Network Layer

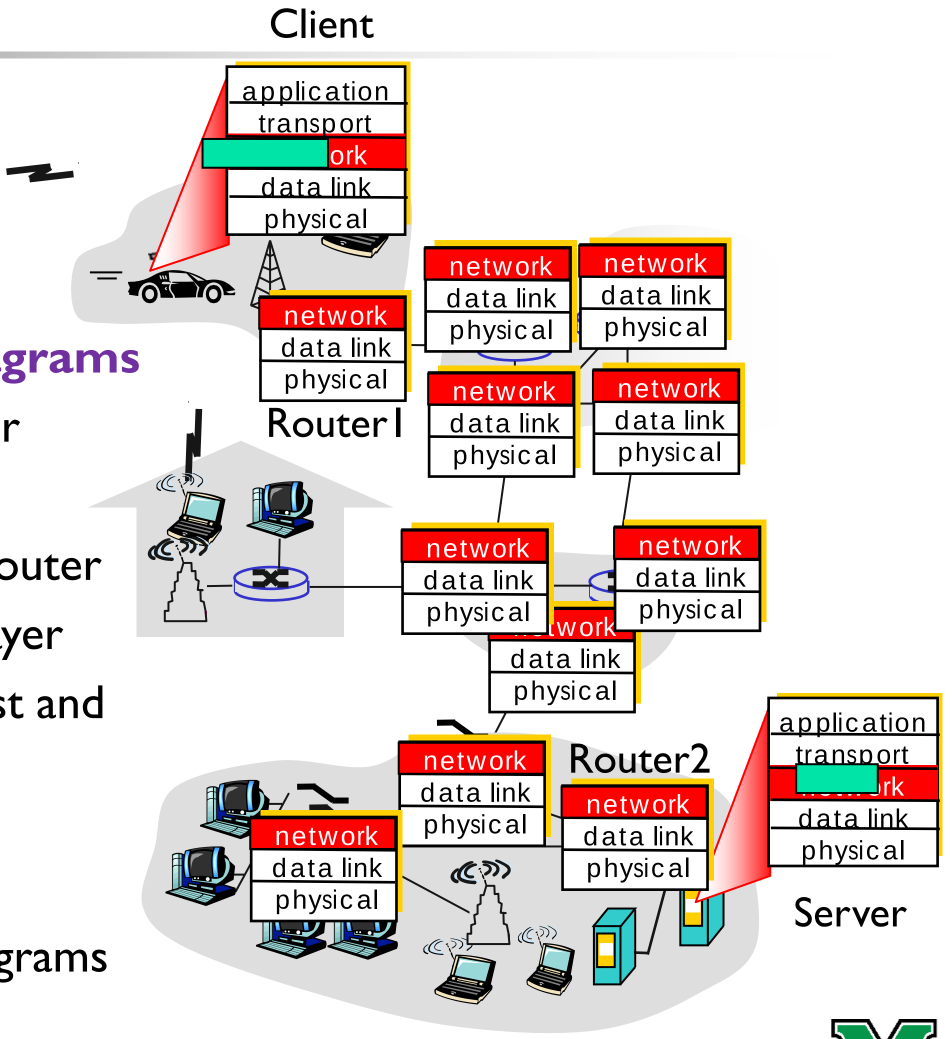
Instructor: C. Pu (Ph.D., Assistant Professor)

Lecture 13

puc@marshall.edu

Network Layer

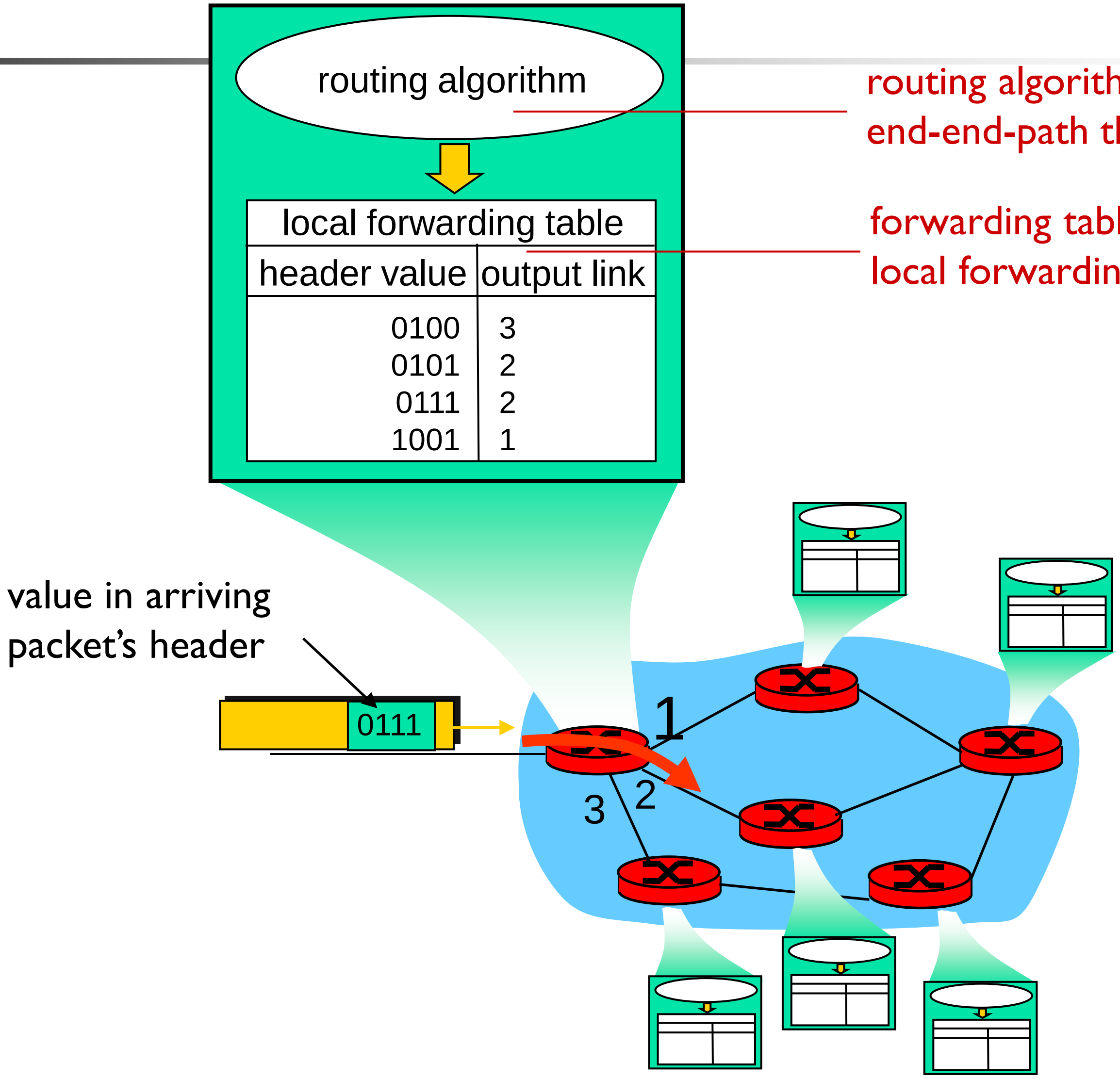
- transport segment from sending to receiving host
- on sending side
  - encapsulates segments into **datagrams**
  - sends datagrams to nearby router
- on receiving side
  - receive datagrams from nearby router
  - delivers segments to transport layer
- ***network layer*** protocols in every host and router
- router examines header fields in all IP datagrams passing through it
  - ***the role of router***: forward datagrams from input links to output links



Two Key Network-Layer Functions

- primary role of network layer: move packets from a sending host to a receiving host
- two important network-layer functions:
  - ***forwarding***:
    - move packets from router's input to appropriate router's output
  - ***routing***:
    - determine route taken by packets from source to destination
    - the algorithm that calculates these paths: routing algorithms
- analogy:
  - ***routing***: process of planning trip from source to destination
  - ***forwarding***: process of getting through single interchange

Interplay Between Routing and Forwarding



header value	output link
0100	3
0101	2
0111	2
1001	1

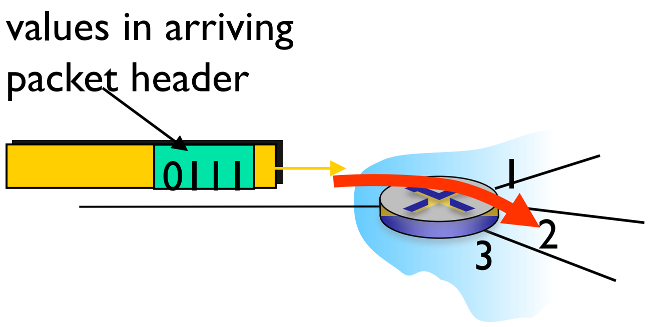
routing algorithm determines end-end-path through network

forwarding table determines local forwarding at this router

Network Layer: Data Plane vs Control Plane

Data plane

- Local and per-router function
- determines how datagram arriving on **router input port** is forwarded to **router output port**
- **forwarding function**

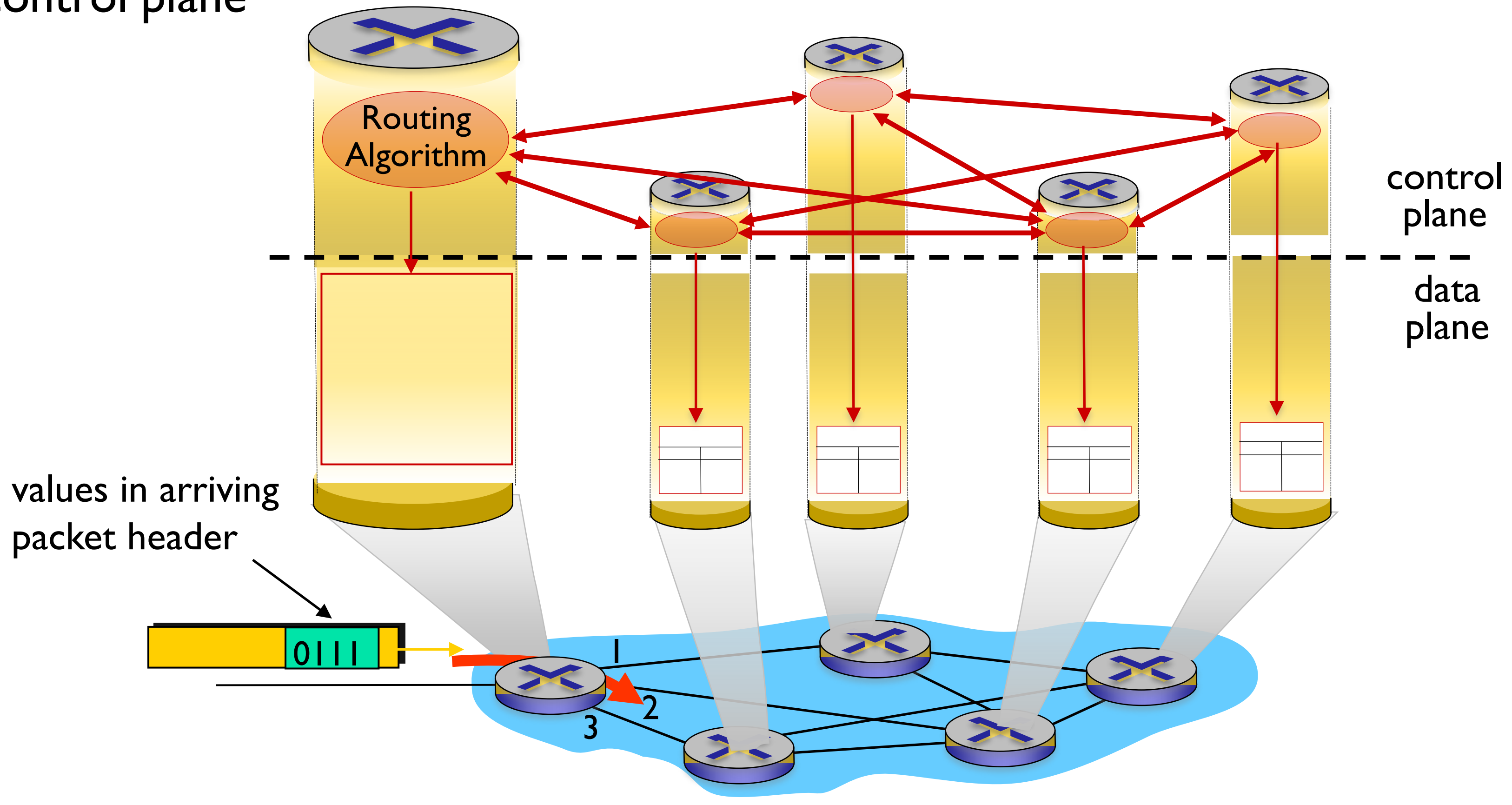


Control plane

- network-wide logic
- determines how datagram is routed among routers along **end-end path** from source host to destination host
- two control-plane approaches:
  - *traditional routing algorithms:* implemented in routers
  - *software-defined networking (SDN)*: implemented in (remote) servers

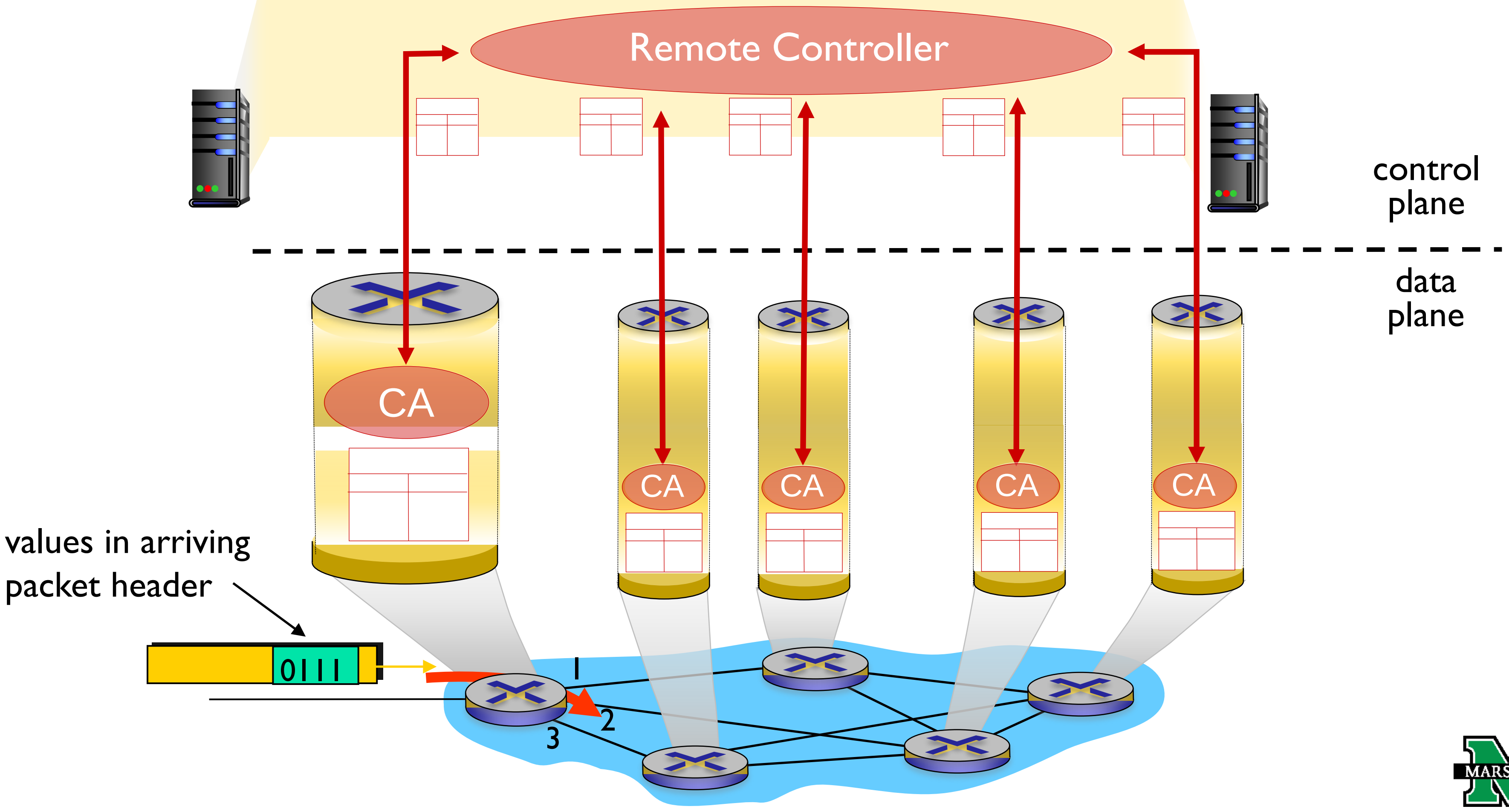
Per-router Control Plane: The Traditional Approach

Individual routing algorithm components *in each and every router* interact in the control plane



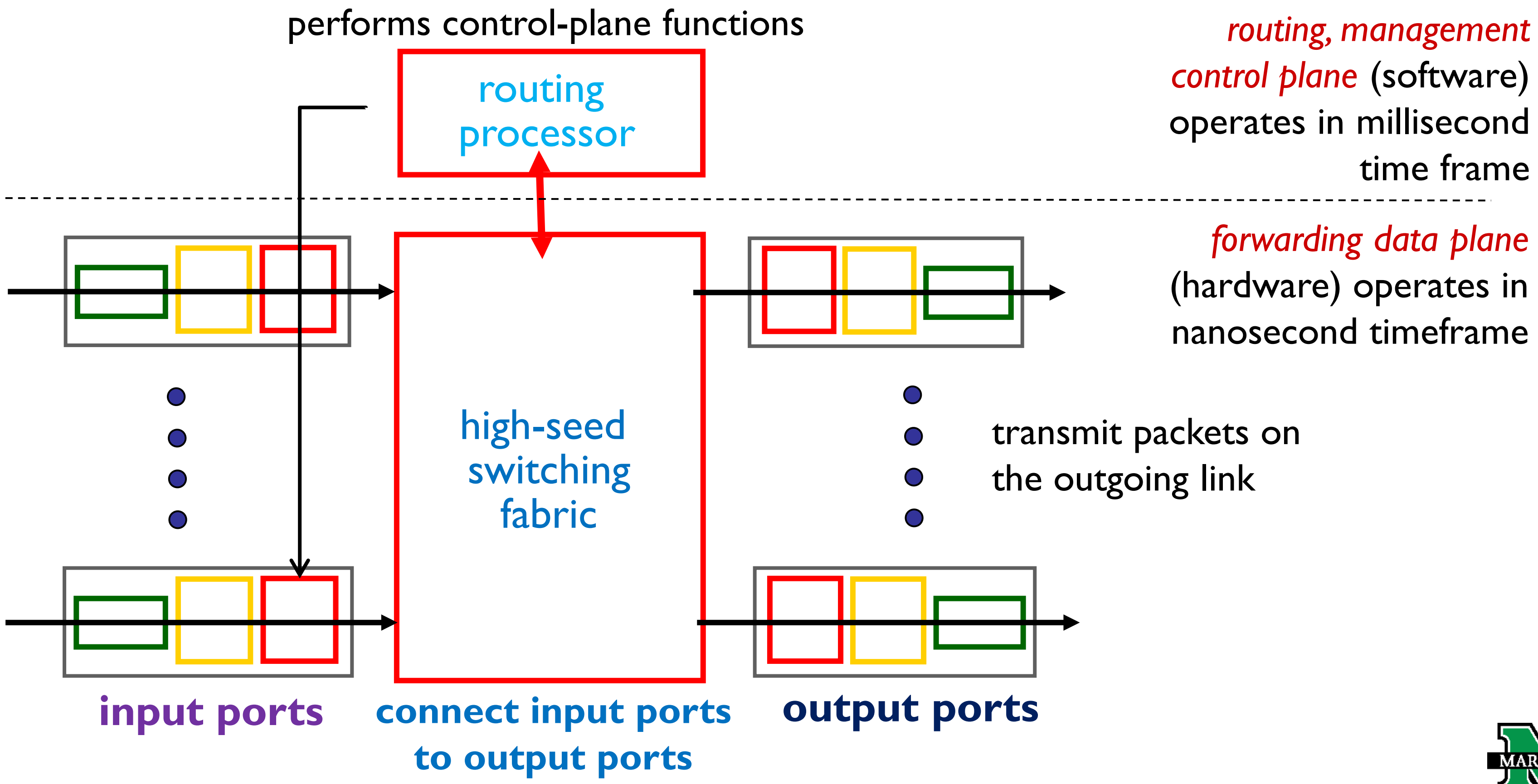
Logically Centralized Control Plane: The SDN Approach

A distinct (typically remote) controller interacts with local control agents (CAs)



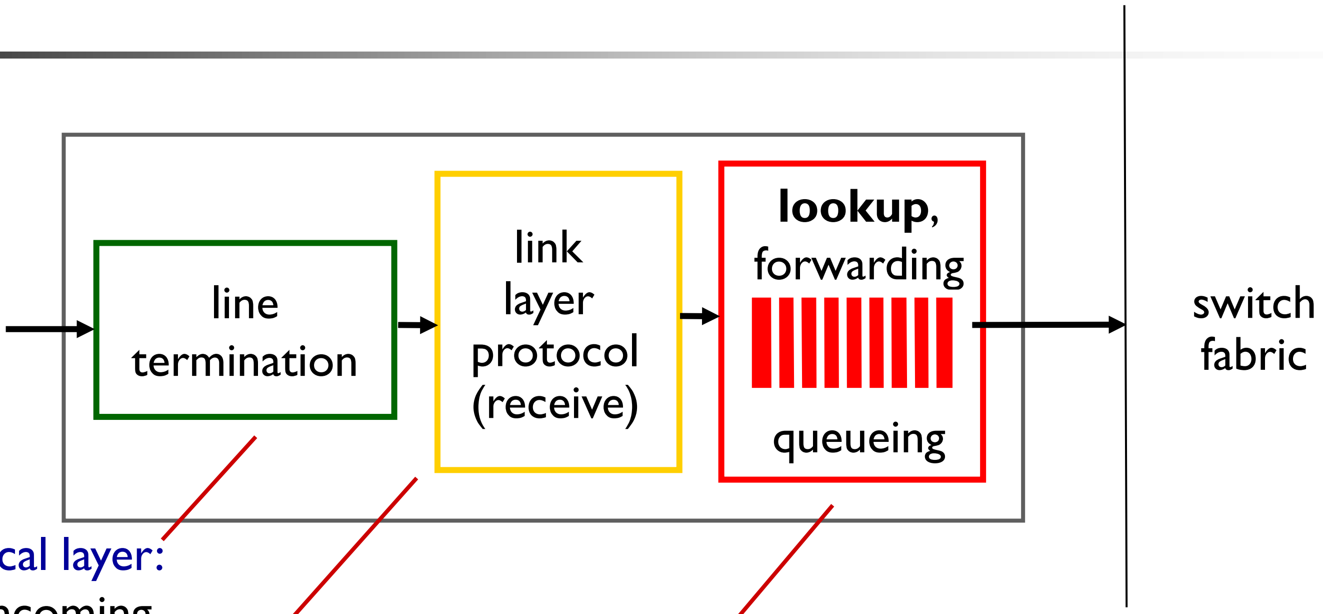
Router Architecture Overview

- high-level view of generic router architecture:



performs control-plane functions
routing processor
routing, management
control plane (software) operates in millisecond time frame
forwarding data plane (hardware) operates in nanosecond timeframe
high-seed switching fabric
transmit packets on the outgoing link
input ports
connect input ports to output ports
output ports

Input Port Functions



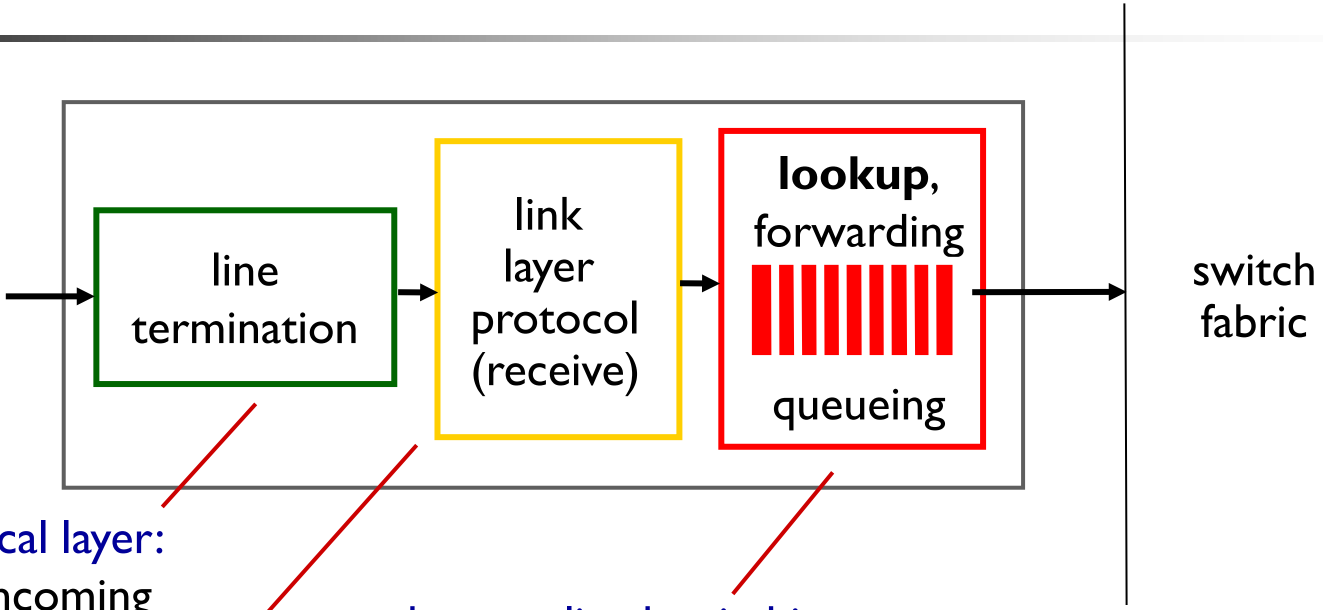
physical layer:
terminating an incoming physical link

data link layer:
interoperate with link layer

decentralized switching:

- using header field values, lookup output port using forwarding table in input port memory (*"match plus action"*)
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Input Port Functions



physical layer:
terminating an incoming physical link

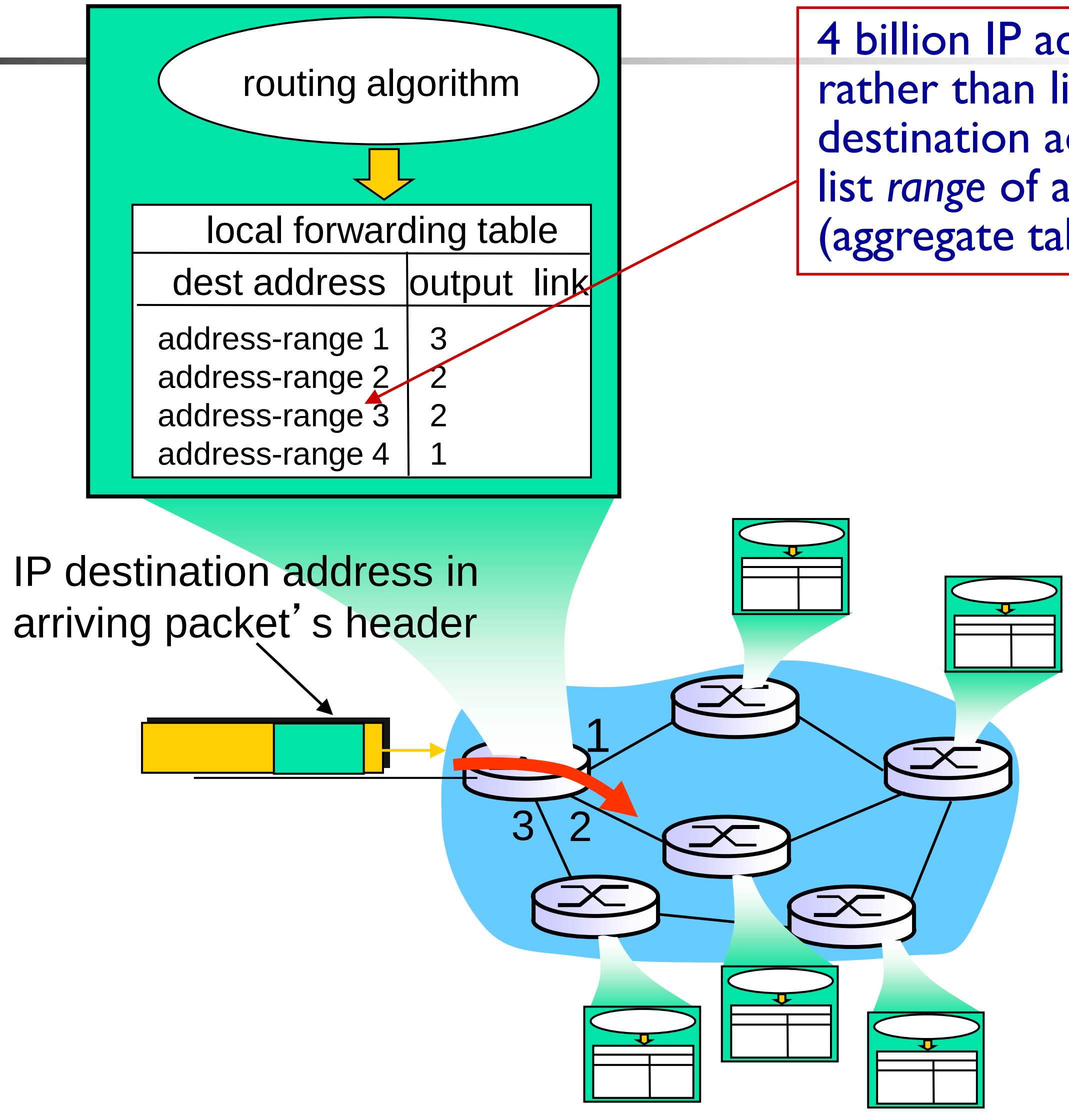
data link layer:
interoperate with link layer

decentralized switching:

- using header field values, lookup output port using forwarding table in input port memory *("match plus action")*
- *destination-based forwarding:* forward based only on destination IP address (traditional)
- *generalized forwarding:* forward based on any set of header field values

Datagram Networks: Forwarding Table

4 billion IP addresses, so rather than list individual destination address, list *range* of addresses (aggregate table entries)



local forwarding table	
dest address	output link
address-range 1	3
address-range 2	2
address-range 3	2
address-range 4	1

Datagram Networks: Forwarding Table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through **11001000 00010111 00010111 11111111**	0
11001000 00010111 00011000 00000000 through **11001000 00010111 00011000 11111111**	1
11001000 00010111 00011001 00000000 through **11001000 00010111 00011111 11111111**	2
otherwise	3

Datagram Networks: Longest Prefix Matching

longest prefix matching

when looking for forwarding table entry for given destination address, use ***longest* address prefix** that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** *********	0
11001000 00010111 00011000 *********	1
11001000 00010111 00011*** *********	2
otherwise	3

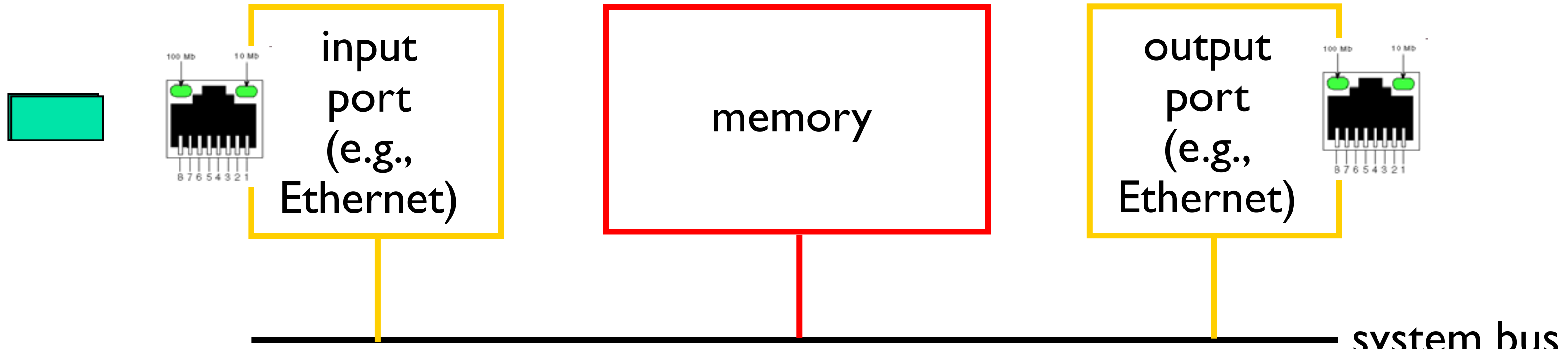
examples:

DA: 11001000 00010111 00010110 10100001 which interface?

DA: 11001000 00010111 00011000 10101010 which interface?

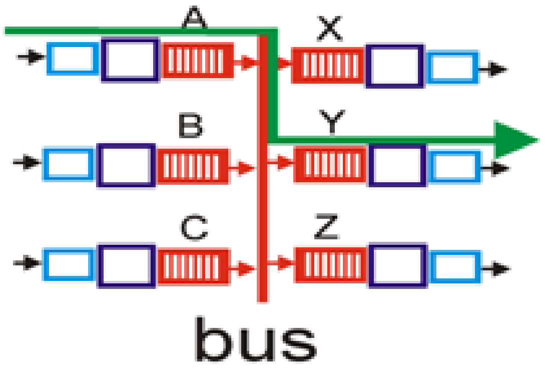
Three Types of Switching Fabrics: Switching Via Memory

- first generation routers:
  - traditional computers with switching under direct control of CPU
  - packet copied to system's memory
  - speed limited by memory bandwidth (2 bus crossings per datagram)
    - two packets cannot be forwarded at the same time
    - only one memory read/write over the shared bus can be done at a time



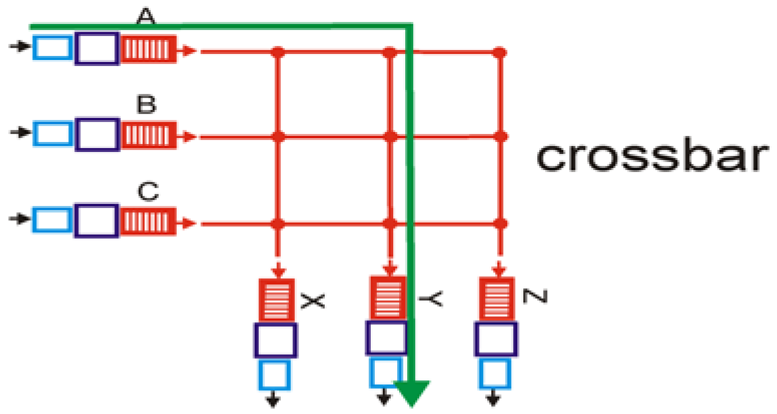
Three Types of Switching Fabrics: Switching Via a Bus

- datagram from input port memory to output port memory via a shared bus
- bus contention: switching speed limited by bus bandwidth
  - Only one packet can cross the bus at a time
- for example, 32 Gbps bus, Cisco 5600
  - sufficient speed for access and enterprise routers



Three Types of Switching Fabrics: Switching Via an Interconnection Network

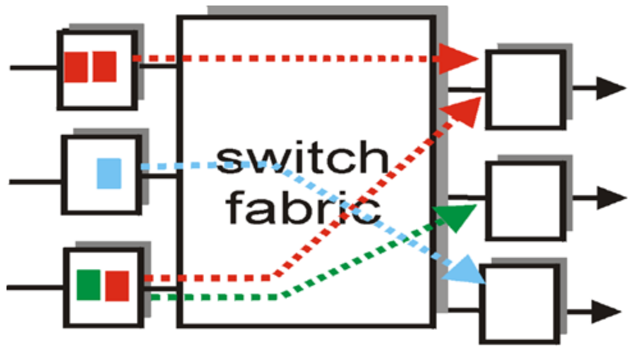
- overcome bus bandwidth limitations
- interconnection network initially developed to connect processors in multiprocessor
- Crossbar networks,
  - forwarding multiple packets in parallel
- for example, Cisco 12000
  - switches 60 Gbps through the interconnection network



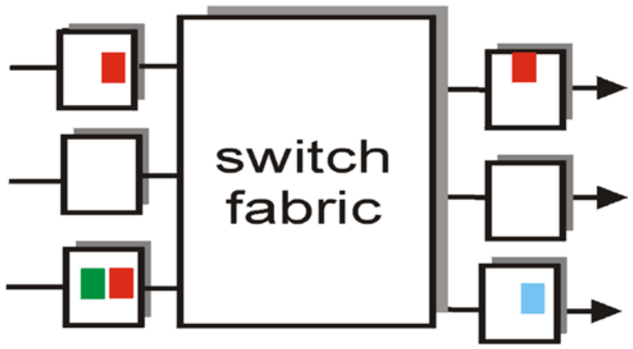
multiple packets can be transferred in parallel,
→ as long as their output ports are different

Input Port Queuing

- fabric slower than input ports combined -> queueing may occur at input queues
- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward



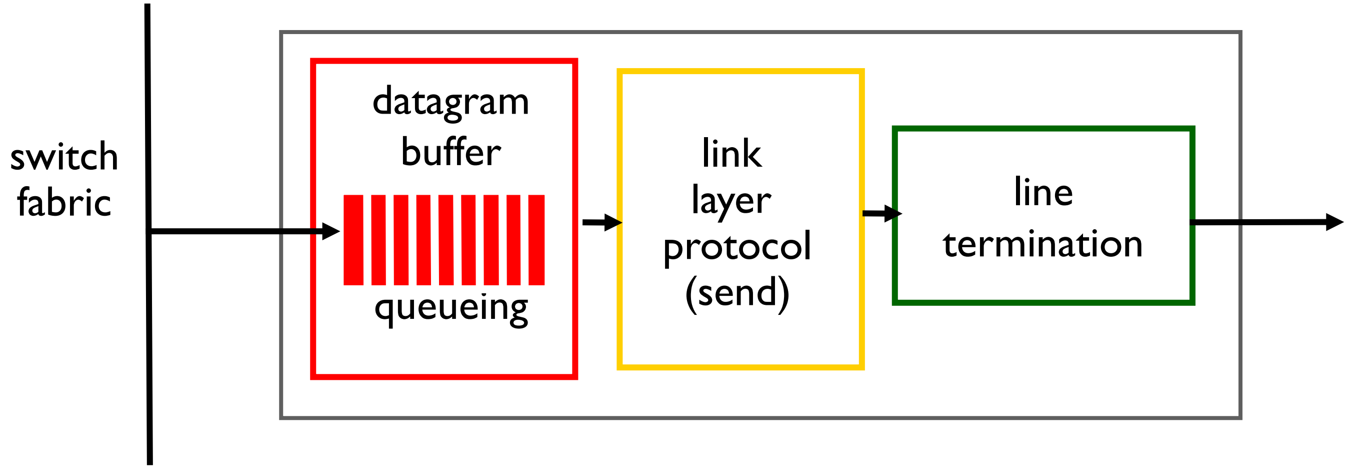
output port contention at time t - only one red packet can be transferred



green packet experiences HOL blocking

If two packets at the front of two input queues are destined for the same output queue,
→ one of the packets will be blocked and must wait at the input queue

Output Ports



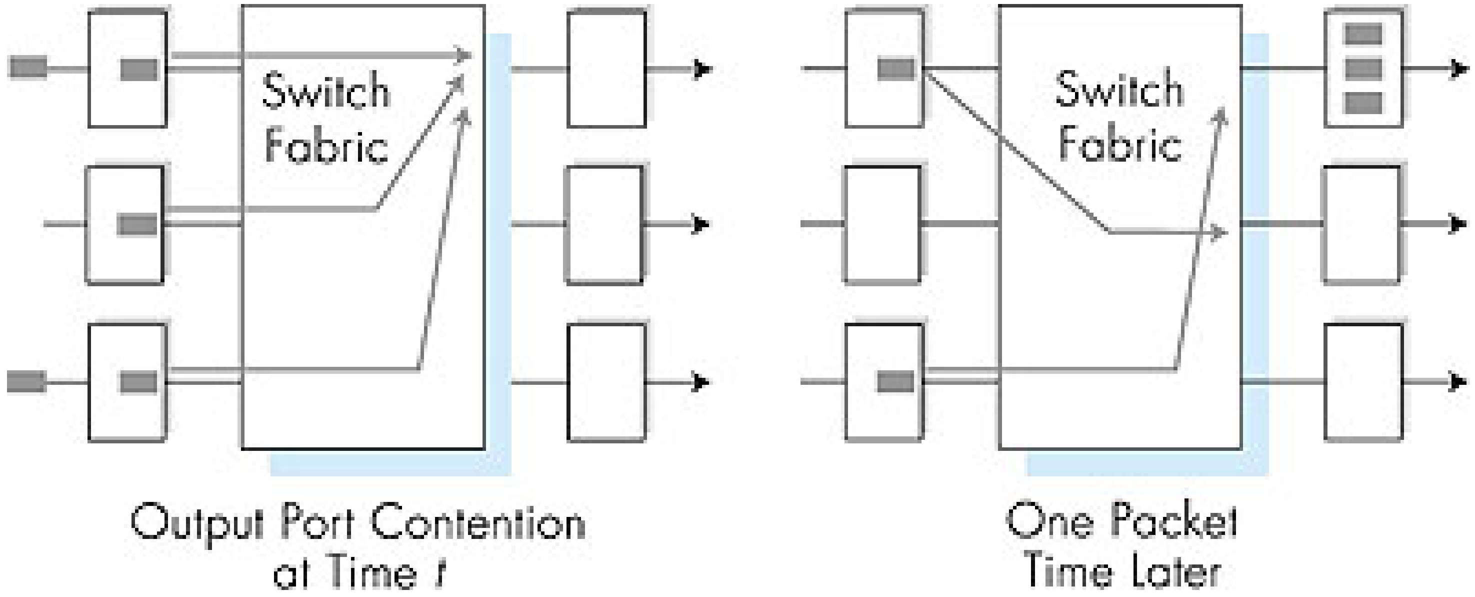
- *buffering* required when datagrams arrive from fabric faster than the transmission rate

Datagram (packets) can be lost due to congestion, lack of buffers

- *scheduling discipline* chooses among queued datagrams for transmission

Priority scheduling – who gets best performance, network neutrality

Output Port Queueing



- buffering when arrival rate via switch exceeds output line speed
- *queueing (delay) and loss due to **output** port buffer overflow!*
- **packet scheduler** at the output port
  - choose one packet among those queued for transmission, FCFS, etc.

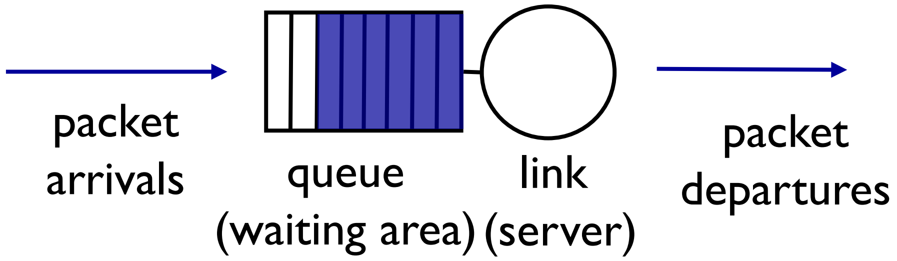
How much buffering?

- RFC 3439 rule of thumb:
  - average buffering equal to "typical" RTT (say 250 msec) times link capacity C
  - e.g., C = 10 Gbps link: 2.5 Gbit buffer
- recent recommendation: with *N* flows, buffering equal to

$$\frac{RTT \cdot C}{\sqrt{N}}$$

Scheduling Mechanisms

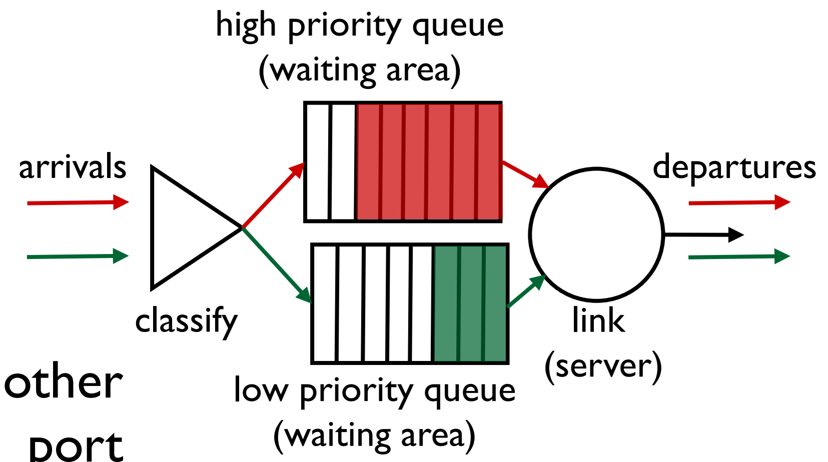
- *scheduling:* choose next packet to send on link
- *FIFO (first in first out) scheduling:* send in order of arrival to queue
  - real-world example?
  - *discard policy:* if packet arrives to full queue: who to discard?
    - *tail drop:* drop arriving packet
    - *priority:* drop/remove on priority basis
    - *random:* drop/remove randomly

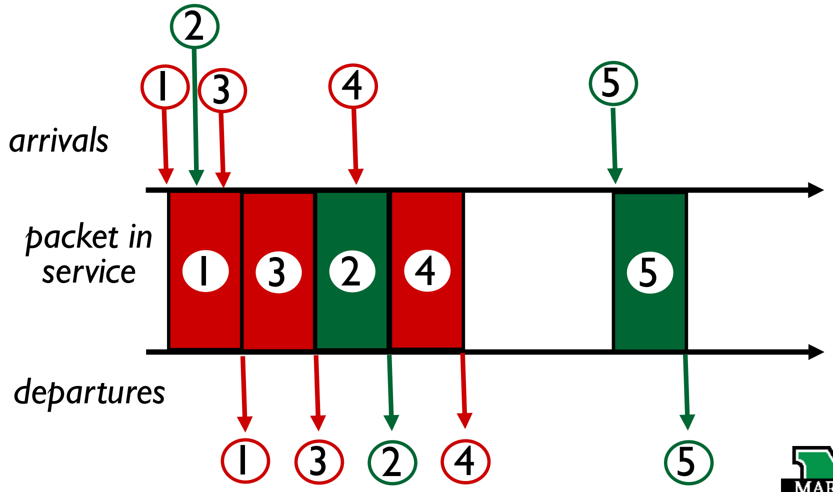


Scheduling Policies: Priority

priority scheduling: send highest priority queued packet

- multiple *classes*, with different priorities
  - class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc.

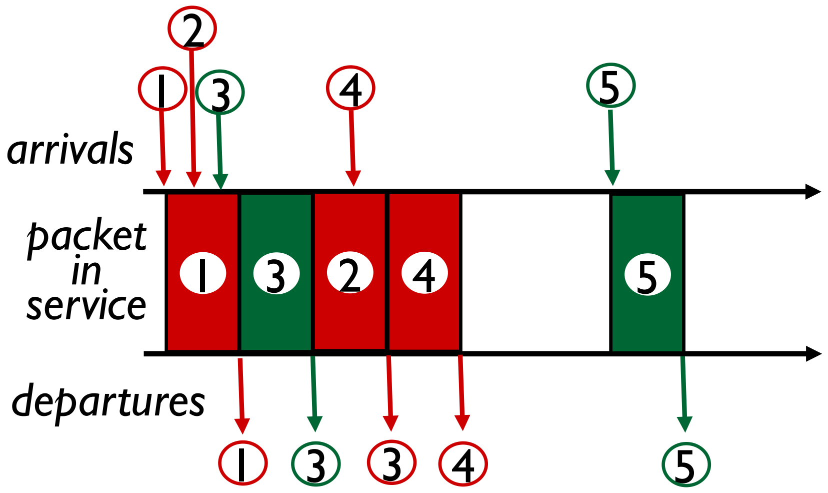




Scheduling Policies: Round Robin

Round Robin (RR) scheduling:

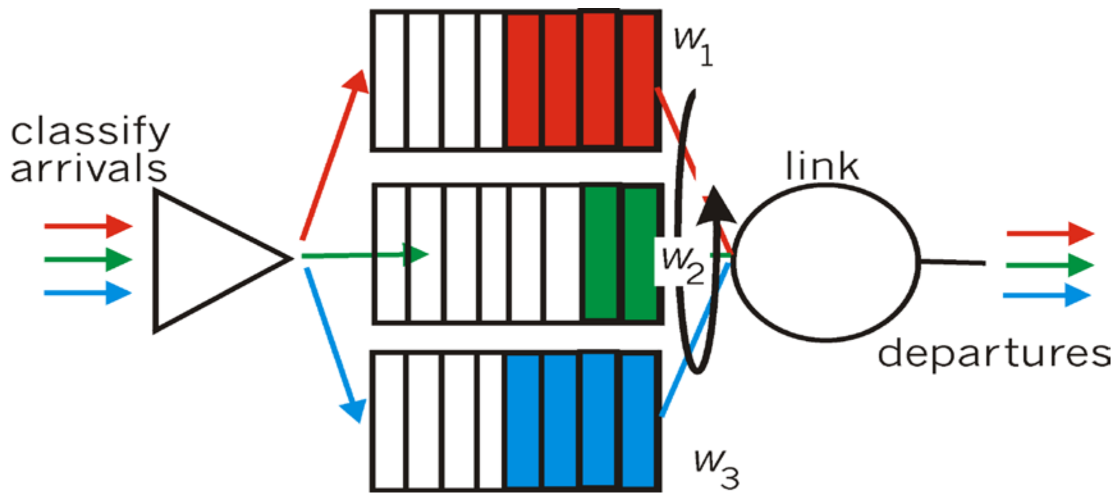
- multiple classes
- cyclically scan class queues, sending one complete packet from each class (if available)



Scheduling Policies: Weighted Fair Queuing

Weighted Fair Queuing (WFQ):

- generalized Round Robin
- each class gets weighted amount of service in each cycle



classify arrivals
w_1
w_2
w_3
link
departures