot traffic
  - All send to 1 (or small number) of destinations

# Microarchitecture Summary

- Ties together topological, routing and flow control design decisions
- Pipelined for fast cycle times
- Area and power constraints important in NoC design space

# Interconnection Network S ry




- Latency vs. Offered Traffic

# Flow Control and Microarchitecture References


- Flow control
  - William J. Dally. Virtual-channel flow control. In Proceedings of the International Symposium on Computer Architecture, 1990.
  - P. Kermani and L. Kleinrock. Virtual cut-through: a new computer communication switching technique. Computer Networks, 3(4):267–286.
  - Jose Duato. A new theory of deadlock-free adaptive routing in wormhole networks. IEEE Transactions on Parallel and Distributed Systems, 4:1320–1331, 1993.
  - Amit Kumar, Li-ShiuanPeh, ParthaKundu, and Niraj K. Jha. Express virtual channels: Toward the ideal interconnection fabric. In Proceedings of 34th Annual International Symposium on Computer Architecture, San Diego, CA, June 2007.
  - Amit Kumar, Li-ShiuanPeh, and Niraj K Jha. Token flow control. In Proceedings of the 41st International Symposium on Microarchitecture, Lake Como, Italy, November 2008.
  - Li-ShiuanPeh and William J. Dally. Flit reservation flow control. In Proceedings of the 6th International Symposium on High Performance Computer Architecture, February 2000.
- Router Microarchitecture
  - Robert Mullins, Andrew West, and Simon Moore. Low-latency virtual-channel routers for on-chip networks. In Proceedings of the International Symposium on Computer Architecture, 2004.
  - Pablo Abad, Valentin Puente, Pablo Prieto, and Jose Angel Gregorio. Rotary router: An efficient architecture for cmp interconnection networks. In Proceedings of the International Symposium on Computer Architecture, pages 116–125, June 2007.
  - Shubhendu S. Mukherjee, PetterBannon, Steven Lang, Aaron Spink, and David Webb. The Alpha 21364 network architecture. IEEE Micro, 22(1):26–35, 2002.
  - Jongman Kim, ChrysostomosNicopoulos, Dongkook Park, Vijaykrishnan Narayanan, Mazin S. Yousif, and Chita R. Das. A gracefully degrading and energy- efficient modular router architecture for on-chip networks. In Proceedings of the International Symposium on Computer Architecture, pages 4–15, June 2006.
  - M. Galles. Scalable pipelined interconnect for distributed endpoint routing: The SGI SPIDER chip. In Proceedings of Hot Interconnects Symposium IV, 1996.

# Lecture Outline

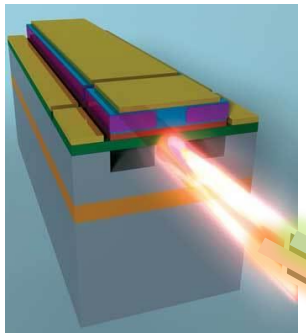
- Introduction to Networks
- Network Topologies
- Network Routing
- Network Flow Control
- Router Microarchitecture
- Technology example: On-chip Nanophotonics

# Distributed processing on chip

- Future chips rely on distributed processing
  - Many computation/cache/DRAM/IO nodes
  - Placement, topology, core uarch/strength, tbd
- Conventional interconnects may not suffice
  - Buses not viable
  - Crossbars are slow, power-hungry, expensive
  - NOCs impose latency, power overhead
- Nanophotonics to the rescue
  - Communicate with photons
  - Inherent bandwidth, latency, energy advantages
  - Silicon integration becoming a reality
- Challenges & opportunities remain

# Si Photonics: How it works

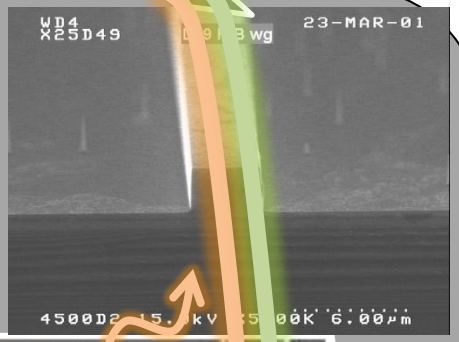
Laser
- Off-Chip Power



[Koch '07]

0.5 μm

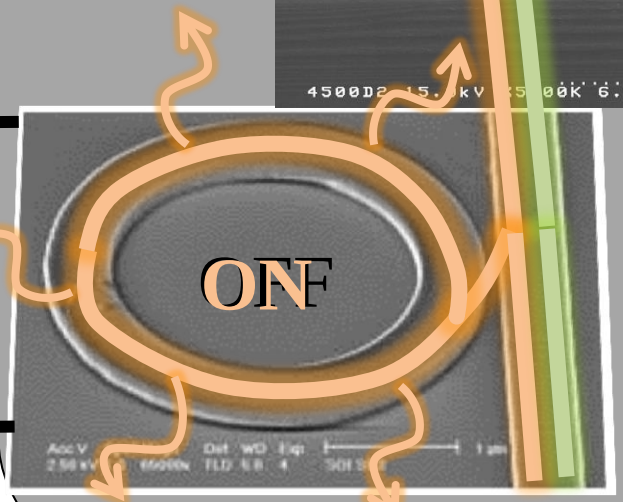
Waveguide
- Optical Wire




[Intel]

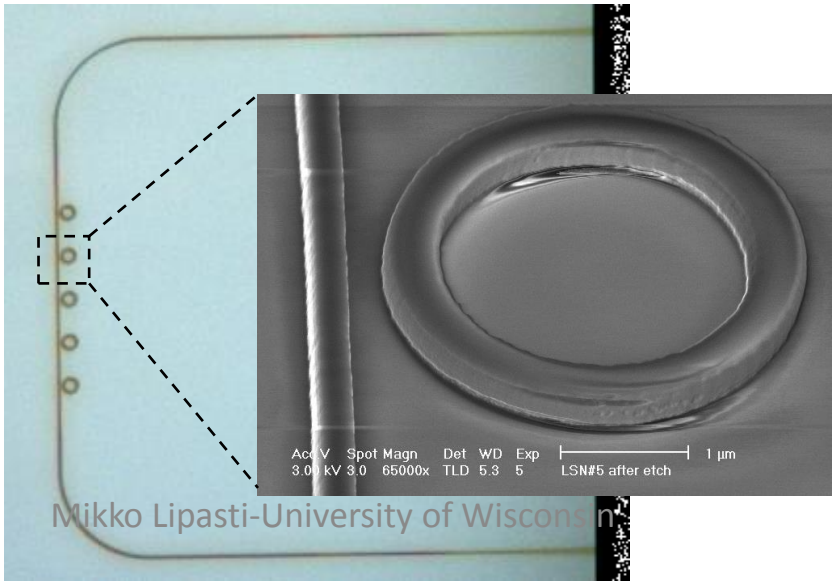
~3.5 μm

Ring Resonator
- Wavelength Magnet




[HP]

# Ring Resonators



# Key attributes of Si photonics

- Very low latency, very high bandwidth
- Up to 1000x energy efficiency gain
- Challenges
  - Resonator thermal tuning: heaters
  - Integration, fabrication, *is this real?*
- Opportunities
  - Static power dominant (laser, thermal)
  - Destructive reads: fast wired or

# Nanophotonics

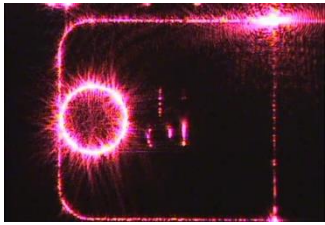
☑Nanophotonics overview

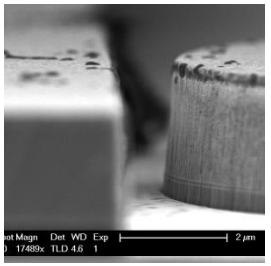
- Sharing the nanophotonic channel
  - Light-speed arbitration [MICRO 09]
- Utilizing the nanophotonic channel
  - Atomic coherence [HPCA 11]

# Corona substrate [ISCA08]

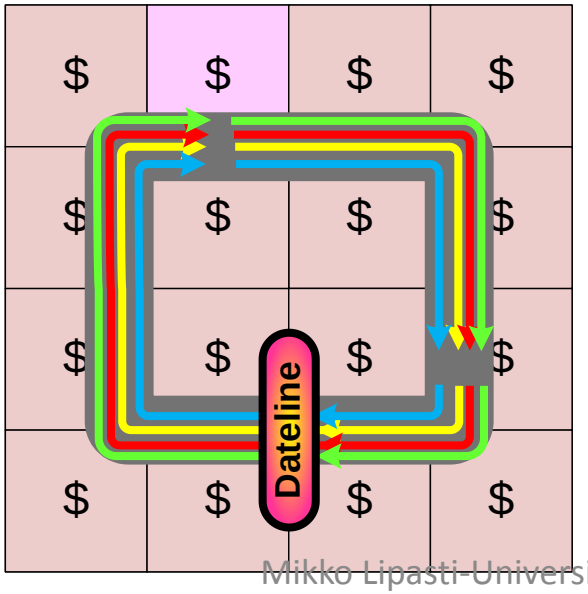
Targeting Year 2017

- Logically a ring topology
- One concentric ring per node
- 3D stacked: optical, analog, digital

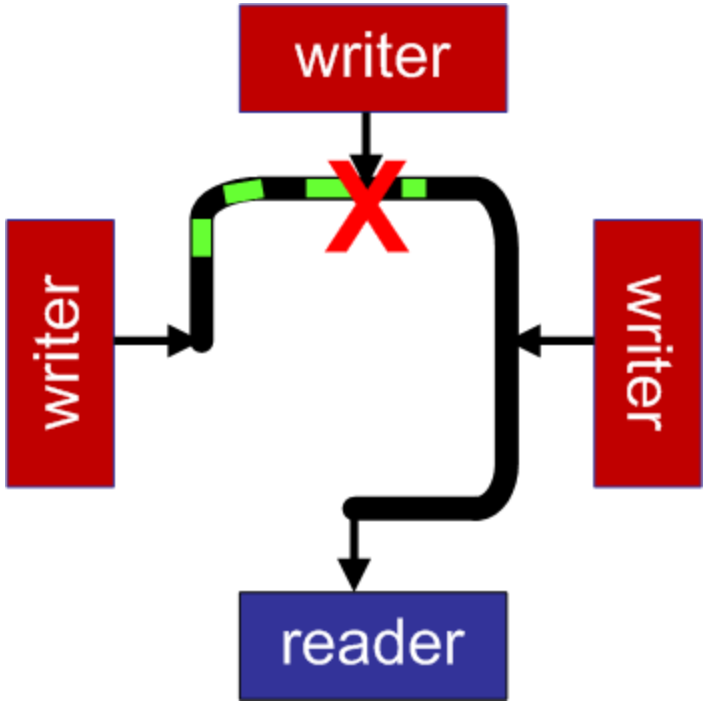








# Multiple writer single reader (MWSR) interconnects




latchless/
wave-pipelined

Arbitration prevents corruption of in-flight data

# Motivating an optical arbitration solution

MWSR Arbiter must be:

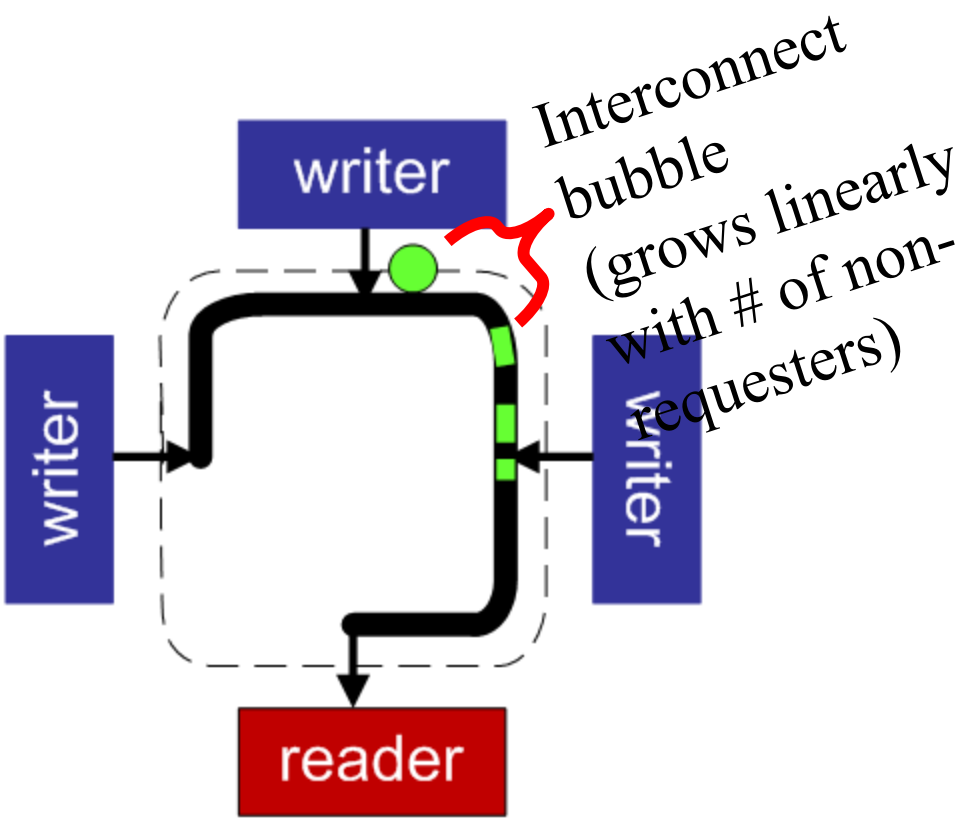
1. ***Global*** - Many writers requesting access
2. ***Very fast*** – Otherwise bottleneck

Optical arbiter avoids OEO conversion delays, provides light-speed arbitration

# Proposed optical protocols

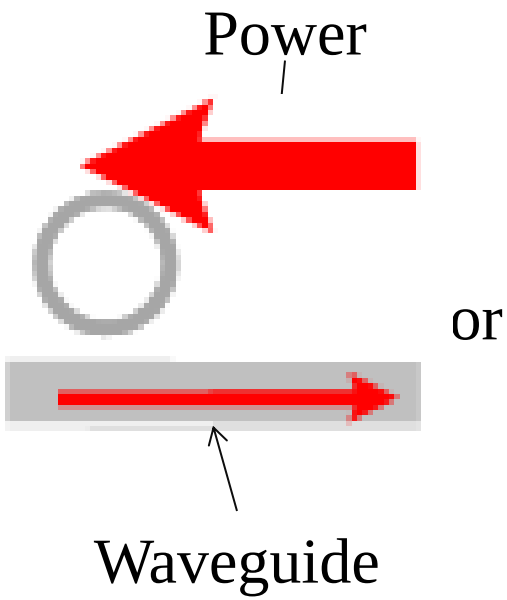
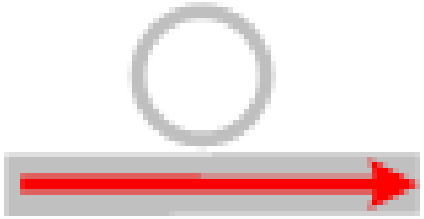
- Token-based protocols
  - Inspired by classic token ring
  - Token == transmission rights
  - Fits well with ring-shaped interconnect
  - Distributed, Scalable
  - (limited to ring)

# Baseline




- Based on traditional token protocols
- Repeat token at each node
  - But data is not repeated!
  - Poor utilization

# Optical arbitration basics

| Token - Inject | Token - Seize | Token - Pass |
| --- | --- | --- |
|  |  |  |

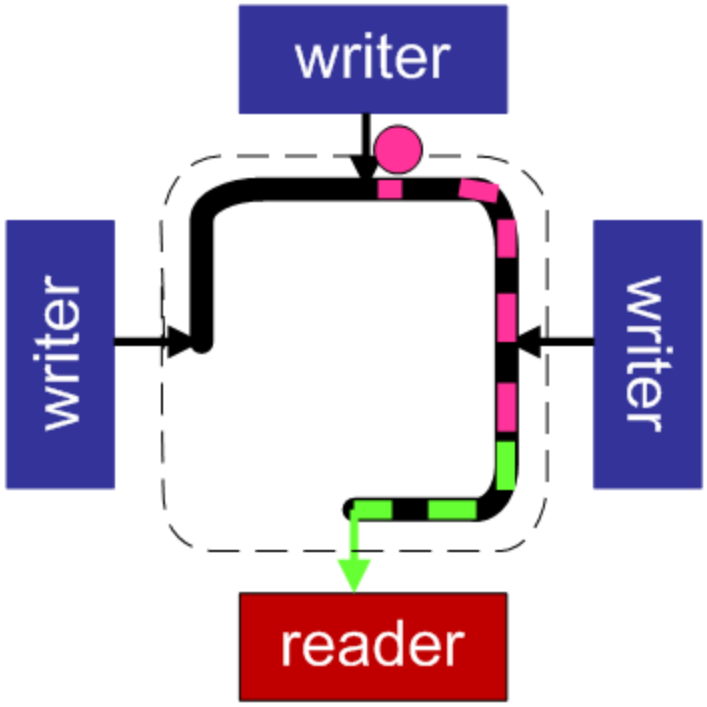


- No Repeat!
- Token latency bounded by the time of flight between requesters.

# Arbitration solutions

## Token Channel

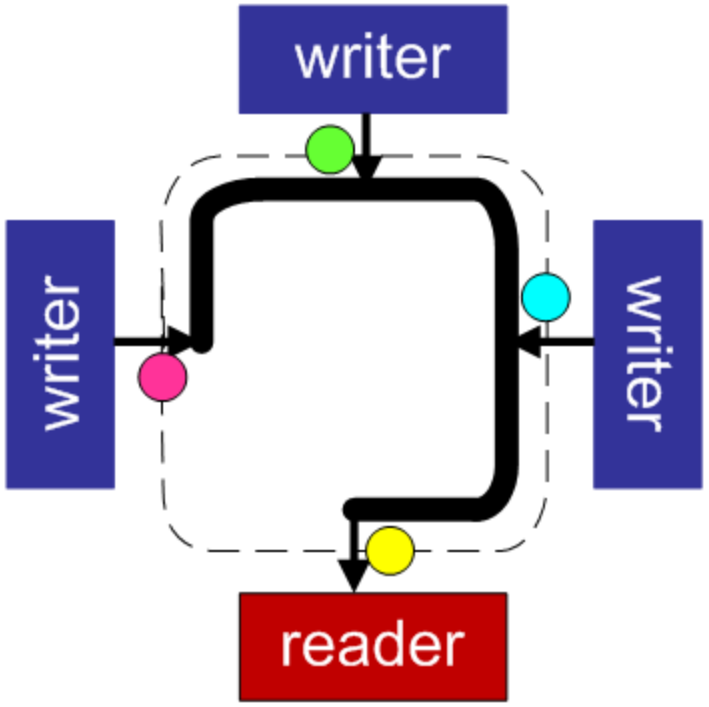
Single Token / Serial Writes




Token passing allows token to pace transmission tail (no bubbles)

## Token Slot

Multiple Tokens / Simultaneous Writes




Token passing allows token to directly precede slot

# Flow control and fairness

Flow Control:

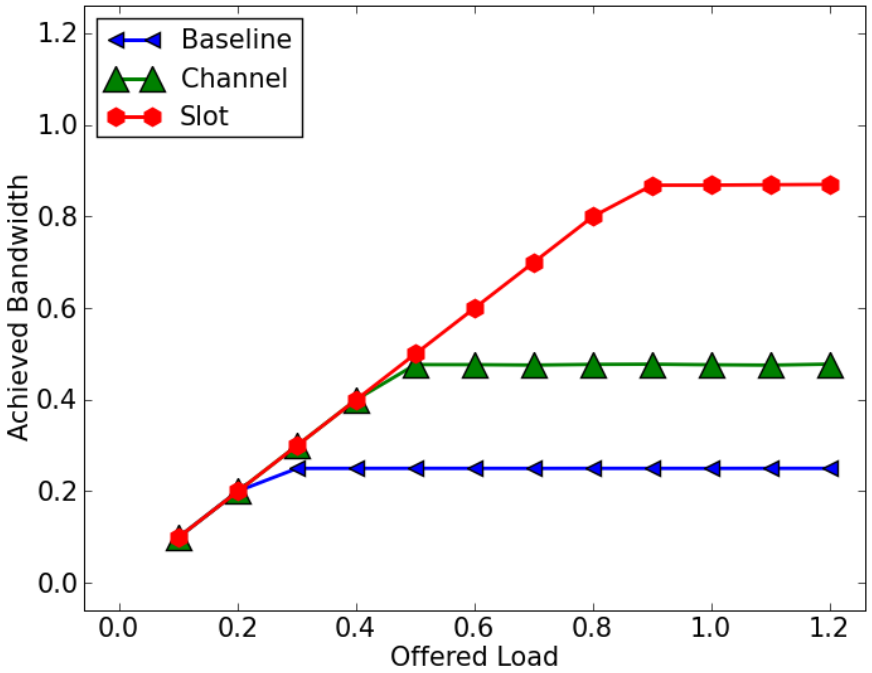
- Use token refresh as opportunity to encode flow control information (credits available)
- Arbitration winners decrement credit count

Fairness:

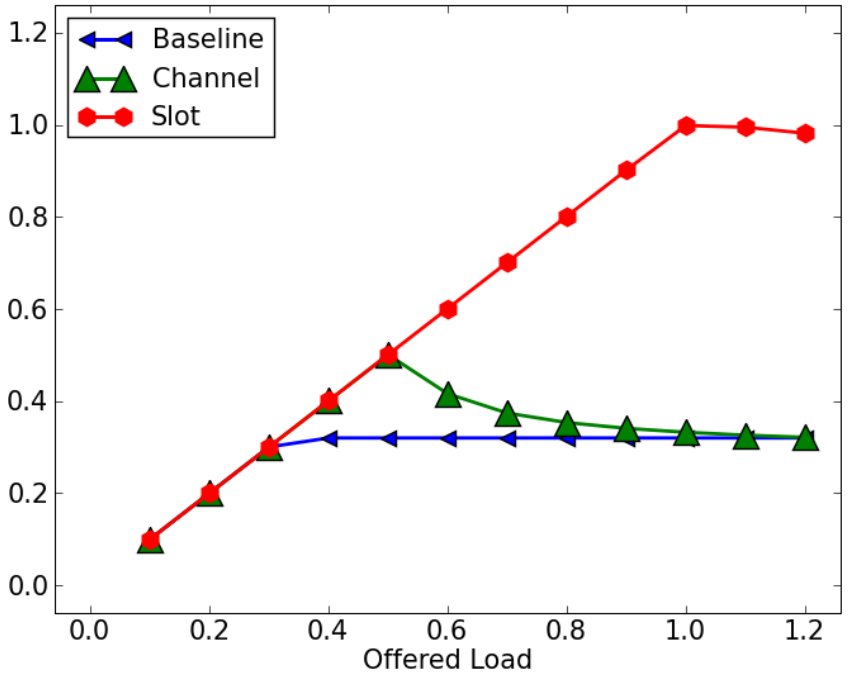
- Upstream nodes get first shot at tokens
- Need mechanism to prevent starvation of downstream nodes

# Results - Performance

Uniform




HotSpot

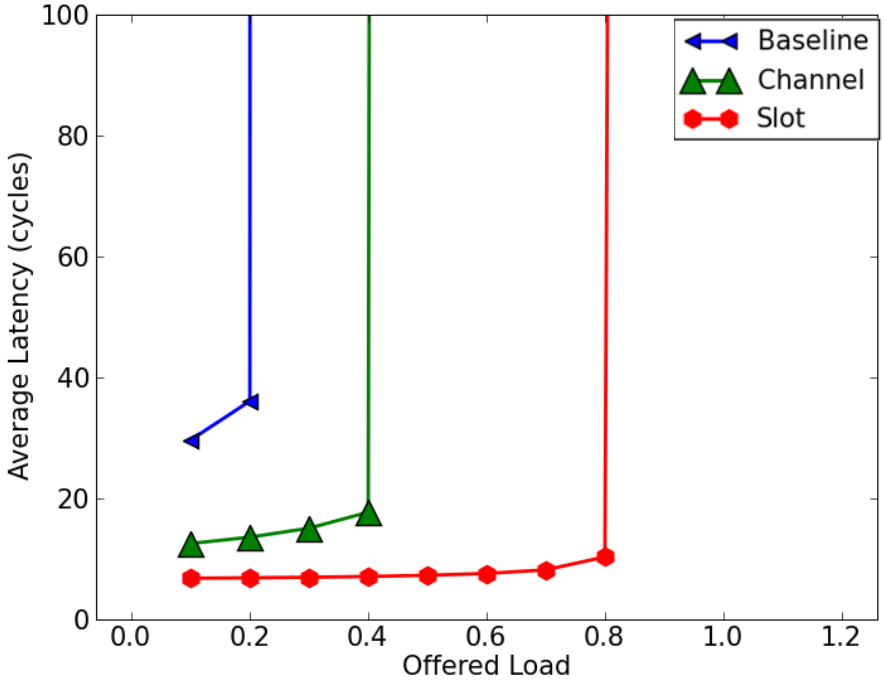



Token Slot benefits from

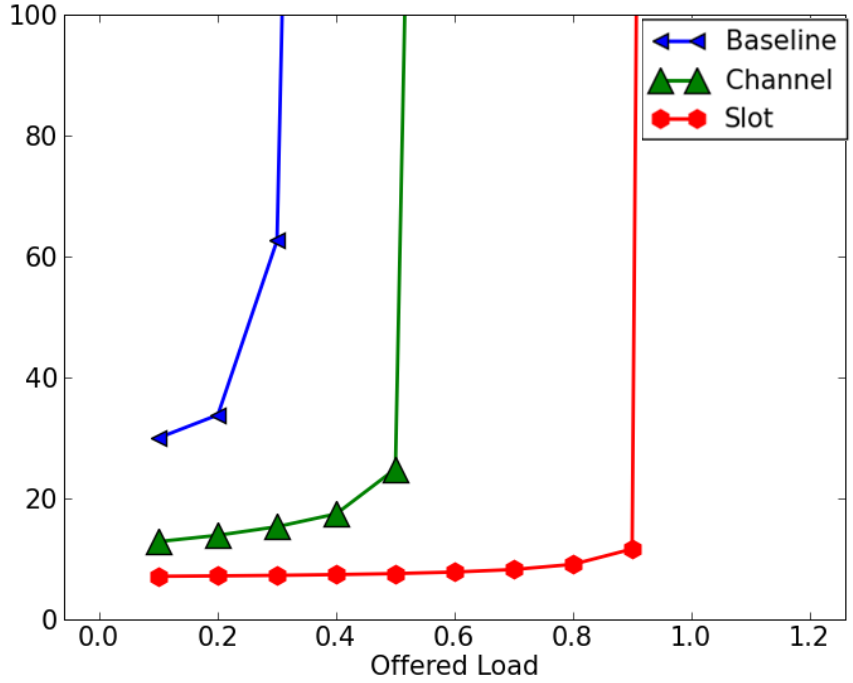
- the availability of multiple tokens (multiple writers)
- fast turn-around time of flow-control mechanism

# Results - Latency

Uniform




HotSpot




Token Slot has the lowest latency and saturates at 80%+ load

# Optical arbitration summary

- Arbitration speed has to match transfer speed for fine-grained communication
  - Arbiter has to be optical
- High throughput is achievable
  - 85+% for token slot
- Limited to simple topologies (MWSR)
- Implementation challenges
  - Opt-elec-logic-elec-opt in 200ps (@5GHz)

# Nanophotonics

☑Nanophotonics overview

☑ Sharing the nanophotonic channel

– Light-speed arbitration [MICRO 09]

- Utilizing the nanophotonic channel

– Atomic coherence [HPCA 11]

# What makes coherence hard?




Unordered interconnects
- split transaction buses, meshes, etc

Speculation
- Sharer-prediction, speculative data use, etc.

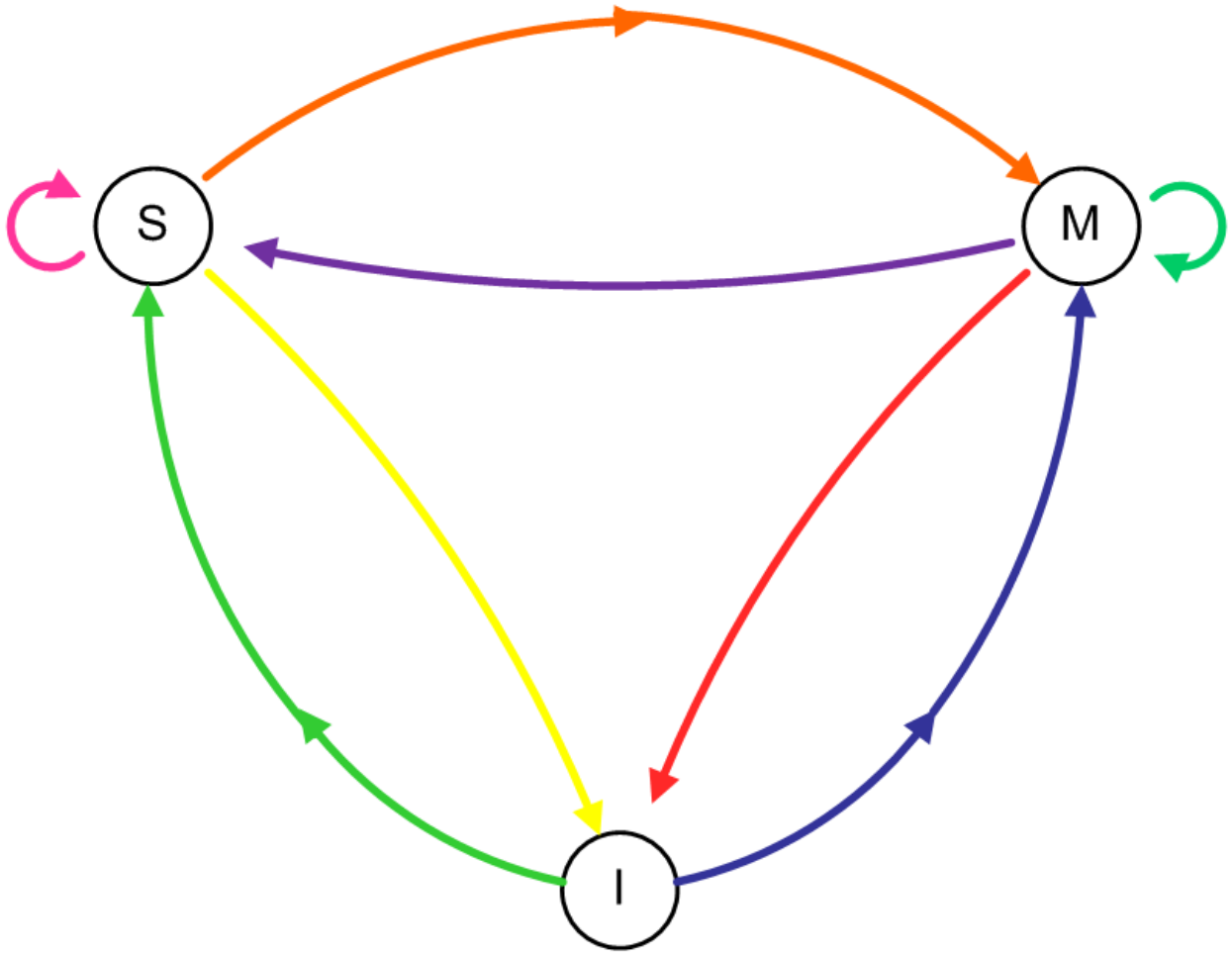
Multiple initiators of coherence requests
- L1-to-L2, Directory Caches, Coherence Domains, etc

→State-event pair explosion

- →Verification headache

# Example: MSI (SGI-Origin-like, directory, invalidate)

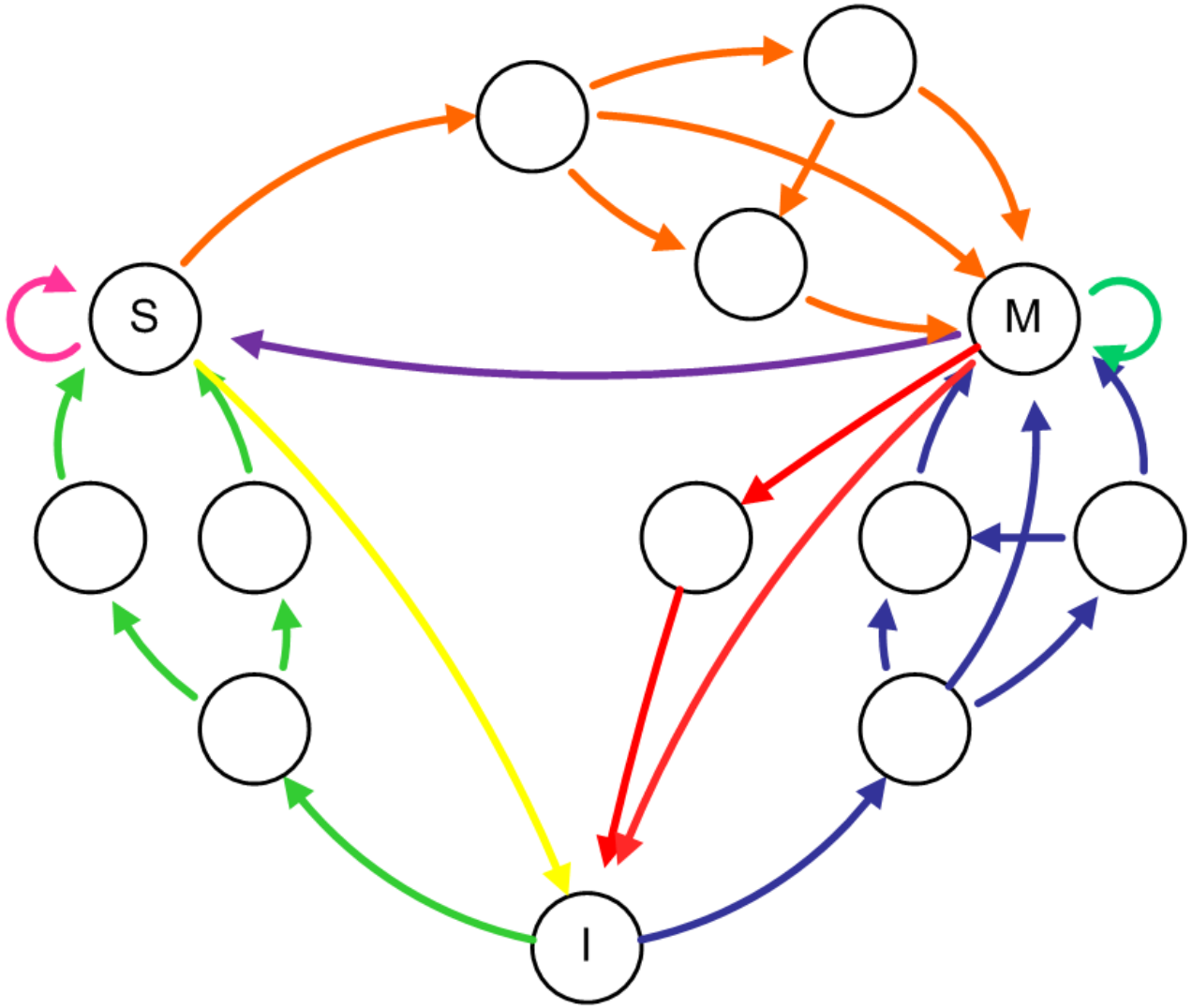
Stable States



# Example: MSI (SGI-Origin-like, directory, invalidate)

Stable States

Busy States

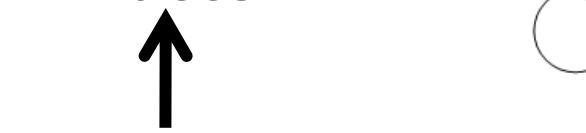


# Example: MSI (SGI-Origin-like, directory, invalidate)

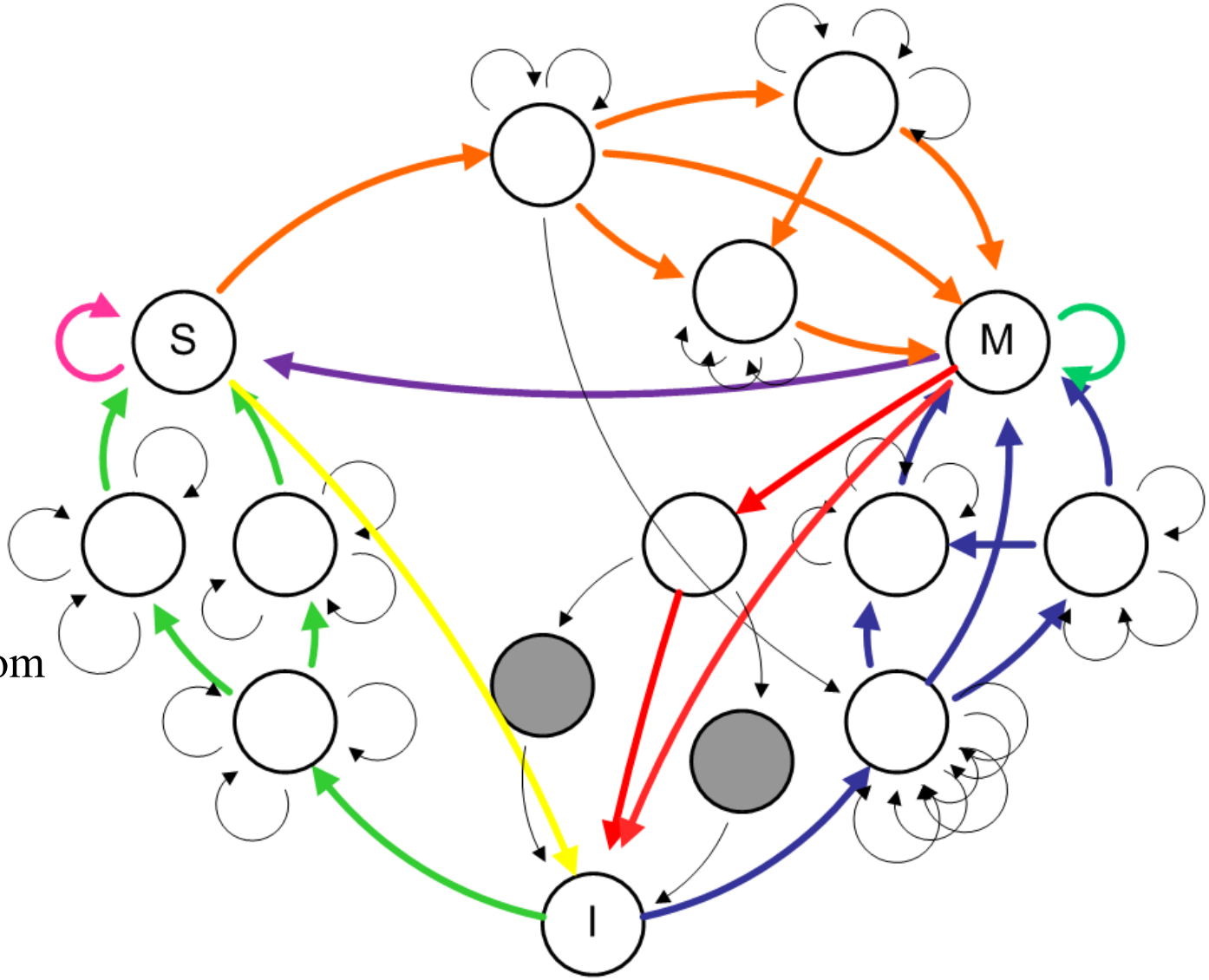
Stable States

Busy States

Races

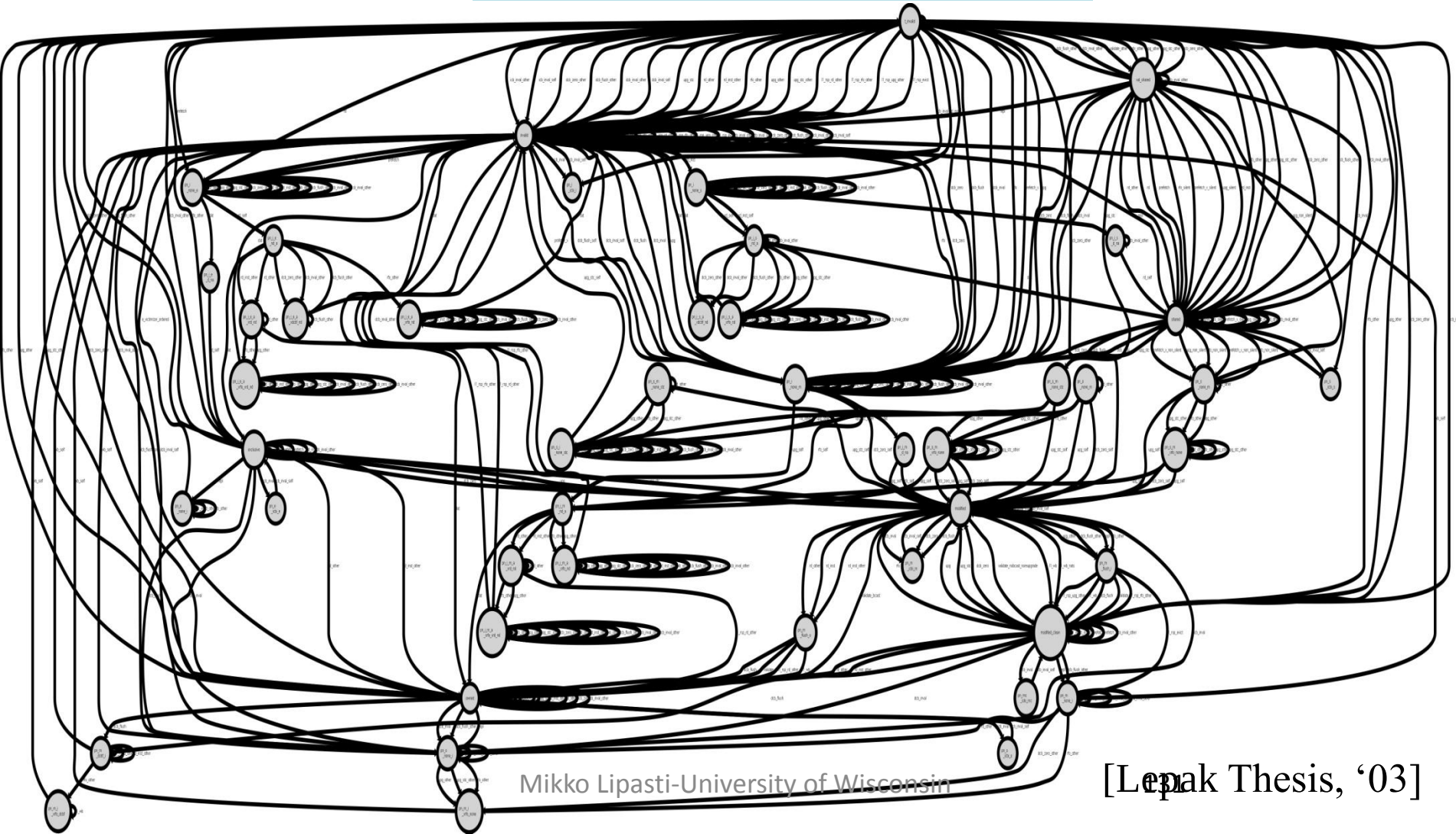


"unexpected" events from concurrent requests to same block



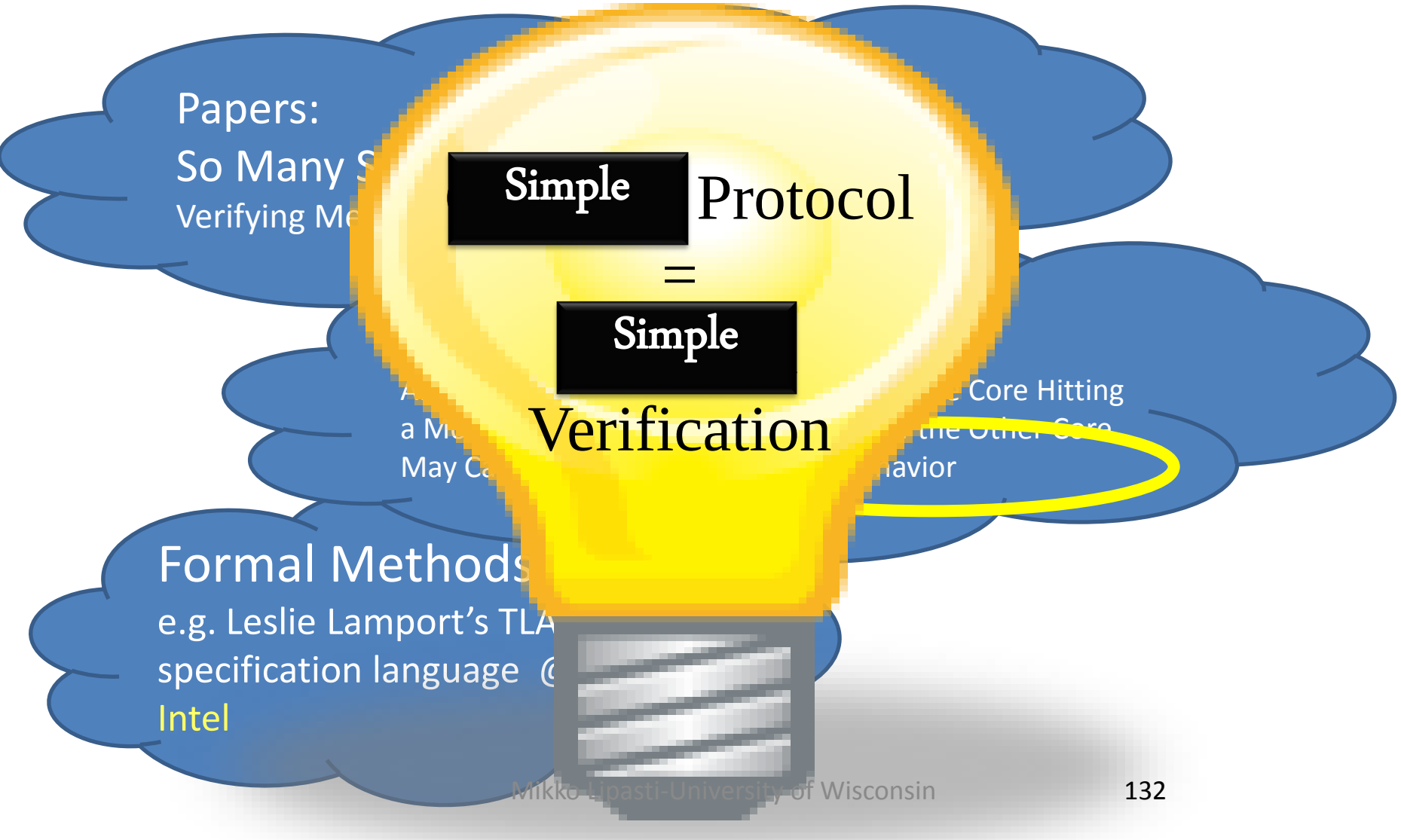
# Cache coherence complexity

## L2 MOETSI Transitions



[Lepak Thesis, '03]

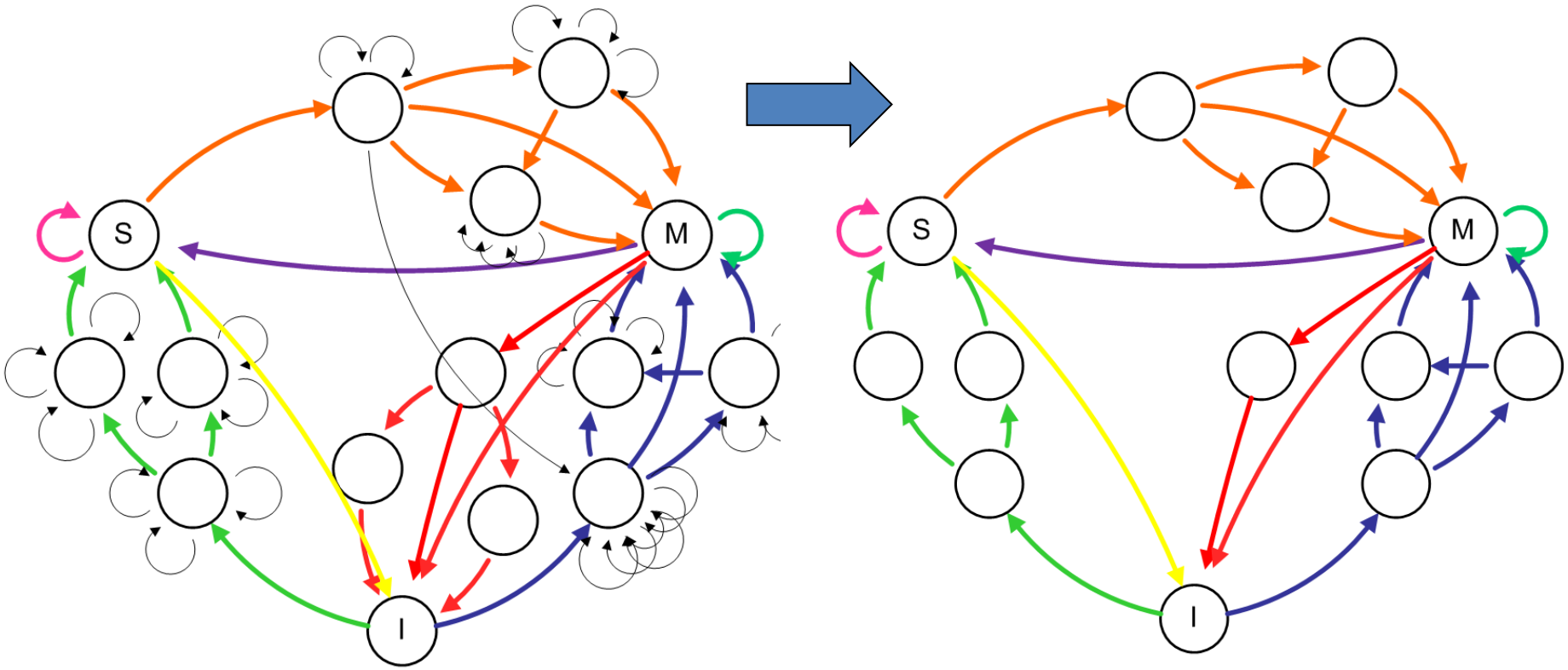
# Cache coherence verification headache



# Atomic Coherence: Simplicity

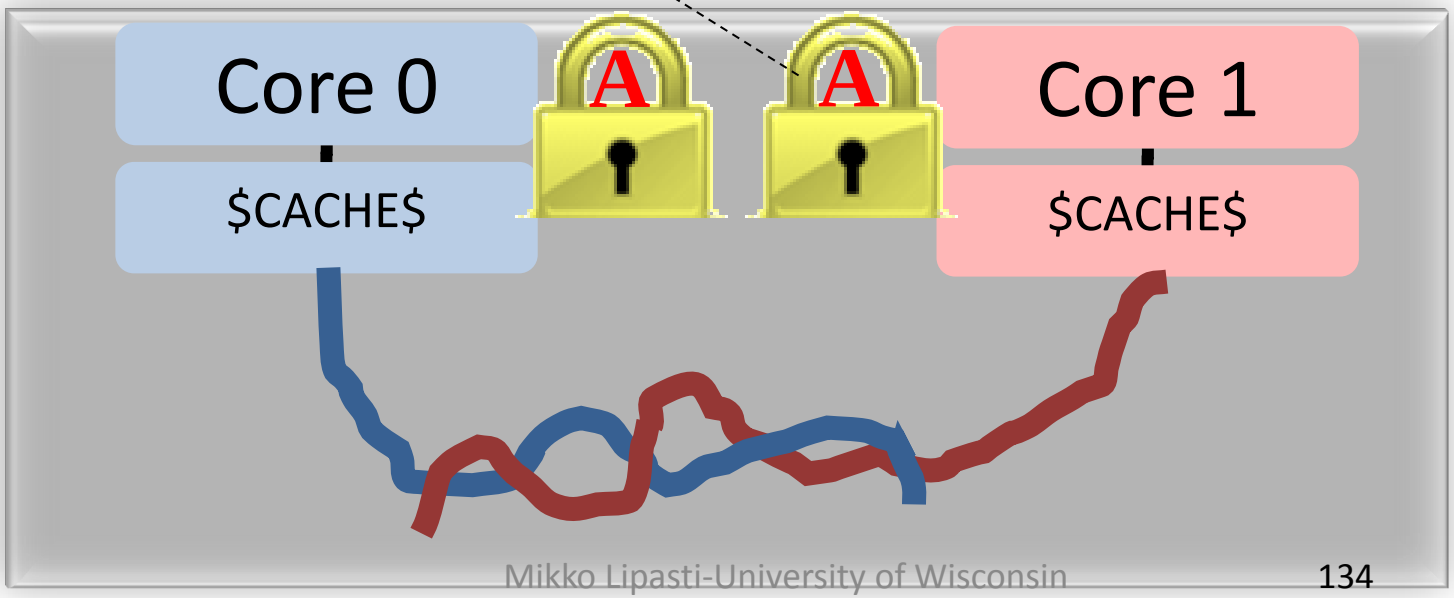
w/ races

w/o races

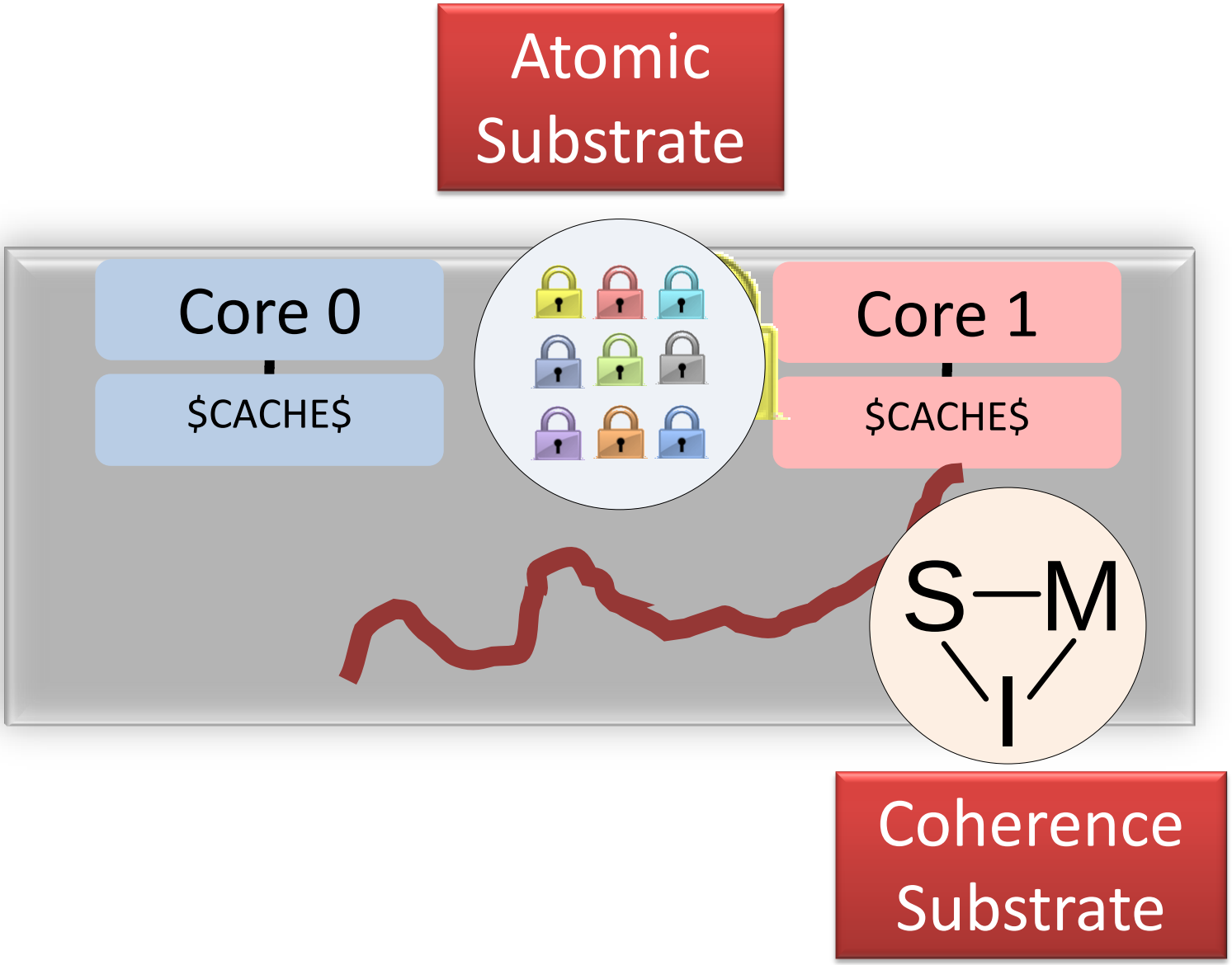


# Race resolution

- Cause:
- Concurrently active coherence requests to block A
- Remedy:
- ~~Only allow one~~ coherence request to block A to be active at a time.



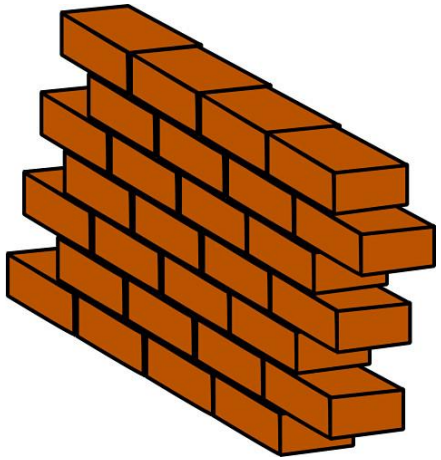
# Race resolution



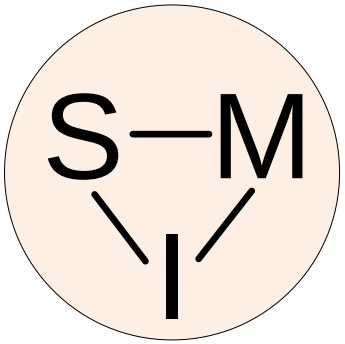
# Race resolution

Atomic Substrate





Coherence Substrate




-- Atomic Substrate is on critical path

\+ Can optimize substrates separately

# Atomic & Coherence Substrates

*aggressive*

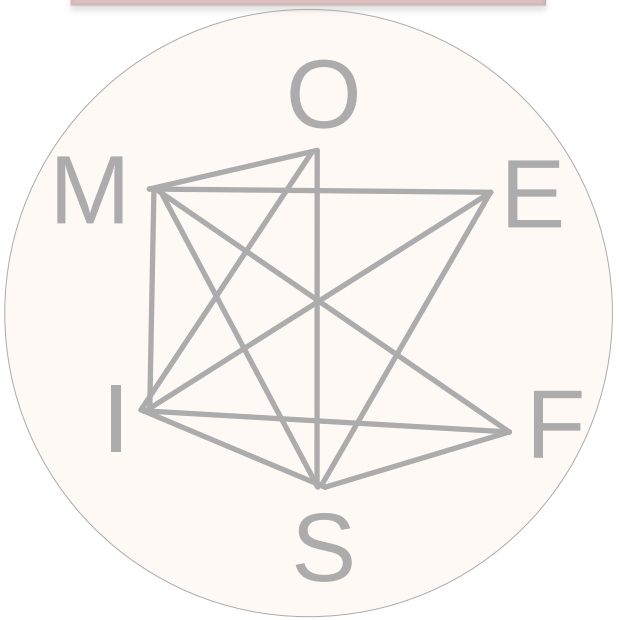
Atomic Substrate



(Apply Fancy Nanophotonics Here)

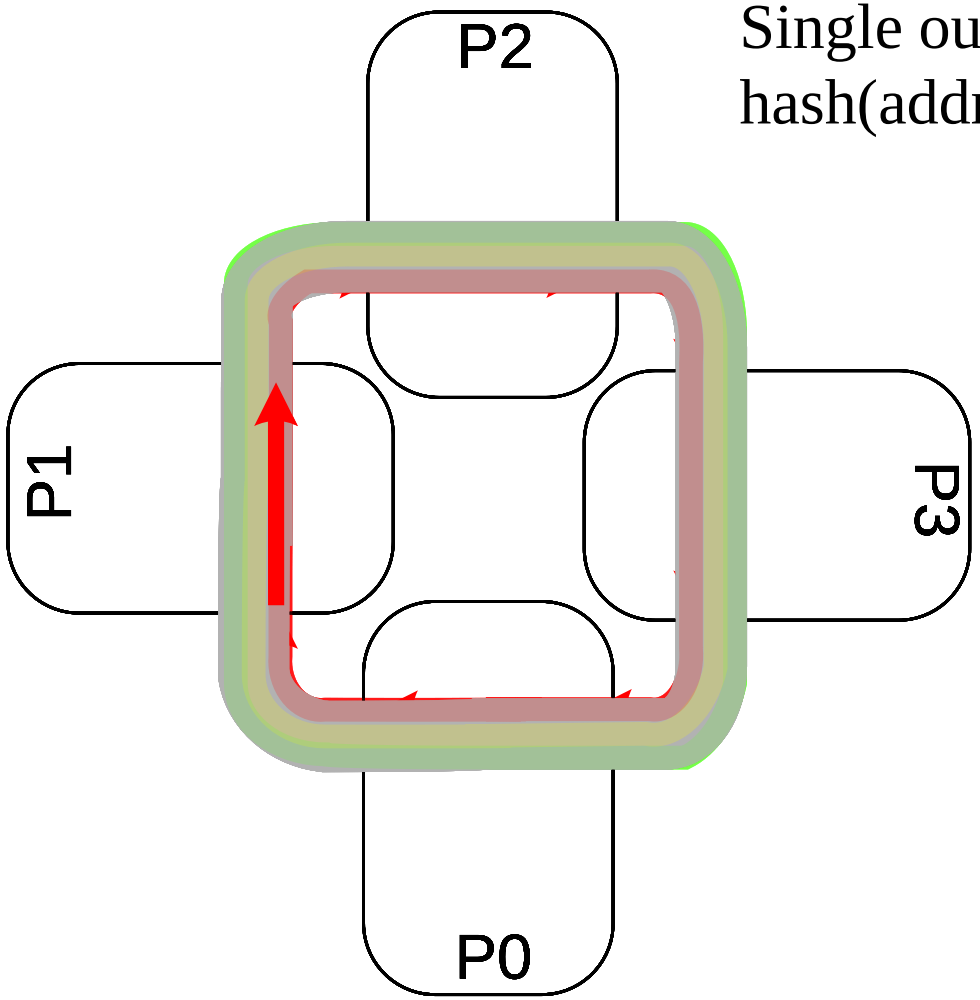
*aggressive*

Coherence Substrate



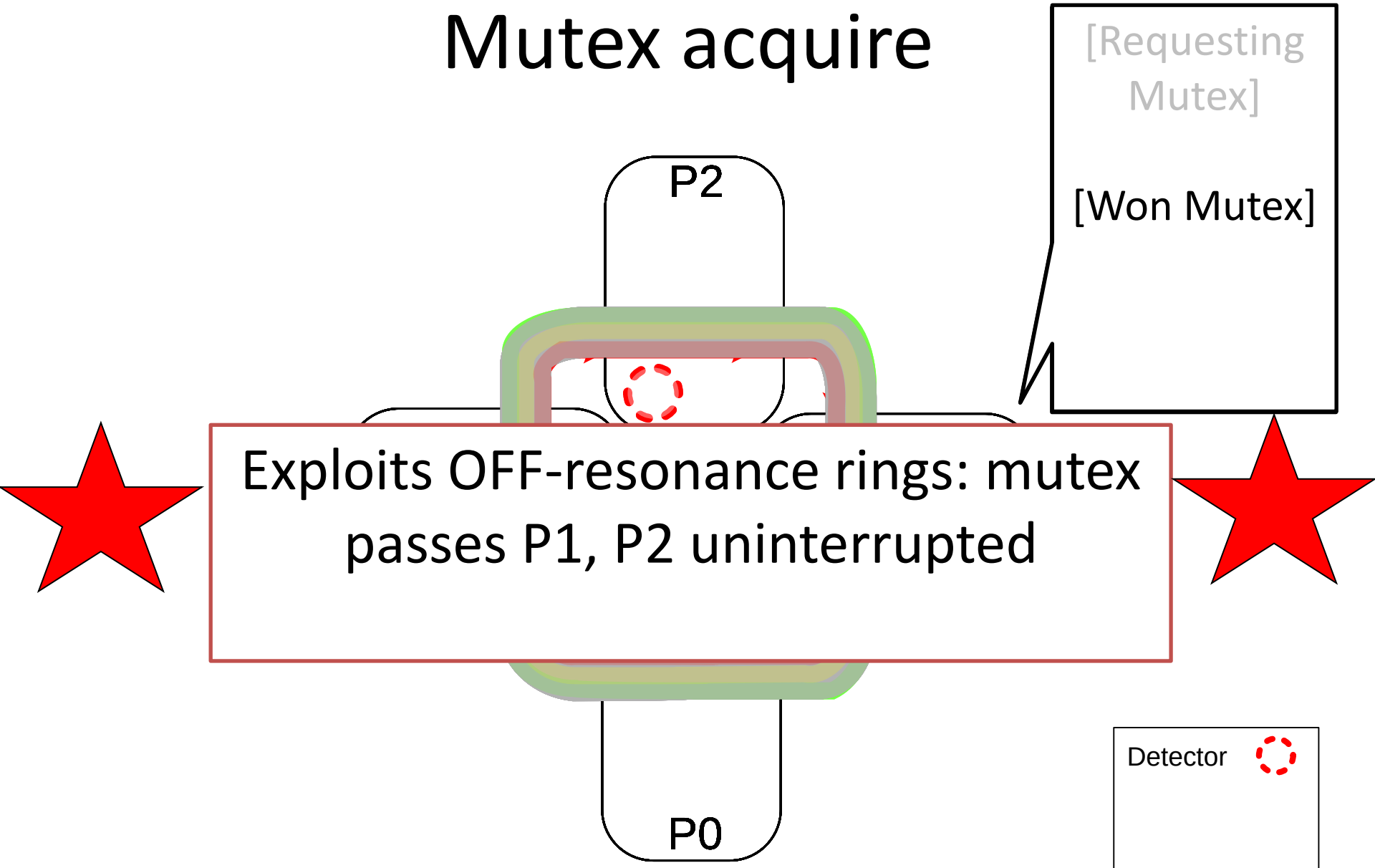

(Add speculation to a traditional protocol)

# Mutexes circulate on ring

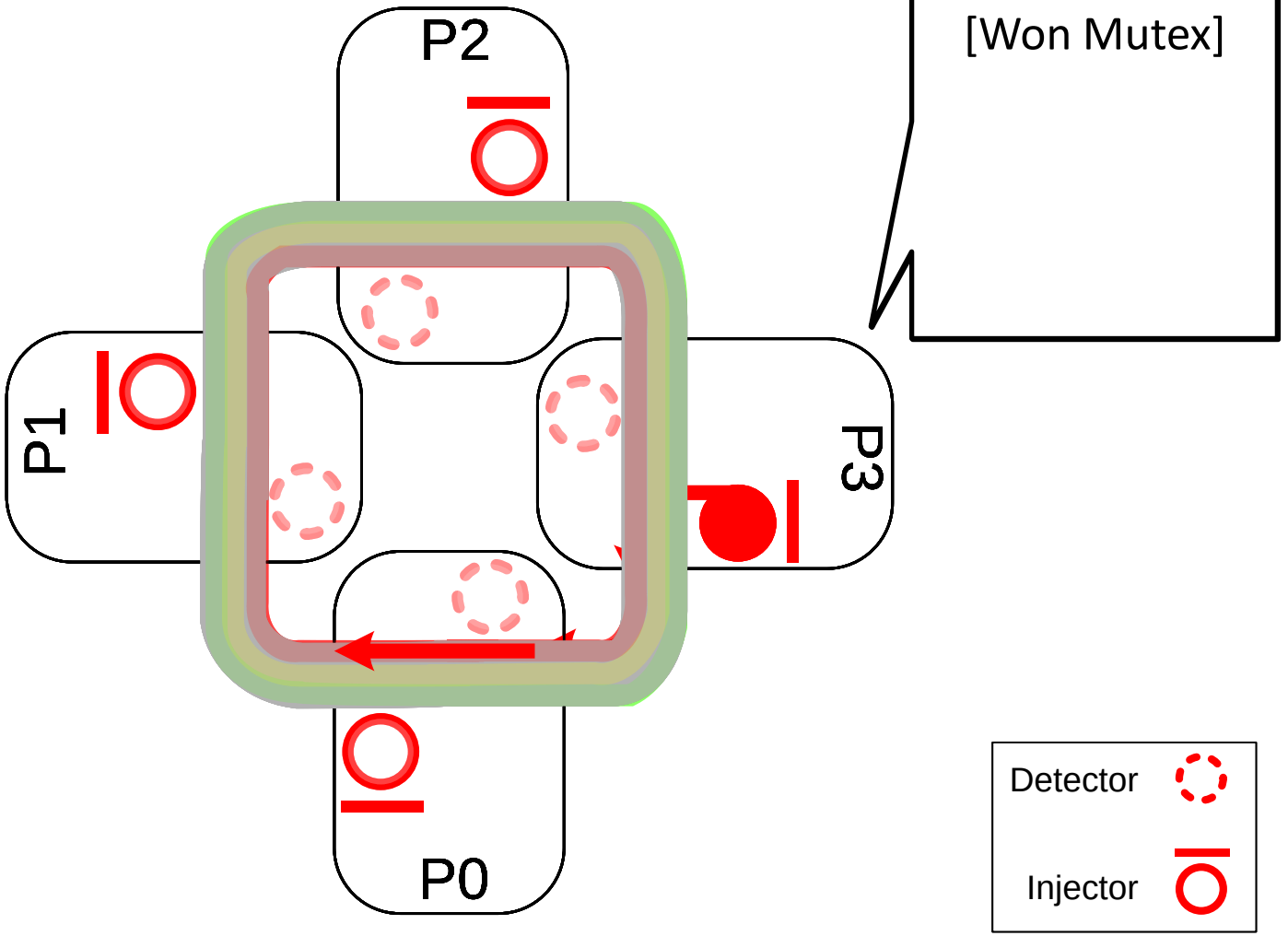



Single out mutex:
hash(addr *X*) →λ *Y* @ cycle *Z*

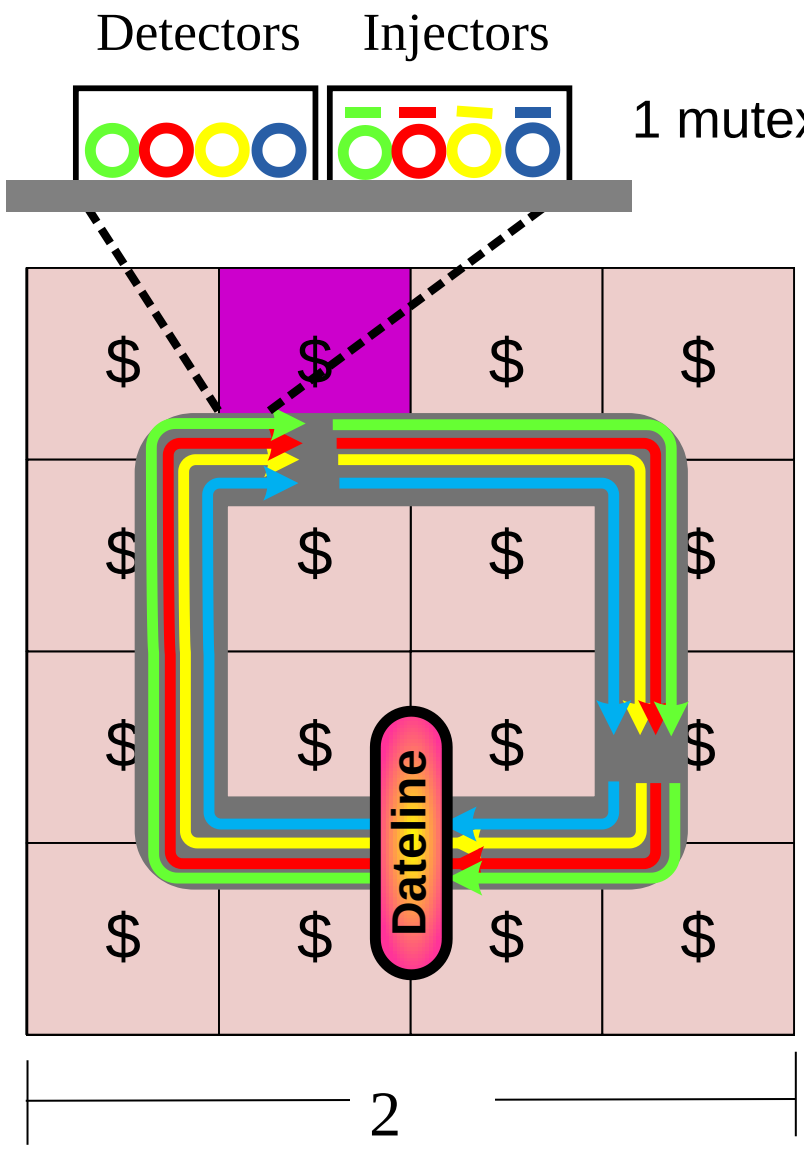
# Mutex acquire



# Mutex release



# Mutexes on ring




1 mutex = 200 ps = ~2cm = 1 cycle @ 5 GHz

# Mutex

$$= \frac{4mutex}{\lambda} \times \frac{64\lambda}{waveguide} \times 4waveguides$$

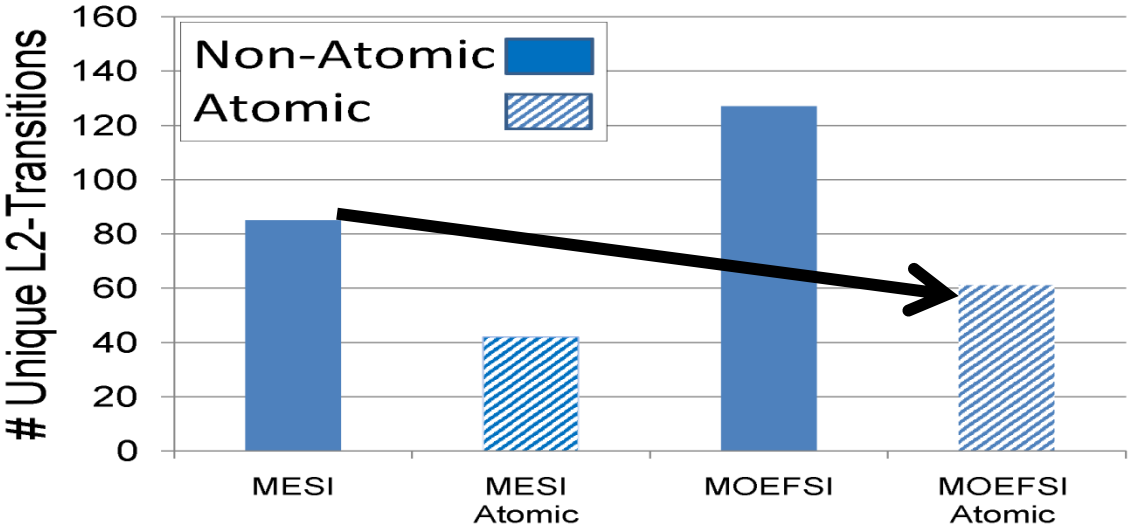
$$= 1024$$

Latency To:

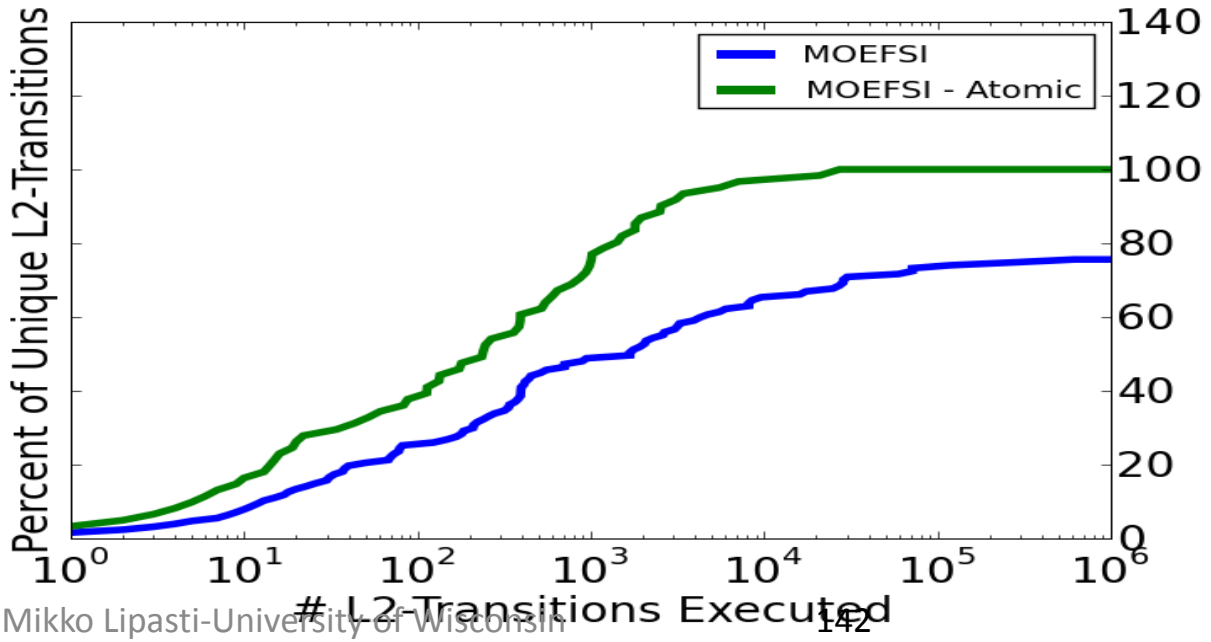
- seize free mutex : ≤ 4 cycles
- tune ring resonator: < 1 cycle

# Atomic Coherence: Complexity

Static:




Dynamic:
(random tester)

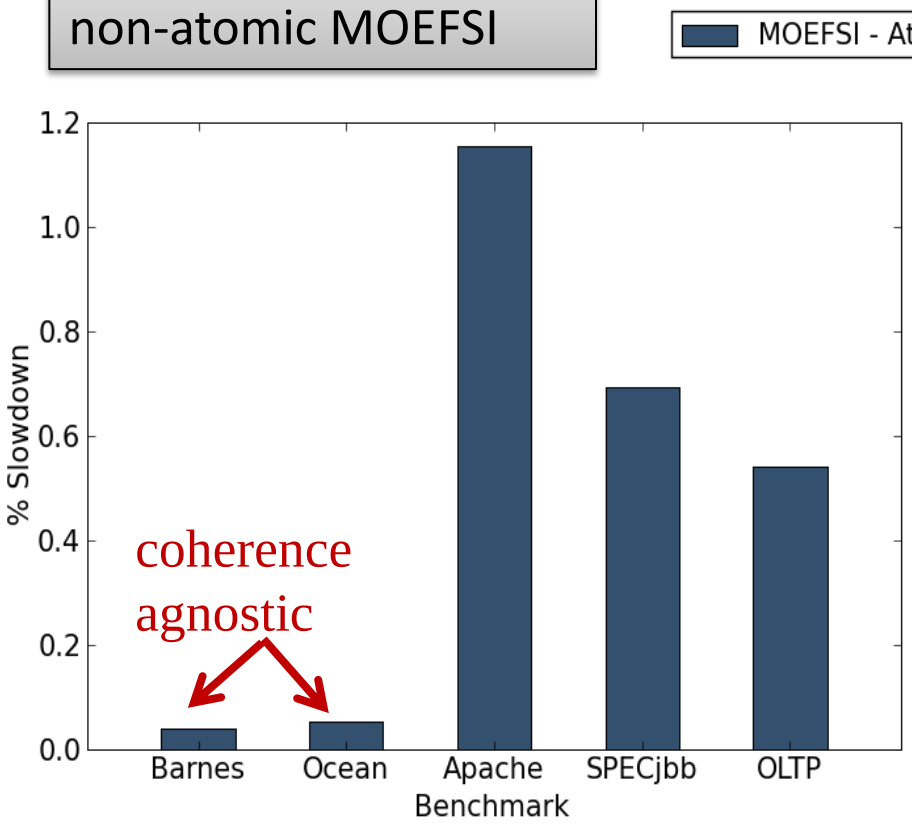



* Atomic Coherence reduces complexity

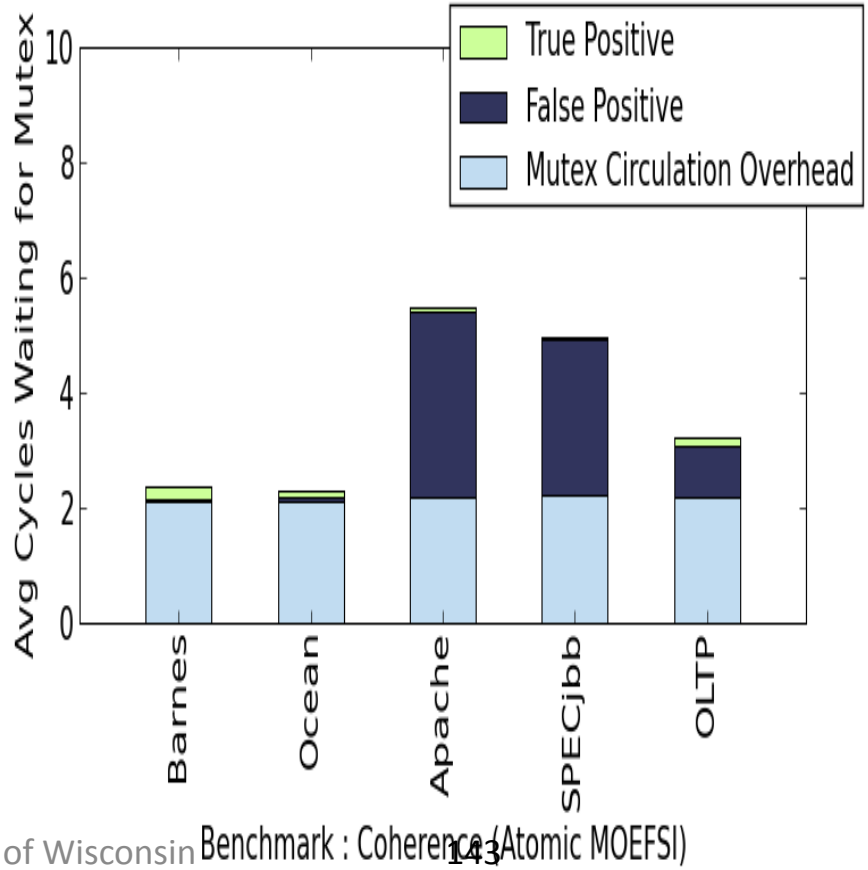
# Performance

(128 in-order cores, optical data interconnect, MOEFSI directory)

Slowdown relative to non-atomic MOEFSI




What is causing the slowdown?



# Optimizing coherence

**Observation:**

Holding Block B's mutex gives holder free reign over coherence activity related to block B

**`O.wned`** and **`F.orward`** State:

- Responsible for satisfying on-chip read misses

Opportunity:

- Try to keep `O`/`F` alive
- If `O` (or `F`) block evicted:

  While mutex is held, 'shift' `O`/`F` state to sharer

(or hand-off responsibility)

# Optimizing coherence

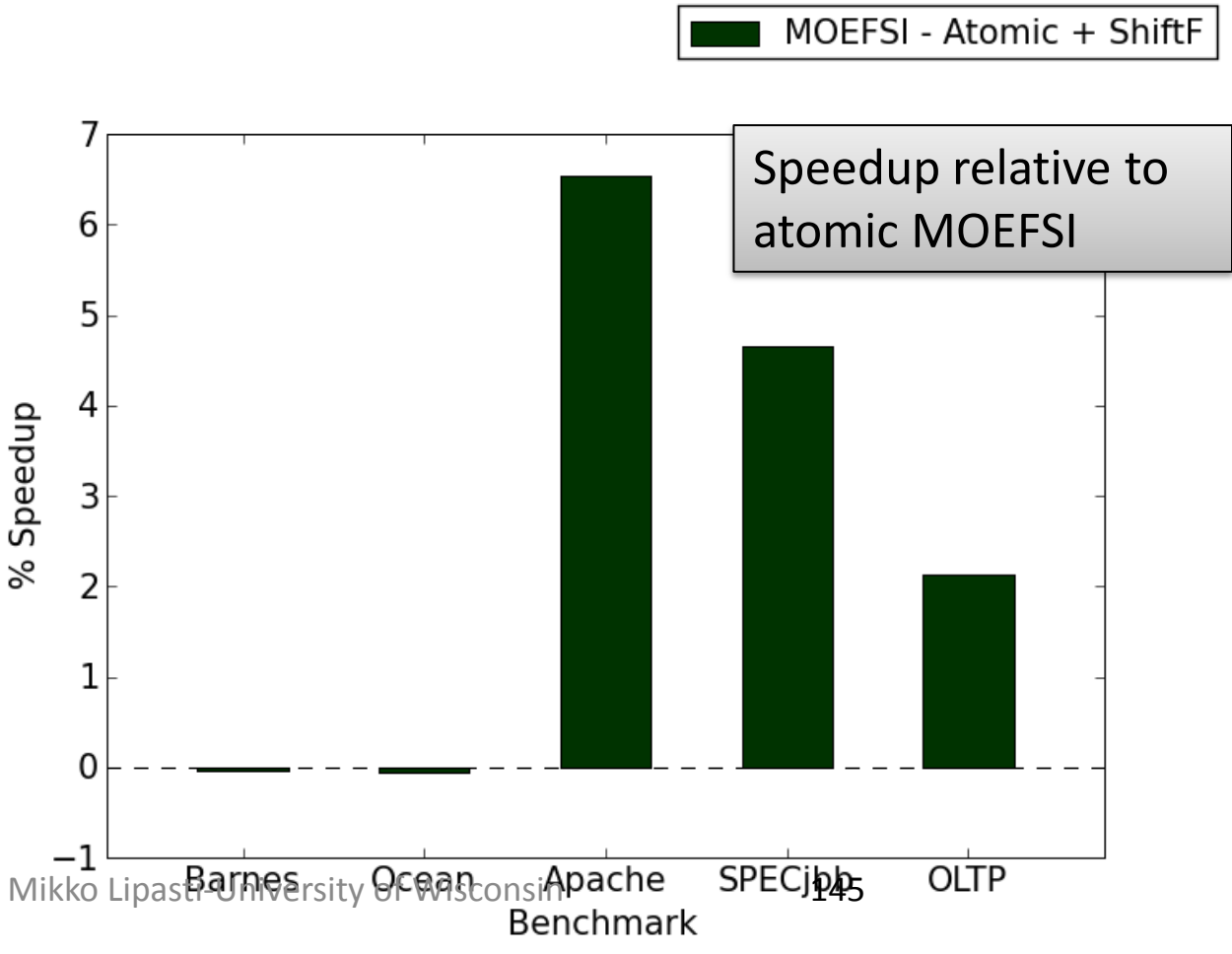
• If O (or F) block evicted: 'Shift' O/F state to sharer

*Complexity:*

# L2 transitions

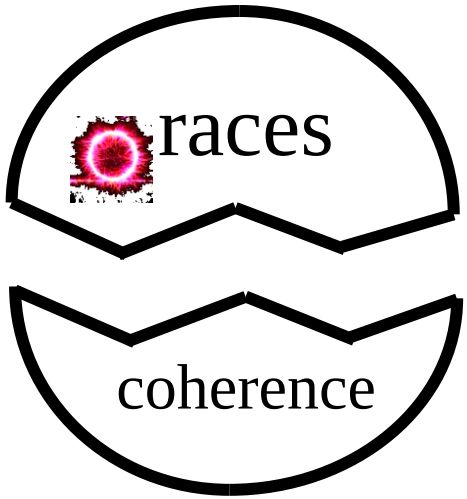
(b/c less variety in sharing possibilities)

*Performance:*



# Atomic Coherence Summary

- Nanophotonics as enabler
  - Very fast chip-wide consensus
- Atomic Protocols are simpler protocols
  - And can have minimal cost to performance (w/ nanophotonics)
  - Opportunity for straightforward protocol enhancements: ShiftF
- More details in HPCA-11 paper
  - Push protocol (update-like)



# Nanophotonics

- Nanophotonics overview
- Sharing the nanophotonic channel
  - Light-speed arbitration [MICRO 09]
- Utilizing the nanophotonic channel
  - Atomic coherence [HPCA 11]

# Nanophotonics References

- References:
  - Vantrease et al., “Corona…”, ISCA 2008
  - Vantrease et al., “Light-speed arbitration…”, MICRO 2009
  - Vantrease et al., “Atomic Coherence,” HPCA 2011
  - Dana Vantrease Ph.D. Thesis, Univ. of WI 2010

# Lecture Outline

- Introduction to Networks
- Network Topologies
- Network Routing
- Network Flow Control
- Router Microarchitecture
- Technology example: On-chip Nanophotonics

# Readings

- D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. F. B. III, and A. Agarwal. On-Chip Interconnection Architecture of the Tile Processor. IEEE Micro, vol. 27, no. 5, pp. 15-31, 2007
- D. Vantrease et al. Corona: System Implications of Emerging Nanophotonic Technology. Proceedings of ISCA-35, Beijing, June 2008