Networks

Network Layer - Introduction

Adopted from material in “Computer Networking: A Top Down Approach” by Kurose and Ross and slides developed by William Conner

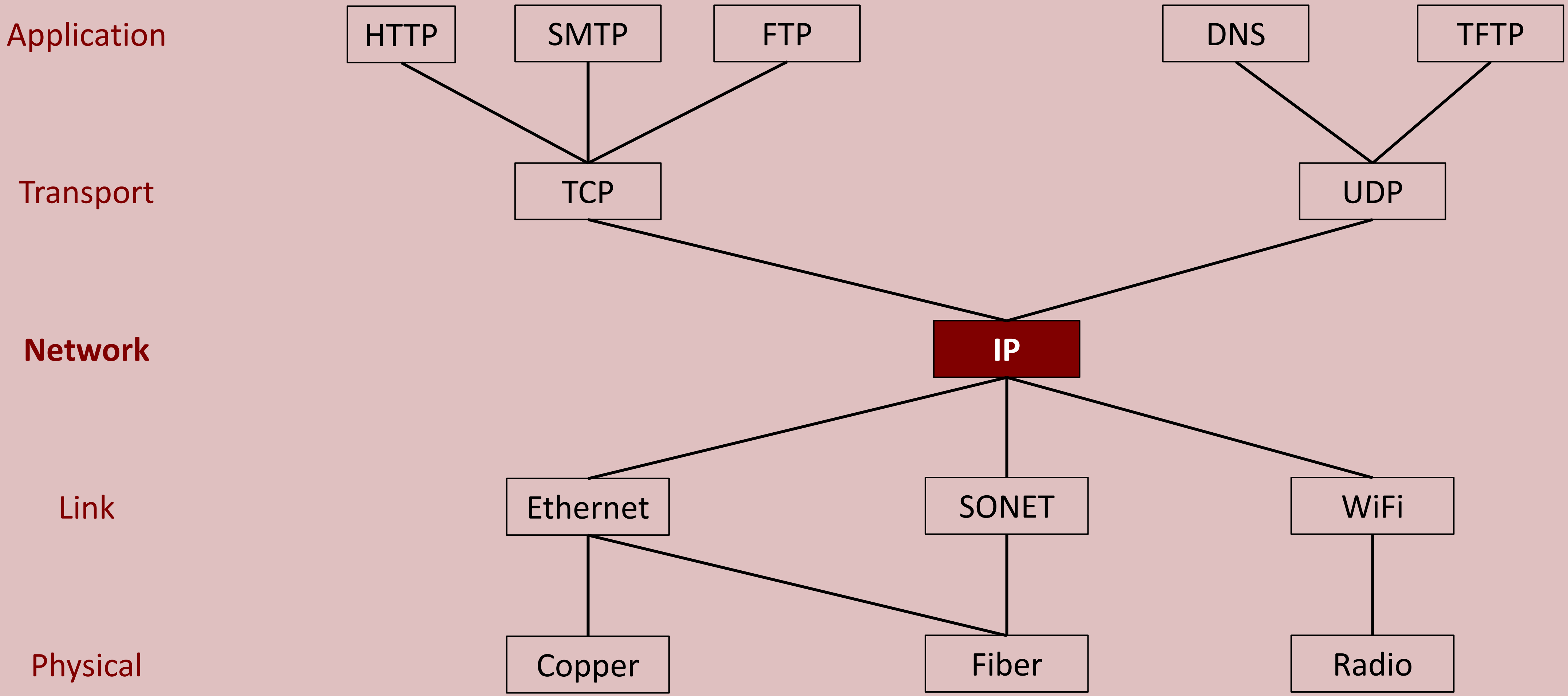
Network Layer - Introduction

Section 4.1

Network Layer

- Transports either TCP segments or UDP datagrams from sending host to receiving host
  - Encapsulates transport layer data into packets at sender
  - Delivers transport layer data at receiver
- Runs on every host and router
  - IP is common layer across entire Internet
  - IP enables internetworking

"Narrow Waist"



Network Layer Functions

- Forwarding: move packets from router input ports to appropriate router output ports

- Routing: determine route taken by packet from source to destination

Data vs. Control Plane

- Data plane (this week)
  - Local, per-router function
  - Forwarding
- Control plane (next week)
  - Network-wide logic
  - Routing
    - Traditional routing algorithms
    - Software-defined networking (SDN)

Network Layer Services

Service	TCP*	IP	ATM**
Best effort		✓	✓
Reliable delivery	✓		✓
In-order delivery	✓		✓
Congestion control	✓		✓
Delay guarantees			✓
Bandwidth guarantees			✓

* Included transport layer protocol for comparison
** Actually depends on the specific service class

Asynchronous Transfer Mode (ATM)

- Developed in the late 1980s and 1990s
- Unrelated to automated teller machines
- Telecommunications standard for carrying voice, video, and data
- Eventually lost out to IP

ATM

- Virtual circuits over packet-switched networks
- Variations offered bandwidth, delay, loss, and ordering guarantees
- Support multiple service classes (CBR, ABR, VBR, and UBR)

Thank You!

Networks

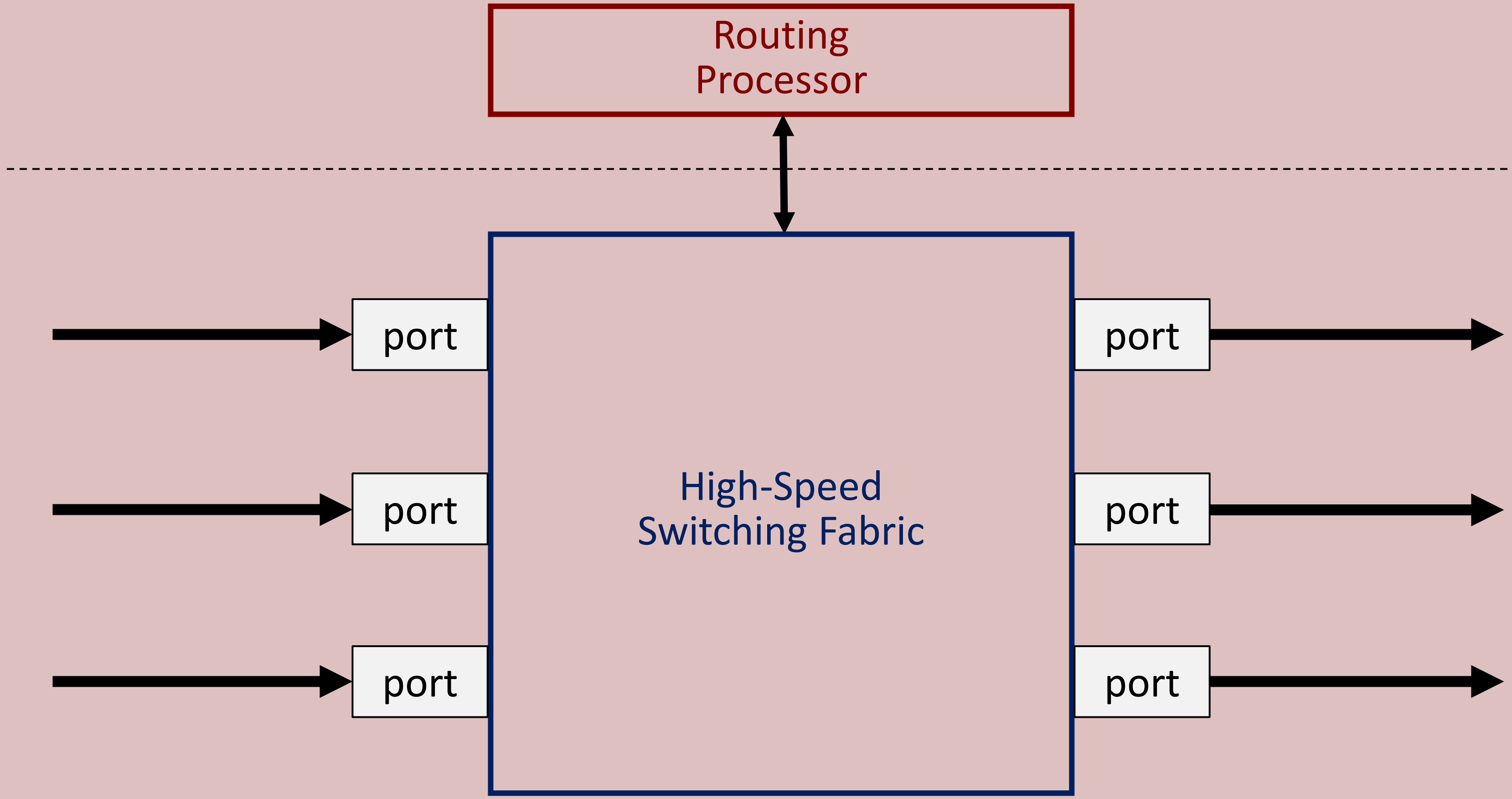
Routers

Adopted from material in “Computer Networking: A Top Down Approach” by Kurose and Ross and slides developed by William Conner

Routers

Section 4.2

Router Architecture



Routing management control plane software operates on millisecond time frame

Forwarding data plane hardware operates on nanosecond timeframe

Switching Functions

- Lookup
  - Determine output port using forwarding table in input port memory
- Forwarding
  - Use destination IP address (traditional)
  - Use any set of header fields (generalized)
- Queueing
  - Temporarily hold packets if arrival rate exceeds forwarding rate of switching fabric

Destination-Based Forwarding

Forwarding Table	
Destination Address Range	Output Port
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
Default	3

What if ranges don't divide up so nicely?

Longest Prefix Matching

Choose longest address prefix that matches destination address

Destination Address Range	Output Port
11001000 00010111 00010*** *********	0
11001000 00010111 00011000 *********	1
11001000 00010111 00011*** *********	2
Default	3

Examples:

Destination Address: 11001000 00010111 00010110 10100001

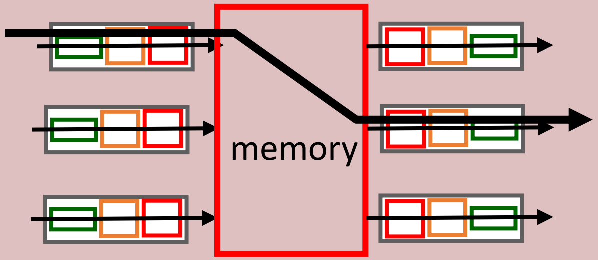
Destination Address: 11001000 00010111 00011000 10101010

Switching Fabric

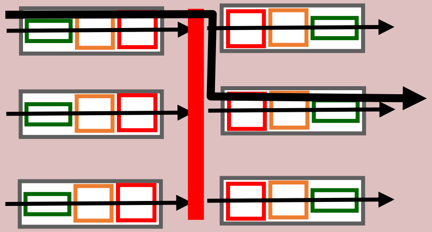
- Transfers packets from input buffer to appropriate output buffer
- Switching rate: rate at which packets can be transferred from inputs to outputs
- Operates on the order of nanoseconds (why?)

Switching Fabric

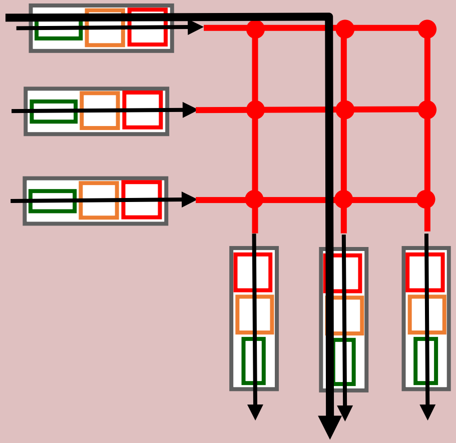
- Three types of switching fabric



memory



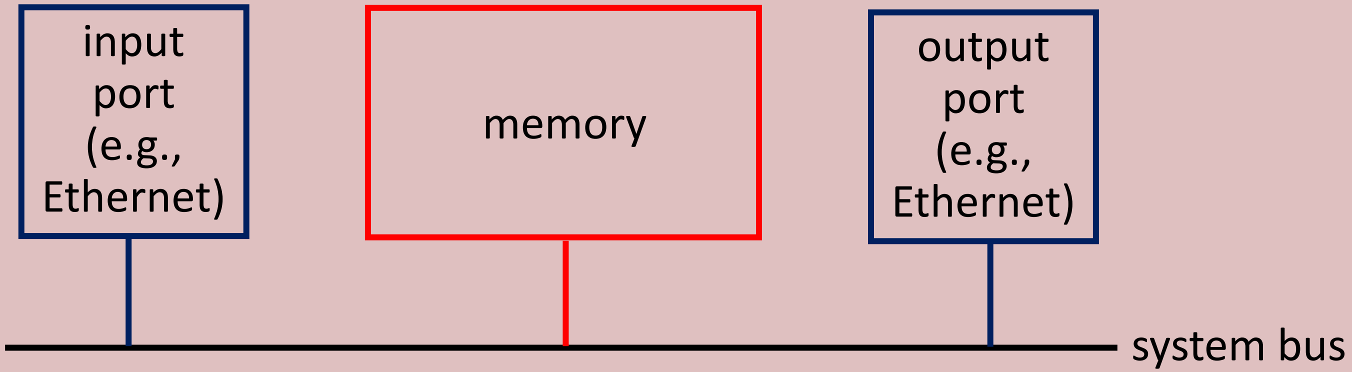
bus



crossbar

Switching via Memory

- Used by first generation routers
- CPU directly controls switching on a traditional computer
- Packets are copied to memory
- Speed limited by memory bandwidth
  - Two bus crossings per datagram

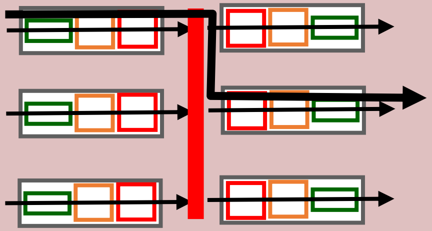


Switching via Bus

- Packets transferred from input port memory to output port memory via shared bus

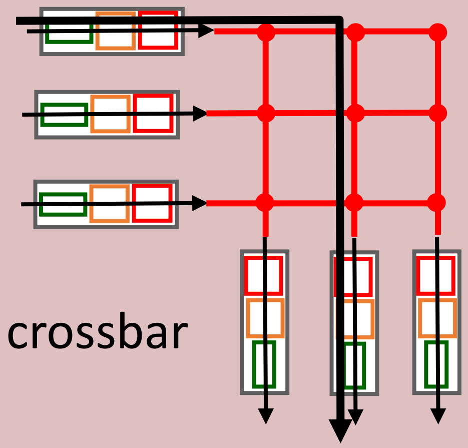
- Switching speed limited by bus bandwidth (bus contention)

- Example: Cisco 5600
  - 32 Gbps bus
  - Sufficient speed for access and enterprise routers



Switching via Interconnection Network

- Overcomes bus bandwidth limitations
- Originally developed to connect multiprocessors (e.g., banyan and crossbar switching)
- Advanced design
  - Fragment packet into fixed length cells
  - Switch cells through fabric
- Example: Cisco 12000
  - Switches 60 Gbps
  - “Carrier class” backbone network



Input Port Queuing

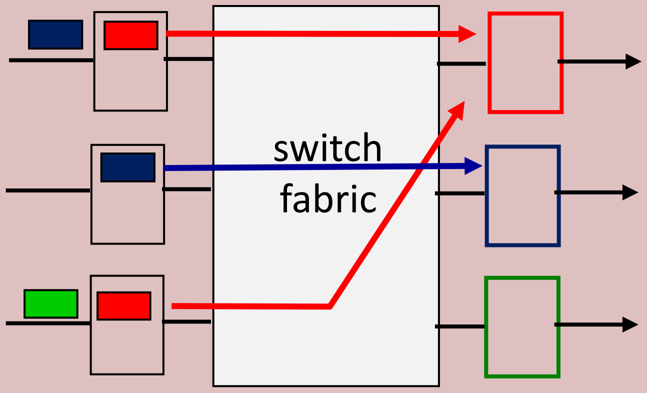
- Can occur if switching fabric rate is lower than aggregate input line rate

- Large input queues
  - Queuing delays
  - Packet loss (input buffer overflow)

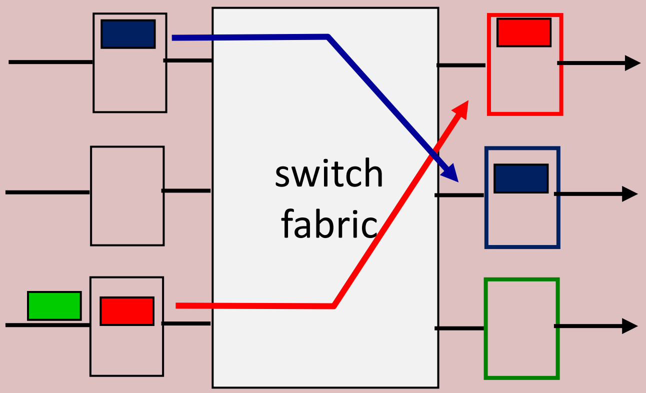
What transport layer feedback mechanism helps to reduce large input queues at routers?

Head-of-Line (HOL) Blocking

- Occurs if packet at front of queue prevents other queued packets from moving forward



output port contention:
only one red datagram can be transferred.
lower red packet is blocked



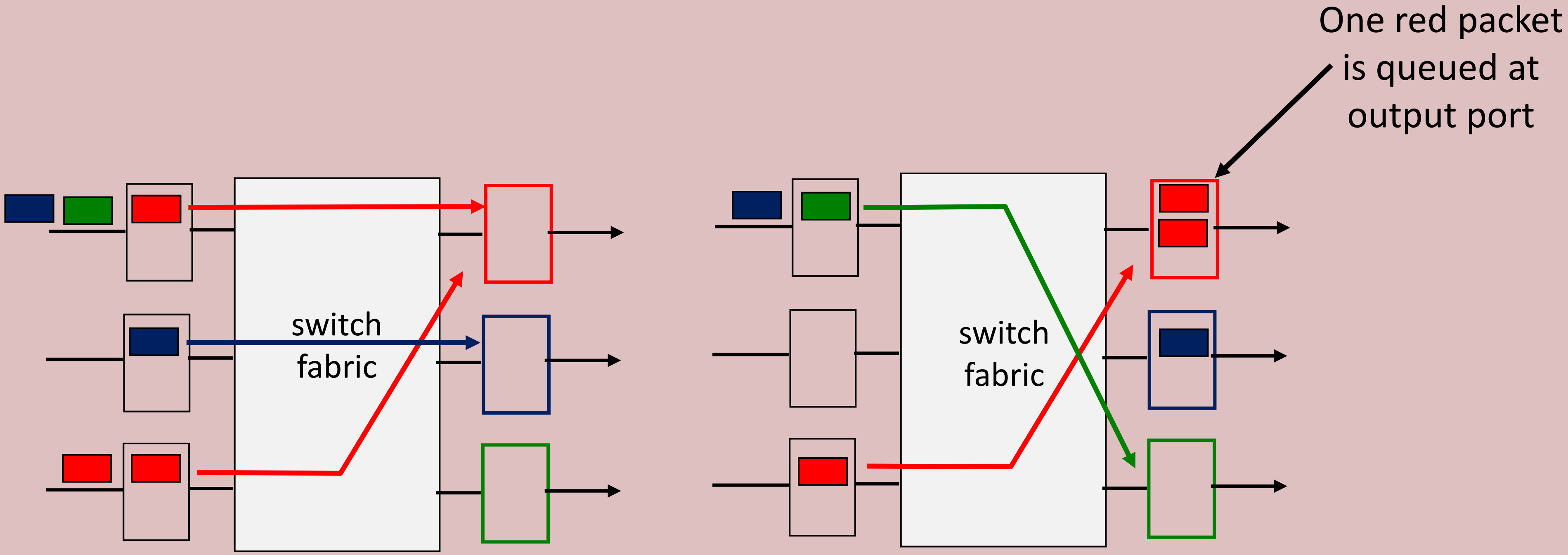
one packet time later:
green packet experiences HOL blocking

Output Ports

- Buffering: required when packets arrive from fabric faster than output line transmission rate
  - Queuing delays
  - Packet loss

- Scheduling discipline: choose among queued packets for next transmission on link

Output Port Queuing

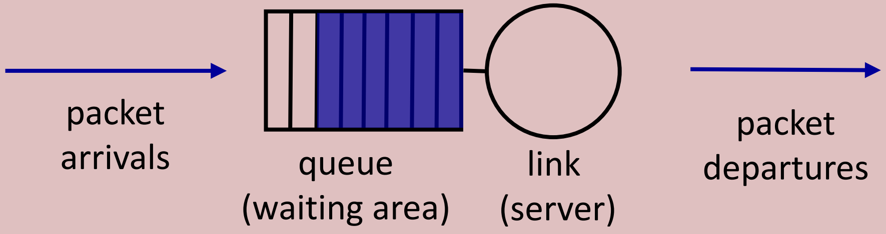


at *t,* packets more
from input to output

a little time later

FIFO Scheduling

- First in, first out (FIFO)



- Real-world examples
  - Sending a package at the post office
  - Boarding the bullet train in Japan

- Send packets in order of arrival

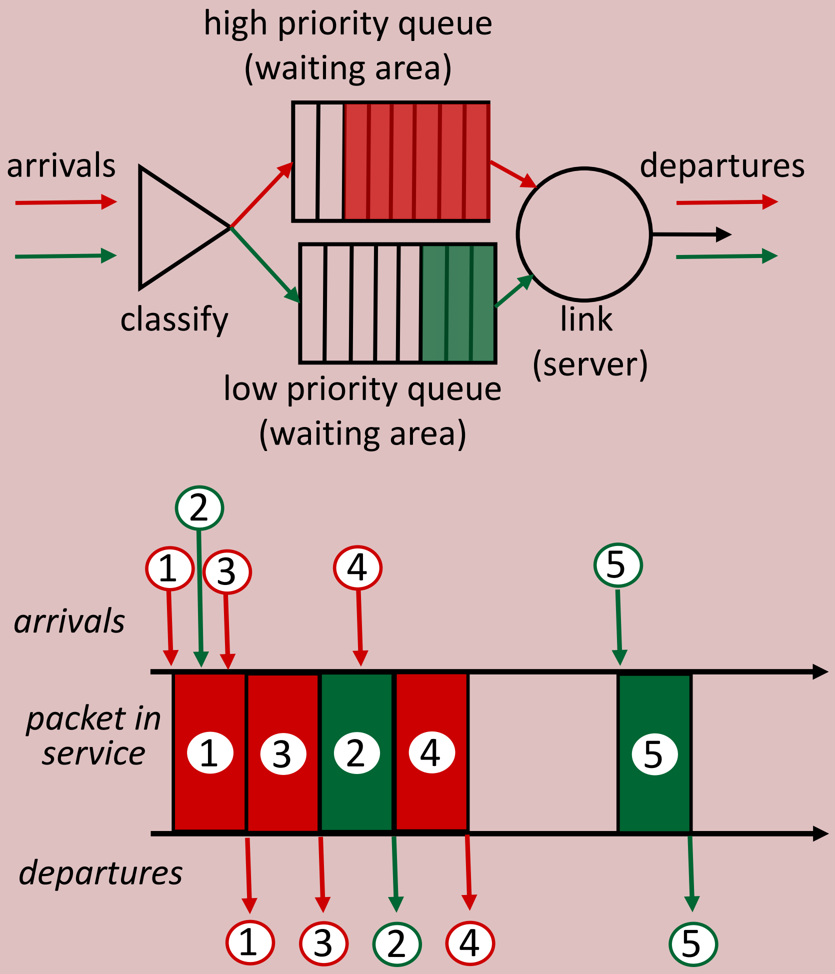
- Used in conjunction with discard policy

Discard Policies

- Determines which packet will be dropped when new packet arrives to full queue
- Tail: drop newly arriving packet
- Priority: drop/remove on a priority basis
- Random: drop/remove random packet

Priority Scheduling

- Send highest priority queued packet
- Packets separated by traffic classes with different priorities
  - Packet marking
  - Packet header (e.g., IP address and/or port)
- What are some real world examples of priority scheduling?



Question

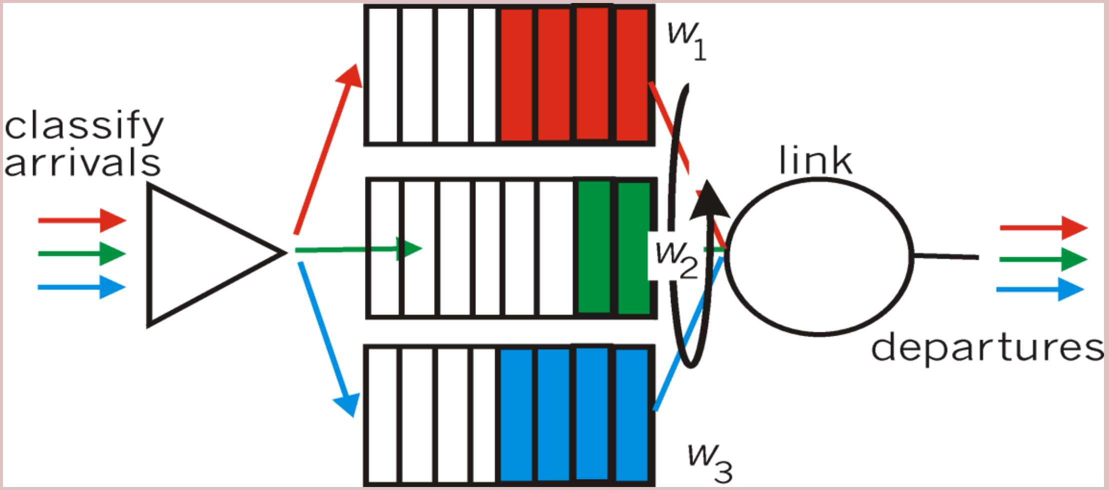
What will happen to lower priority packets during a continuous stream of higher priority packets?

Round Robin Scheduling

- Multiple classes
- Cyclically scan queues
- Send one packet from each queue
- What are some real world examples of round robin scheduling?

Weighted Fair Queuing

- Generalized round robin
- Each class is assigned weight proportional to amount of service received in each cycle



Thank You!

Networks

Internet Protocol

Adopted from material in “Computer Networking: A Top Down Approach” by Kurose and Ross and slides developed by William Conner

Internet Protocol

Section 4.3

IPv4 Datagram Header

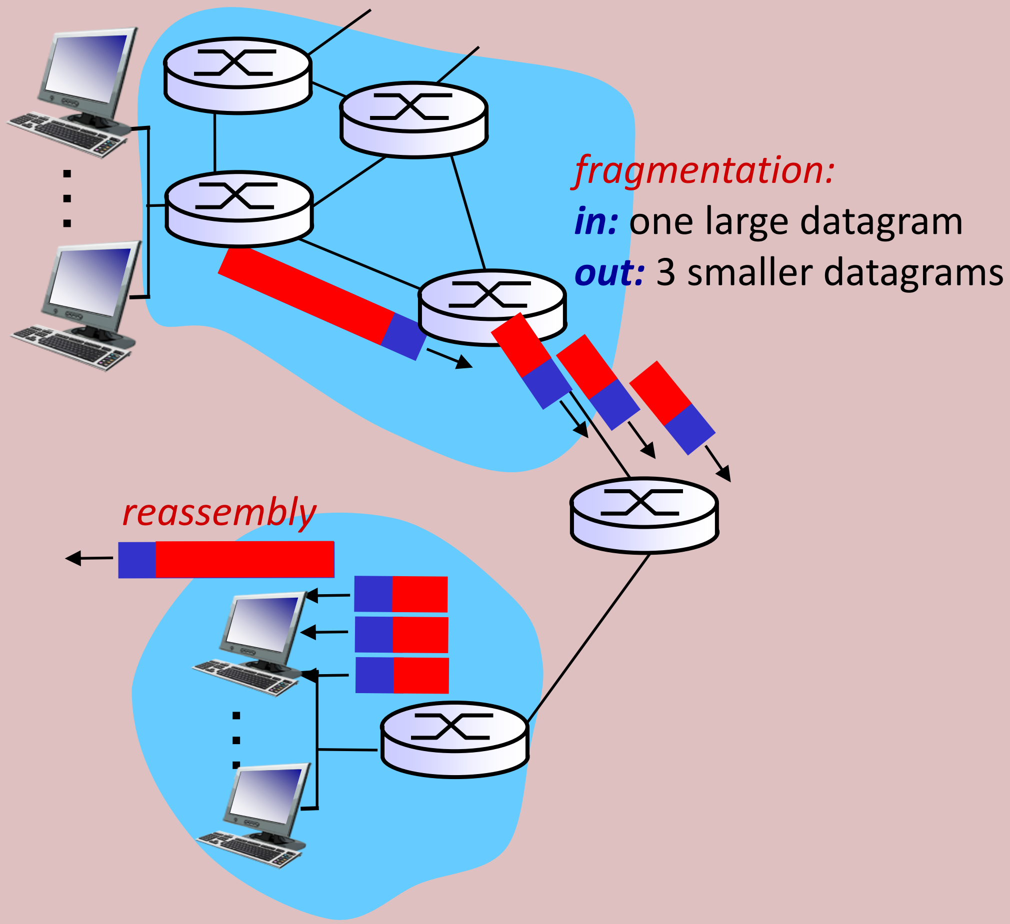
<table>
<tr><td>version
(4 bits)</td><td>IHL
(4 bits)</td><td>diffserv
(6 bits)</td><td>ECN
(2)</td><td colspan="2">total length (16 bits)</td></tr>
<tr><td colspan="4">identification (16 bits)</td><td>flags
(3)</td><td>fragmentation offset (13 bits)</td></tr>
<tr><td colspan="2">TTL (8 bits)</td><td colspan="2">protocol (8 bits)</td><td colspan="2">checksum (16 bits)</td></tr>
<tr><td colspan="6">source IP address (32 bits)</td></tr>
<tr><td colspan="6">destination IP address (32 bits)</td></tr>
<tr><td colspan="6">options (variable length)</td></tr>
</table>

IPv4 Datagram Header

- Version: IPv4 or IPv6
- IHL: number of 32-bit words in header
- Total length: byte length of datagram
- TTL: time to live
- Protocol: upper layer protocol
- Checksum: only protects IP header
- Fragmentation fields

IP Fragmentation, Reassembly

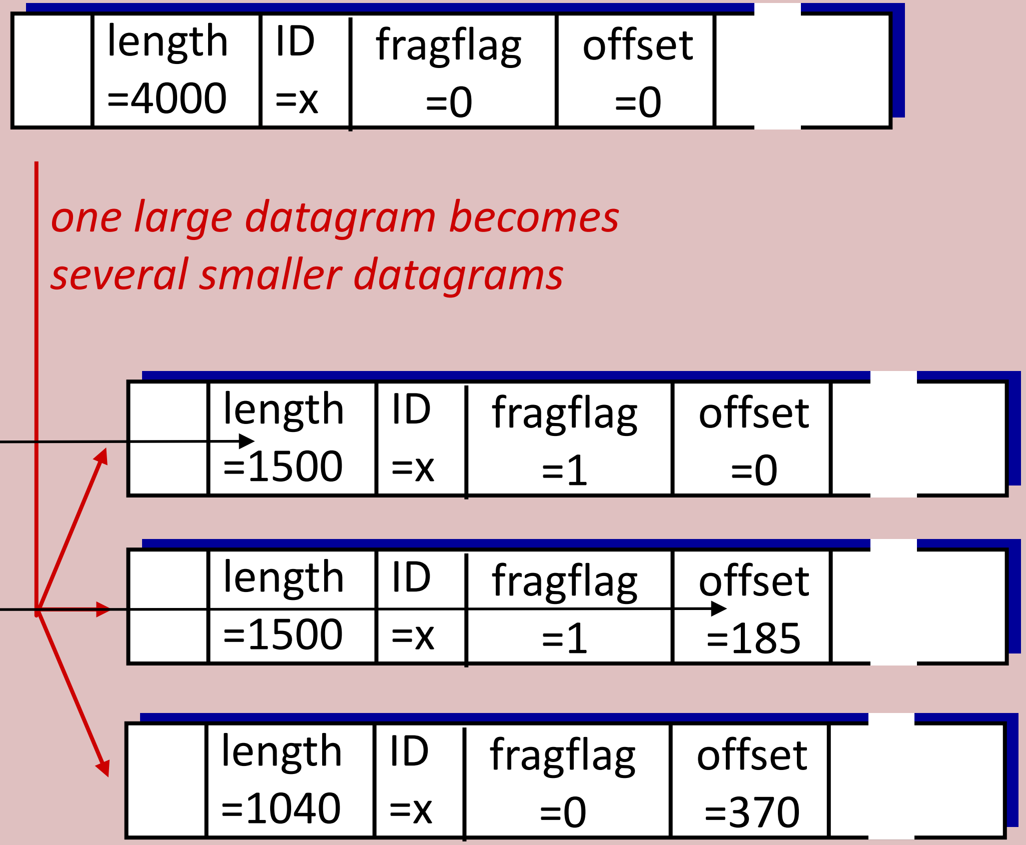
- Maximum transmission unit (MTU): largest link level frame for given link type
- Large IP packet might be fragmented
- Fragments reassembled at destination
- IP header values used to identify and order the fragments



IP Fragmentation, Reassembly

example:

- 4000 byte datagram
- MTU = 1500 bytes



1480 bytes in data field

offset = 1480/8

IPv4 Addressing

- Interface: connection between host/router and physical link
  - Routers typically have multiple
  - Hosts typically have one or two (wired and/or wireless)
- IP address: 32-bit network interface identifier

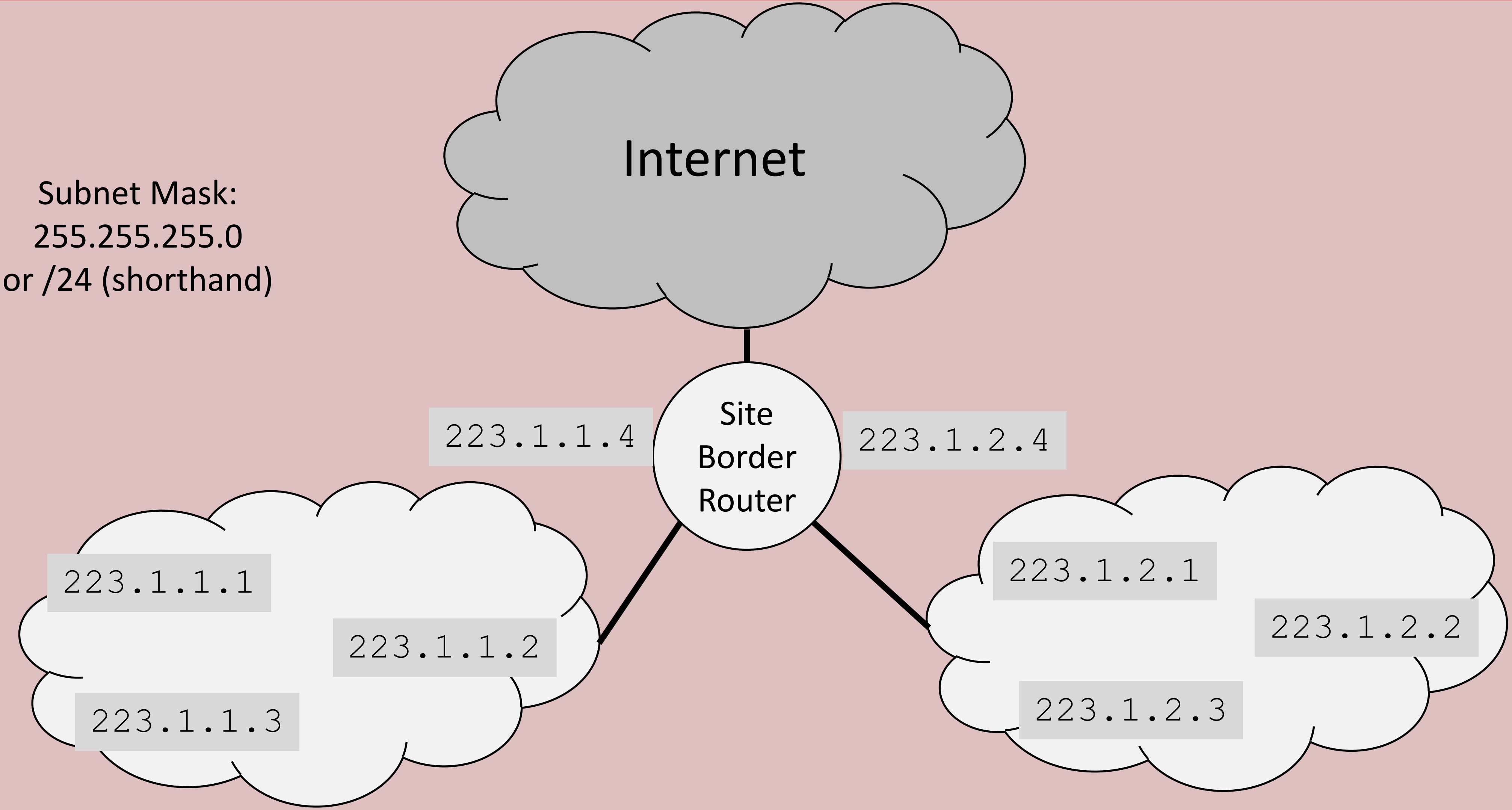
Subnet

- IP addresses consist of two parts
  - Subnet: high order bits
  - Host: low order bits

- Subnet (subnetwork)
  - Device interfaces with same subnet part of IP address
  - Can physically reach each other without an intervening router

Subnet

Subnet Mask:
255.255.255.0
or /24 (shorthand)

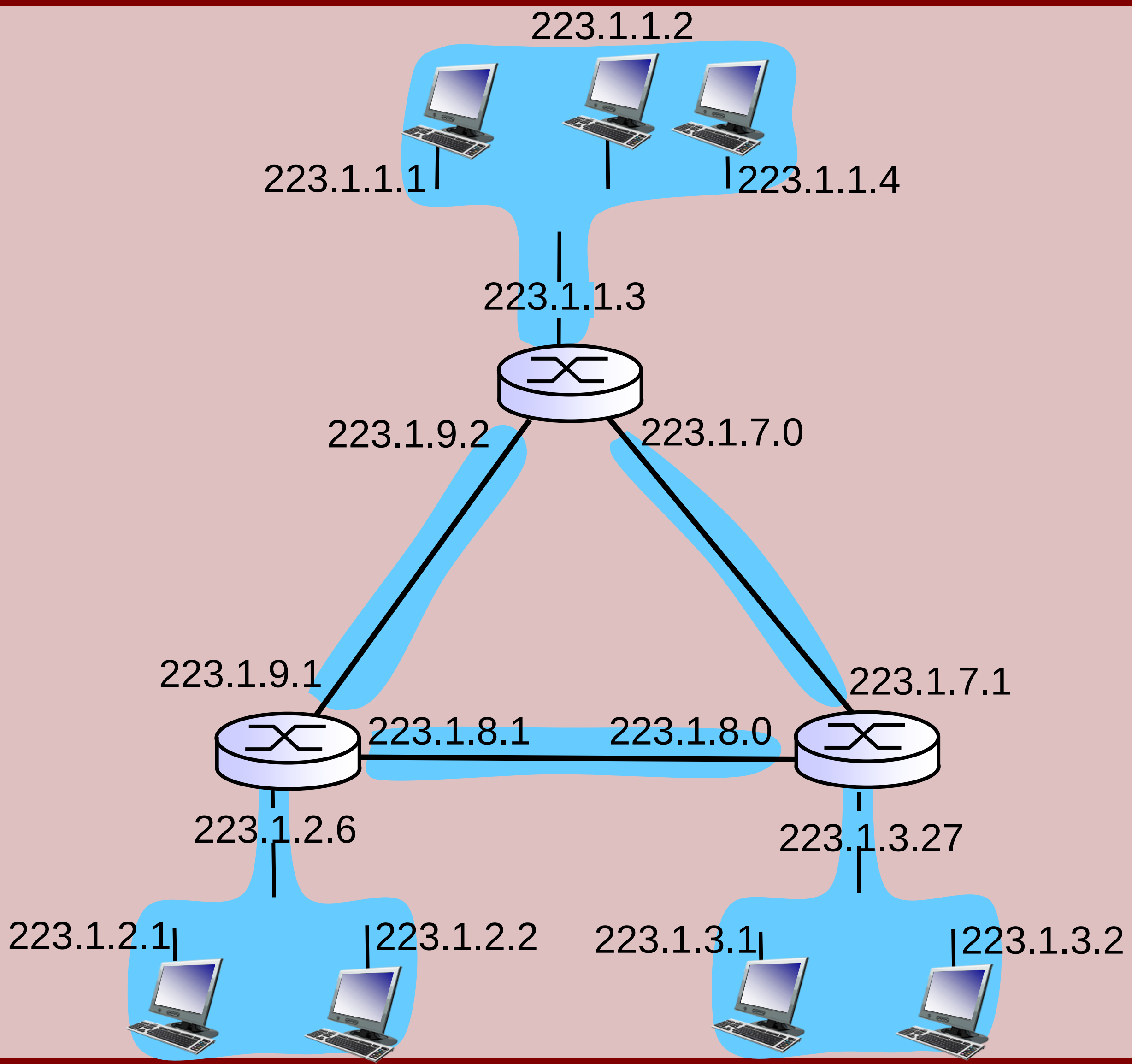


Subnet: 223.1.1.0/24

Subnet: 223.1.2.0/24

Subnets

- How many subnets?
- How many high order bits for subnet part?



Classful Addressing

Class	Subnet Bits	Host Bits
A	8	24
B	16	16
C	24	8
D	multicast	
E	reserved for future use	

Classless Inter-Domain Routing (CIDR)

- Classless is more flexible and efficient than classful (why?)
- Subnet portion of address of arbitrary length
- Address format: a.b.c.d/x where x is the number of bits in the subnet portion of the address

subnet part | host part

11001000 00010111 00010000 00000000

200.23.16.0/23

IP Address Assignment

- Manual configuration by sysadmin updating file
  - UNIX: /etc/rc.config
  - Windows: Control Panel > Network > Configuration > TCP/IP > Properties

- Automatic "plug-and-play" configuration via Dynamic Host Configuration Protocol (DHCP)

Dynamic Host Configuration Protocol (DHCP)

- Allow host to dynamically obtain IP address from server when it joins network
- Renew lease on address in use
- Allows reuse of addresses
- Supports mobile users

DHCP overview

- Host broadcasts DHCP discover message (optional)
- DHCP server responds with DHCP offer message (optional)
- Host requests IP address via DHCP request message
- DHCP server sends assigned IP address via DHCP ack message

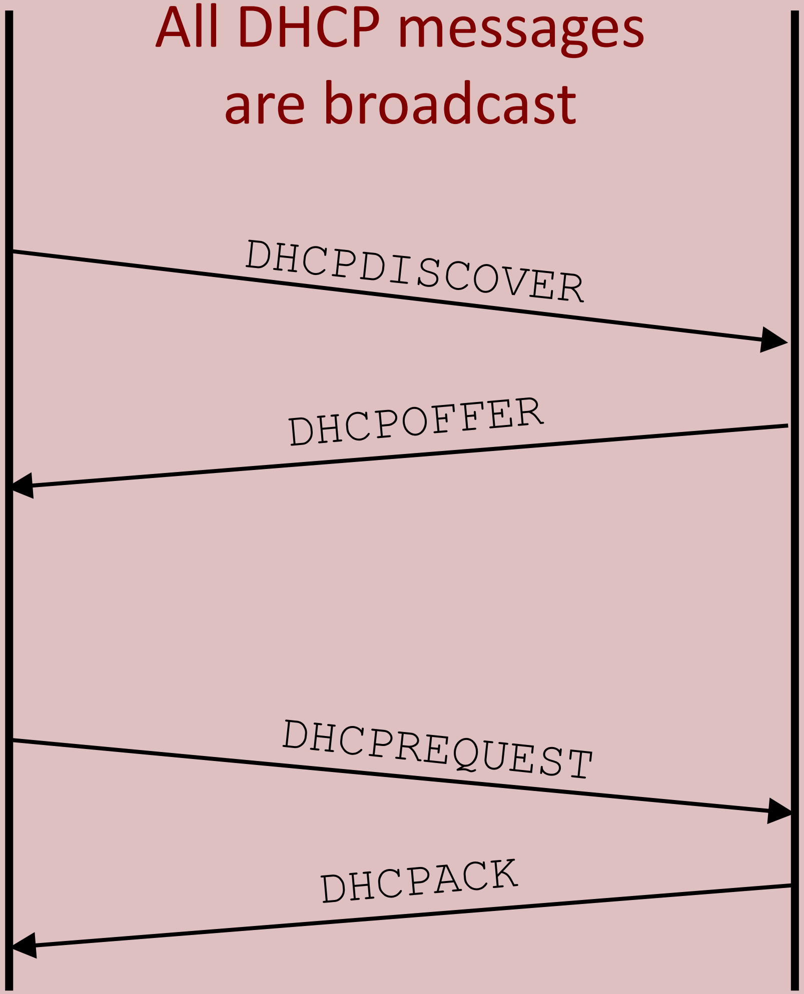
DHCP

DHCP
Client

All DHCP messages
are broadcast

DHCP
Server

I need to find a DHCP
server for an IP address



I am a DHCP server, here is
an IP address you can use

Okay, I'll take it

Confirmed, it's yours

DHCPDISCOVER

```
// no client IP yet, clients use port 68
src: 0.0.0.0, 68
// broadcast message, servers use port 67
dst: 255.255.255.255, 67
// “your” IP address, not yet filled in by server
yiaddr: 0.0.0.0
// 32-bit transaction ID for exchange
xid: 654
```

DHCPOFFER

// server IP and port

```
src: 223.1.2.4, 67
```

// broadcast message, clients use port 68

```
dst: 255.255.255.255, 68
```

// IP address offered by server

```
yiaddr: 223.1.2.5
```

// 32-bit transaction ID for request-response

```
xid: 654
```

// lease lifetime (in seconds)

```
lifetime: 3600
```

DHCPREQUEST

// still no client IP yet, clients use port 68

```
src: 0.0.0.0, 68
```

// broadcast message, servers use port 67

```
dst: 255.255.255.255, 67
```

// IP address offered by server

```
yiaddr: 223.1.2.5
```

// 32-bit transaction ID for request-response

```
xid: 655
```

// lease lifetime (in seconds)

```
lifetime: 3600
```

DHCPACK

// server IP and port

```
src: 223.1.2.4, 67
```

// broadcast message, clients use port 68

```
dst: 255.255.255.255, 68
```

// IP address offered by server

```
yiaddr: 223.1.2.5
```

// 32-bit transaction ID for request-response

```
xid: 655
```

// lease lifetime (in seconds)

```
lifetime: 3600
```

Additional Configuration via DHCP

- DHCP can return more than just allocated IP address on subnet
- Address of first-hop router for client
- Name and IP address of DNS server
- Network mask (specify network vs. host portion of IP address)

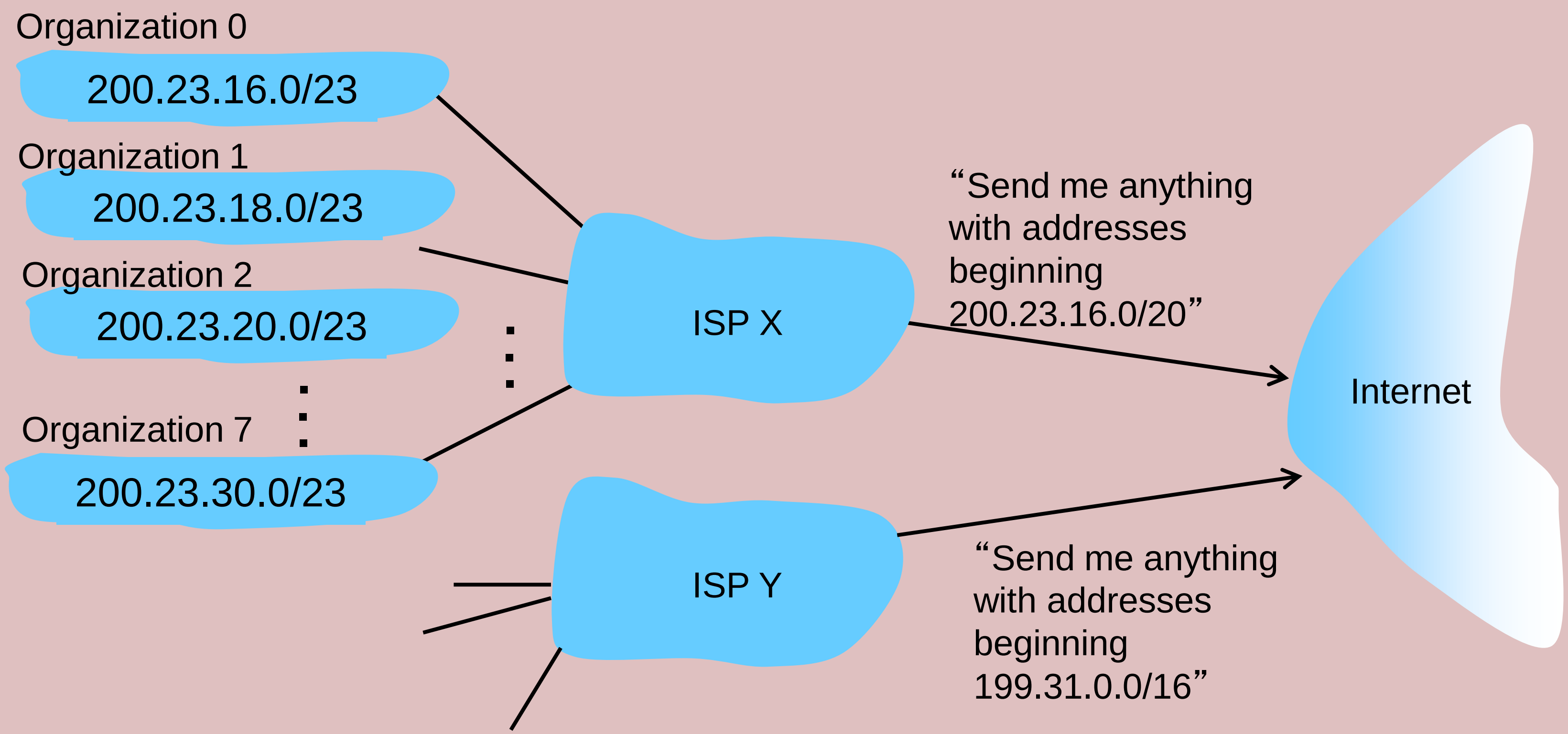
Obtaining IP Address Blocks

- ISP allocations a portion of its own IP address space to customer organizations

ISP's block	<u>11001000 00010111 0001</u>0000	00000000	200.23.16.0/20		
Organization 0	<u>11001000 00010111 0001000</u>0	00000000	200.23.16.0/23		
Organization 1	<u>11001000 00010111 0001001</u>0	00000000	200.23.18.0/23		
Organization 2	<u>11001000 00010111 0001010</u>0	00000000	200.23.20.0/23		
...	…..	….	….		
Organization 7	<u>11001000 00010111 0001111</u>0	00000000	200.23.30.0/23		

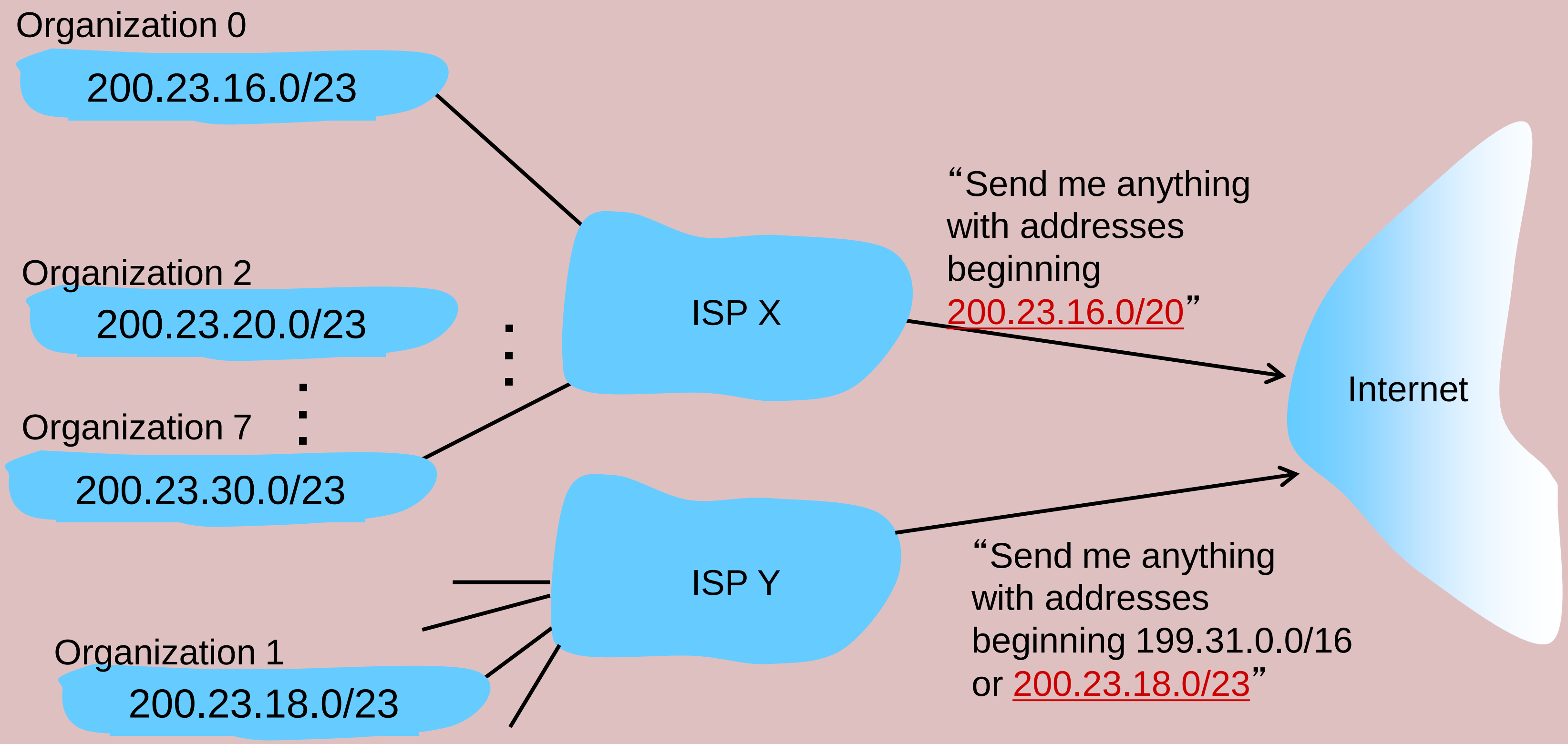
Route Aggregation

Hierarchical addressing allows efficient advertisement of routing information:



More Specific Routes

ISP Y has a more specific route to Organization 1

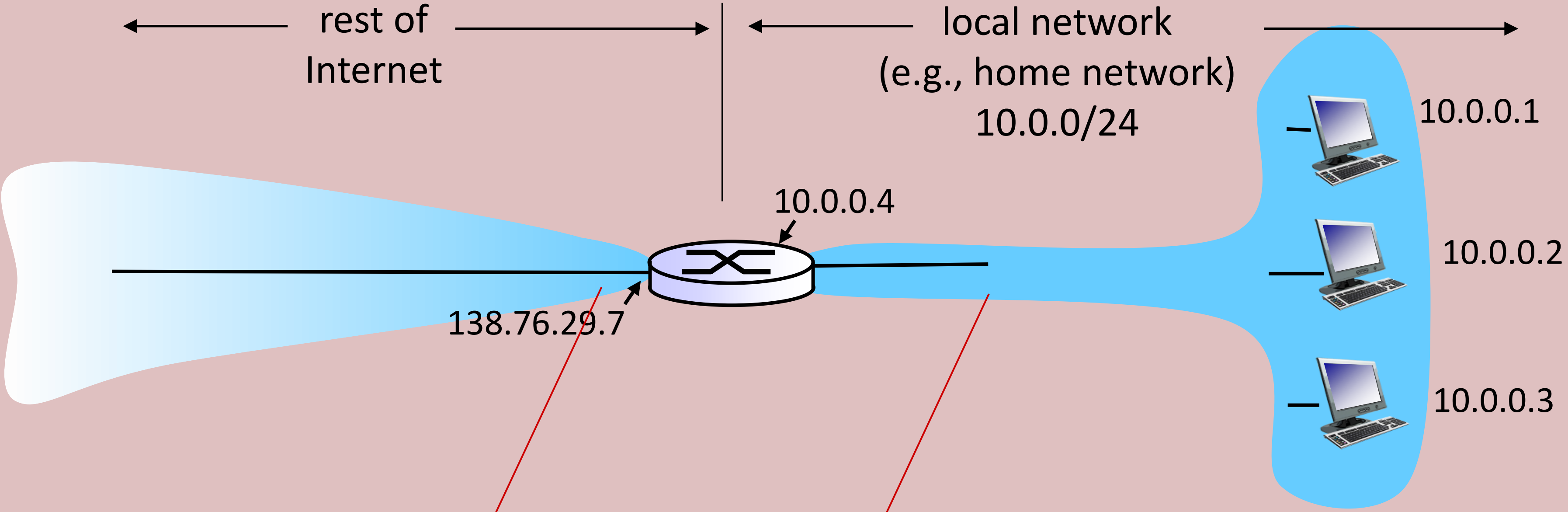


Why doesn't range overlap cause conflict?

ICANN

- Internet Corporation for Assigned Names and Numbers (https://www.icann.org)
- Allocates blocks of IP addresses to ISPs
- Manages DNS
- Assigns domain names and resolves disputes

Network Address Translation (NAT)



all datagrams *leaving* local network have *same* single source NAT IP address: 138.76.29.7,different source port numbers

datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

NAT: Motivation

- Only need a single IP address from ISP
- Can change IP addresses of local network devices without notifying outside world
- Can change ISPs without changing IP addresses of local network devices
- Security bonus: local network devices not explicitly addressable by outside world

NAT: Implementation

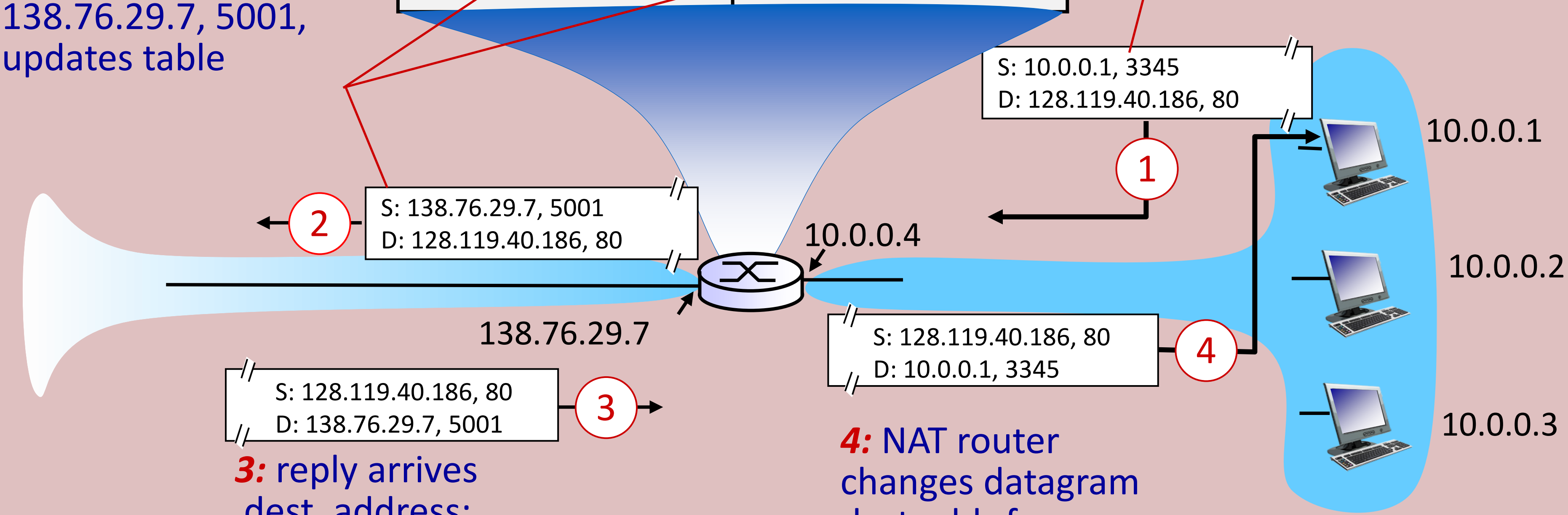
- Translation table: map public IP address/port to private address/port
- Outgoing packets: translate private address/port to public address/port
- Incoming packets: translate public address/port to private address/port

NAT

NAT Translation Table	
WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
⋮	⋮

2: NAT router changes datagram source addr from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80



3: reply arrives dest. address: 138.76.29.7, 5001

4: NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345

NAT

- 16-bit port number field
  - How many devices can be on a private network?
- Used to be somewhat controversial
  - Routers should strictly be layer 3
  - IPv6 should solve address shortage
  - Violates end-to-end argument
    - App designers must account for NAT
    - E.g., P2P applications like Skype
- NAT traversal: clients connecting to servers behind NATs

IPv6

- Replaces 32-bit IP addresses with 128-bit IP addresses (why?)
- New header format to speed up packet processing/forwarding
- Header changes to facilitate QoS
- Fixed-length 40-byte header
- No fragmentation

IPv6 Datagram Header

<table>
<tr><td>version
(4 bits)</td><td>diffserv
(4 bits)</td><td>ECN
(2)</td><td colspan="3">flow label (20 bits)</td></tr>
<tr><td colspan="4">payload length (16 bits)</td><td>next header
(8 bits)</td><td>hop limit
(8 bits)</td></tr>
<tr><td colspan="6">source IP address (128 bits)</td></tr>
<tr><td colspan="6">destination IP address (128 bits)</td></tr>
</table>

IPv6 Packet Header Fields

- Flow label: identify packets in same flow
- Next header: identify upper layer protocol for payload data
- Hop limit: similar to TTL

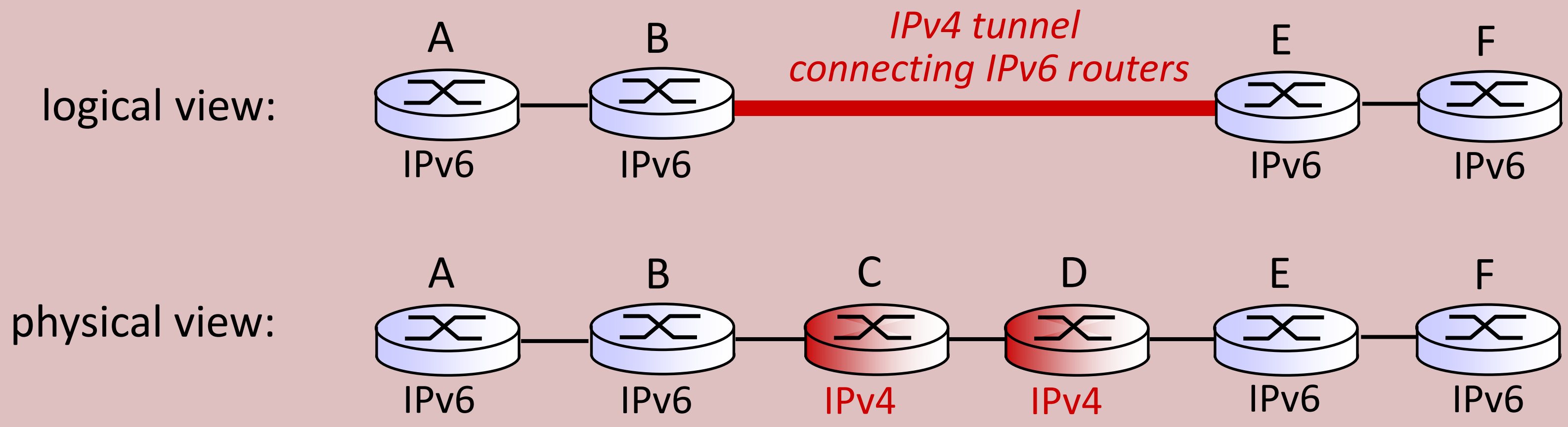
Other changes from IPv4

- Checksum: removed entirely to reduce processing time at each hop
- Options: allowed, but outside of header, indicated by “Next Header” field
- ICMPv6: new version with multicast group management

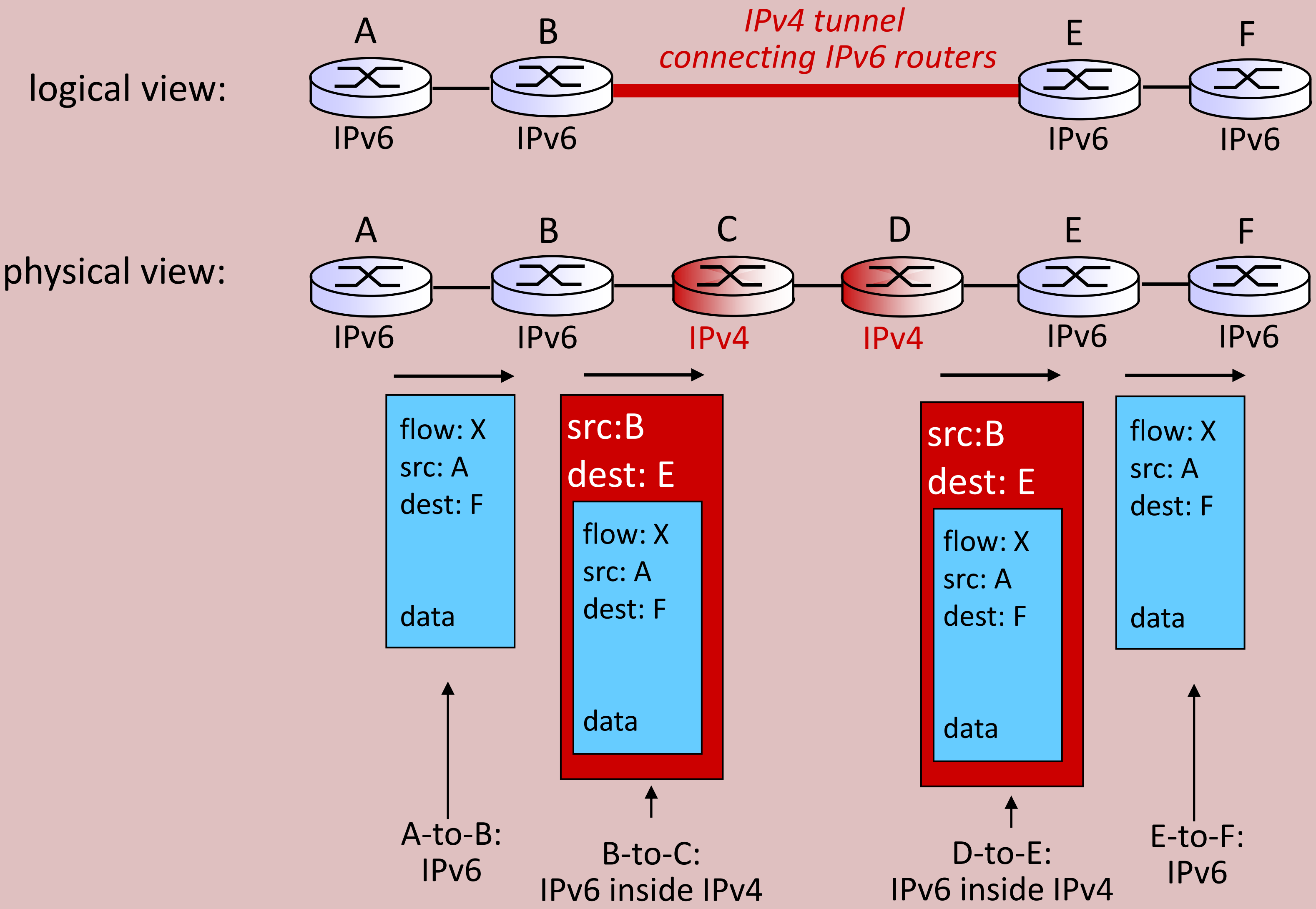
IP Tunneling

- Routers cannot all be upgraded simultaneously (i.e., no “flag days”)
- Network must operate with both IPv4 and IPv6 routers (dual stack)
- Tunneling: carry IPv6 packet as payload in IPv4 among IPv4 routers

IP Tunneling



IP Tunneling



IPv6 Adoption

- Google: 12.9% of clients access services via IPv6
- NIST: One-third of all US government domains are IPv6 capable
- Deployment is 20 years and counting (why?)

Thank You!

Networks

Generalized Forwarding and SDN

Adopted from material in “Computer Networking: A Top Down Approach” by Kurose and Ross and slides developed by William Conner

Generalized Forwarding and SDN

Section 4.4

Generalized Forwarding

- Thus far, routing tables have only considered destination-based forwarding
- Flow tables consider more than just destination IP addresses
- Logically centralized controller computes and distributes flow tables (software defined networking)

Generalized Forwarding

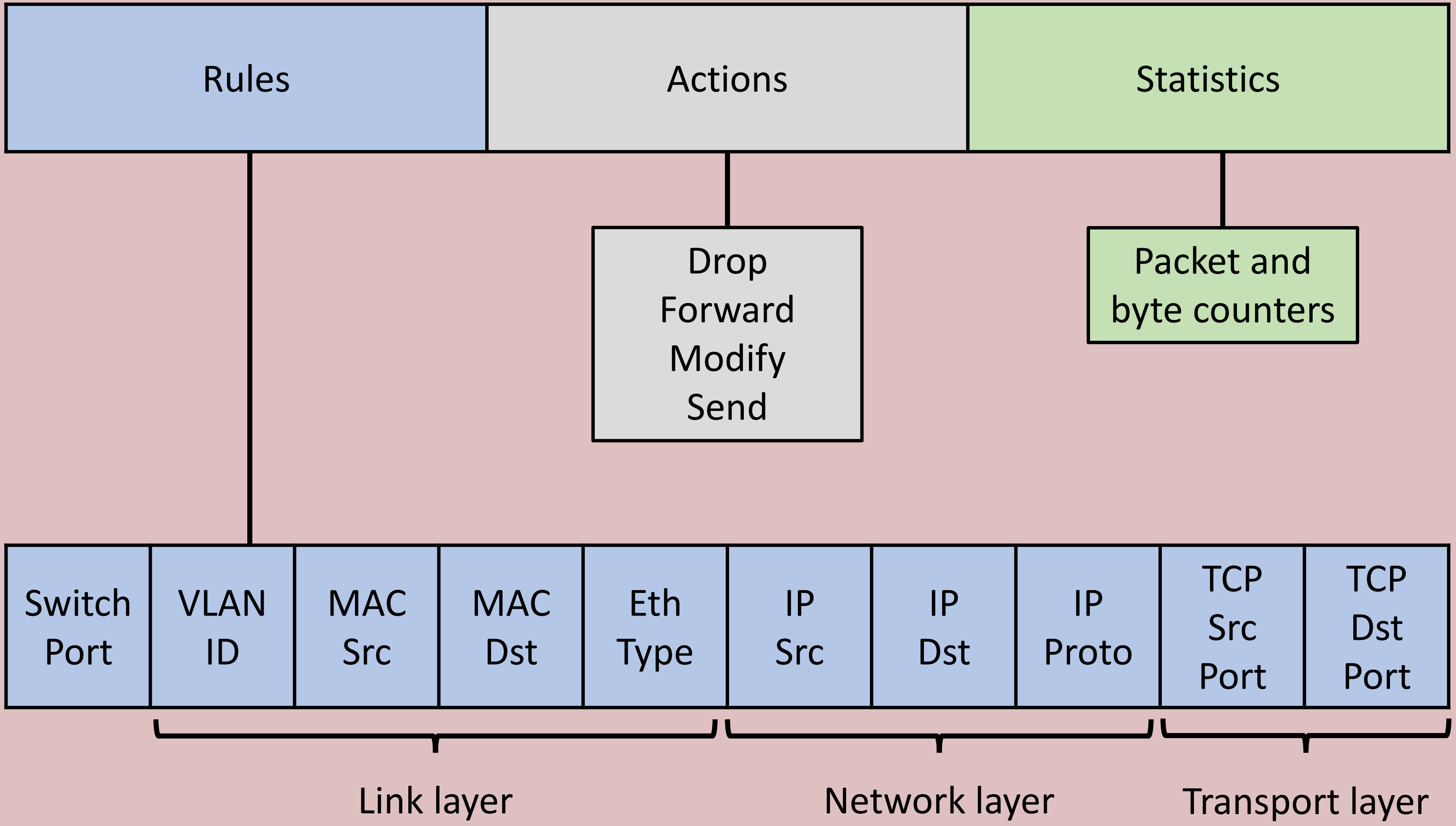
- More than just forwarding a packet to a destination (router)

- The following actions are also supported
  - Drop packet (firewall)
  - Modify packet (NAT)
  - Send packet to SDN controller

OpenFlow Data Plane Abstration

- Flow: defined by header fields
- Flow table: defines match + action rules for flows
- Generalized forwarding: flow-based packet-handling rules
  - Match flow patterns
  - Perform action for matched packets

OpenFlow Table Format



OpenFlow Table Examples

Destination-Based Forwarding: datagrams headed towards 128.10.11.12 should be forwarded on output port 6

IP Src	IP Dst	Action
*	128.10.11.12	forward(6)

OpenFlow Table Examples

Firewall: block all datagrams headed towards TCP port 22 (SSH traffic)

IP Src	IP Dst	TCP Src Port	TCP Dst Port	Action
*	*	*	22	drop

OpenFlow Table Examples

Firewall: block all datagrams sent by 128.10.11.12

IP Src	IP Dst	Action
128.10.11.12	*	drop

OpenFlow Table Examples

Layer-2 Switch: forward all frames from AA:BB:CC:DD:EE:FF on output port 3

MAC Src	MAC Dst	Action
AA:BB:CC:DD:EE:FF	*	forward(3)

OpenFlow Abstraction

- match+action: paradigm unifies different kinds of devices into one rule set

- Router
  - match: longest destination IP prefix
  - action: forward out on link

- Switch
  - match: destination MAC address
  - action: forward or flood

OpenFlow Abstraction

- match+action: paradigm unifies different kinds of devices into one rule set

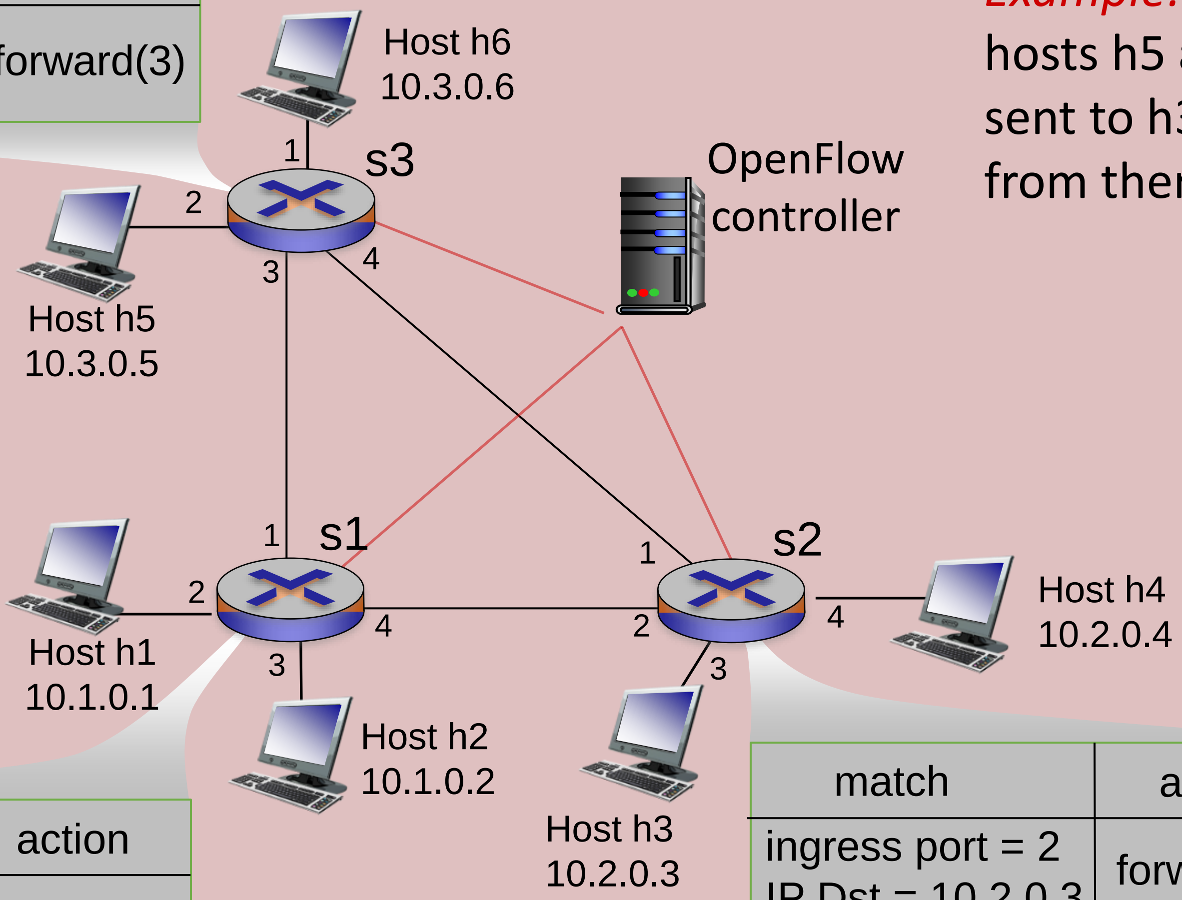
- Firewall
  - match: IP address and port
  - action: permit or deny

- NAT
  - match: IP address and port
  - action: rewrite address and port

OpenFlow Example

Example: datagrams from hosts h5 and h6 should be sent to h3 or h4, via s1 and from there to s2

match	action
IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(3)



match	action
ingress port = 1 IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(4)

match	action
ingress port = 2 IP Dst = 10.2.0.3	forward(3)
ingress port = 2 IP Dst = 10.2.0.4	forward(4)

Thank You!