High-Performance On-Chip Interconnects for Emerging SoCs

http://tusharkrishna.ece.gatech.edu/teaching/nocs_acaces17/

ACACES Summer School 2017

Lecture 2: Routing

Tushar Krishna

Assistant Professor
School of Electrical and Computer Engineering
Georgia Institute of Technology

tushar@ece.gatech.edu

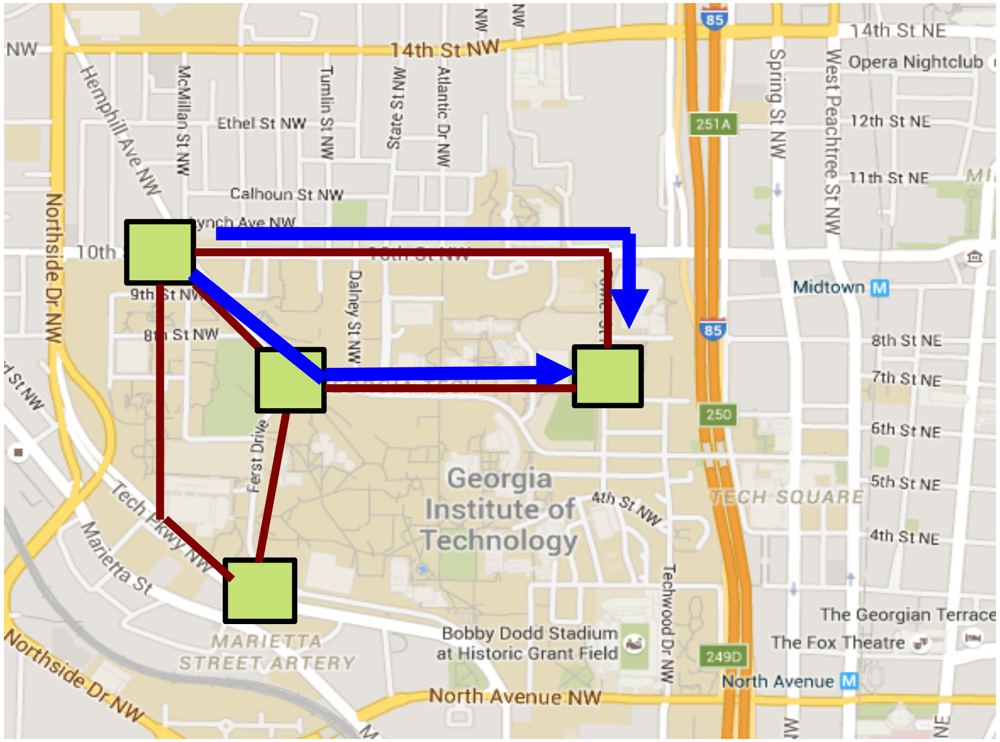
Acknowledgment: Some slides adapted from Univ of Toronto ECE 1749 H (N Jerger) and MIT 6.883 (L-S Peh)

Network Architecture

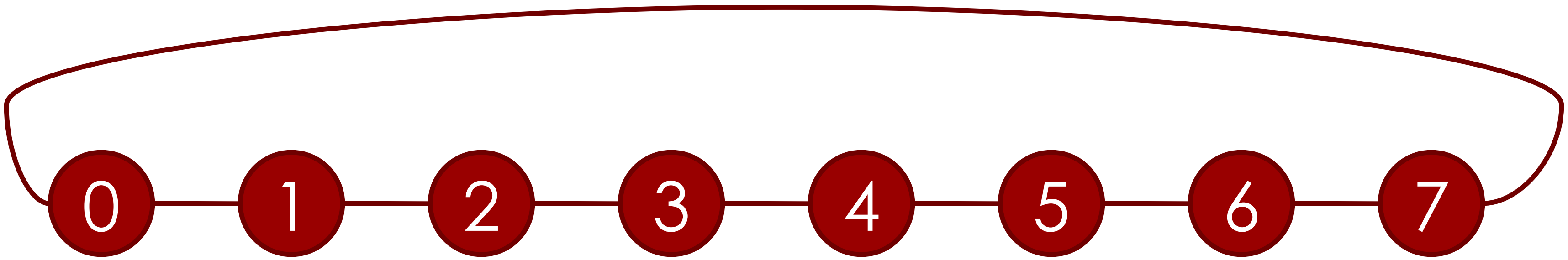
- **Topology**
  - How to connect the nodes
  - ~Road Network
- **Routing**
  - Which path should a message take
  - ~Series of road segments from source to destination
- **Flow Control**
  - When does the message have to stop/proceed
  - ~Traffic signals at end of each road segment
- **Router Microarchitecture**
  - How to build the routers
  - ~Design of traffic intersection (number of lanes, algorithm for turning red/green)

Routing

- Once topology is fixed, routing determines exact path from source to destination
- Analogous to the series of road segments from source to destination

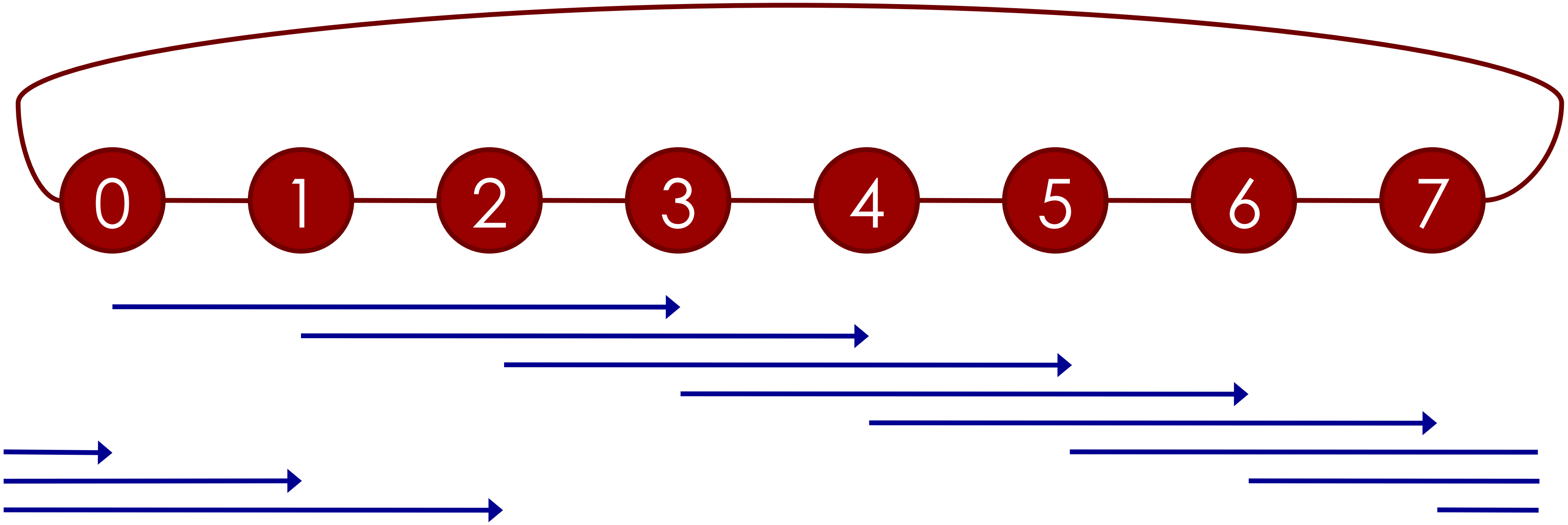


Why does Routing matter?



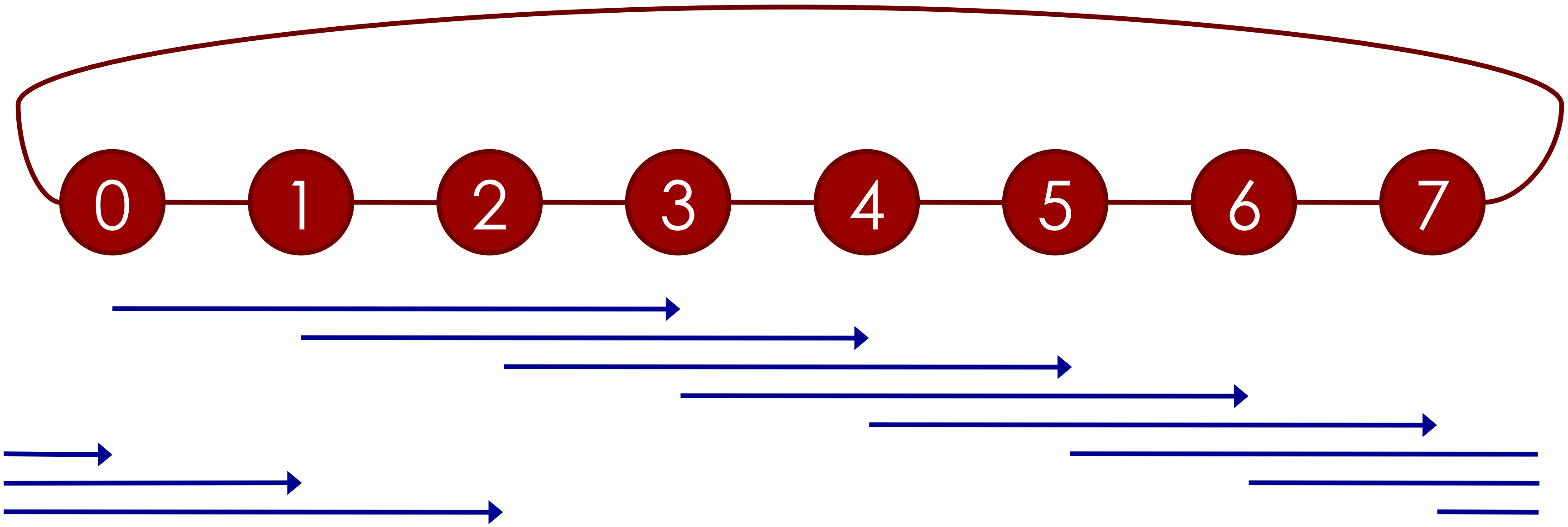
- Suppose three routing options
  - **Greedy:** shortest path
  - **Random:** randomly pick direction
  - **Adaptive:** monitor load in each direction and send
- Which routing algorithm is the best?
  - Depends …what is the traffic pattern?
  - What metric (latency/throughput/energy) do we care about?

Suppose Traffic = Tornado



- k-ary n-cube, $node_i \rightarrow node_{(i + (k/2) - 1) \bmod k}$
  - Here k = 8, $node_i \rightarrow node_{i+3 \bmod 8}$

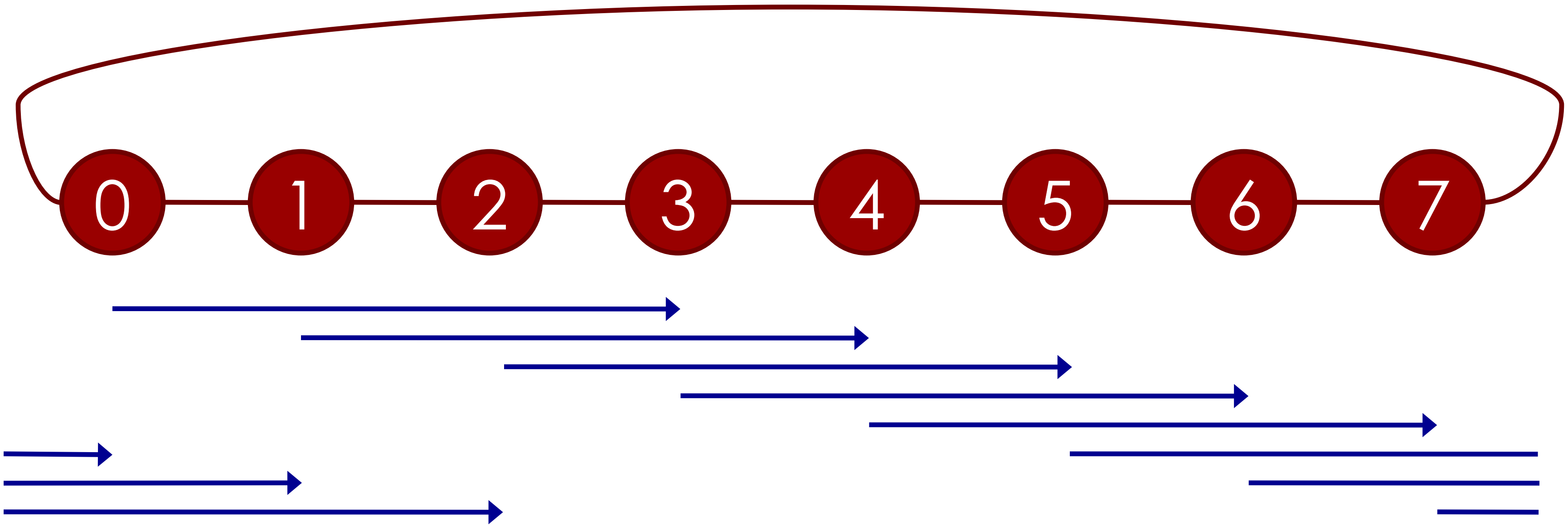
Metric = Zero-load Latency



- Best routing algorithm?

	Greedy	Random	Adaptive
Hops	3	(3+5)/2 = 4	3 at low-loads

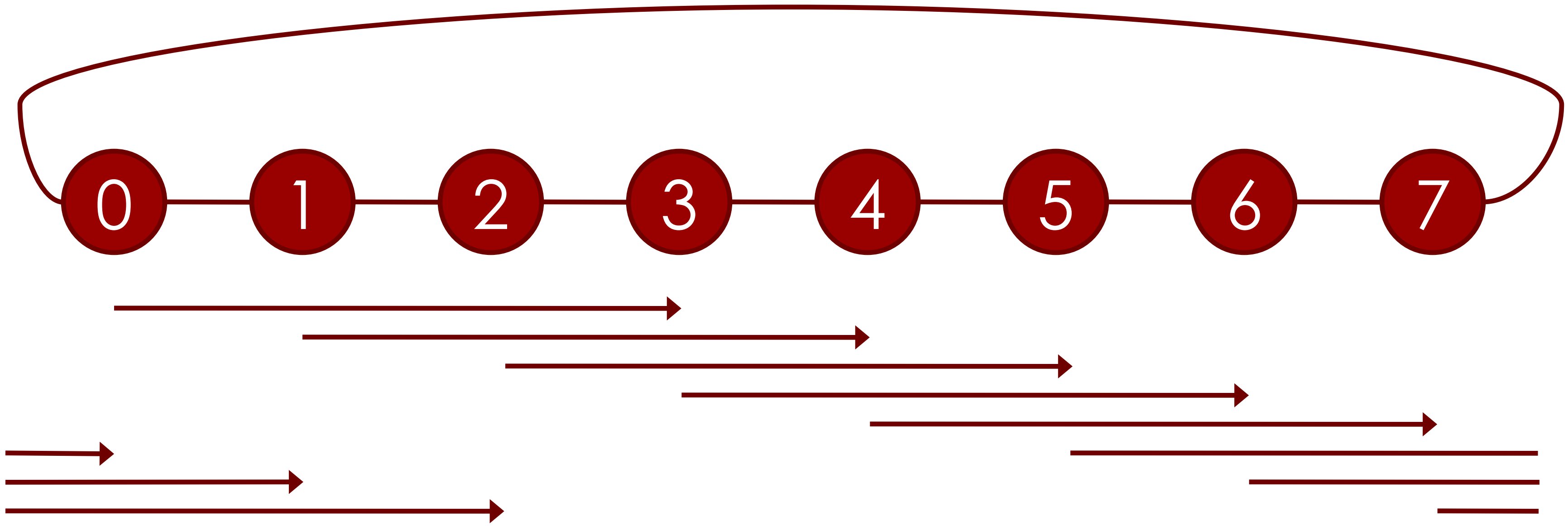
Metric = Energy



- Best routing algorithm?

	Greedy	Random	Adaptive
Hops	3	(3+5)/2 = 4	3 at low-loads

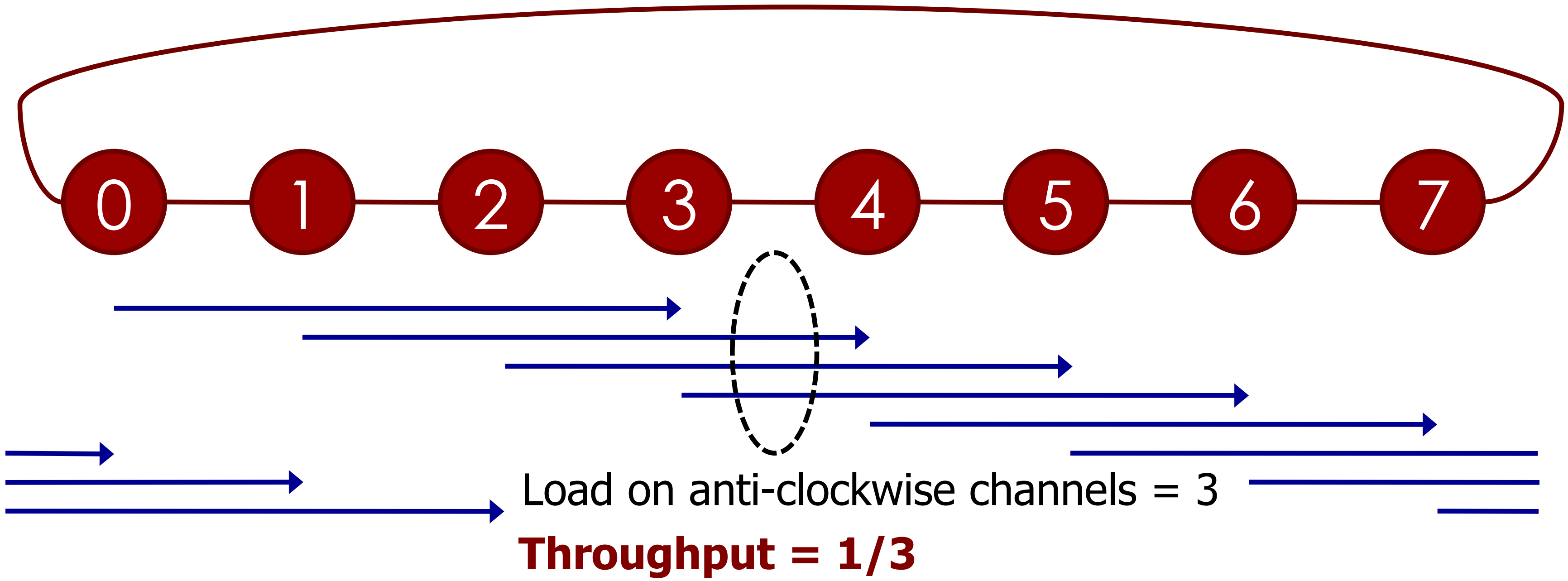
Metric = Throughput



- Best routing algorithm?

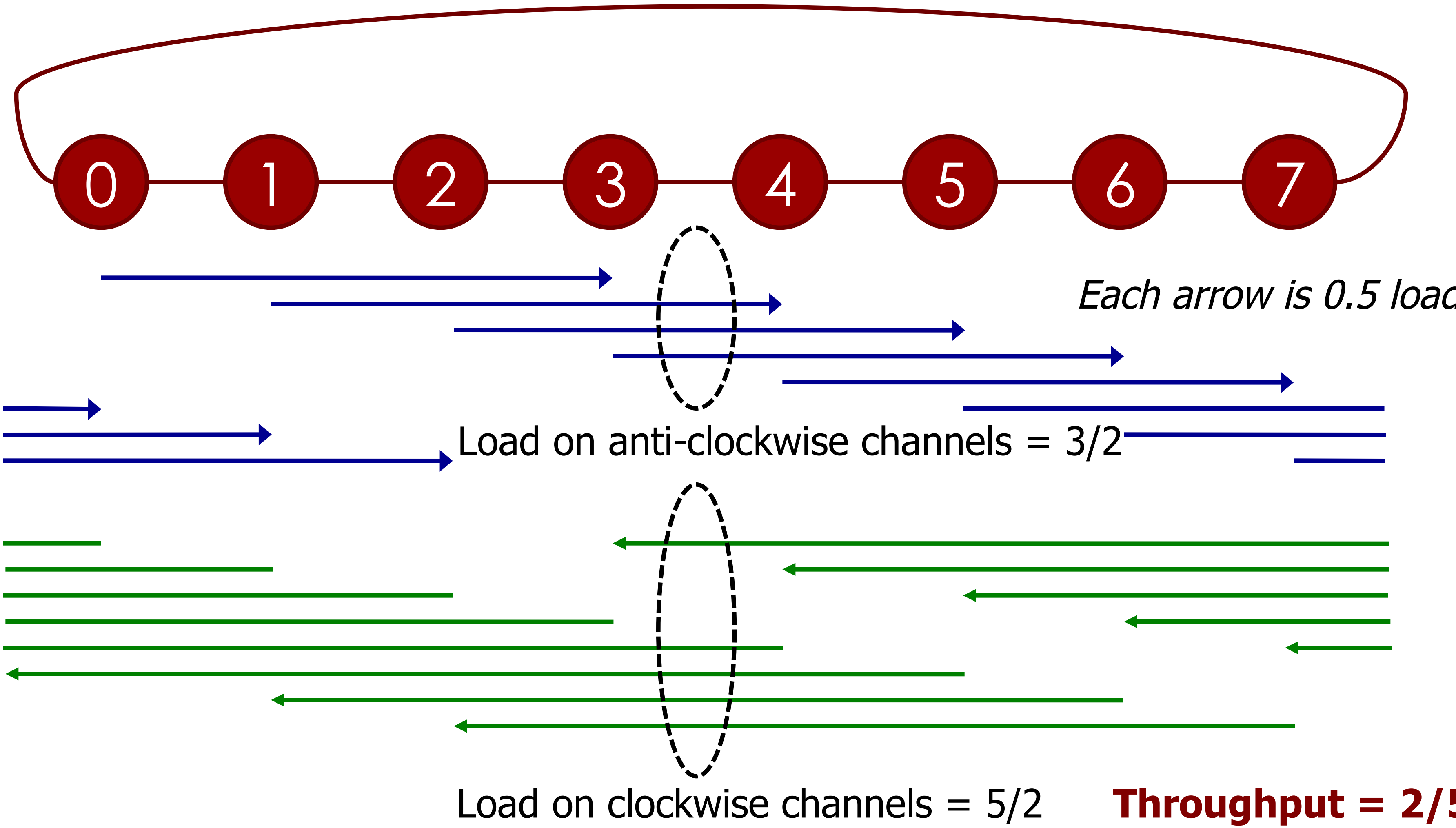
	Greedy	Random	Adaptive
Max Channel Load			
Throughput			

Channel Load for Greedy Traffic



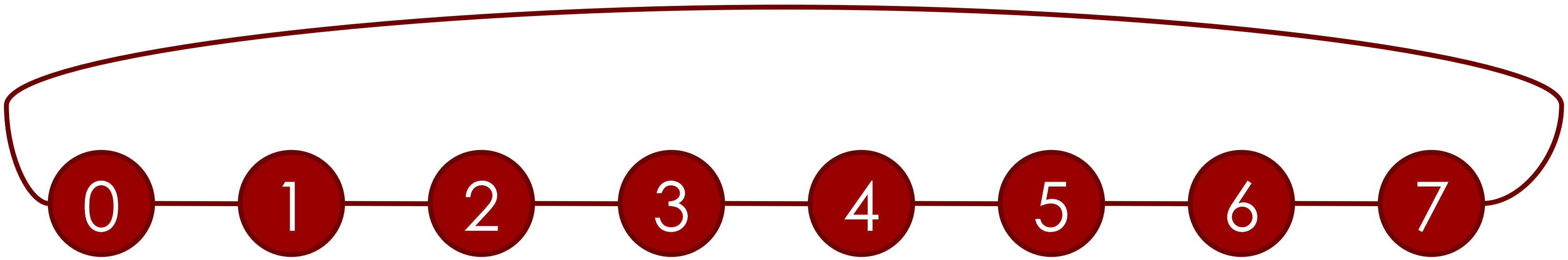
- All traffic moves anti-clockwise
  - Clockwise channels are idle

Channel Load for Random Traffic



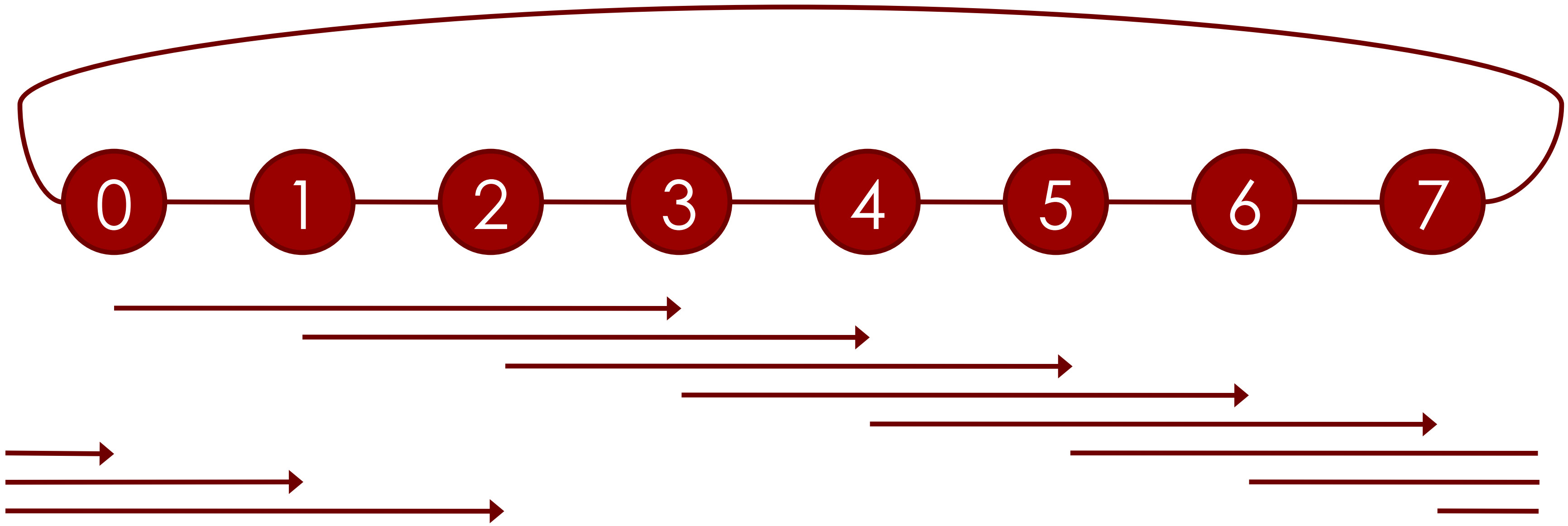
Throughput = 2/5

Channel Load for Adaptive Traffic



- Assume ideal implementation
- For equal load on both anti-clockwise and clockwise links, suppose each node sends a fraction f anticlockwise, and (1-f) clockwise
  - Channel Load = 3f = 5(1-f)
    - f = 5/8
    - Send 5/8th traffic anticlockwise, 3/8th traffic clockwise
  - Channel Load = 15/8, Throughput = 8/15

Metric = Throughput



- Best routing algorithm?

	Greedy	Random	Adaptive
Max Channel Load	3	5/2 = 2.5	15/8= 1.875
Throughput	1/3 = 0.33	2/5 = 0.4	8/15 = 0.53

Routing

- **Taxonomy**
- Deadlocks
- Conclusion

Taxonomy of Routing Algorithms

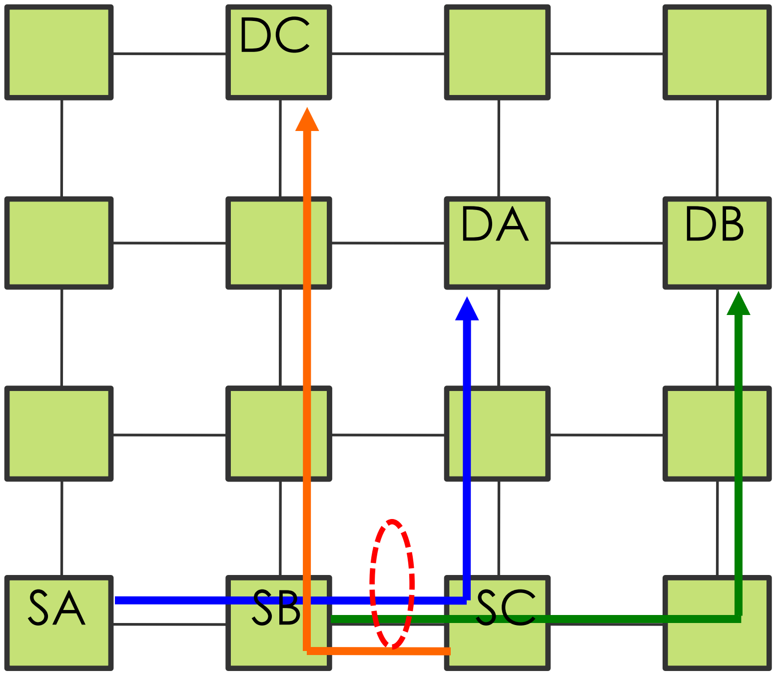
- **Classification I: path length**
  - **Minimal**: shortest paths
    - Example: Greedy over Ring
  - **Non-minimal**: non-shortest paths
    - Example: Random and Adaptive over Ring

Taxonomy of Routing Algorithms

- **Classification II: path diversity** (how to select between the set of all possible paths R_{xy} from the source x to the dest y)
  - **Deterministic:** always choose the same route between x and y, even if $|R_{xy}| > 1$
    - **Example:** Greedy over Ring
    - Most restrictive but **most popular** due to ease of implementation and analysis
  - **Oblivious:** choose any of the routes in R_{xy} without considering any information about current network state (i.e., congestion)
    - **Example:** Random over Ring
    - Deterministic are a subset of oblivious
  - **Adaptive:** choose one of the routes in R_{xy} depending on the current network state (i.e., congestion)
    - **Example:** Adaptive over Ring
    - Congestion Metrics: link availability, buffer occupancy, history of channel load

Dimension-Ordered Routing (DOR) in a Mesh/Torus

XY Routing: Always go X first, then Y



Minimal and Deterministic

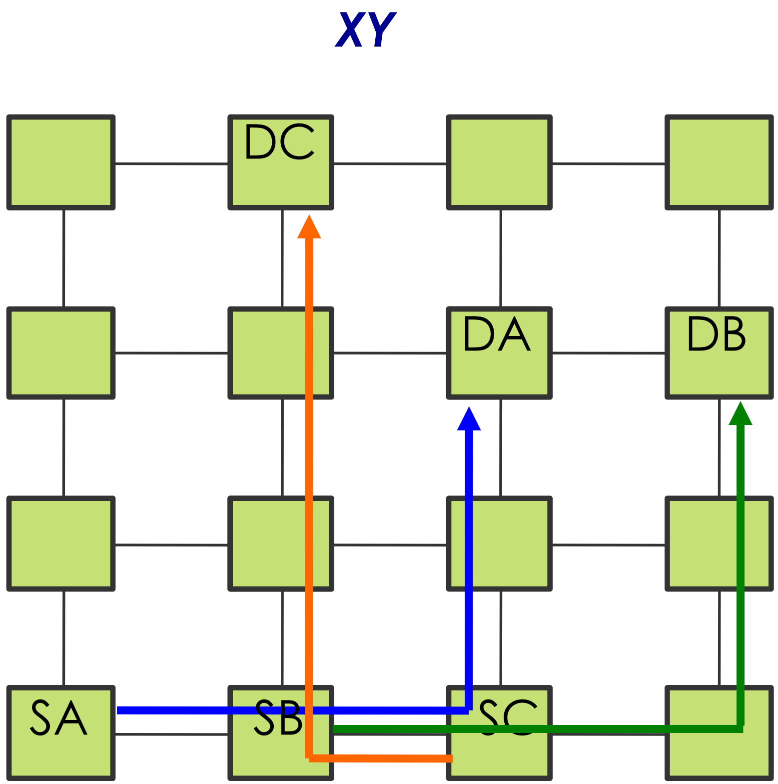
Cons of this approach?

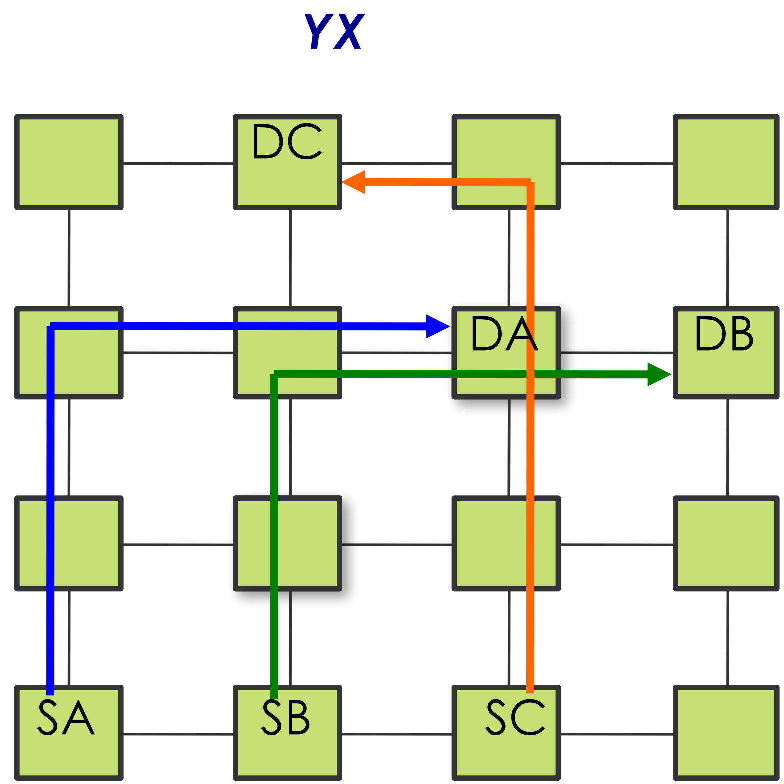
- Eliminates any path diversity provided by topology
- Poor load balancing

Taxonomy of Routing Algorithms

- **Classification II: path diversity** (how to select between the set of all possible paths R_{xy} from the source x to the dest y)
  - **Deterministic:** always choose the same route between x and y, even if $|R_{xy}| > 1$
    - **Example:** Greedy over Ring
    - Most restrictive but **most popular** due to ease of implementation and analysis
  - **Oblivious:** choose any of the routes in R_{xy} without considering any information about current network state (i.e., congestion)
    - **Example:** Random over Ring
    - Deterministic are a subset of oblivious
  - **Adaptive:** choose one of the routes in R_{xy} depending on the current network state (i.e., congestion)
    - **Example:** Adaptive over Ring
    - Congestion Metrics: link availability, buffer occupancy, history of channel load

O1TURN (Seo *et al.*, ISCA 2005)



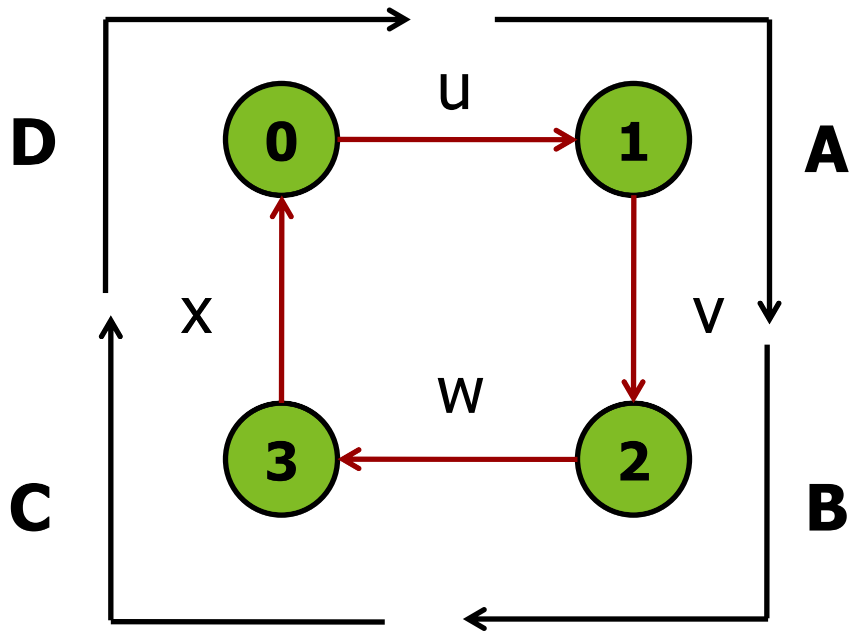


Randomly send over XY or YX

Minimal and Oblivious

Any problem?

Network Deadlock

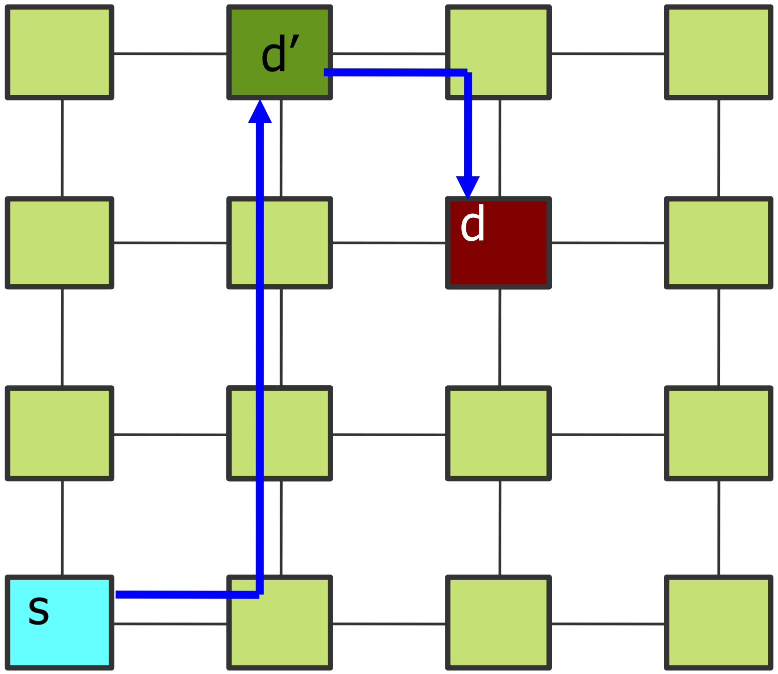


- Flow A holds u and wants v
- Flow B holds v and wants w
- Flow C holds w and wants x
- Flow D holds x and wants u

Later in the lecture!

Valiant's Routing Algorithm

- To route from s to d
  - Randomly choose intermediate node d'
  - Route* from s to d' (Phase I), and d' to d (Phase II)
- Pros
  - Randomizes any traffic pattern
    - All patterns appear uniform random
  - Balances network-load
    - Higher throughput
- Cons
  - Non-minimal
    - Higher latency and energy
  - Destroys locality

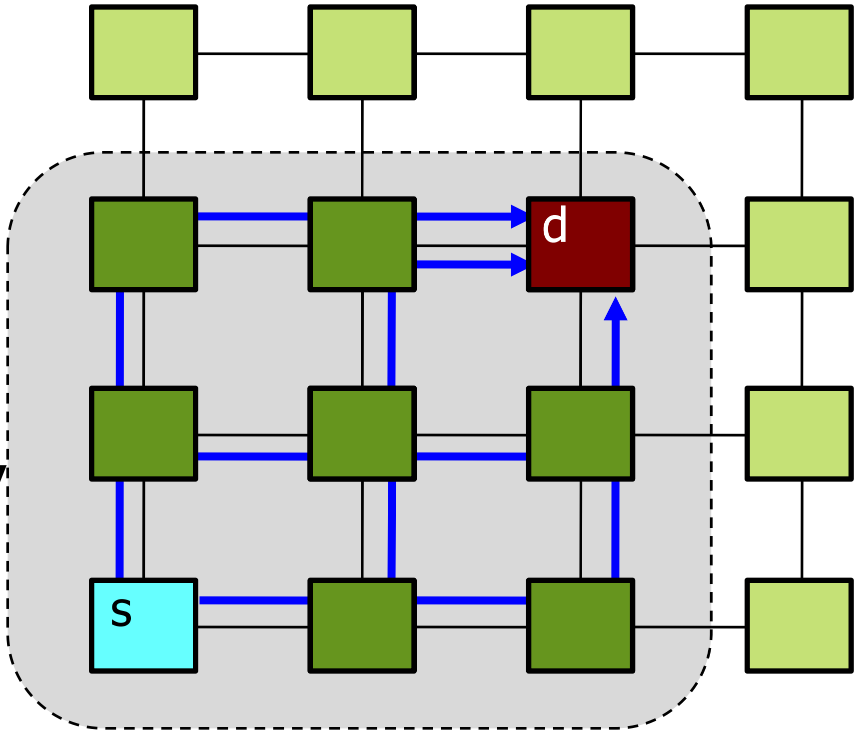


Non-Minimal and *Oblivious

can also be Adaptive

ROMM: Randomized, Oblivious Multi-phase Minimal Routing

- Confine intermediate node to be within minimal quadrant
- Retain locality + some load-balancing
- This approach essentially translates to randomly selecting between all minimal paths from source to destination



Minimal and Oblivious

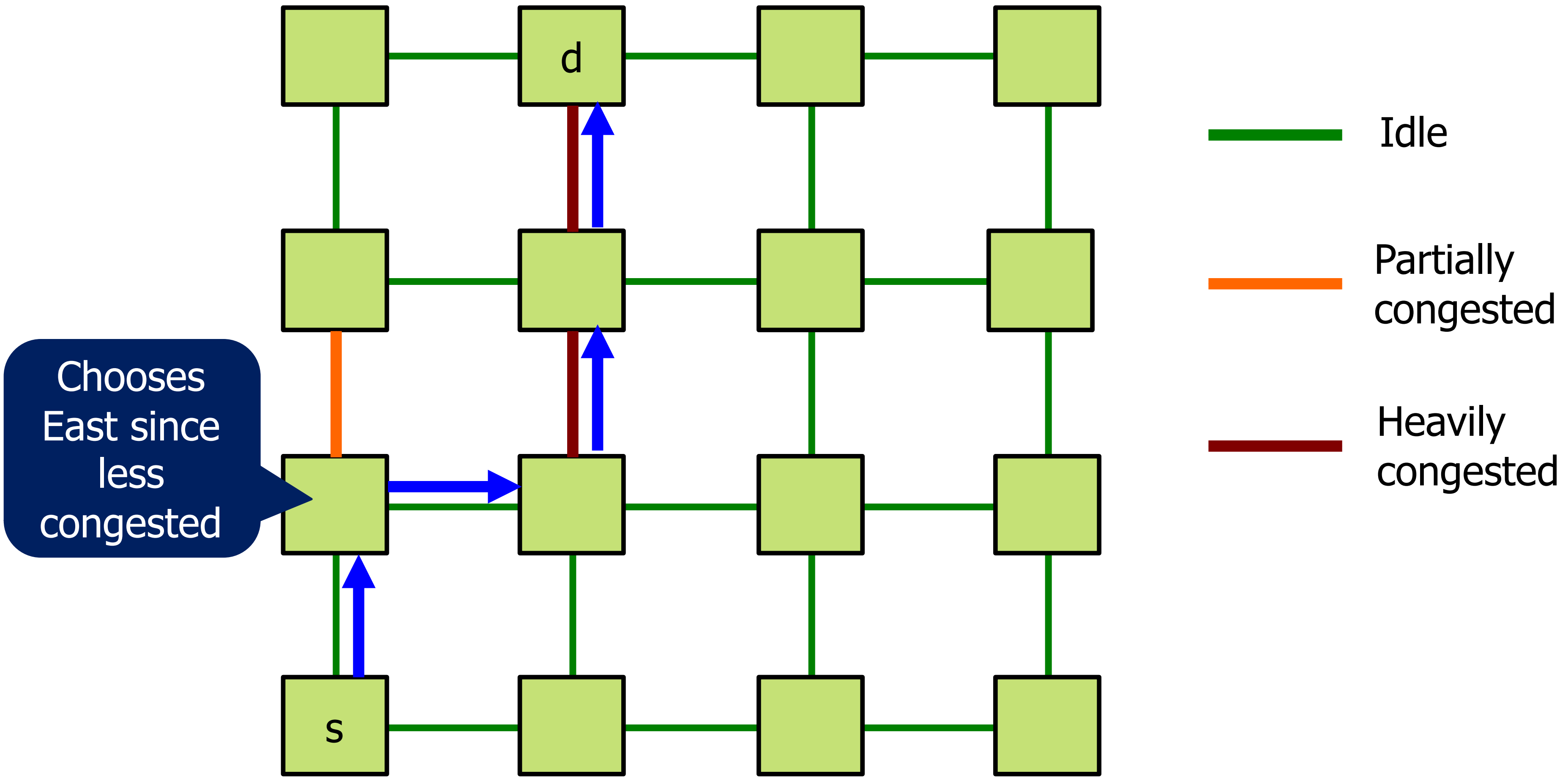
Taxonomy of Routing Algorithms

- **Classification II: path diversity** (how to select between the set of all possible paths R_{xy} from the source x to the dest y)
  - **Deterministic:** always choose the same route between x and y, even if $|R_{xy}| > 1$
    - **Example:** Greedy over Ring
    - Most restrictive but **most popular** due to ease of implementation and analysis
  - **Oblivious:** choose any of the routes in R_{xy} without considering any information about current network state (i.e., congestion)
    - **Example:** Random over Ring
    - Deterministic are a subset of oblivious
  - **Adaptive:** choose one of the routes in R_{xy} depending on the current network state (i.e., congestion)
    - **Example:** Adaptive over Ring
    - Congestion Metrics: link availability, buffer occupancy, history of channel load

Adaptive Routing Algorithms

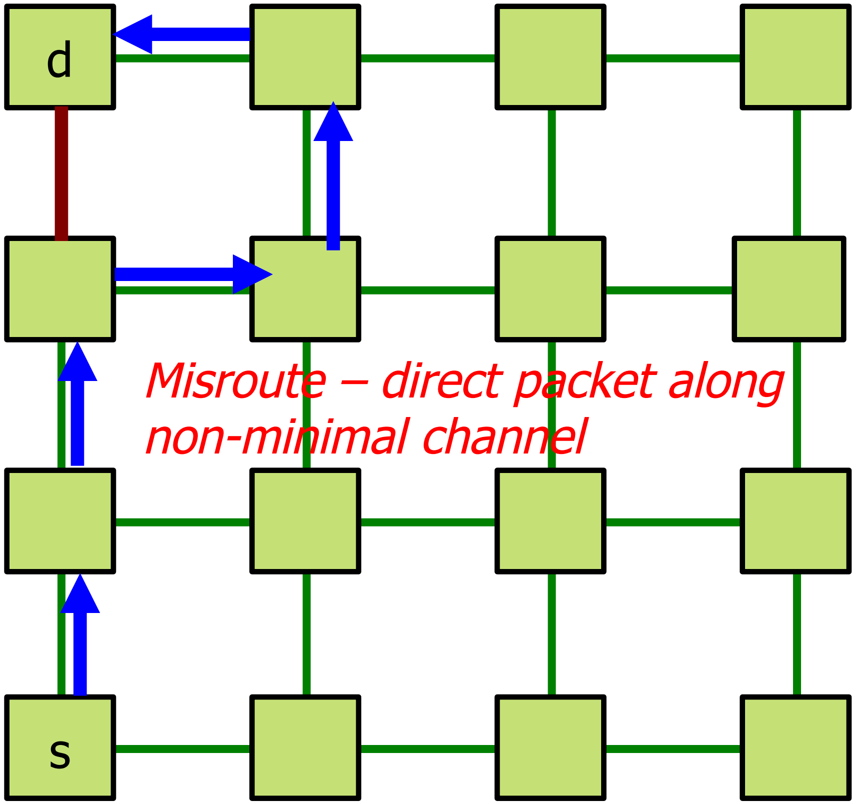
- Exploits path diversity
- Can be minimal or non-minimal
- Uses network state to make routing decisions
  - Buffer occupancies often used
  - Coupled with flow control mechanism
- Local information readily available
  - Global information more costly to obtain
  - Problems
    - Network state can change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

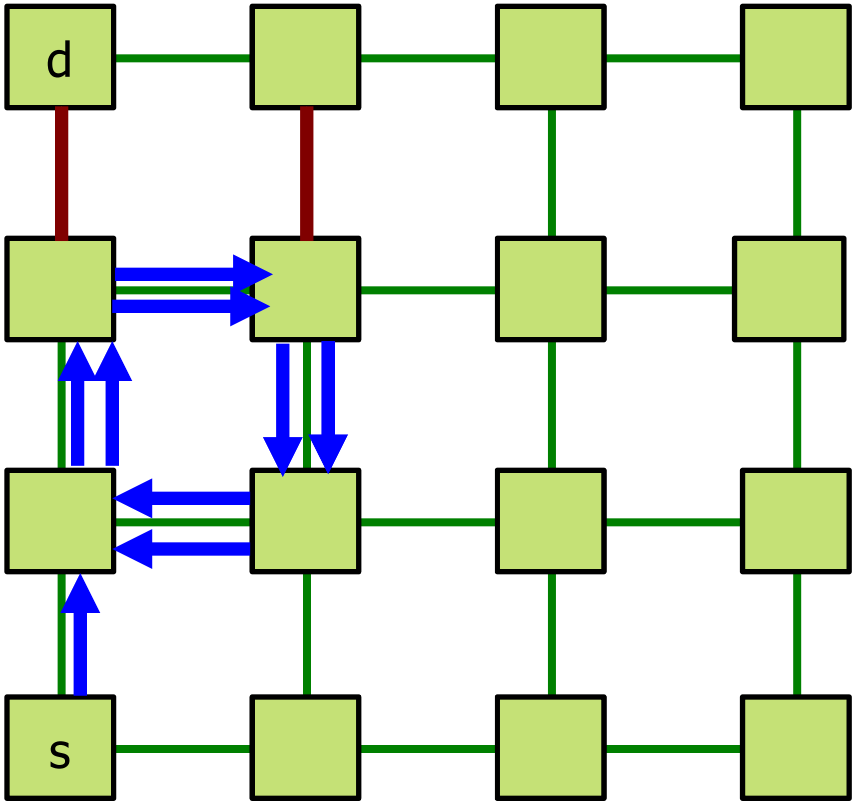
Example 1: Minimal Adaptive Routing



Local info can result in sub-optimal choices

Example 2: Non-Minimal Adaptive Routing





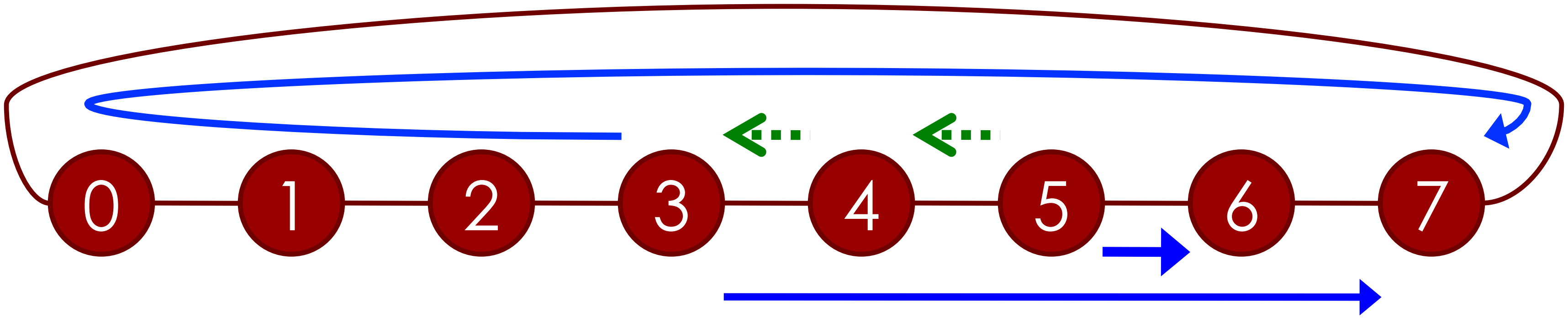
Livelock! – continue routing in cycle

Longer path with potentially lower latency

To guarantee forward progress, limit number of misroutings

How to sense congestion?

5→6 and 3→7



- **5 → 6:** Route counterclockwise (1-hop)
- **3 → 7:** Both clockwise and counterclockwise are 4 hops!
  - Which one should 3 choose?
    - Clockwise, since 5 is using all the capacity of link 5→6
  - **Problem?**
    - Queue at node 5 will sense contention. But node 3 will not, and may continue to send counterclockwise
- **Backpressure** – allows nodes to indirectly sense congestion
  - Queue in node 5 will fill up and stop receiving flits
  - Previous queues will start filling up
    - If each queue holds 4 packets, node 3 will send 8 packets before sensing congestion
  - More on backpressure in flow-control lecture

Taxonomy of Routing Algorithms

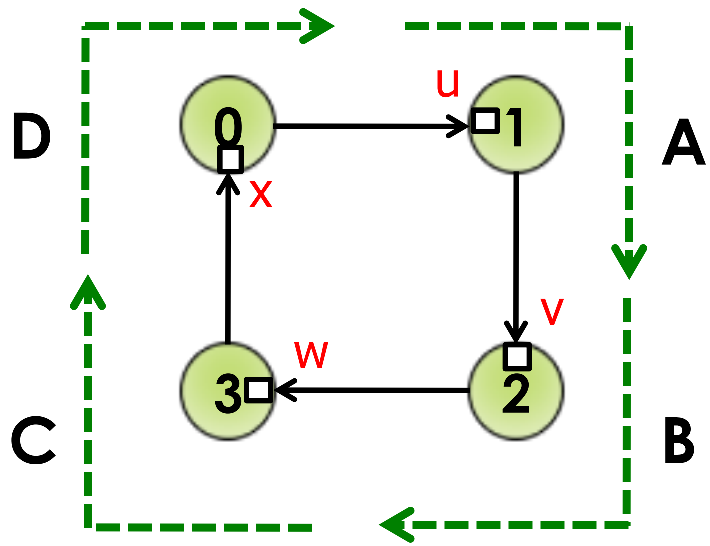
- **Classification III – implementation**
  - **Source Routing:** embed entire route (i.e., list of output ports) in the packet
  - **Node-Table Routing:** every node has a routing table which stores the output link that a packet from each source should take
  - **Combinational Circuits:** packet carries only destination coordinates, and each router computes output port based on packet state and router state
    - e.g., **deterministic:** use remaining hops and direction
    - e.g., **oblivious:** use remaining hops and direction and some randomness factor
    - e.g., **adaptive:** use congestion metrics (such as buffer occupancy), history, etc.

Routing

- Taxonomy
- **Deadlocks**
- Conclusion

Network Deadlock

- A condition in which a set of **agents** wait indefinitely trying to acquire a set of **resources**



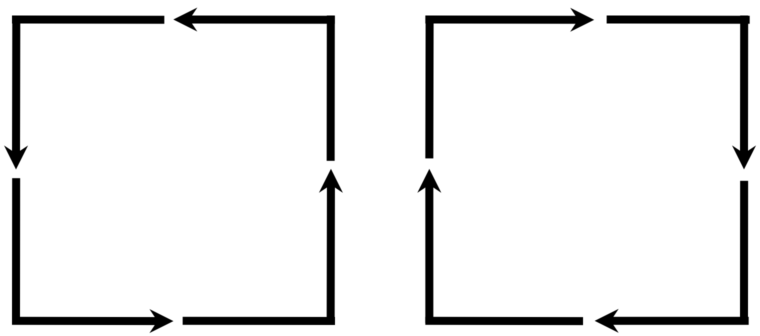
Note: holding buffer u == holding Channel 01 as no other packet can use channel 01 till buffer u becomes free

- Packet A holds buffer u (in 1) and wants buffer v (in 2)
- Packet B holds buffer v (in 2) and wants buffer w (in 3)
- Packet C holds buffer w (in 3) and wants buffer x (in 0)
- Packet D holds buffer x (in 0) and wants buffer u (in 1)

Deadlock Avoidance

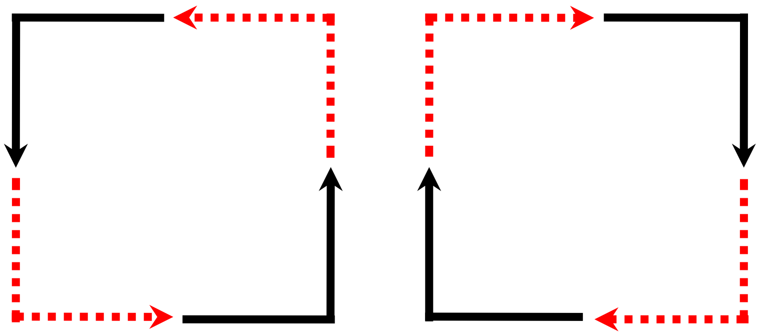
- Eliminate cycles in Resource Dependency Graph
  - Resource Ordering
    - Enforce a partial/total order on the resources, and insist that an agent acquire the resources in ascending order
    - Deadlock avoided since a cycle must contain at least one agent holding a higher numbered resource waiting for a lower-numbered resource which is not allowed by the ordering allocation
  - Implementation
    - Restrict certain routes so that a higher numbered resource cannot wait for a lower numbered resource
    - Partition the buffers at each node such that they belong to different resource classes. A packet only any route can only acquire buffers in ascending order of resource class

Turn Model (Glass and Ni 1994) for Mesh

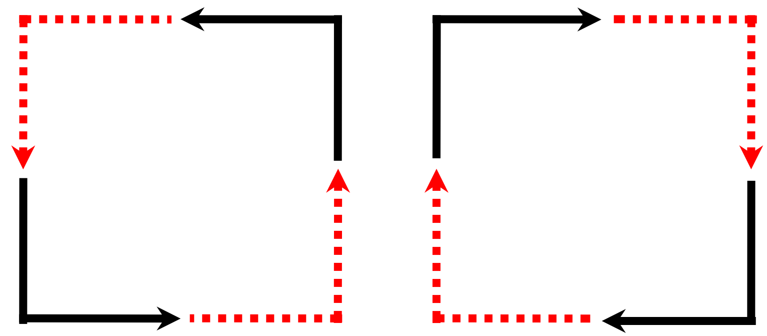


- Deadlocks may occur if the turns taken form a cycle
  - Removing some turns can make the routing algorithm deadlock free

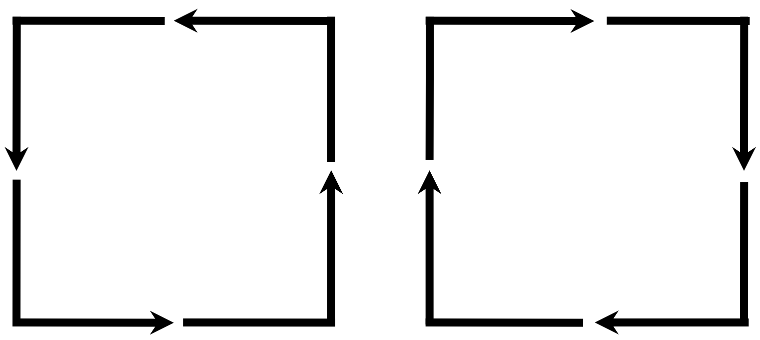
Dimension Ordered Routing



XY Model

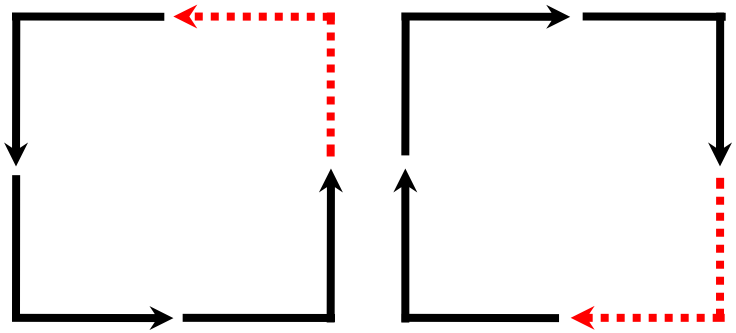


YX Model

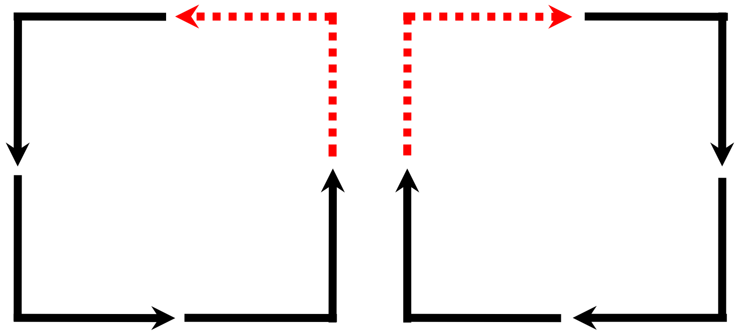


O1Turn **Deadlock!**

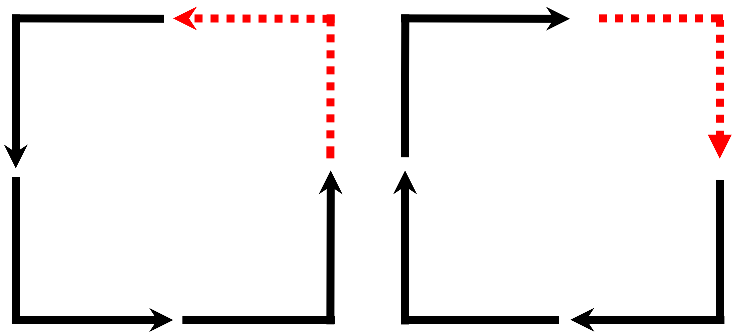
Deadlock-free Oblivious/Adaptive Routing Algorithms



West-First Turn Model

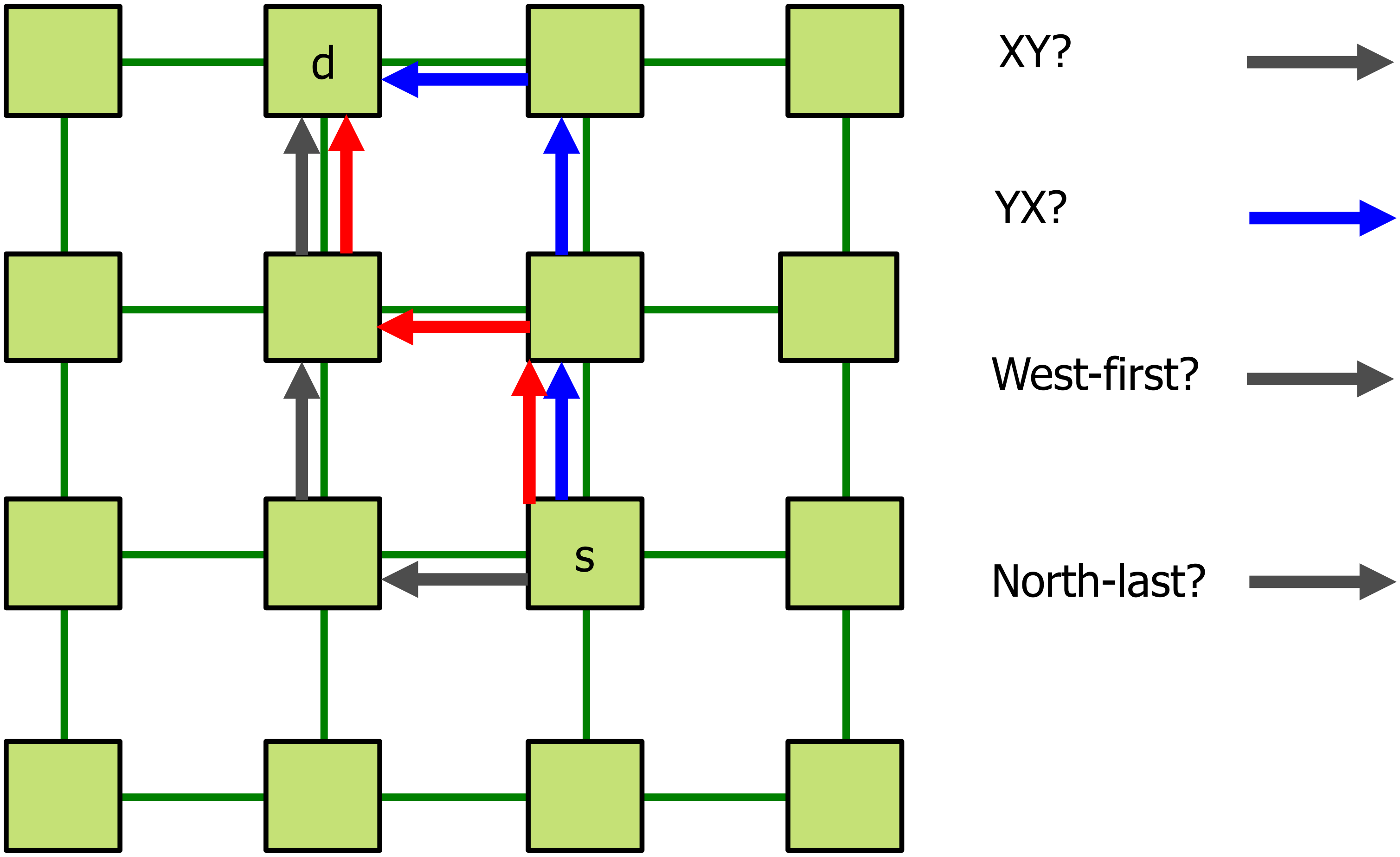


North-Last Turn Model

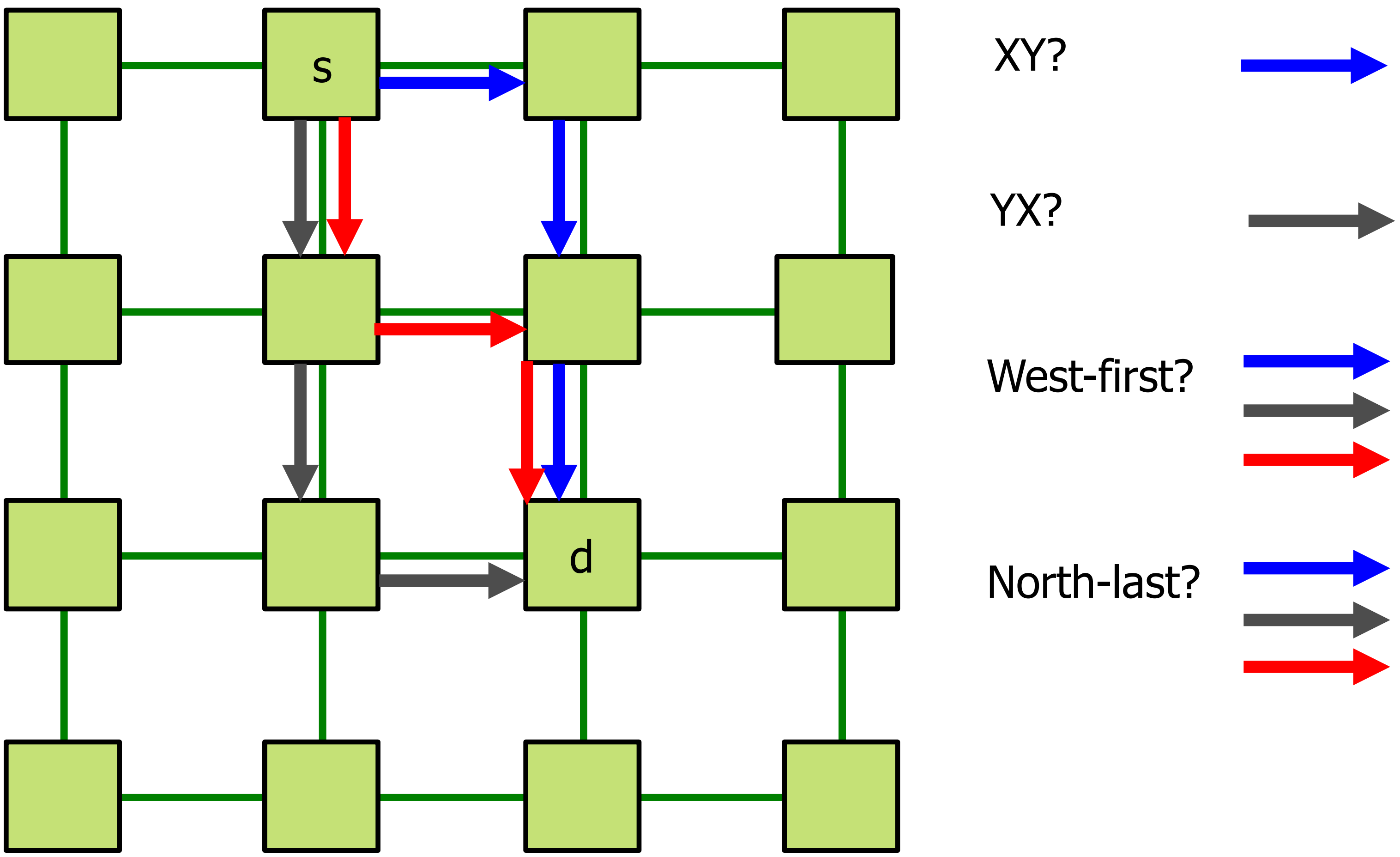


Negative-First Turn Model

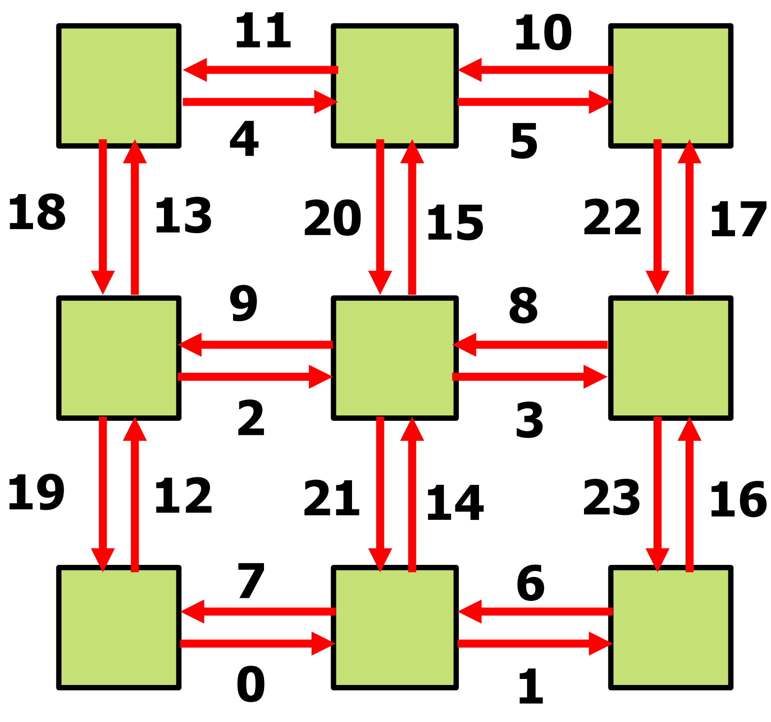
Example 1



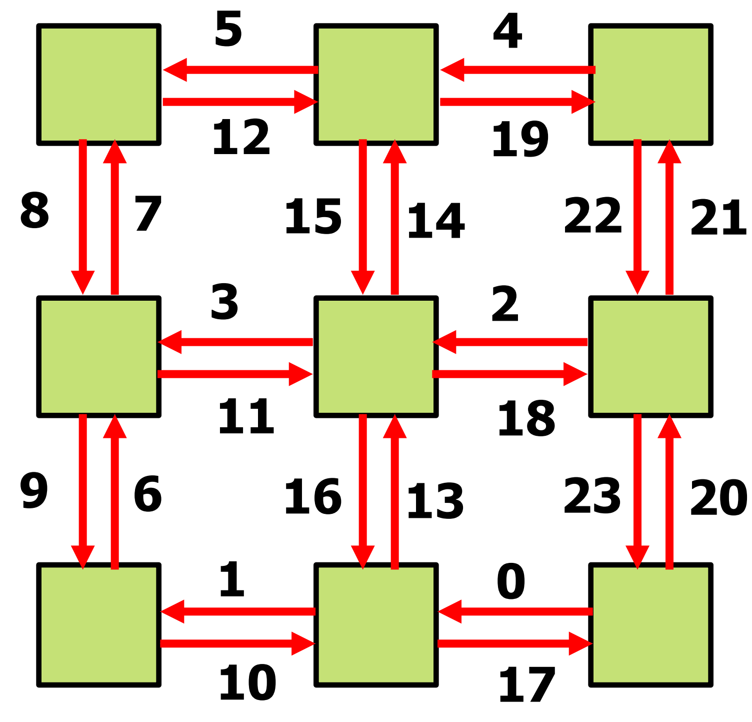
Example 2



Resource (channel) Ordering

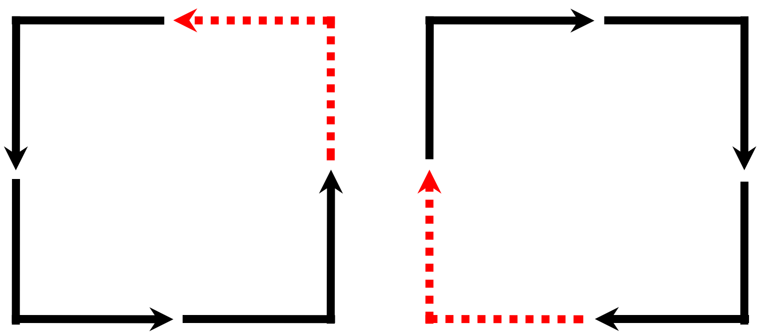


XY Model

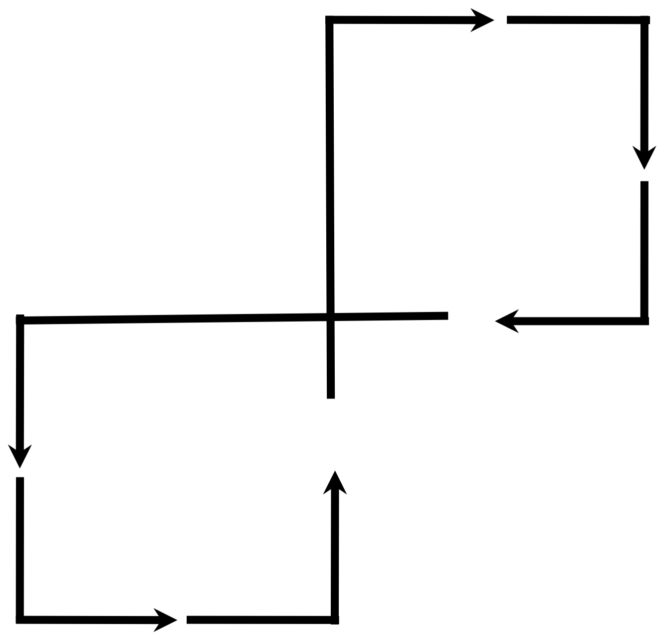


West-First Turn Model

Can we eliminate *any* 2 turns?



Six turn model

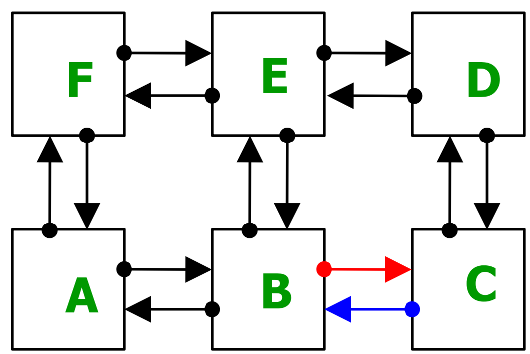


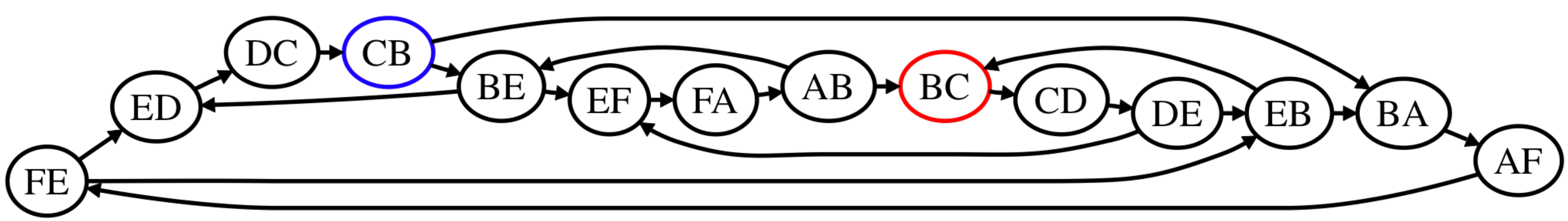
Deadlock!

Total Turn Models = 16
Deadlock Free = 12
Unique (non-symmetrical) = 3

Channel Dependency Graph (CDG)

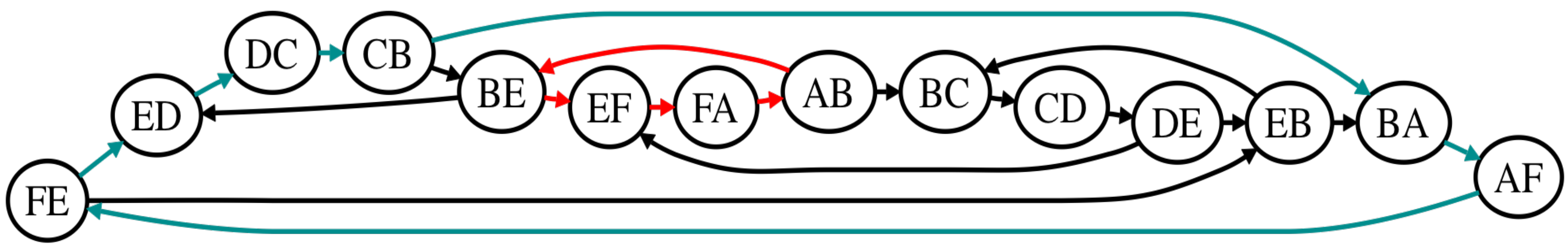
- Vertices represent network links (channels)
- Edges represent turns
  - *180° turns not allowed, e.g., AB → BA*

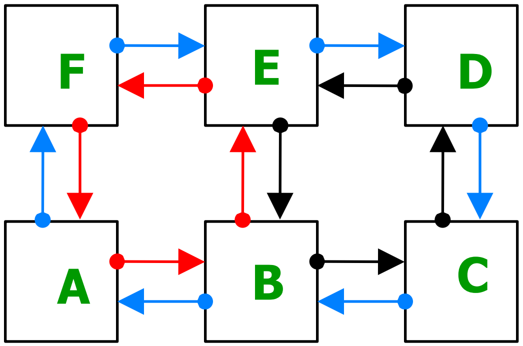




Cycles in the CDG

The channel dependency graph D derived from the network topology may contain many *cycles*

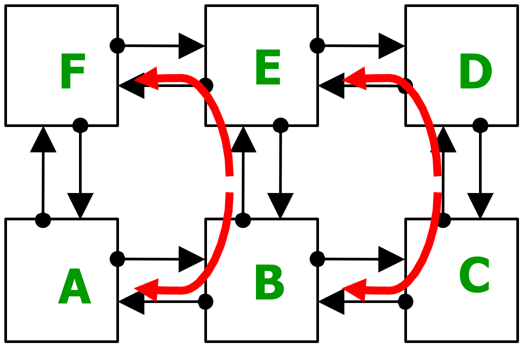




Flow routed through links AB, BE, EF
Flow routed through links EF, FA, AB
Deadlock!

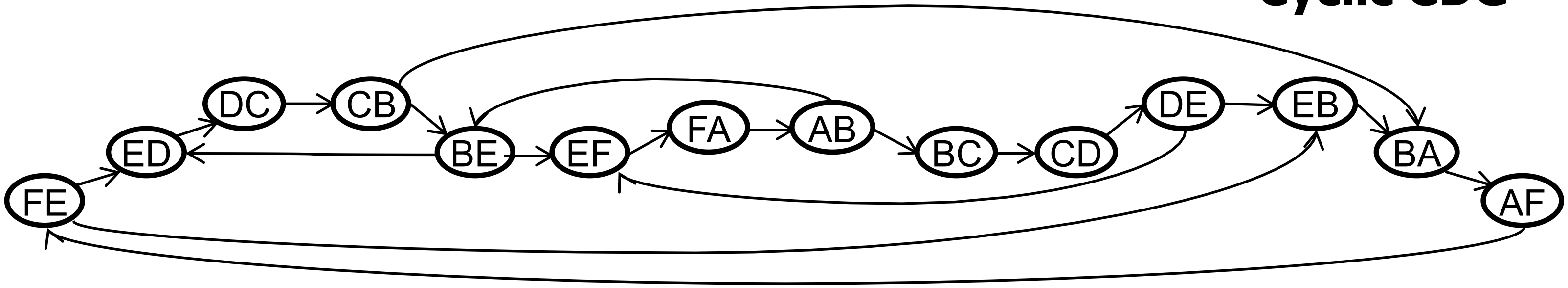
Edges in CDG = Turns in Network
➔ Disallow/Delete certain edges in CDG

Acyclic CDG

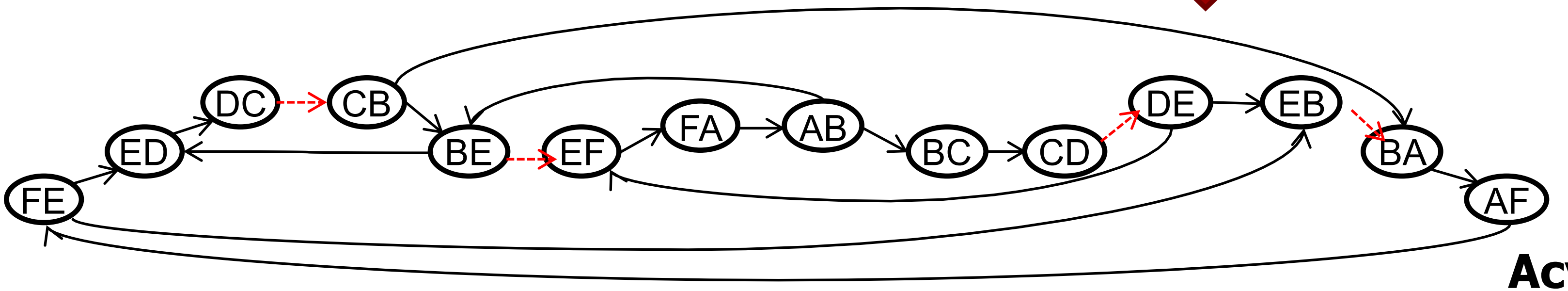


This is the West-first turn model!

Cyclic CDG

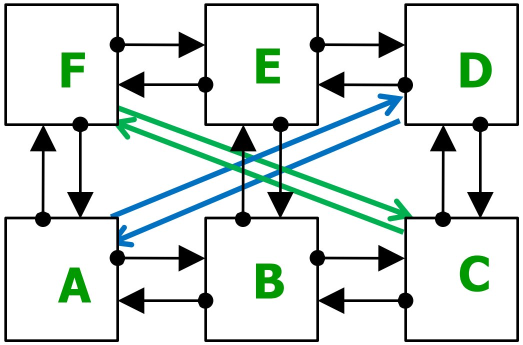


Disable certain edges



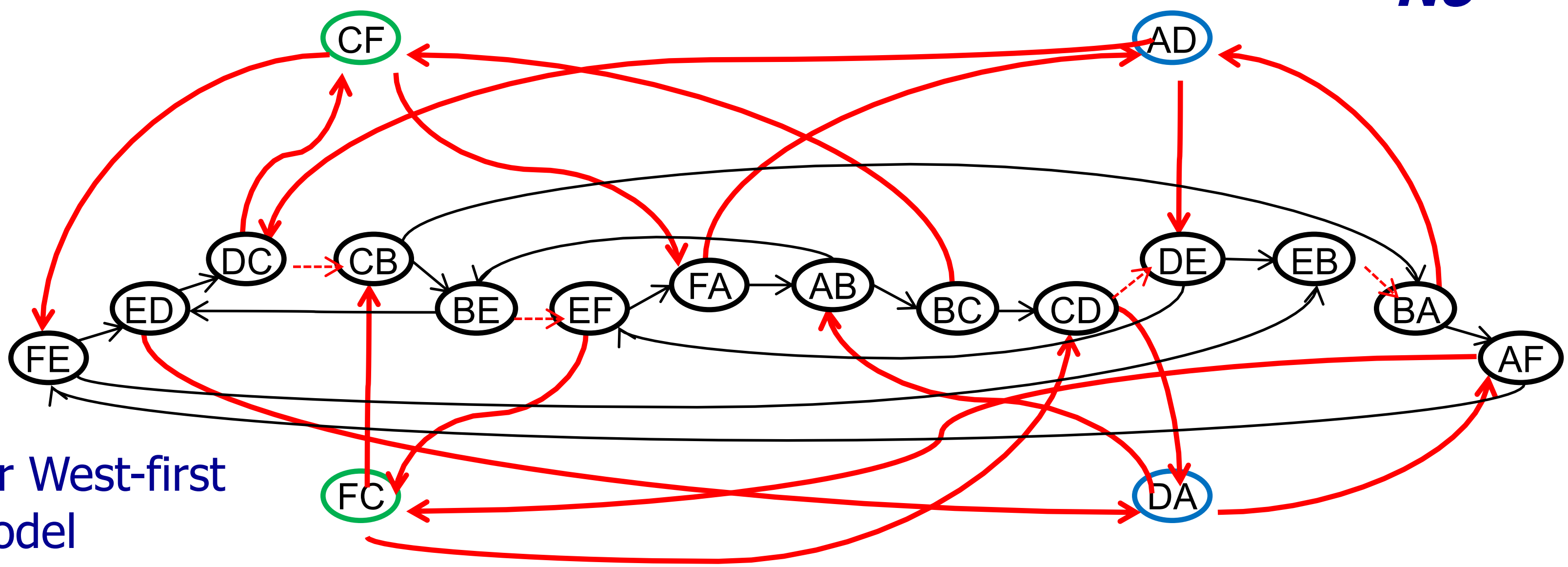
Acyclic CDG

CDG for arbitrary topology



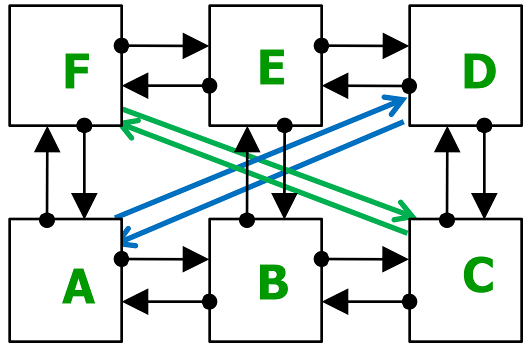
Deadlock free?

No

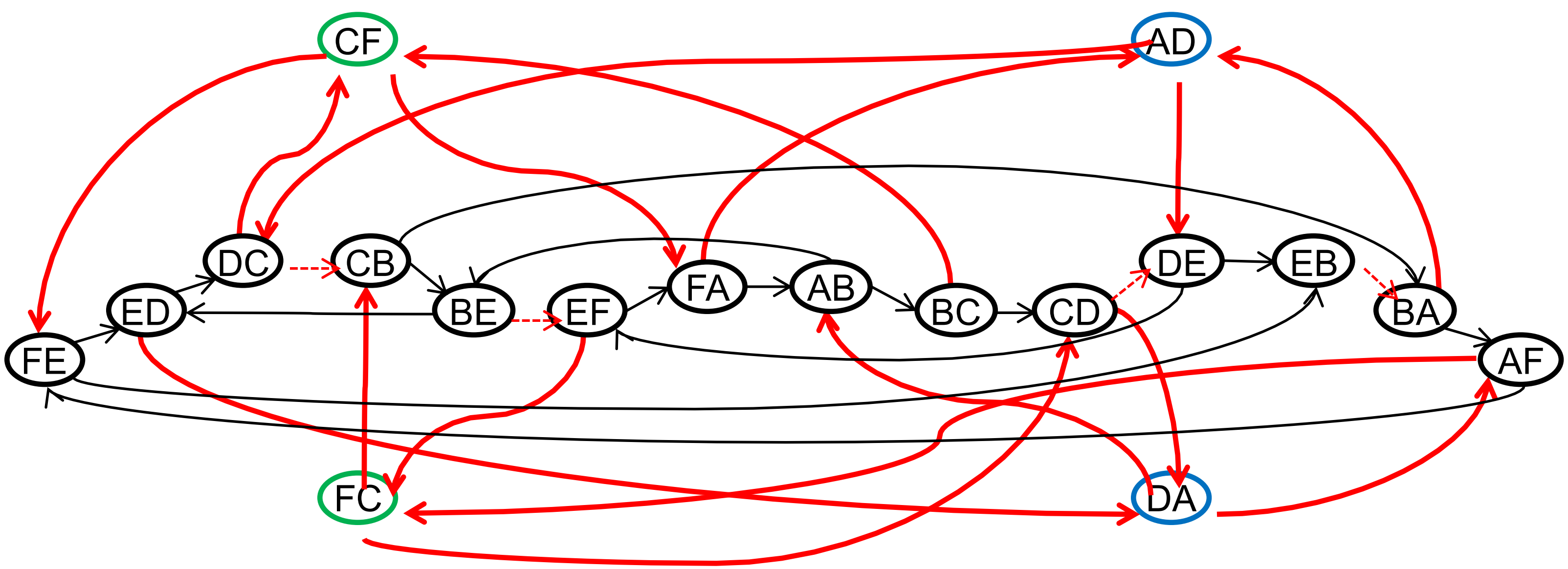


CDG for West-first turn model

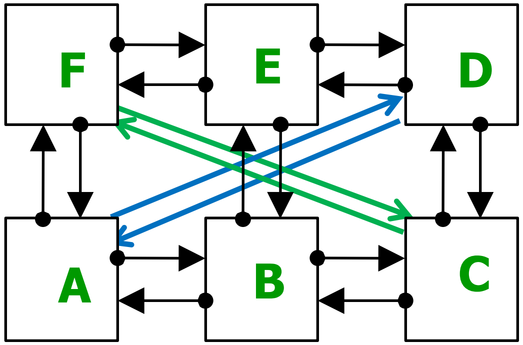
Deadlock-free Routing Algorithm



Suppose: Diagonal links should be traversed last (i.e., no edge from blue/green channel to black)



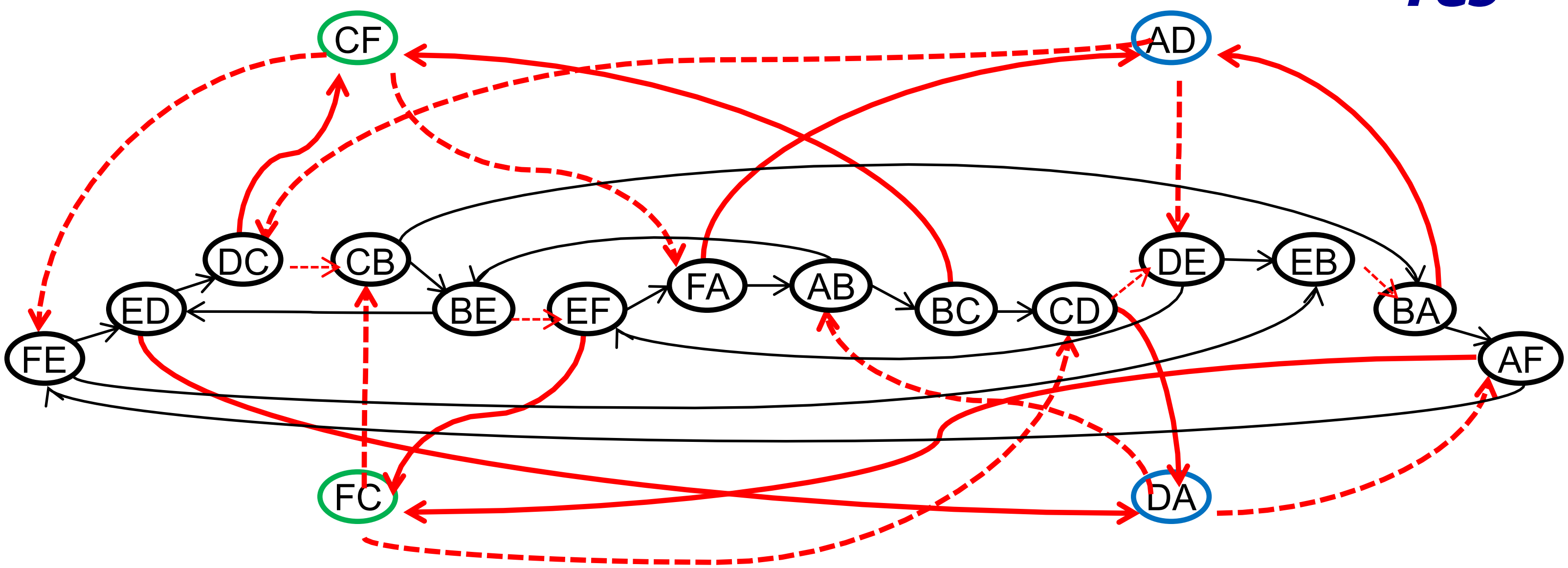
Deadlock-free Routing Algorithm



Suppose: Diagonal links should be traversed last (i.e., no edge from blue/green channel to black)

Deadlock free?

Yes



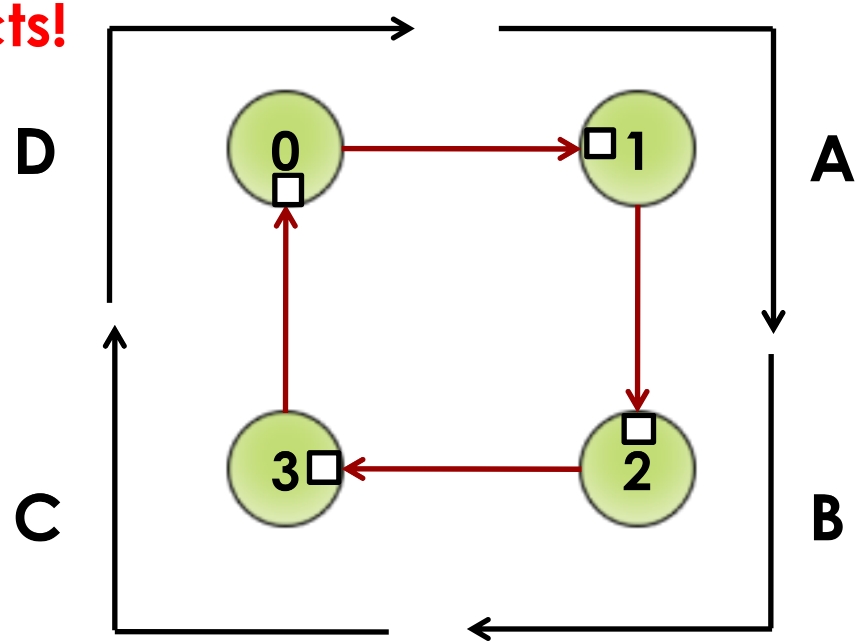
Path Diversity vs Deadlock

- Path diversity required for higher throughput
- Path restrictions because of deadlock-fre routing requirement

- Can we allow all turns and still get deadlock freedom ?

Why do deadlocks occur?

Resource conflicts!

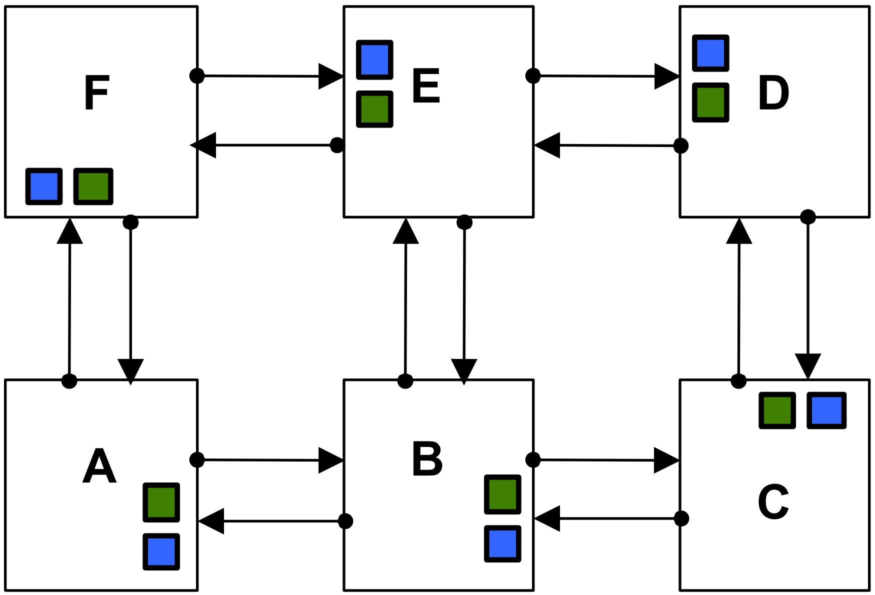


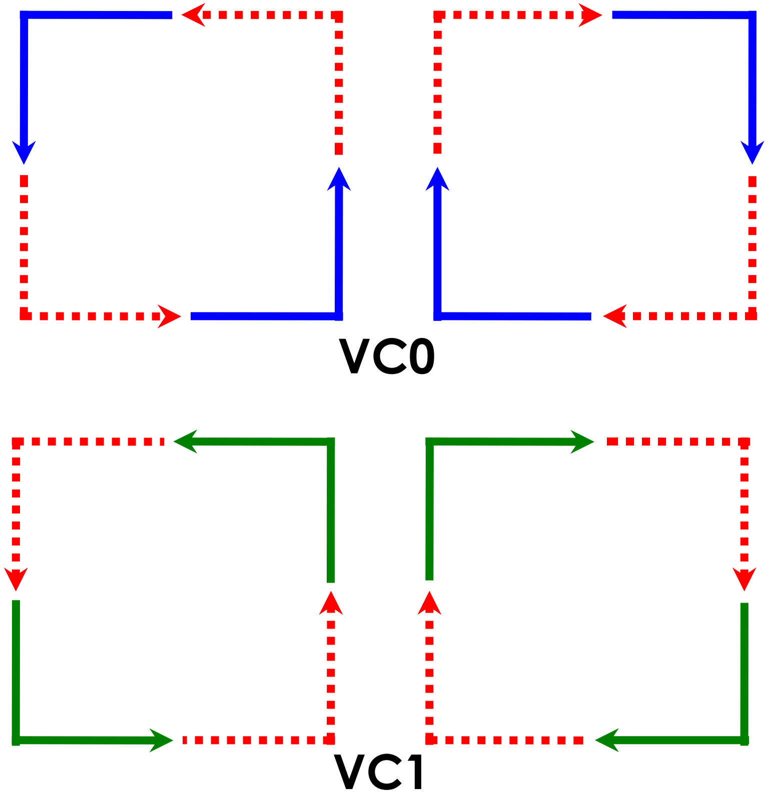
- Flow A holds buffer in 1 and wants buffer in 2
- Flow B holds buffer in 2 and wants buffer in 3
- Flow C holds buffer in 3 and wants buffer in 0
- Flow D holds buffer in 0 and wants buffer in 1

Add more buffers and partition!

Virtual Channels

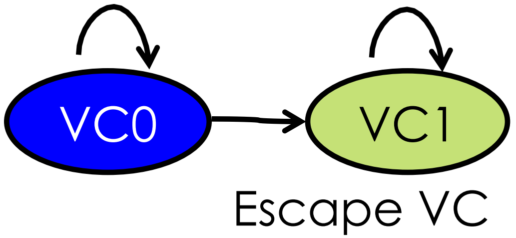
- Same physical link/channel between routers
  - additional buffers in each router to avoid deadlocks – called "virtual" channels

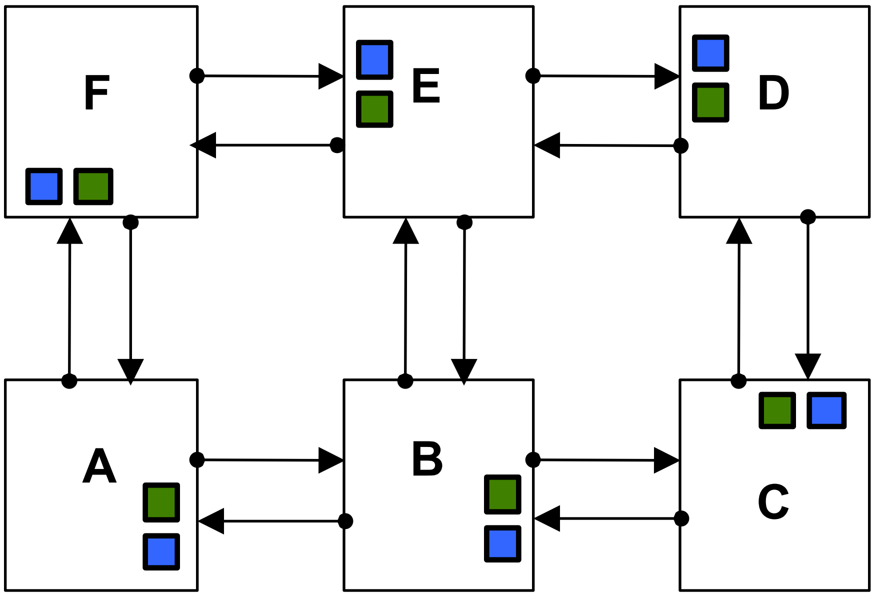


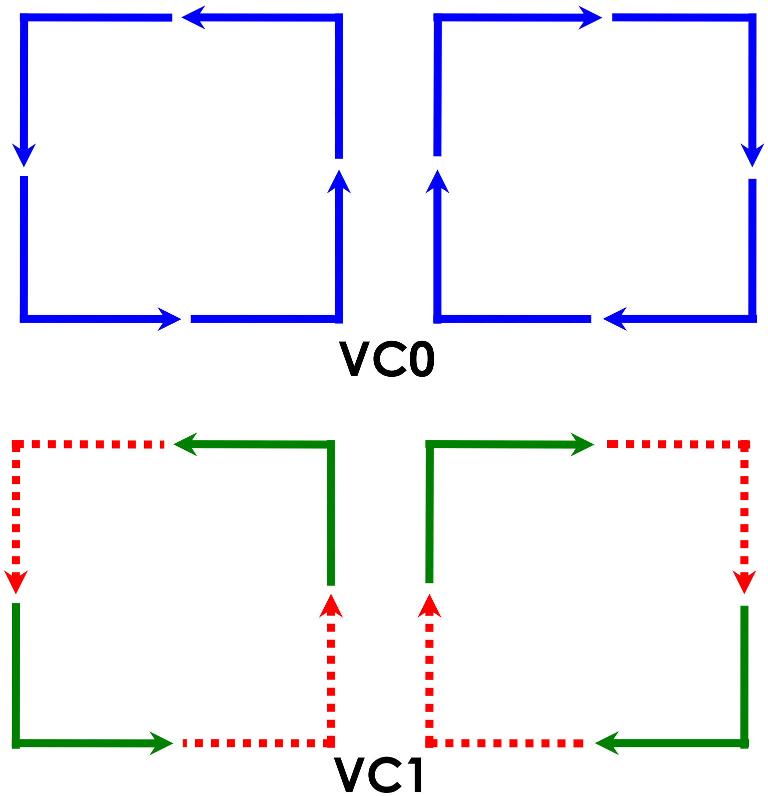


Escape Virtual Channels

- Allow any turns across all VCs except one
  - "Escape" VC → deadlock-free route
- If there is a deadlock, can jump into escape VC which is guaranteed to drain







Routing

- Taxonomy
- Deadlocks
- **Conclusion**

Active Research Topics in NoC Routing

- Adaptive routing
  - How to efficiently convey and utilize traffic information
- Routing for broadcasts/multicasts
- Routing in the presence of router/link faults
  - Important in sub-nm technology nodes
- Deadlock-free routing for irregular topologies
  - E.g., turn off certain routers for power reasons