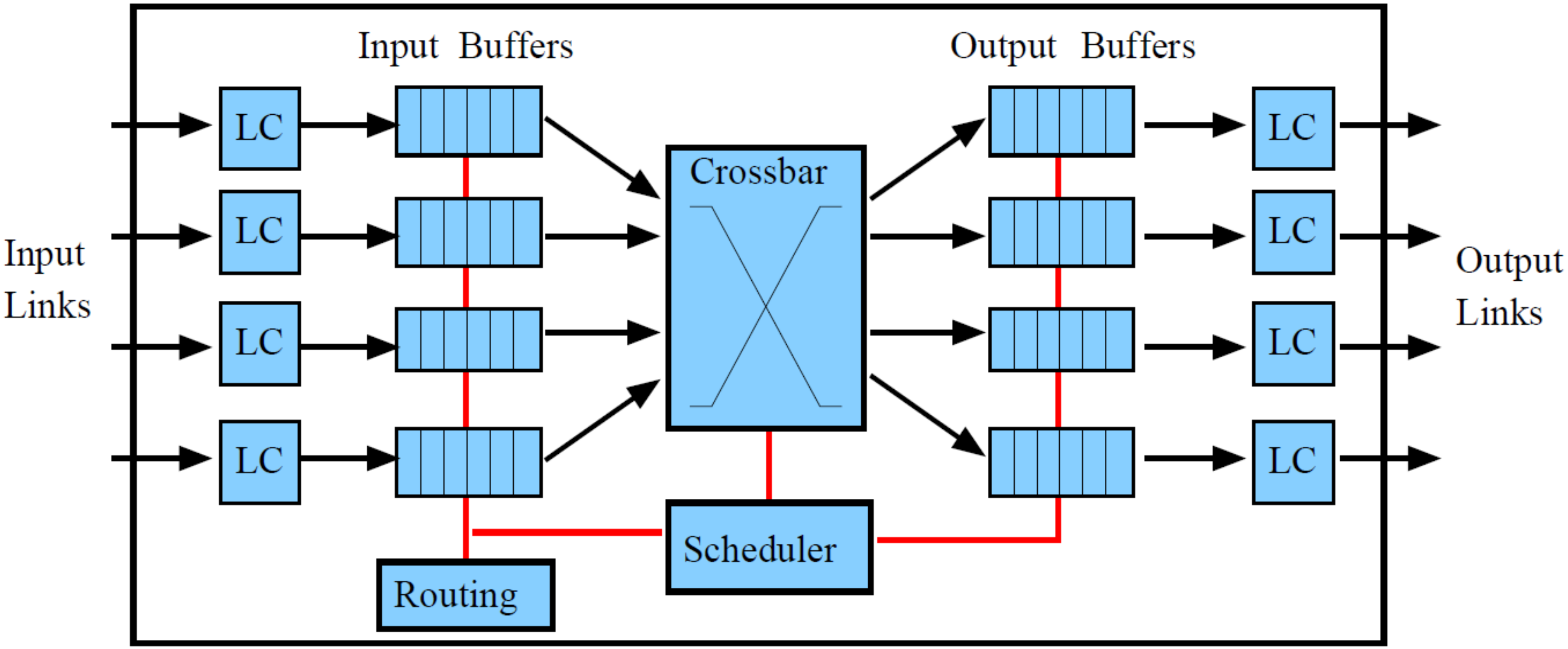
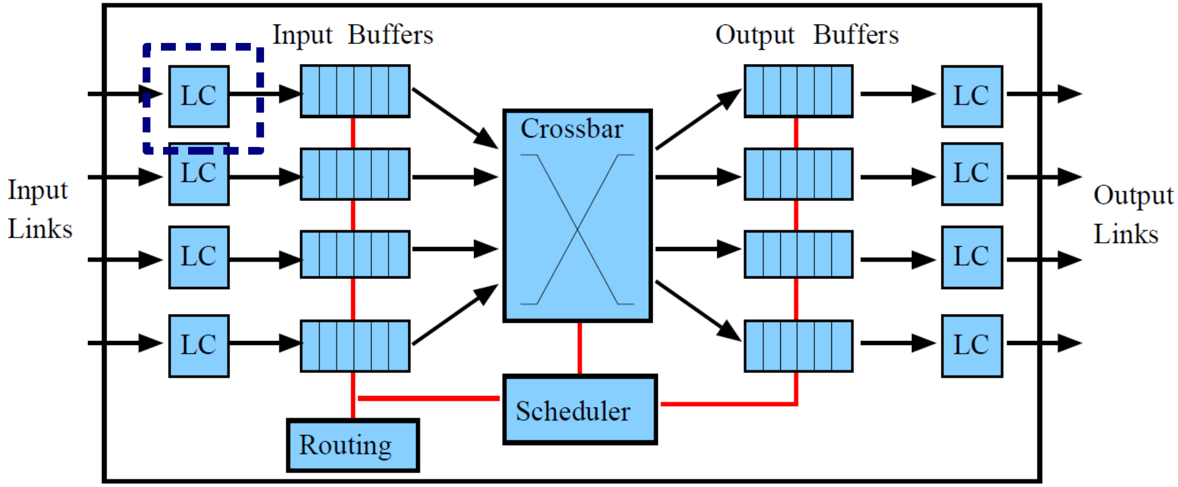
Routing

Basic Switch Organization

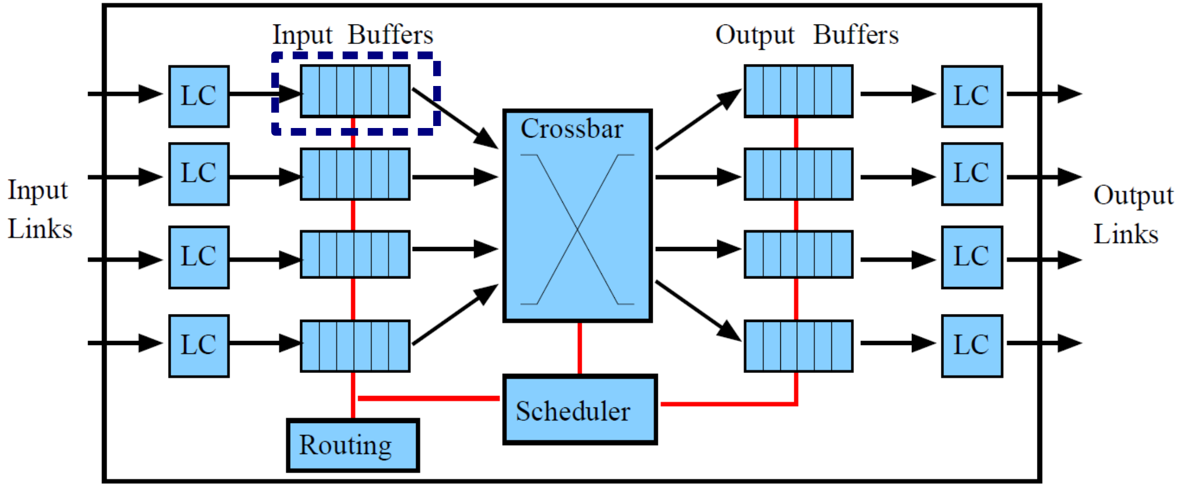


Basic Switch Organization



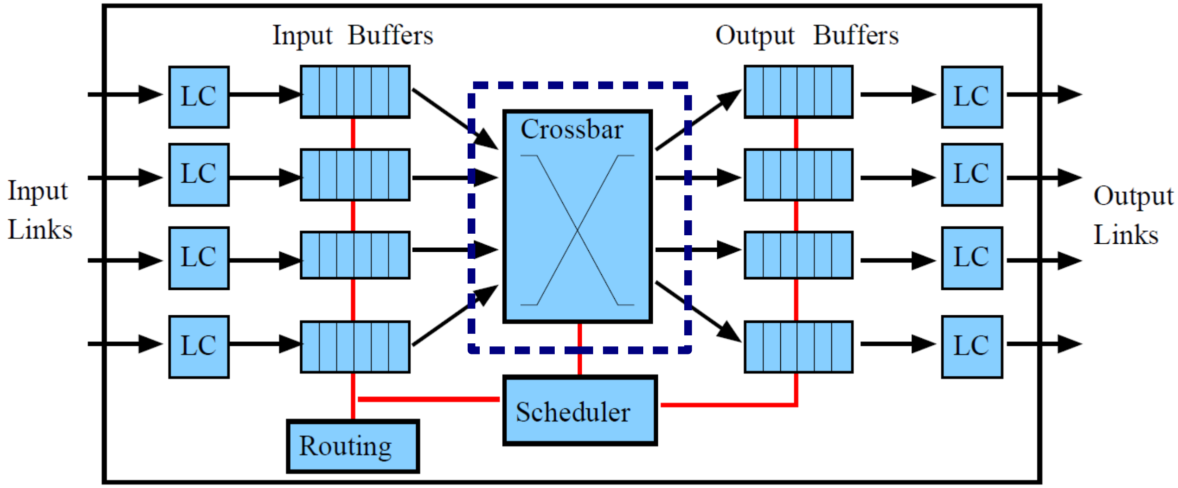
- Link Controller
  - ➔ Used for coordinating the flow of messages across the physical link of two adjacent switches

Basic Switch Organization



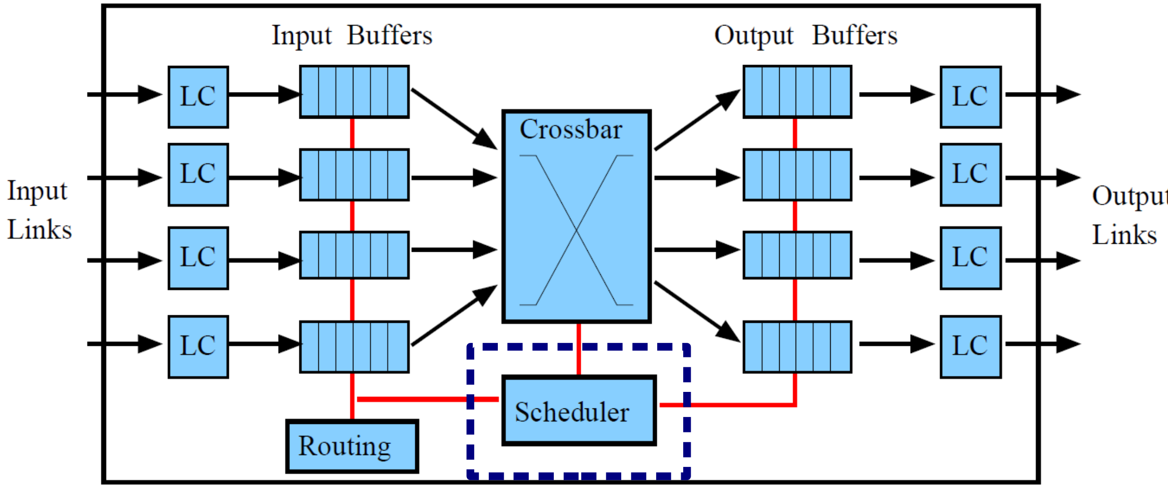
- Buffers
  - In charge of storing messages that go across the router
  - Usually built using first-in first-out (FIFO) queues

Basic Switch Organization



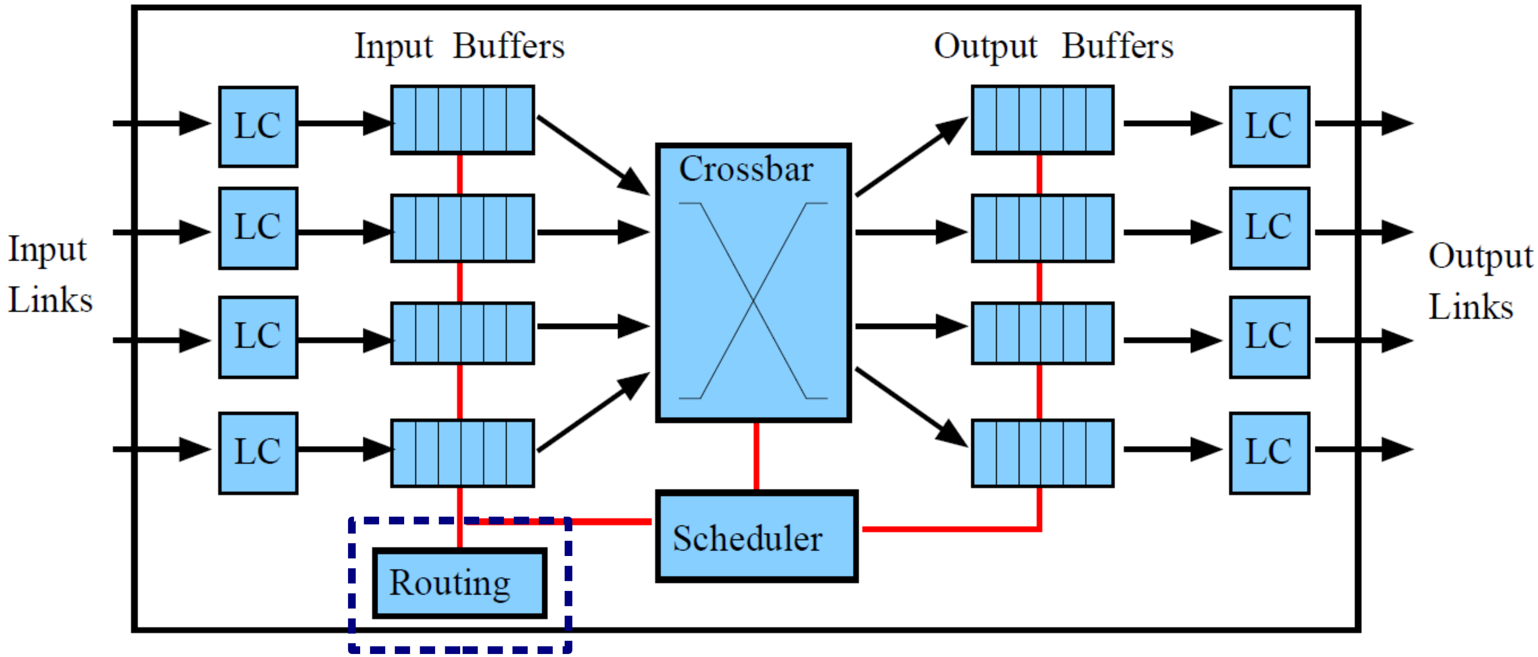
- Crossbar
  - ➔ Connects switch input buffers to switch output buffers

Basic Switch Organization



- Scheduler
  - ➔ The configuration of the crossbar is synchronously updated by a central scheduler
  - ➔ It matches the output port availabilities with the requests coming from the messages (or packets) located at the input ports
  - ➔ Conflicts for the same output port must be resolved
    - ✓ If the requested buffer is busy, the message (or packet) remains in the input buffer until the grant is assigned to the message

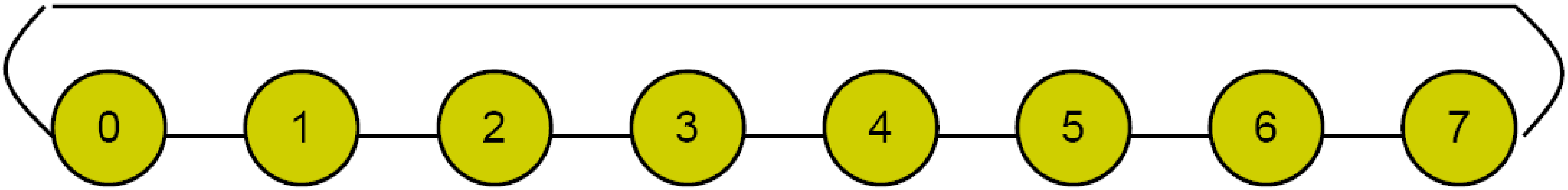
Basic Switch Organization



Routing

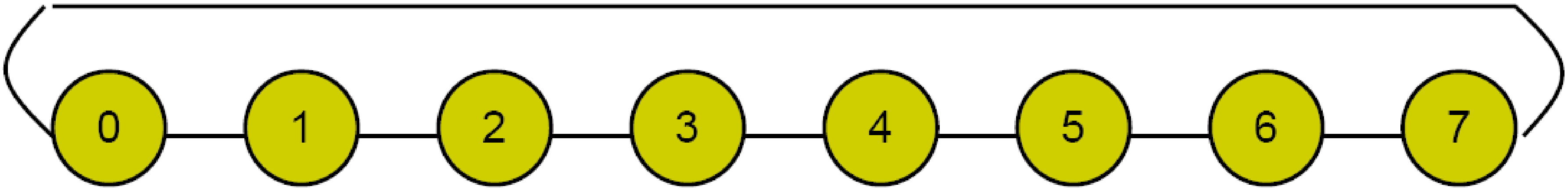
- The objective of routing is to find a path from a source node to a destination node on a given topology
- Routing is one of the key components that determine the performance of the network
- Objectives of a routing algorithm
  - Reduce the number of hops and overall latency
  - Balance the load of network channels
    - The more balanced the channel load → the closer the throughput of the network is to ideal

An Introductory Example



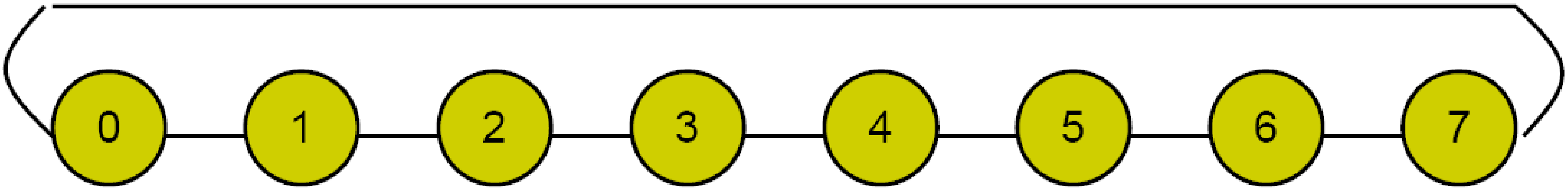
- There are only two directions a packet can take: clockwise and counterclockwise
- There are plenty of possible routing algorithms

An Introductory Example



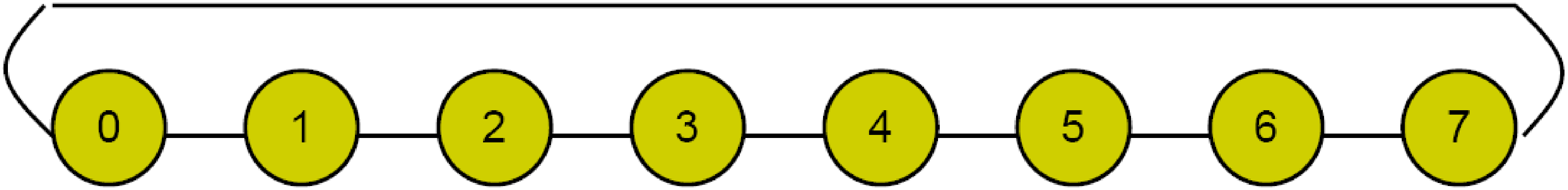
- *Greedy*: Always send a packet in the shortest direction, if the distance is same in both directions pick direction randomly

An Introductory Example



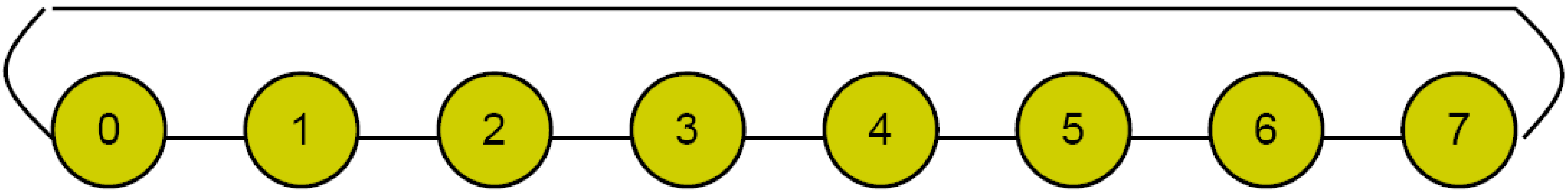
- *Uniform Random*: Send a packet randomly (p=0.5) either clockwise or counterclockwise

An Introductory Example



- *Weighted Random*: Randomly pick a direction for each packet, but weight the short direction with probability 1-*p*/8, and the long direction with *p*/8 where *p* is the minimum distance between source and destination

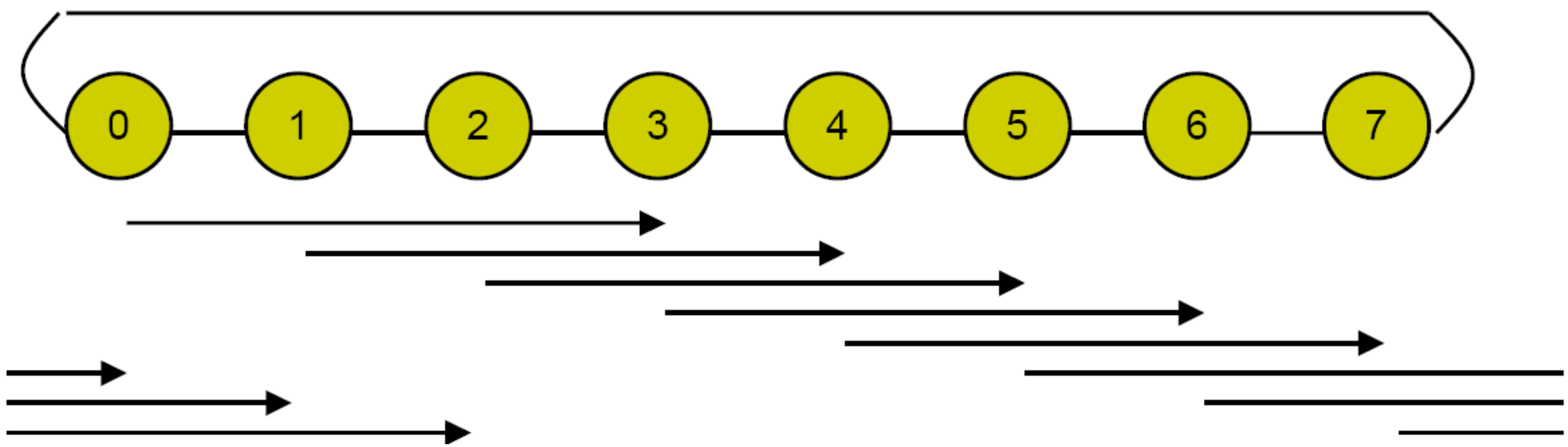
An Introductory Example



- *Adaptive*: Send the packet in the direction for which the local channel has the lowest load
  - ➔ This can be done by either
    - ✓ Measuring the lentgh of the queue in this direction
    - ✓ Recording how many packets a channel has transmitted over the last *T* slots

What is the Best Algorithm?

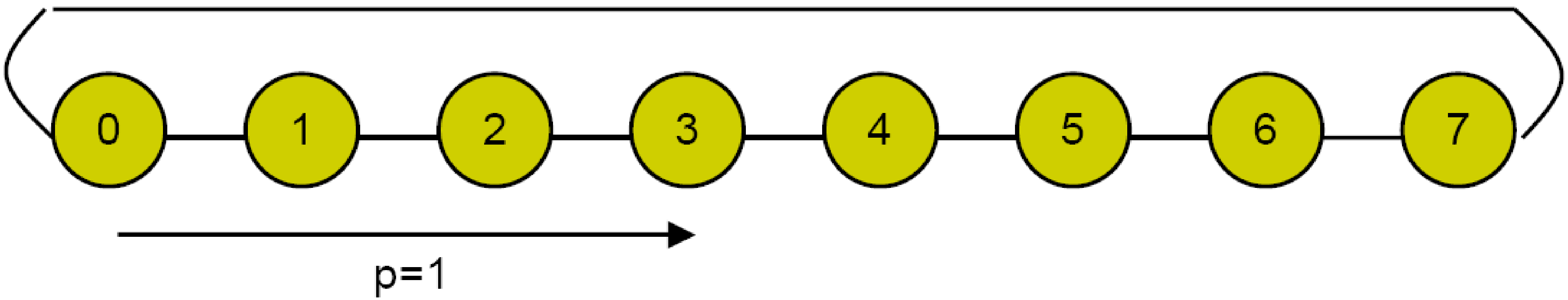
- Performance of Algorithm depends on Topology (and on the pattern)
- For further investigation assume the following traffic pattern (Tornado Traffic)
  - ➔ Every node *i* sends a packet to node *(i+3) mod 8*
  - ➔ 0→3, 1→4, …, 5→0, 6→1, 7→2



What is the Best Algorithm

Greedy Algorithm

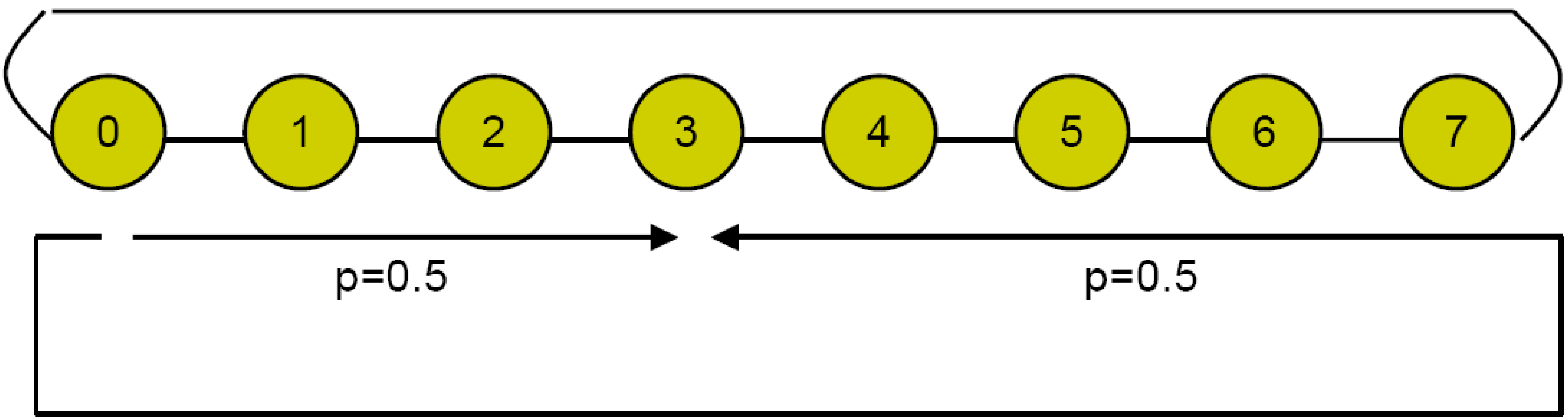
- No traffic is routed counterclockwise ➔ Bad utilization of channels
  - ➔ Thus γ = 3 and Θ = $b/3$



What is the Best Algorithm

Uniform Random Algorithm

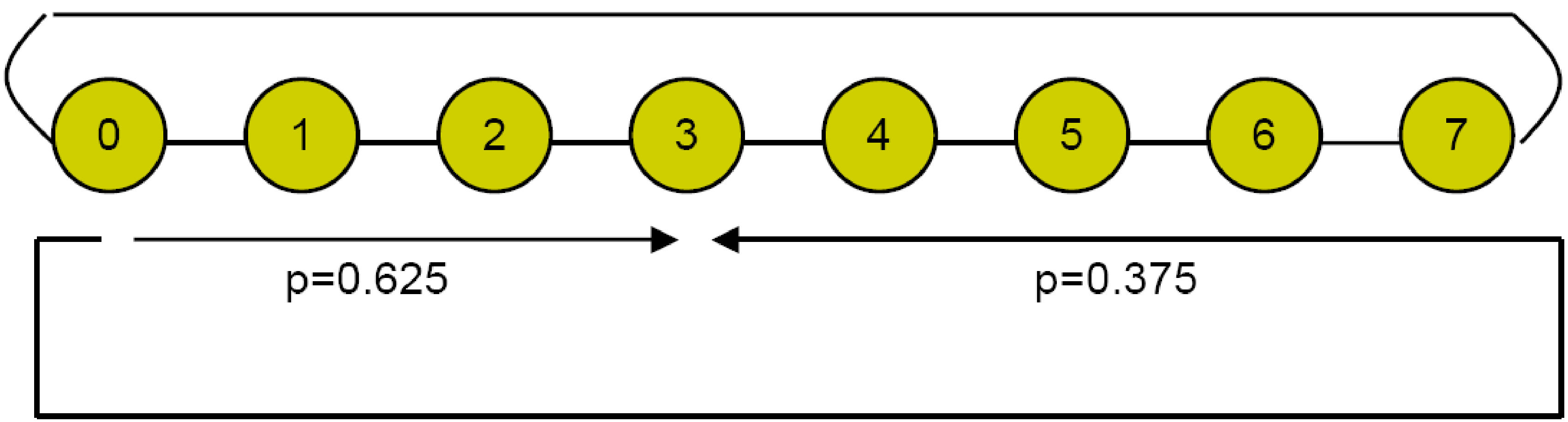
- Half of the traffic is going clockwise and half of the traffic counterclockwise
- Bottleneck counterclockwise traffic, where half of the traffic traverses five links
- γ = 5/2 and thus Θ = 2*b*/5



What is the Best Algorithm

Weighted Random Algorithm

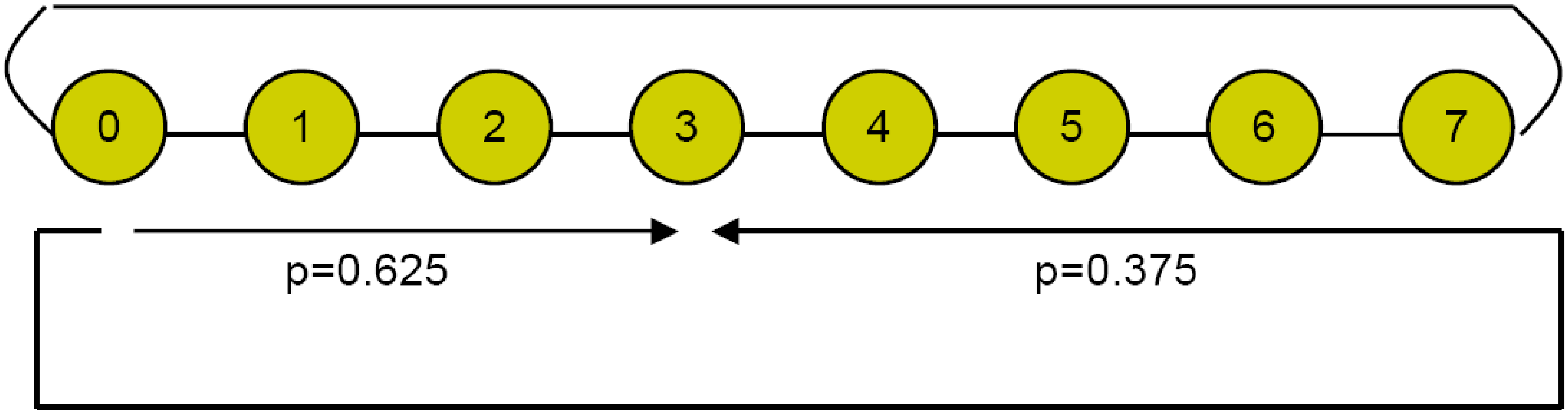
- 5/8 of the traffic is going clockwise (3 links) and 3/8 of the traffic counterclockwise (5 links)
- In both directions γ = 15/8 and thus Θ = 8b/15
- Perfect Load Balance!



What is the Best Algorithm

Adaptive Algorithm

- Adaptive routing will in a steady state match the perfect load
- Thus it will in theory allow in both directions γ = 15/8 and Θ = 8b/15



Taxonomy of Routing Algorithms

- Deterministic
- Oblivious
- Adaptive

- Minimal
- Non-minimal

Deterministic Routing Algorithms

- *Deterministic algorithms* always choose the same path between two nodes
  - → Easy to implement and to make deadlock free
  - → Do not use path diversity and thus bad on load balancing

Oblivious Routing Algorithms

- *Oblivious algorithms* always choose a route without knowing about the state of the networks state
  - ➔ All random algorithms are oblivious algorithms
  - ➔ All deterministic algorithms are oblivious algorithms

Adaptive Routing Algorithms

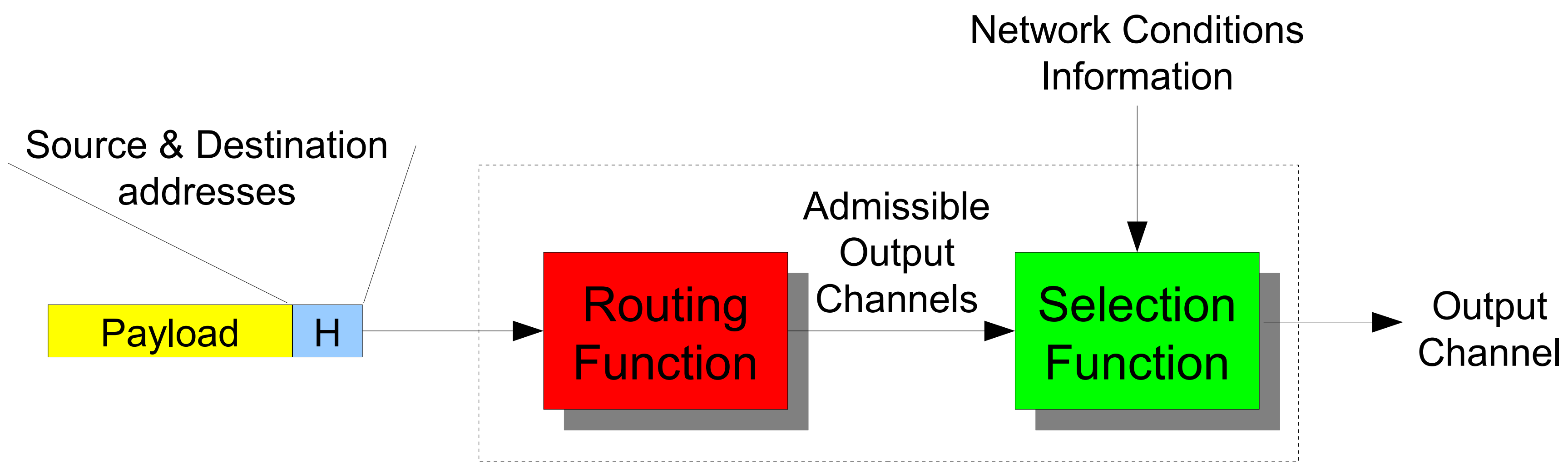
- *Adaptive algorithms* use information about the state of the network to make routing decisions
  - Length of queues
  - Historical channel load

Minimal vs. Non-minimal

- *Minimal algorithms* only consider minimal routes (shortest path)
- *Non-Minimal algorithms* allow even nonminimal routes

Routing Algorithms

- The routing algorithm can be represented as a *routing relation R selection function S*
- *R* returns a set of paths or channels and *S* selects between the route to be taken



Routing Algorithms

- The routing function R can be defined in three ways

$$R: N \times N \rightarrow \mathcal{P}(P)$$

$$R: N \times N \rightarrow \mathcal{P}(C)$$

$$R: C \times N \rightarrow \mathcal{P}(C)$$

Legend

N: set of nodes

C: set of channels

P: set of routing paths

$\mathcal{P}$: power set

Routing Algorithms

$$R: N \times N \rightarrow \mathcal{P}(P)$$

- All-at-once (a.k.a. *source routing*)
  - The routing relation takes source node and destination node as arguments and returns a set of possible paths
  - One of these paths is selected and assigned to the packet
  - The routing relation is only evaluated once at the source node

Routing Algorithms

$$R: N \times N \rightarrow P(C)$$

- Incremental Routing
  - ➔The routing relation takes the current node and the destination node as arguments and returns a set of possible channels
  - ➔One of these channels is selected and the packet will be forwarded via this channel
  - ➔The routing relation is evaluated for every hop until the packet arrives at its final destination

Routing Algorithms

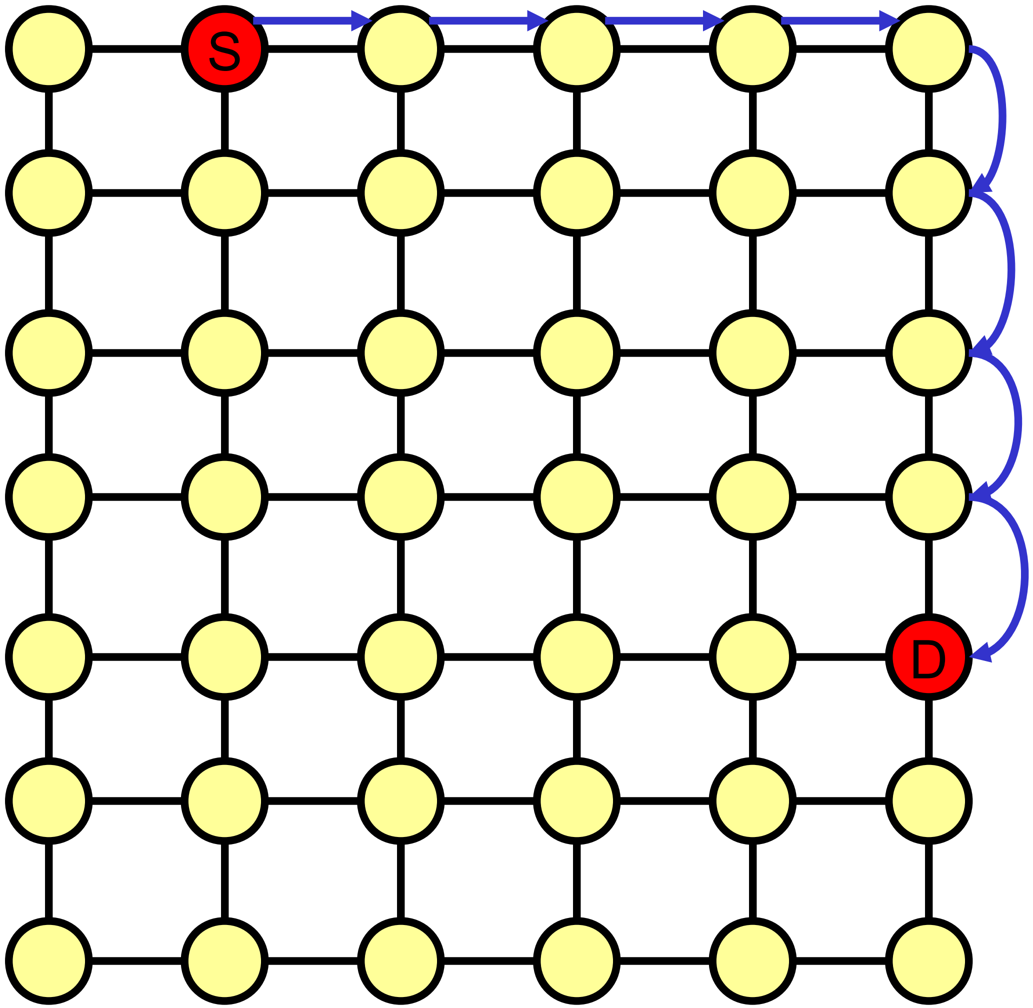
$$R: C \times N \rightarrow P(C)$$

- Incremental Routing
  - The routing relation takes the previous channel and the destination node as arguments and returns a set of possible channels
  - One of these channels is selected and the packet will be forwarded via this channel
  - The routing relation is evaluated for every hop until the packet arrives at its final destination
  - Since the previous channel is considered there is history information that is useful, for instance to avoid deadlock

Deterministic Routing

- A packet from a source node *x* to a destination node *y* is always sent over exactly the same route
- *Advantages*
  - ➔Simple and inexpensive to implement
  - ➔Usual deterministic routing is minimal, which leads to short path length
  - ➔Packets arrive in order
- *Disadvantage*
  - ➔Lack of path diversity can create large load imbalances

Example: Deterministic XY Routing



#Paths =

$$\frac{(\Delta_x + \Delta_y)!}{\Delta_x! \times \Delta_y!}$$

where

$\Delta_x = |S_x - D_x|$

$\Delta_y = |S_y - D_y|$

Summary

- The routing algorithm plays a very important role for the performance of a network
- Load Balancing is often a critical factor
- Deterministic routing is a simple and inexpensive routing algorithm, but does not utilize path diversity and thus is very weak on load balancing