Queueing disciplines and switches

Lecturer: Venkat Arun

CS 356

Some figures are borrowed from Nick McKeown’s slides

Recap: Congestion control

- Congestion window (cwnd) caps the maximum number of packets inflight (i.e. packets that have been sent but neither acknowledged, nor declared as lost)
- If a flow is alone on a network path with bottleneck capacity C and propagation delay R_m, then it will maintain **max(0, cwnd - C R_m)** packets in the queue. Propagation delay refers to the Round Trip Time (RTT) when the queue is empty
- C R_m is called the Bandwidth Delay Product (BDP) and is an important quantity that characterizes a link
- If cwnd >= C R_m, sending rate = C in steady state. Else, it is cwnd / R_m
- If cwnd > C R_m + buffer_size, then packets will get lost
- Thus, if a CCA maintains cwnd between C R_m and C R_m + buffer_size, it will fully utilize the link and not drop packets
- AIMD oscillates between these two values. Ideally, you'd want cwnd to be exactly C R_m to also minimize queuing delay. Modern algorithms do this, but not particularly well

Recap: Congestion control fairness

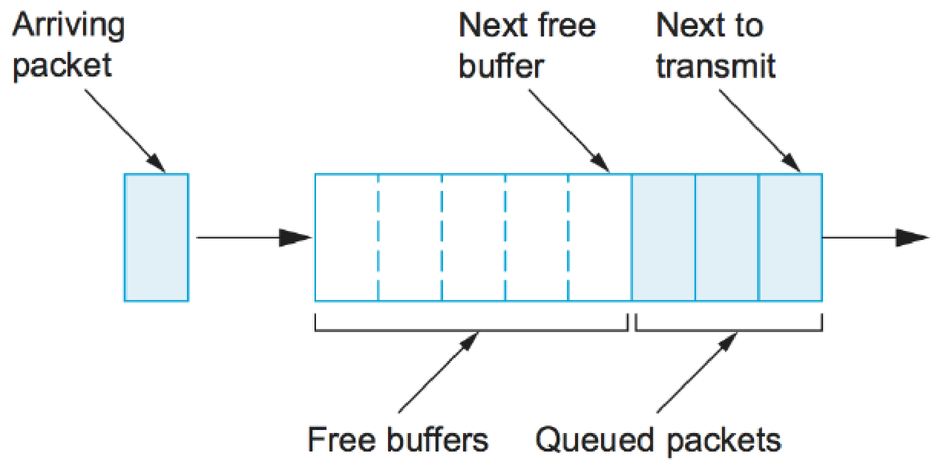
- If there are multiple flows with congestion windows cwnd1, cwnd2, … sharing a bottleneck bandwidth C and all of them have propagation delay R_m, they maintain $\sum_i cwnd_i \ - CR_m$ bytes in the queue
  - The expression for when they have different propagation delays is more complex. In fact, we found this year that it takes a while for the system to reach steady state after one flow changed its cwnd suddenly
- If the flows use AIMD to vary their cwnd, they will eventually converge to all of them having the same cwnd (not true if R_m is different for different flows)
- This happens because every time they do additive increase, the difference in cwnd remains constant. When they do multiplicative decrease the difference reduces by half
- Flows can "go rogue" and transmit faster than their fair share. If the network uses First-In-First-Out (FIFO) queues, there is nothing others can do about it

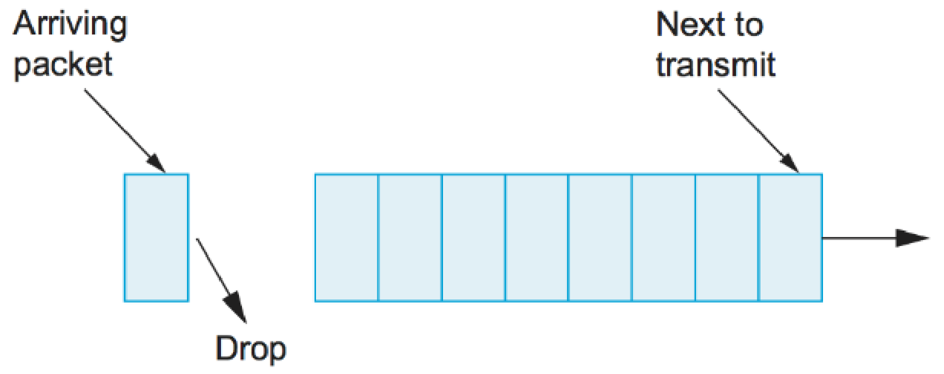
Theme of today’s class

- Subtle differences in design decisions lead to strange outcomes

Queuing discipline: FIFO

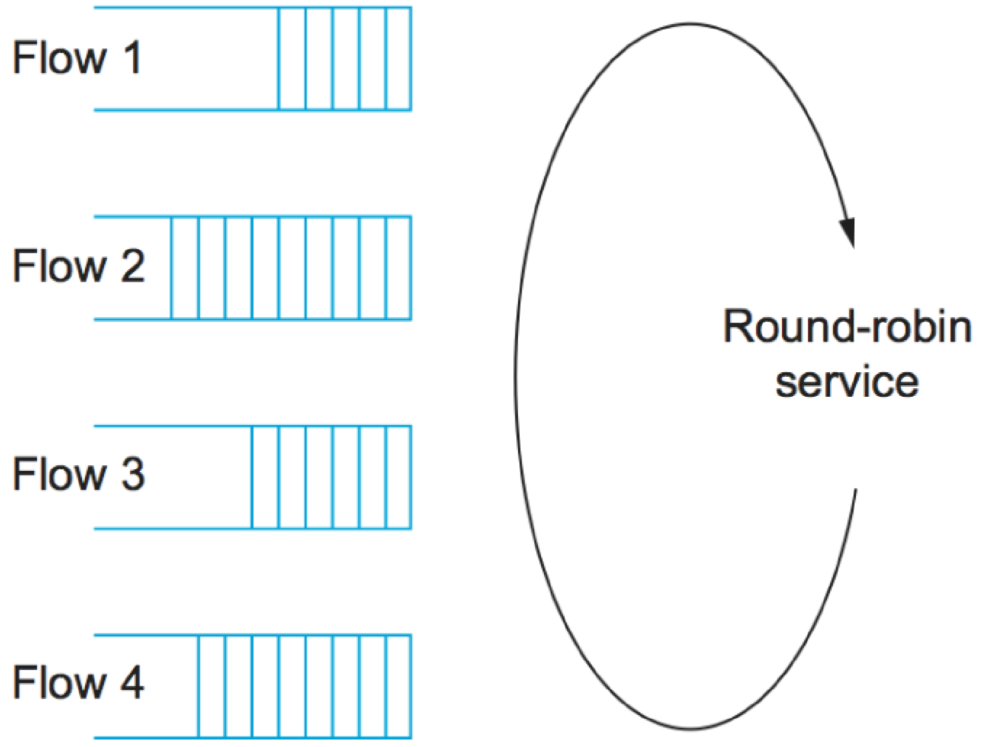
- This is what we had assumed when we discussed AIMD congestion control
- All packets arriving at a port areenqueued here
- Advantage: simple to implement
- Disadvantage 1: Flows that send more get more bandwidth. They can go “rogue”, blast packets into the network and get rewarded for it (e.g. by modifying TCP congestion control)
- Disadvantage 2: Some flows want throughput. They want to maintain a larger queue so that if the link rate suddenly increases, there are packets to transmit. Others prefer lower latency, but their packets will be stuck behind the throughput-sensitive flows’ packets
- Q. what will happen if we dropped packets at the front of the queue instead of at the tail?





Queuing discipline: Fair queuing

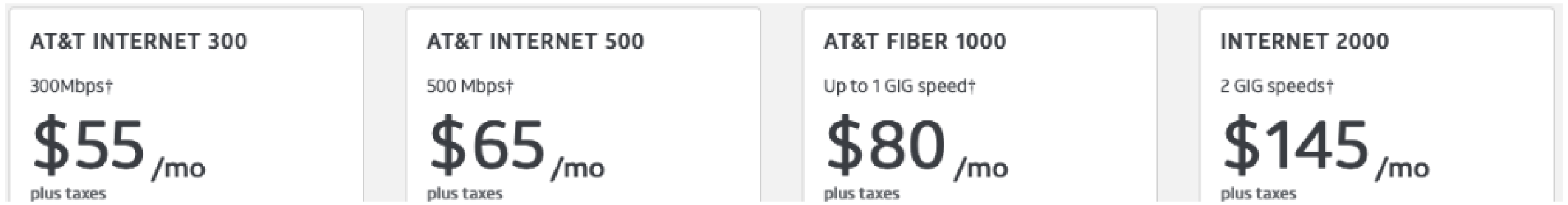
- Input queues discussed earlier are splitfurther into multiple queues.
- Packets are classified into “flows” in someway and put into separate queues
  - E.g. using src/dst IP and port and the protocolnumber. This is commonly called a “5 tuple”
- Packets are dequeued so that each queue gets the same number of bytes per second
- Advantage: prevents rogue flows (Q. does it?)
- Disadvantage: much more complex. In particular, hardware is bad at maintaining a variable number of queues since thenumber of packets is not known a priori



Queuing discipline: artificial bottlenecks

- Sometimes routers are explicitly instructed to forward at a lower speed than what they are capable
- This is extremely simple to implement and ensures the congestion stays near the edges of the internet.
- Mechanisms like fair queuing are nearly impossible to implement near the core where millions of flows may go through the same link

Commercial artificial bottleneck



AT&T INTERNET 300
300Mbps†
$55 /mo
plus taxes

AT&T INTERNET 500
500 Mbps†
$65 /mo
plus taxes

AT&T FIBER 1000
Up to 1 GIG speed†
$80 /mo
plus taxes

INTERNET 2000
2 GIG speeds†
$145 /mo
plus taxes

- This usually just means that there is some place between you and the network that is bottlenecked at the advertised speed.
- It caps the maximum, but says very little about what speed you will actually get.
- When you change your plan, nobody is going out to install a new router or wire. It is just a software change that asks the router to change how much it is throttling your packets.
- ISP contracts with commercial entities (e.g. UT Austin) are similar, except with bigger numbers

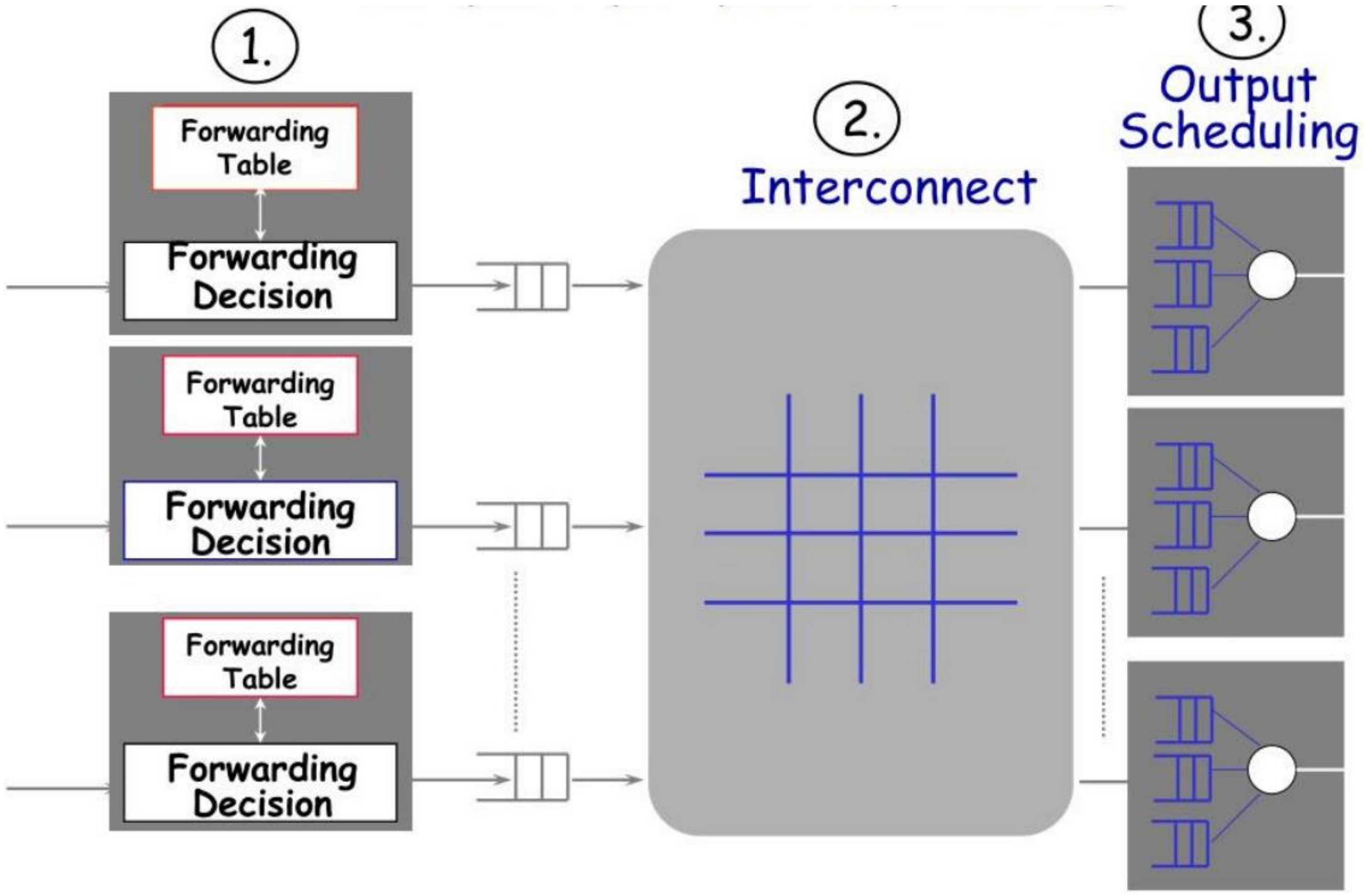
Routers

- A router is more than a single queue. It takes packets from one port (pictured), reads its destination address and forwards it to the appropriate port
- How does it know the destination port? By looking up the destination address in the routing table
- How do you implement this efficiently in hardware?

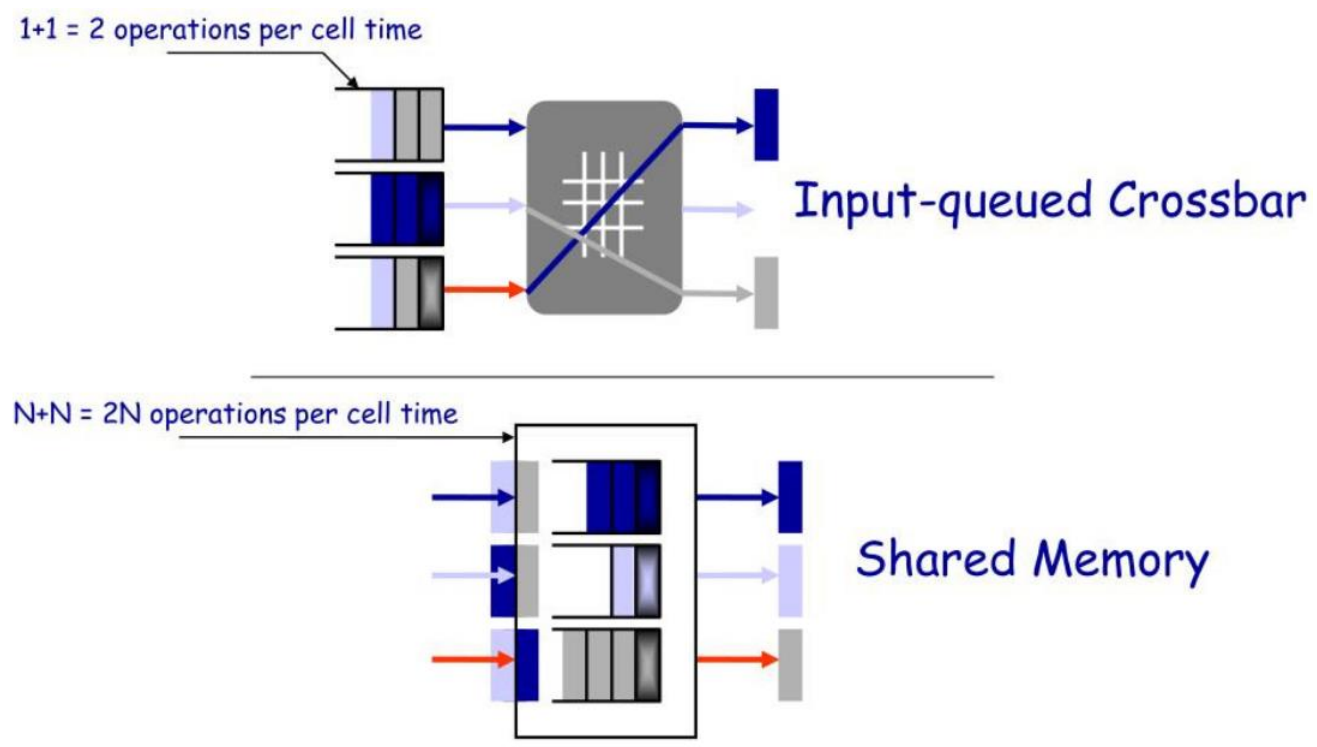
How do you switch packets?

1. When a packet comes, you decide where to forward it
2. You send it to the output port somehow
3. You decide the order in which to send packets at the output port (Q. Why can't you do this at the input?)

Note: we treat input ports as distinct from output ports. Every physical port has an input and output port



Shared memory architecture vs input-queued crossbar

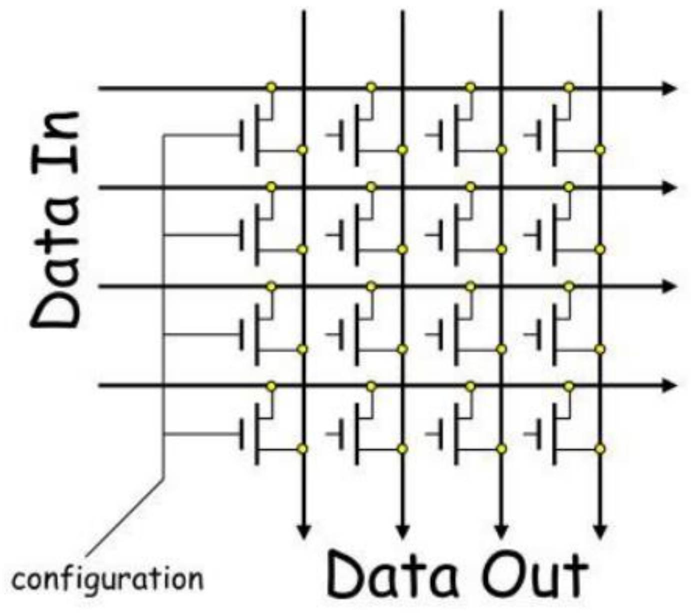

1+1 = 2 operations per cell time
Input-queued Crossbar
N+N = 2N operations per cell time
Shared Memory

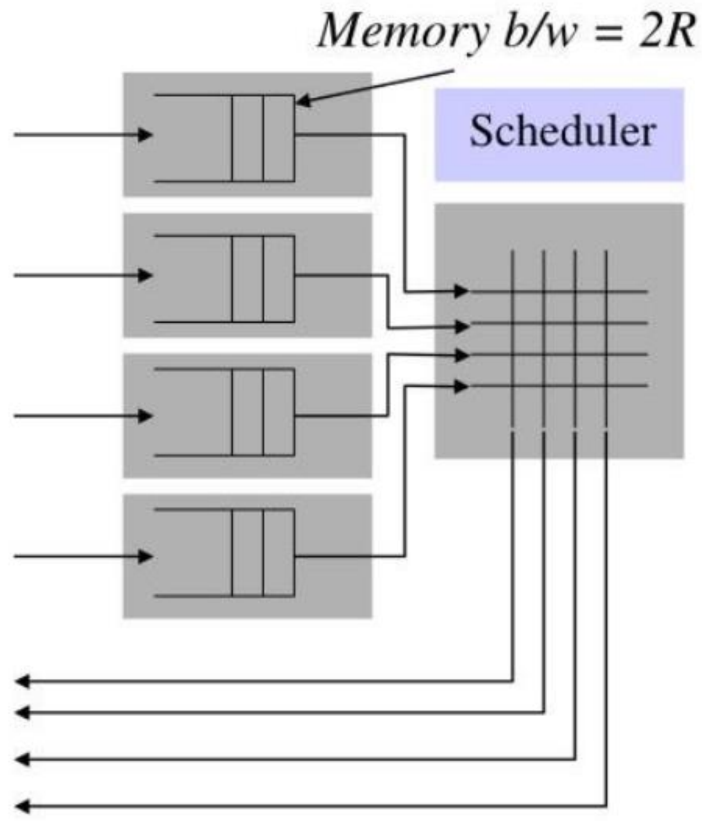
Input queued: Ports act independently except for the crossbar which takes packets from the input to the output side. Fragments memory

Shared memory: Input ports write to a datastructure in a shared memory that output ports read. Requires too much memory bandwidth!

High speed switches today use crossbars

Crossbars

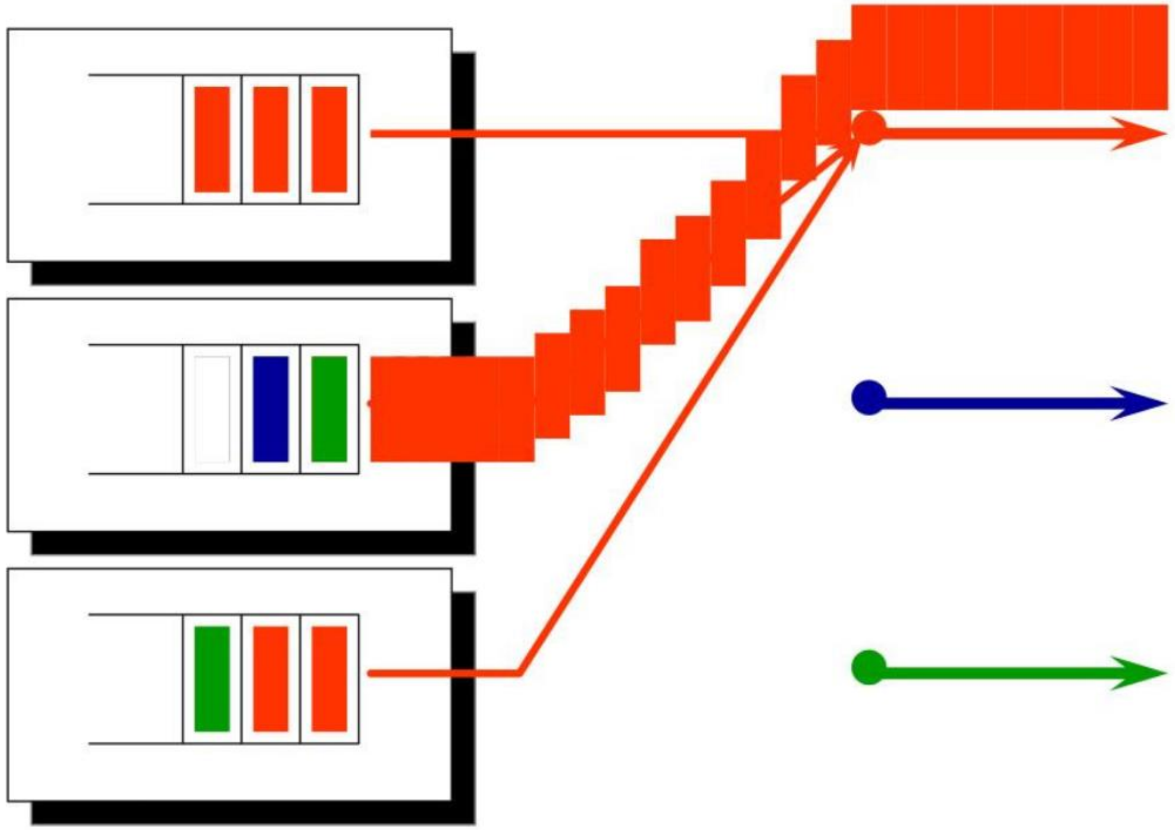




- The crossbar (left) selects which input port is connected to which output port. Input (output) ports can be connected to at most one output (input) port
- You can imagine this as a bipartite graph (shown on board)

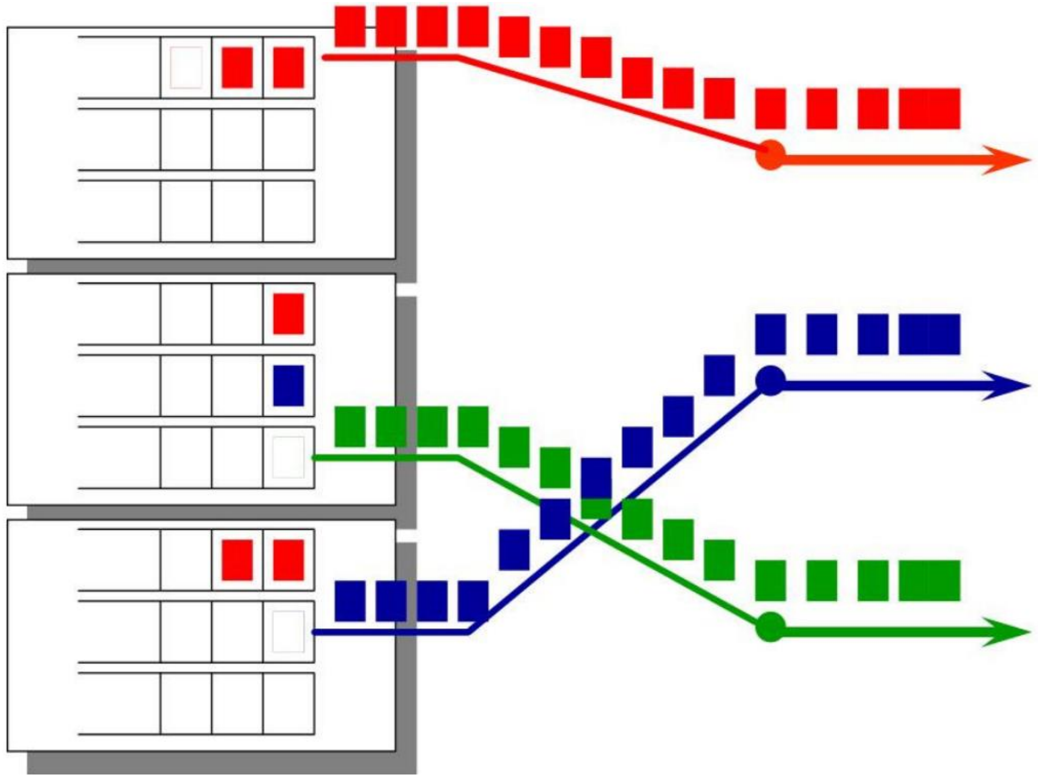
Head of line blocking

- Greed and blue packets can be transmitted in parallel with the red ones, but are blocked by the red packets



Head of line blocking solved with VOQs

- Each input queue was split into a separate “Virtual Output Queue (VOQ)”, one for each output
- This solves the head-of-line blocking problem



Where will you implement fair queuing?

- Often done by having another set of queues at each output port. This can get quite expensive