ECE 6115 / CS 8803 - ICN
Interconnection Networks for High Performance Systems
Spring 2020

ROUTING

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

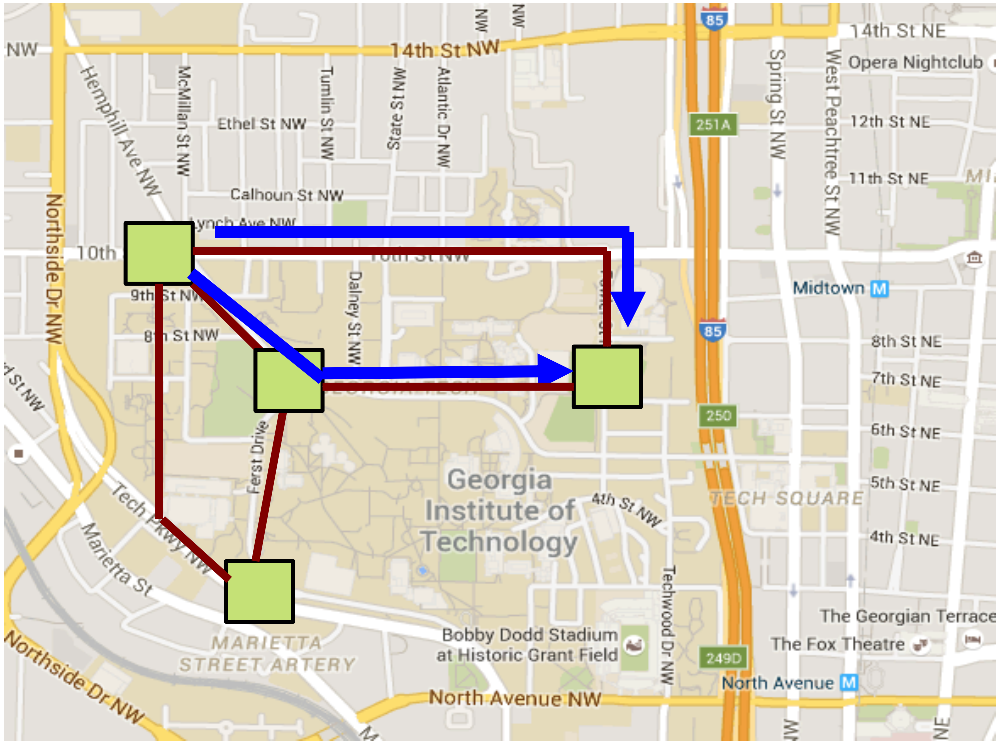
tushar@ece.gatech.edu

NETWORK ARCHITECTURE

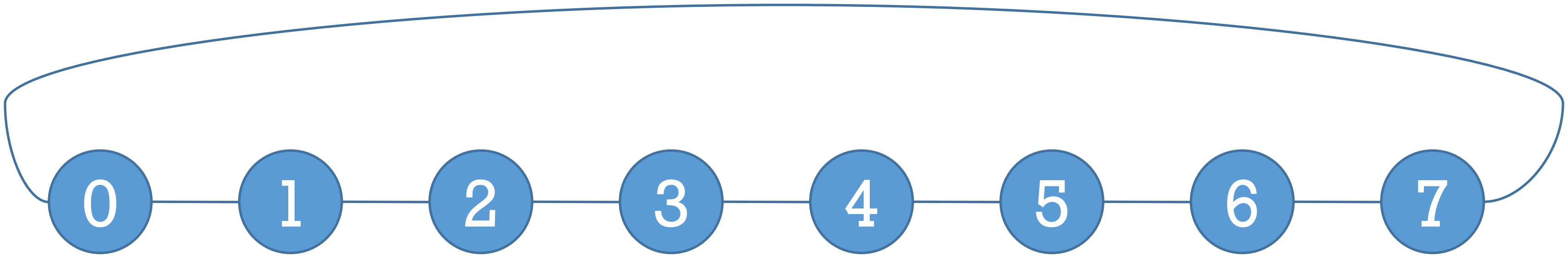
- **Topology**
  - How to connect the nodes
  - ~Road Network
- **Routing**
  - Which path should a message take
  - ~Series of road segments from source to destination
- **Flow Control**
  - When does the message have to stop/proceed
  - ~Traffic signals at end of each road segment
- **Router Microarchitecture**
  - 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

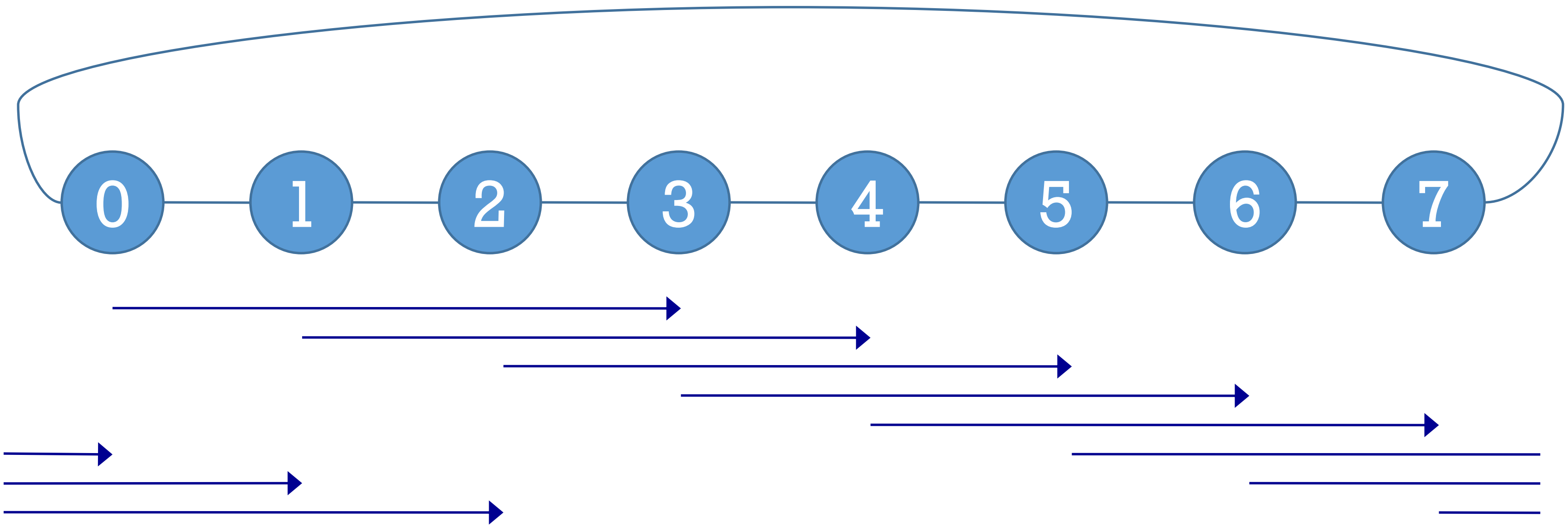


WHY DOES ROUTING MATTER?



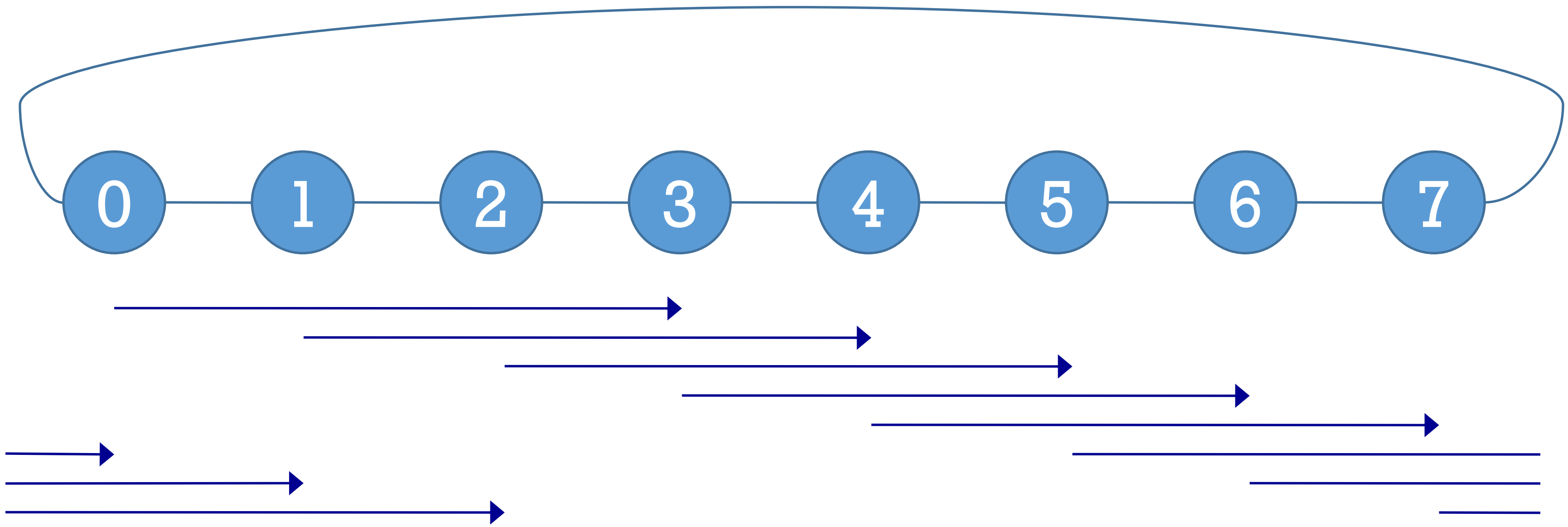
- Suppose three routing options
  - **Greedy:** shortest path
  - **Random:** randomly pick direction
  - **Adaptive:** monitor load in each direction and send
- Which routing algorithm is the best?
  - Depends …what is the traffic pattern?
  - What metric (latency/throughput/energy) do we care about?

SUPPOSE TRAFFIC = TORNADO



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

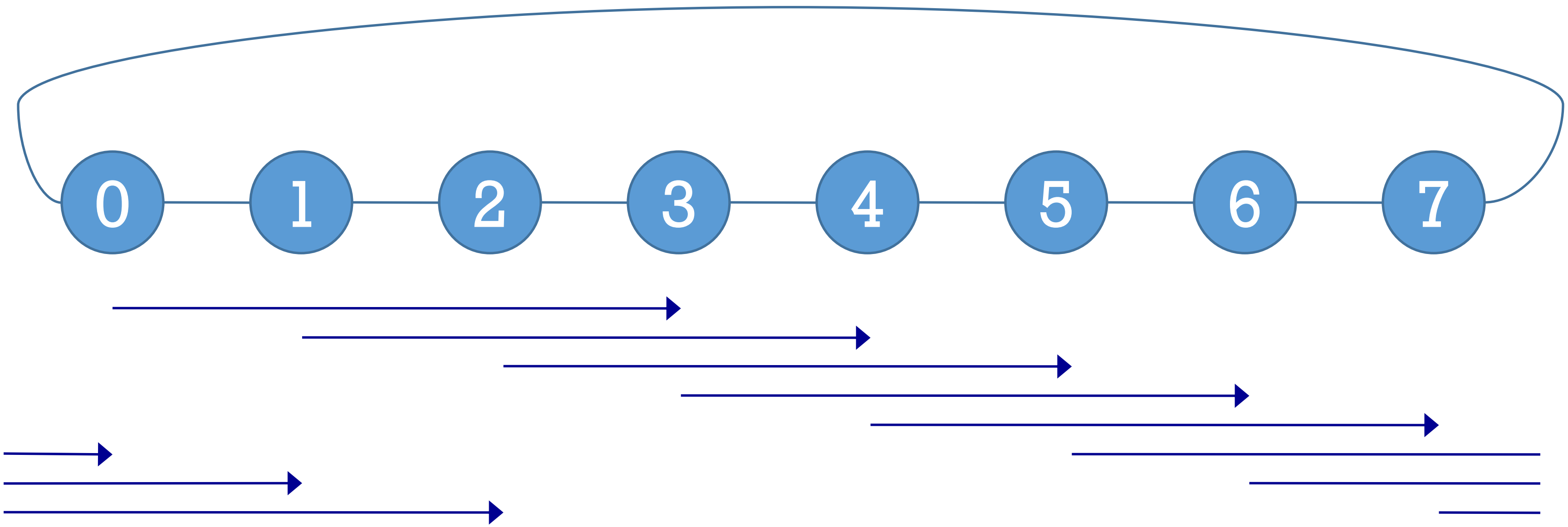
METRIC = ZERO-LOAD LATENCY



- Best routing algorithm?

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

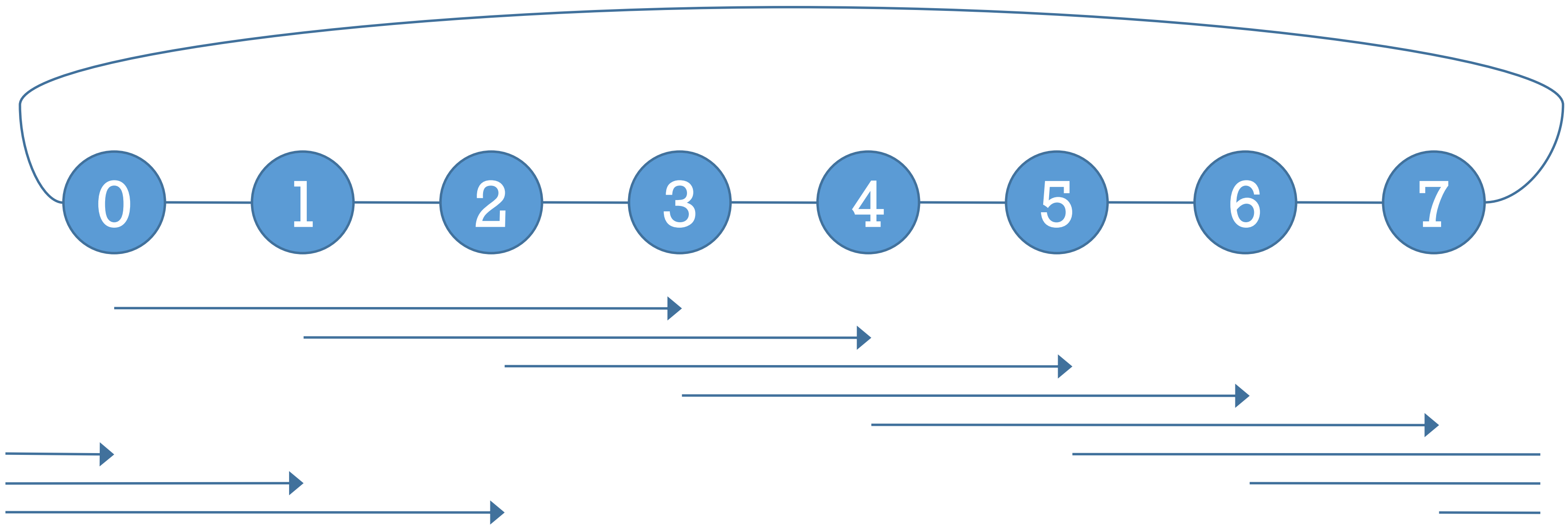
METRIC = ENERGY



- Best routing algorithm?

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

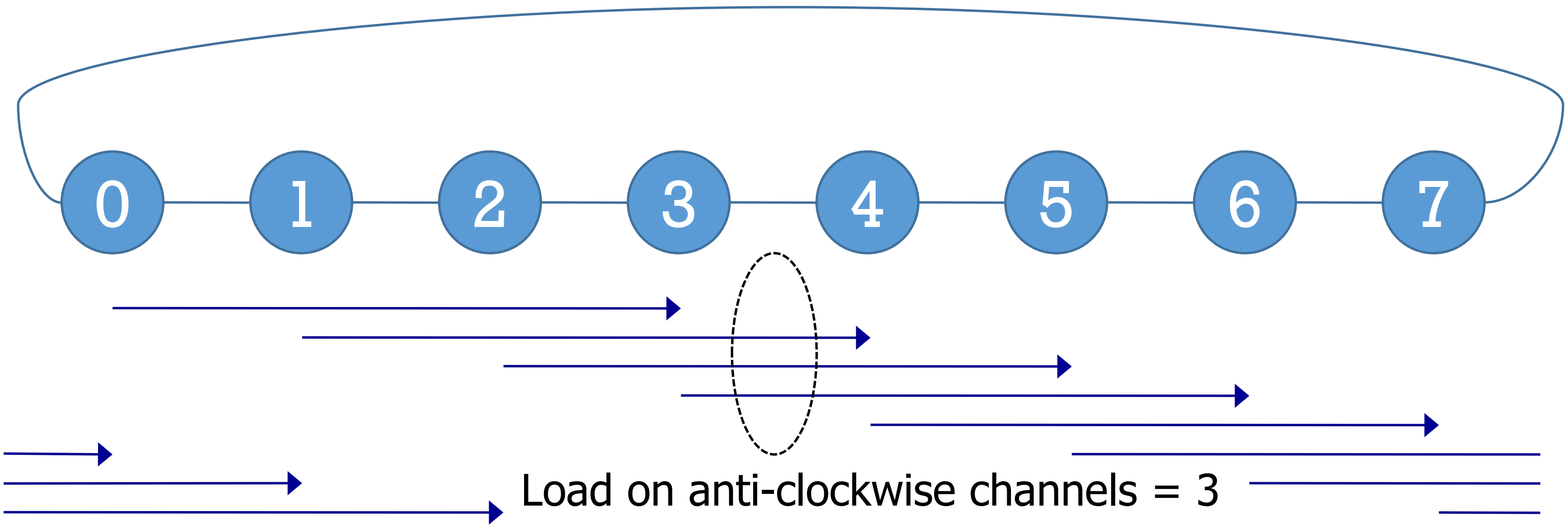
METRIC = THROUGHPUT



- Best routing algorithm?

	Greedy	Random	Adaptive
Max Channel Load			
Throughput			

CHANNEL LOAD FOR GREEDY TRAFFIC

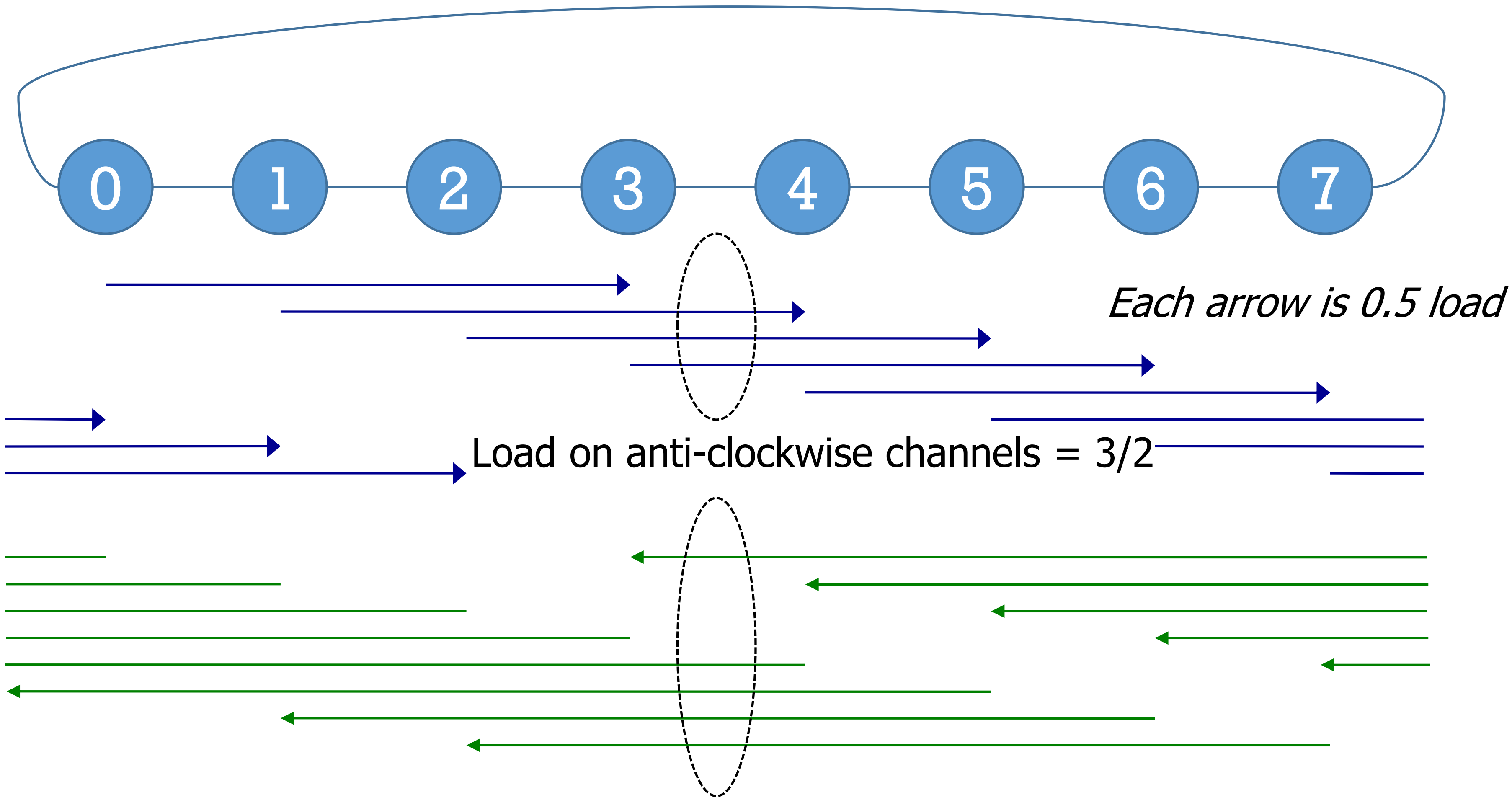


Load on anti-clockwise channels = 3

Throughput = 1/3

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

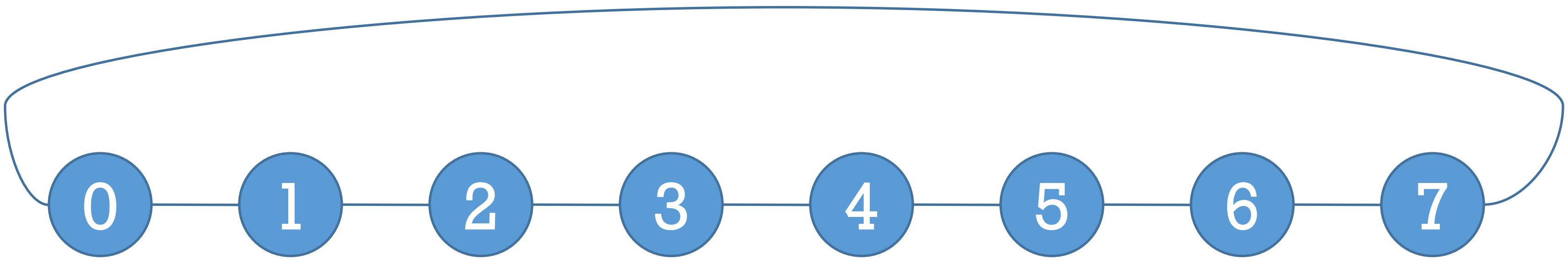
CHANNEL LOAD FOR RANDOM TRAFFIC



Load on clockwise channels = 5/2

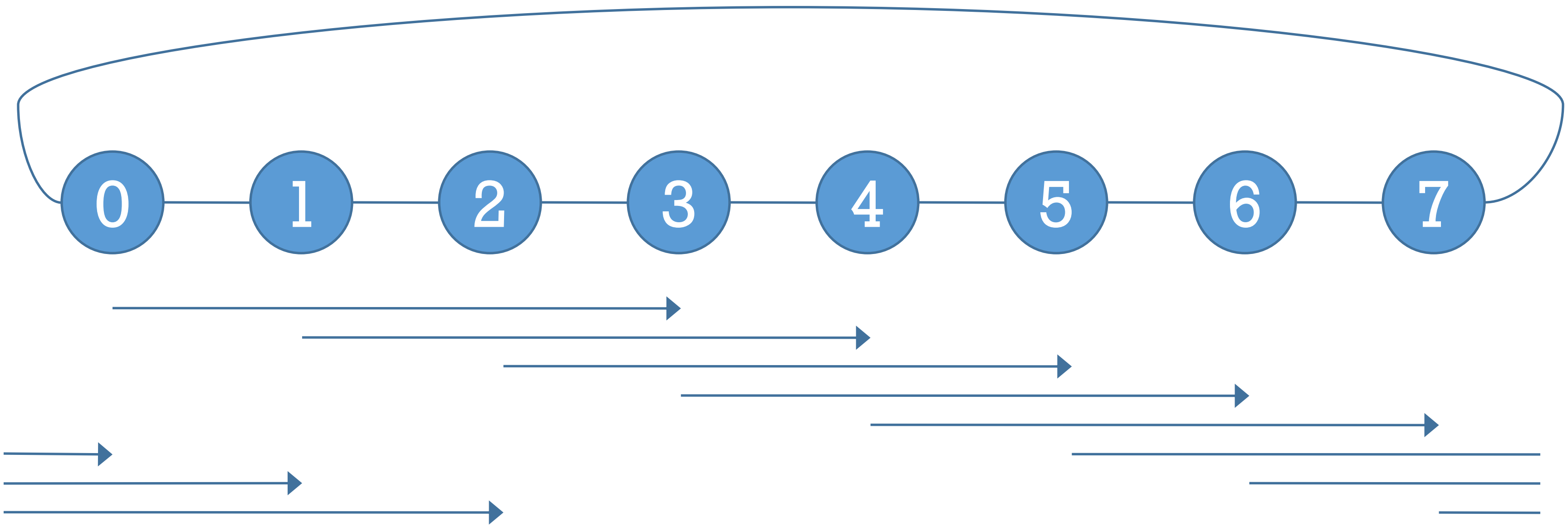
Throughput = 2/5

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

TAXONOMY OF ROUTING ALGORITHMS

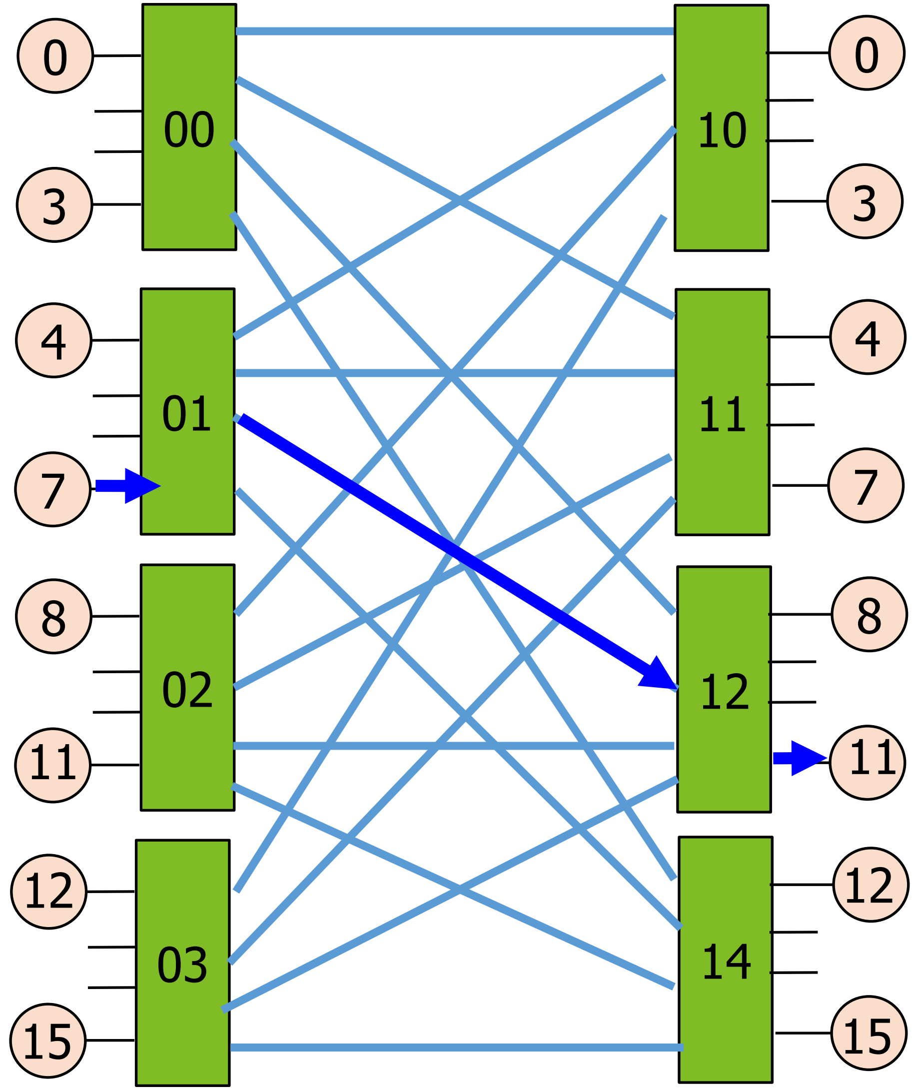
- **Classification I: path length**
  - **Minimal**: shortest paths
    - Example: Greedy over Ring
  - **Non-minimal**: non-shortest paths
    - Example: Random and Adaptive over Ring

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

DESTINATION-TAG ROUTING IN BUTTERFLY (2)

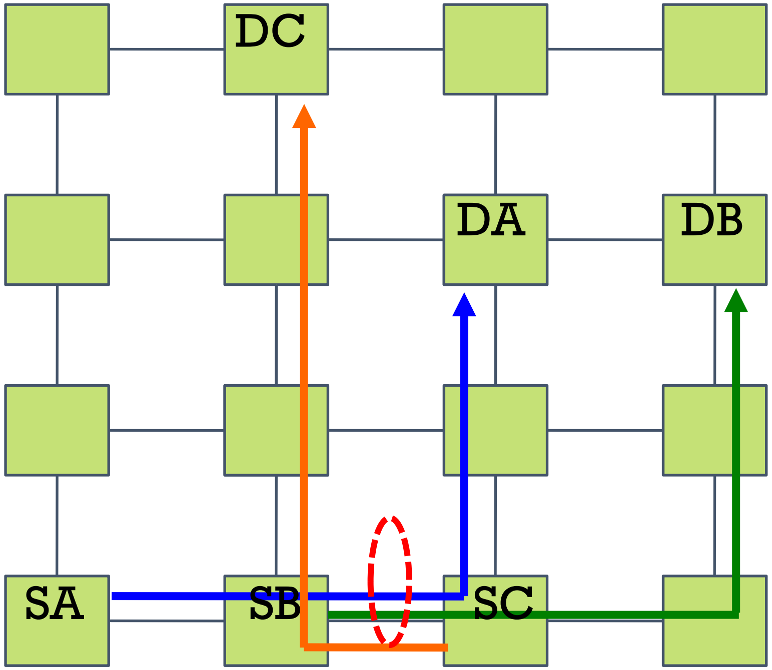
- Routing from 7 to 11
  - k = 4 (ports per switch)
  - Destination node 11 = 1011_2 = 23_4
  - To route to Node 11 use port 2 then 3
- Source does not play any role in routing



Minimal and Deterministic

DIMENSION-ORDERED ROUTING (DOR) IN A MESH

XY Routing: Always go X first, then Y



Minimal and Deterministic

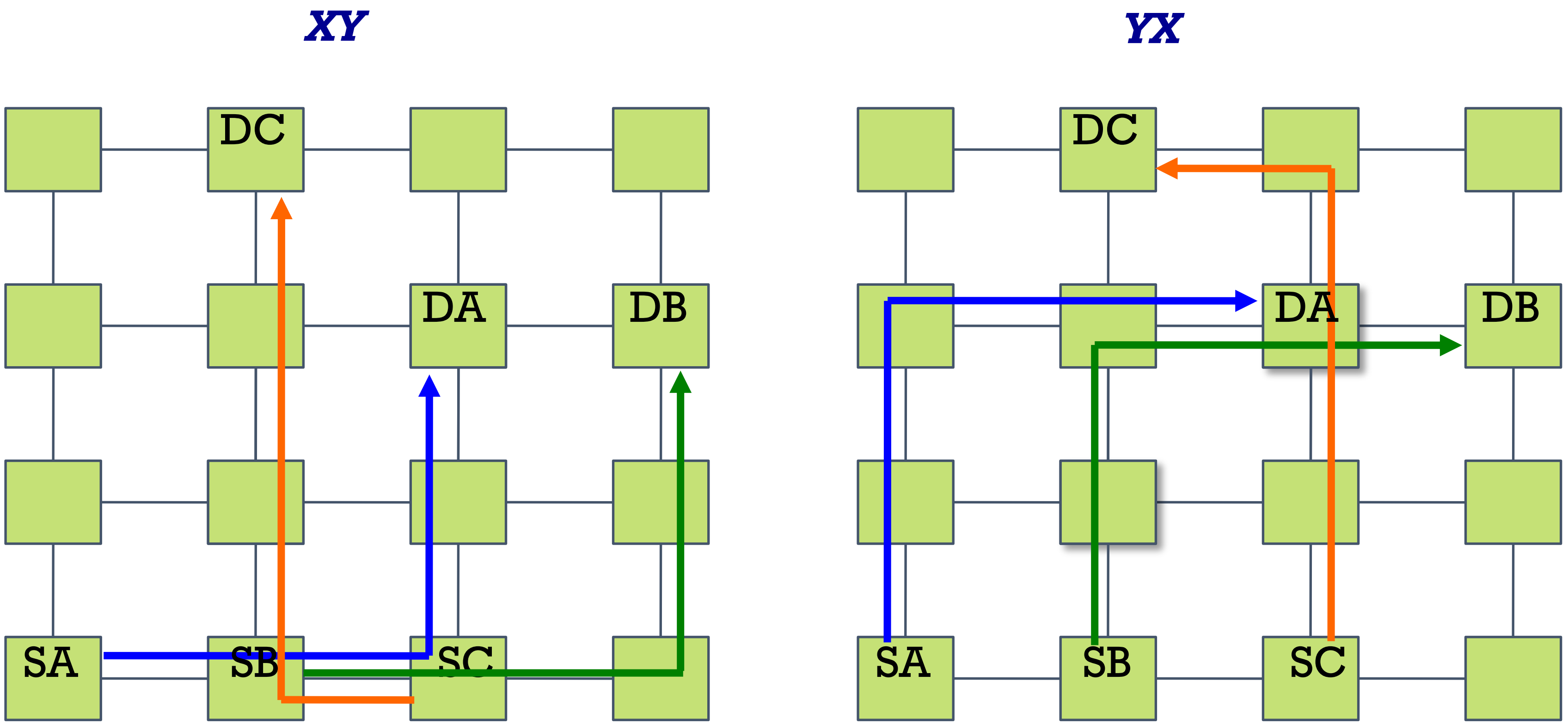
Cons of this approach?

- 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)

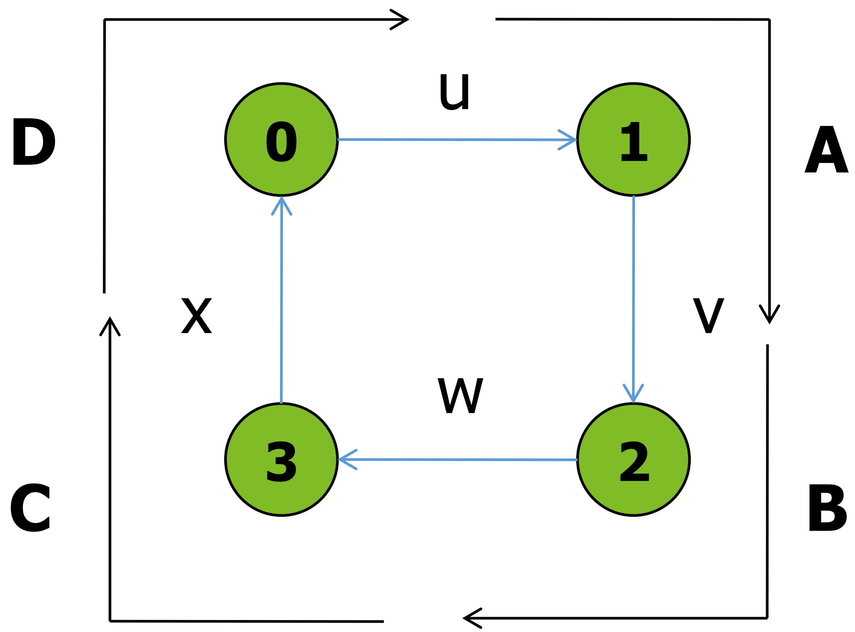


Randomly send over XY or YX

Minimal and Oblivious

Any problem?

NETWORK DEADLOCK

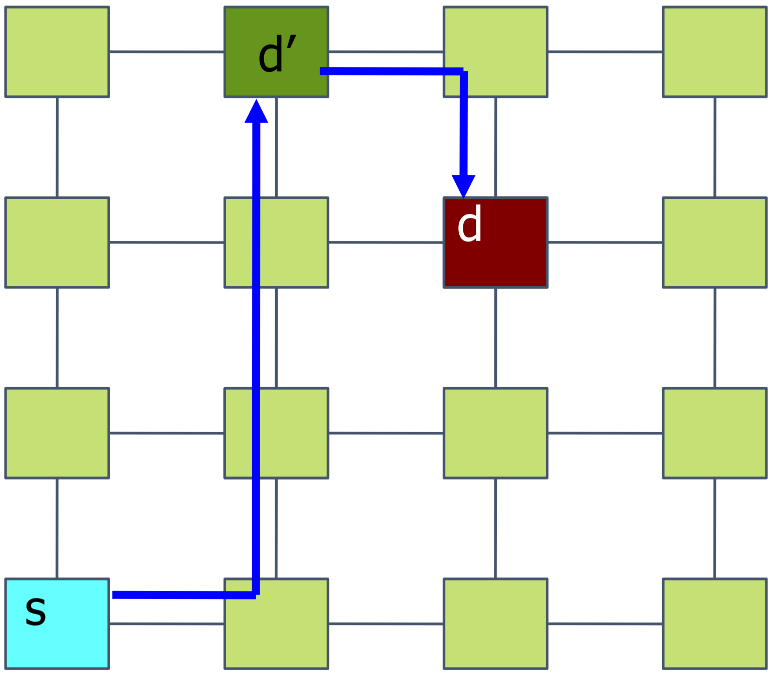


- Flow A holds u and wants v
- Flow B holds v and wants w
- Flow C holds w and wants x
- Flow D holds x and wants u

Next 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

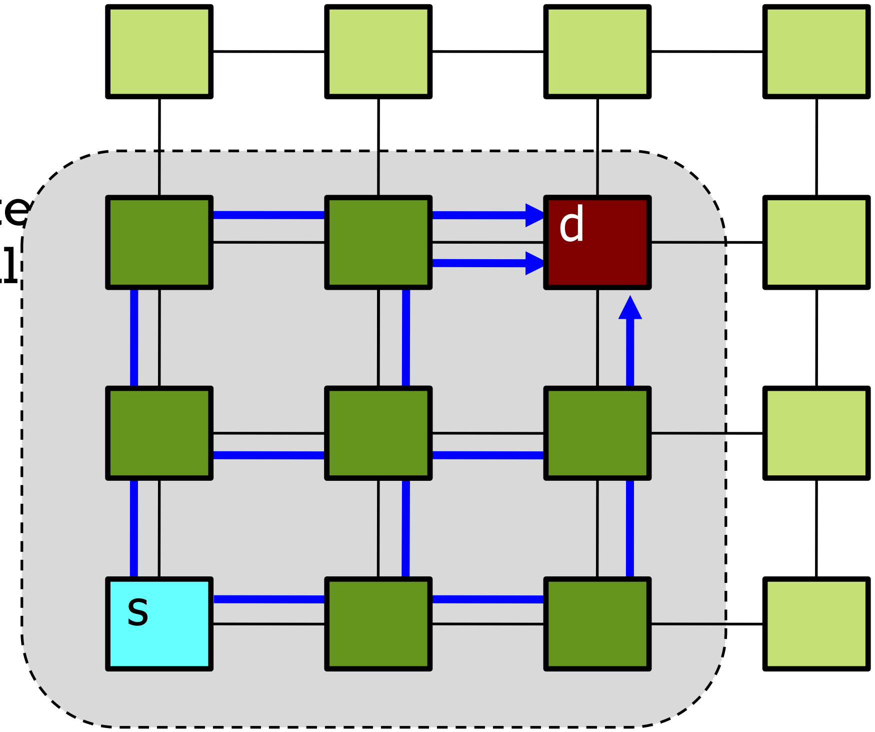


Non-Minimal and *Oblivious

can also be Adaptive

ROMM: RANDOMIZED, OBLIVIOUS MULTI-PHASE MINIMAL ROUTING

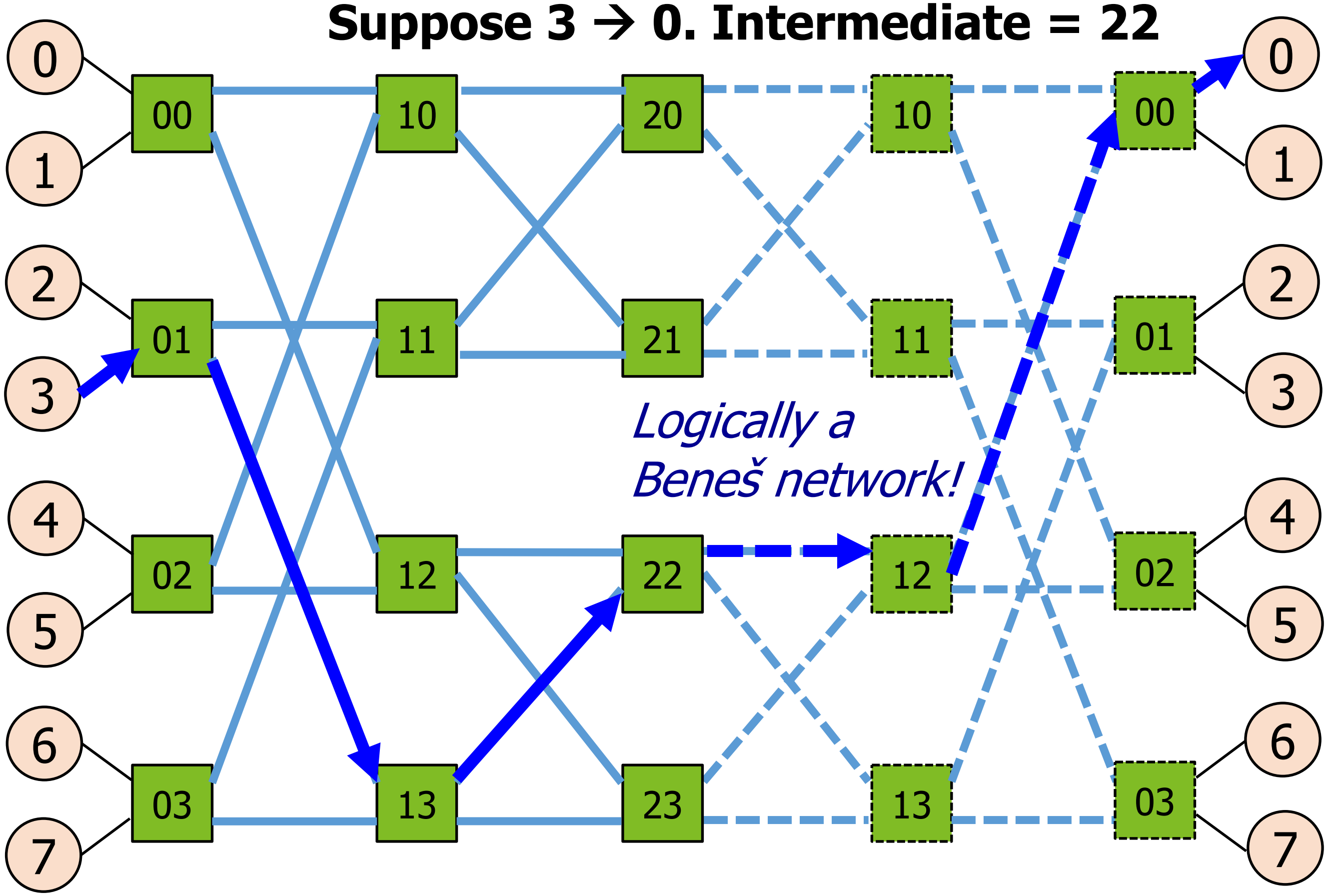
- Confine intermediate node to be within minimal quadrant
- Retain locality + some load-balancing
- This approach essentially translates to randomly selecting between all minimal paths from source to destination



Minimal and Oblivious

VALIANT'S ALGORITHM ON INDIRECT NETWORKS

Suppose 3 → 0. Intermediate = 22



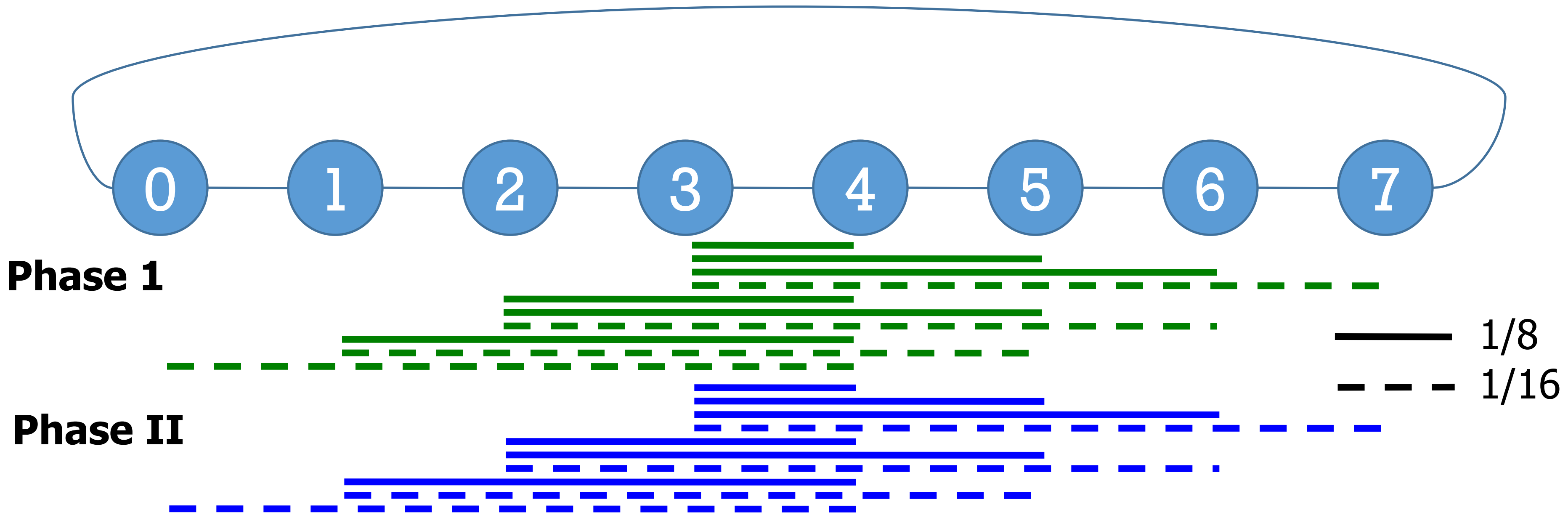
Two-phase Valiant routing equivalent to logically duplicating butterfly network

Can eliminate bottlenecks causes by certain traffic patterns. e.g., Traffic = {0,1,2,3}→ {0,1,2,3} leads to a channel load of 2 on top half of the links

Non-Minimal and Oblivious

	Max Channel Load	Throughput
Dest = (0, 1, 2, 3)	2	0.5
Valiant (Uniform Random)	1	1

VALIANT'S ON OUR RING FOR TORNADO?



	Greedy	Random	Adaptive	Valiant's
Max Channel Load	3	5/2 = 2.5	15/8= 1.875	2 (two phases)
Throughput	1/3 = 0.33	2/5 = 0.4	8/15 = 0.53	½ = 0.5

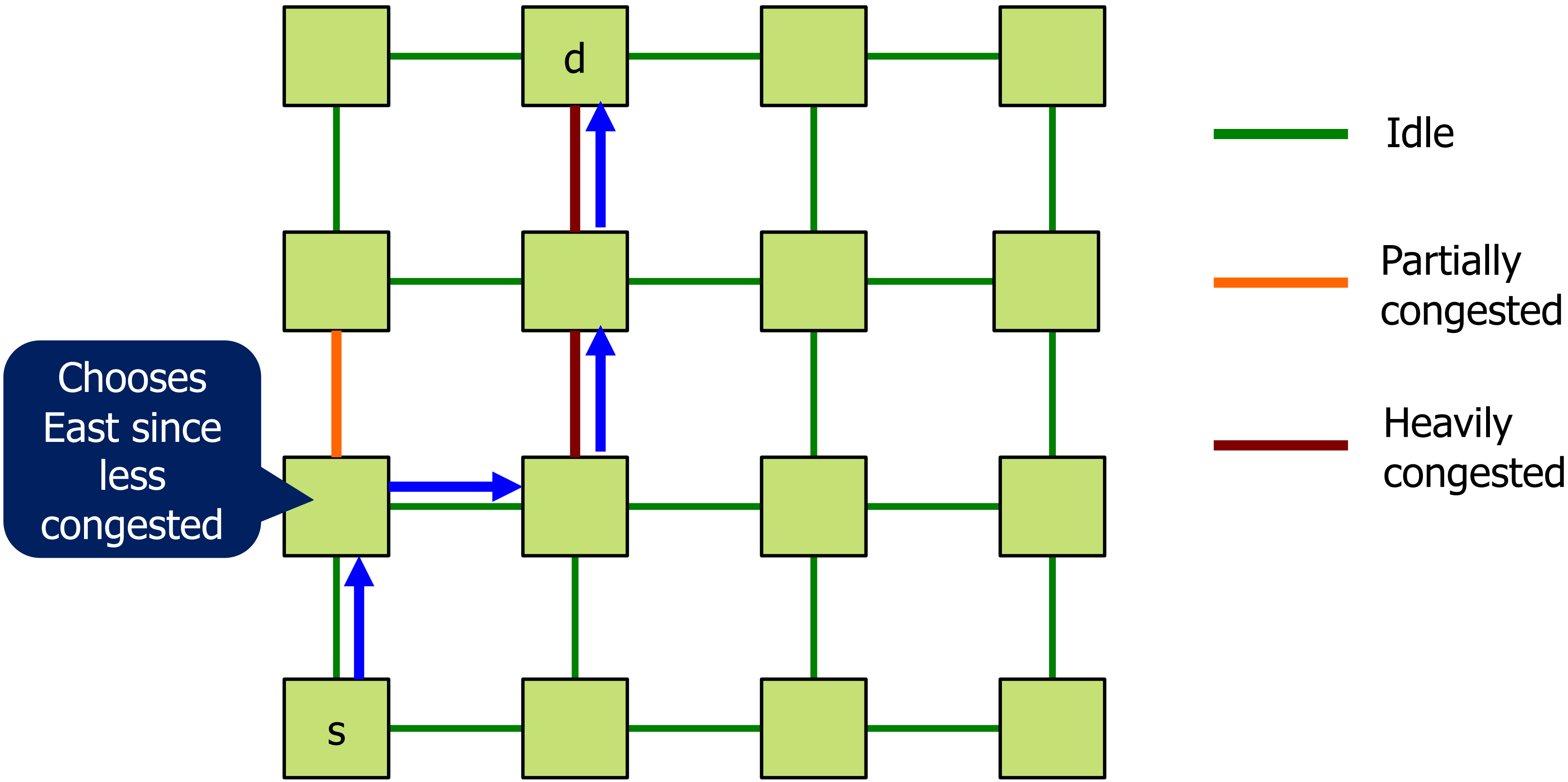
TAXONOMY OF ROUTING ALGORITHMS

- **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

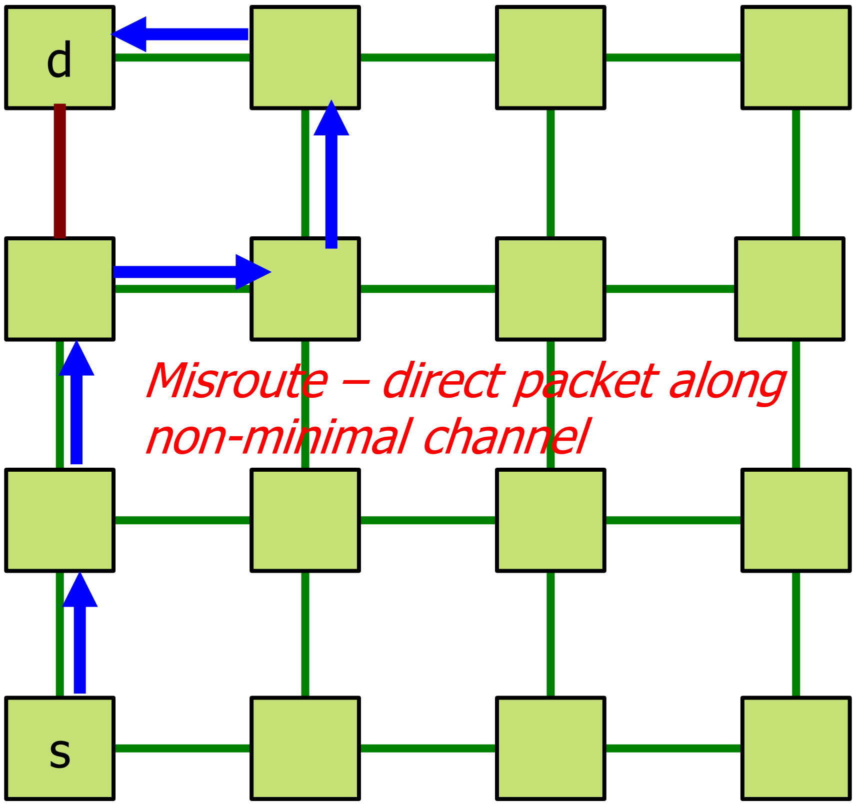
- Exploits path diversity
- Can be minimal or non-minimal
- Uses network state to make routing decisions
  - Buffer occupancies often used
  - Coupled with flow control mechanism
- Local information readily available
  - Global information more costly to obtain
  - Problems
    - Network state can change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

EXAMPLE 1: MINIMAL ADAPTIVE ROUTING

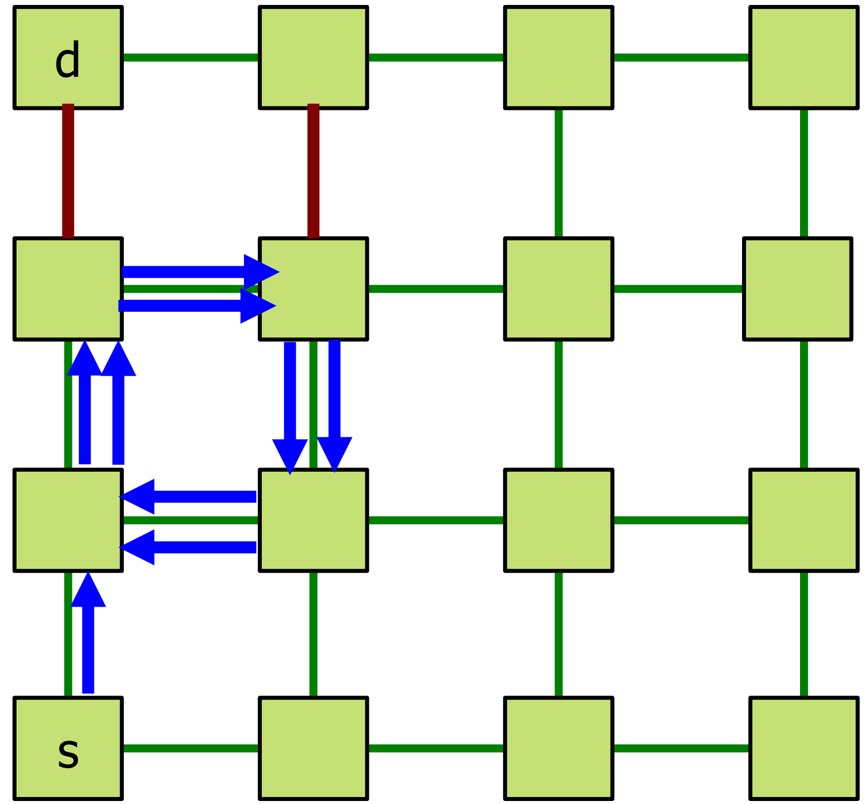


Local info can result in sub-optimal choices

EXAMPLE 2: NON-MINIMAL ADAPTIVE ROUTING



Longer path with potentially lower latency

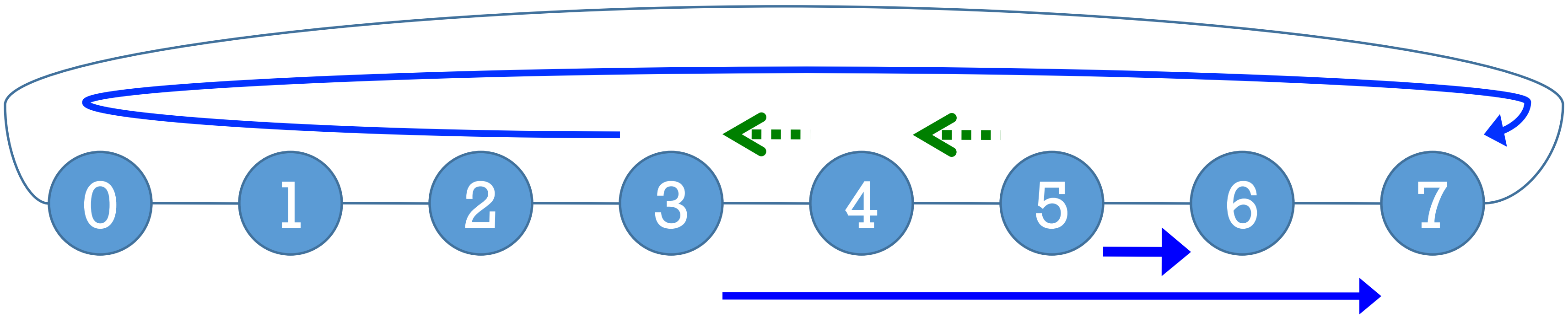


Livelock! – continue routing in cycle

To guarantee forward progress, limit number of misroutings

HOW TO SENSE CONGESTION?

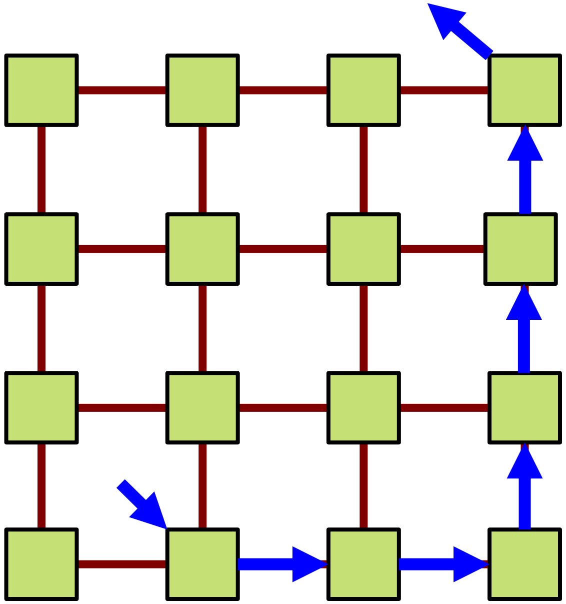
5→6 and 3→7



- **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 later in Flow Control lectures!

TAXONOMY OF ROUTING ALGORITHMS

- **Classification III – implementation**
  - **Source Routing:** embed entire route (i.e., list of output ports) in the packet
    - Example: (E, E, N, N, N, N, Eject)
    - Each router reads left most entry, and then strips it away for next hop
    - Pros
      - Save latency at each hop
      - Save routing-hardware at each hop
      - Can reconfigure routes based on faults
      - Supports irregular topologies
    - Cons
      - Overhead to store all routes at NIC
      - Overhead to carry routing bits in every packet (3-bits port x max hops)
      - Cannot adapt based on congestion



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.

THAT'S ALL FOR TODAY!

- What will be the combinational circuit / pseudo-code for generating the `output_port` for the XY routing algorithm in a Mesh at every hop?
  - Use the following signals:
    - **From Flit:** `x_hops_remaining`, `x_direction`, `y_hops_remaining`, `y_direction`