CSCI 466: Networks

Network Layer – Data Plane

Reese Pearsall

Fall 2022

Announcements

NO CLASS this Friday (10/14) and next Monday (10/17)

- You can always email/DM me if you have questions

Wireshark Lab 1 due TONIGHT @ 11:59 PM

- If you work with a partner, both need to submit to D2L!

Extra Credit survey still available !!

OSI Model

Application Layer

Presentation Layer *

Session Layer *

Transport Layer

Network Layer

Data Link Layer

Physical Layer

Application Layer

Messages from Network Applications

Physical Layer

Bits being transmitted over a medium

In the textbook, they condense it to a 5-layer model, but 7 layers is what is most used

OSI Model

Application Layer

Presentation Layer *

Session Layer *

Transport Layer

Network Layer

Data Link Layer

Physical Layer

Application Layer

Messages from Network Applications

↓

Physical Layer

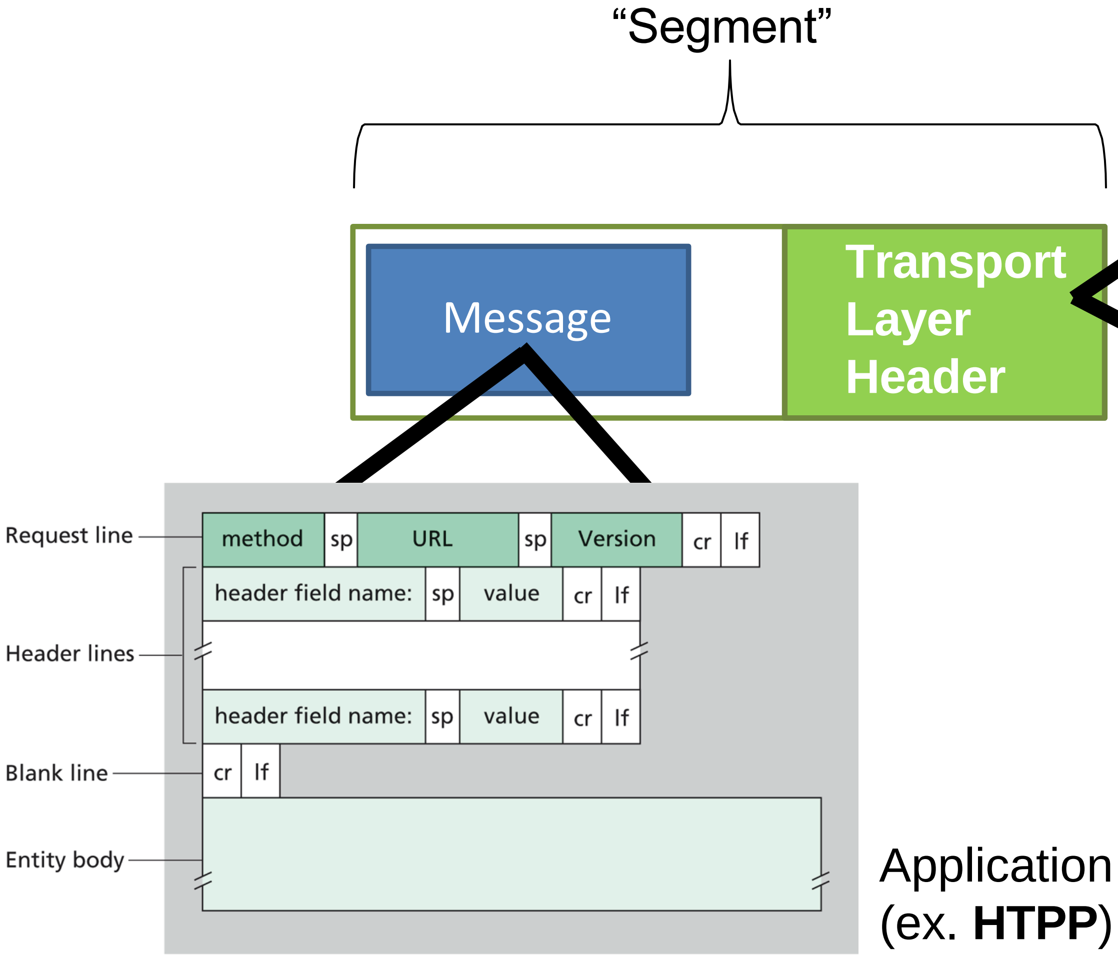
Bits being transmitted over a medium

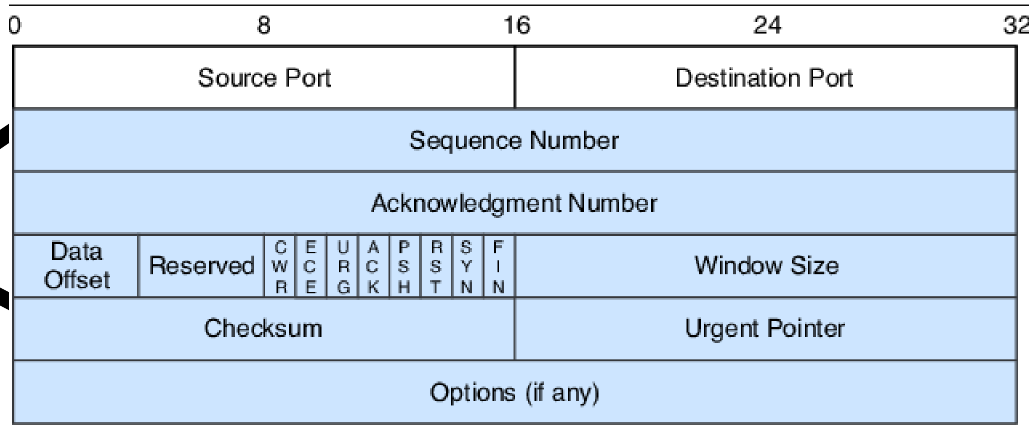
**In the textbook, they condense it to a 5-layer model, but 7 layers is what is most used*

Application Layer

Transport Layer

Our packet of information so far…





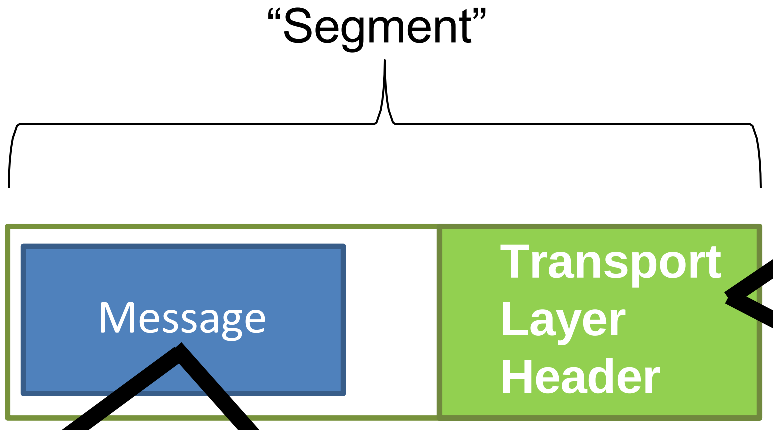
Transport Layer Protocol
(ex. **TCP**)

Application Layer Protocol
(ex. **HTPP**)

Application Layer

Transport Layer

Our packet of information so far…



Source Port	Destination Port
Length	Checksum
Data	

UDP Header (Source Port, Destination Port, Length, Checksum)

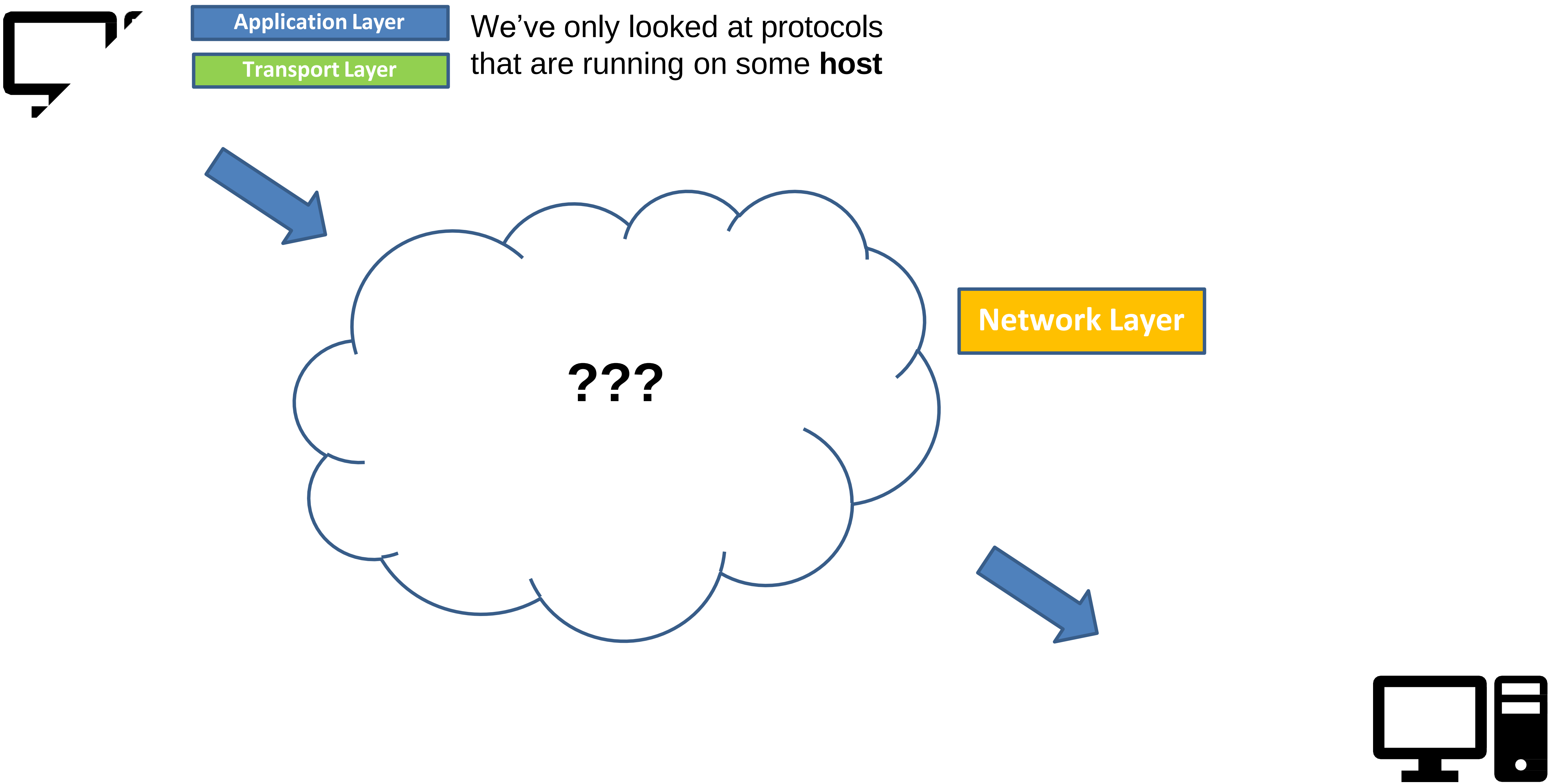
Transport Layer Protocol (ex. **UDP**)

Header Format

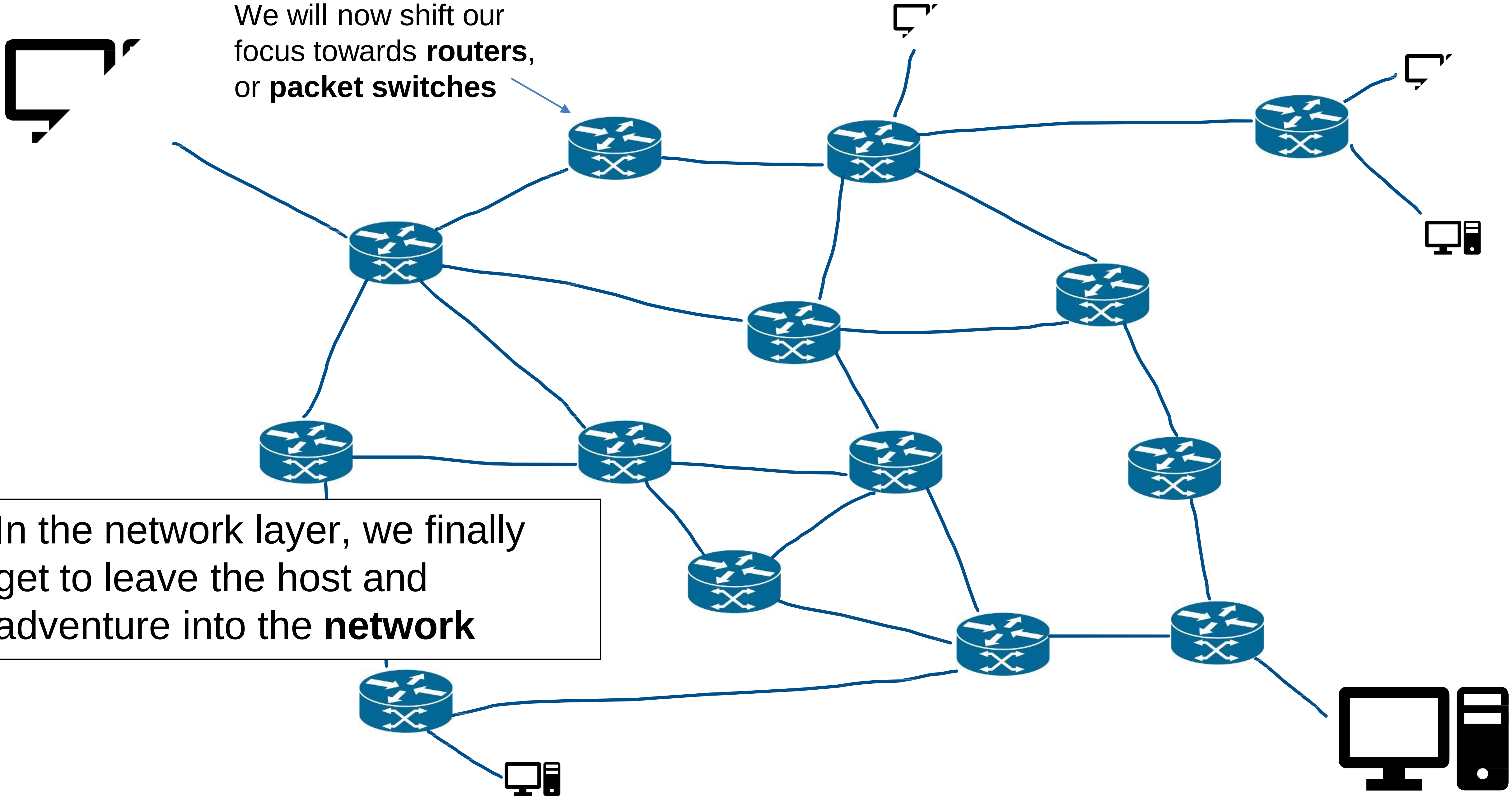
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ID															
QR	Opcode				AA	TC	RD	RA	Z			RCODE			
QDCOUNT															
ANCOUNT															
NSCOUNT															
ARCOUNT															

Application Layer Protocol (ex. **DNS**)

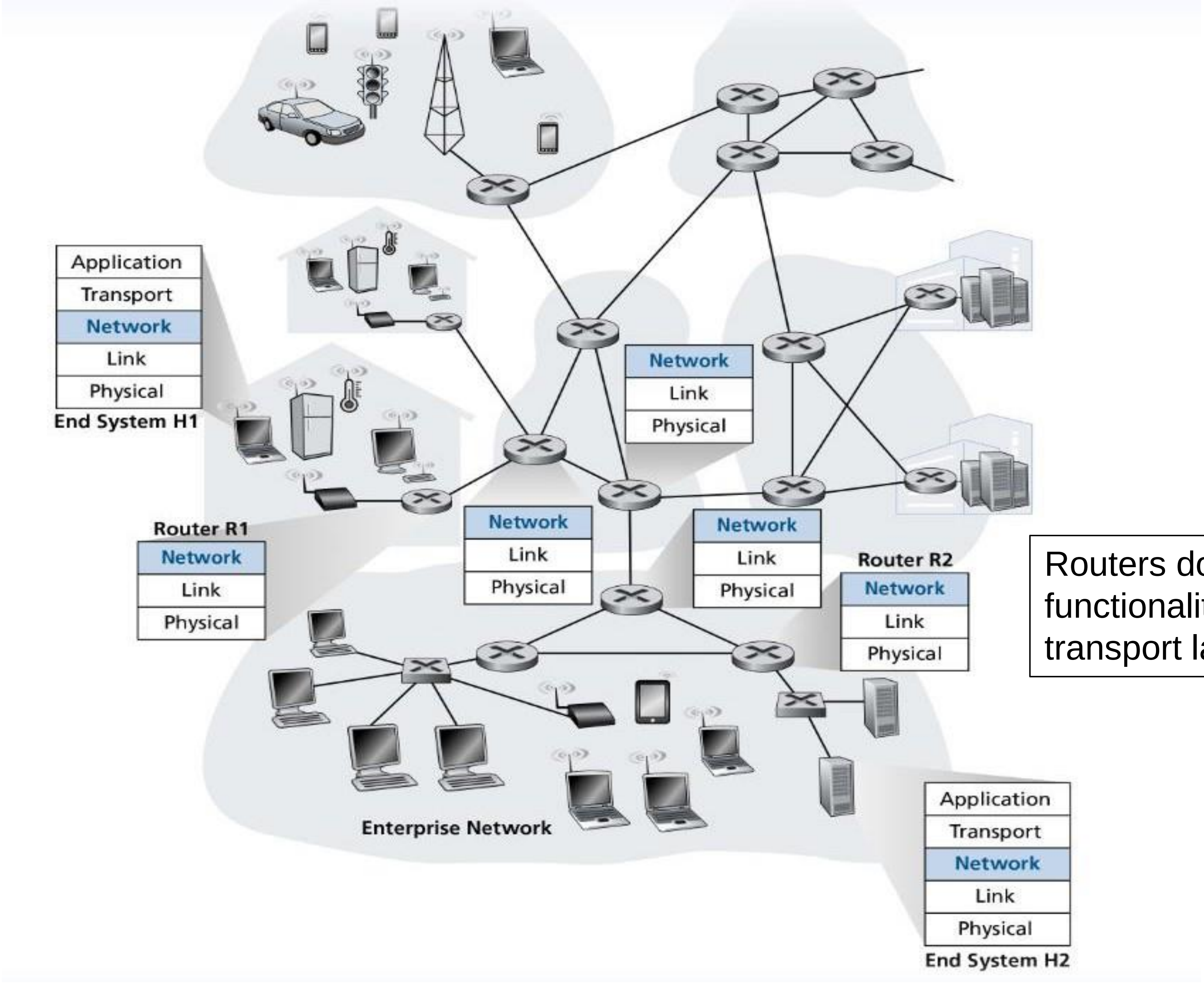
We’ve only looked at protocols that are running on some **host**



We will now shift our focus towards **routers**, or **packet switches**

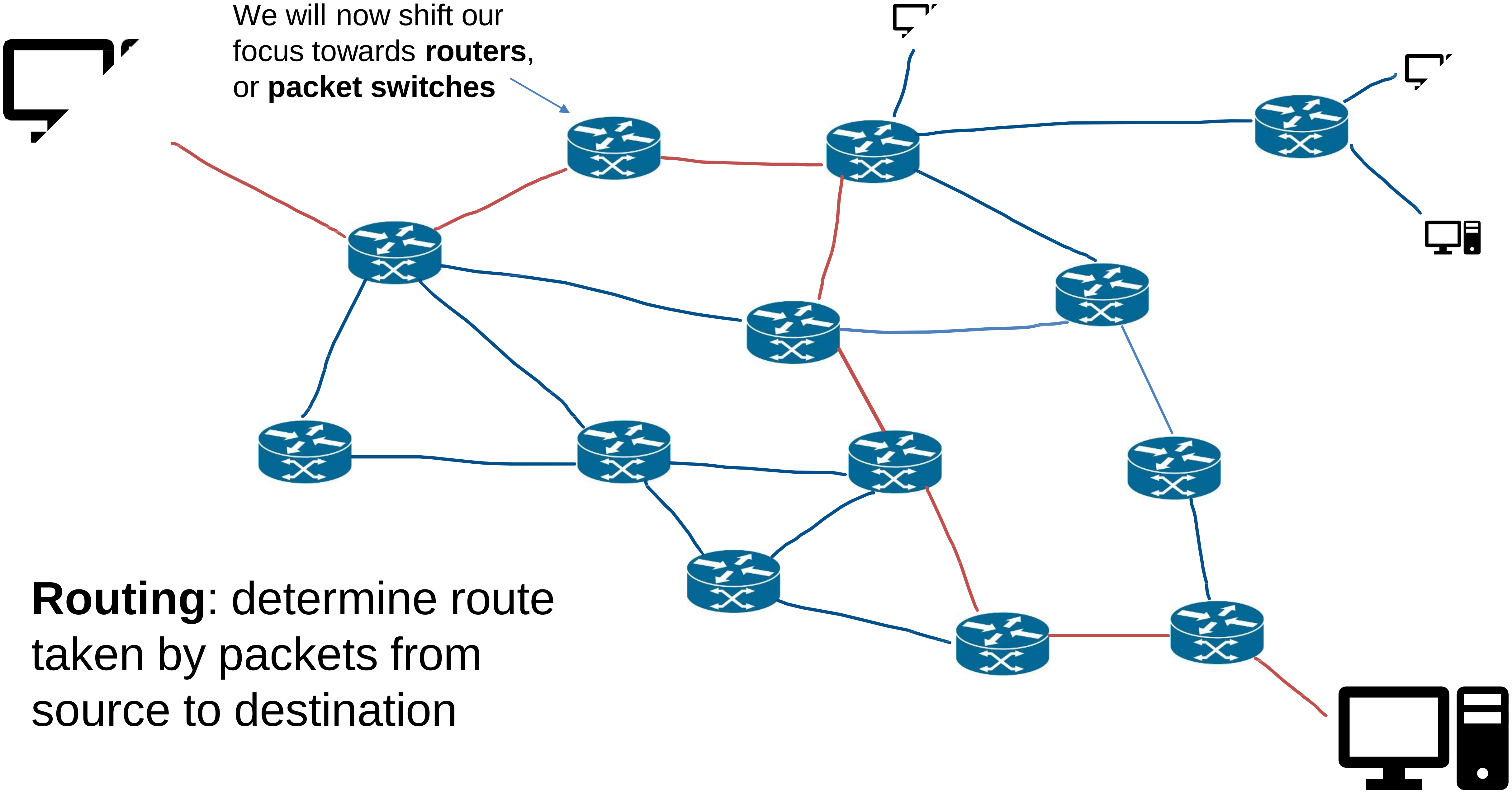


In the network layer, we finally get to leave the host and adventure into the **network**



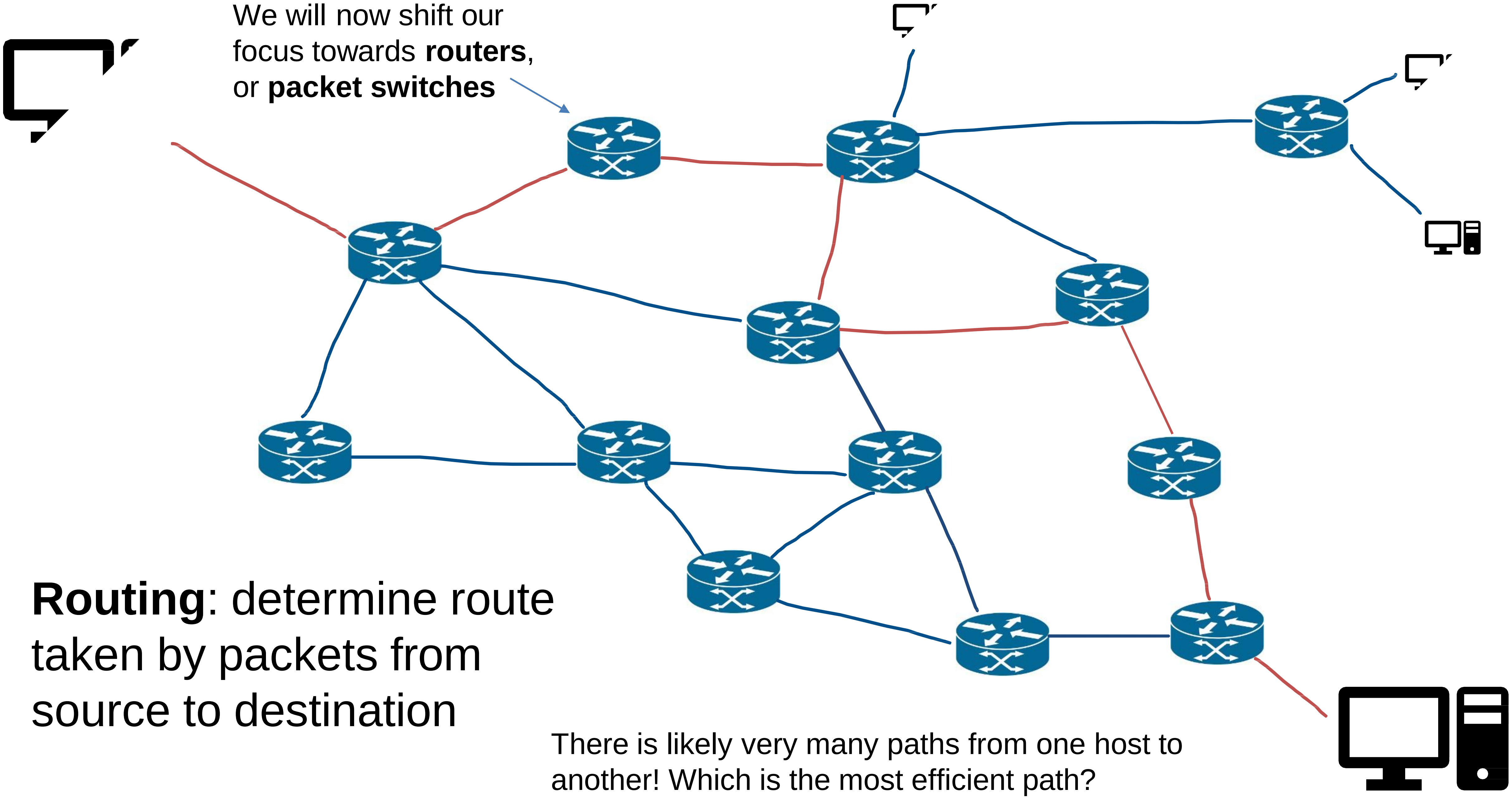
Routers do not have functionality for application & transport layer protocols

We will now shift our focus towards **routers**, or **packet switches**



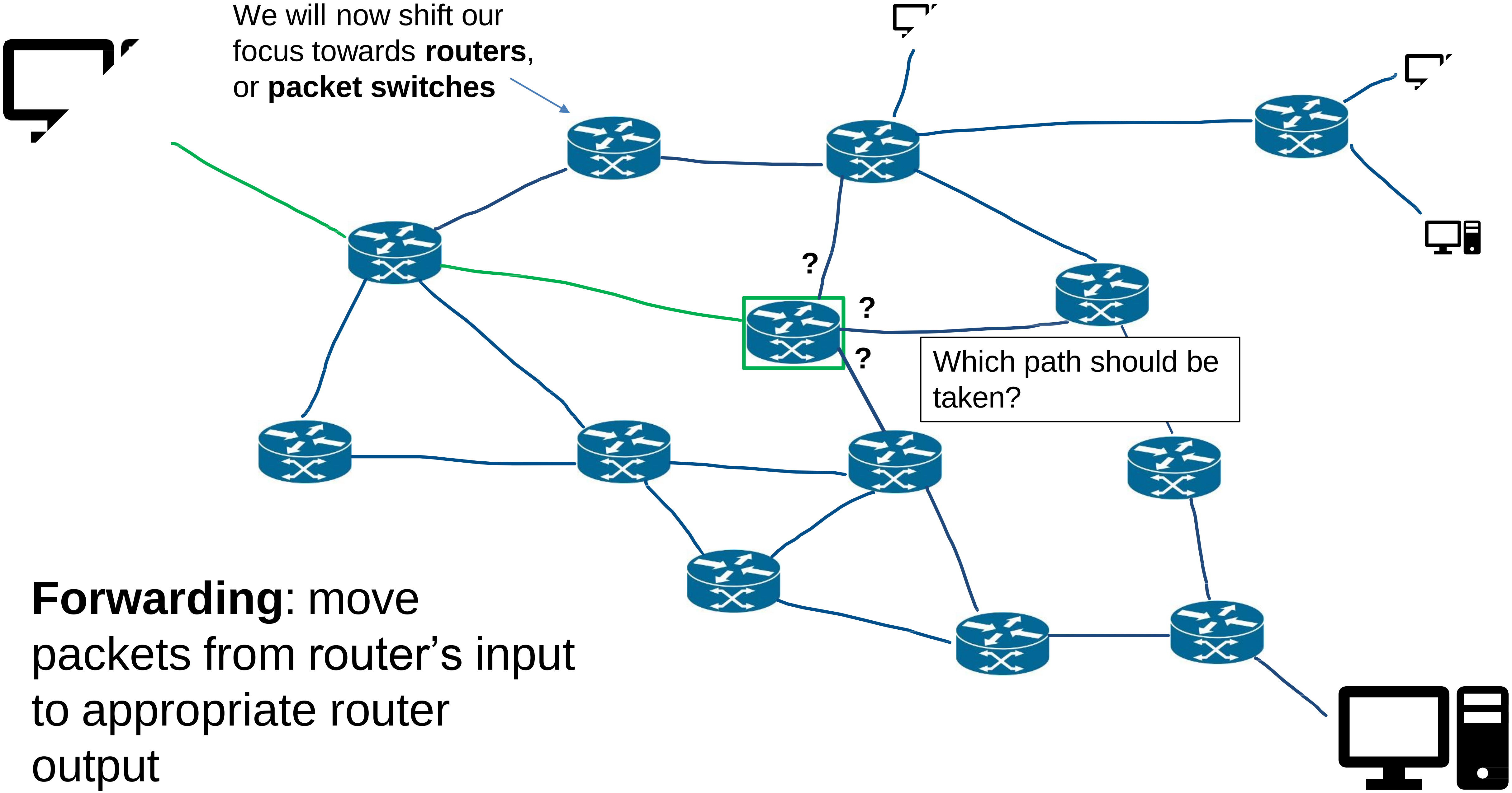
Routing: determine route taken by packets from source to destination

We will now shift our focus towards **routers**, or **packet switches**



Routing: determine route taken by packets from source to destination

There is likely very many paths from one host to another! Which is the most efficient path?



Forwarding: move packets from router’s input to appropriate router output

Network Layer

Responsible for the delivery of data through a network

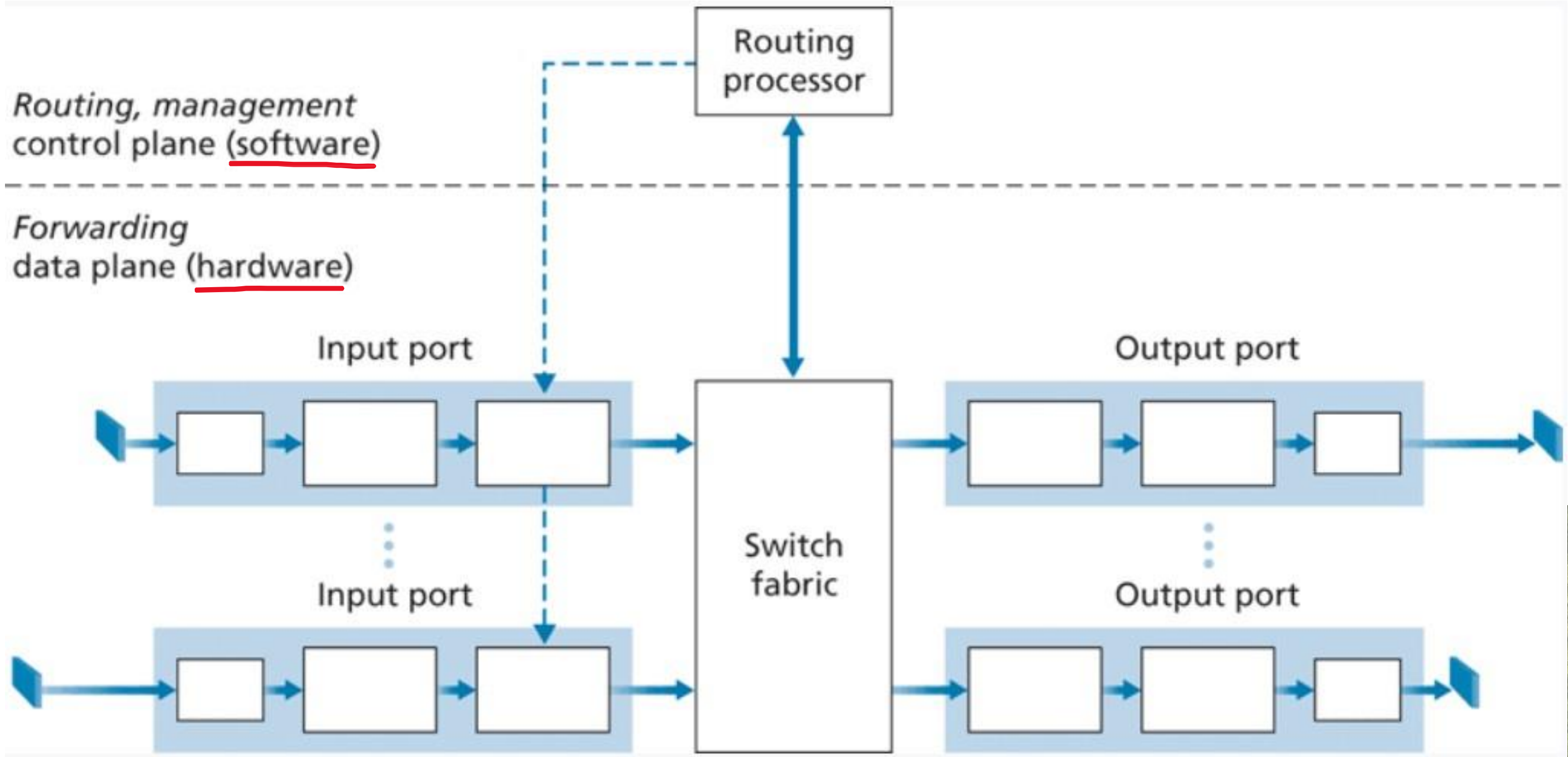
Forwarding

Data Plane

Routing

Control Plane

Router Architecture Overview



1. Destination-based forwarding

- Forwarding decisions are based on the **destination** of the packet

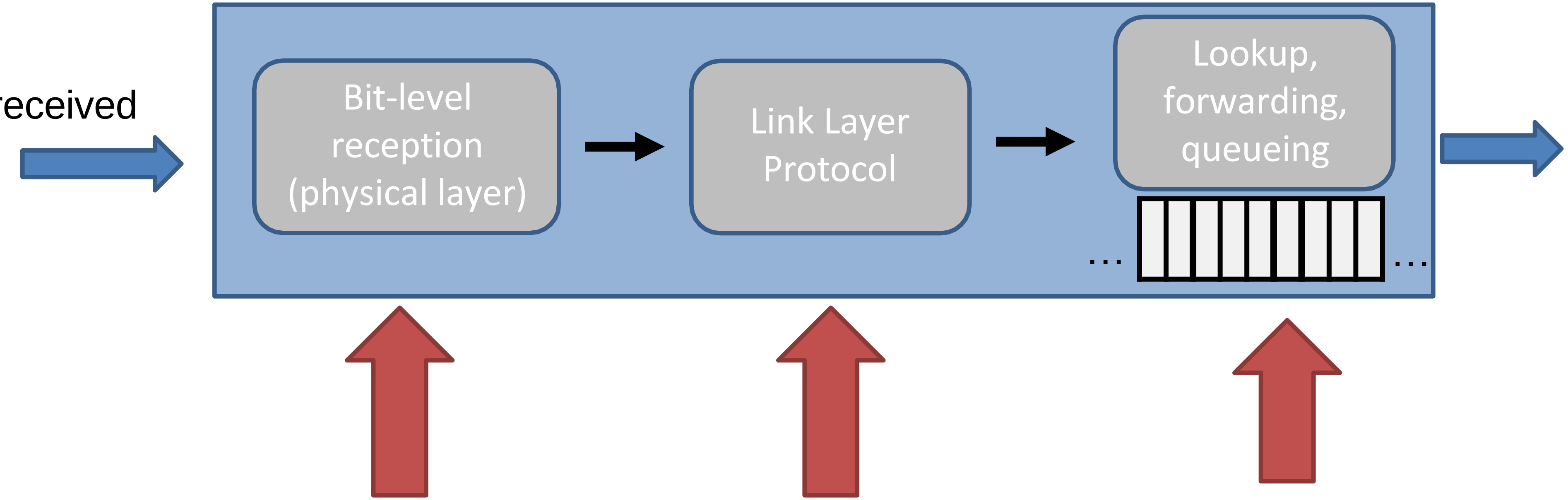
2. Generalized forwarding

- Forwarding decisions based on any set of header field values

Router Architecture Overview

INPUT PORT

Bits received



"Switching Fabric"

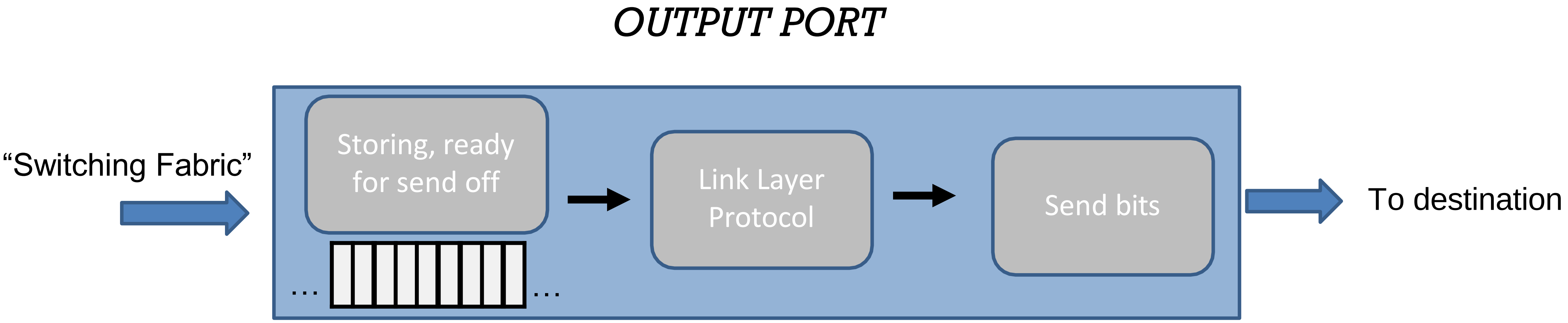
Such as over the wire or through the air

Next layer of OSI model
Ex. Ethernet

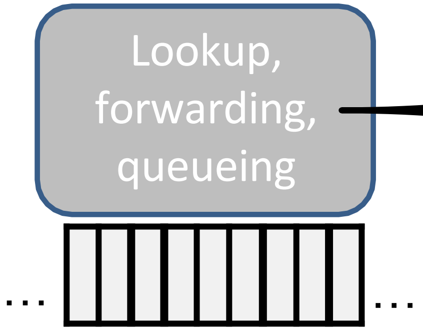
Evaluate header field values, lookup output port using forwarding table in memory

Queueing will occur if datagrams arrive faster than forwarding rate into the switching fabric

Router Architecture Overview



Forwarding Techniques



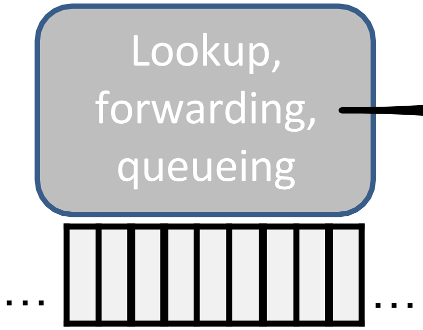
- Connection-less
- does not require startup,
- Has no idea where the final destination is

Routing Table

Address range	**Interface** (output link)
128.11.52.0 – 128.11.52.255	1
153.90.2.0 – 153.90.2.255	2
153.90.2.87 – 153.90.2.89	3

This routing table could get very big…
IP addresses need 32/64 bits of memory each

Forwarding Techniques



- Connection-less
- does not require startup,
- Has no idea where the final destination is

Routing Table

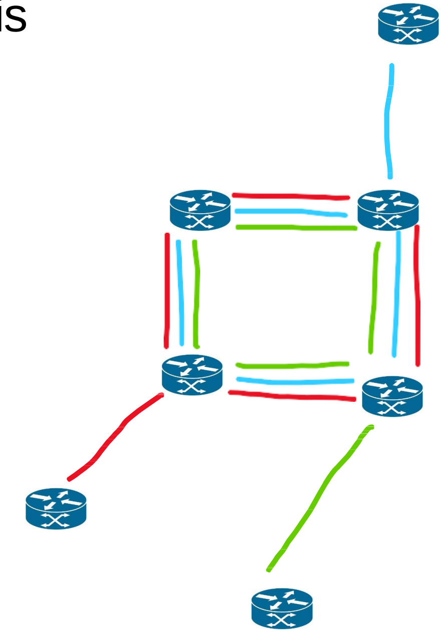
Address range	Interface (output link)
128.11.52.0 – 128.11.52.255	1
153.90.2.0 – 153.90.2.255	2
153.90.2.87 – 153.90.2.89	3

- Connection-oriented
- Requires startup

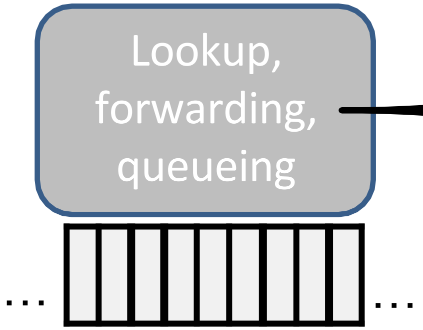
Virtual Circuit Networks

VC identifier	Interface (output link)
22	1
13	2
5	3

Costly ☹



Forwarding Techniques



Longest prefix matching

Address range	Interface (output link)
11001000 00010111 00010*** *********	1
11001000 00010111 00011000 *********	2
11001000 00010111 00011*** *********	3
otherwise	4

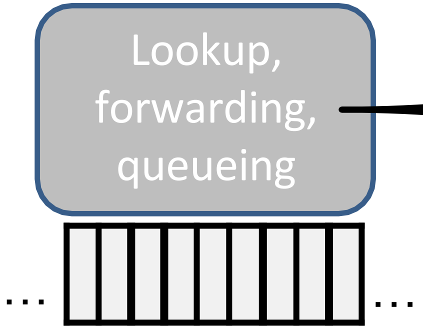
when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address

examples:

DA: 11001000 00010111 000**10110 10100001** which interface?

DA: 11001000 00010111 000**11000 10101010** which interface?

Forwarding Techniques



Longest prefix matching

Address range	Interface (output link)
11001000 00010111 00010*** *********	1
11001000 00010111 00011000 *********	2
11001000 00010111 00011*** ********	3
otherwise	4

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

These lookups need to happen in nanoseconds for our network to function

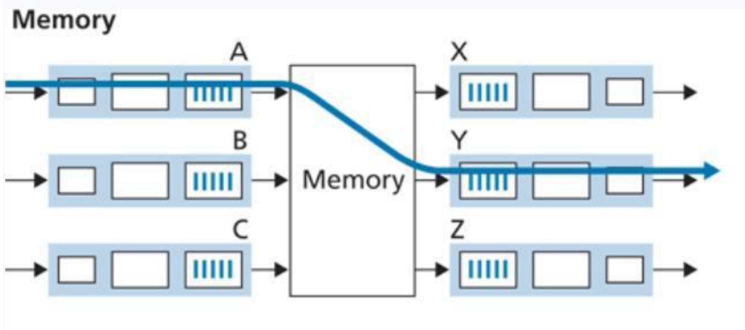
Ternary content addressable memories (TCAMs) are used in modern routers to do **LPM**

Cisco routers can carry millions of TCAM entries in their routers

Fabric Switch and Switching

Switching fabric: Mechanism that forwards data from an input port to output port

Switching via memory:



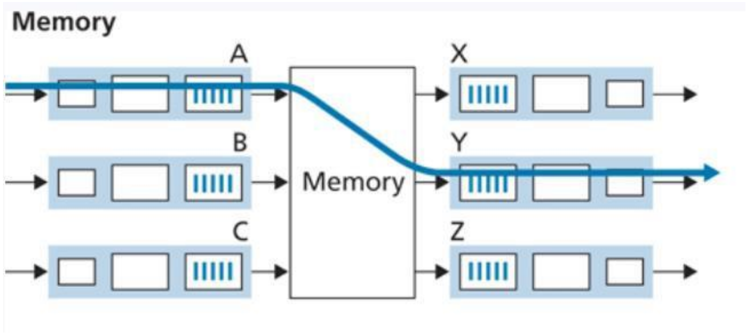
Handled by a CPU (routing processor)

Cannot forward data in parallel

Fabric Switch and Switching

Switching fabric: Mechanism that forwards data from an input port to output port

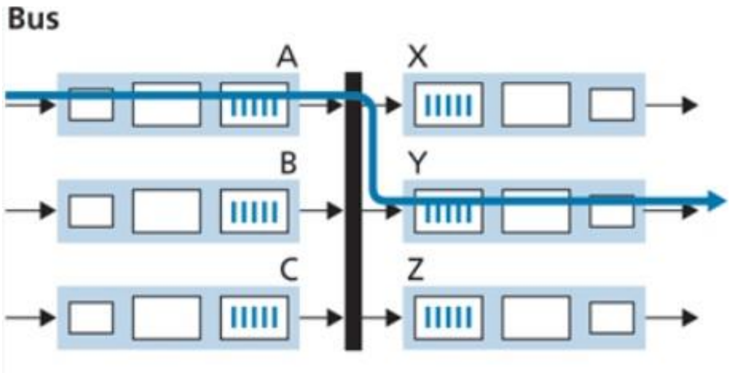
Switching via memory:



Handled by a CPU (routing processor)

Cannot forward data in parallel

Switching via bus:



Datagrams are prepended with a header

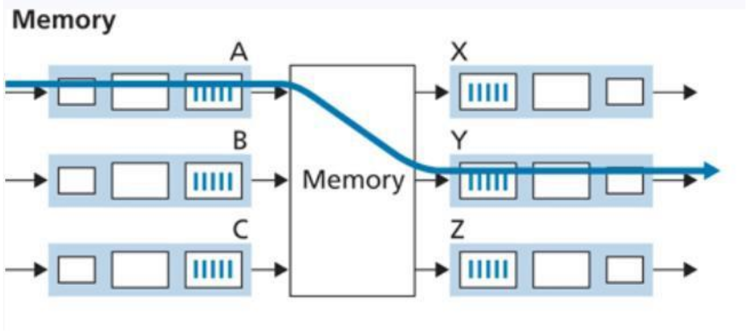
A “bus” transports input port datagrams to output ports

“keep going around the roundabout until you find your port”

Fabric Switch and Switching

Switching fabric: Mechanism that forwards data from an input port to output port

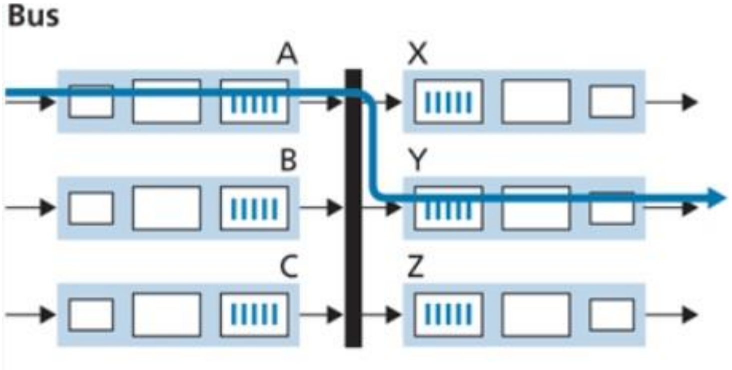
Switching via memory:



Handled by a CPU (routing processor)

Cannot forward data in parallel

Switching via bus:

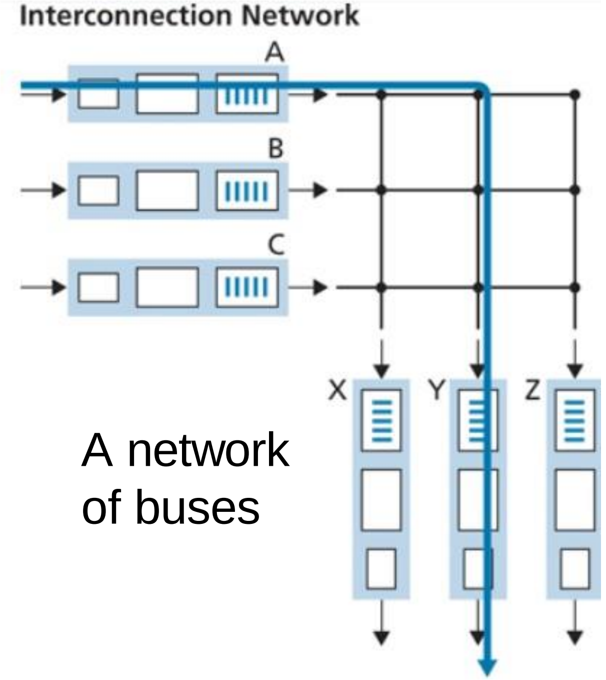


Datagrams are prepended with a header

A “bus” transports input port datagrams to output ports

“keep going around the roundabout until you find your port”

Switching via network:

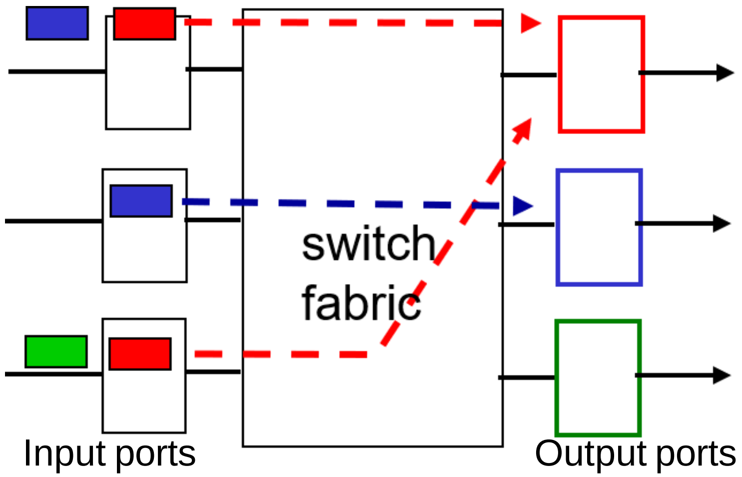


A network of buses

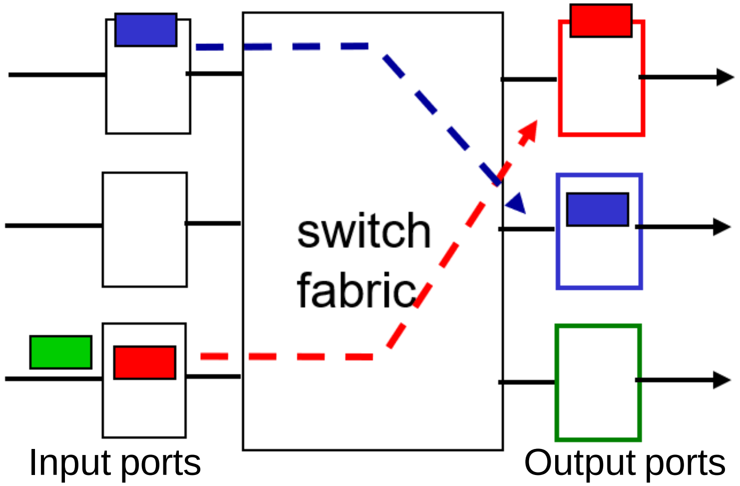
One shared bus, but can only fit one datagram

Input Queueing

fabric slower than input ports combined ➔*queueing may occur at input queues*



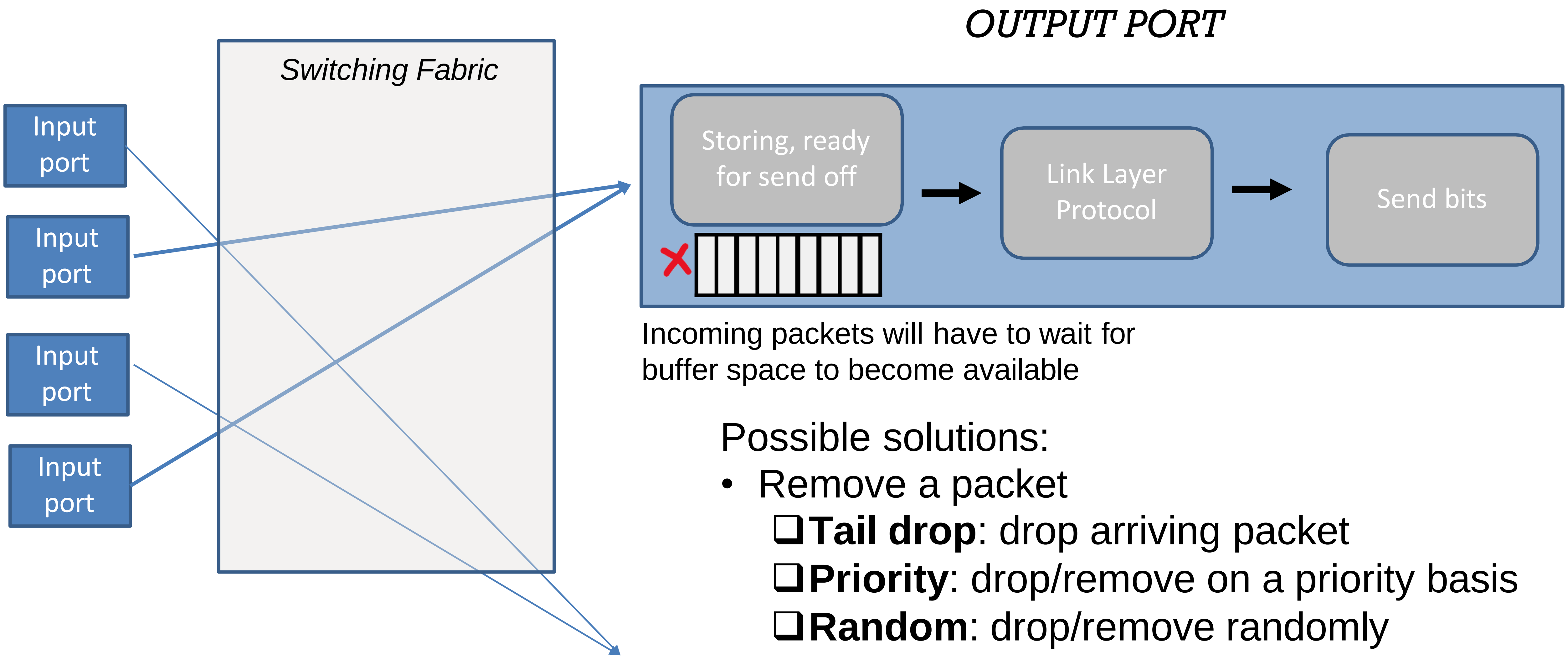
output port contention:
only one red datagram can be transferred.
lower red packet is blocked



one packet time later:
green packet experiences HOL blocking

Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward

Output Queueing



Incoming packets will have to wait for buffer space to become available

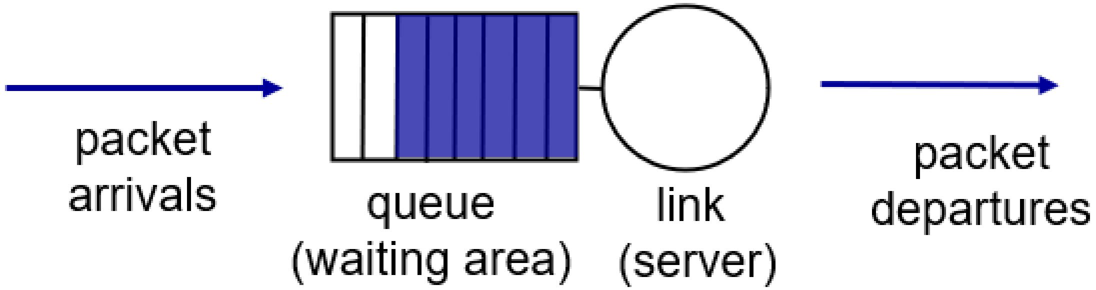
Possible solutions:

- Remove a packet
  - ❑ **Tail drop**: drop arriving packet
  - ❑ **Priority**: drop/remove on a priority basis
  - ❑ **Random**: drop/remove randomly

Packet Scheduling

Scheduling is used to determine the next packet to send on the link

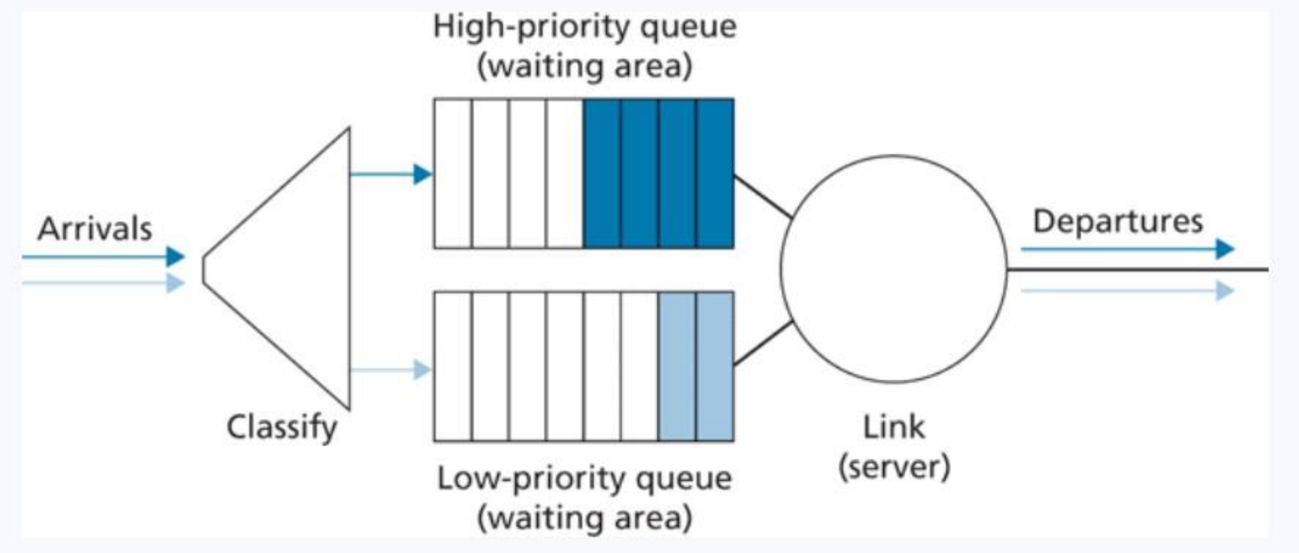
FIFO (first in first out): Send in order of arrival to queue

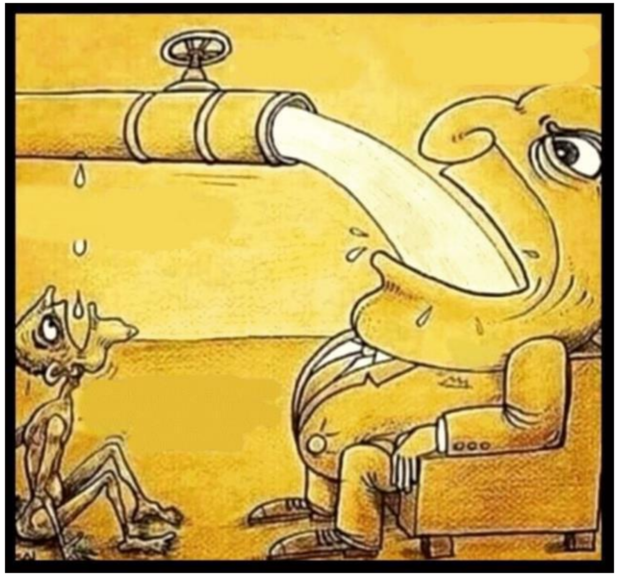


Packet Scheduling

Scheduling is used to determine the next packet to send on the link

Priority: packets are classified into priority classes. High priority = sent over link first

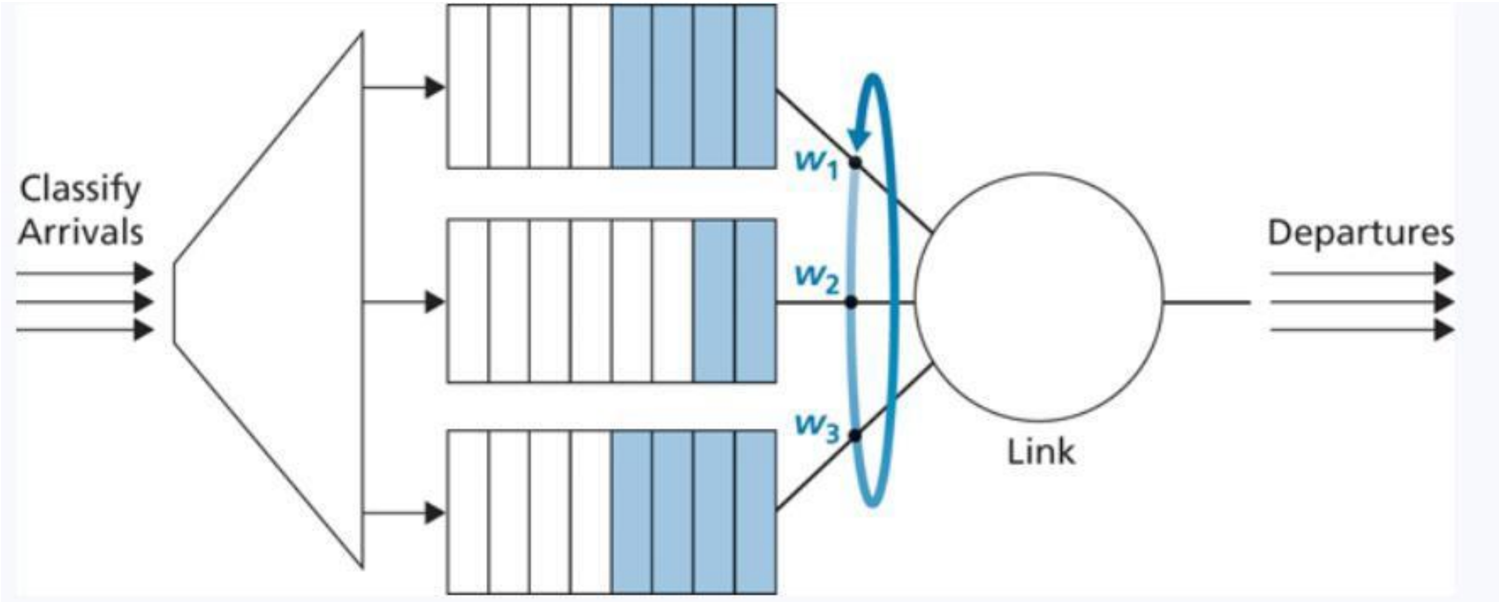




Packet Scheduling

Scheduling is used to determine the next packet to send on the link

Round robin weighted fair queueing: packets are classified into priority classes. Each class gets to send one packet during a "cycle"



Packet from the priority group gets sent

Packet from the middle class gets sent

Packet from the plebian class gets sent

IP PROTOCOL

IP addressing, IPv4, and IPv6

https://www.rfc-editor.org/rfc/rfc791

Packets traversing through the network layer are referred to as a **datagram**. Each packet gets an IPv4/IPv6 header

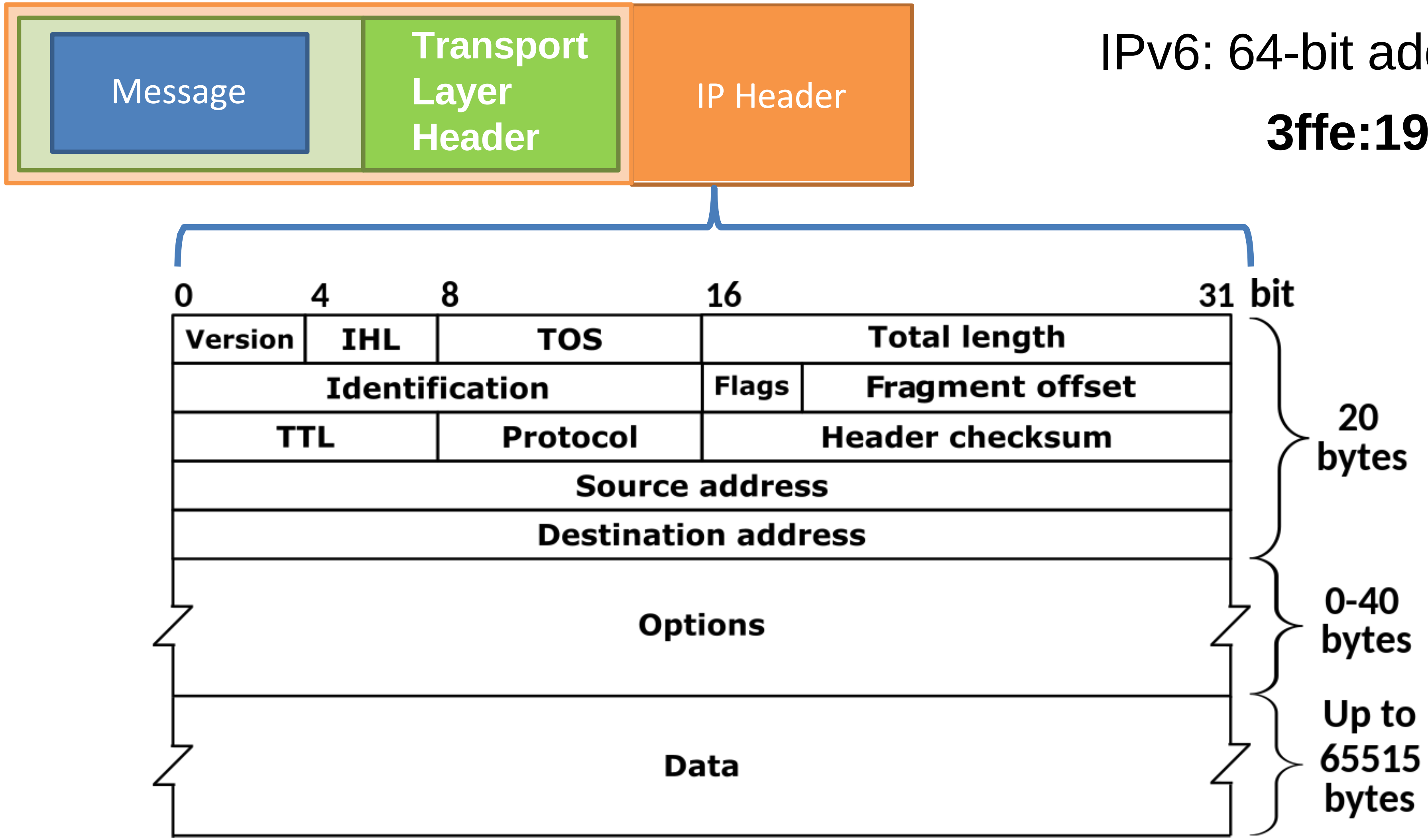
IPv4: 32-bit addresses (decimal)

192.149.252.76

IPv6: 64-bit addresses (hexademical)

3ffe:1900:fe21:4545::

Why do we need IPv6?



IPv4 Header

Announcements

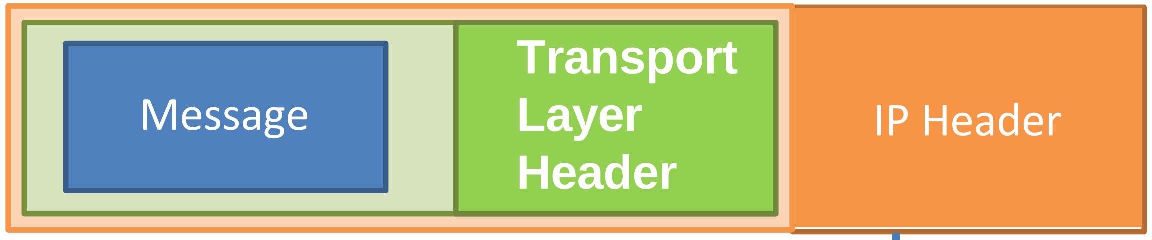
NO CLASS this Friday (10/14) and next Monday (10/17)

- You can always email/DM me if you have questions

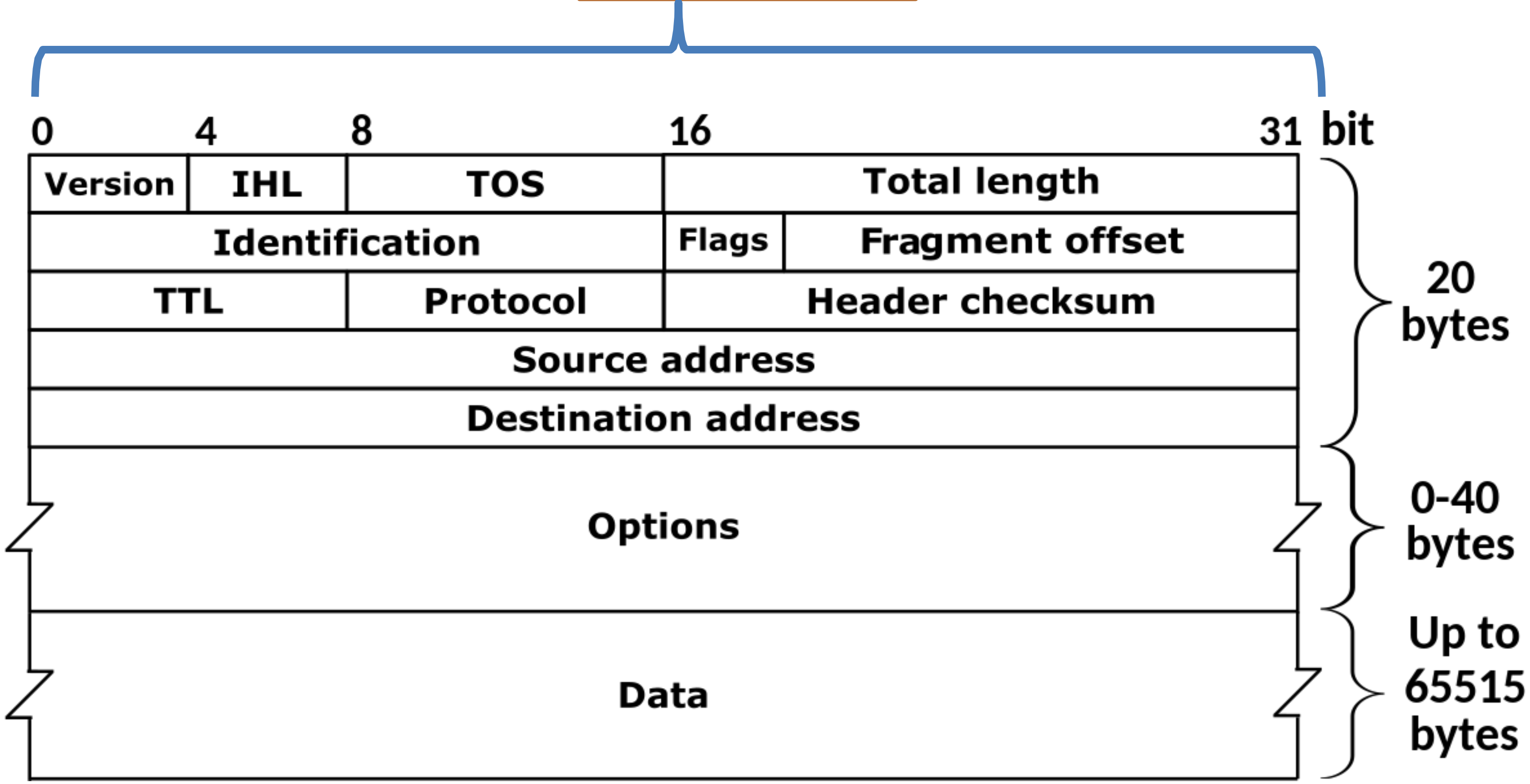
PA2 due in one week (10/19)

Put `time.sleep()` commands after you `send()` to prevent weird timing errors

Packets traversing through the network layer are referred to as a **datagram**. Each packet gets an IPv4/IPv6 header



TCP header = 20 bytes
IP header = 20 bytes



IPv4 Header

IP Address: Globally unique* 32 bit (4 byte) **dotted decimal** number assigned to interfaces on hosts and routers

193.32.216.9

IP Address: Globally unique* 32 bit (4 byte) **dotted decimal** number assigned to interfaces on hosts and routers

(1 byte = 8 bits)

193.32.216.9

4 bytes 4 bytes 4 bytes 4 bytes

128	64	32	16	8	4	2	1
1	**1**	0	0	0	0	0	**1**

128 + 64 + 1 = **193**

193 = `11000001` in binary

IP Address: Globally unique* 32 bit (4 byte) **dotted decimal** number assigned to interfaces on hosts and routers

(1 byte = 8 bits)

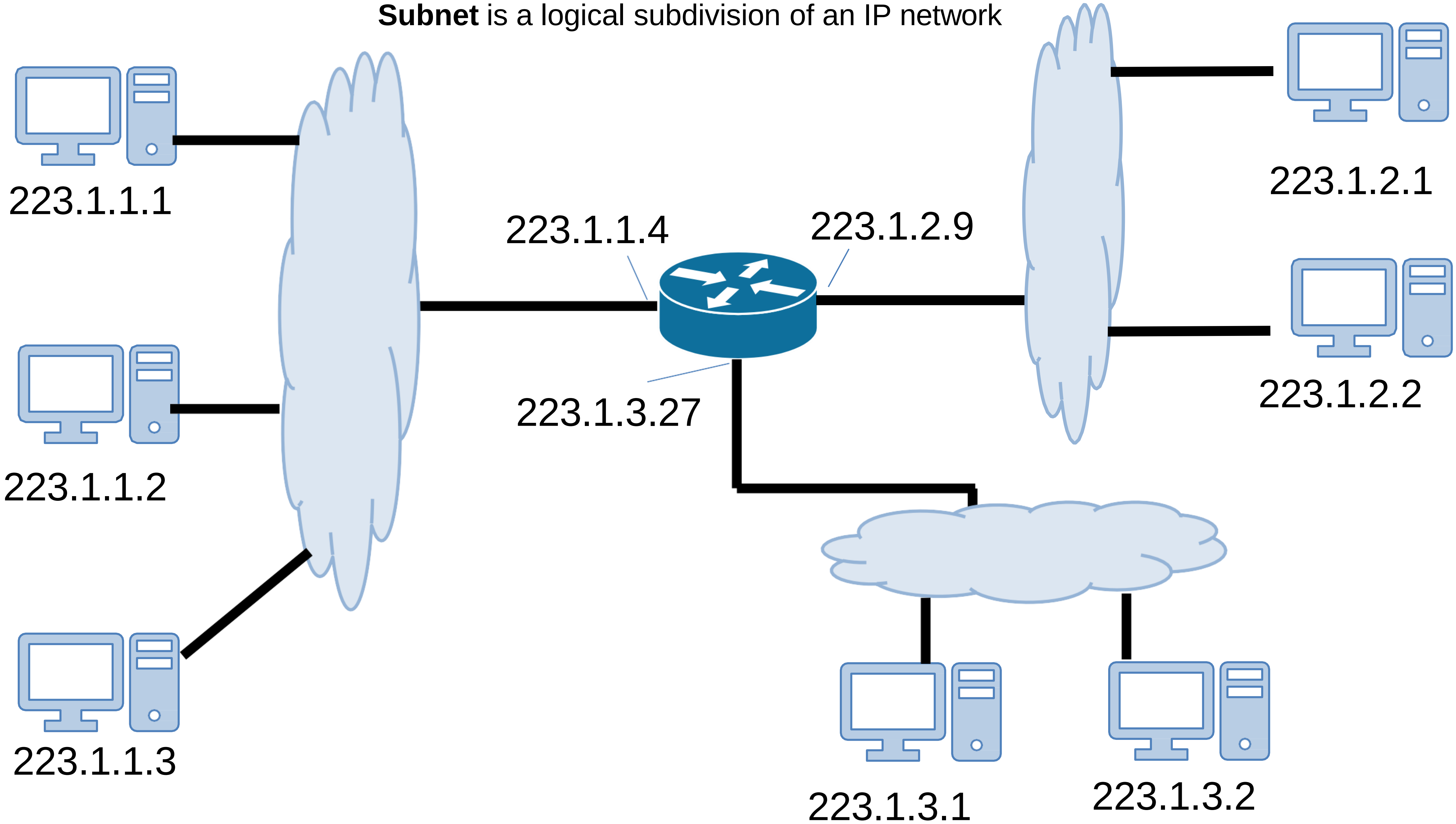
193.32.216.9

=

11000001 00100000 11011000 00001001

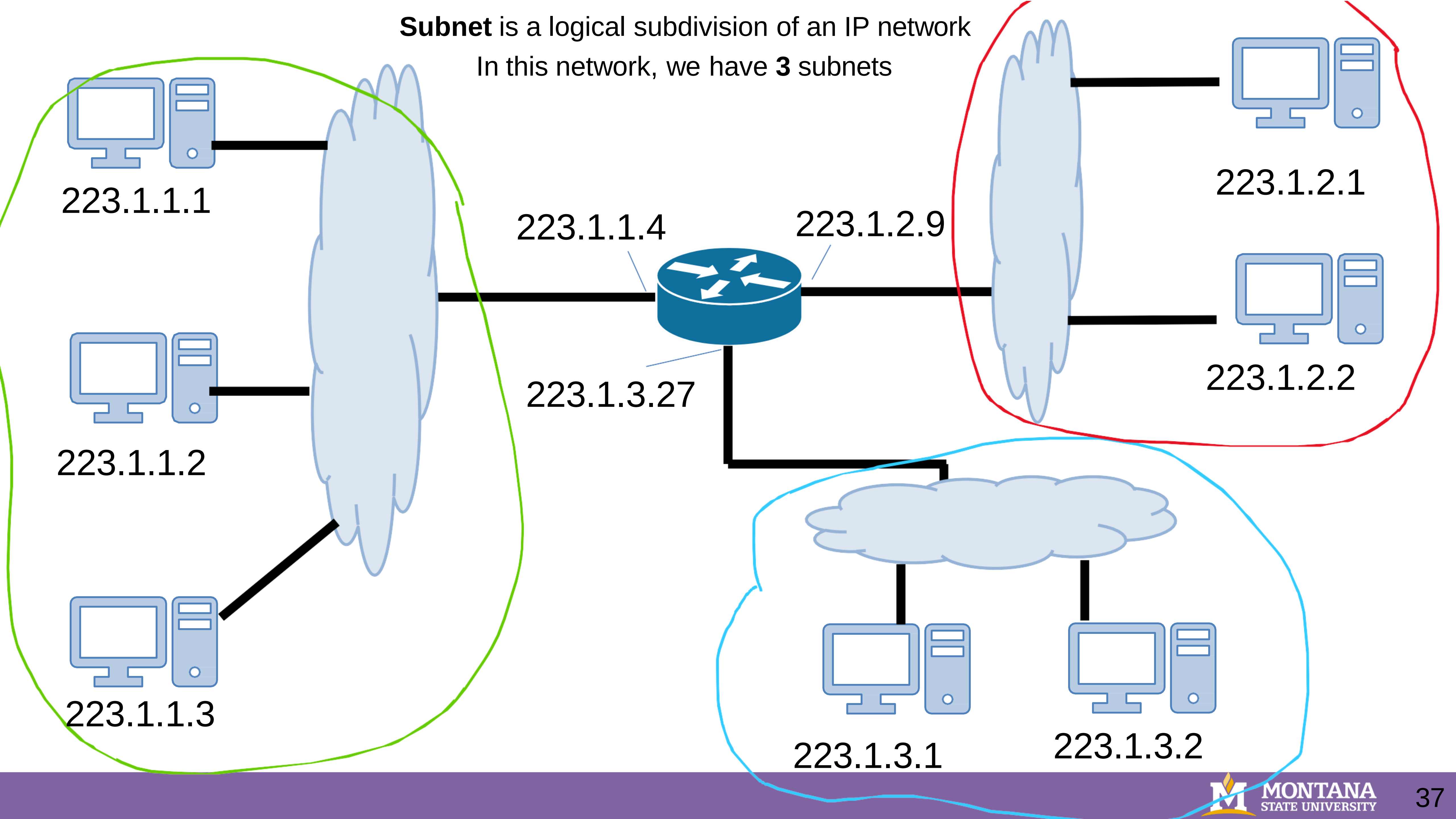
How many addresses are possible with a 32 bit number? **~4 billion possible IPv4 addresses**

Subnet is a logical subdivision of an IP network



Subnet is a logical subdivision of an IP network

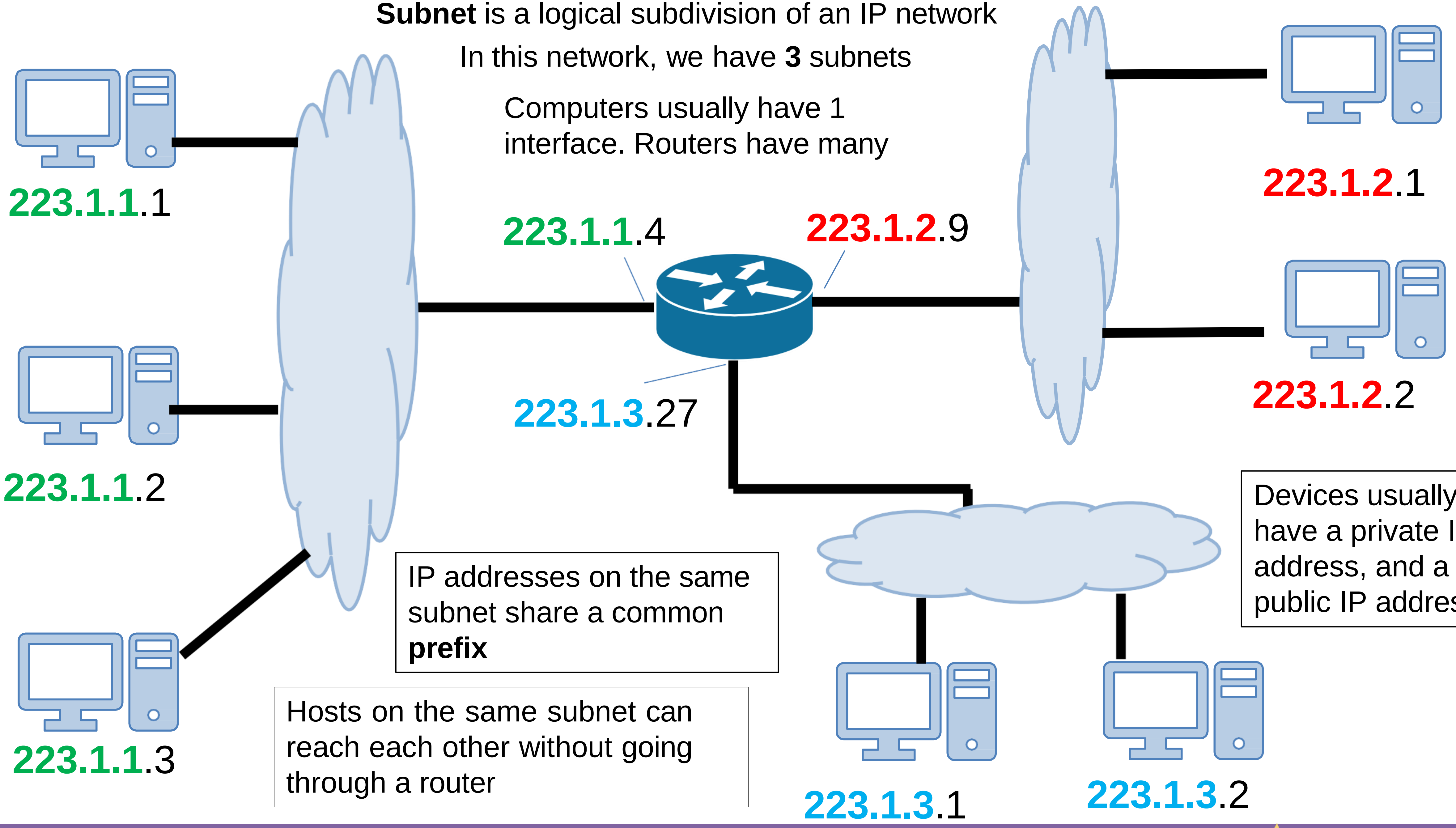
In this network, we have **3** subnets



Subnet is a logical subdivision of an IP network

In this network, we have **3** subnets

Computers usually have 1 interface. Routers have many

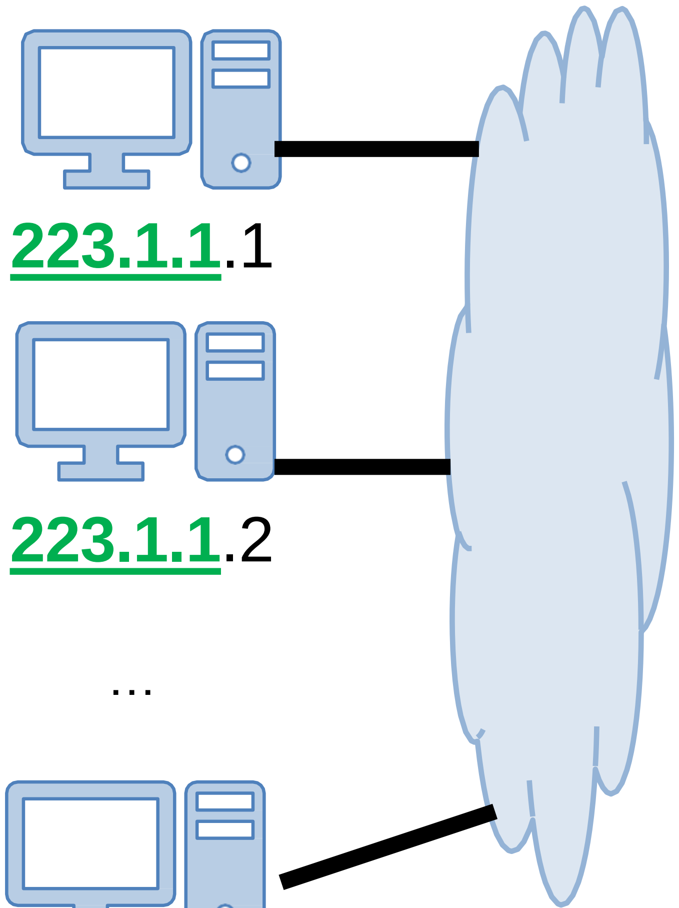


IP addresses on the same subnet share a common **prefix**

Hosts on the same subnet can reach each other without going through a router

Devices usually have a private IP address, and a public IP address

It is very common to have a **range** of IP addresses assigned to you (random assignment would be chaos)



223.1.1.1

223.1.1.2

...

223.1.1.253

Subnet mask

223.1.1.0/24

The leftmost 24 bits represent the prefix of the subnet

11111111 11111111 11111111 XXXXXXXX = **255.255.255**.0

223.1.1.67 ✓

223.2.1.67 ✗

223.2.1.255 ✗

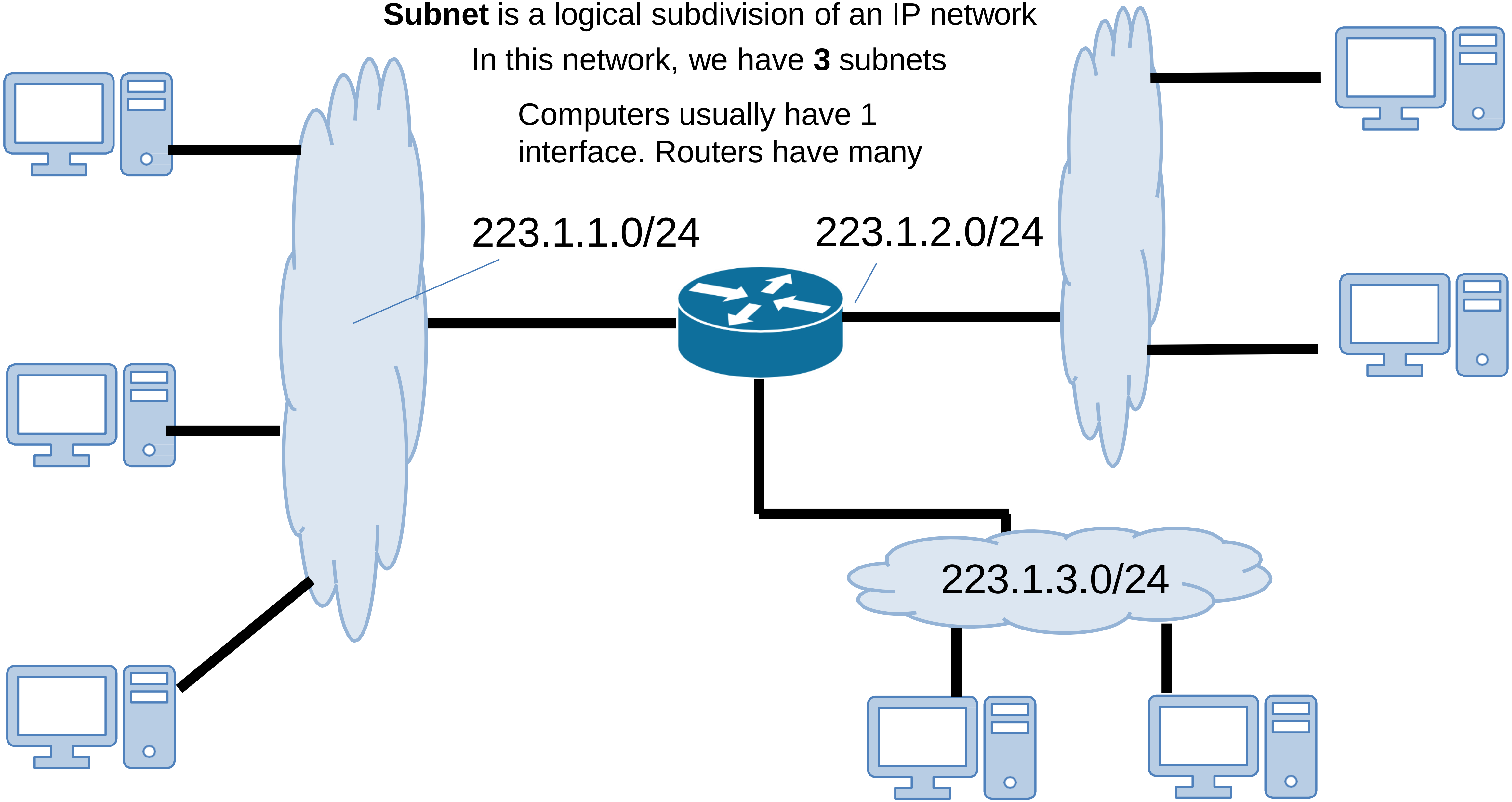
Network bits **193.32.216.9** **Host bits**

11000001 00100000 11011000 00001001

Subnet is a logical subdivision of an IP network

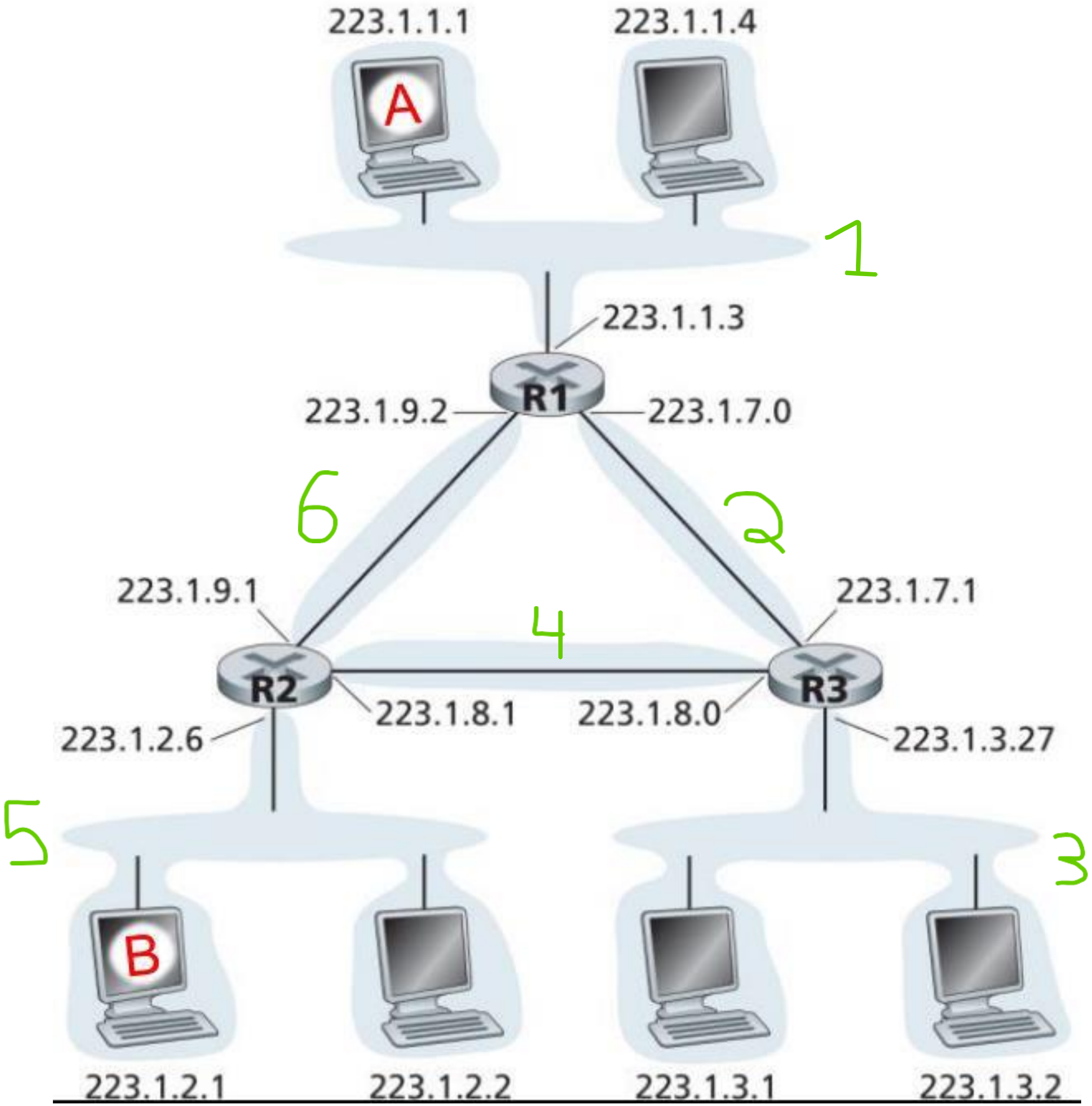
In this network, we have **3** subnets

Computers usually have 1 interface. Routers have many



6 subnets

1. 223.1.1.0/24
2. 223.1.7.0/24
3. 223.1.3.0/24
4. 223.1.8.0/24
5. 223.1.2.0/24
6. 223.1.9.0/23



Managing and Assigning Subnets

The subnet `223.1.1.0/24` contains the following addresses

```
223.1.1.1
223.1.1.2
223.1.1.3
223.1.1.4
223.1.1.5
…
223.1.1.252
223.1.1.253
223.1.1.254
```

How many addresses does a /24 subnet provide?

0 and 255 are reserved for special services (?)

Managing and Assigning Subnets

The subnet `223.1.1.0/24` contains the following addresses

```
223.1.1.1
223.1.1.2
223.1.1.3
223.1.1.4
223.1.1.5
…
223.1.1.252
223.1.1.253
223.1.1.254
```

How many addresses does a /24 subnet provide?

2^8 – 2 = **253**

Class C network

0 and 255 are reserved for special services (?)

Managing and Assigning Subnets

Subnet Type	Slash	Subnet Mask	Available Addresses
Class A	/8	255.0.0.0	2^24 – 2 = **16777216**
Class B	/16	255.255.0.0	2^16 – 2 = **65,634**
Class C	/24	255.255.255.0	2^8 – 2 = **254**

Issues with this type of assignment?

Managing and Assigning Subnets

Subnet Type	Slash	Subnet Mask	Available Addresses
Class A	/8	255.0.0.0	2^24 – 2 = **16777216**
Class B	/16	255.255.0.0	2^16 – 2 = **65,634**
Class C	/24	255.255.255.0	2^8 – 2 = **254**

If I have 2000 devices, I will either need 8 class C subnets or 1 class B subnet (but waste 63,000ish IP addresses)

We need a better way to subnet!

Managing and Assigning Subnets

Our only available subnet masks have been:

255.255.255.0	11111111 11111111 11111111 00000000	/24
255.255.0.0	11111111 11111111 00000000 00000000	/16
255.0.0.0	11111111 00000000 00000000 00000000	/8

CIDR introduces a more flexible way for subnetting

We can any number of bits for our mask

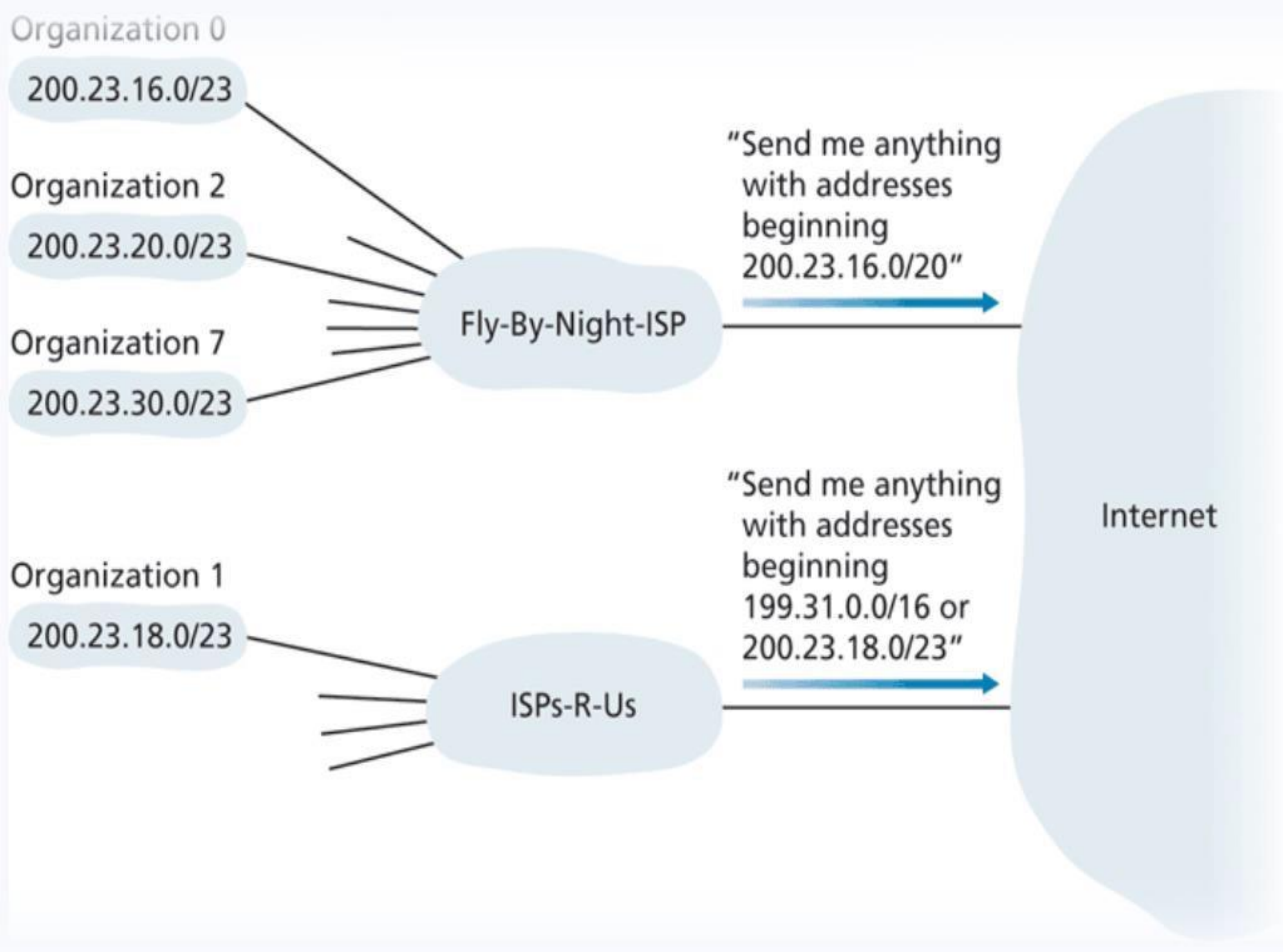
Host bits!

200.23.16.0**/20** = 11001000.00010111.0001**0000.00000000**

Mask = 11111111.11111111.1111**0000.00000000**

ISPs can now group and advertise organizations by IP blocks

We once again have another hierarchy ☺

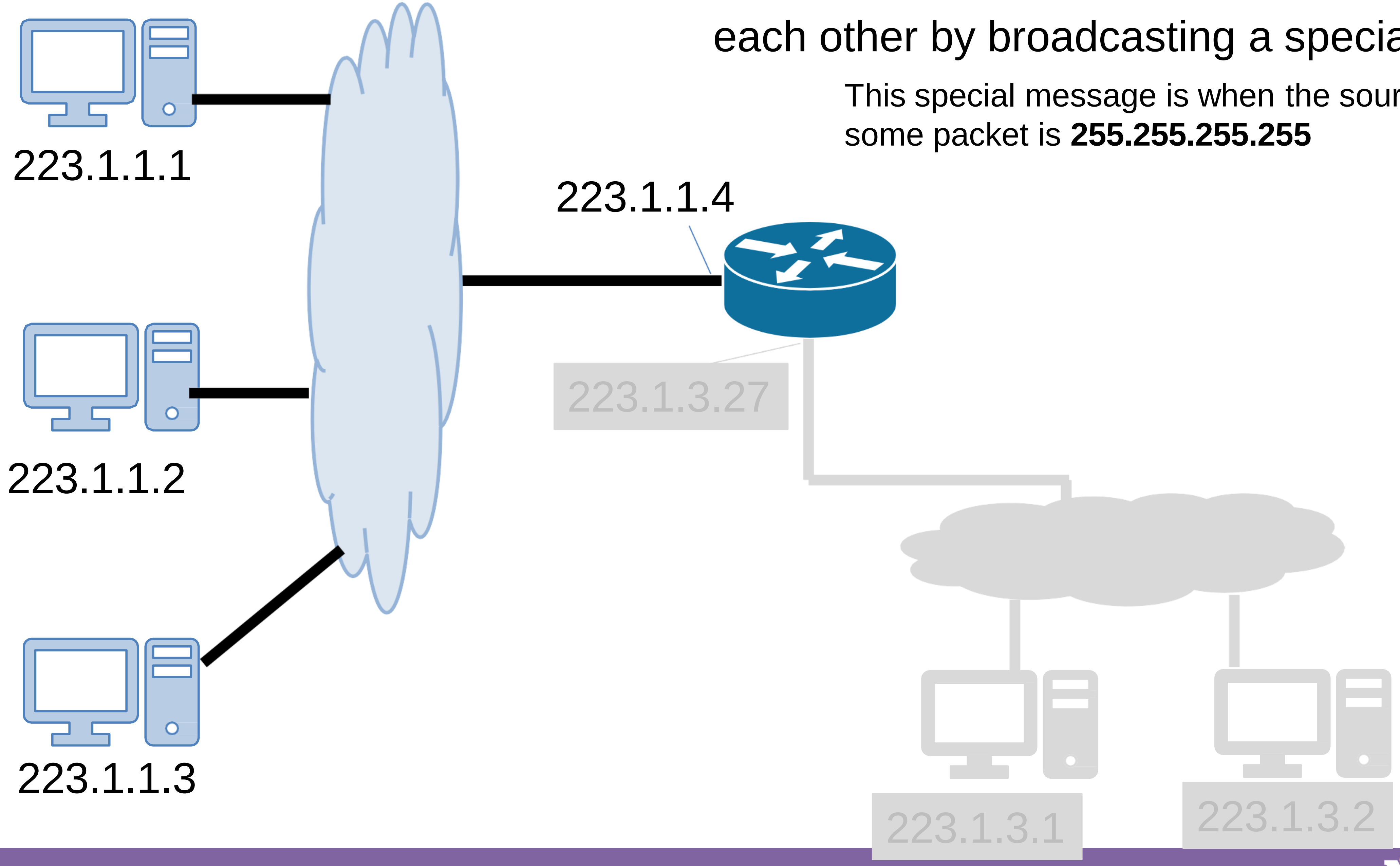


One possible way an ISP could organize IP blocks

ISP's block	200.23.16.0/20	11001000 00010111 00010000 00000000
Organization 0	200.23.16.0/23	**11001000 00010111 0001000**0 00000000
Organization 1	200.23.18.0/23	**11001000 00010111 0001001**0 00000000
Organization 2	200.23.20.0/23	**11001000 00010111 0001010**0 00000000
…		
Organization 7	200.23.30.0.23	**11001000 00010111 0001111**0 00000000

Devices on the same subnet communicate with each other by broadcasting a special message

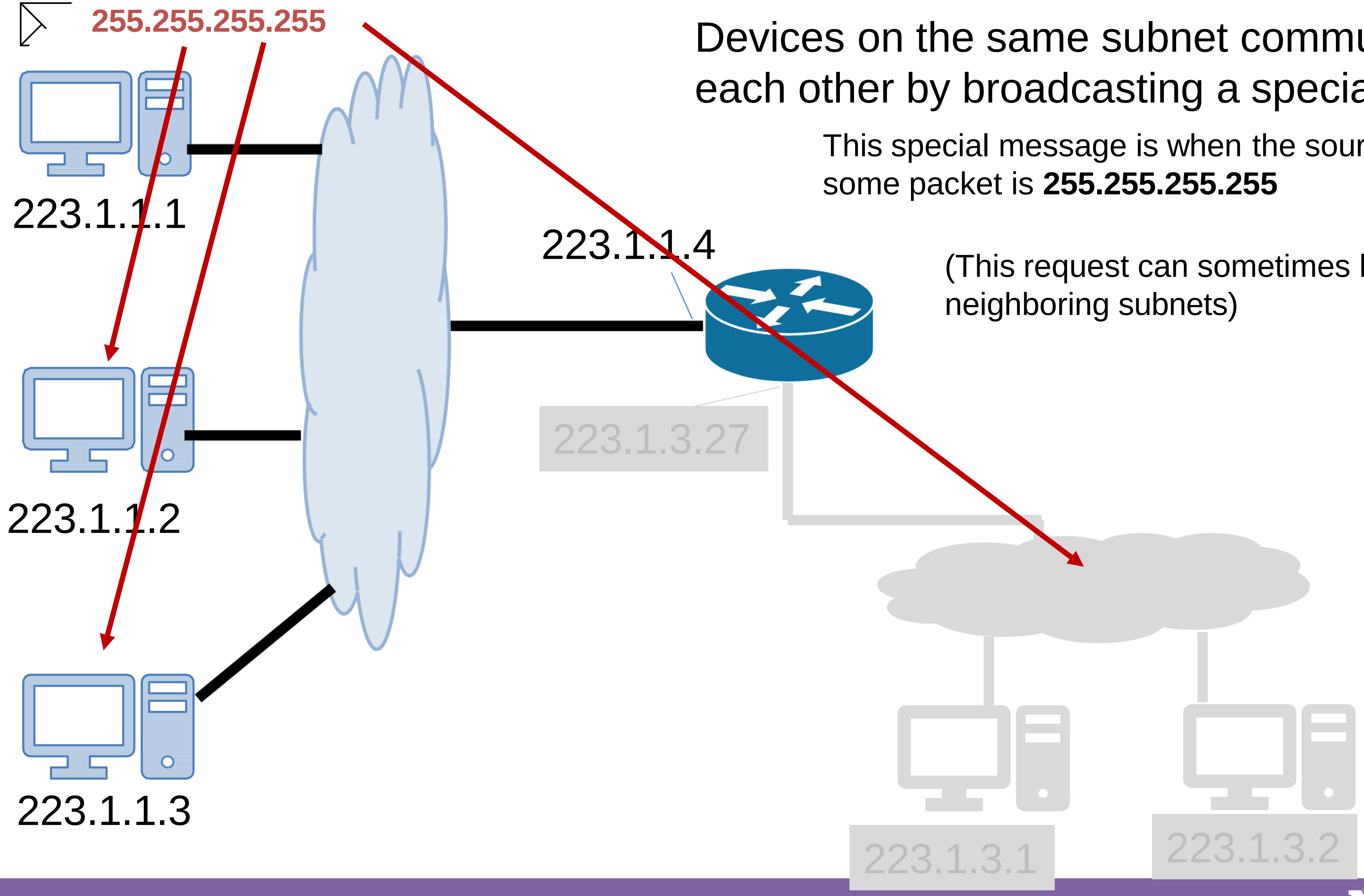
This special message is when the source IP address of some packet is **255.255.255.255**



Devices on the same subnet communicate with each other by broadcasting a special message

This special message is when the source IP address of some packet is **255.255.255.255**

(This request can sometimes be echoed to neighboring subnets)



How do IPs get obtained/assigned?

The Internet Corporation for Assigned Names and Numbers (ICANN) is responsible for managing and allocating IP address space for ISPs and organizations

(they also manage the DNS root servers!)

When an organization gets a range of IP addresses to use, how to we give assign them to devices?

Do we do it manually?

we could….

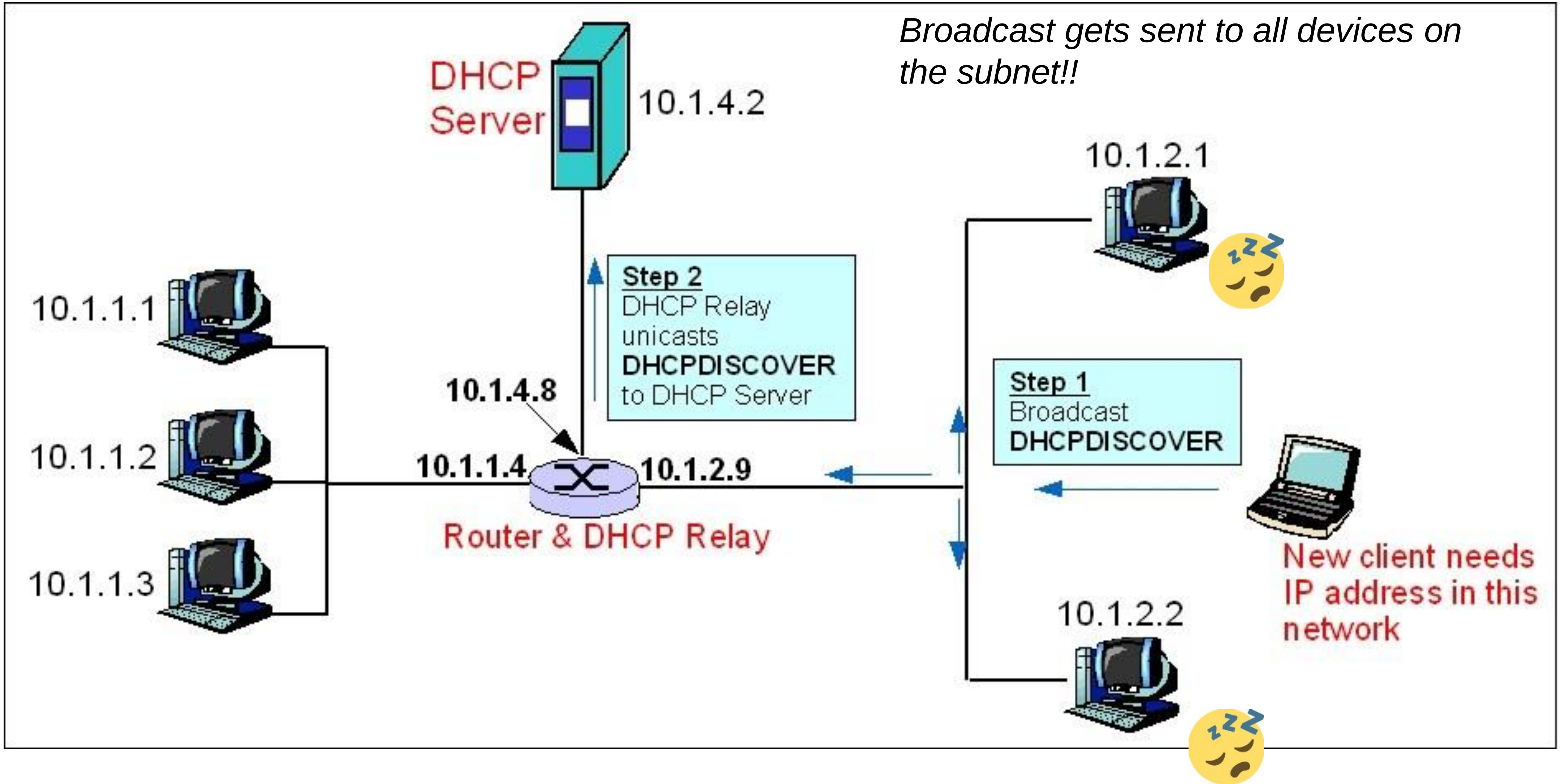
DHCP

Dynamic Host Configuration Protocol (DHCP) is a **plug-and-play**, client-server protocol that allows a host to obtain an IP address automatically

When a host is automatically assigned an IP address, it might keep that one forever, or the IP addresses can be temporary

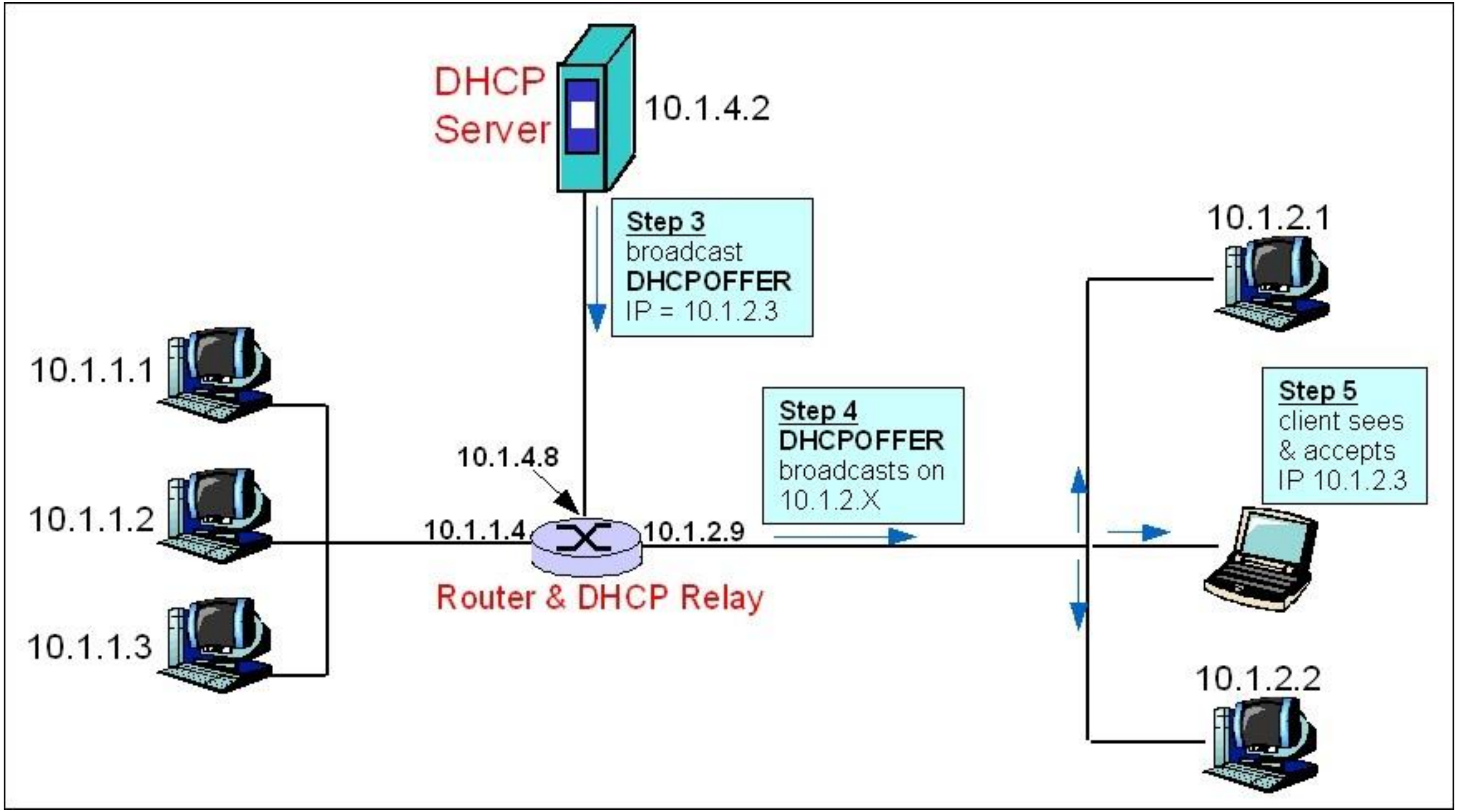
(more common)

This process is similar to a TCP handshake!

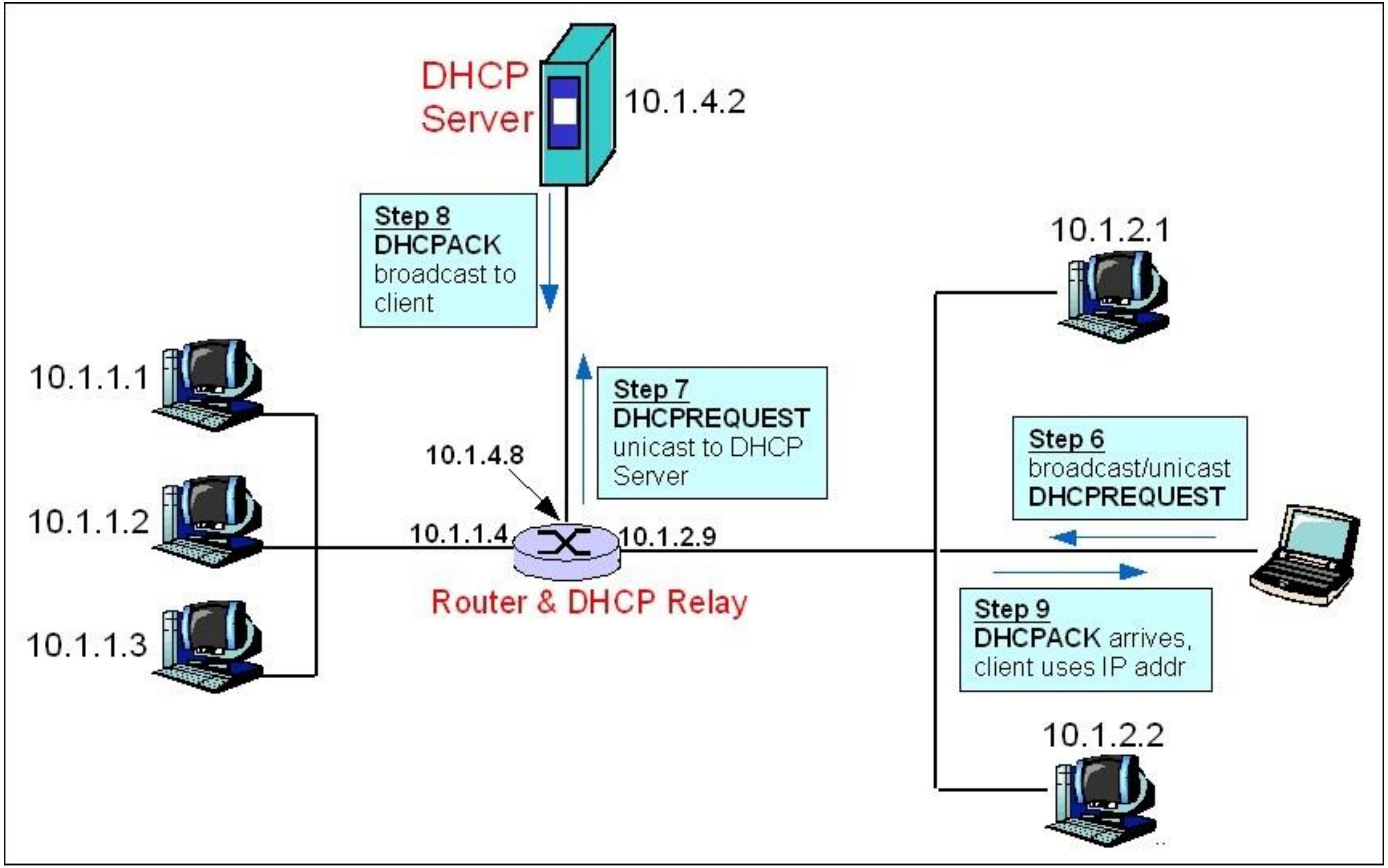


But devices that are **not** the dhcp server will ignore it

This process is similar to a TCP handshake!



This process is similar to a TCP handshake!

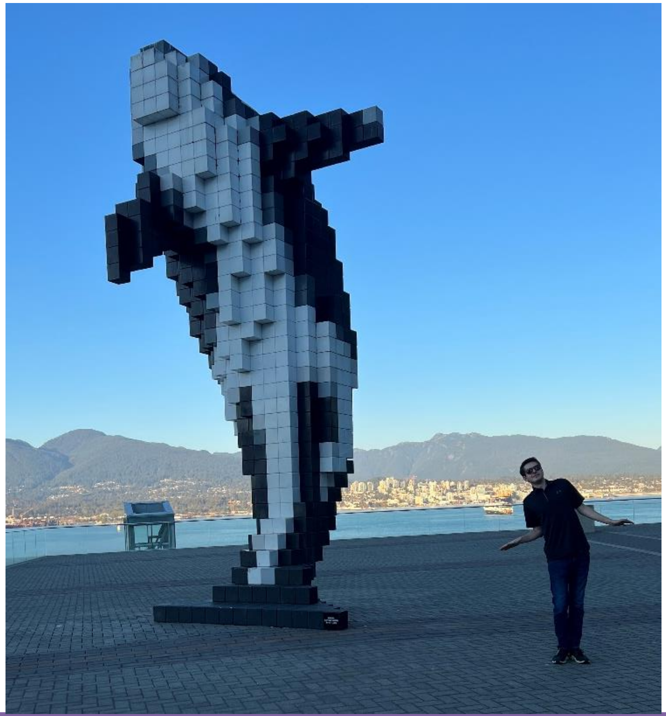


Announcements

If I haven't responded to a DM/email, please poke me about it

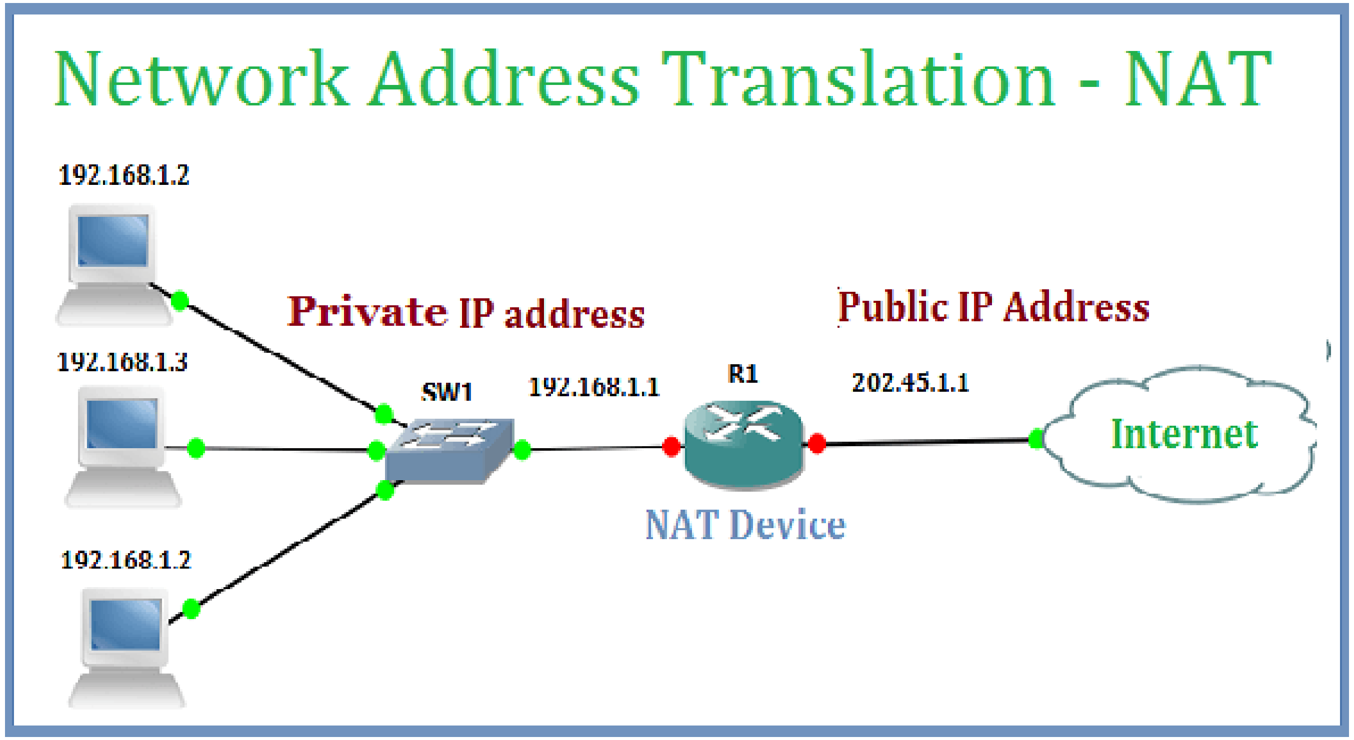
PA2 due on Friday 10/21
- I will allow you to submit it any time this weekend with no late penalty (no late pass required!)
- Lecture will be asynchronous (recorded) on Monday

PA2: Flow Demo, FIN ACKs



NAT (Network Address Translation)

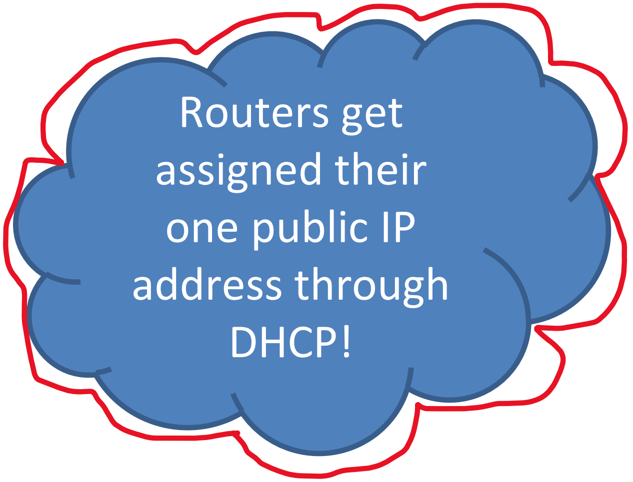
NAT is a translation of multiple private IP addresses to one single public IP address

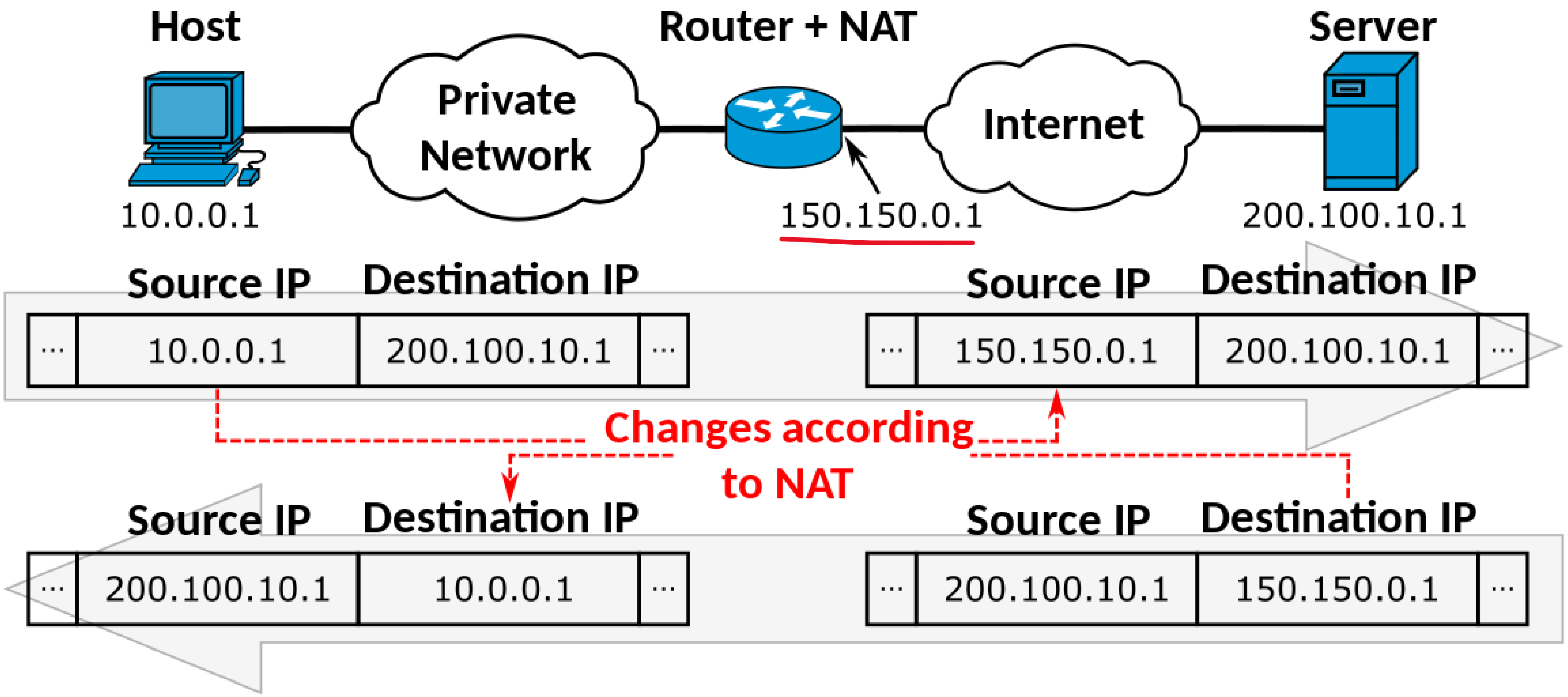


NAT (Network Address Translation)

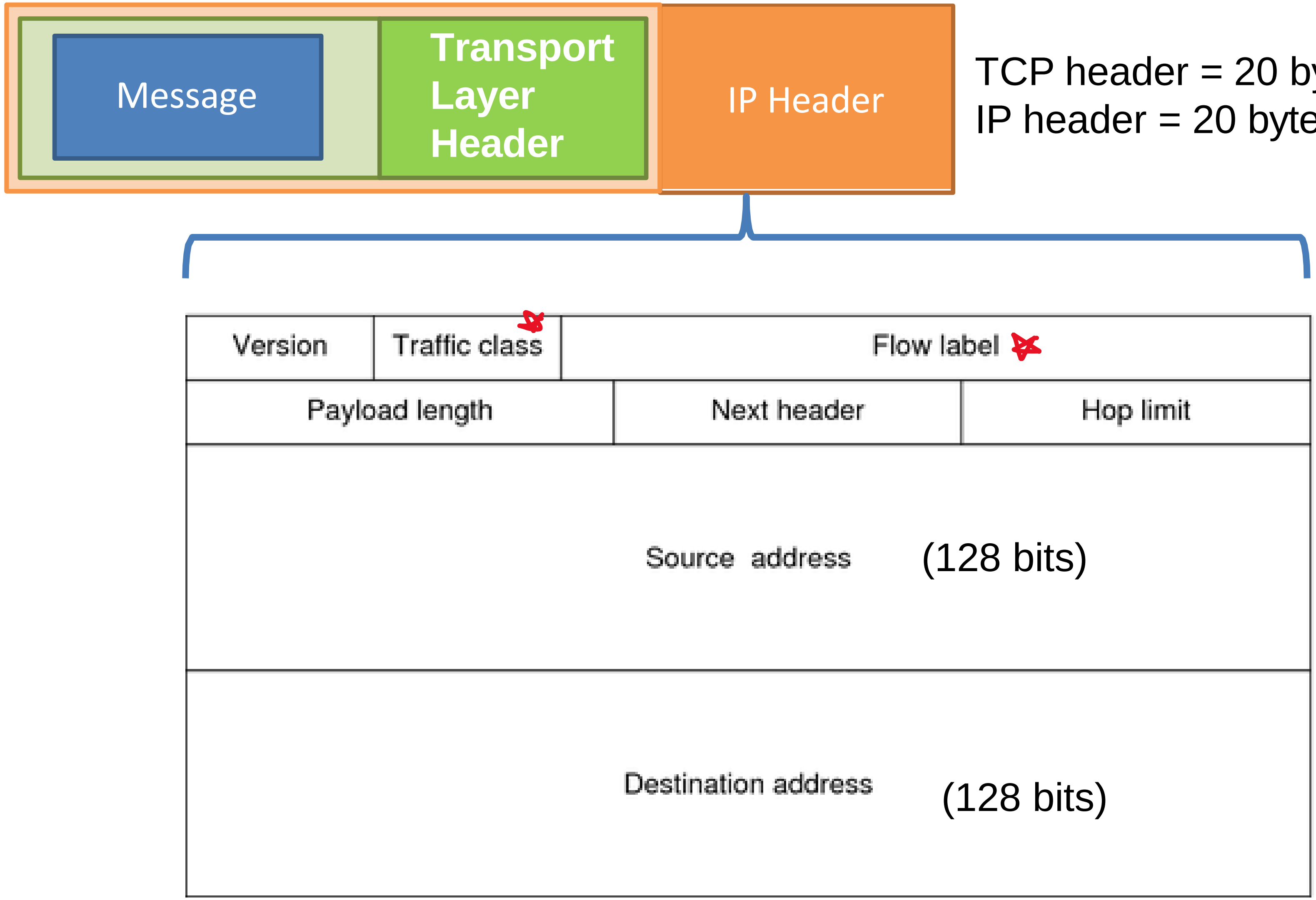
NAT is a translation of multiple private IP addresses to one single public IP address

- Hides details of inner home network from outside world
- All incoming traffic will have same public IP, all outgoing will have same public IP





Packets traversing through the network layer are referred to as a **datagram**. Each packet gets an IPv4/IPv6 header



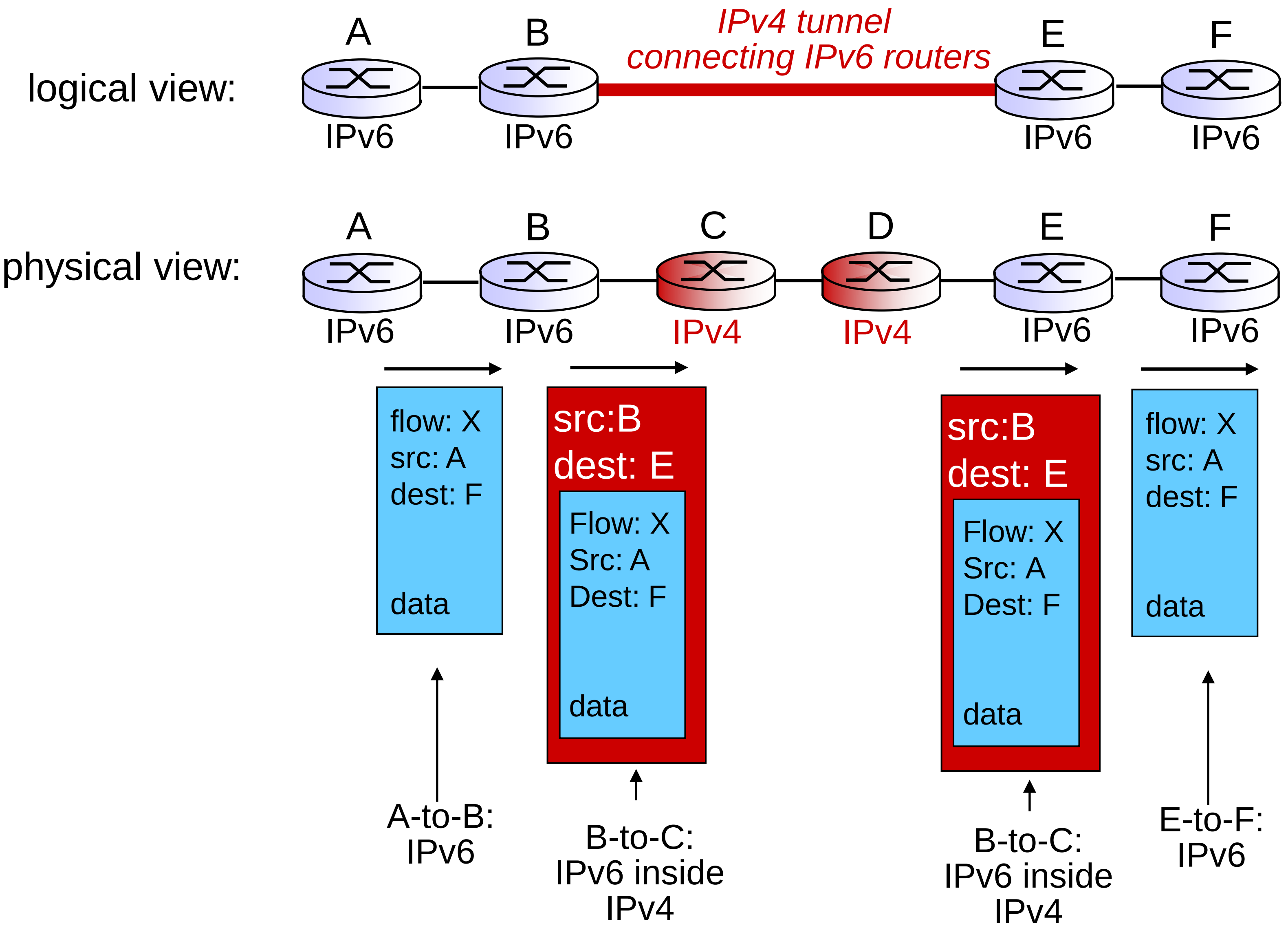
TCP header = 20 bytes
IP header = 20 bytes

IPv6 = **128 bits** (not 64)

IPv6 Header

The goal is for our internet to be only using IPv6 in the near future? Issues?

Tunneling



logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6
IPv6
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
flow: X
src: A
dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
flow: X
src: A
dest: F
data
A-to-B:
IPv6
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
E-to-F:
IPv6

So far, a router takes input from input port, and then outputs on some output port

What else might a router need to do?

So far, a router takes input from input port, and then outputs on some output port

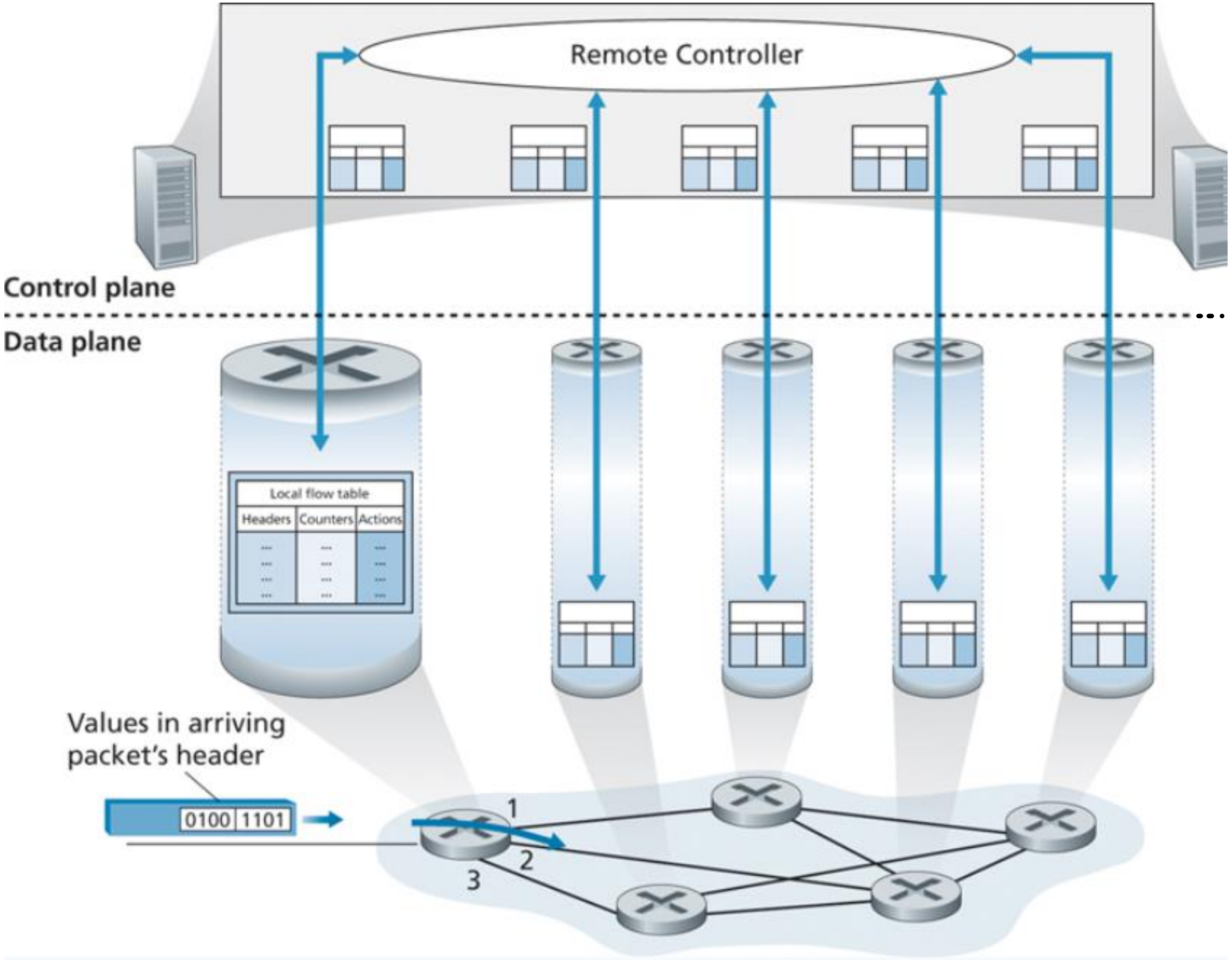
What else might a router need to do?

Forward, Drop stuff, Modify, Load balance

We need more flexibility and functionality with our forwarding!!

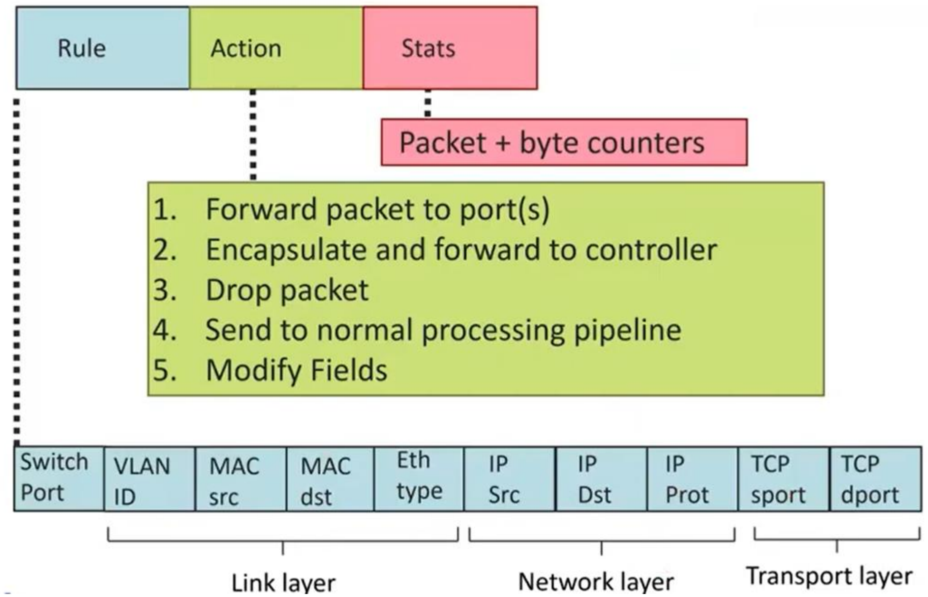
Generalized Forwarding and Software Defines Network (SDN)

OpenFlow standard

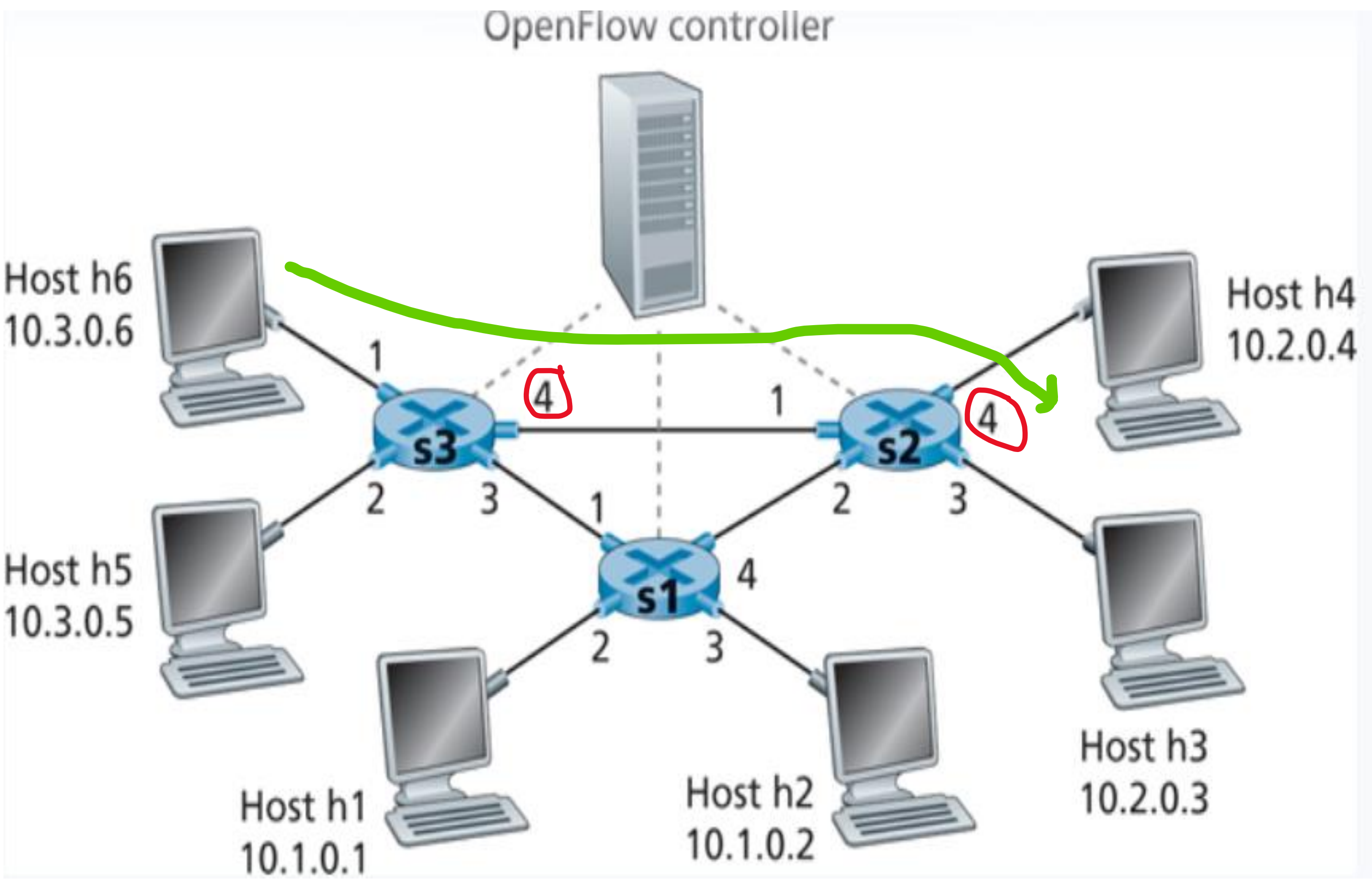


We need **headers/rules**, which are going to the values the remote controller is going to evaluate

We need **actions** to do based on some pattern match



Generalized Forwarding and Software Defines Network (SDN)

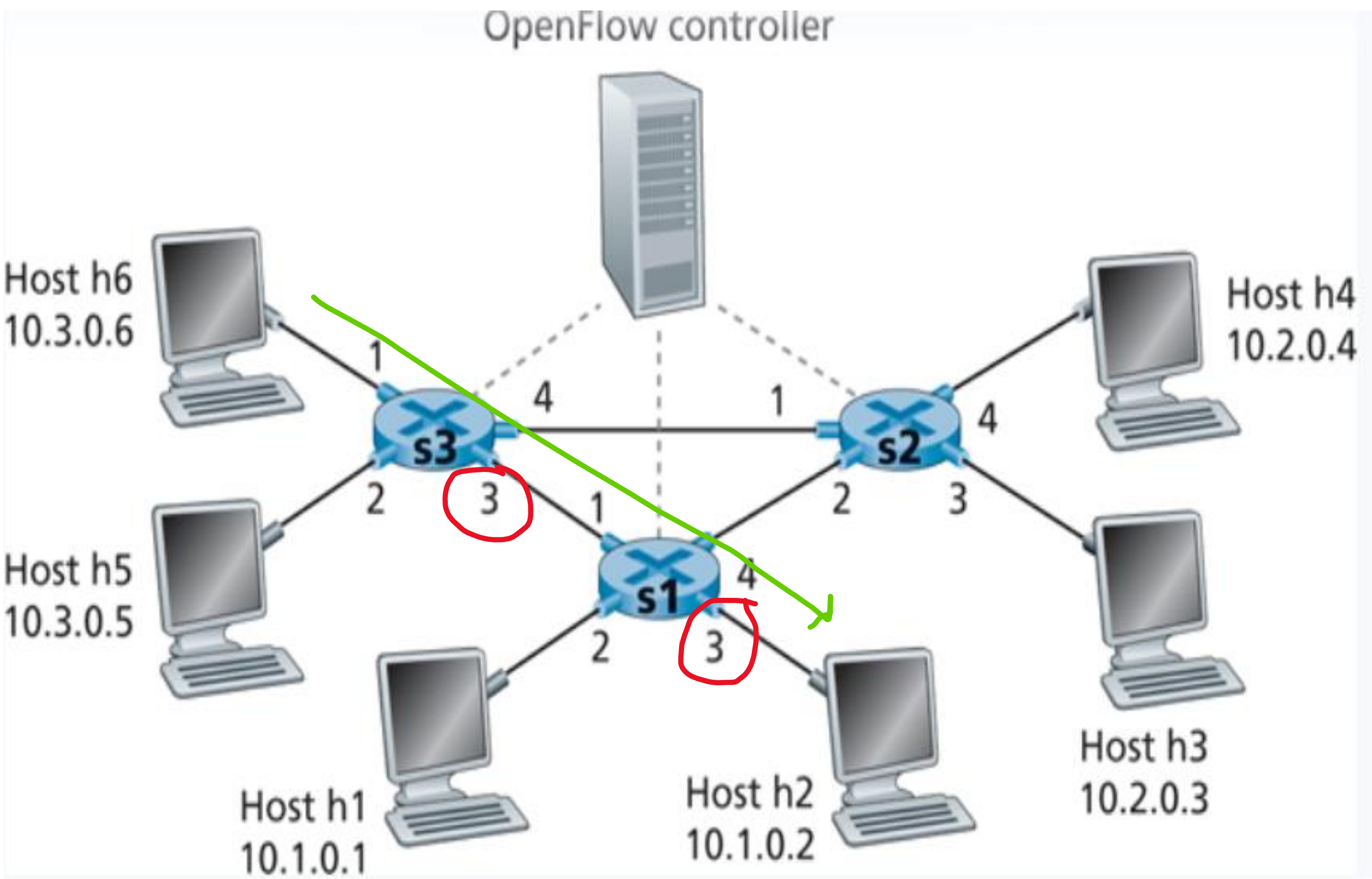


Match	Action
Ingress Port = 1 ; IP Src = 10.3.*.* ; IP Dst = 10.2.*.*	Forward(4)

Match	Action
IP Src = 10.3.*.* ; IP Dst = 10.2.*.*	Forward(3)

Match	Action
Ingress port = 2 ; IP Dst = 10.2.0.3	Forward(3)
Ingress port = 2 ; IP Dst = 10.2.0.4	Forward(4)

Generalized Forwarding and Software Defines Network (SDN)

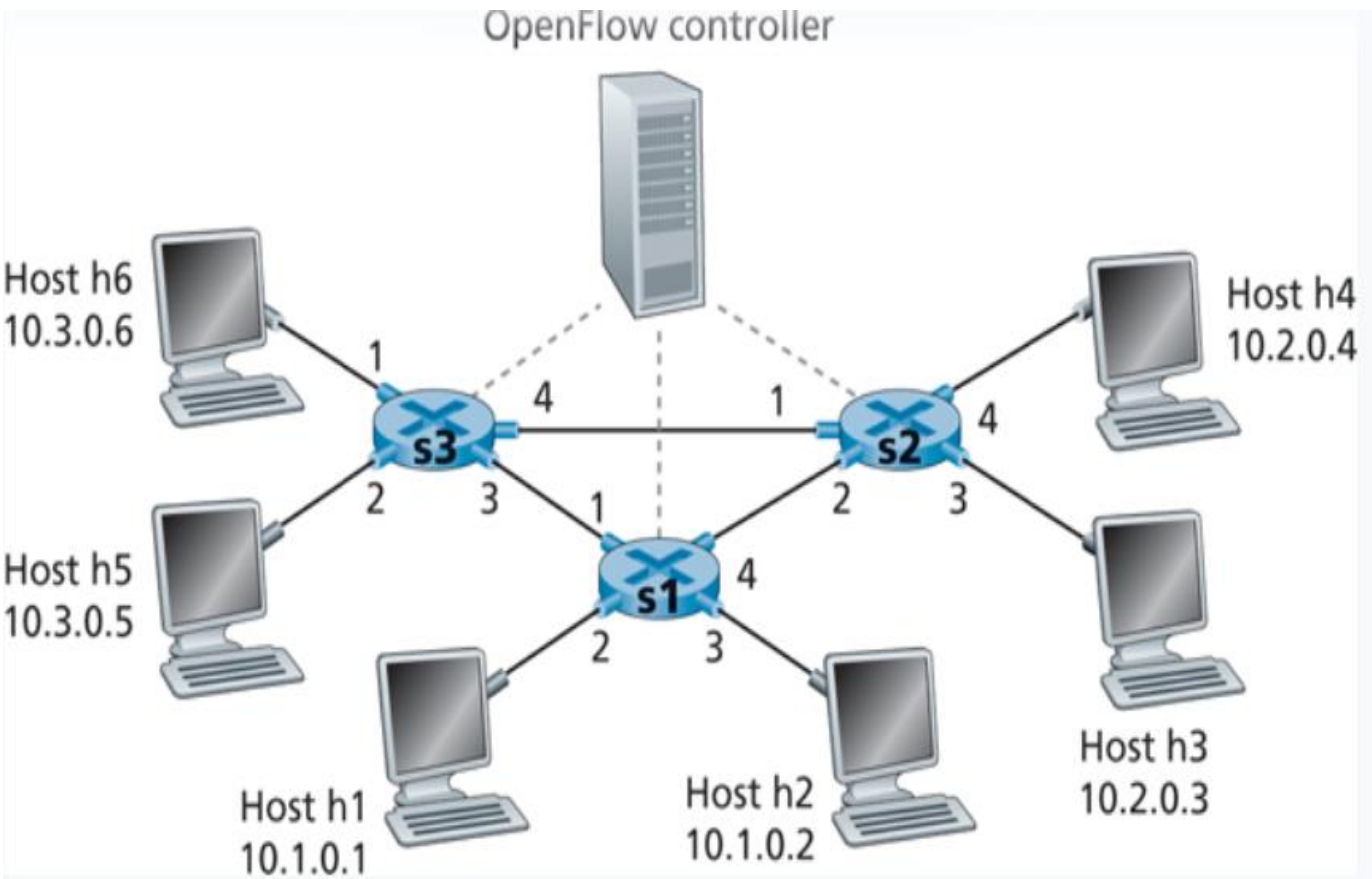


Match	Action
Ingress Port = 1 ; IP Src = 10.3.*.* ; IP Dst = 10.2.*.*	Forward(4)

Match	Action
IP Src = 10.3.*.* ; IP Dst = 10.2.*.*	Forward(3)

Match	Action
Ingress port = 2 ; IP Dst = 10.2.0.3	Forward(3)
Ingress port = 2 ; IP Dst = 10.2.0.4	Forward(4)

Generalized Forwarding and Software Defines Network (SDN)

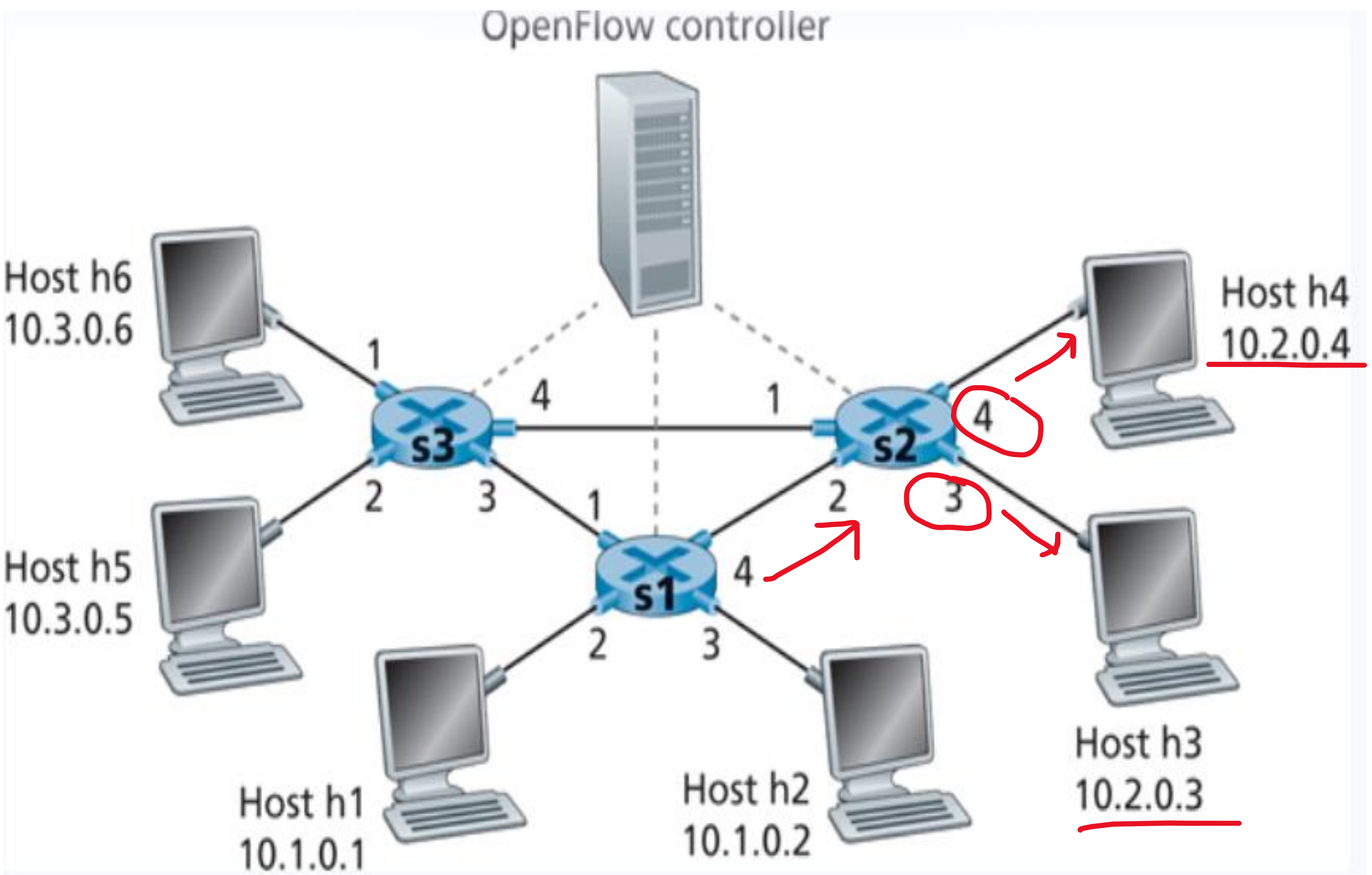


Match	Action
Ingress Port = 1 ; IP Src = 10.3.*.* ; IP Dst = 10.2.*.*	Forward(4)

Match	Action
IP Src = 10.3.*.* ; IP Dst = 10.2.*.*	Forward(3)

Match	Action
Ingress port = 2 ; IP Dst = 10.2.0.3	Forward(3)
Ingress port = 2 ; IP Dst = 10.2.0.4	Forward(4)

Generalized Forwarding and Software Defines Network (SDN)



Match	Action
Ingress Port = 1 ; IP Src = 10.3.*.* ; IP Dst = 10.2.*.*	Forward(4)

Match	Action
IP Src = 10.3.*.* ; IP Dst = 10.2.*.*	Forward(3)

Match	Action
Ingress port = 2 ; IP Dst = 10.2.0.3	Forward(3)
Ingress port = 2 ; IP Dst = 10.2.0.4	Forward(4)

load balancing

Destination-based forwarding:

	Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
Pattern →	*	*	*	*	*	*	51.6.0.8	*	*	*	port6

← Action

IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6

Firewall:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	*	*	*	*	22	drop

do not forward (block) all datagrams destined to TCP port 22

Destination-based layer 2 (switch) forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	22:A7:23:11:E1:02	*	*	*	*	*	*	*	*	port3

layer 2 frames from MAC address 22:A7:23:11:E1:02 should be forwarded to output port 6