Routing, Routers, Switching Fabrics

Outline

- Link state routing
- Link weights
- Router Design / Switching Fabrics

Link State Routing Summary

- One of the oldest algorithm for routing
- Finds SP by developing paths in order of increasing length
  - Requires each node to have complete information about the network
  - Nodes exchange information with all other nodes in the network
  - Known to converge quickly under static conditions
  - ~~Does not generate much network traffic~~
  - Converges quickly -- hence data packets are not stuck in the loops in the network for too long
- Other possible routing algorithms?

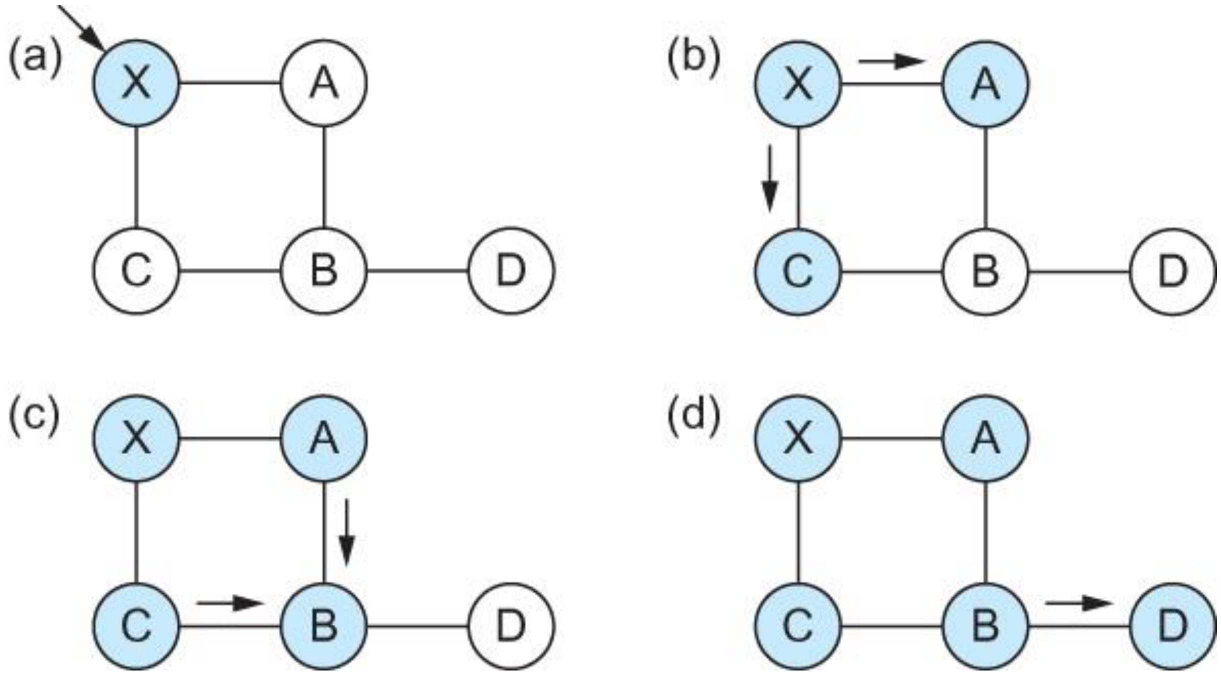
Link State Routing

Strategy: Send to all nodes (not just neighbors) information about directly connected links (not entire routing table).

- Link State Packet (LSP)
  - id of the node that created the LSP
  - cost of link to each directly connected neighbor
  - sequence number (SEQNO)
  - time-to-live (TTL) for this packet
- Reliable Flooding
  - store most recent LSP from each node
  - forward LSP to all nodes but one that sent it
- Generate new LSP periodically; increment SEQNO
  - start SEQNO at 0 when reboot
  - decrement TTL of each stored LSP; discard when TTL=0

Link State

Reliable Flooding



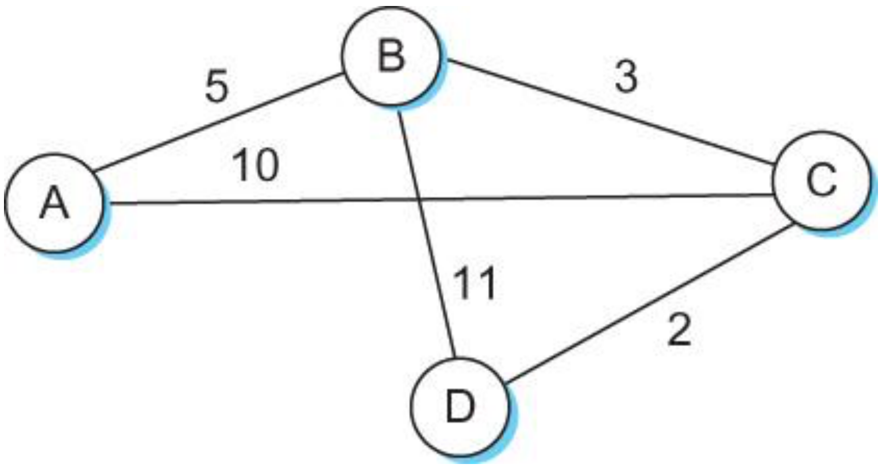
Flooding of link-state packets. (a) LSP arrives at node X; (b) X floods LSP to A and C; (c) A and C flood LSP to B (but not X); (d) flooding is complete

Shortest Path Routing

- Dijkstra's Algorithm - Assume non-negative link weights
  - N: set of nodes in the graph
  - l((i, j): the non-negative cost associated with the edge between nodes i, j ∈N and l(i, j) = ∝ if no edge connects i and j
  - Let s ∈N be the starting node which executes the algorithm to find shortest paths to all other nodes in N
  - Two variables used by the algorithm
    - M: set of nodes incorporated so far by the algorithm
    - C(n) : the cost of the path from s to each node n
    - The algorithm

```
M = {s}
For each n in N - {s}
    C(n) = l(s, n) /* costs of directly connected nodes */
while ( N ≠ M)
  M = M U {w} such that C(w) is the minimum
            for all w in (N-M) /* add a node */
  For each n in (N-M)        /* recalculate costs */
        C(n) = MIN (C(n), C(w) + l(w, n))
```

Routing Table for Node A



C(N)	B	C	D
M={A}	5	10	INFINITY
M={A, B}	5	8	16
M={A,B,C}	5	8	10

Open Shortest Path First (OSPF)

0	8	16	31
Version	Type	Message length	
SourceAddr			
AreaId			
Checksum		Authentication type	
Authentication			

OSPF Header Format

LS Age			Options	Type=1
Link-state ID				
Advertising router				
LS sequence number				
LS checksum			Length	
0	Flags	0	Number of links	
Link ID				
Link data				
Link type		Num_TOS	Metric	
Optional TOS information				
More links				

OSPF Link State Advertisement

In a Nutshell

- OSPF (LS) vs DV
  - DV
    - Slow convergence
    - Race Conditions
  - LS
    - High messaging overhead

Introduction to switching fabrics

- Switches must not only determine routing but also do forwarding quickly and efficiently
  - If this is done on a general purpose computer, the I/O bus limits performance
    - This means that a system with 1Gbps I/O could not handle OC12
  - Special purpose hardware is required
  - Switch capabilities drive protocol decisions
- Context – a “router” is defined as a datagram “switch”
- Switching fabrics are internal to routers and facilitate forwarding

Goals in switch design

- Throughput
  - Ability to forward as many pkts per second as possible
- Size
  - Number of input/output ports
- Cost
  - Minimum cost per port
- Functionality
  - QoS

Throughput

- Consider a switch with n inputs and m outputs and link speed of s_n
  - Typical notion of throughput: Σs_n
    - This is an upper bound
    - Assumes all inputs get mapped to a unique output
  - Another notion of throughput is packets per second (pps)
    - Indicates how well switch handles fixed overhead operations
- Throughput depends on traffic model
  - Goal is to be representative
  - This is VERY tricky!

Size/Scalability/Cost

- Maximum size is typically limited by HW constraints
  - Eg. fanout
- Cost is related to number of inputs/outputs
  - How does cost scale with inputs/outputs?

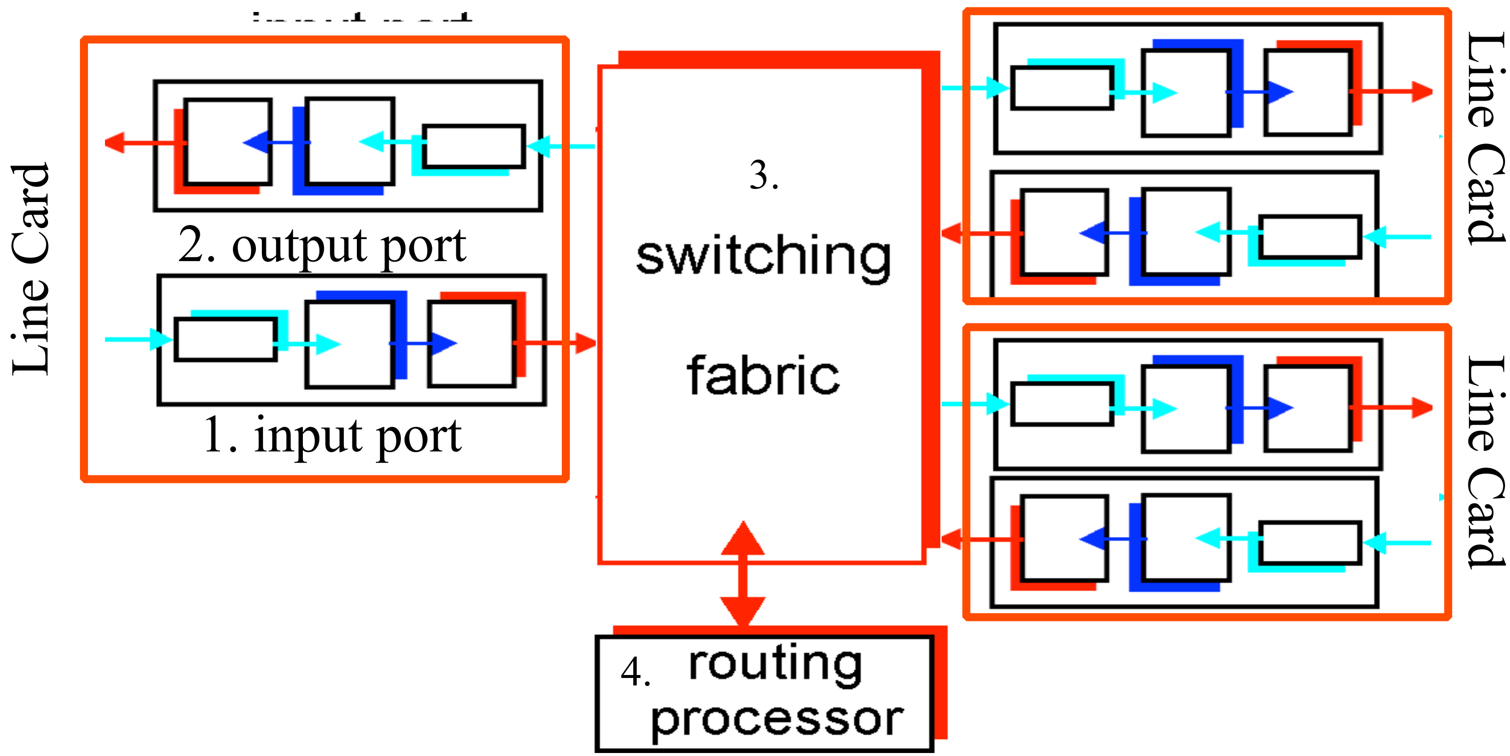
Ports and Fabrics

- *Ports* on switches handle the difficult functions of signaling, buffering, circuits, RED, etc.
  - Most buffering is via FIFO on output ports
    - This prevents head-of-the-line blocking on input ports which is possible if only one input port can forward to one output port at a time
- *Switching fabrics* in switches handle the simple function of forwarding data from input ports to output ports
  - Typically fabric does not buffer (but it can)
  - Contention is an issue
  - Many different designs
  - More details coming up

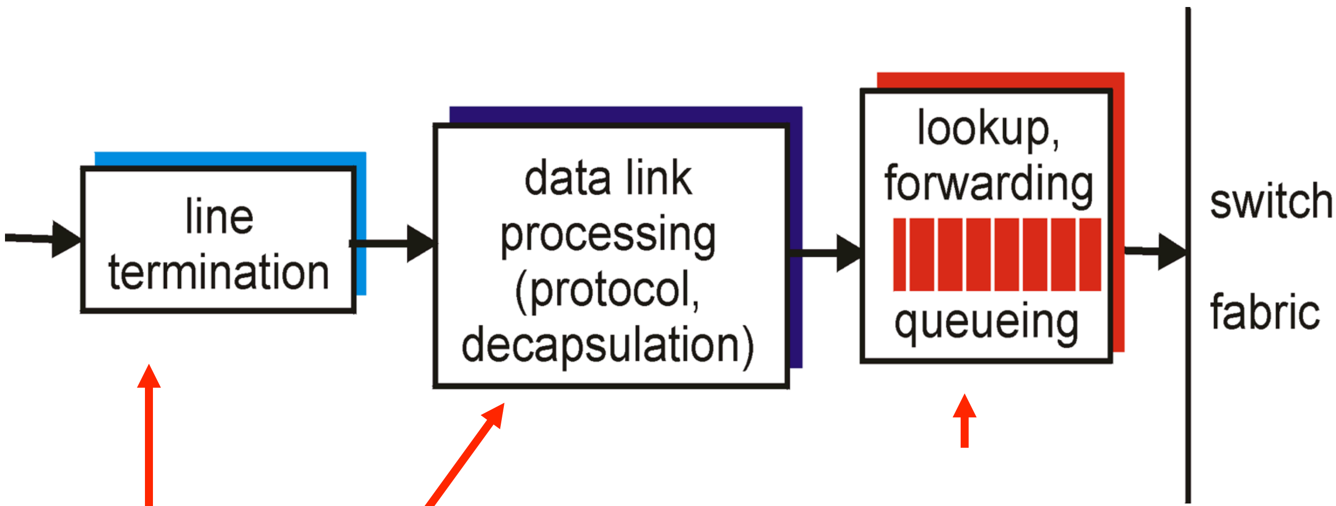
Router Architecture Overview

Two key router functions:

- Run routing algorithms/protocol (RIP, OSPF, BGP)
- *Switching* datagrams from incoming to outgoing link



Line Card: Input Port



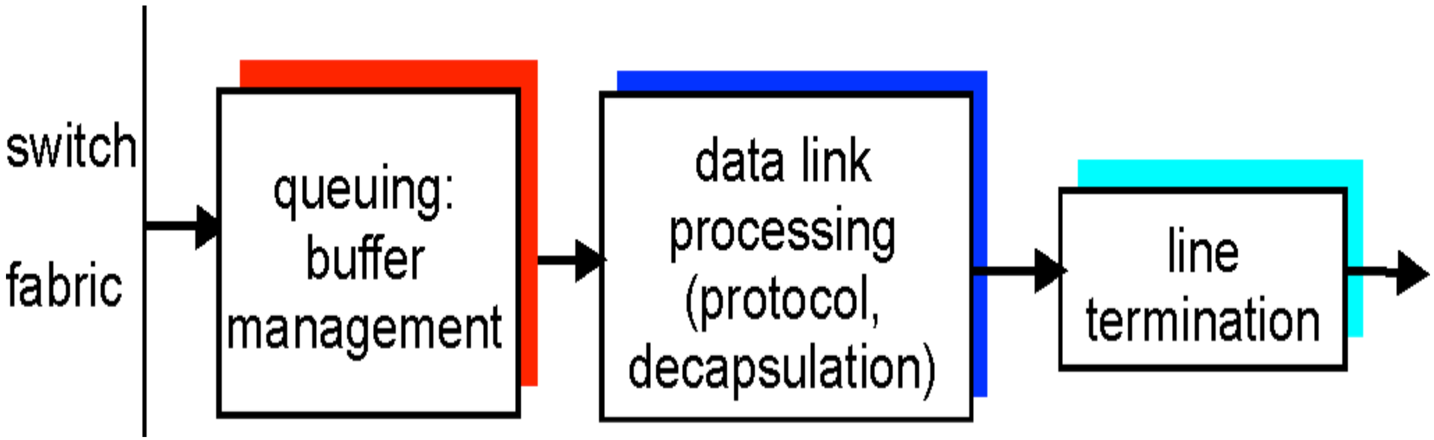
Physical layer:
bit-level reception

Data link layer:
e.g., Ethernet

Decentralized switching:

- Process common case ("fast-path") packets
  - Decrement TTL, update checksum, forward packet
- Given datagram dest., lookup output port using routing table in input port memory
- Queue needed if datagrams arrive faster than forwarding rate into switch fabric

Line Card: Output Port



- Queuing required when datagrams arrive from fabric faster than the line transmission rate

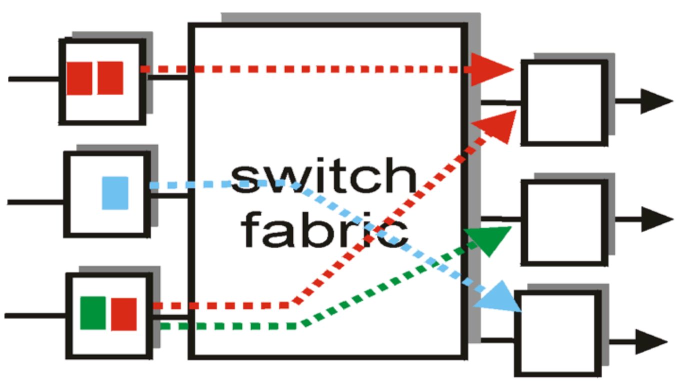
Buffering

- 3 types of buffering
  - Input buffering
    - Fabric slower than input ports combined → queuing may occur at input queues
  - Output buffering
    - Buffering when arrival rate via switch exceeds output line speed
  - Internal buffering
    - Can have buffering inside switch fabric to deal with limitations of fabric

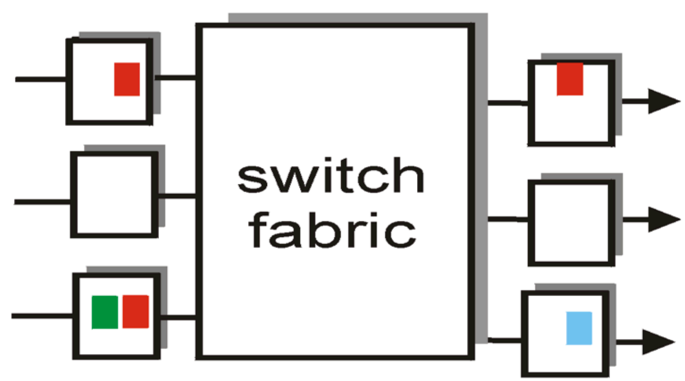
- What happens when these buffers fill up?
  - Packets are THROWN AWAY!! This is where (most) packet loss comes from

Input Port Queuing

- Which inputs are processed each slot – schedule?
- Head-of-the-Line (HOL) blocking: datagram at front of queue prevents others in queue from moving forward

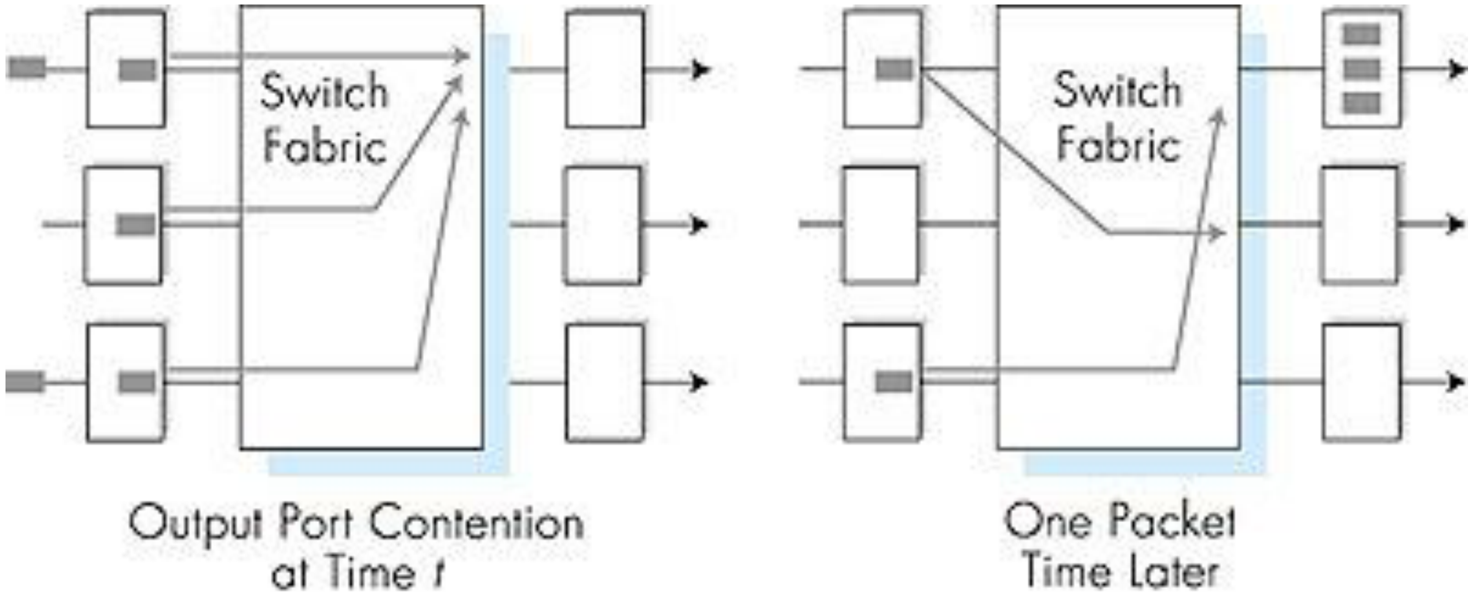


output port contention at time t - only one red packet can be transferred



green packet experiences HOL blocking

Output Port Queuing



- Scheduling discipline chooses among queued datagrams for transmission
  - Can be simple (e.g., first-come first-serve) or more clever (e.g., weighted round robin)

Virtual Output Queuing (VOQ)

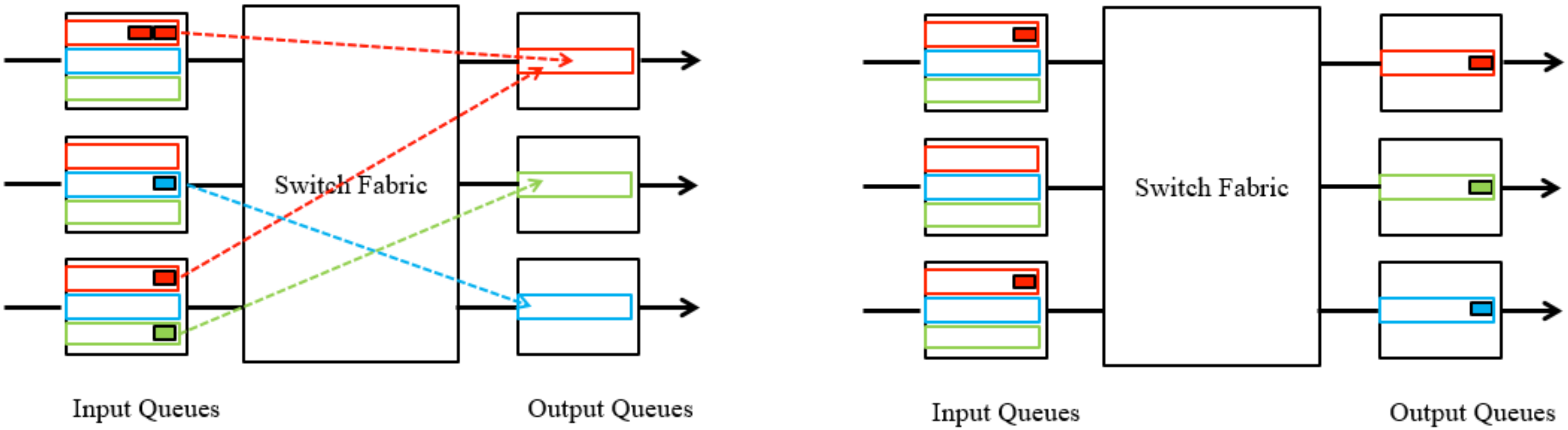


Figure: No head-of-line blocking for green packet even though red packet on the same input port was not scheduled

- Input queue has several virtual queues
  - Instead of a single FIFO queue, maintain a separate queue for each output port
  - Advantage: no head-of-line blocking. (see figure)