UNIVERSITY OF WATERLOO

CS 456/656
Computer Networks

Lecture 18: Router/Switch Architecture

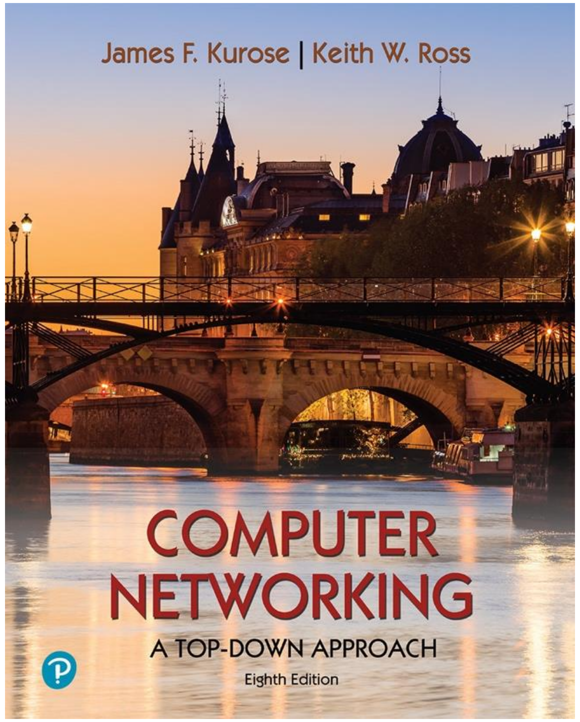
Mina Tahmasbi Arashloo and Uzma Maroof

Fall 2025

A note on the slides

Adapted from the slides that accompany this book.

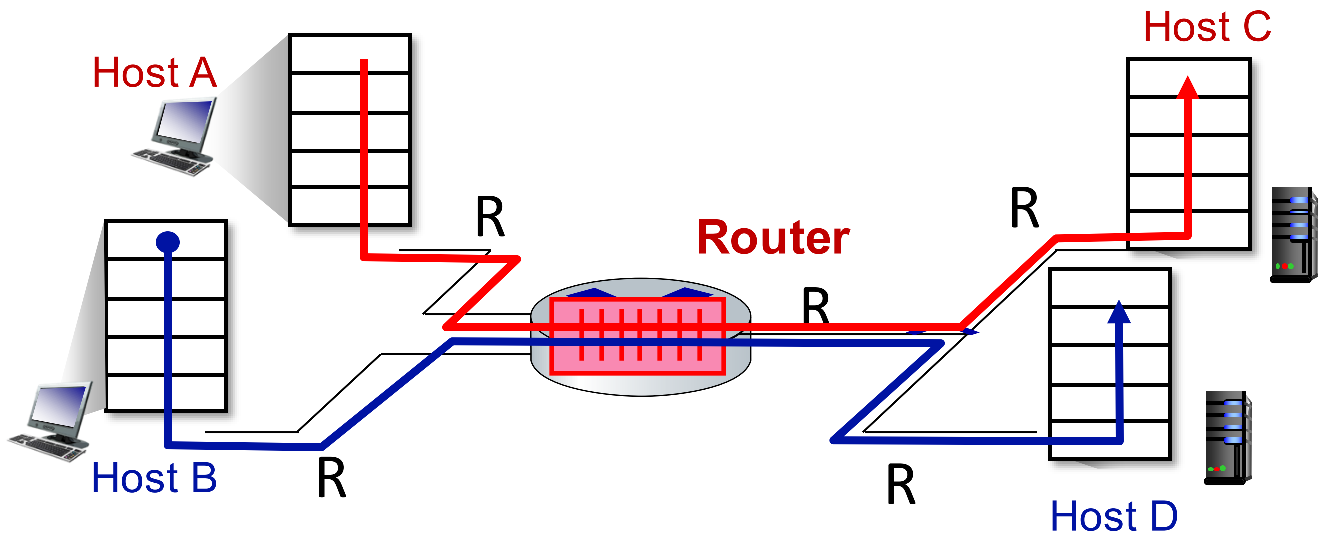
And lecture notes from Anirudh Sivaraman, NYU



Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

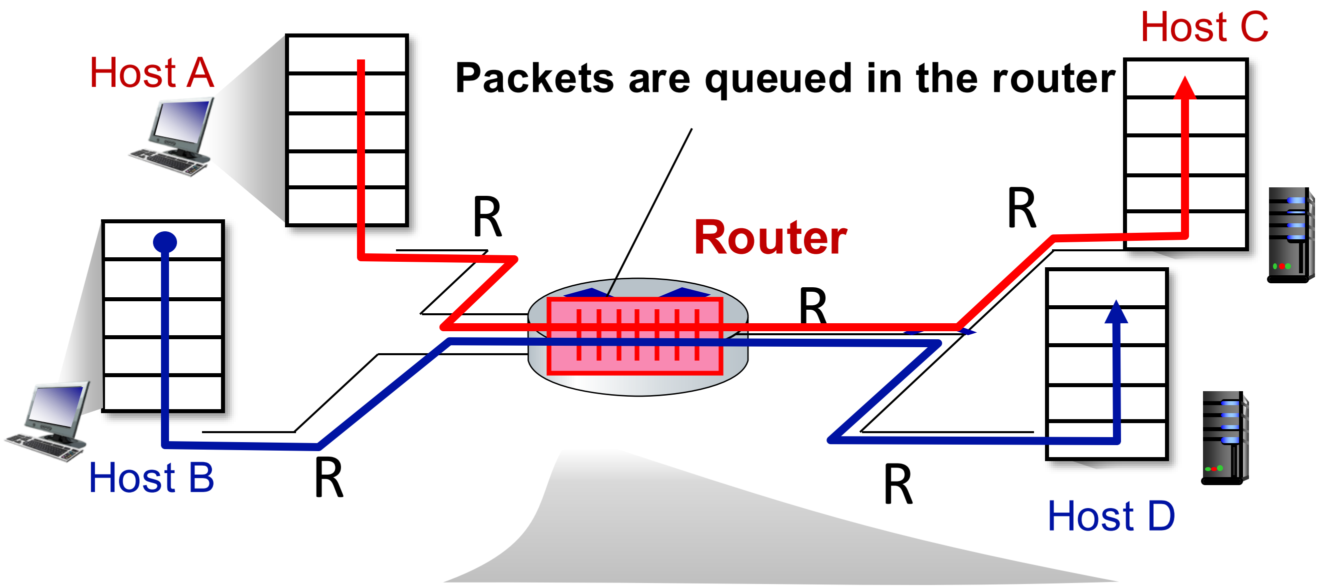
What we discussed before

- Packets can be buffered in routers
  - delay and loss
  - network congestion



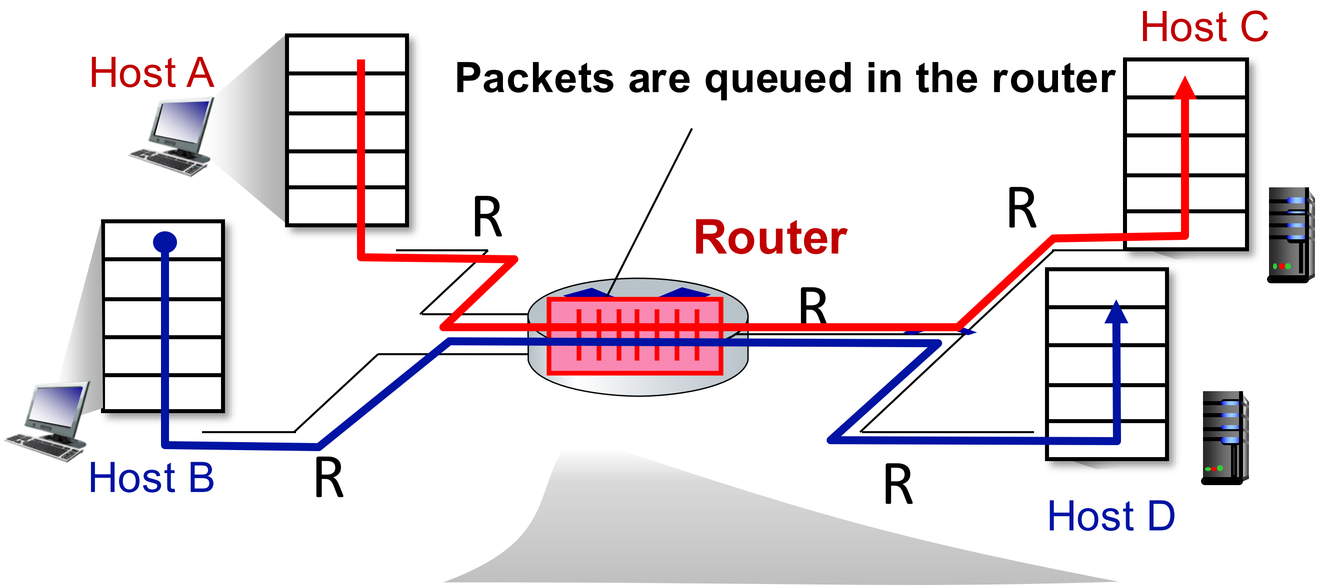
What we discussed before

- Packets can be buffered in routers
  - delay and loss
  - network congestion



What we discussed before

- Packets can be buffered in routers
  - delay and loss
  - network congestion



This Lecture: How are packets buffered and managed in routers?

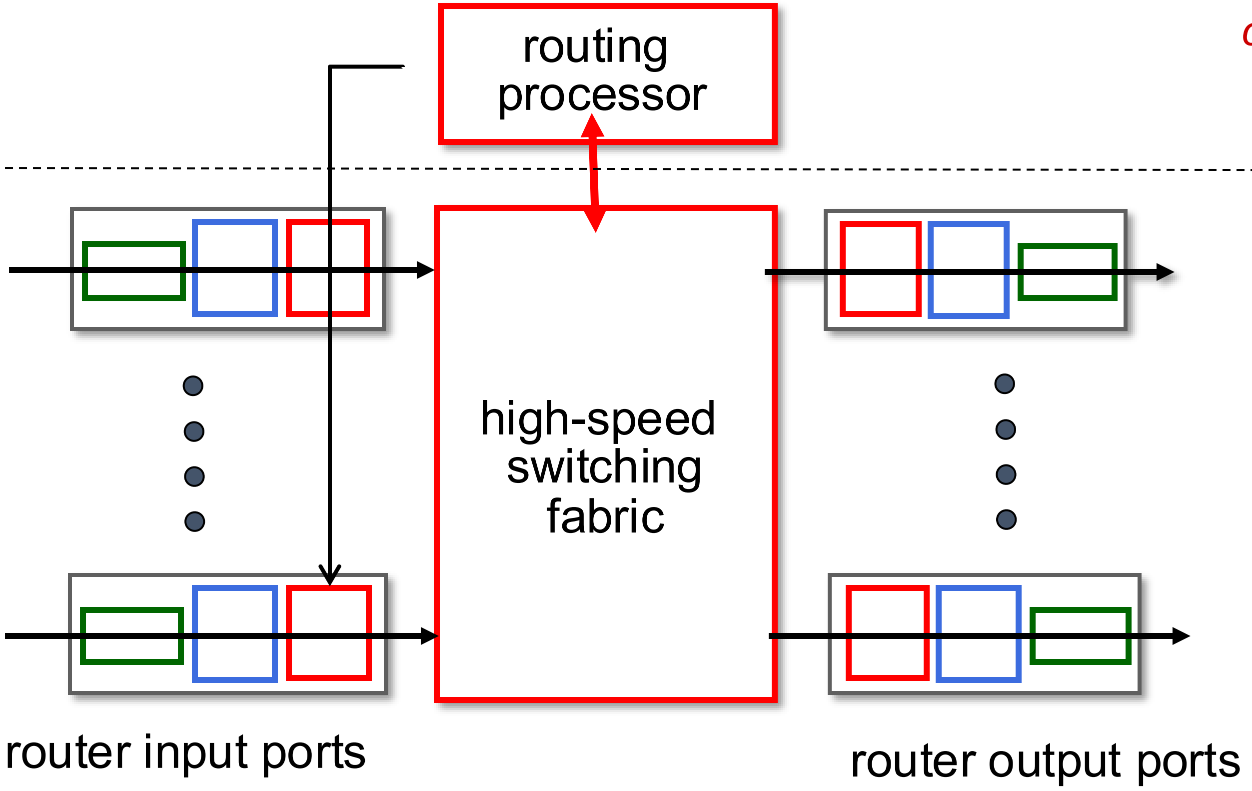
Router architecture and buffer management

- Router architecture
  - Output queueing
  - Input queueing
  - …

- Buffer management and scheduling

What is inside a router?

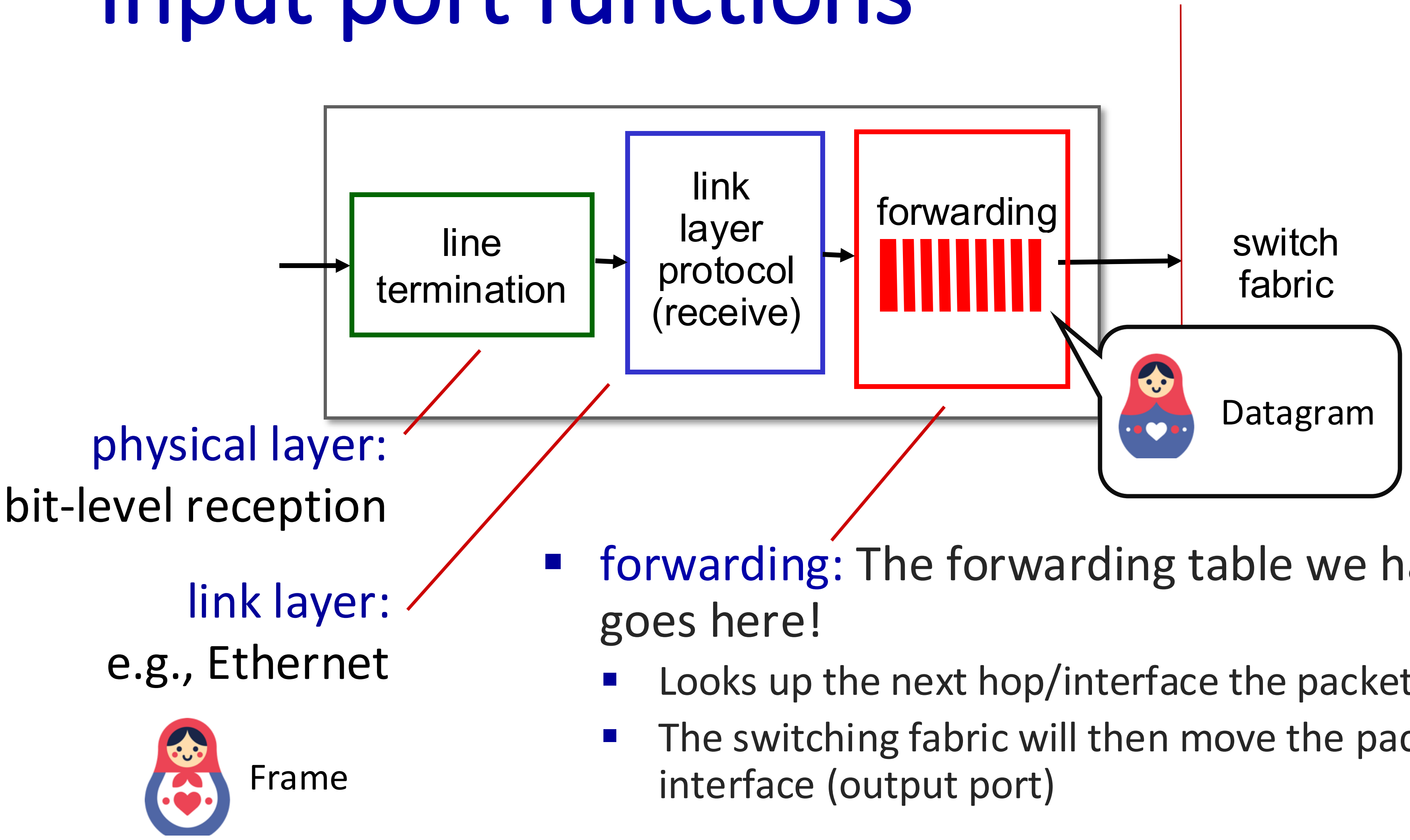
high-level view of generic router architecture:



routing, management control plane (software) operates in millisecond time frame

forwarding data plane (hardware) operates in nanosecond timeframe

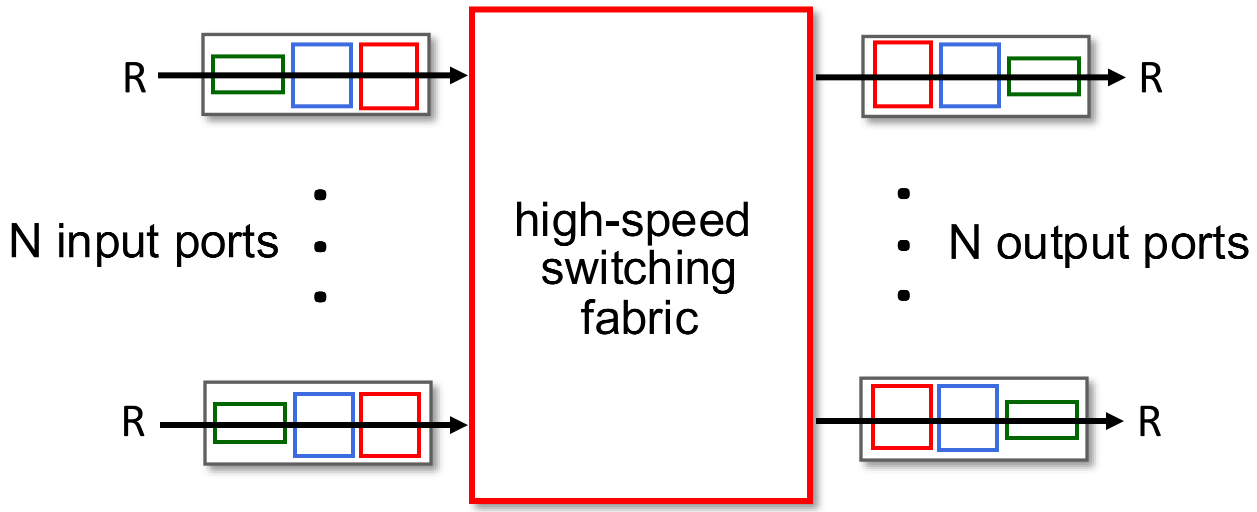
Input port functions



- forwarding: The forwarding table we have been talking about goes here!
  - Looks up the next hop/interface the packet should go to.
  - The switching fabric will then move the packet to exit from that interface (output port)

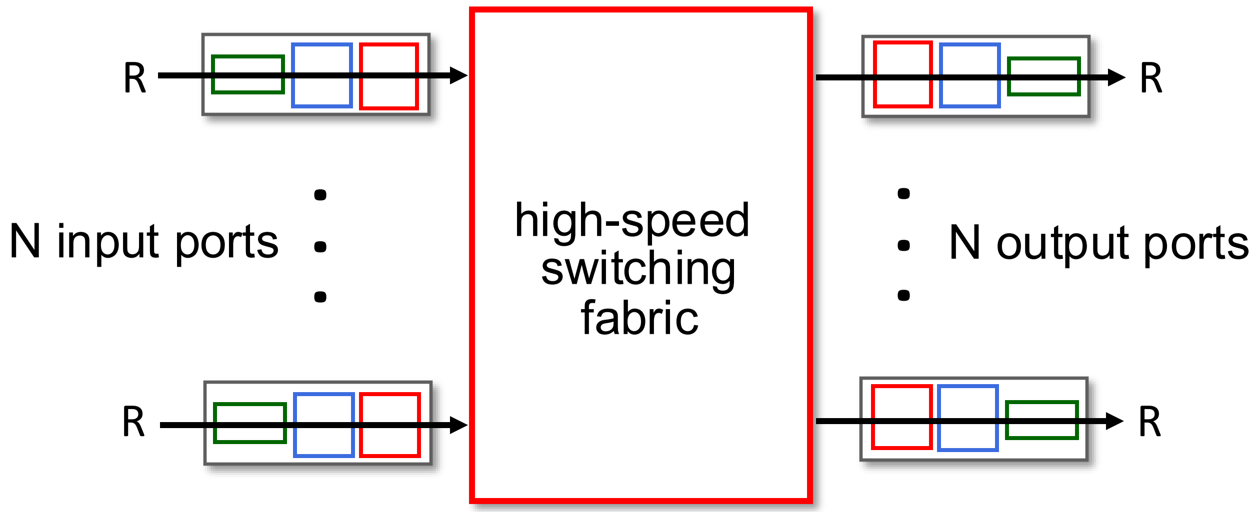
Switching fabrics

- transfer packets from input links to appropriate output links
- Suppose
  - All packets are of the same size
  - Define the time it takes to send/receive a packet on a port as our time unit, and call it a *tick* (today, that's usually a few nanoseconds!)



Switching fabrics

- In each tick, we can
  - receive at most a packet on each input port (up to N ports)
  - send at most a packet on each output port (up to N ports)



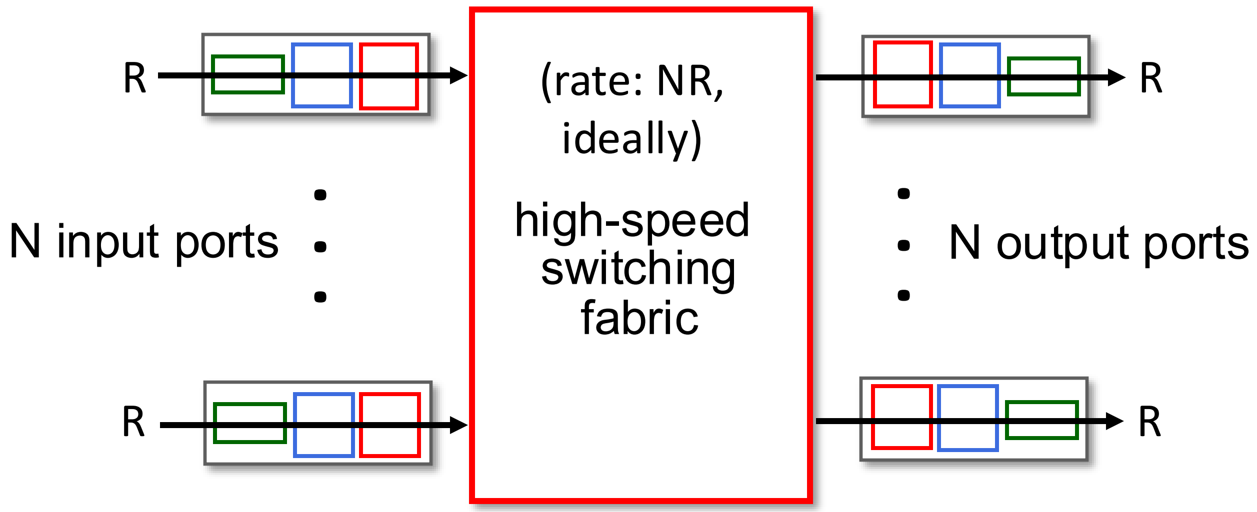
Switching fabrics

On each tick, we can

receive a packet on each input port

send a packet on each output port

- Ideally, the switching fabric can move N packets in each tick
  - If the link rates are R, an ideal switching fabric moves packets at rate NR.



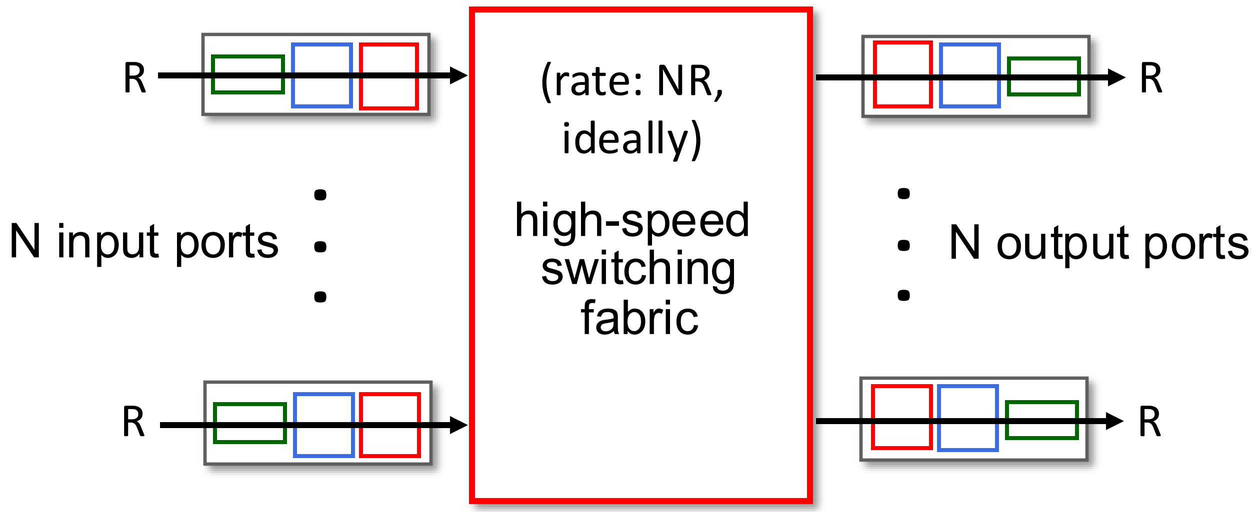
Switching fabrics

On each tick, we can

receive a packet on each input port

send a packet on each output port

- If two or more input ports have a packet destined to the same output port
  - only one can go out in the next tick(s)
  - the rest have to wait somewhere

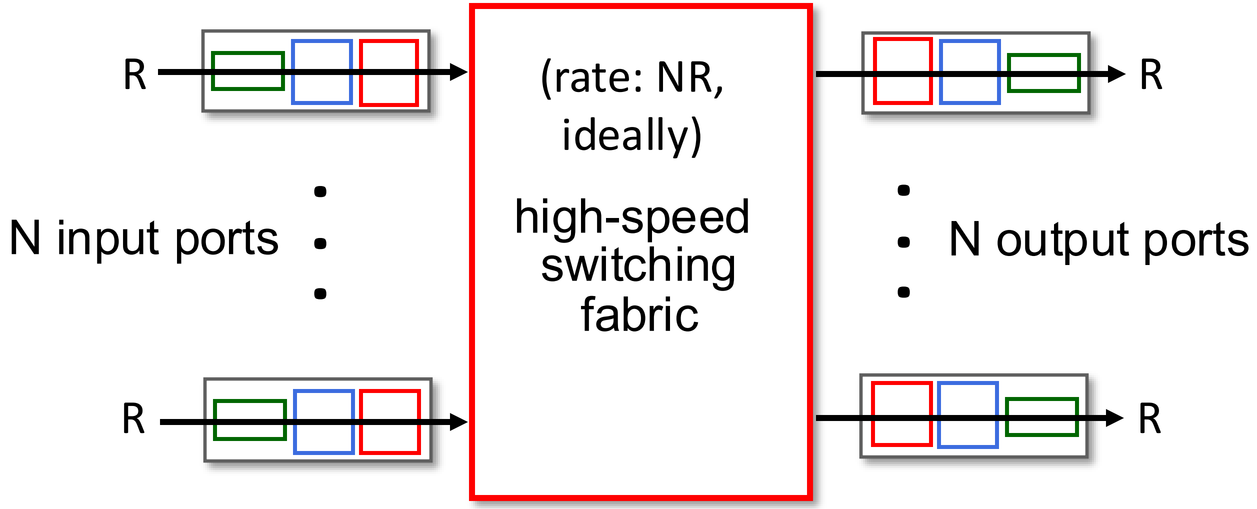


Switching fabrics

On each tick, we can
- receive a packet on each input port
- send a packet on each output port

- If two or more input ports have a packet destined to the same output port
  - only one can go out in the next tick(s)
  - the rest have to wait somewhere

Queued in a buffer (*where?*)

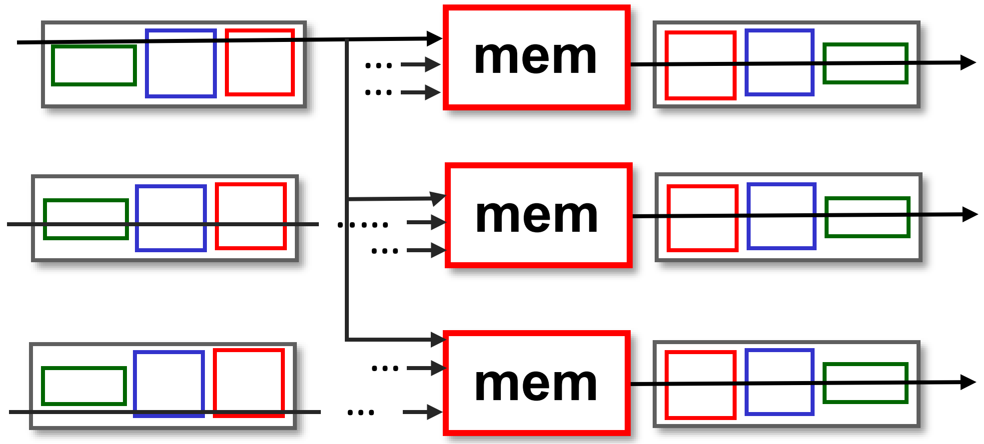


Router architecture and buffer management

- Router architecture
  - Output queueing
  - Input queueing
  - …
- Buffer management and scheduling

Output queuing

- N separate memories, one for each output port
- Input port receives a packet, and then puts it in the memory of the output port it is supposed to exit from
- Output port pulls the next packet from its corresponding memory



Router architecture and buffer management

- Router architecture
  - <u>Output queueing</u>
  - <u>Input queueing</u>
  - …

- Buffer management and scheduling

Input queuing

- N separate memory pools, one for each input port.
- Input port receives a packet, puts it in the dedicated memory for that input port.
  - 1 enqueue per tick
- Output port gets the next packet from the memory of one of the input ports that have packets destined to it
  - 1 dequeue per tick

Input queuing

At most one enqueue in each memory per tick

Input queuing

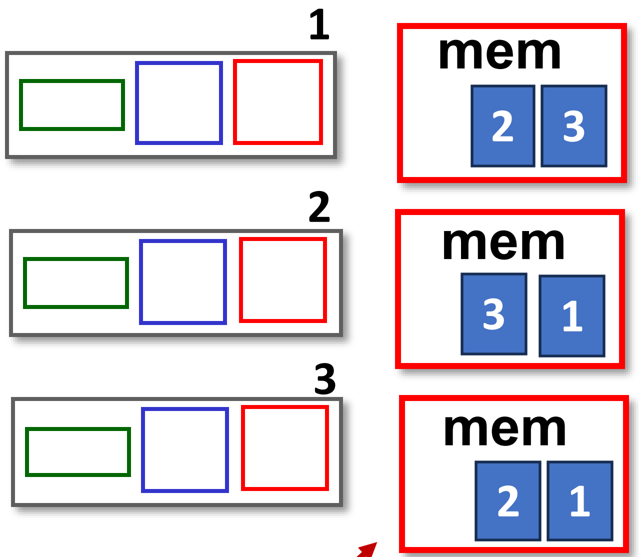
At most one dequeue from each memory per tick

Input queuing

At most one dequeue from each memory per tick

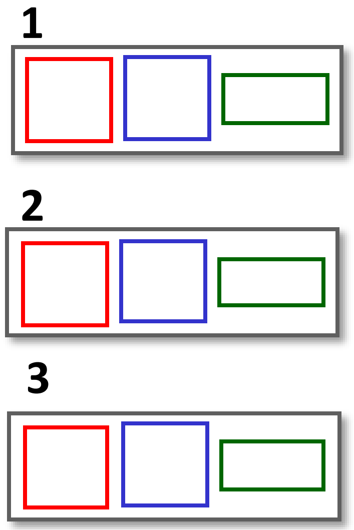
Input queuing

How do we coordinate packets between inputs and outputs?



In every tick

- Each input sends to at most one output
- Each output receives from at most each input



A single buffer per input can lead to head of line blocking… look up virtual output queuing for a common solution

Coordinates between inputs and outputs by solving a bi-partite matching problem!

Architecture trends

- Today, there is a renewed interest in output-queued (and shared memory) architectures
- Data centers have *many* switches (100s of thousands)
- To keep the costs down, vendors have reduced the amount of memory available for buffering in these switches
  - Easier, e.g., compared to a WAN, to keep the queues shorter in DCs, specially with the help of congestion control algorithms.
- Easier to make smaller high-speed memory with multiple enqueues and/or dequeues per tick
- With output-queued (or shared-memory architectures), no need for dealing with efficient scheduling of a crossbar.

Router architecture and buffer management

- Router architecture
  - <u>Output queueing</u>
  - <u>Input queueing</u>
  - …

- Buffer management and scheduling

Queue/Buffer management and scheduling

- Independent of where the queues are in the router architecture, there are some important questions:
  - *Buffer size*: How large should a buffer be?
  - *Queue management*: When the queue is full, which packet do we drop? What do we do when the queue starts building up?
  - *Packet scheduling*: Which packet in the queue gets dequeued first? Should it be first-in first-out? Something else?

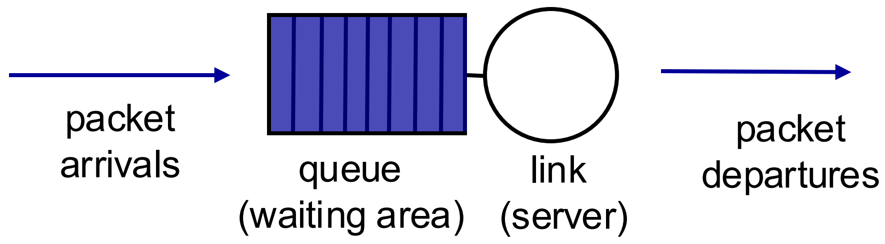
How large should a buffer be?

There is no easy answer:

- Too small: can’t absorb bursts, keeps dropping packets
- Too large: can hurt performance
  - *buffer bloat:* When the buffer is too large, it will take a long time to fill up before a packet is dropped (however, TCP only realizes there is congestion when a packet is dropped and will not decrease its sending rate). In the meantime, all packets will experience increasing queueing delay.
  - Delay-based congestion control algorithms do better here.

Queue management – Drop policy

- When a new packet arrives to a full queue, which packet do we drop?
- *Tail drop*: drop arriving packet
- *Priority*: drop/remove based on priority
  - E.g., if the incoming packet has higher priority than a packet already in the queue, drop the lower priority packet and insert the incoming packet into the queue.



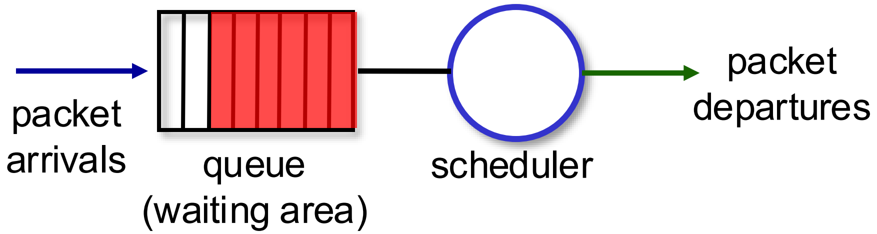
Queue management – Marking

- When the queue starts filling up, one strategy is to mark packets to signal the onset of congestion to the end points
- Recall our discussion about Early Congestion Notification (ECN) and its role in congestion control
- When should we start/stop marking packets?
- Which packets do we mark?
  - All packets after the queue size passes a threshold?
  - From the flow with the most packets in the queue?
  - …

Packet scheduling

- So far, we have assumed that our queues are *first in first out (FIFO)*

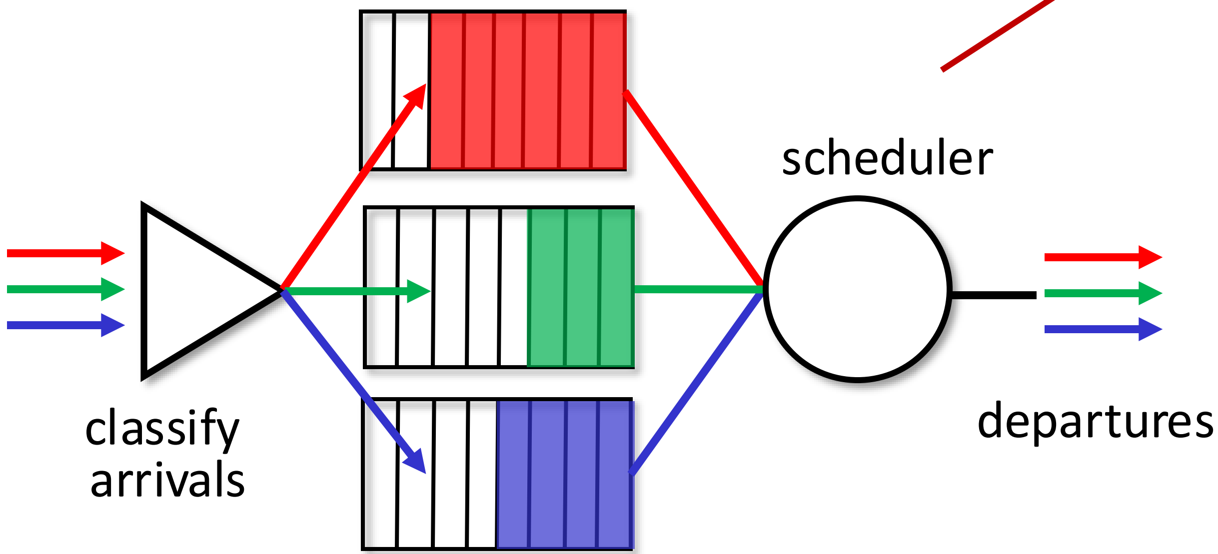
Abstraction: queue



- But, there are other packet scheduling algorithms as well.

Packet scheduling

Priority
Round-Robin
Weighted Round Robin
…



Instead of one queue for all packets going to the same output port (in an output-queued switch), there are separate queues for different ”traffic classes”

Additional Slides