

# EN.601.414/614

# Computer Networks

## IP Router

Xin Jin

Fall 2020 (TuTh 1:30-2:45pm on Zoom)



<https://github.com/xinjin/course-net>

# Midterm Exam

- **Time: 75 minutes on October 15 (next Thursday)**
- **Location: Take-home**
- **Form: Open-book**
  - Can use slides for reference
  - Can use a calculator
  - Anything else is prohibited
- **Midterm review next Tuesday**

# Agenda

- **IP routers**
- **Router-assisted congestion control**

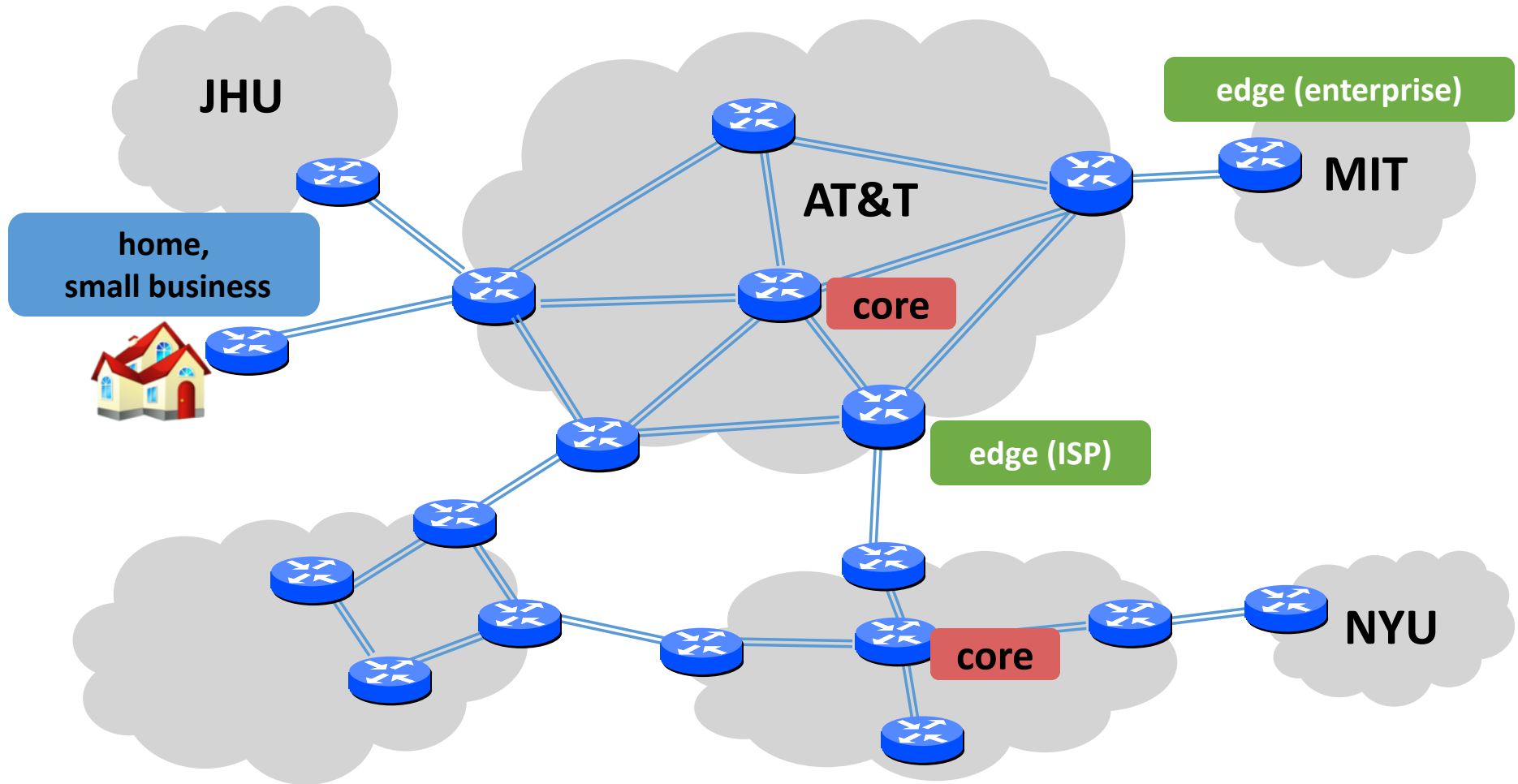
# IP routers

- **Core building block of the Internet infrastructure**
- **\$120B+ industry**
- **Vendors: Cisco, Juniper, Nokia**

# Router definitions

- **Router capacity** =  $N \times R$
- **N** = Number of external router “ports”
- **R** = Speed (“line rate”) of a port

# Networks and routers



# Many types of routers

- **Core**

- $R = 10/40/100$  Gbps
- $NR = O(100)$  Tbps (Aggregated)

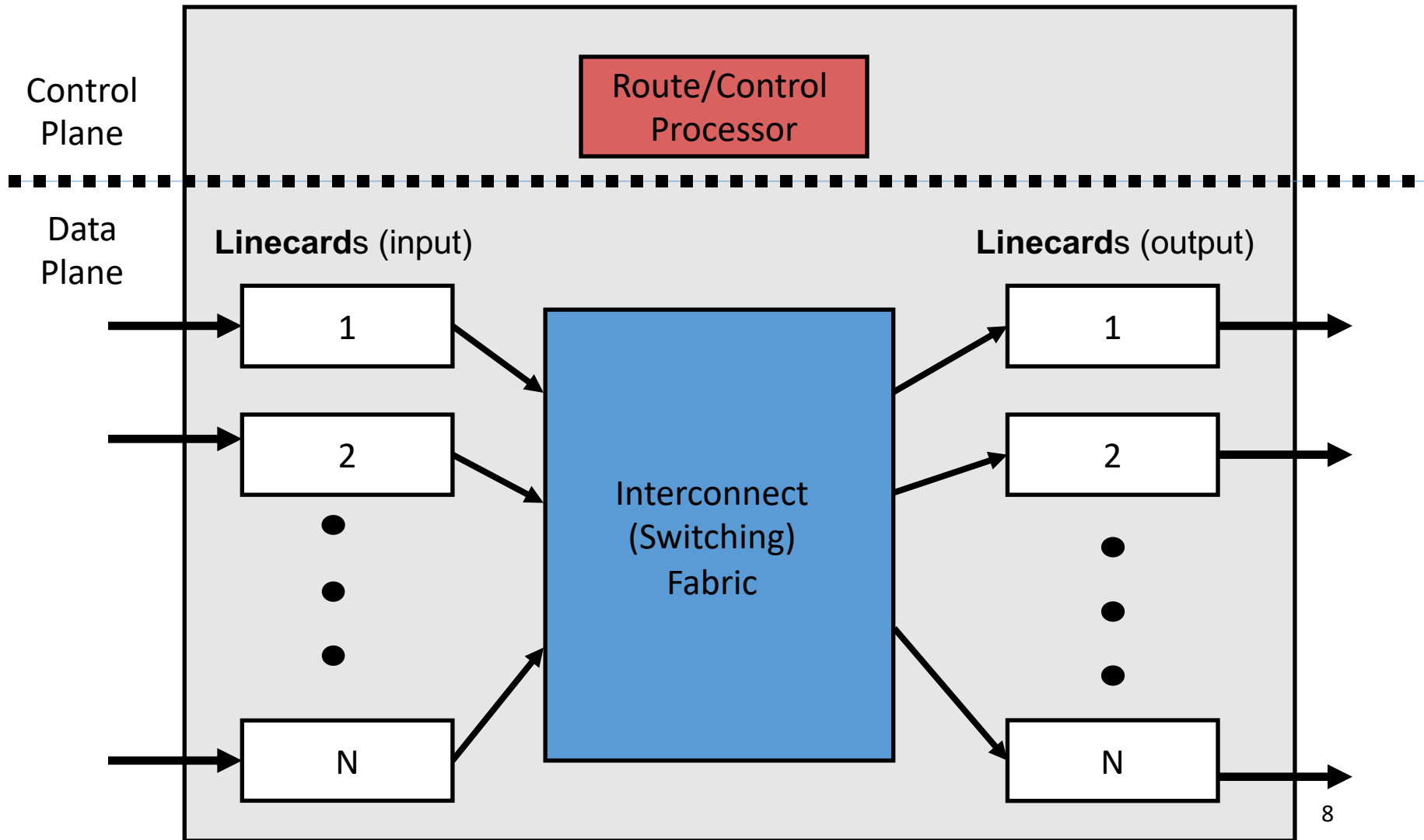
- **Edge**

- $R = 1/10/40$  Gbps
- $NR = O(100)$  Gbps

- **Small business**

- $R = 10/100/1000$  Mbps
- $NR < 10$  Gbps

# What's inside a router?





# What's inside a router?

- **Linecards**

- Input linecards process packets on their way in
- Output linecards process packets on way out
- Input and output for the same port are on the same physical linecard

- **Interconnect/switching fabric**

- Transfers packets from input to output ports

# Input linecards

- **Tasks**

- Receive incoming packets (physical layer stuff)
- Update the IP header
  - TTL, Checksum, Options and Fragment (maybe)
- Lookup the output port for the destination IP address
- Queue the packet at the switch fabric

- **Challenge: speed!**

- 100B packets @ 40Gbps → new packet every 20 nano seconds!
- Typically implemented with specialized ASICs (network processors)

# Looking up the output port

- **One entry for each address → 4 billion entries!**
- **For scalability, addresses are aggregated**

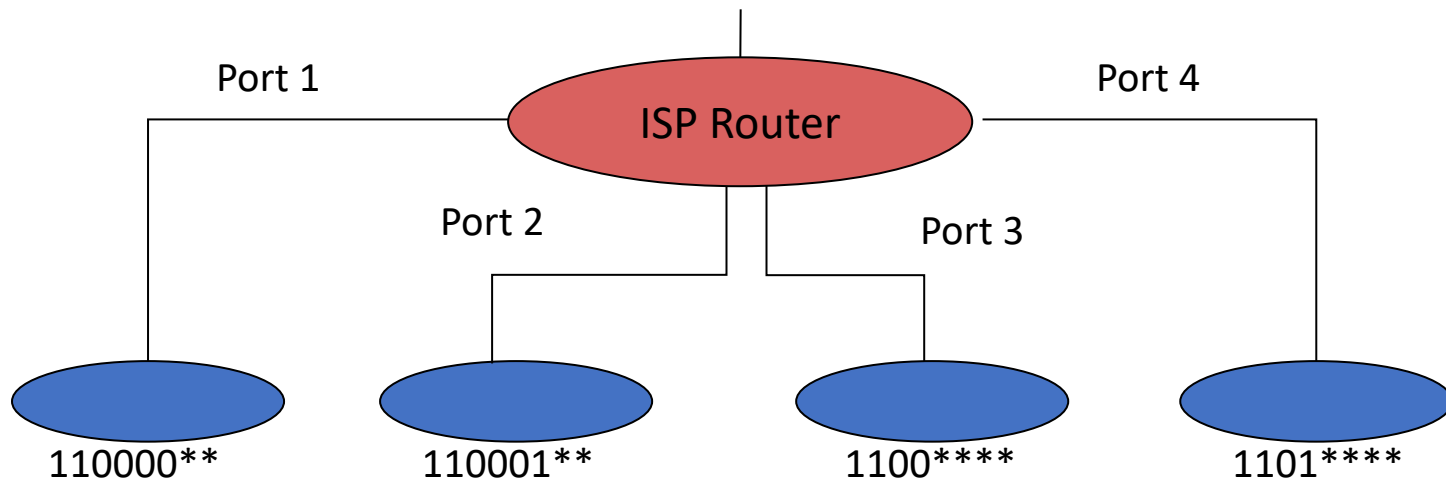
# Example

- **Router with 4 ports**
- **Destination address range mapping**
  - 11 00 00 00 to 11 00 00 11: Port 1
  - 11 00 01 00 to 11 00 01 11: Port 2
  - 11 00 10 00 to 11 00 11 11: Port 3
  - 11 01 00 00 to 11 01 11 11: Port 4

# Example

- Router with 4 ports
- Destination address range mapping
  - 11 00 00 00 to 11 00 00 11: Port 1
  - 11 00 01 00 to 11 00 01 11: Port 2
  - 11 00 10 00 to 11 00 11 11: Port 3
  - 11 01 00 00 to 11 01 11 11: Port 4

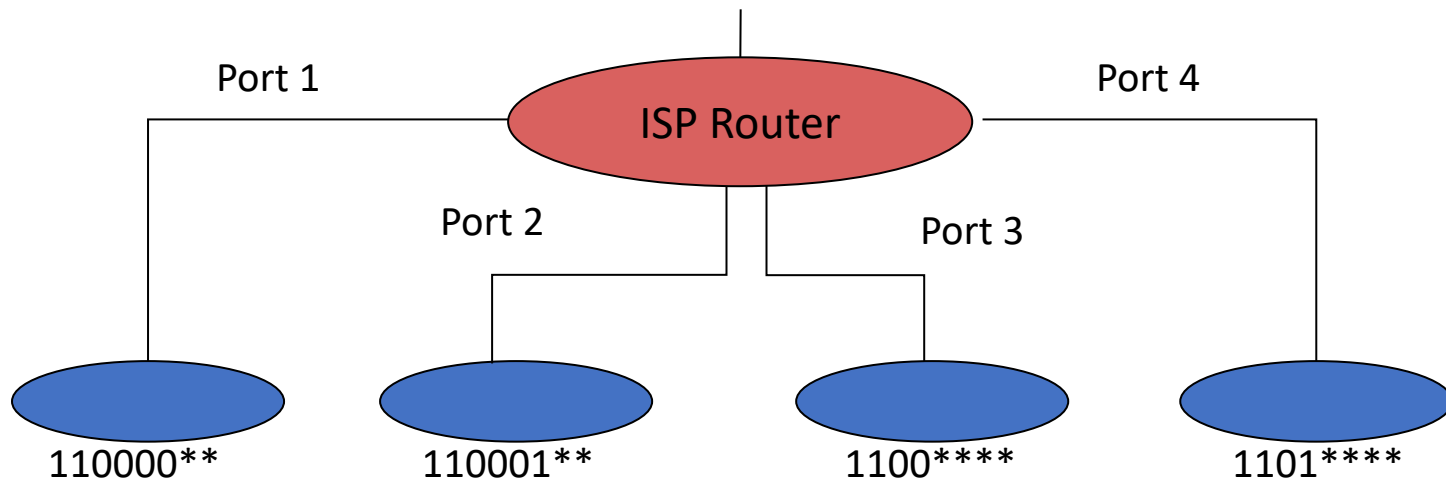
# Longest prefix matching



# Finding match efficiently

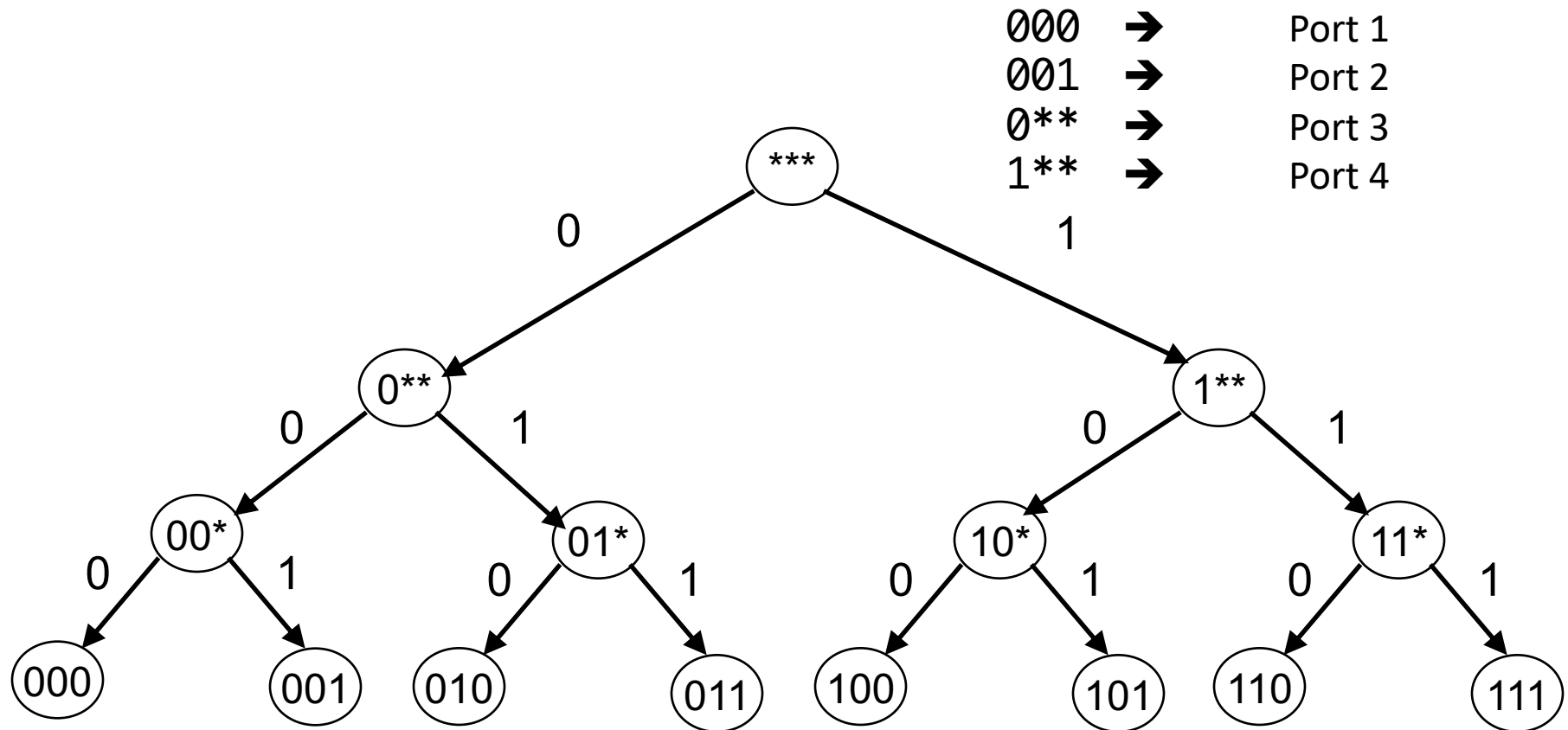
- **Testing each entry to find a match scales poorly**
  - On average:  $O(\text{number of entries})$
- **Leverage tree structure of binary strings**
  - Set up tree-like data structure

# Longest prefix matching

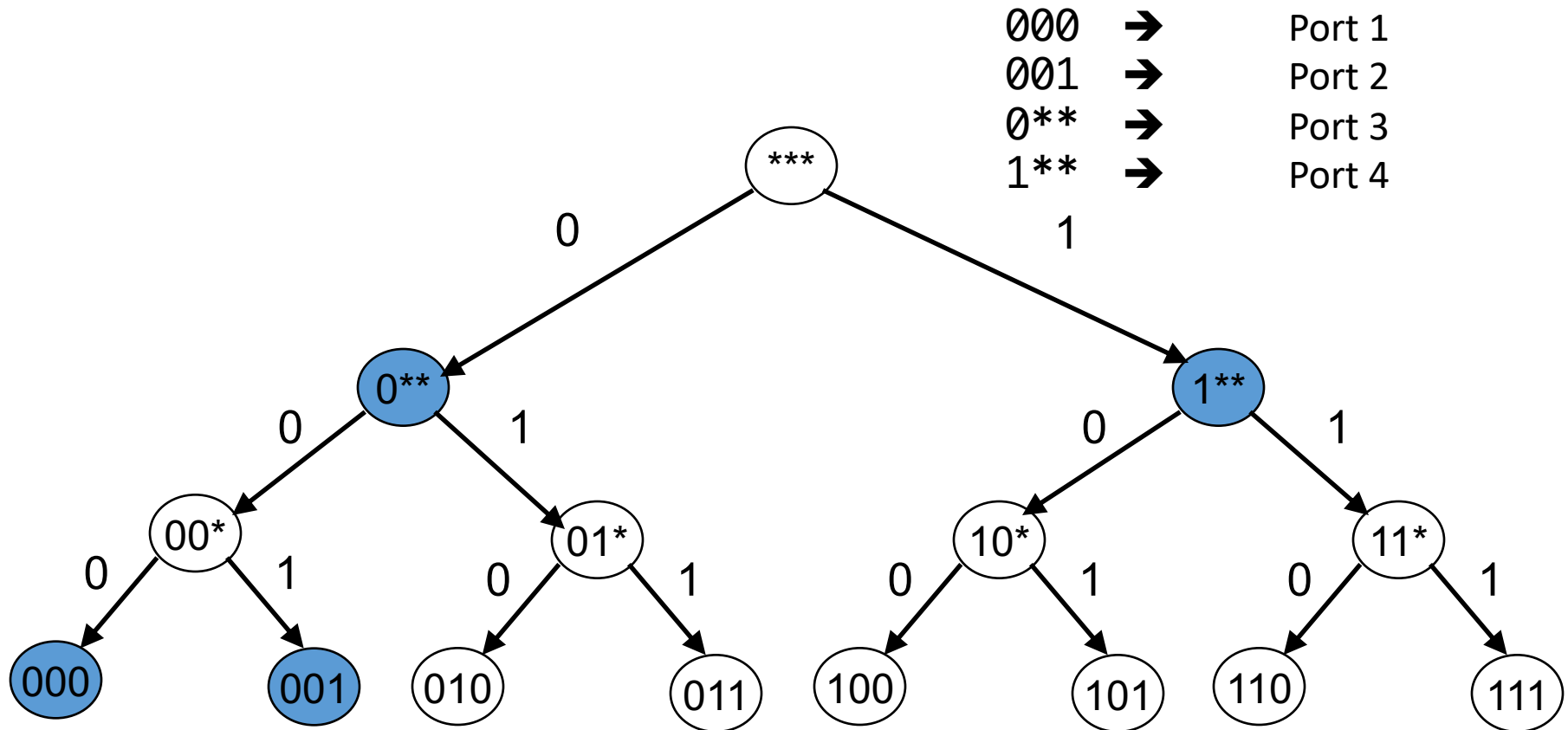




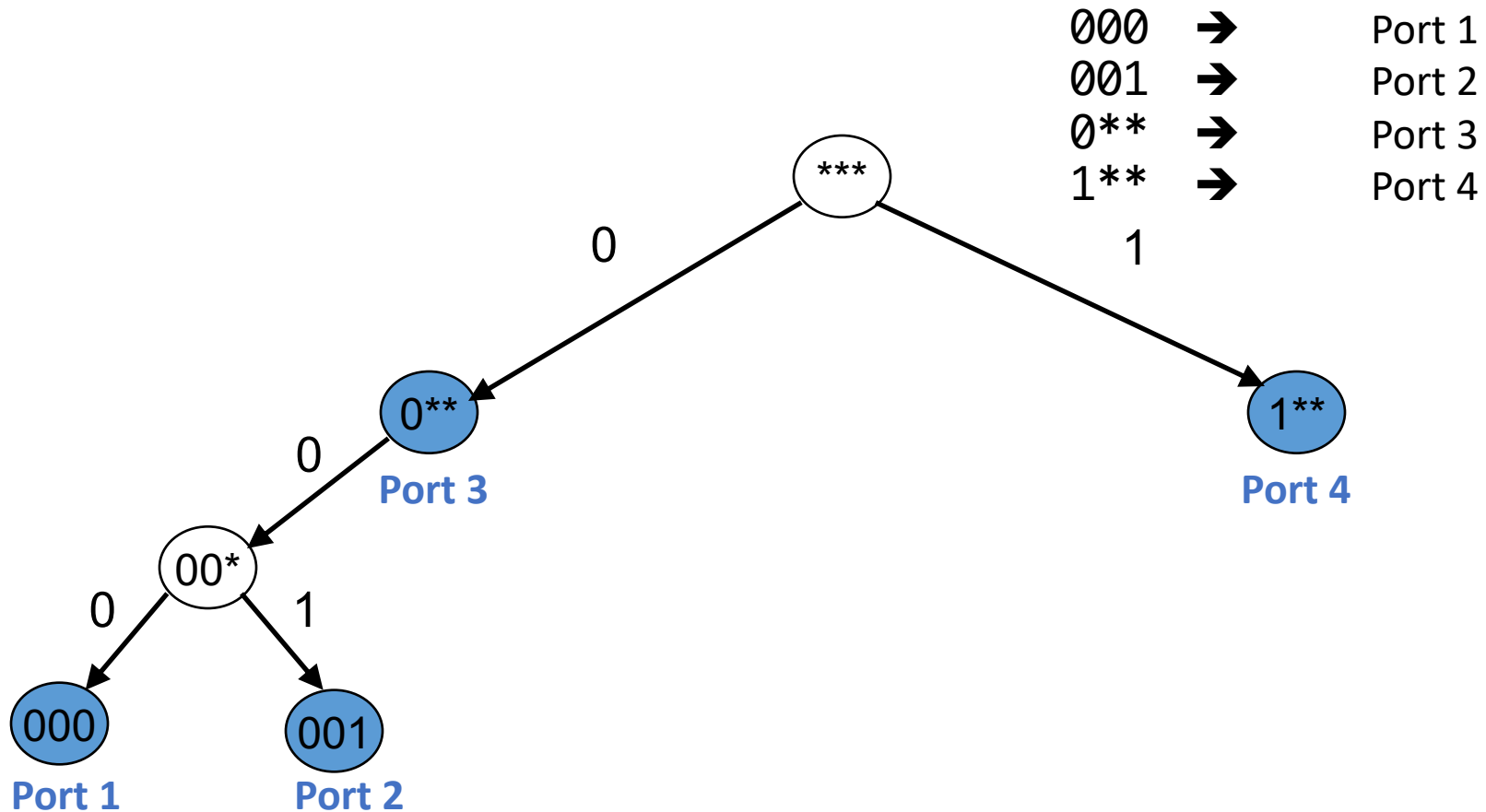
# Tree structure



# Tree structure



# Tree structure



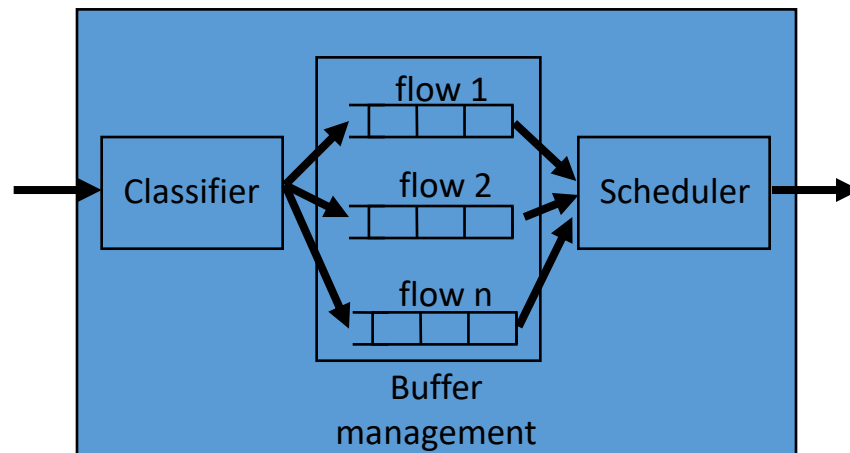
Record port associated with latest match, and only override when it matches another prefix during walk down tree

# Input linecards

- **Main challenge is processing speeds**
- **Tasks involved:**
  - Update packet header (easy)
  - LPM lookup on destination address (harder)
- **Mostly implemented with specialized hardware**

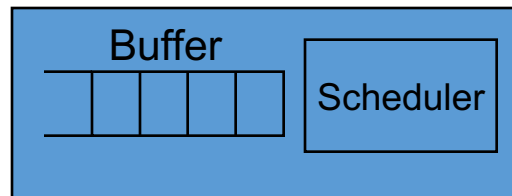
# Output linecards

- **Packet classification:** map packets to flows
- **Buffer management:** decide when and which packet to drop
- **Scheduler:** decide when and which packet to transmit
- Some of these functionalities can also be implemented at input linecards



# Simplest: FIFO router

- **No classification**
- **Drop-tail buffer management:** when buffer is full drop the incoming packet
- **First-In-First-Out (FIFO) Scheduling:** schedule packets in the same order they arrive



# Packet classification

- **Classify an IP packet based on a number of fields in the packet header, e.g.,**
  - Source/destination IP address (32 bits)
  - Source/destination TCP port number (16 bits)
  - Type of service (TOS) byte (8 bits)
  - Type of protocol (8 bits)
- **In general fields are specified by range**
  - Classification requires a **multi-dimensional range search!**

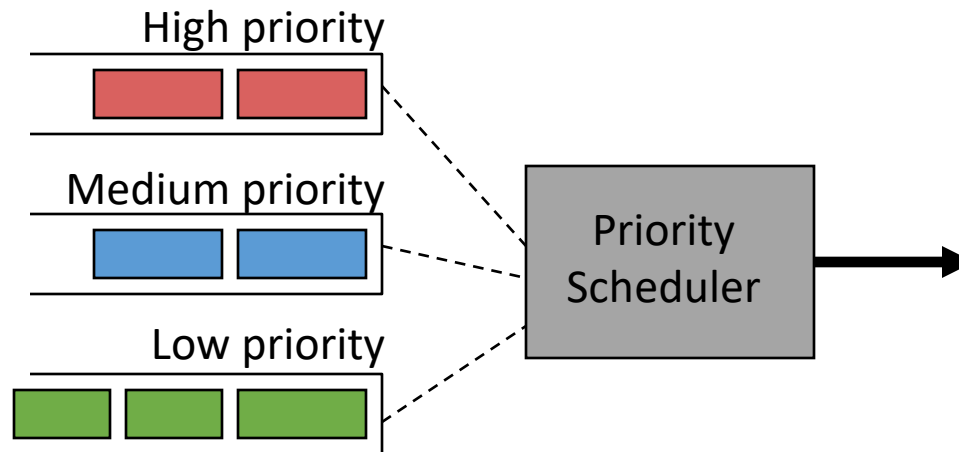
# Scheduler

- **One queue per “flow”**
- **Scheduler decides when and from which queue to send a packet**
- **Goals of a scheduling algorithm**
  - Fast!
  - Depend on the policy being implemented (fairness, priority, etc.)



# Priority scheduler

- **Priority scheduler: packets in the highest priority queue are always served before the packets in lower priority queues**



# Round-robin scheduler

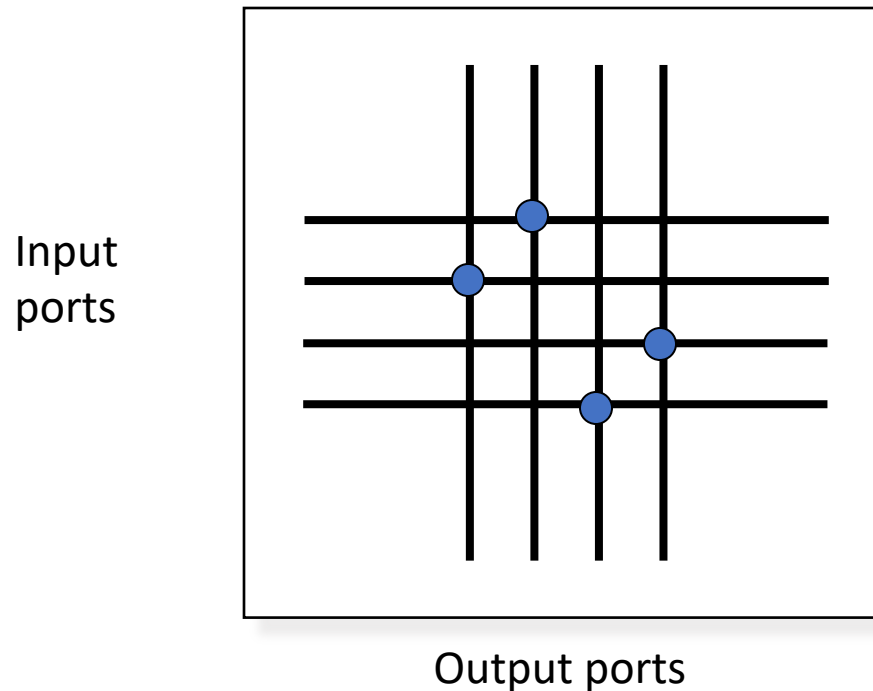
- **Round robin:** packets are served from each queue in turn
- **Fair queuing (FQ):** round-robin for packets of different size
- **Weighted fair queueing (WFQ):** serve proportional to weight
  - FQ gives equal weight to each flow

# Connecting inputs to outputs: Switching fabric

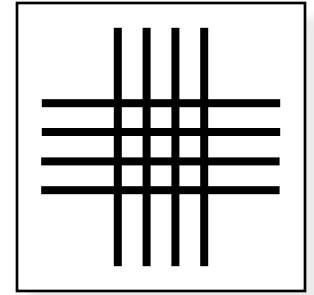
- **Mini-network**
- **Three primary ways to switch**
  - Switching via shared memory
  - Switching via a bus
  - Switching via an inter-connection network
    - For example, cross-bar

# Context

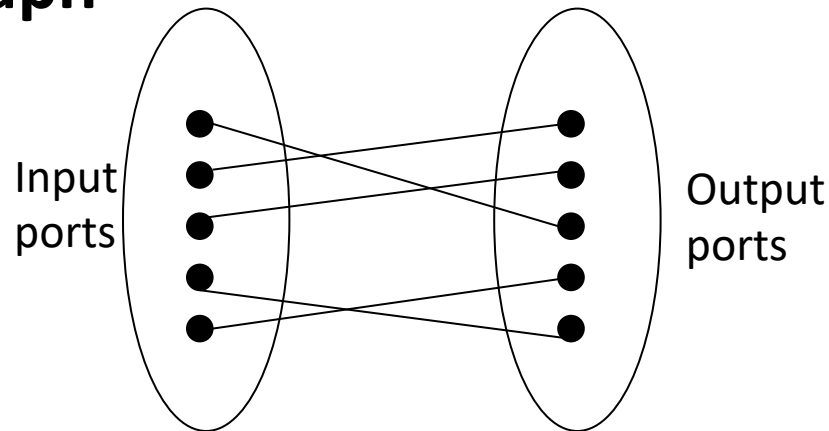
- **Crossbar fabric**
- **Centralized scheduler**



# Scheduling



- Run links at full capacity, fairness across inputs
- Scheduling formulated as finding a matching on a bipartite graph



- Practical solutions look for a good maximal matching (fast)

# Router-assisted Congestion control

# Recap: TCP problems

- Misled by non-congestion losses
- Fills up queues leading to high delays
- Short flows complete before discovering available capacity
- AIMD impractical for high speed links
- Saw tooth discovery too choppy for some apps
- Unfair under heterogeneous RTTs
- Tight coupling with reliability mechanisms
- End hosts can cheat

Routers tell endpoints if they're congested

Routers tell endpoints what rate to send at

Routers enforce fair sharing

Could fix many of these with some help from routers!

# Router-assisted congestion control

- **Three tasks for congestion control**
  - Isolation/fairness
  - Adjustment
  - Detecting congestion

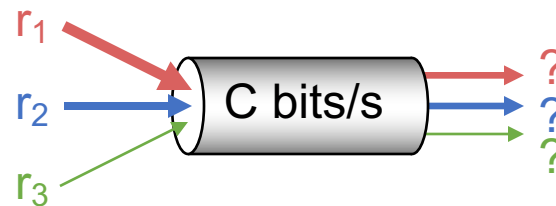


# Fairness: General approach

- **Routers classify packets into “flows”**
  - Let's assume flows are TCP connections
- **Each flow has its own FIFO queue in router**
- **Router services flows in a fair fashion**
  - When line becomes free, take packet from next flow in a fair order
- **What does “fair” mean exactly?**

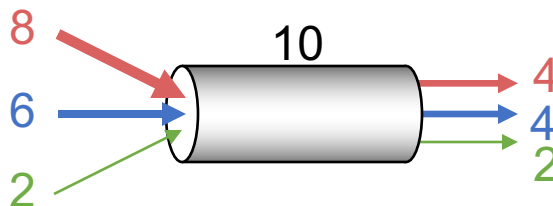
# Max-Min fairness

- Given set of bandwidth demands  $r_i$  and total bandwidth  $C$ , max-min bandwidth allocations are:
  - $a_i = \min(f, r_i)$
  - where  $f$  is the unique value such that  $\text{Sum}(a_i) = C$



# Example

- **$C = 10$ ;  $r_1 = 8$ ,  $r_2 = 6$ ,  $r_3 = 2$ ;  $N = 3$**
- **$C/3 = 3.33 \rightarrow$** 
  - $r_3$ 's need is only 2
    - Can service all of  $r_3$
  - Remove  $r_3$  from the accounting:  $C = C - r_3 = 8$ ;  $N = 2$
- **$C/2 = 4 \rightarrow$** 
  - Can't service all of  $r_1$  or  $r_2$
  - So hold them to the remaining fair share:  $f = 4$



$f = 4$ :  
 $\min(8, 4) = 4$   
 $\min(6, 4) = 4$   
 $\min(2, 4) = 2$

# Max-Min fairness

- Given set of bandwidth demands  $r_i$  and total bandwidth  $C$ , max-min bandwidth allocations are:
  - $a_i = \min(f, r_i)$
  - where  $f$  is the unique value such that  $\text{Sum}(a_i) = C$
- If you don't get full demand, no one gets more than you
- This is what round-robin service gives if all packets are the same size

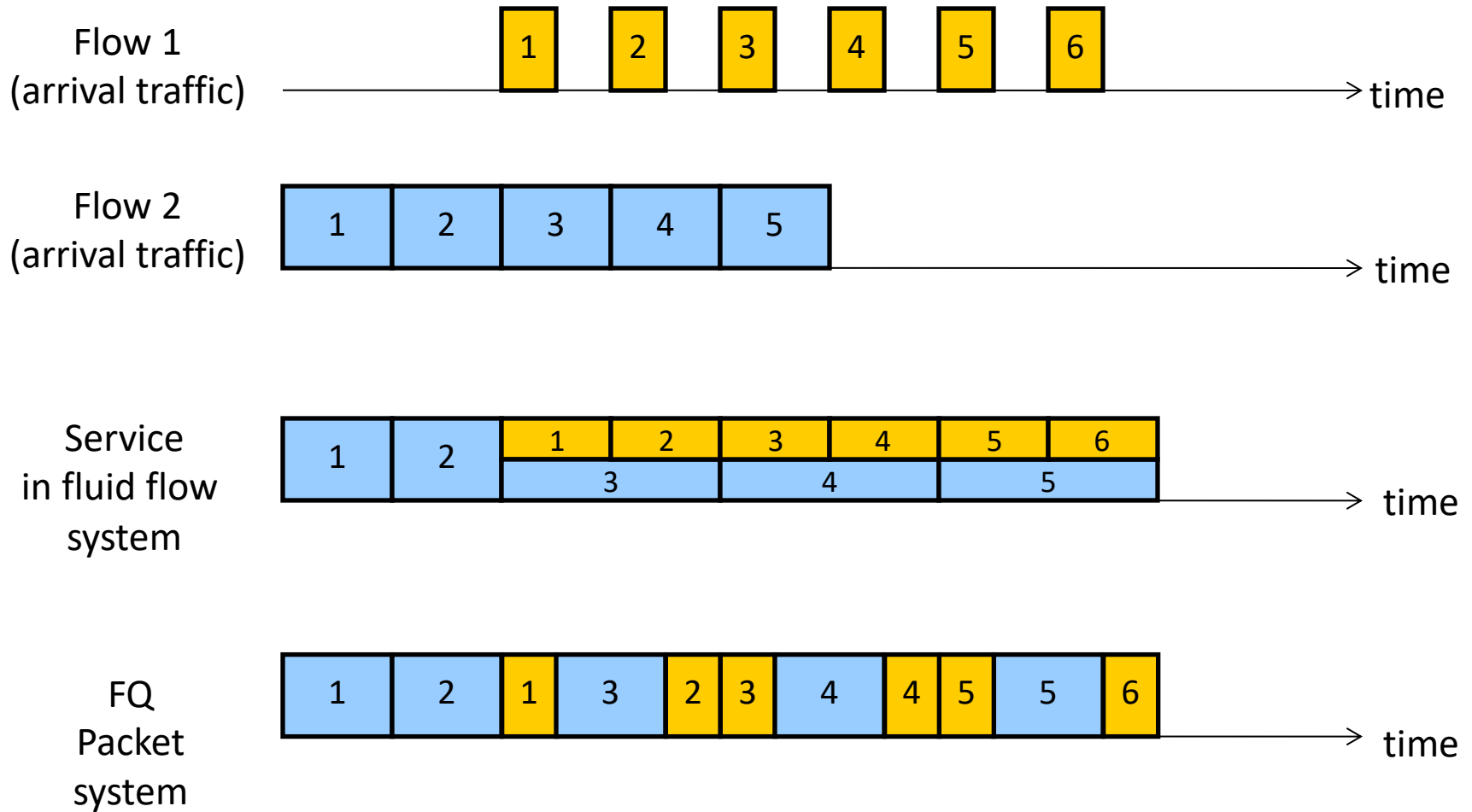
# How do we deal with packets of different sizes?

- **Mental model: Bit-by-bit round robin (“fluid flow”)**
- **Can you do this in practice?**
  - No, packets cannot be preempted
- **But we can approximate it**
  - This is what “fair queuing” routers do

# Fair Queuing (FQ)

- **For each packet, compute the time at which the last bit of a packet would have left the router if flows are served bit-by-bit**
- **Then serve packets in the increasing order of their deadlines**

# Example



# Fair Queuing (FQ)

- Implementation of round-robin generalized to the case where not all packets are equal sized
- **Weighted fair queuing (WFQ)**: assign different flows different shares
- **Today, some form of WFQ implemented in almost all routers**
  - Not the case in the 1980-90s, when CC was being developed
  - Mostly used to isolate traffic at larger granularities (e.g., per-prefix)



# FQ vs. FIFO

- **FQ advantages:**

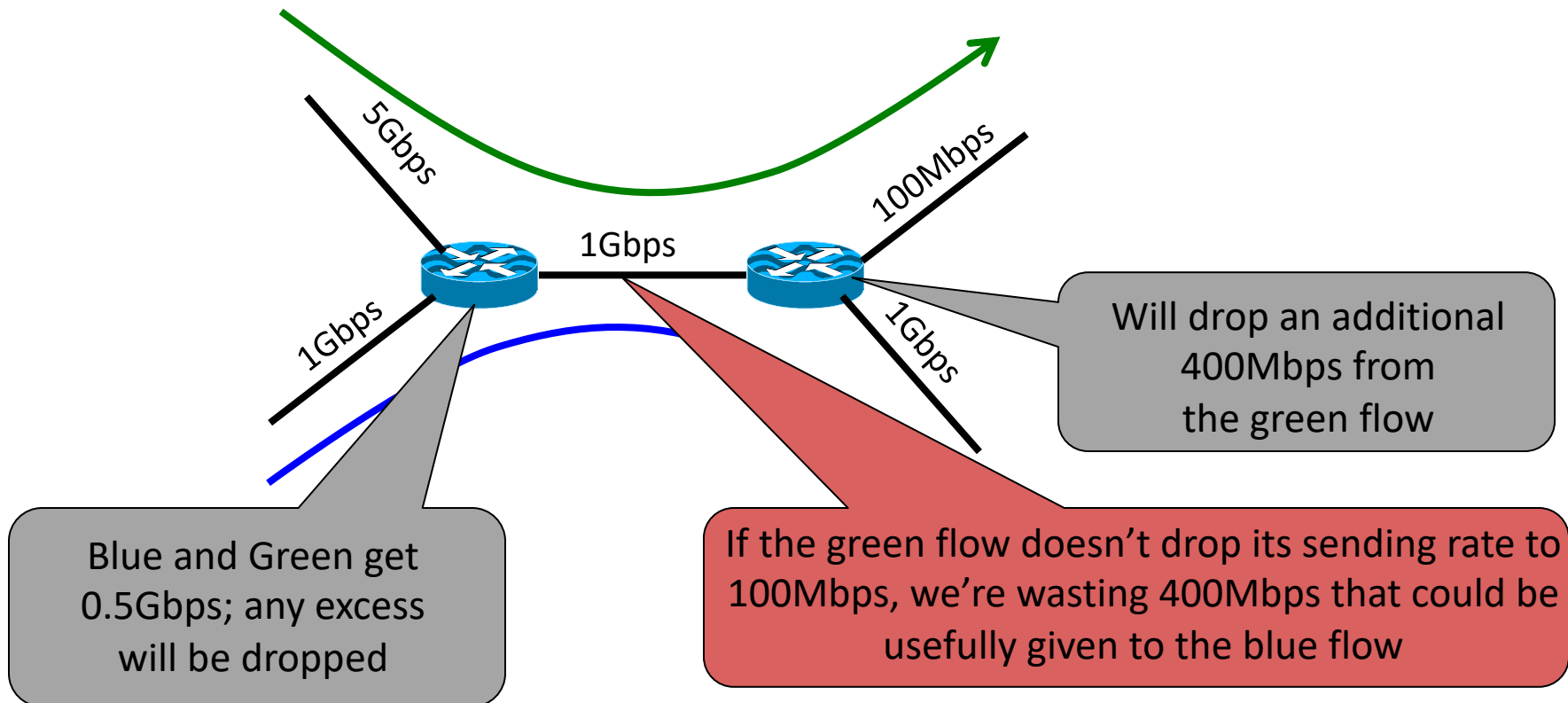
- Isolation: cheating flows don't benefit
- Bandwidth share does not depend on RTT
- Flows can pick any rate adjustment scheme they want

- **Disadvantages:**

- More complex than FIFO: per flow queue/state, additional per-packet book-keeping

# FQ in the big picture

- FQ does not eliminate congestion → it just manages the congestion



# FQ in the big picture

- **FQ does not eliminate congestion → it just manages the congestion**
  - Robust to cheating, variations in RTT, details of delay, reordering, retransmission, etc.
- **But congestion (and packet drops) still occurs**
- **We still want end-hosts to discover/adapt to their fair share!**
- **What would the end-to-end argument say w.r.t. congestion control?**

# Fairness is a controversial goal

- **What if you have 8 flows, and I have 4?**
  - Why should you get twice the bandwidth?
- **What if your flow goes over 4 congested hops, and mine only goes over 1?**
  - Why shouldn't you be penalized for using more scarce bandwidth?
- **What is a flow anyway?**
  - TCP connection
  - Source-Destination pair?
  - Source?

# Router-Assisted Congestion Control

- **CC has three different tasks:**

- Isolation/fairness
- Rate adjustment
- Detecting congestion

# Why not let routers tell what rate end hosts should use?

- **Packets carry “rate field”**
- **Routers insert “fair share”  $f$  in packet header**
- **End-hosts set sending rate (or window size) to  $f$** 
  - Hopefully (still need some policing of end hosts!)
- **This is the basic idea behind the “Rate Control Protocol” (RCP) from Dukkupati et al. '07**
  - Flows react faster

# Router-Assisted Congestion Control

- **CC has three different tasks:**

- Isolation/fairness
- Rate adjustment
- Detecting congestion

# Explicit Congestion Notification (ECN)

- **Single bit in packet header; set by congested routers**
  - If data packet has bit set, then ACK has ECN bit set
- **Many options for when routers set the bit**
  - Tradeoff between (link) utilization and (packet) delay
- **Congestion semantics can be exactly like that of drop**
  - i.e., end-host reacts as though it saw a drop



# ECN

- **Advantages:**

- Don't confuse corruption with congestion; recovery w/ rate adjustment
- Can serve as an early indicator of congestion to avoid delays
- Easy (easier) to incrementally deploy
  - Today: defined in RFC 3168 using ToS/DSCP bits in the IP header
  - Common in datacenters

# One final proposal: Charge people for congestion!

- **Use ECN as congestion markers**
- **Whenever I get an ECN bit set, I have to pay \$\$**
- **Now, there's no debate over what a flow is, or what fair is...**
- **Idea started by Frank Kelly at Cambridge**
  - “Optimal” solution, backed by much math
  - Great idea: simple, elegant, effective
  - Unclear that it will impact practice

# Group Discussion

- **Topic: fairness**

- What is ECN? How can we modify the TCP congestion control algorithm (e.g., TCP newReno) to leverage ECN?

- **Discuss in groups, and each group chooses a leader to summarize the discussion**

- **Everyone should speak.**

- **Turn on your audio and video. Do not mute.**

# Summary

- **IP routers form the backbone of the Internet**
- **Aims for speed while providing fairness**
- **Routers can assist in addressing/mitigating many of TCP's shortcomings**

Thanks!  
Q&A