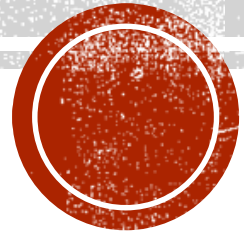
SOFTWARE DEFINED NETWORKING AND SECURITY

CHAPTER 1 INTRODUCTION OF COMPUTER NETWORKS



Dijiang Huang, Ankur Chowdhary, and Sandeep Pisharody

A SIMPLE NETWORKING QUESTION

- An application scenario description:
  - "On the university campus, you boot up a laptop, open an email application, compose an email, provide the receiver's email address (e.g., john@xyz.com) and other related information such as email subject, make sure the email content and everything else is fine, and finally click send."

- Question:
  - "During this procedure, what networking and application protocols have been invoked and what magic happened in the computer network to allow John to receive and view your email?"

AN ANALOGY BY SENDING A REGULAR MAIL?

- Write down both sender's and receiver's addresses on the envelope, sealing it, and finally adding the requisite postage on the envelope, which is quite similar to invoking an email application to write an email before clicking the "send" button.

- Drop the letter into a mailbox of a local post office is equivalent to click on the "send" button on the email client and the email is encapsulated in IP packets.

AN ANALOGY BY SENDING A REGULAR MAIL?

- On the letter:
  - The receiver's name tells who should open the letter, which is similar to the port number in the IP packet that tells which application (i.e., email) should open the message
  - The street number serves the similar function as the host address of an IP address.
  - The city, country and post codes serve the similar function as the network prefix of an IP address.

- Delivery:
  - The inter-city delivery of a letter only look at the city, country, and post codes, while Internet routers only look at the network prefix to deliver IP packet.
  - The local delivery of the letter will look at the street number, while the on the destination's local network, the host address is used to deliver the message to the destination host.

AN ANALOGY BY SENDING A REGULAR MAIL?

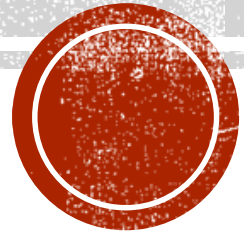
- Mail ←→ Email
- Message ←→ Data content
- Envelope ←→ Packet headers
- Sender and receiver's addresses ←→ Source and destination IP addresses
- Street name and number ←→ Host address (IP)
- City, country, Zip code ←→ Network prefix (IP)
- Receiver's name ←→ Port number (TCP)
- Post offices ←→ Internet routers

This example tells us that many of the existing Internet protocols works similar to realizing real-life applications or workflows, and the invoked networking protocols can be analogized in the similar ways.

OUTLINE

- Foundation of Computer Networks
- Addresses
- Physical, Logical, and Overlay Networks
- Computer Networking Services
- IP Network Routing

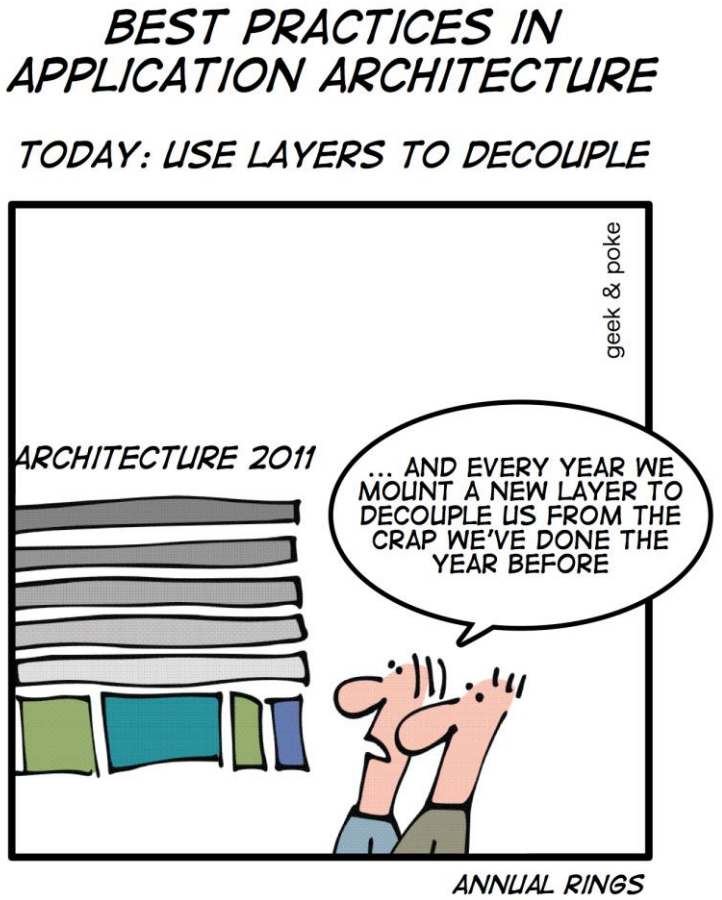
FOUNDATION OF COMPUTER NETWORKS



Protocol Layers, Network Services, Network Interconnections, and Encapsulation

WHAT IS A NETWORK?

- A network consists of two or more computers that are *linked* in order to share resources (such as printers), exchange files, or allow electronic communications.
- The computers on a network may be linked through cables, telephone lines, radio waves, satellites, or infrared light beams.
- Network protocols form a layered-structure



Picture from http://geekandpoke.typepad.com/geekandpoke/2011/03/architectural-best-practices.html

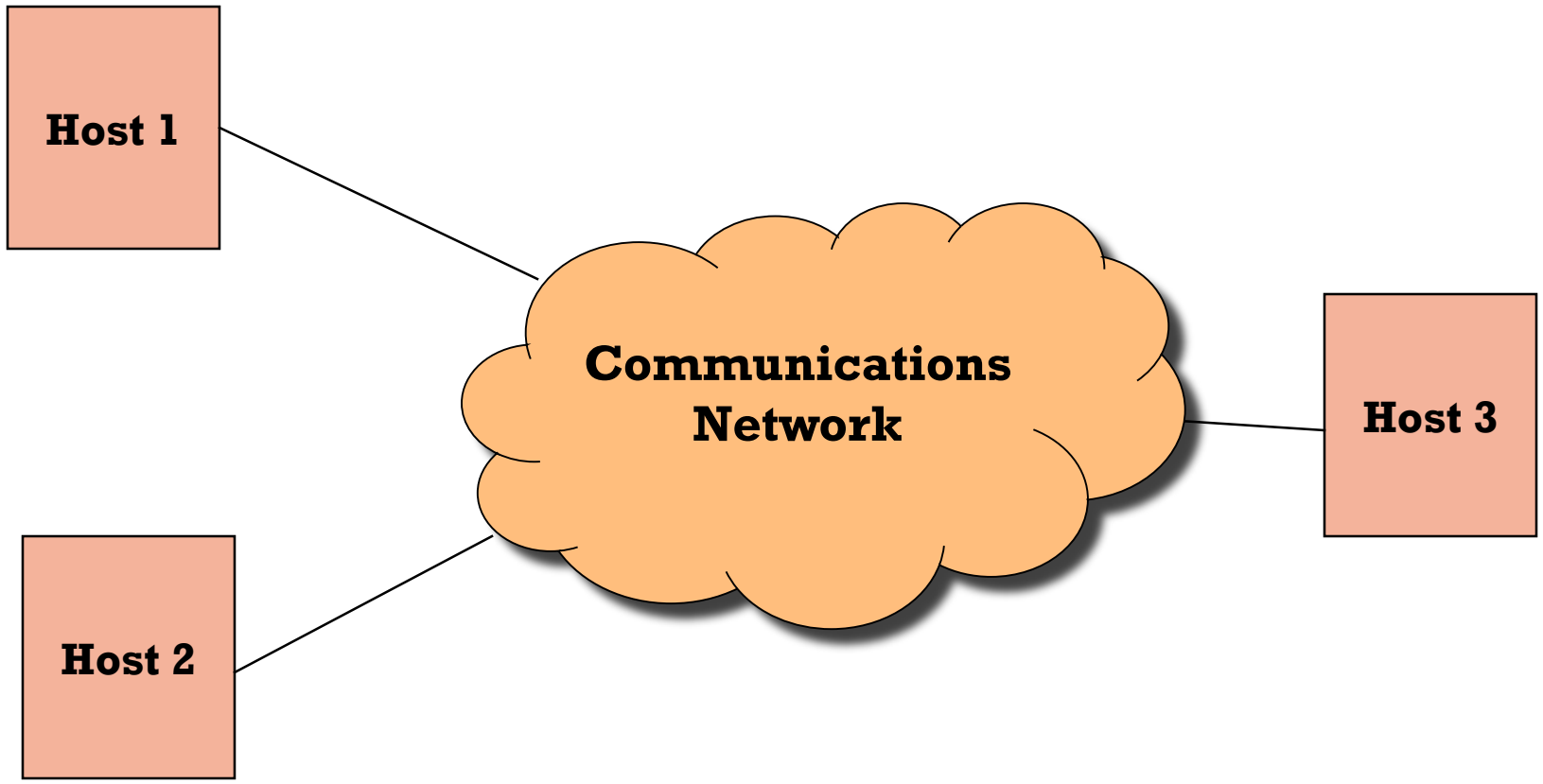
NETWORKING DEVICES

- Hosts
  - Client: a computer that accesses a service made available by a server across a network.
  - Server: a system that responds to requests across a network to provide network services.
- Repeater
  - It just amplifies electronic signals for making them be transferred in a long distance.
- Ethernet Hub
  - A device for connecting multiple Ethernet devices together and making them act as a single network segment (the same broadcast domain and the same collision domain).
- Switch
  - A device that receives a message from any device connected to it and then transmits the message only to the device for which the message was meant. It splits the traffic and sends it to different destinations rather than to all system on the network.

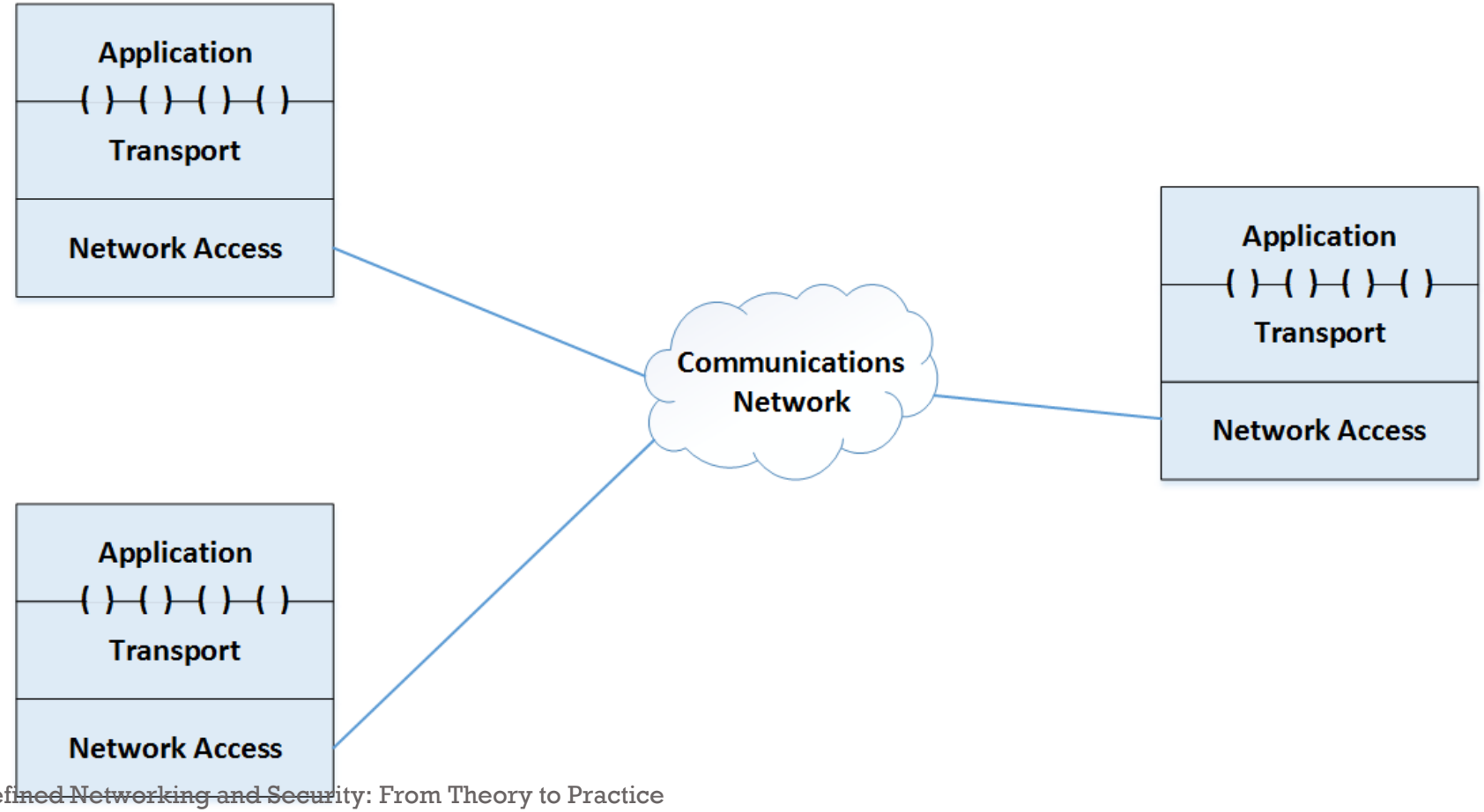
NETWORKING DEVICES (CONT.)

- Bridge
  - A device that connects multiple network segments. Similar to a switch, but bridges can connect different media and standards.
- Router
  - A device that determines the next network point to which it can forward a data packet towards the ultimate destination of the packet. A stop-and-forward device.
- Gateway
  - A device that is placed at a network node and interfaces with another network that uses different protocols. Similar to a router, however, gateways are network points that acts as an entrance to another network.

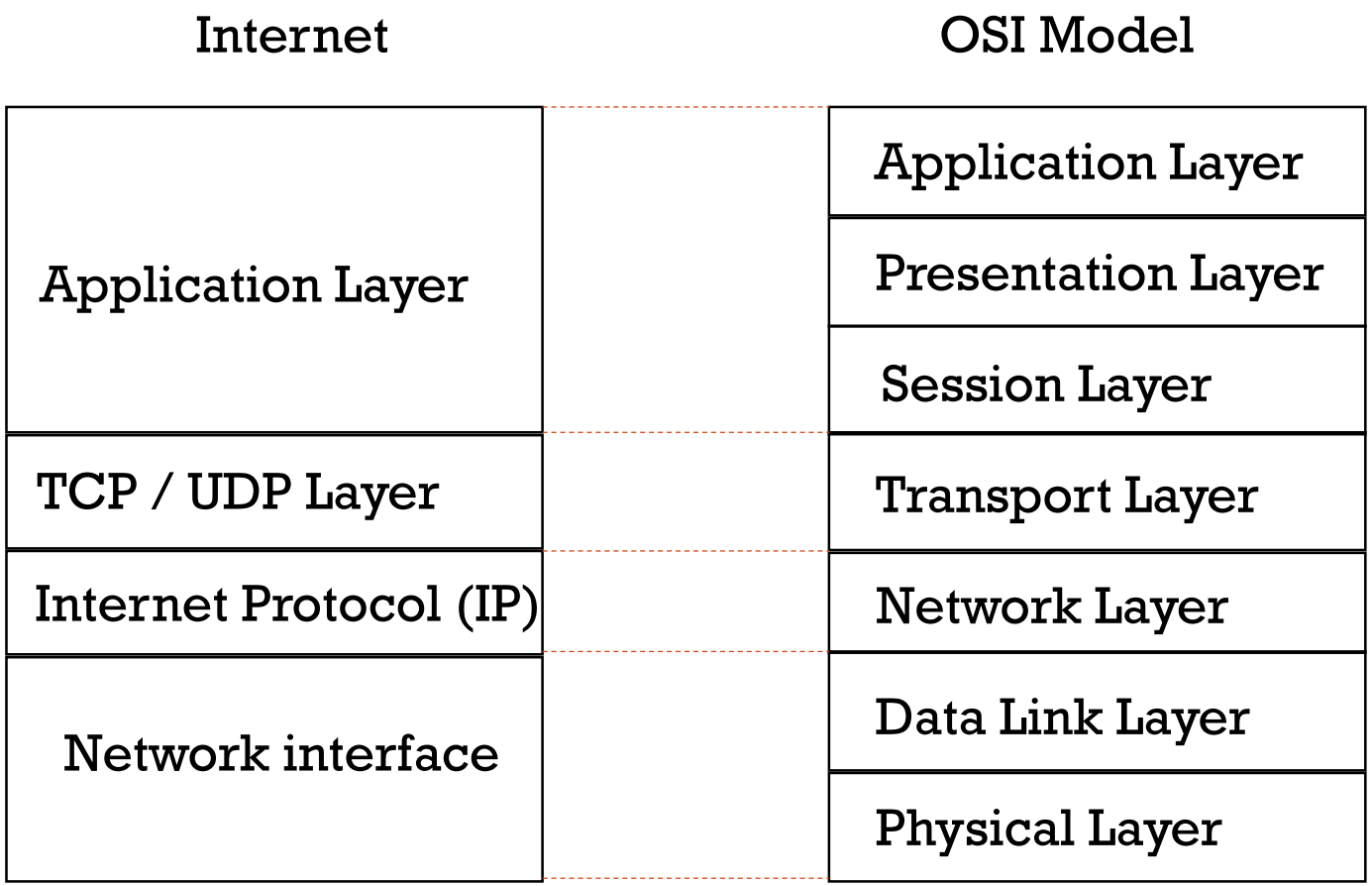
SIMPLE NETWORK VIEW



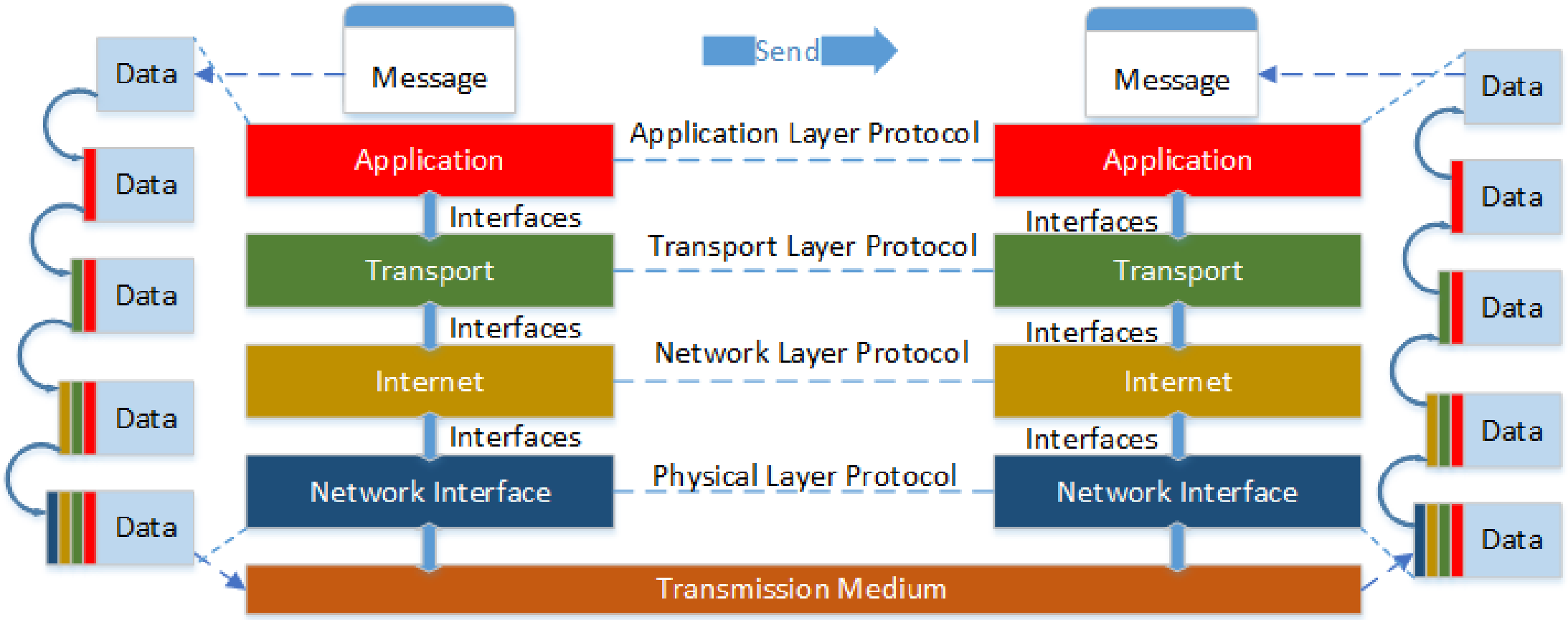
LAYERING VIEW OF A NETWORK



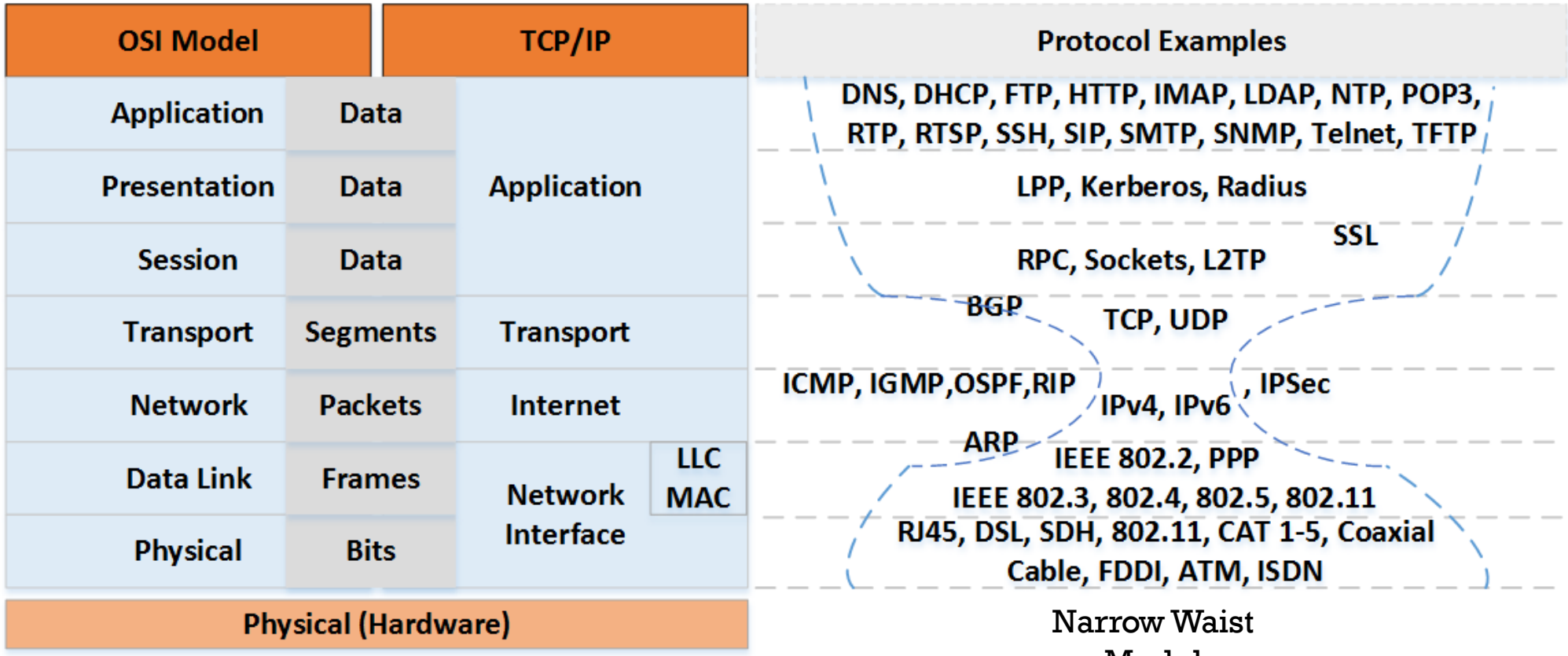
STANDARD LAYERED ARCHITECTURE



NETWORKING SERVICES AND PACKET ENCAPSULATION



PROTOCOL LAYERS-OVERVIEW



NETWORK INTER-CONNECTION

Source Machine

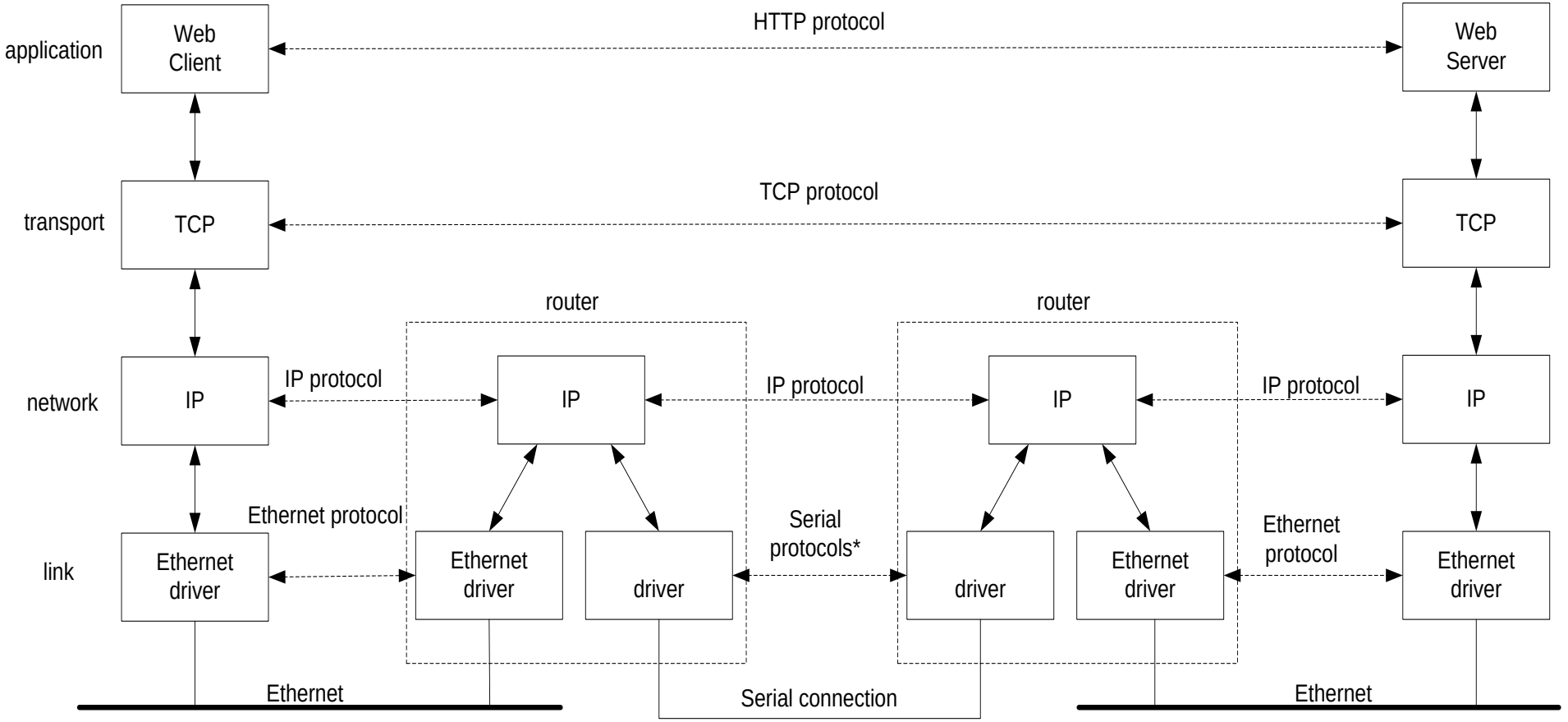
router

router

Destination Machine

- Routers are layer-3 devices
- Routers are used to connect two different types of networks.

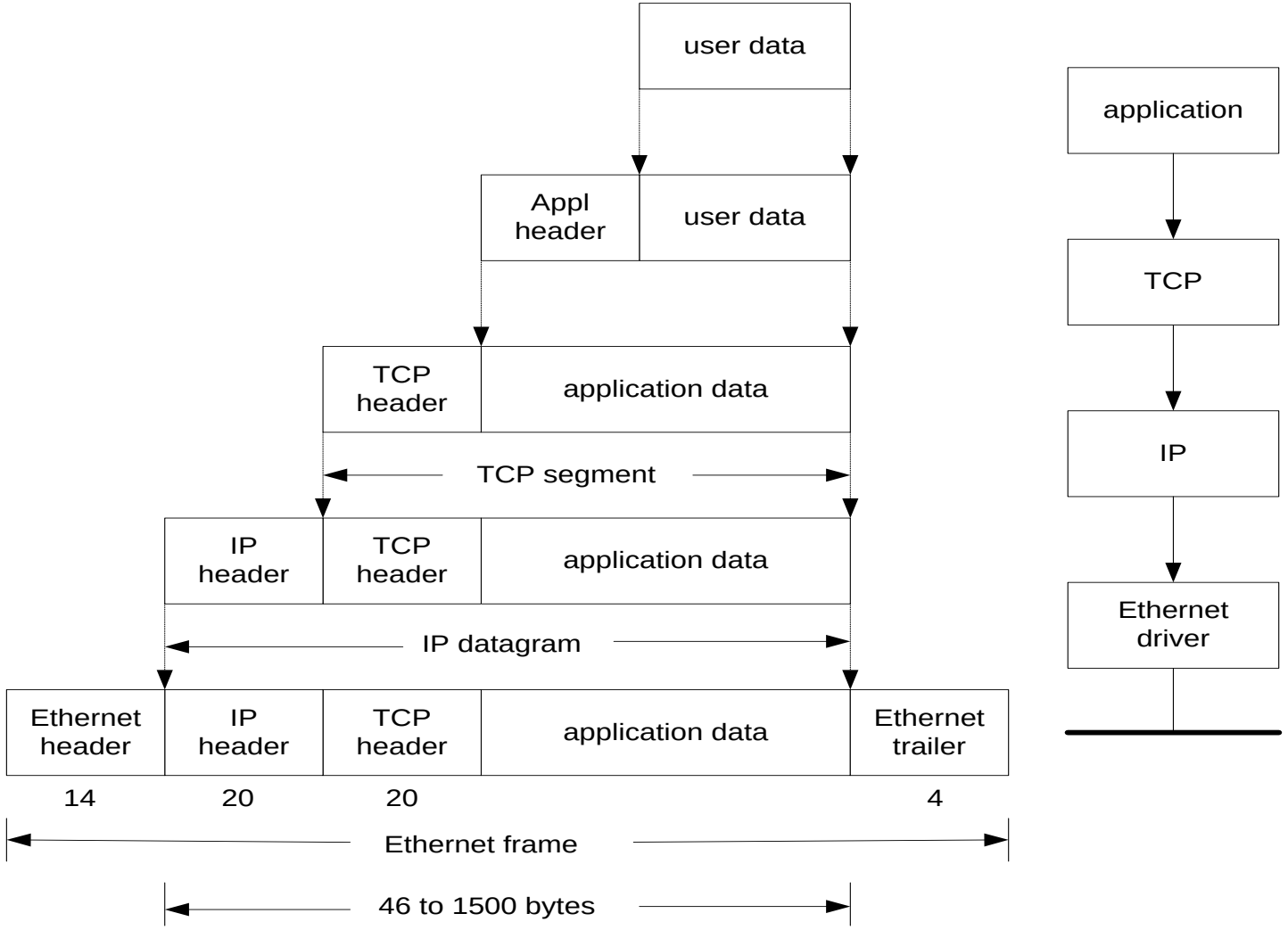
NETWORK INTER-CONNECTION



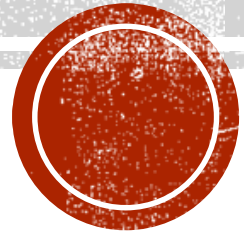
Two networks connected with two routers

* (Cisco support: EIA/TIA-232, EIA/TIA-449, V.35, X.21 and EIA-530 etc.)

ENCAPSULATION

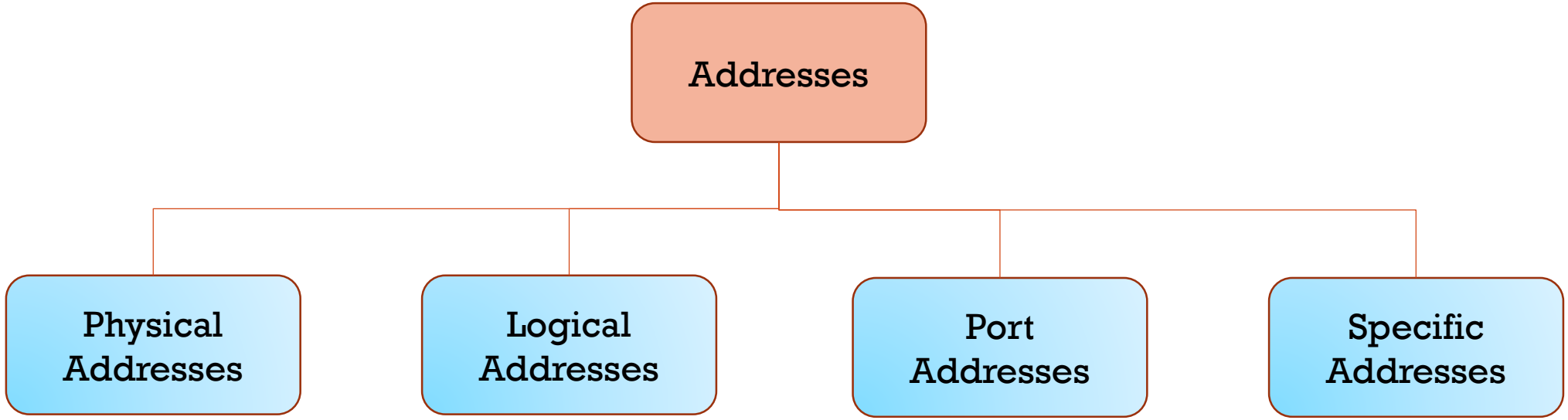


ADDRESS



MAC Address, IP Address, and Port Number

ADDRESS IN COMPUTER AND NETWORKS



E.g., MAC Address

Point-to-Point for directly connected neighbors

E.g., IP Address

End-to-End for the source and destination pair

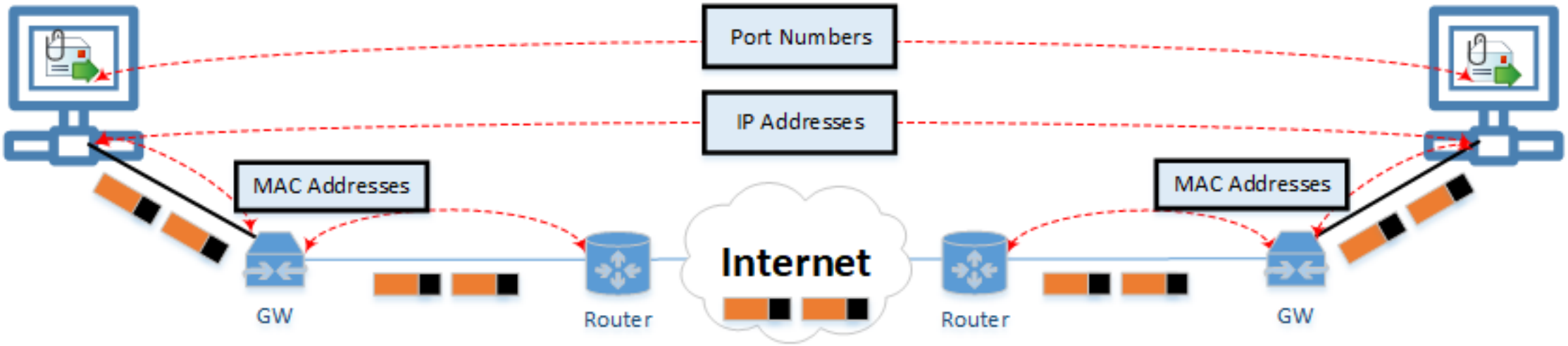
E.g., TCP/UDP Port Number

Identifying the application to send and receive the packet

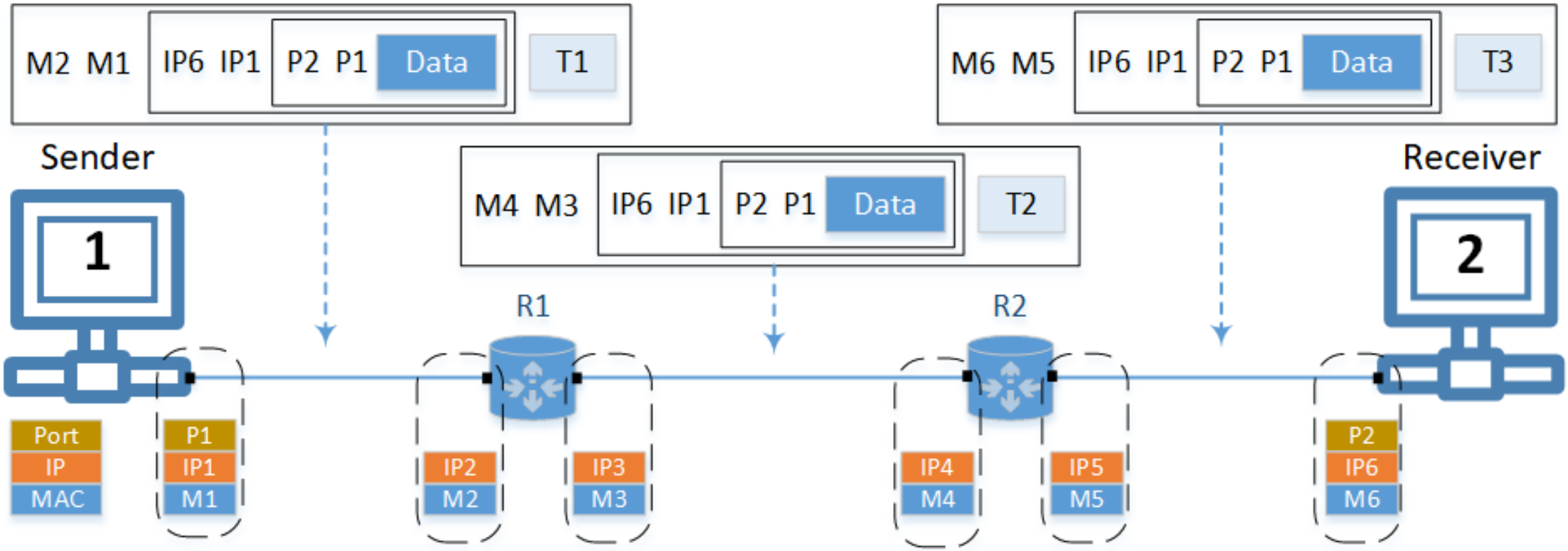
E.g., Process ID

Identifying the function/process in the OS

ADDRESS OVERVIEW



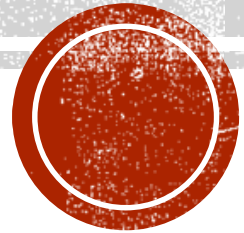
ADDRESSES AND PACKET FORWARDING



T1-T3: trailers of data frame

MAC ADDRESS

Ethernet Frame, MAC address, Link Layer Communication

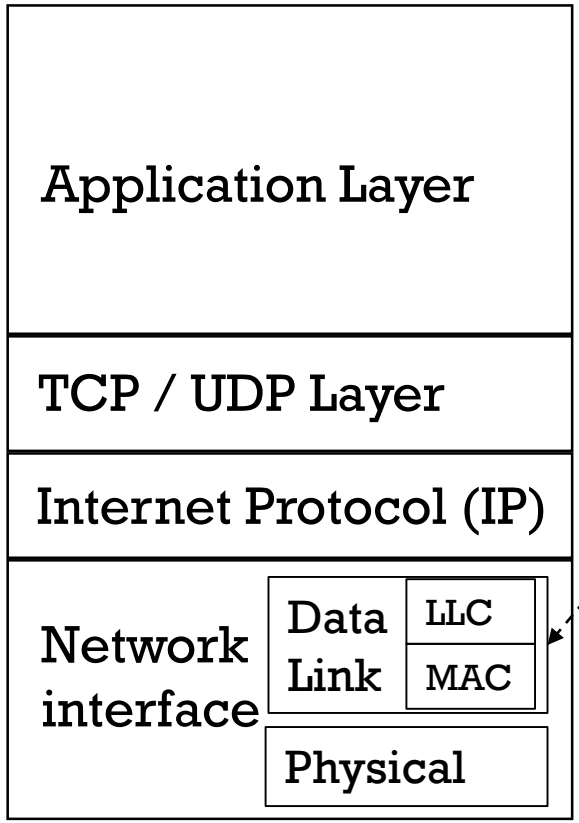


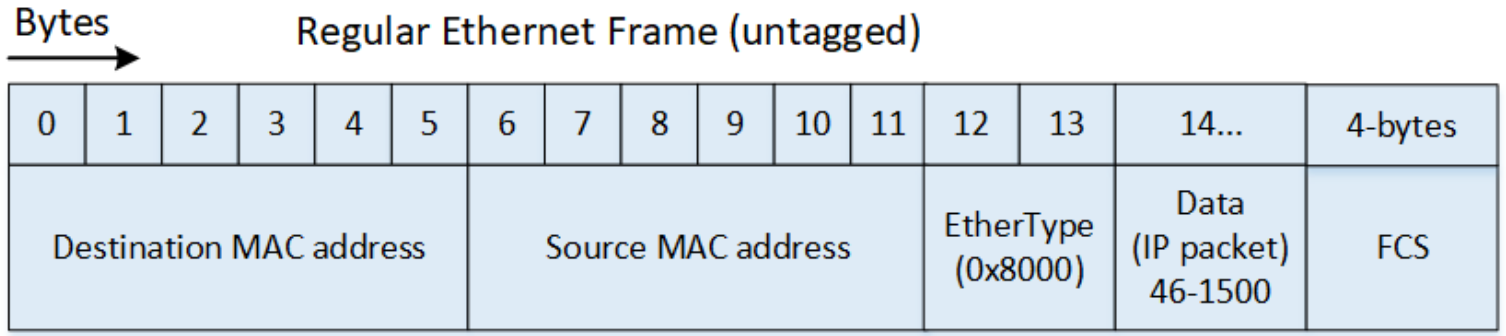
ETHERNET FRAME

802.3 Ethernet frame structure

Preamble	Start of frame delimiter	MAC destination	MAC source	802.1Q tag (optional)	Ethertype (Ethernet II) or length (IEEE 802.3)	Payload	Frame check sequence (32-bit CRC)	Interframe gap
7 octets	1 octet	6 octets	6 octets	(4 octets)	2 octets	42[note 2]–1500 octets	4 octets	12 octets
	← 64–1518 octets (68-1522 octets for 802.1Q tagged frames) →							
← 84–1538 octets (88-1542 octets for 802.1Q tagged frames) →								

DATA FRAME





LLC: Logical Link Control
MAC: Media Access Control
FCS: Frame Check Sequence

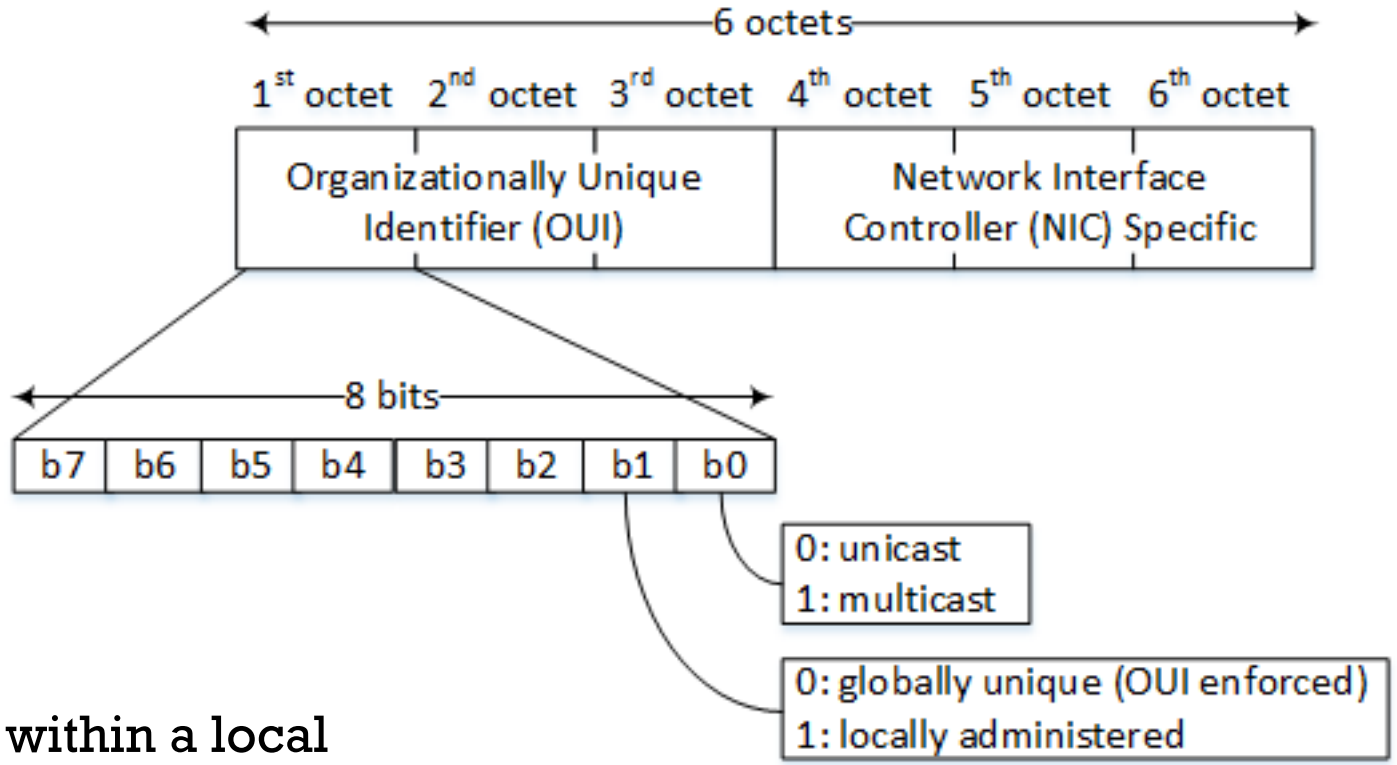
MAC ADDRESS

- 48 bits at the Data link layer

Example:

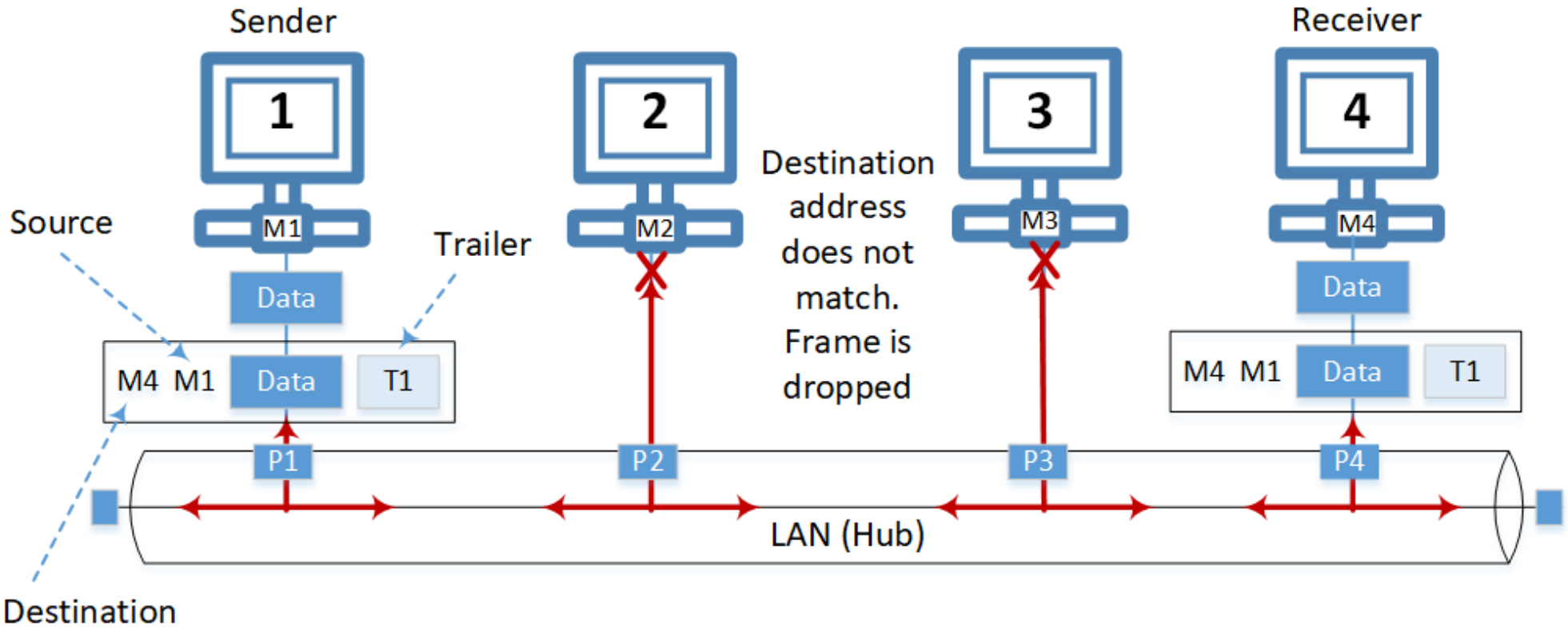
07:01:02:01:2C:4B

A 6-byte (12 hexadecimal digits) physical address.

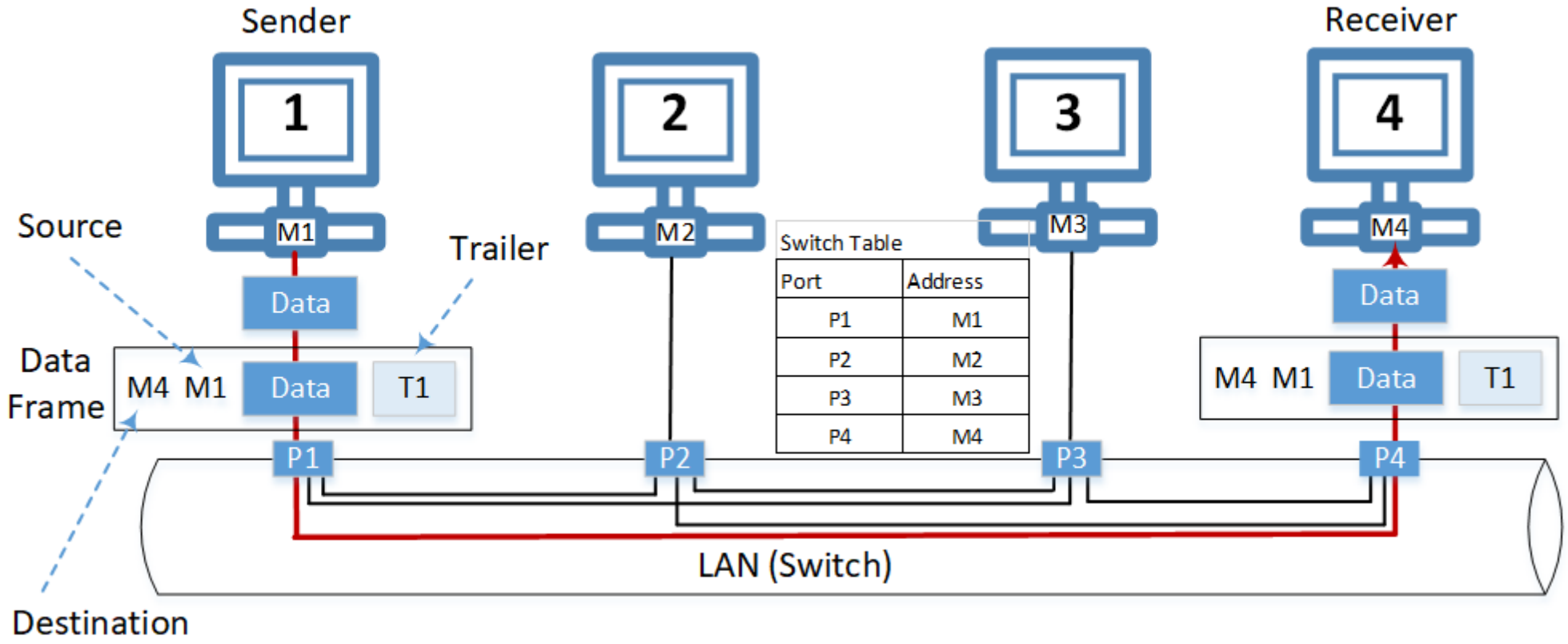


- Devices use MAC addresses within a local networking environment.
- A networking device needs to change its source and destination MAC addresses when forwarding it to the next network segment.

A FRAME IS TRANSMITTED ON A LAN (HUB)

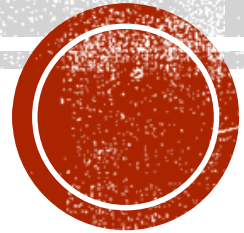


A FRAME IS TRANSMITTED ON A LAN (SWITCH)



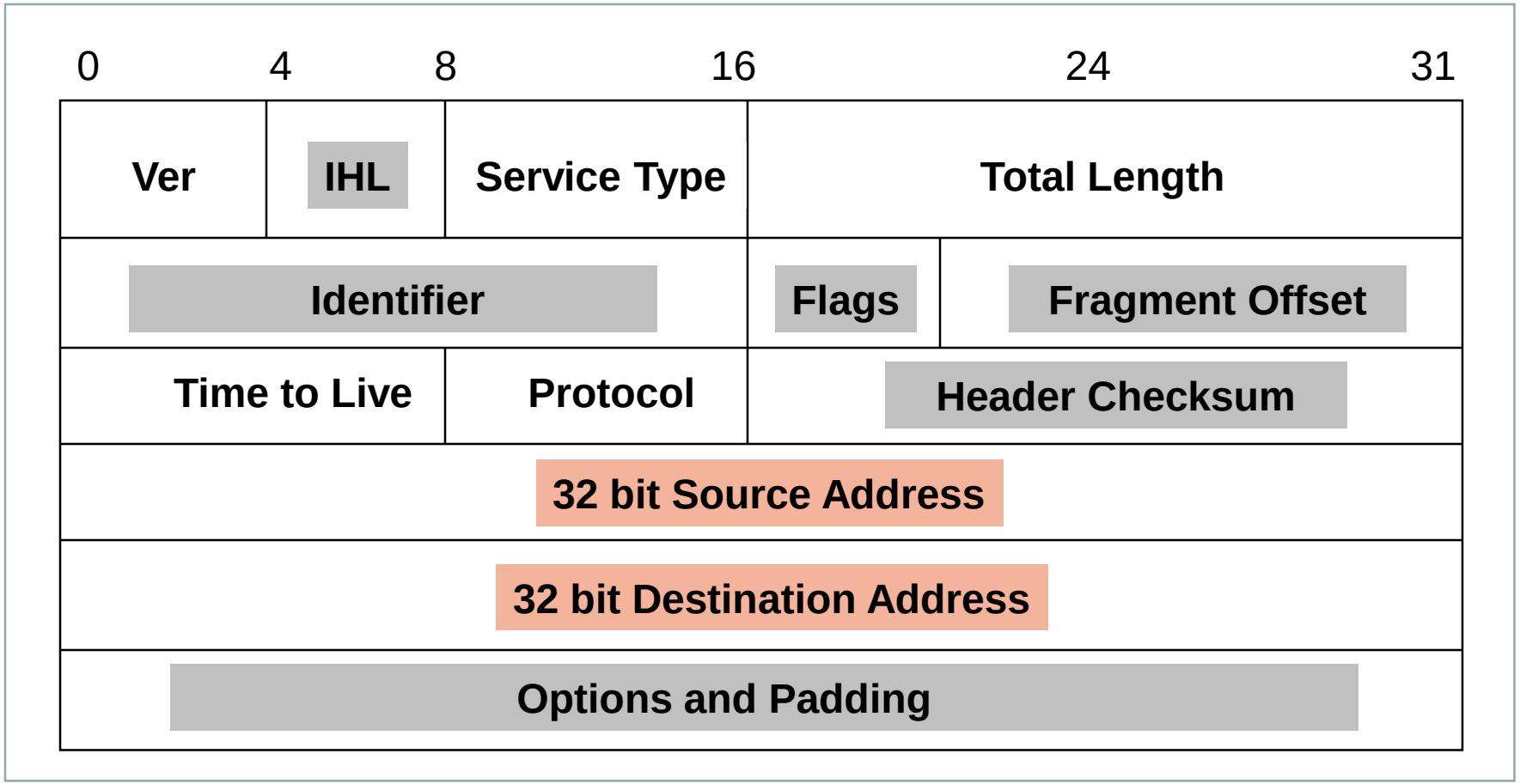
IP ADDRESS

Classful IP Addresses



THE IPV4 HEADER

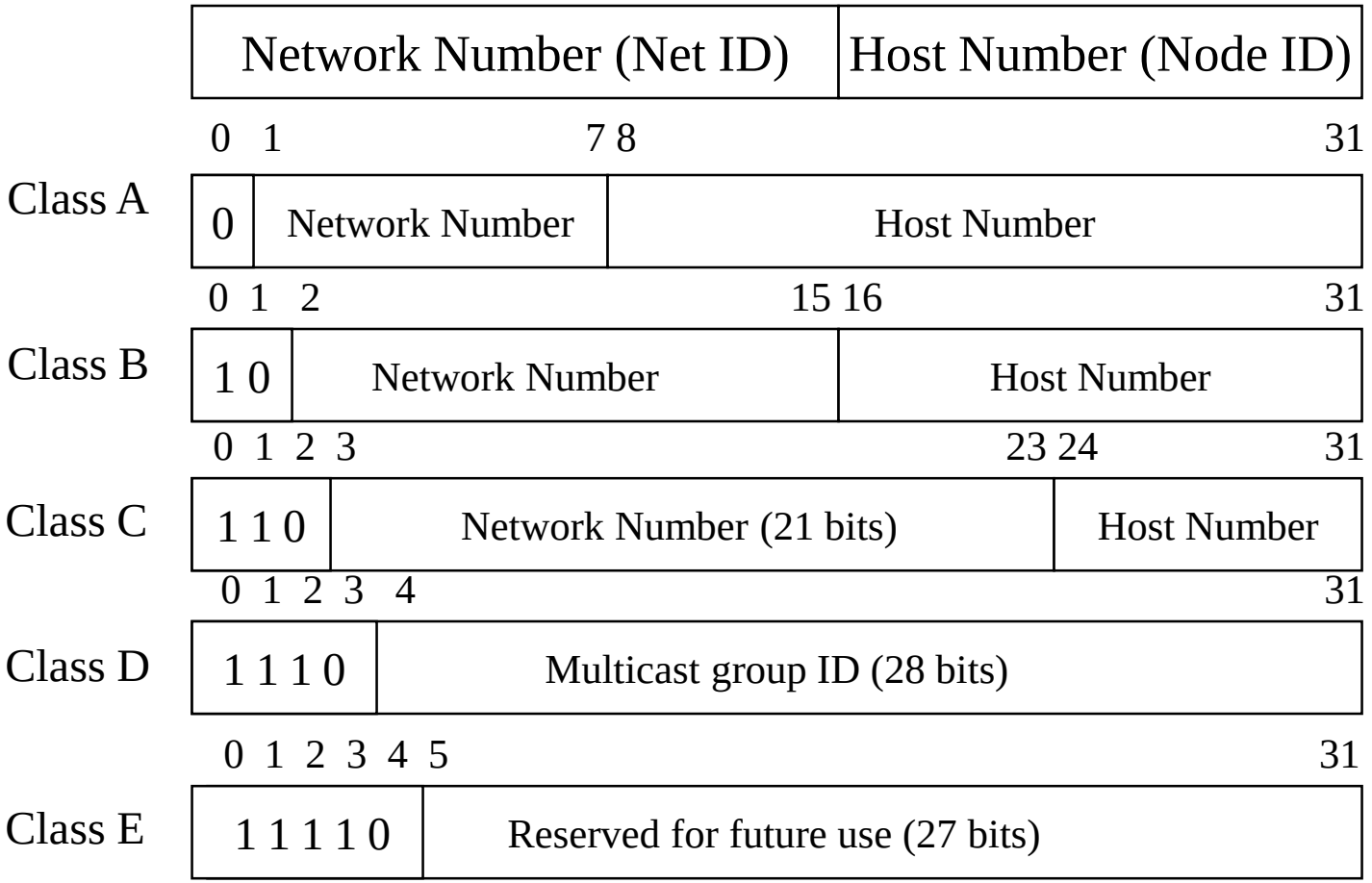
20 OCTETS + OPTIONS : 13 FIELDS, INCLUDING 3 FLAG BITS



shaded fields are absent from IPv6 header

IP ADDRESS (V4) (CLASSFUL IP ADDRESSES)

Two levels IP address (32-bits)



0. 0. 0. 0 → 00000000.00000000.00000000.00000000
127.255.255.255 → 01111111.11111111.11111111.11111111

128. 0. 0. 0 → 10000000.00000000.00000000.00000000
191.255.255.255 → 10111111.11111111.11111111.11111111

192. 0. 0. 0 → 11000000.00000000.00000000.00000000
239.255.255.255 → 11011111.11111111.11111111.11111111

224. 0. 0. 0 → 11100000.00000000.00000000.00000000
239.255.255.255 → 11101111.11111111.11111111.11111111

240. 0. 0. 0 → 11110000.00000000.00000000.00000000
255.255.255.255 → 11111111.11111111.11111111.11111111

IP ADDRESS DOT NOTATION

- Binary presentation: (N-network, n-node).
  - Class A -- NNNNNNNN.nnnnnnnn.nnnnnnnn.nnnnnnnn.
  - Class B -- NNNNNNNN.NNNNNNNN.nnnnnnnn.nnnnnnnn.
  - Class C -- NNNNNNNN.NNNNNNNN.NNNNNNNN.nnnnnnnn.
- Decimal presentation:
  - **Class A (0xxx), 1 ~ 126** in decimal. 2^7 -2 nets and 2^24-2 (=16M) hosts/net
  - **Class B (10xx), 128 ~ 191** in decimal. 2^14 nets and 2^16-2 (=64K) hosts/net
  - **Class C (110x), 192 ~ 223** in decimal. 2^21 nets and 2^8-2 (=254) hosts/net
  - **Class D (1110), 224 ~ 239** in decimal (for multicast).
  - **Class E (1111), 240 ~ 254** in decimal (reserved for future use).

PRIVATE IP ADDRESS (RFC 1918)

There are three IP network addresses reserved for private networks. The addresses are

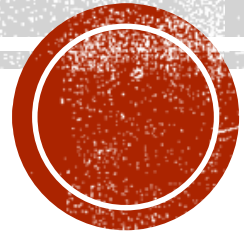
10.0.0.0/8	**One Class A address**
172.16.0.0/12	**16 Class B addresses**
192.168.0.0/16	**256 Class C addresses**

LIMITATIONS OF CLASSFUL IP ADDRESSING

- Designed almost 40 years later
- Only 3 classes of networks
- Address space used *inefficiently*
- For example,
  - If a network needs to support 260 hosts, which class of address it requires? How many addresses waste?

IP ADDRESS

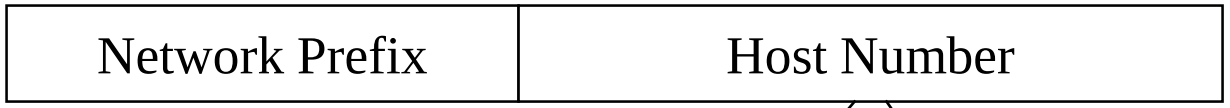
Subnetting



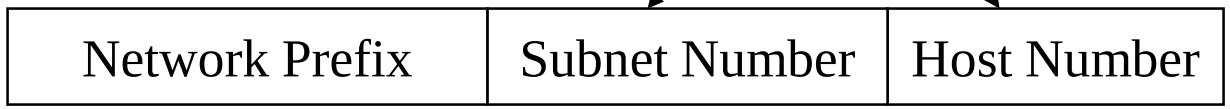
SUBNETTING

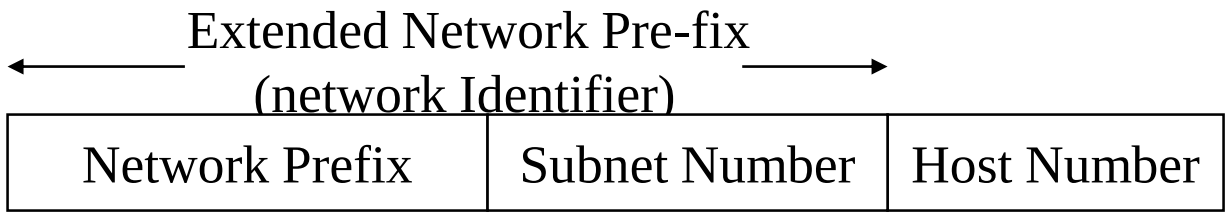
IETF RFC 950:
Create multiple logical networks that exist within a single class.

Two levels classful hierarchy



Three levels subnet hierarchy





SUBNET MASKING

Applying a subnet mask to an IP address allows you to identify the network and node parts of the address. The **network bits** are represented by the **1s** in the mask, and the **node bits** are represented by the **0s**. Performing a bitwise logical AND operation between the IP address and the subnet mask results in the *Network Address* or Identifier.

Example:

```
    10001100.10110011.11110000.11001000   140.179.240.200 Class B IP Address
&)  11111111.11111111.00000000.00000000   255.255.000.000 Default Class B Subnet Mask
    -----------------------------------------------------
    10001100.10110011.00000000.00000000   140.179.000.000 Network Address
```


SUBNETTING PRESENTATION:

- Binary Presentation:
  - **Class A** - 11111111.00000000.00000000.00000000
  - **Class B** - 11111111.11111111.00000000.00000000
  - **Class C** - 11111111.11111111.11111111.00000000

- Decimal Presentation:
  - **Class A** - 255.0.0.0
  - **Class B** - 255.255.0.0
  - **Class C** - 255.255.255.0

- “/” or Length presentation (CIDR notation)
  - **Class A** - /8
  - **Class B** - /16
  - **Class C** - /24

EXTENDED NETWORK PREFIX – SUBNET MASKS

A subnet address cannot be all "0"s or all "1"s.

This also implies that a 1 bit subnet mask is not allowed

Example:

← Extended Network Prefix →

← Network Prefix →⟷

10001100.10110011.11011100.11001000	140.179.220.200 IP Address
11111111.11111111.**111**00000.00000000	255.255.**224**.000 Subnet Mask
10001100.10110011.110**00000.00000000**	140.179.192.000 Subnet Address
10001100.10110011.11011111.11111111	140.179.223.255 Broadcast Address

140.179.220.200/19

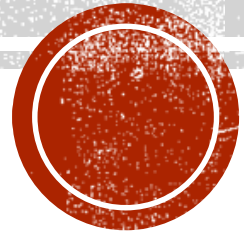
How many **subnets** do we have from above example?

AN EXAMPLE: 200.133.175.X/28

Subnet bits	Network Number	Node Addresses	Broadcast Address
0000	200.133.175.0	.1 thru .14	200.133.175.15
0001	200.133.175.16	.17 thru .30	200.133.175.31
0010	200.133.175.32	.33 thru .46	200.133.175.47
0011	200.133.175.48	.49 thru .62	200.133.175.63
0100	200.133.175.64	.65 thru .78	200.133.175.79
0101	200.133.175.80	.81 thru .94	200.133.175.95
0110	200.133.175.96	.97 thru .110	200.133.175.111
0111	200.133.175.112	.113 thru .126	200.133.175.127
1000	200.133.175.128	.129 thru .142	200.133.175.143
1001	200.133.175.144	.145 thru .158	200.133.175.159
1010	200.133.175.160	.161 thru .174	200.133.175.175
1011	200.133.175.176	.177 thru .190	200.133.175.191
1100	200.133.175.192	.193 thru .206	200.133.175.207
1101	200.133.175.208	.209 thru .222	200.133.175.223
1110	200.133.175.224	.225 thru .238	200.133.175.239
1111	200.133.175.240	.241 thru .254	200.133.175.255

IP ADDRESS

IP Network Address Management & Design

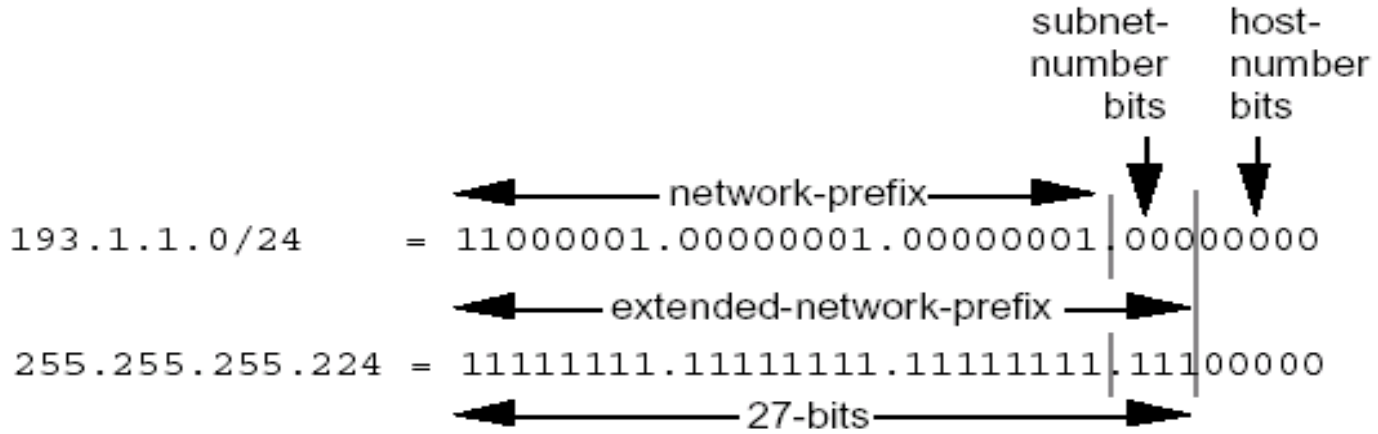


SUBNET DESIGN CONSIDERATIONS

1. How many *total subnets* does the organization need today?
2. How many total subnets will the organization need in the future?
3. How many *hosts* are there on the organization's largest subnet today?
4. How many hosts will there be on the organization's largest subnet in the future?

DESIGN EXAMPLES

Subnet Example #1
Given
An organization has been assigned the network number 193.1.1.0/24 and it needs to define 6 subnets. The largest subnet is required to support 25 hosts.



DESIGN EXAMPLES - CONTINUED

Base Net: 11000001.00000001.00000001.00000000 = 193.1.1.0/24

Subnet #0: 11000001.00000001.00000001.00000000 = 193.1.1.0/27
Subnet #1: 11000001.00000001.00000001.00100000 = 193.1.1.32/27
Subnet #2: 11000001.00000001.00000001.01000000 = 193.1.1.64/27
Subnet #3: 11000001.00000001.00000001.01100000 = 193.1.1.96/27
Subnet #4: 11000001.00000001.00000001.10000000 = 193.1.1.128/27
Subnet #5: 11000001.00000001.00000001.10100000 = 193.1.1.160/27
Subnet #6: 11000001.00000001.00000001.11000000 = 193.1.1.192/27
Subnet #7: 11000001.00000001.00000001.11100000 = 193.1.1.224/27

DEFINING HOST ADDRESS FOR EACH SUBNET

- The all-1s host-number and all 0s host-number are both reserved.
- In our current example, there are 5 bits in the host-number field of each subnet address. This means that each subnet represents a block of 30 host addresses. The hosts on each subnet are numbered 1 through 30.

DEFINING HOST ADDRESS FOR EACH SUBNET

Subnet #2: 11000001.00000001.00000001.010 00000 = 193.1.1.64/27
Host #1: 11000001.00000001.00000001.010 00001 = 193.1.1.65/27
Host #2: 11000001.00000001.00000001.010 00010 = 193.1.1.66/27
Host #3: 11000001.00000001.00000001.010 00011 = 193.1.1.67/27
Host #4: 11000001.00000001.00000001.010 00100 = 193.1.1.68/27
Host #5: 11000001.00000001.00000001.010 00101 = 193.1.1.69/27

.
.

Host #15: 11000001.00000001.00000001.010 01111 = 193.1.1.79/27
Host #16: 11000001.00000001.00000001.010 10000 = 193.1.1.80/27

.
.

Host #27: 11000001.00000001.00000001.010 11011 = 193.1.1.91/27
Host #28: 11000001.00000001.00000001.010 11100 = 193.1.1.92/27
Host #29: 11000001.00000001.00000001.010 11101 = 193.1.1.93/27
Host #30: 11000001.00000001.00000001.010 11110 = 193.1.1.94/27

Defining the Broadcast Address for Subnet #2
The broadcast address for Subnet #2 is the all 1's host address or:
11000001.00000001.00000001.010 11111 = 193.1.1.95

DESIGN EXAMPLES

Subnet Example #2
Given
An organization has been assigned the network number 140.25.0.0/16 and it needs to create a set of subnets that supports up to 60 hosts on each subnet.

140.25.0.0/16
10001100.00011001.00000000.00000000/16

How many subnets can be created?

DESIGN EXAMPLES - CONTINUED

Subnet Example #2

Given

An organization has been assigned the network number 140.25.0.0/16 and it needs to create a set of subnets that supports up to 60 hosts on each subnet.

140.25.0.0/26

10001100.00011001.00000000.00000000/26

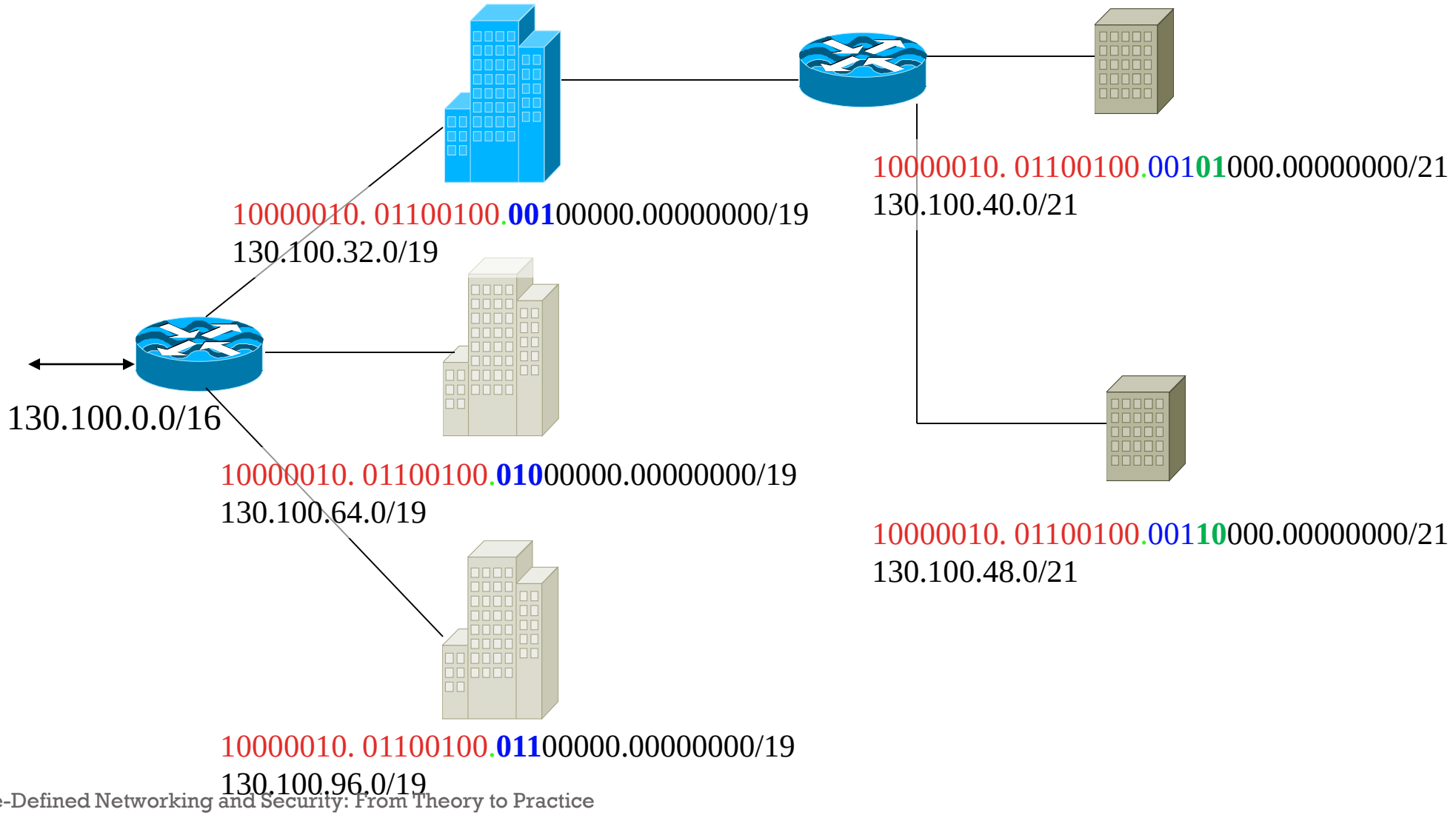
subnet #1, host 1

10001100.00011001.00000000.01000001/26

etc.

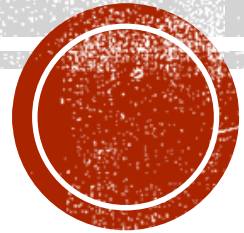
How many hosts for each subnet?

AN IP ADDRESS ALLOCATION EXAMPLE



IP ADDRESS

CIDR-Classless Interdomain Routing



PROS AND CONS OF SUBNETTING

- Advantages
  - Assign IP address more efficiently
  - Speeds up the network (reduce the size of broadcast domain)
  - Easy to organize the network resources
  - Improve security
- Disadvantages
  - Added layer of complexity (more routers)
  - Difficult to change once hierarchy is established.

CIDR -- CLASSLESS INTERDOMAIN ROUTING

CIDR was officially documented in September 1993 in RFCs 1517, 1518, 1519, and 1520.

CIDR :

- Breaks the rigid boundaries between class A, B, C
  - Allocate IPv4 address more efficiently
- Supports Route Aggregation
  - Single routing entry in the routing table to specify how to route traffic to many individual network addresses.

CIDR -- CLASSLESS INTERDOMAIN ROUTING

Superneting Example:

192.60.128.0/24	(11000000.00111100.1000000**00**.00000000)	Class C address
192.60.129.0/24	(11000000.00111100.1000000**01**.00000000)	Class C address
192.60.130.0/24	(11000000.00111100.1000000**10**.00000000)	Class C address
192.60.131.0/24	(11000000.00111100.1000000**11**.00000000)	Class C address
---	---	---
192.60.128.0/22	(11000000.00111100.10000000.00000000)	Supernet
255.255.252.0	(11111111.11111111.11111100.00000000)	Supernet Mask
192.60.131.255	(11000000.00111100.10000011.11111111)	Broadcast address

So classful addresses can easily be written in CIDR notation (Class A = /8, Class B = /16, and Class C = /24)

CIDR ADDRESS BLOCKS EXAMPLE

CIDR	Dotted-Decimal mask	# of addresses	# of Classful Networks
/13	255.248.0.0	512K	8 Bs or 2048 Cs
/14	255.252.0.0	256K	4 Bs or 1024 Cs
/15	255.254.0.0	128K	2 Bs or 512 Cs
/16	255.255.0.0	64K	1 B or 256 Cs
/17	255.255.128.0.0	32K	128 Cs
/18	255.255.192.0.0	16K	64 Cs
/19	255.255.224.0	8K	32 Cs
/20	255.255.240.0	4K	16 Cs
…	…	…	…
/24	255.255.255.0	256	1 C
/25	255.255.255.128	128	½ C

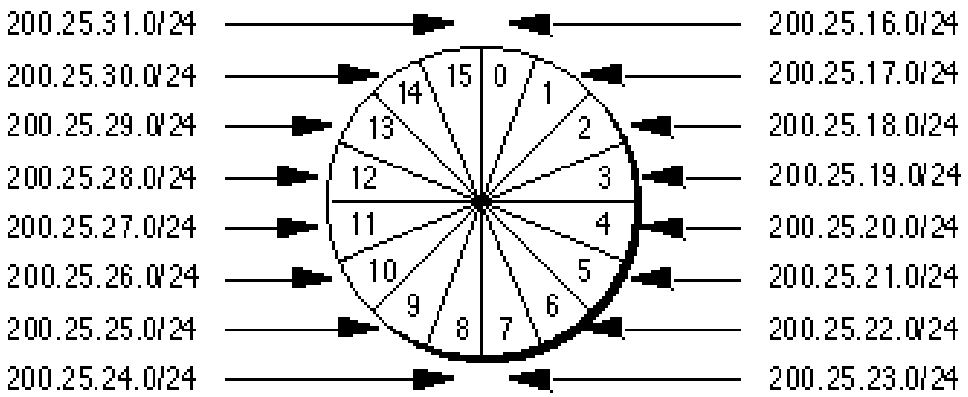
CIDR ADDRESS ALLOCATION EXAMPLE

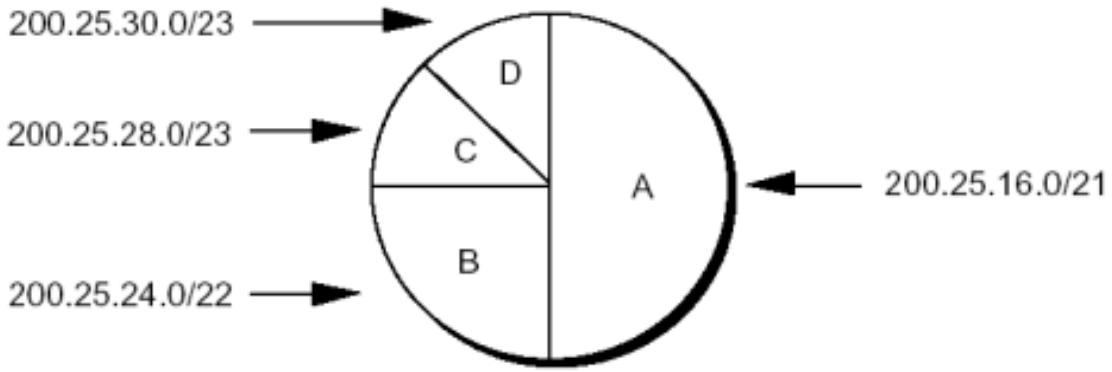
Assume that an ISP owns the address block 200.25.0.0/16.

- This block represents 65, 536 (2^16) IP addresses (or 256 /24s).

If the ISP wants to allocate the 200.25.16.0/20 address block.

- In a classful environment, this smaller block represents 4,096 (2^12) IP addresses (or 16 /24s).
- In a classless environment, the ISP is free to cut up the pie any way it wants.
    - One-half of the address space for Organization A
    - One-fourth of the address space for Organization B
    - One-eight of the address space for Organization C and Organization D





CIDR EXAMPLE - CONTINUED

Step #1: Divide the address block 200.25.16.0/20 into two equal size slices.
Each block represents one-half of the address space or 2,048 (2^11) IP addresses.

ISP's Block 11001000.00011001.00010000.00000000 200.25.16.0/20
Org A: 11001000.00011001.00010000.00000000 200.25.16.0/21
Reserved: 11001000.00011001.00011000.00000000 200.25.24.0/21

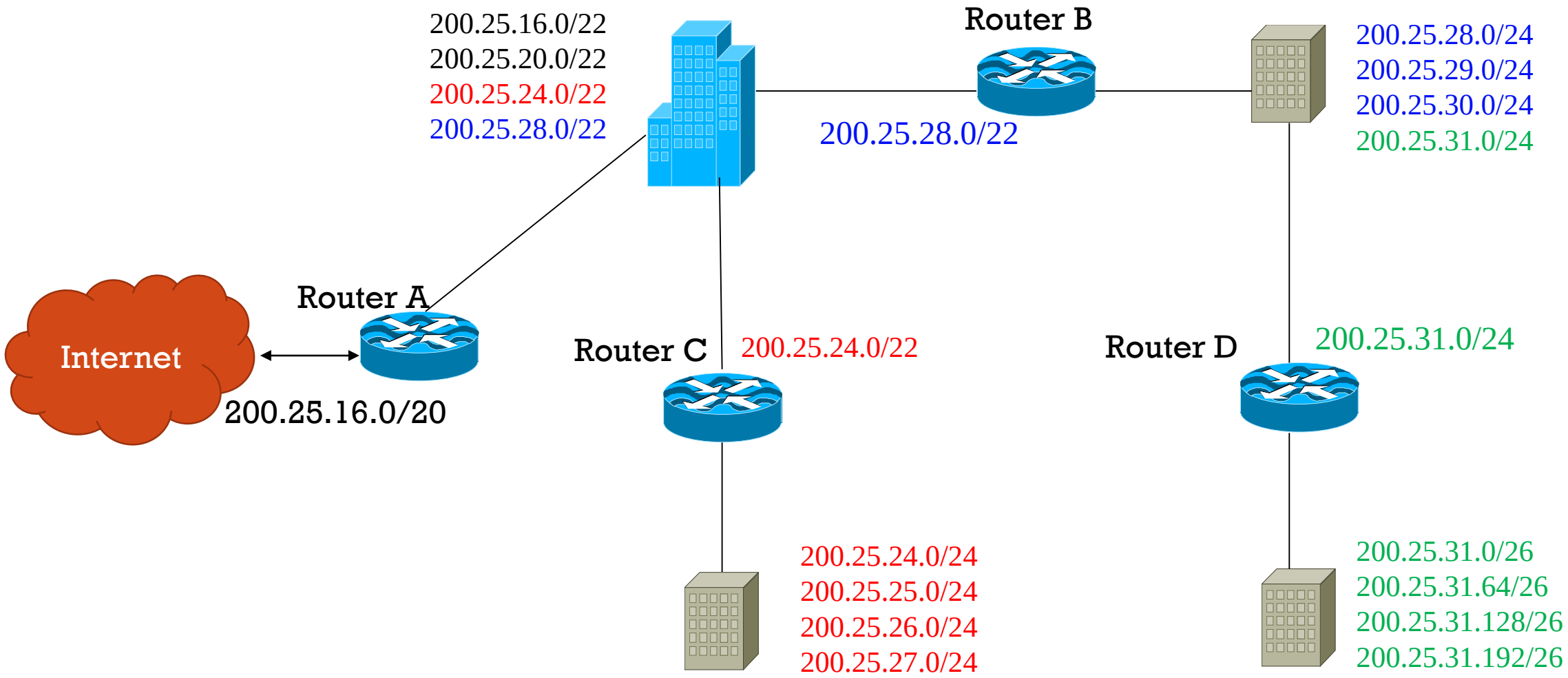
Step #2: Divide the reserved block (200.25.24.0/21) into two equal size slices.
Each block represents one-fourth of the address space or 1,024 (2^10) IP addresses.

Reserved 11001000.00011001.00011000.00000000 200.25.24.0/21
Org B: 11001000.00011001.00011000.00000000 200.25.24.0/22
Reserved 11001000.00011001.00011100.00000000 200.25.28.0/22

Step #3: Divide the reserved address block (200.25.28.0/22) into two equal size blocks.
Each block represents one-eight of the address space or 512 (2^9) IP addresses.

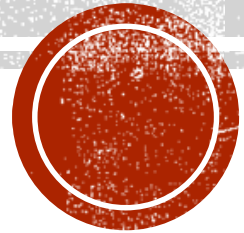
Reserved 11001000.00011001.00011100.00000000 200.25.28.0/22
Org C: 11001000.00011001.00011100.00000000 200.25.28.0/23
Org D: 11001000.00011001.00011110.00000000 200.25.30.0/23

ROUTE AGGREGATION

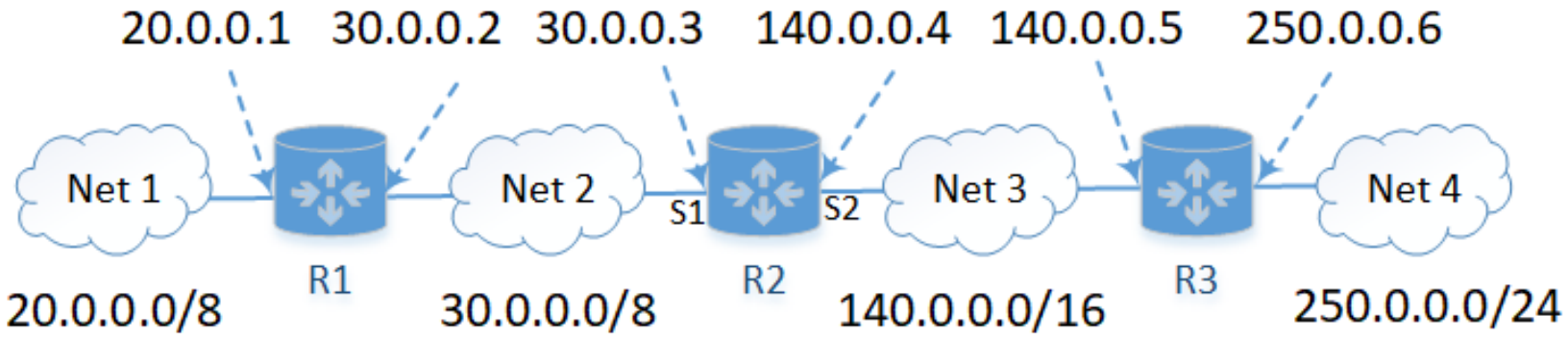


IP ADDRESS

IP Forwarding



IP FORWARDING

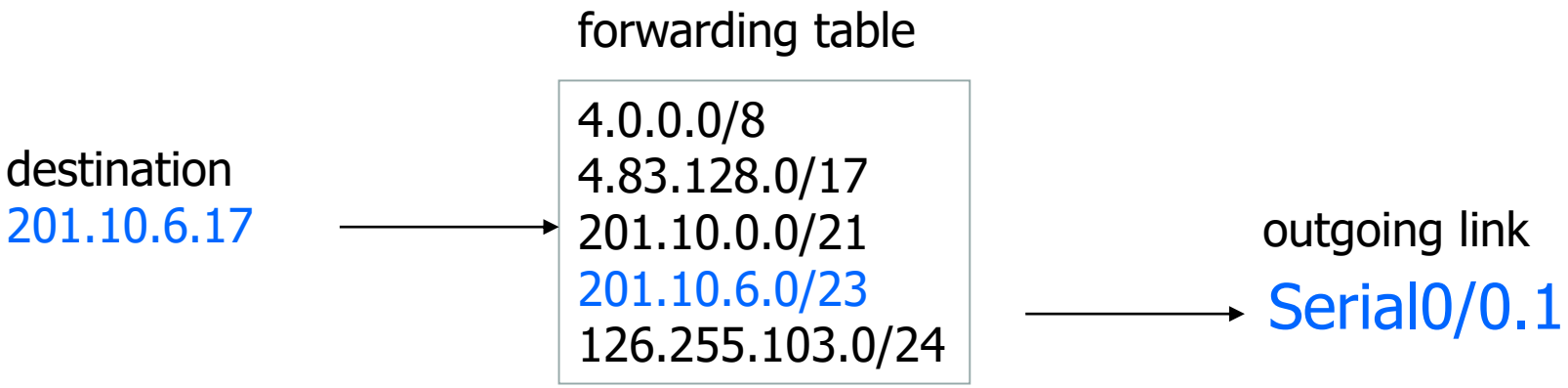


Destination	Network Mask	Next Hop
20.0.0.0	255.0.0.0	30.0.0.2
30.0.0.0	255.0.0.0	S1
140.0.0.0	255.255.0.0	S2
250.0.0.0	255.255.255.0	140.0.0.5

(Routing Table of R2)

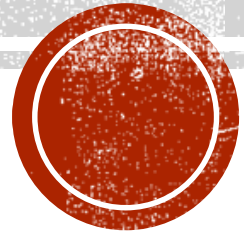
LONGEST PREFIX MATCH FORWARDING

- Forwarding tables in IP routers
  - Maps each IP prefix to next-hop link(s)
- Destination-based forwarding
  - Packet has a destination address in the IP Header
  - Router identifies longest-matching prefix



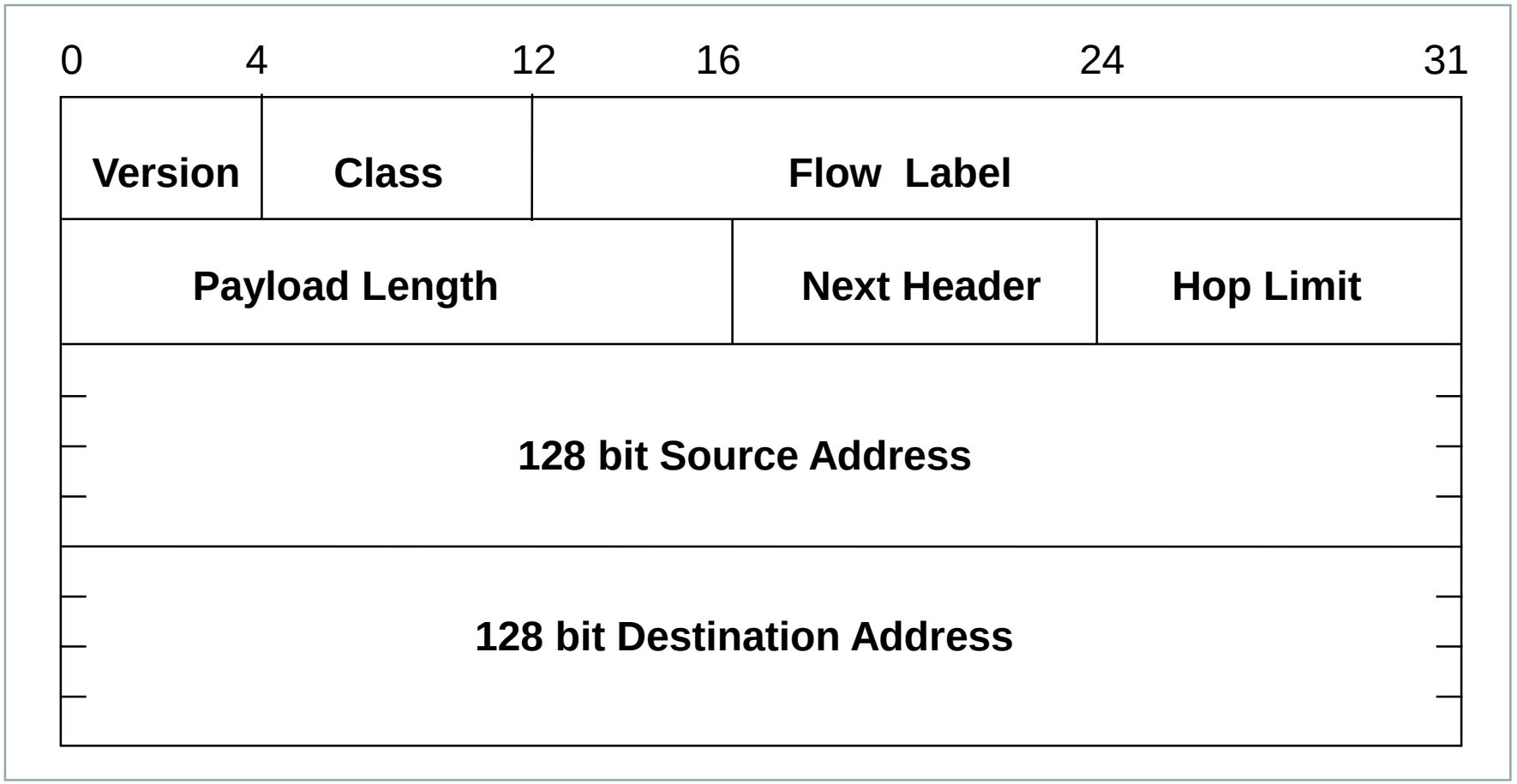
IPV6

IP Version 6



THE IPV6 HEADER

40 OCTETS, 8 FIELDS



TEXT REPRESENTATION OF ADDRESSES

"Preferred" form:
1080:0:FF:0:8:800:200C:417A

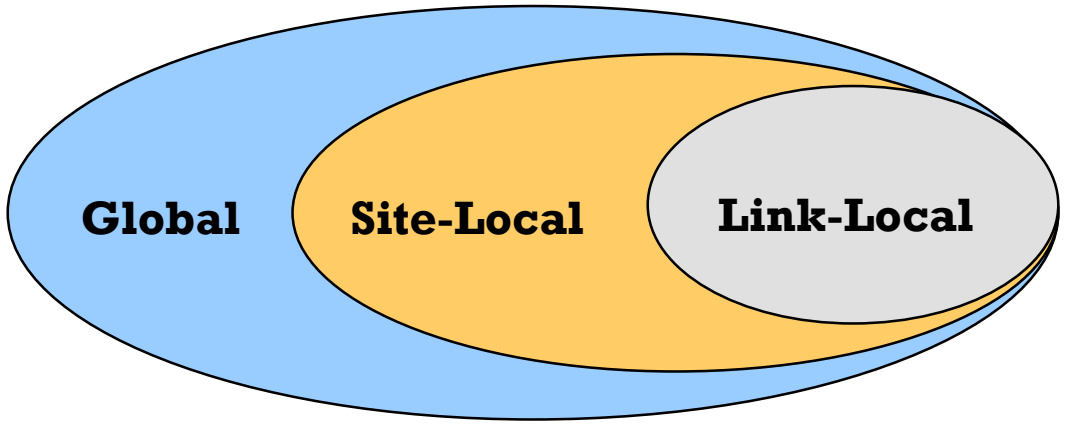
Compressed form: FF01:0:0:0:0:0:0:43
becomes FF01::43

IPv4-compatible: 0:0:0:0:0:0:13.1.68.3
or ::13.1.68.3

IPV6 - ADDRESSING MODEL

- Addresses are assigned to interfaces
  - No change from IPv4 Model
- Interface ‘expected’ to have multiple addresses

- Addresses have scope
  - Link Local
  - Site Local
  - Global



- Addresses have lifetime
  - Valid and Preferred lifetime

TYPES OF IPV6 ADDRESSES

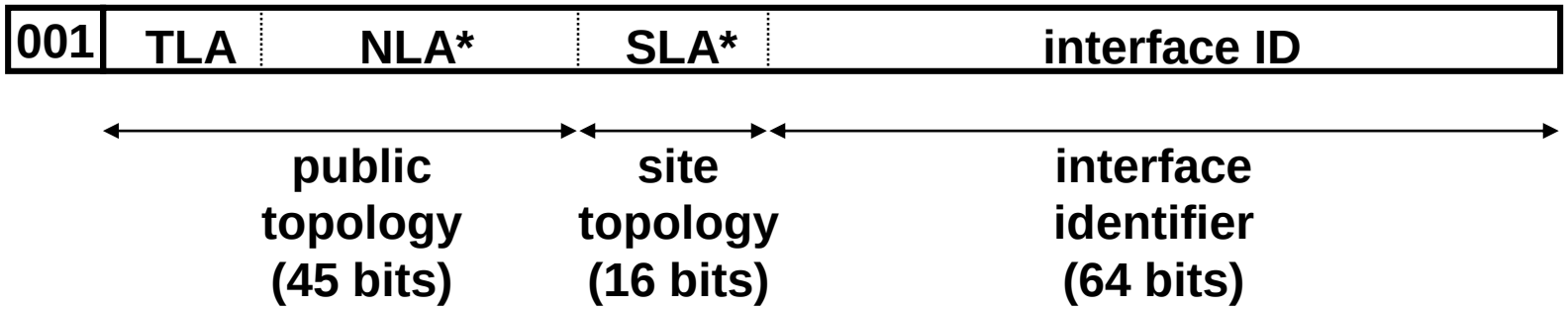
- Unicast
  - Address of a single interface
  - Delivery to single interface
- Multicast
  - Address of a set of interfaces
  - Delivery to all interfaces in the set
- Anycast
  - Address of a set of interfaces
  - Delivery to a single interface in the set
- No more broadcast addresses

ADDRESS TYPE PREFIXES

Address type	Binary prefix
IPv4-compatible	0000…0 (96 zero bits)
global unicast	001
link-local unicast	1111 1110 10
site-local unicast	1111 1110 11
multicast	1111 1111

- all other prefixes reserved (approx. 7/8ths of total)
- anycast addresses allocated from unicast prefixes

GLOBAL UNICAST ADDRESSES



- TLA = Top-Level Aggregator
  NLA* = Next-Level Aggregator(s)
  SLA* = Site-Level Aggregator(s)
- all subfields variable-length, non-self-encoding (like CIDR)
- TLAs may be assigned to providers or exchanges

LINK-LOCAL & SITE-LOCAL UNICAST ADDRESSES

Link-local addresses for use during auto-configuration and when no routers are present:

Site-local addresses for independence from changes of TLA / NLA*:

SOME SPECIAL-PURPOSE UNICAST ADDRESSES

- The unspecified address, used as a placeholder when no address is available:

  0:0:0:0:0:0:0:0

- The loopback address, for sending packets to self:

  0:0:0:0:0:0:0:1

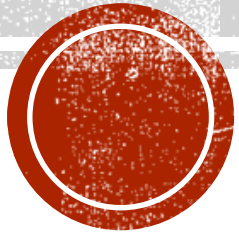
MULTICAST ADDRESS FORMAT

FP (8bits)	Flags (4bits)	Scope (4bits)	RESERVED (80bits)	Group ID (32bits)
11111111	000T	Lcl/Sit/Gbl	MUST be 0	Locally administered

- flag field
  - **low-order bit indicates permanent/transient group**
  - **(three other flags reserved)**
- scope field:
  - **1 - node local**
  - **2 - link-local**
  - **5 - site-local**
  - **8 - organization-local**
  - **B - community-local**
  - **E - global**
  - **(all other values reserved)**
- map IPv6 multicast addresses directly into low order 32 bits of the IEEE 802 MAC

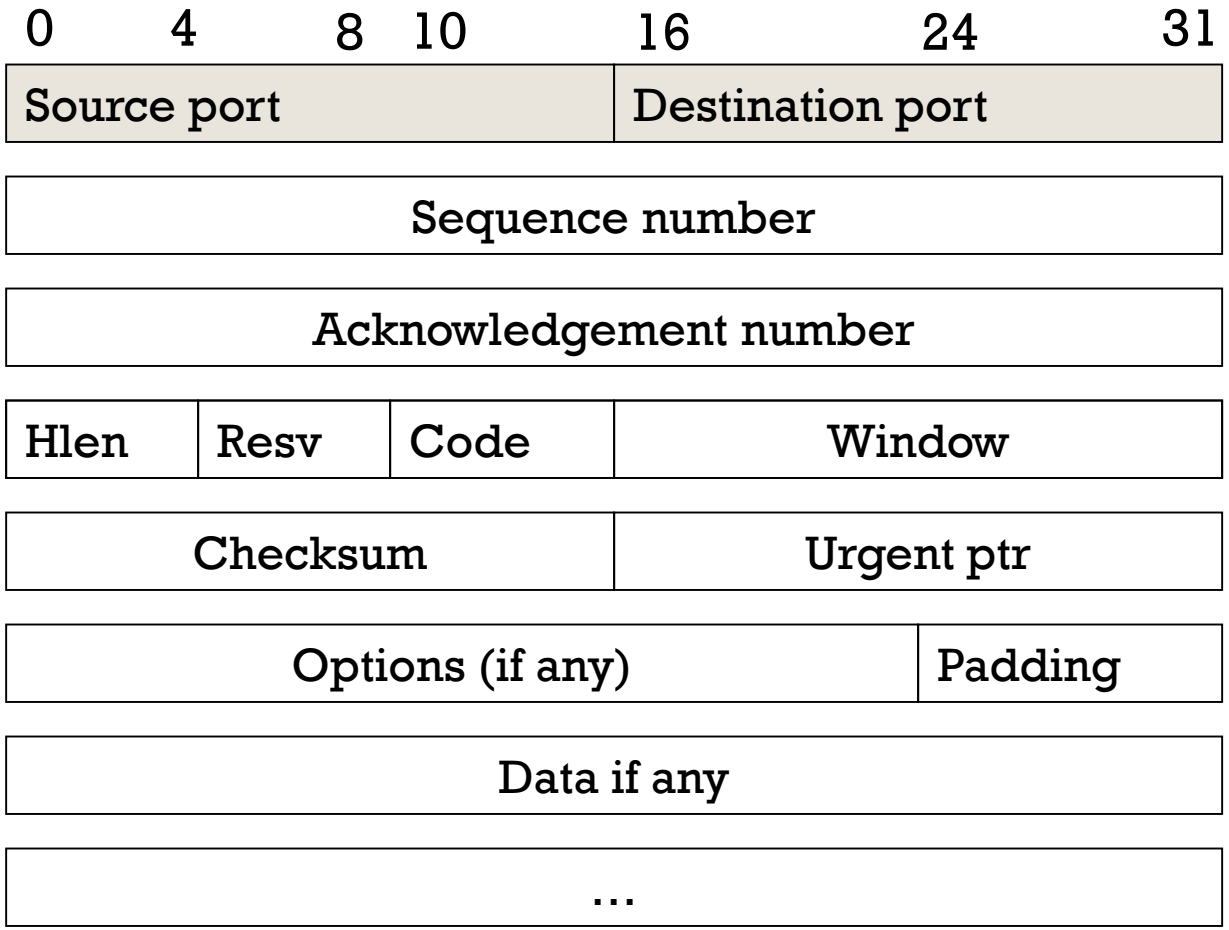
PORT NUMBER

TCP/UDP

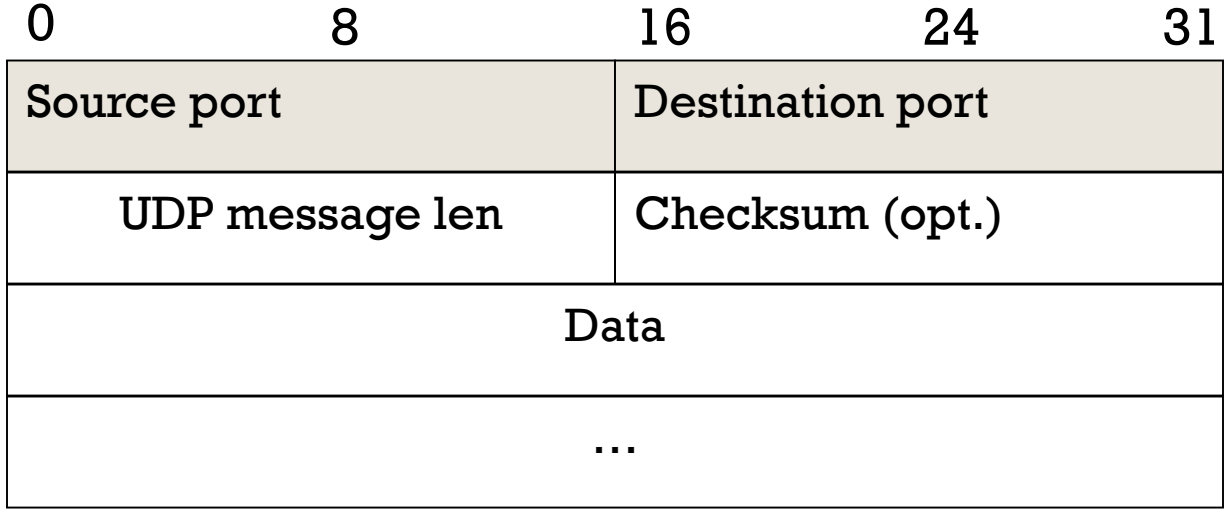


TCP/UDP SEGMENT FORMAT

TCP Header



UDP Header

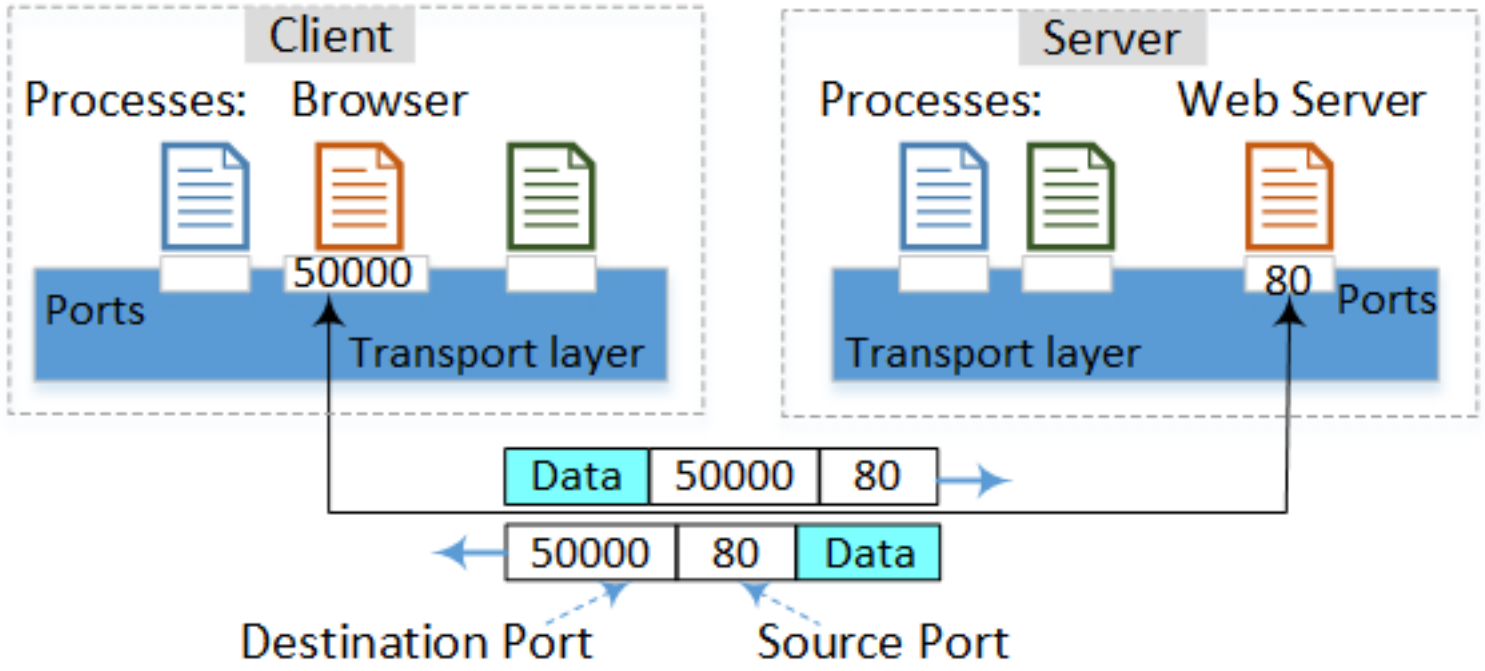


- Source/destination port: port numbers identify sending & receiving processes
  - Port number & IP address allow any application in any computer on Internet to be uniquely identified

TRANSPORT LAYER ADDRESSING

Addresses

- Data link layer → MAC address
- Network layer → IP address
- Transport layer → *Port number* (choose among multiple processes running on destination host)

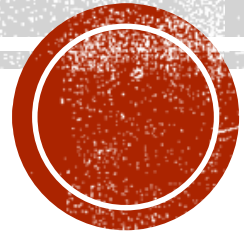


PORT NUMBERS

- Port numbers are 16-bit integers (0 → 65,535)
  - Servers use *well know ports*, 0-1023 are privileged
  - Clients use *ephemeral* (short-lived) ports
- *Internet Assigned Numbers Authority* (IANA) maintains a list of port number assignment
  - Well-known ports (0-1023) → controlled and assigned by IANA
  - Registered ports (1024-49151) → IANA registers and lists use of ports as a convenience (49151 is ¾ of 65536)
  - Dynamic ports (49152-65535) → ephemeral ports
- For well-known port numbers, see /etc/services on a UNIX or Linux machine

COMPUTER NETWORKS

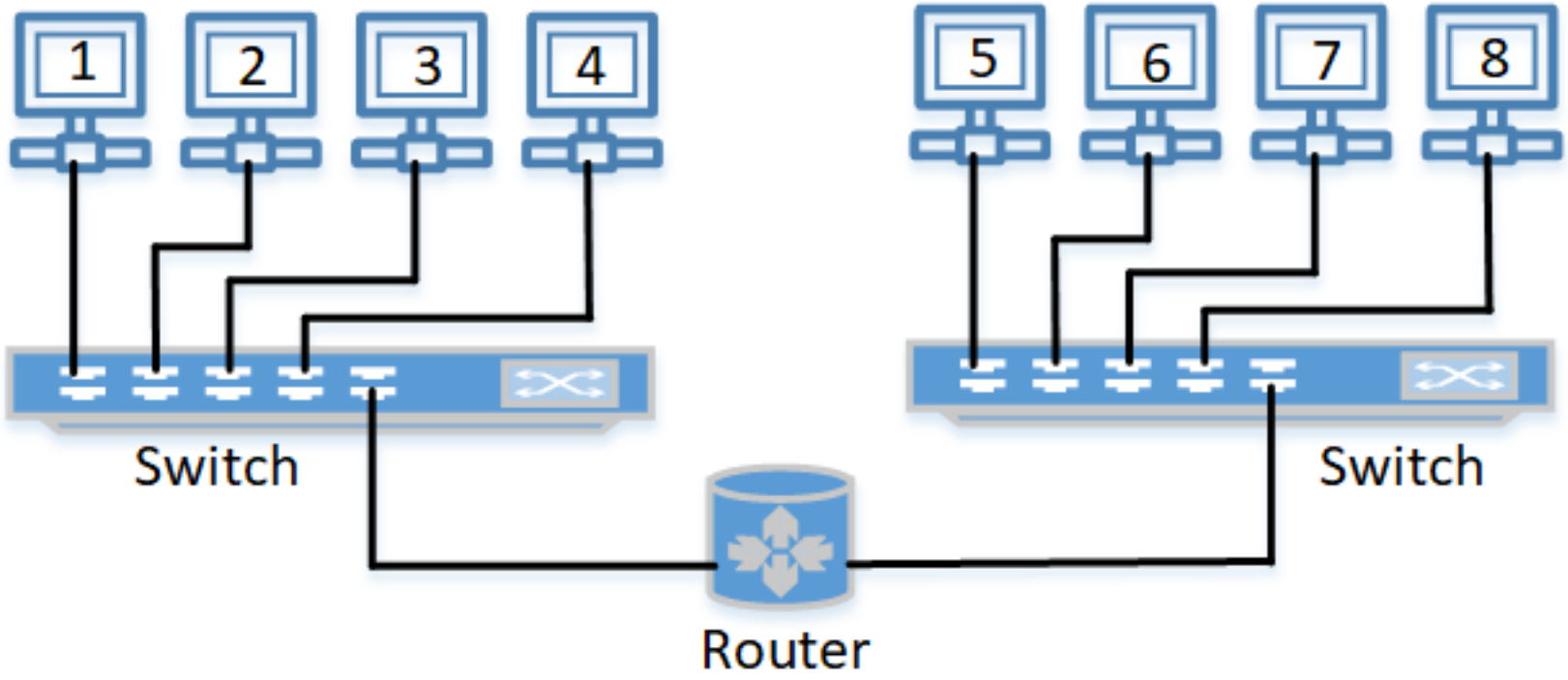
Physical, Logical, and Overlay Networks



PHYSICAL, LOGICAL, AND OVERLAY NETWORKS

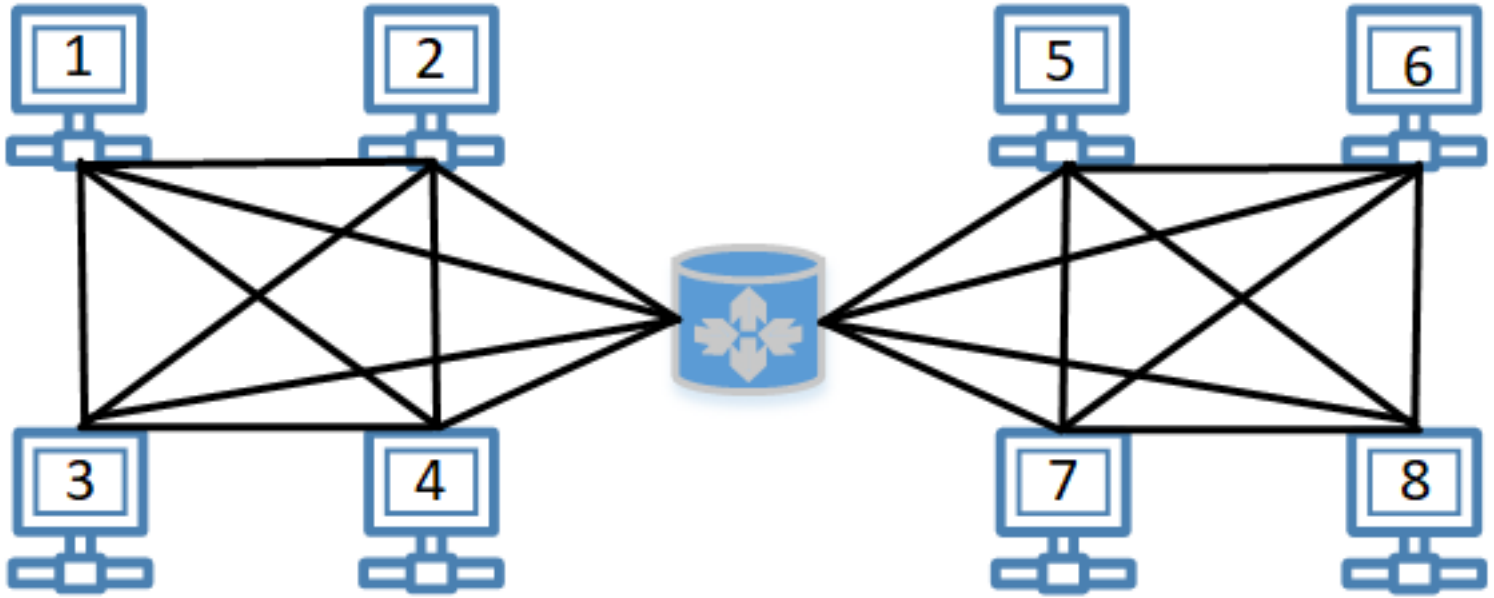
- Physical Network
  - A physical network is visible and physically presented to connect physical computers.
- Logical Network
  - A logical network is a virtual representation of a network that appears to the user as an entirely separate and self-contained network even though it might physically be only a portion of a larger network or a local area network.
- Overlay Network
  - An overlay network is a virtual network of nodes and logical links that are built on top of an existing network with the purpose of implementing a network service that is not available in the existing network.

PHYSICAL NETWORK



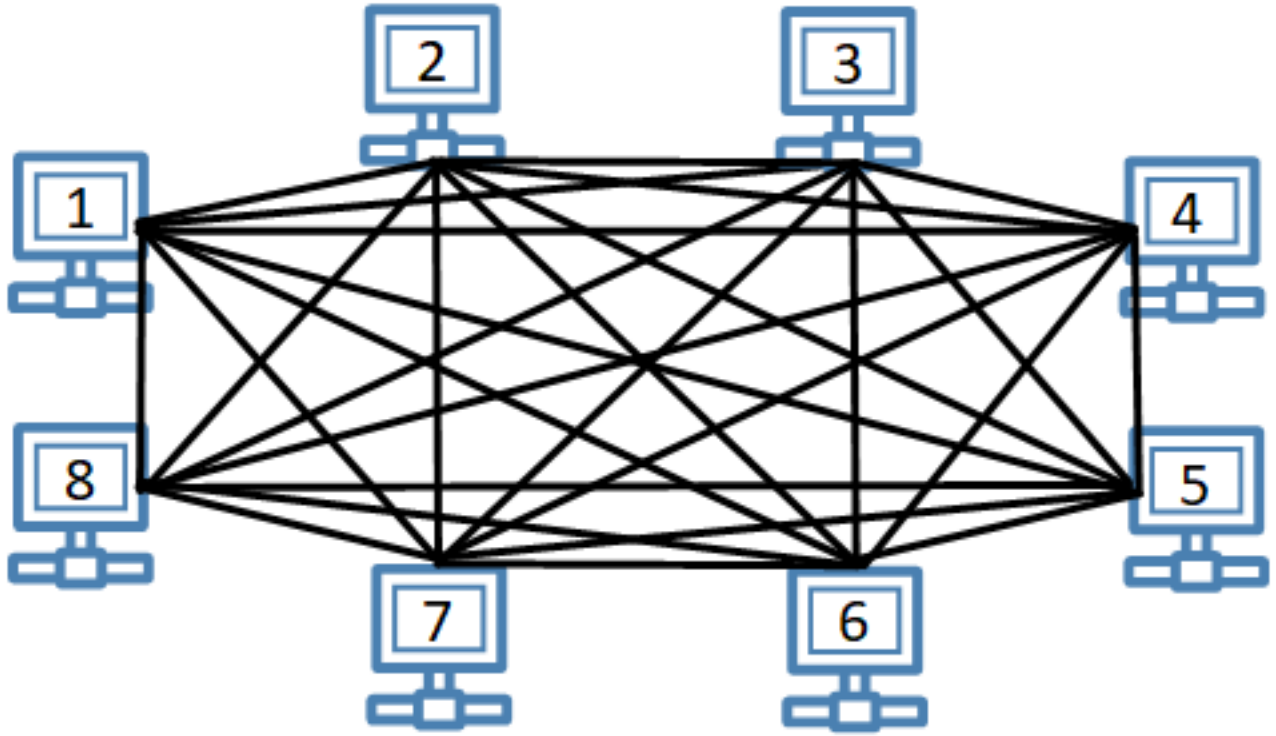
(*a*) Physical Network.

LOGICAL NETWORK (DATA LINK LAYER)



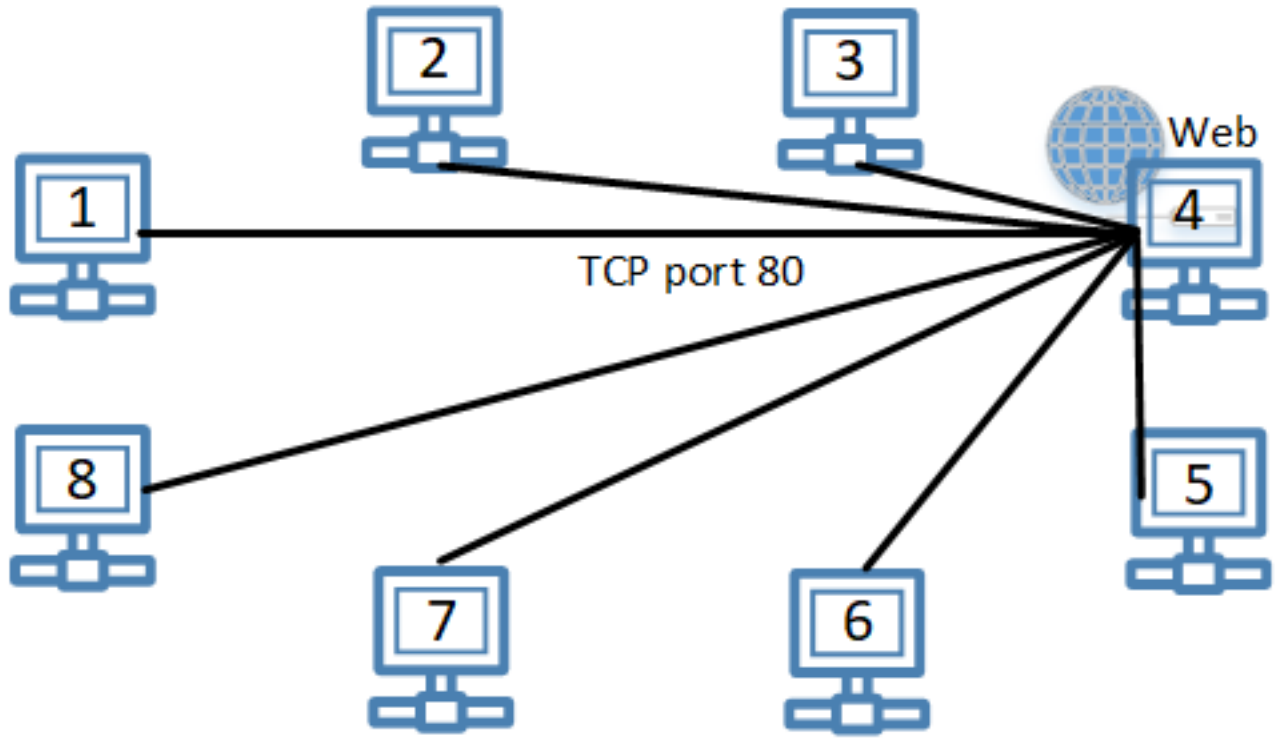
(*b*) Logical Network (Link-layer View).

LOGICAL NETWORK (NETWORK LAYER)



(*c*) Logical Network (Network-layer View).

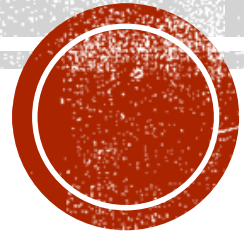
LOGICAL NETWORK (APPLICATION LAYER)



(*d*) Logical Network (Application-layer View-Web Service).

COMPUTER NETWORKING SERVICES

DHCP



REVISIT THE SIMPLE NETWORKING QUESTION

- An application scenario description:
  - "On the university campus, you boot up a laptop, open an email application, compose an email, provide the receiver's email address (e.g., john@xyz.com) and other related information such as email subject, make sure the email content and everything else is fine, and finally click send."
- Question:
  - "During this procedure, what networking protocols have been invoked and what magic happened in the computer network to allow John to receive and view your email?"

ANSWERS TO THE SIMPLE NETWORKING QUESTION

- DHCP Protocol is used after the computer booted up to get:
  - The computer's IP Address/Network Mask
  - The default gateway's IP address to transfer the data to a destination located on other networks (or not on the same local network)
  - The DNS's IP address to help the user to translate the target's URL to its IP address.
  - And others….
- ARP protocol is used to get the MAC address for a given IP address on a local network
- DNS protocol is used to translate the target's URL to its IP address
- Now, we talk about DHCP, ARP, and DNS

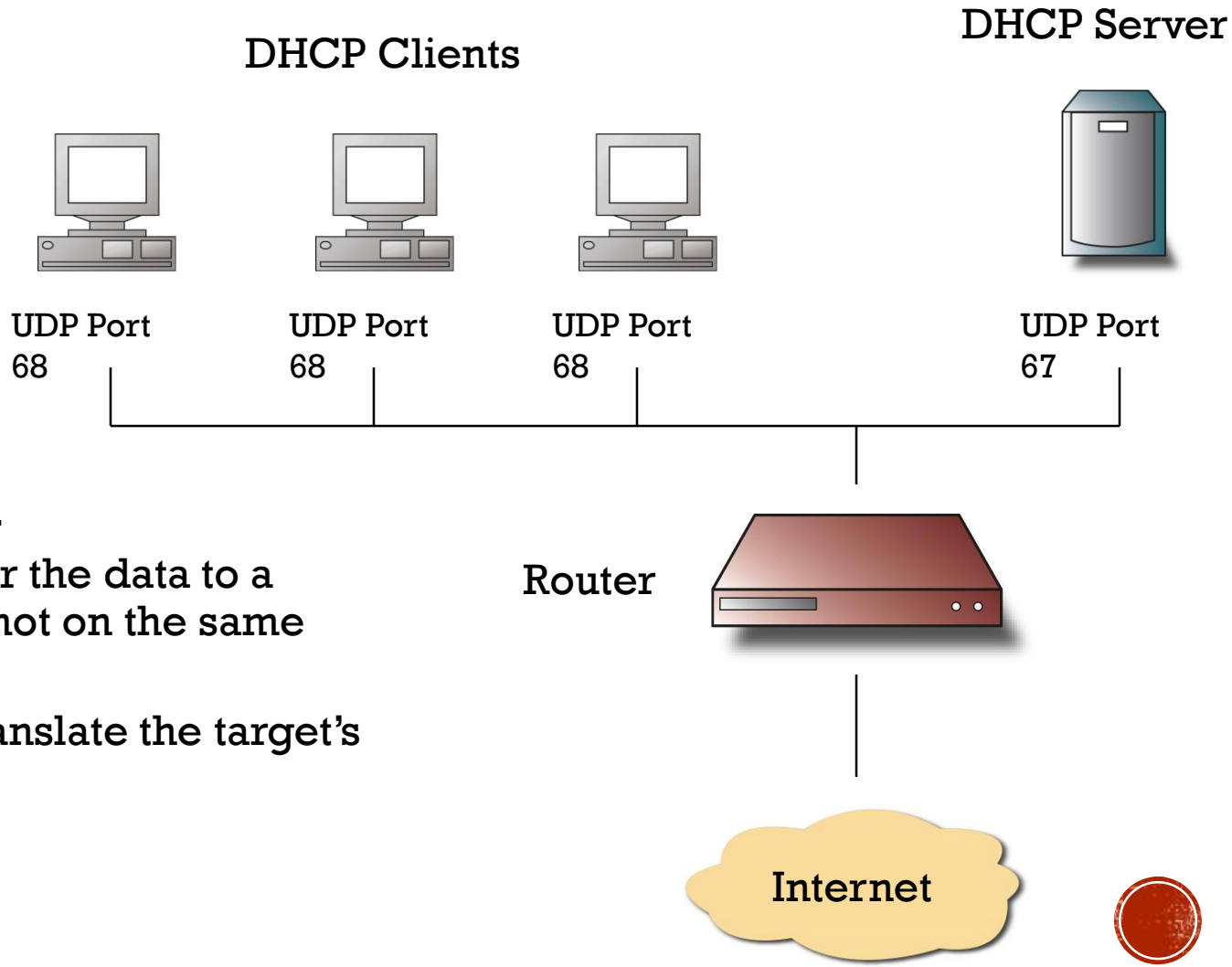
PROTOCOL LAYERS-OVERVIEW

TCP/IP		Protocols	OSI Model
Application		DNS, DHCP, FTP, HTTPS, IMAP, LDAP, NTP, POP3, RTP, RTSP, SSH, SIP, SMTP, SNMP, Telnet, TFTP	Application
		JPEG, MIDI, MPEG, PICT, TIFF	Presentation
		NetBIOS, NFS, PAP, SCP, SQL, ZIP	Session
Transport	Segement	TCP, UDP	Transport
Internet	Packet	ICMP, IGMP, IPv4, IPv6, IPSec, RIP	Network
Network Interface	Frame	ARP, ATM, CDP, FDDI, Frame Relay, HDLC, MPLS, PPP, STP, Token Ring	Data Link
	Bit	Bluetooth, Ethernet, DSL, ISDN, 802.11 Wi-Fi	Physical

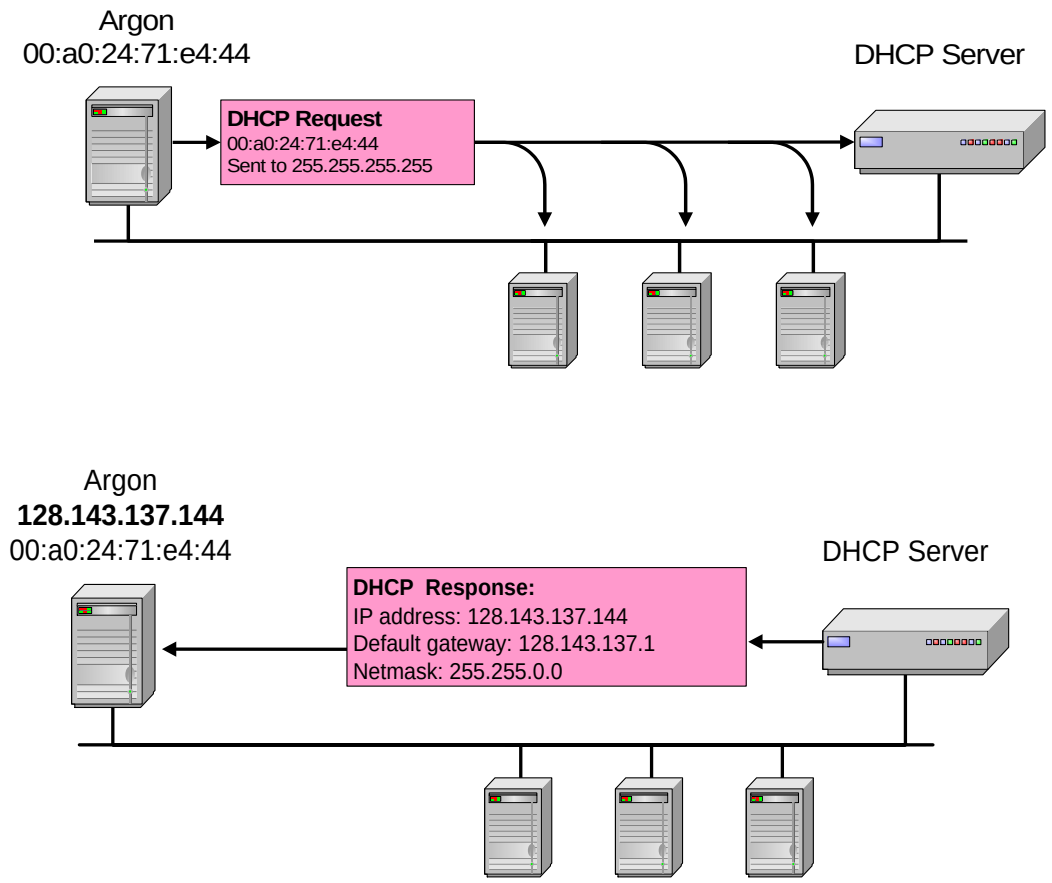
Narrow Waist Model

DHCP FEATURES

- Minimal human intervention
- It is running over UDP
- Dynamic allocation of IP addresses and providing computer network configuration parameters to hosts over network
  - The computer's IP Address/Network Mask.
  - The default gateway's IP address to transfer the data to a destination located on other networks (or not on the same local network).
  - The DNS's IP address to help the user to translate the target's URL to its IP address.
  - And others….

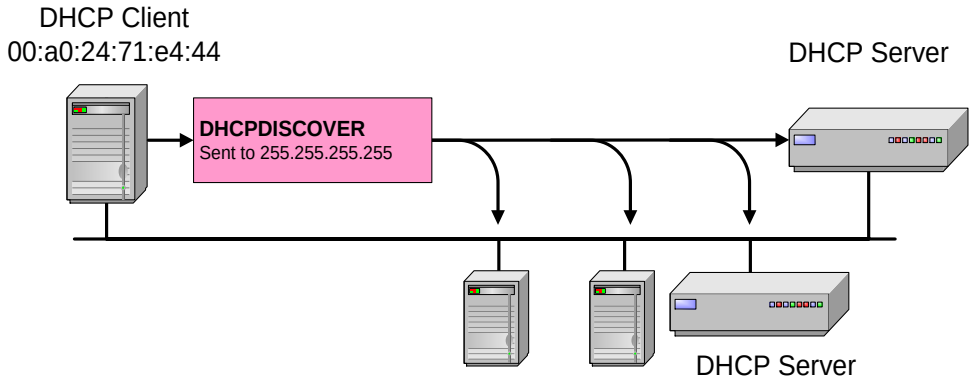


DHCP INTERACTION (SIMPLIFIED)

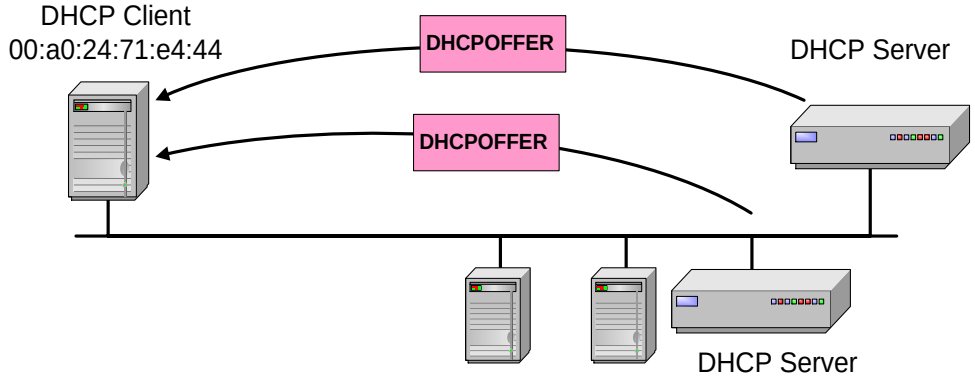


DHCP OPERATION

- DCHP DISCOVER



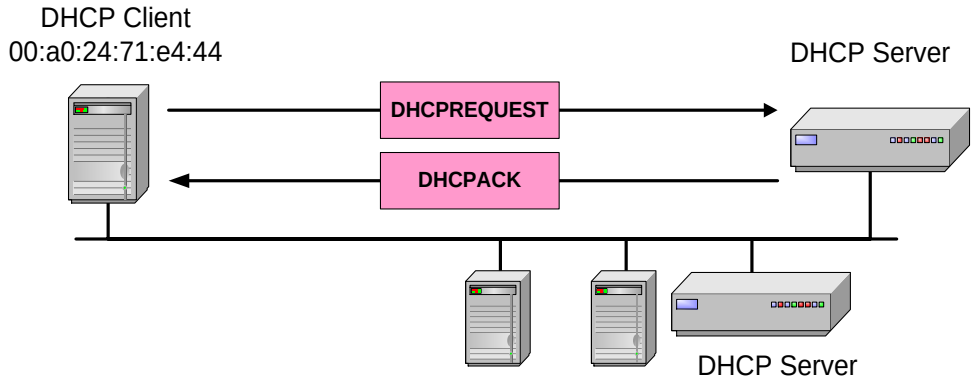
DCHP OFFER



DHCP OPERATION

DCHP DISCOVER

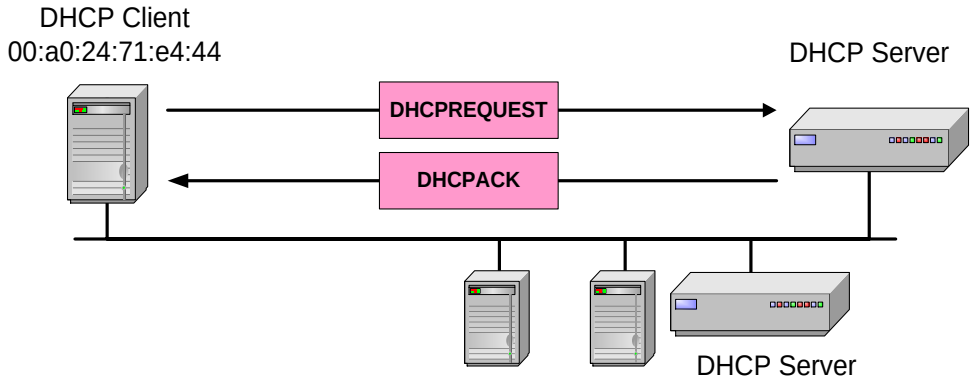
At this time, the DHCP client can start to use the IP address



Renewing a Lease

(sent when 50% of lease has expired)

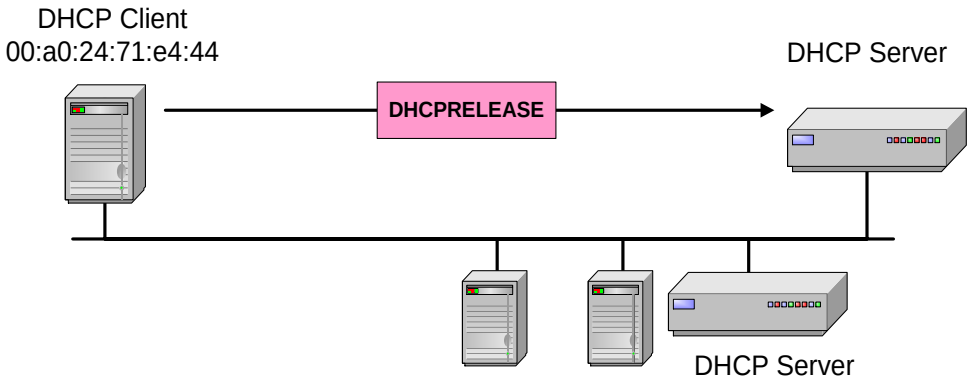
If DHCP server sends DHCPNACK, then address is released.



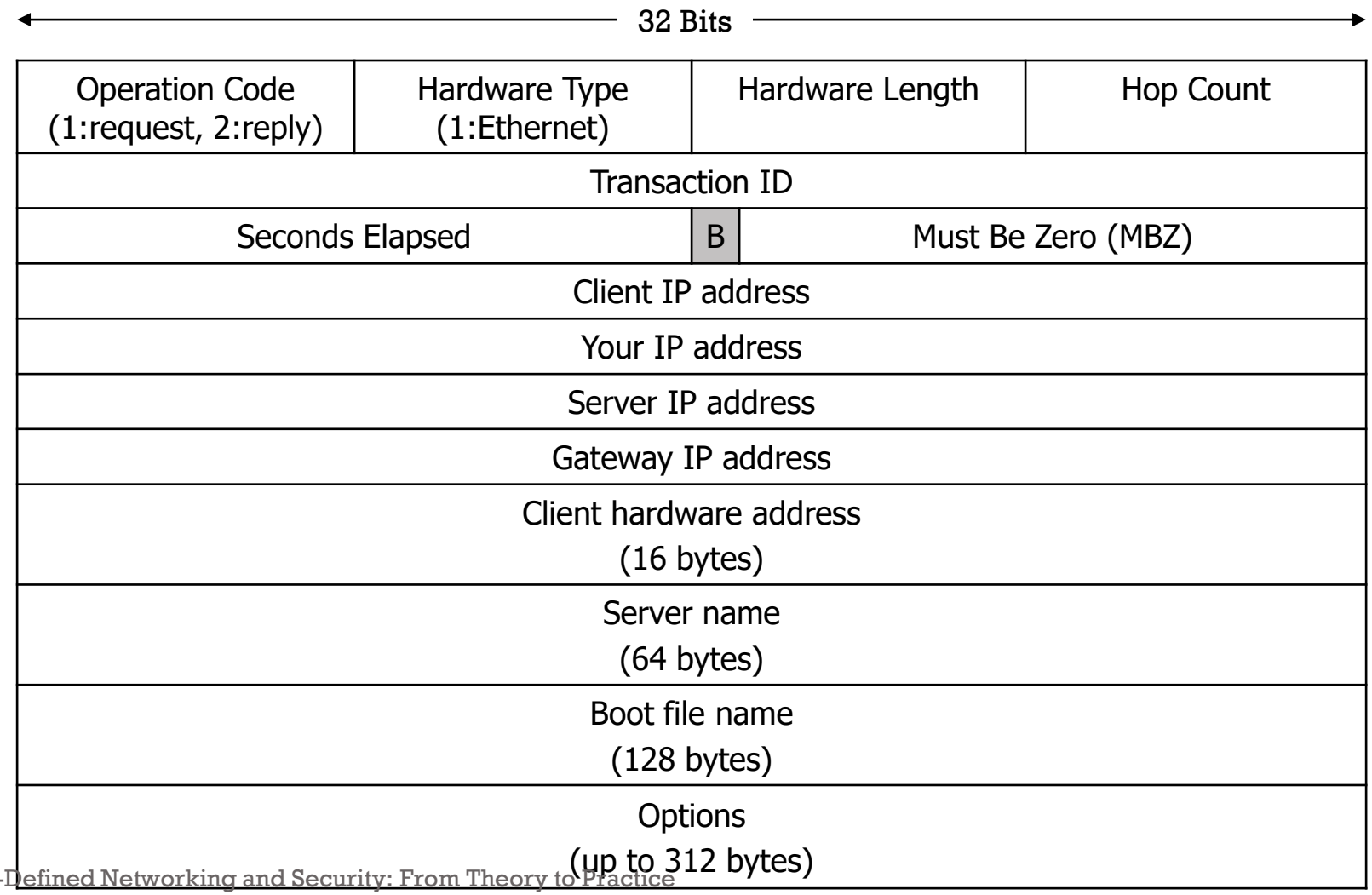
DHCP OPERATION

DCHP RELEASE

At this time, the DHCP client has released the IP address



DHCP FORMAT

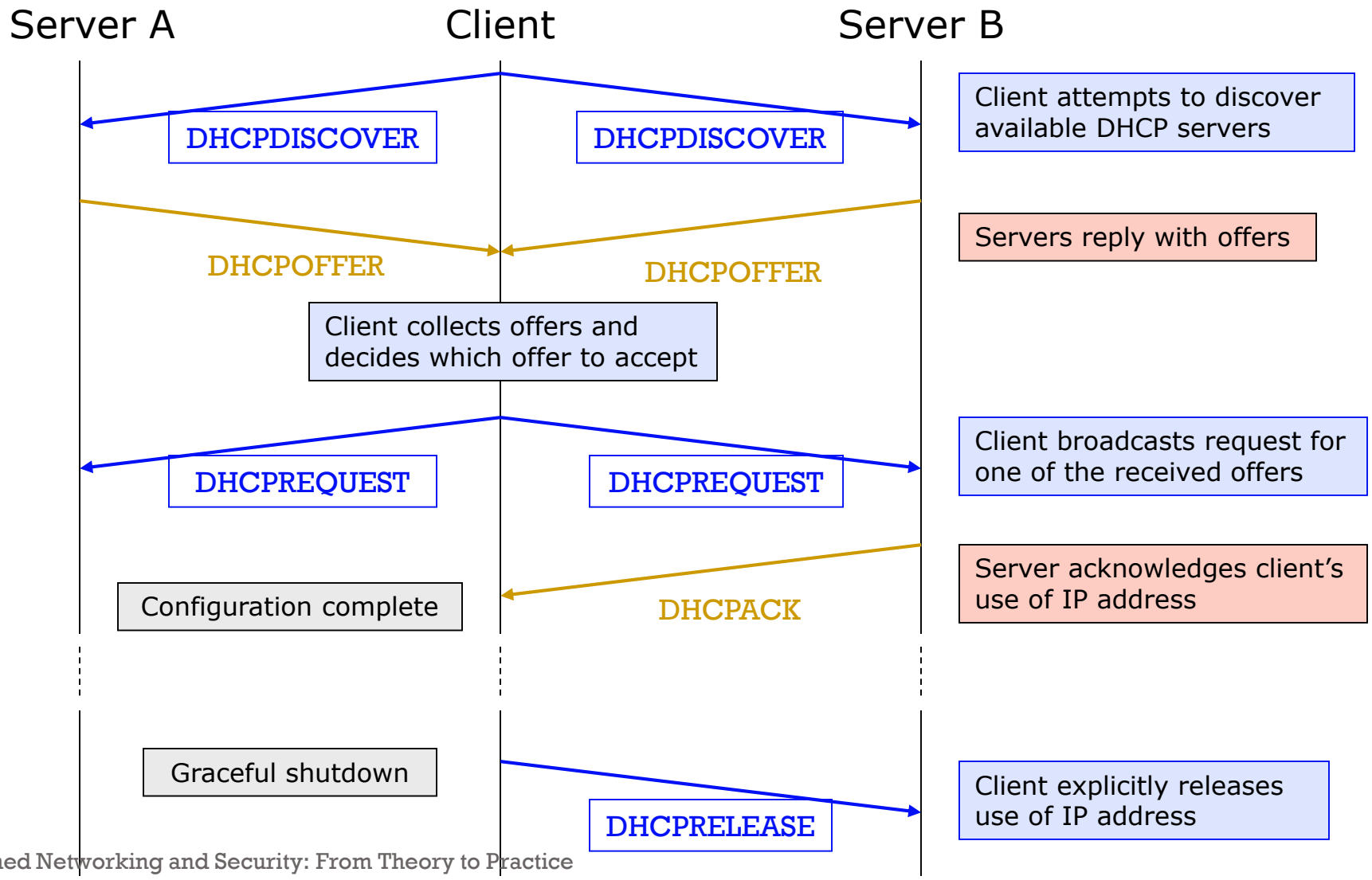


B: 1, DHCP
B: 0, BOOTP

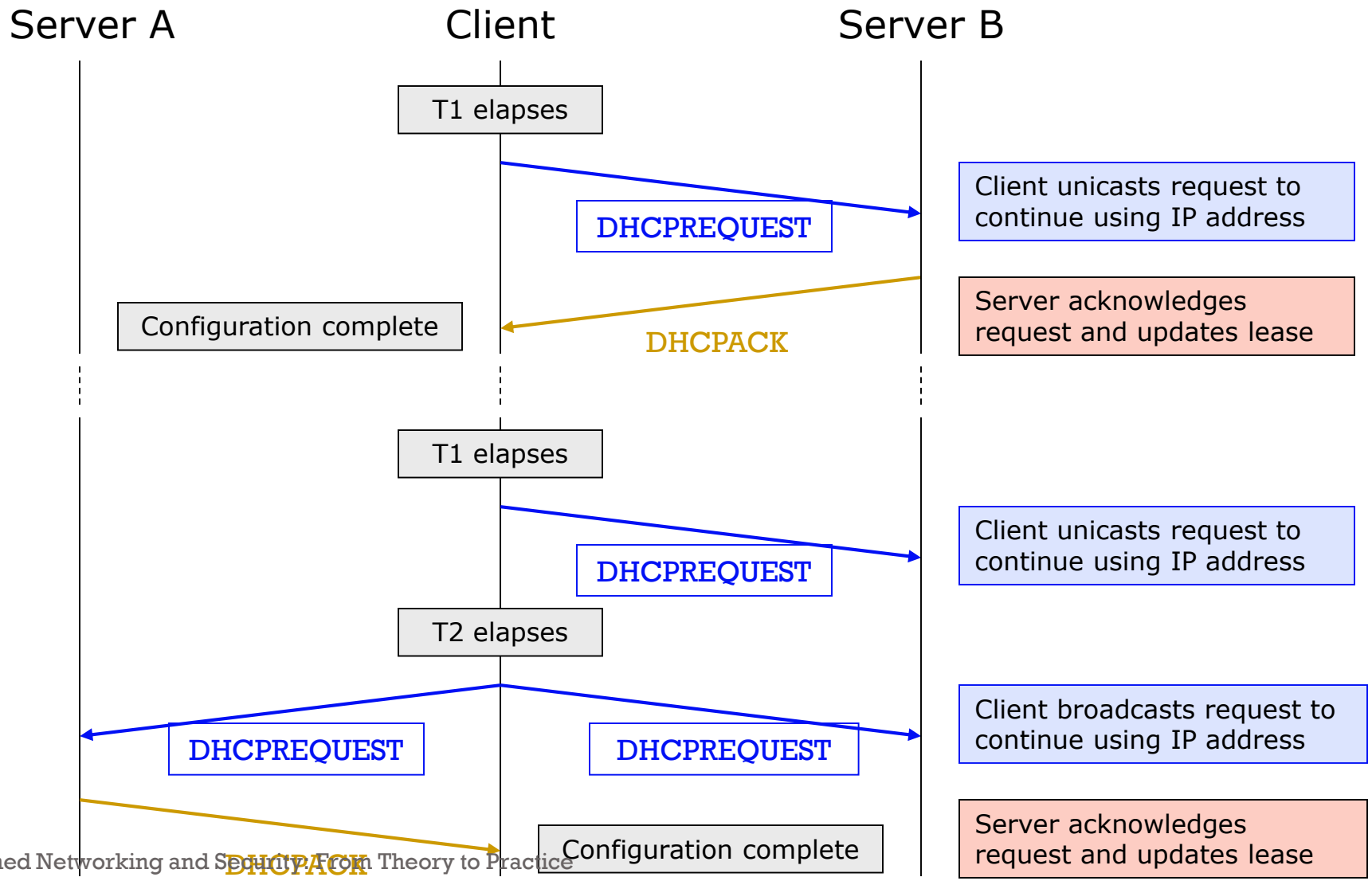
DHCP MESSAGE TYPES

DHCP Message	Use
DHCPDISCOVER	Client broadcast to locate available servers
DHCPOFFER	Server to client response offering configuration parameters
DHCPREQUEST	Client broadcast requesting offered parameters
DHCPDECLINE	Client to server notification that IP address is in use
DHCPACK	Server to client response confirming a request
DHCPNAK	Server to client response denying a request
DHCPRELEASE	Client to server request to relinquish IP address
DHCPINFORM	Client to server request for configuration parameters

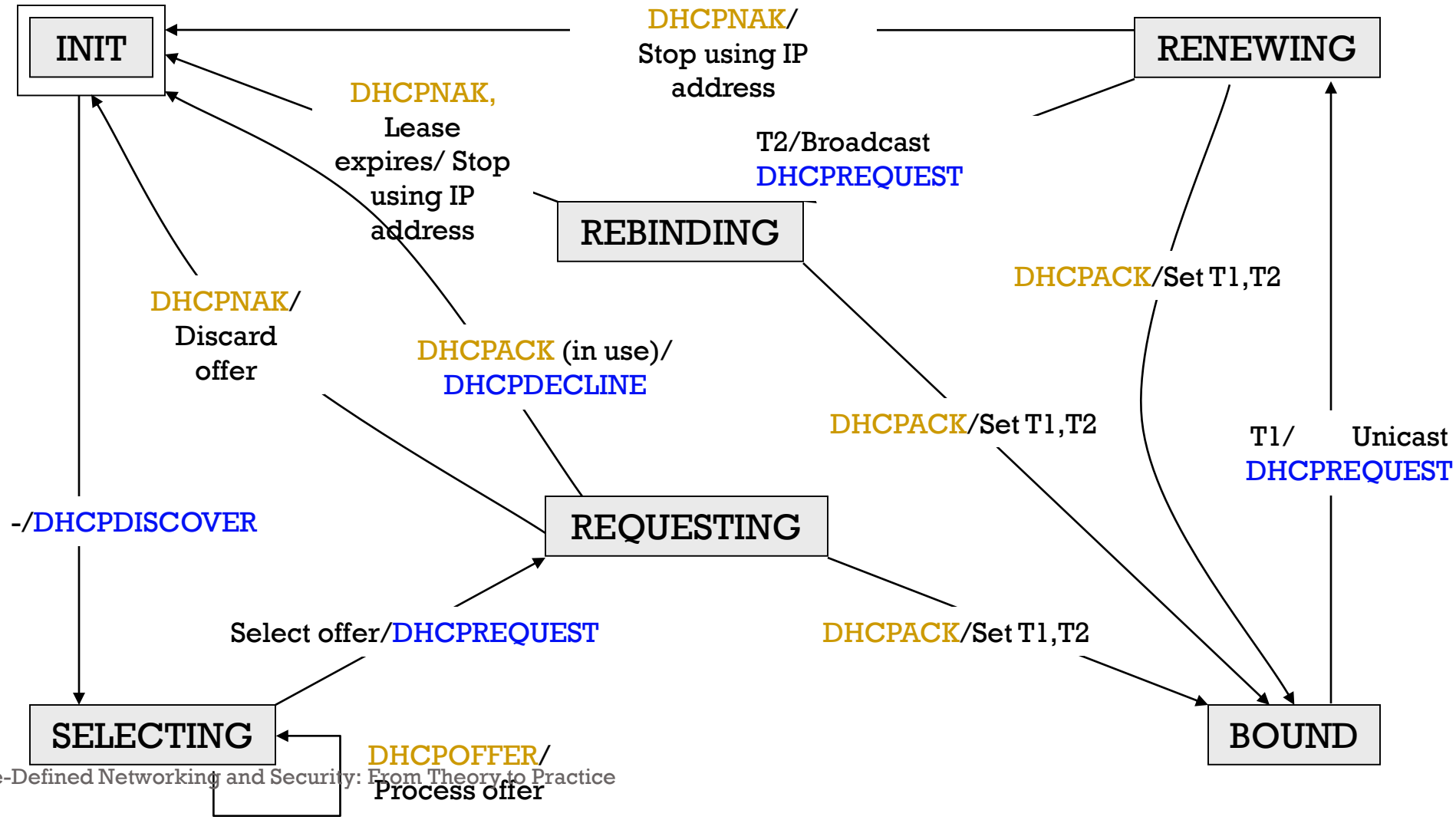
INITIAL MESSAGE FLOW



RENEWAL MESSAGE FLOW

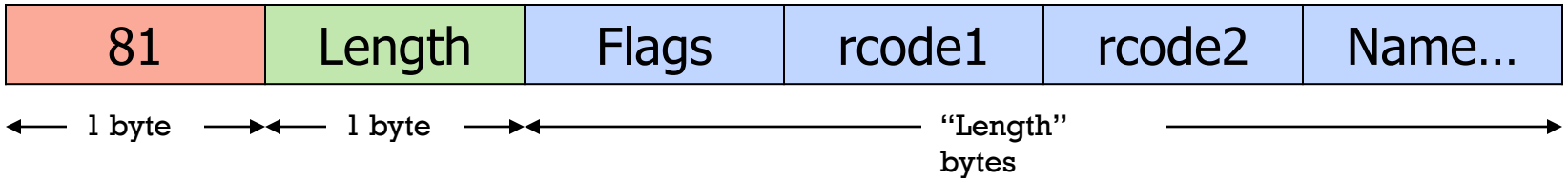


CLIENT FSM (SIMPLIFIED)



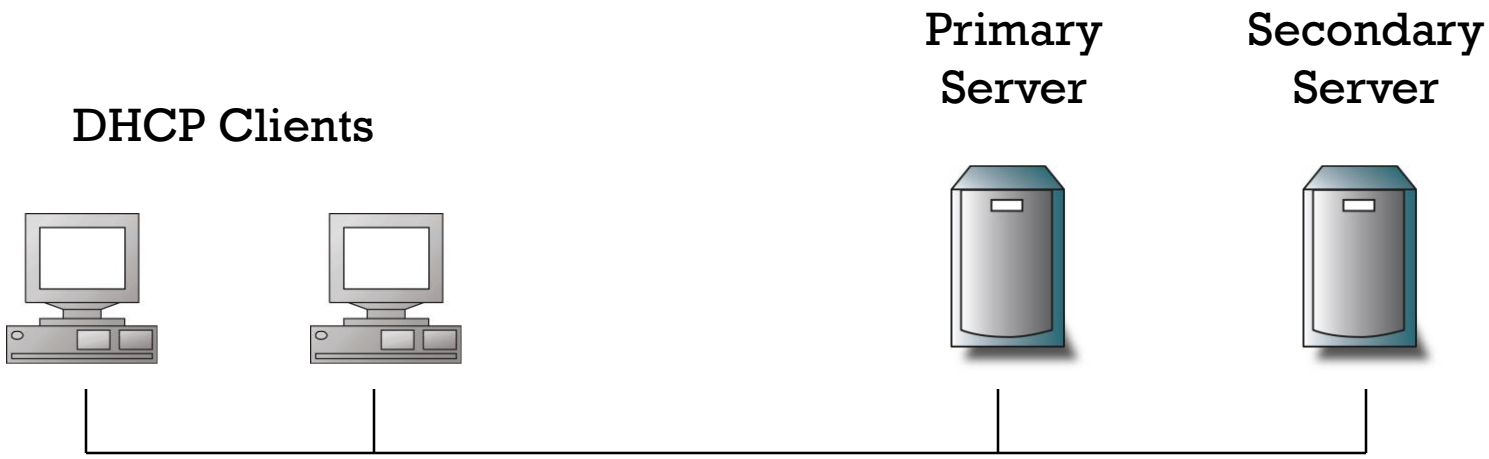
DYNAMIC DNS

- If IP address changes due to DHCP, DNS entry is wrong
- Client or server can update DNS
- Option 81: Client FQDN



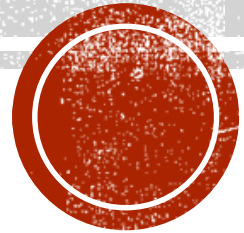
RELIABILITY

- Two synchronized DHCP servers on the same network: Primary, Secondary
- Permanent storage constantly communicated
- Failure: Secondary server takes over

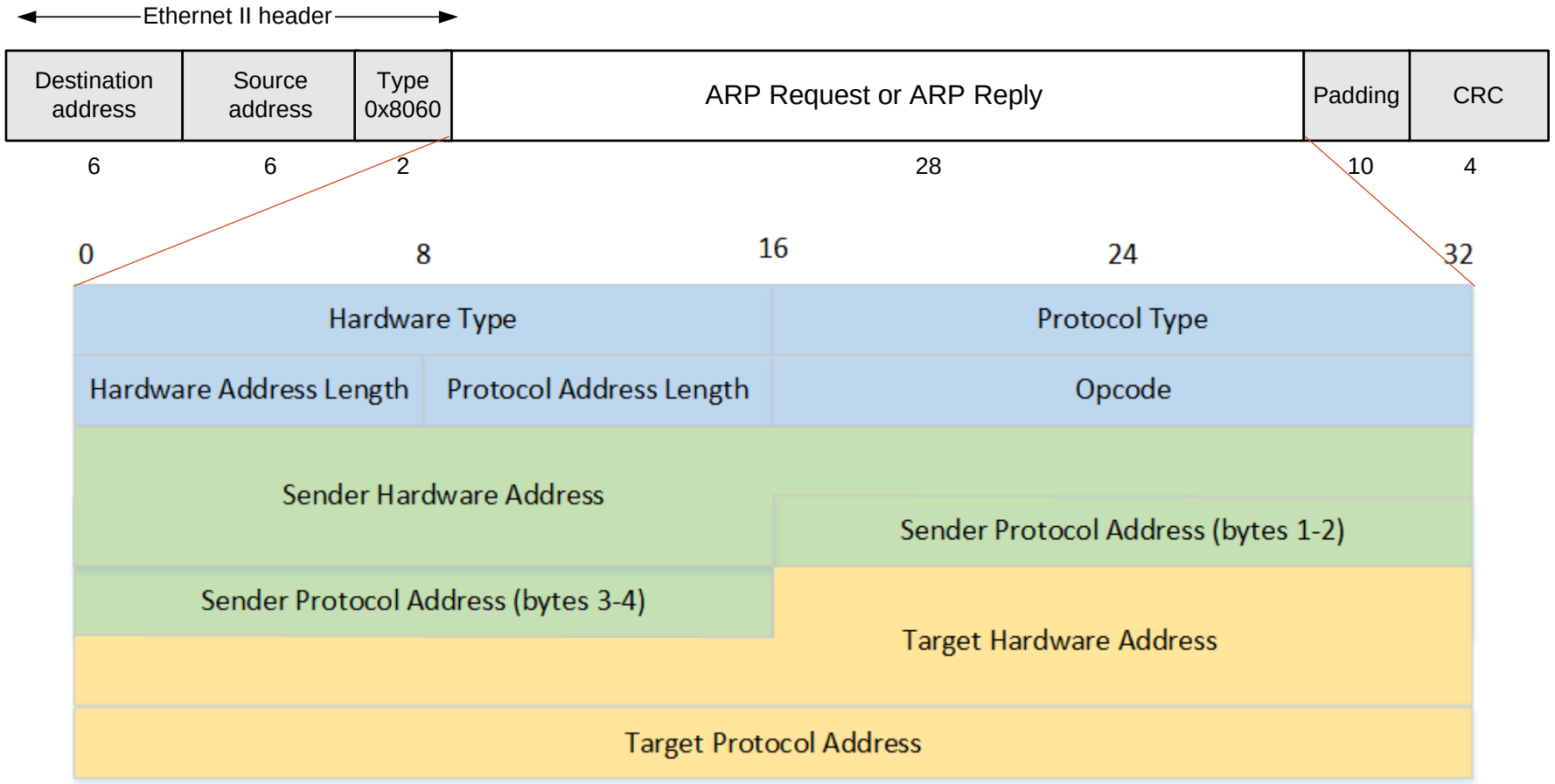


COMPUTER NETWORKING SERVICES

ARP



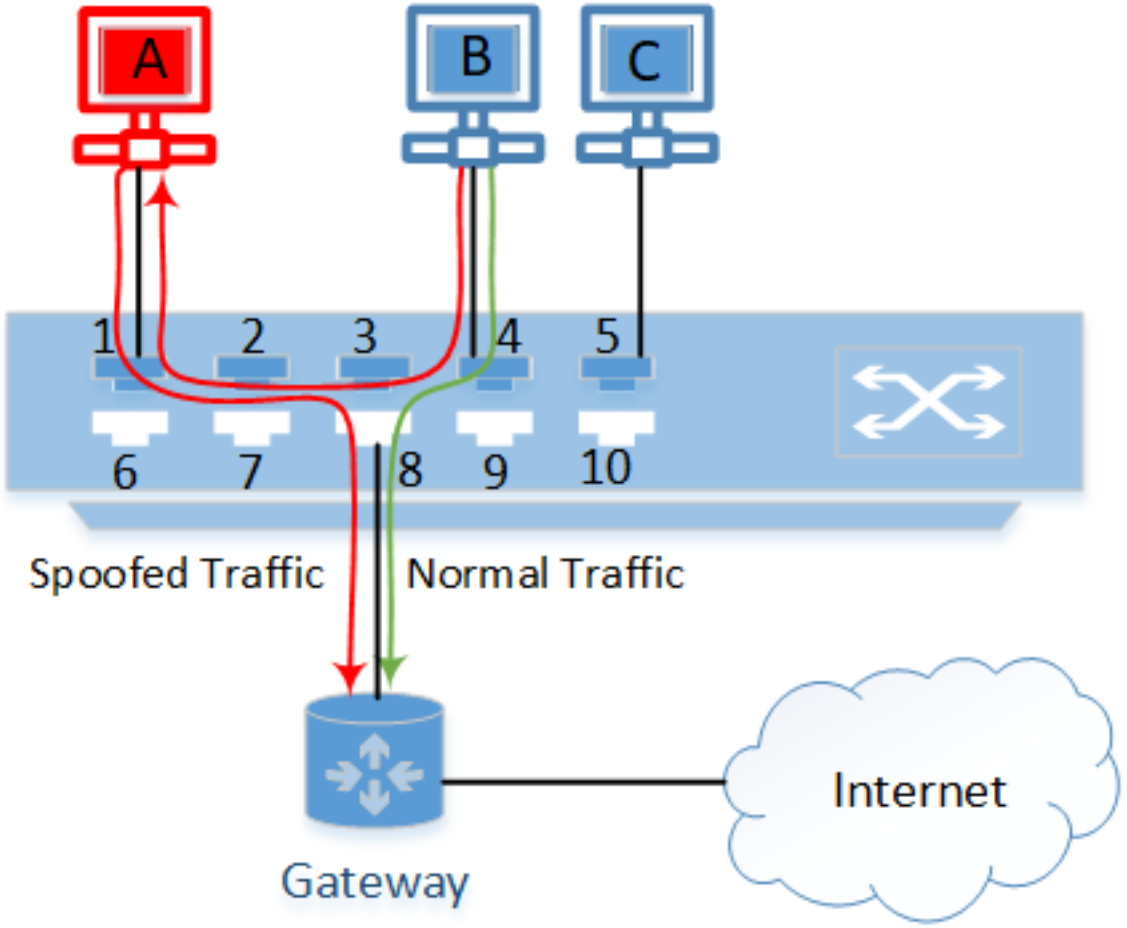
ARP PROTOCOL



HTYPE: 1 (Ethernet); PTYPE: 0x0800 (IPv4); HLEN: Len in octets, 6 (Ethernet); PLEN: 4 (IPv4)
Opcode: 1 (request), 2 (reply); SHA: MAC; SPA: IP address; THA: MAC; TPA: IP address

ARP SPOOFING

- ARP protocol is on top of datalink layer
- B sends a broadcast ARP message to MAC address FF:FF:FF:FF:FF:FF and asks what MAC address with the target IP 100.0.0.1 (e.g., C's IP), C then replies e.g., 00:ab:24:b2:05:ac, and then B can put the MAC in the dataframe and sends it to port 4, then the switch will sends it to port 5 based on its MAC table.

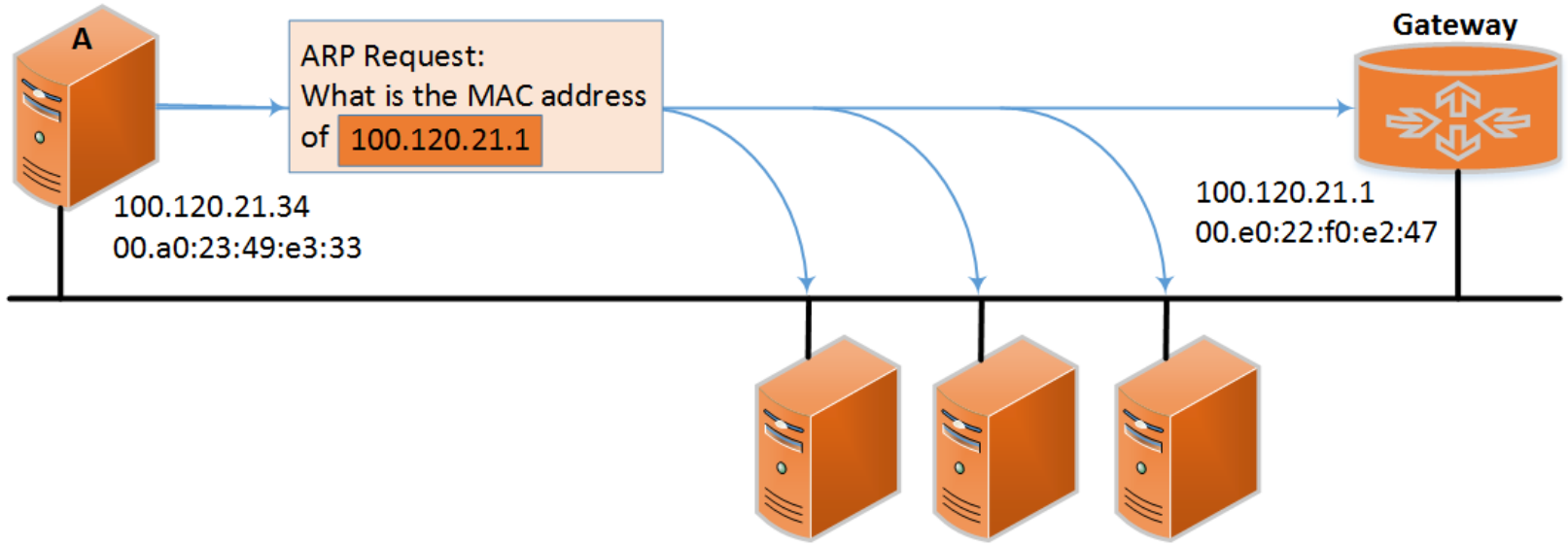


- In the scenario of ARP Spoofing attack, malicious A keeps on broadcasting the *ARP reply* message and says its MAC address is 00:ab:24:b2:05:ac to spoof C's MAC address, B then sends all the dataframes to A but not to C.
- A can also spoof the default gateway's MAC address to intercept all the traffic to the external network

ADDRESS TRANSLATION WITH ARP

ARP Request:

Node A broadcasts an ARP request to all stations on the network:
"What is the hardware address of 100.120.21.1?"



ARP Request from node A:

Source hardware address:

00:a0:23:49:e3:33

Source protocol address:

100.120.21.34

Target hardware address:

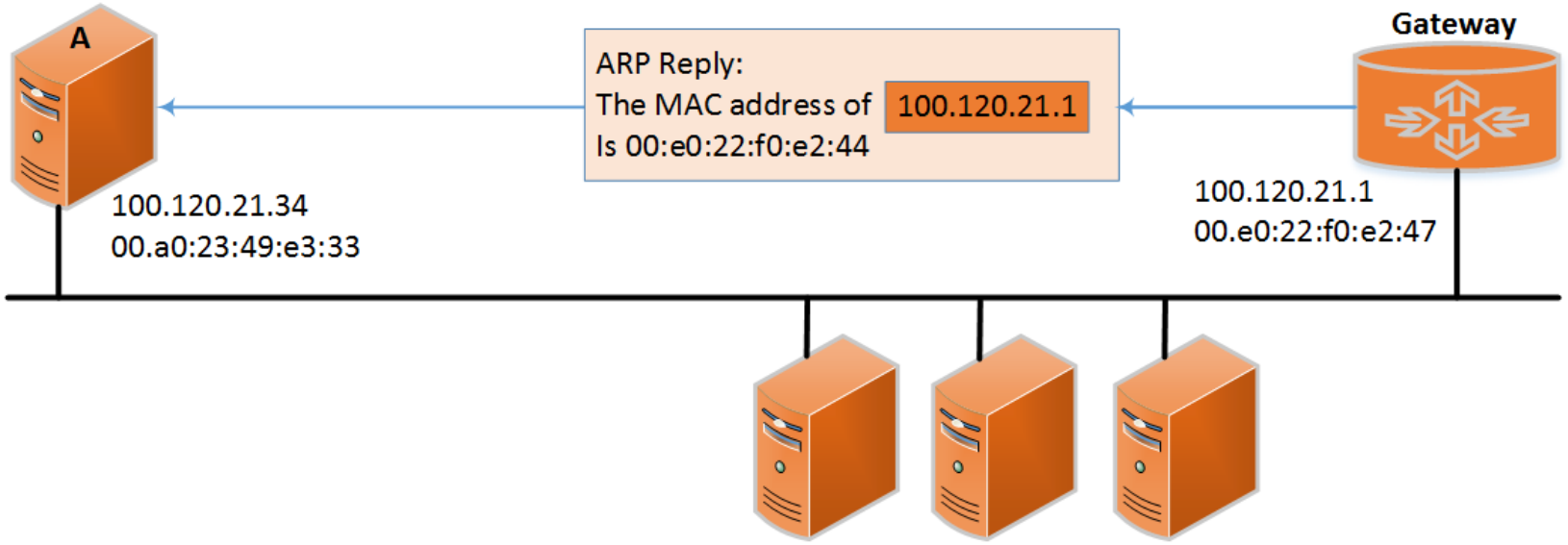
00:00:00:00:00:00

Target protocol address:

100.120.21.1

ADDRESS TRANSLATION WITH ARP

ARP Reply:
The gateway responds with an ARP Reply which contains the hardware address



ARP Reply from the gateway:

Source hardware address:
00:e0:22:f0:e2:47

Source protocol address:
100.120.21.1

Target hardware address:
00:a0:23:49:e3:33

Target protocol address:
100.120.21.34

ARP CACHE (TABLE)

- Contents of the ARP Cache (example):

  (100.120.21.37) at 00:10:4B:C5:D1:15 [ether] on eth0
  (100.120.21.36) at 00:B0:D0:E1:17:D5 [ether] on eth0
  (100.120.21.35) at 00:B0:D0:DE:70:E6 [ether] on eth0
  (100.120.136.90) at 00:05:3C:06:27:35 [ether] on eth1
  (100.120.136.34) at 00:B0:D0:E1:17:DB [ether] on eth1
  (100.120.21.13) at 00:B0:D0:E1:17:DF [ether] on eth0

- ARP is "plug-and-play":
  - nodes create their ARP tables *without intervention from net administrator*

THINGS TO KNOW ABOUT ARP

- What happens if an ARP Request is made for a non-existing host?

  Several ARP requests are made with increasing time intervals between requests. Eventually, ARP gives up.

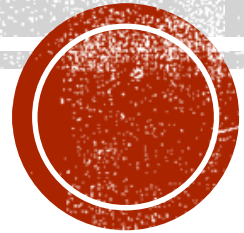
- What if a host sends an ARP request for its own IP address? Know as gratuitous ARP

  No response hopefully

  This is useful for detecting if an IP address has already been assigned (via DHCP).

COMPUTER NETWORKING SERVICES

DNS



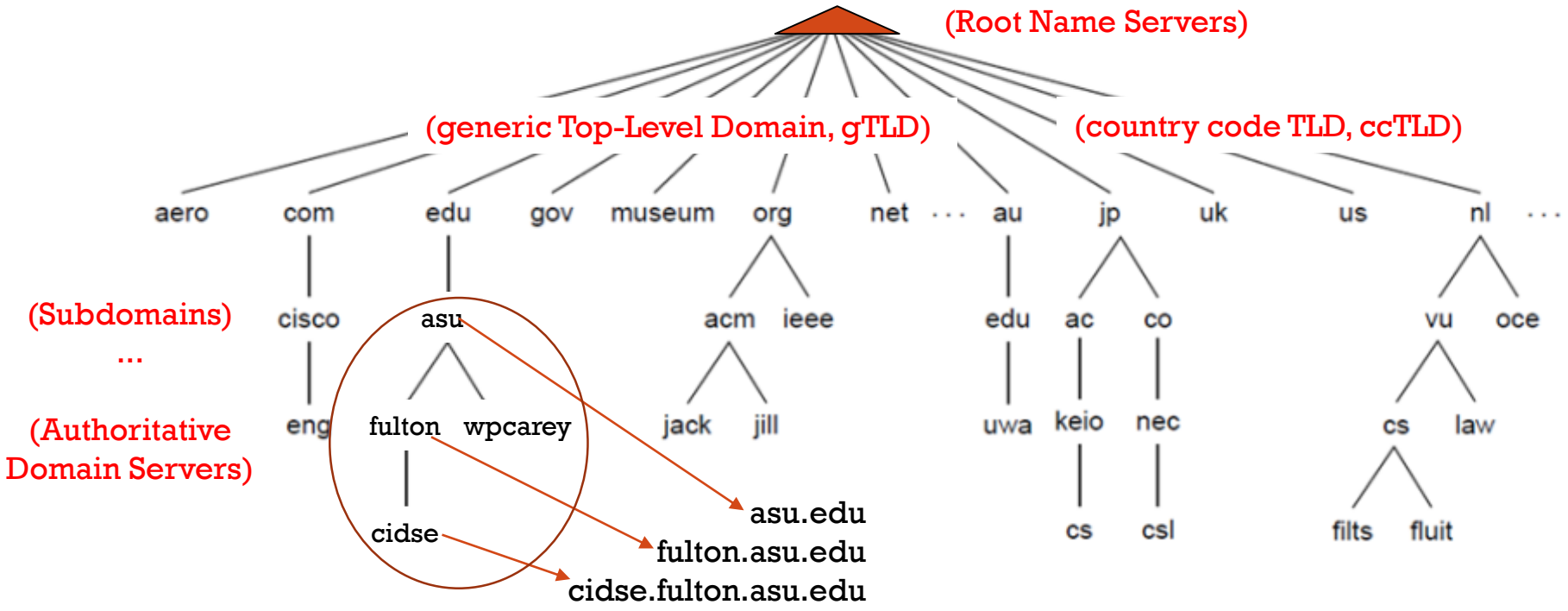
LOCAL HOSTNAME LOOKUP

- A simple way to map hostname into IP address locally.
- Linux → /etc/hosts
- Windows → %Systemroot%\System32\Drivers\Etc\hosts

```
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host
127.0.0.1       localhost
192.168.10.5    printer
192.168.10.12   fileserver
192.168.10.12   fs.home
150.10.1.20     mail.lon.mybank.com
150.10.2.20     mail.mybank.co.uk
150.10.2.20     mail.atl.mybank.com
150.10.2.21     www.mybank.com
150.10.2.21     webserver
```

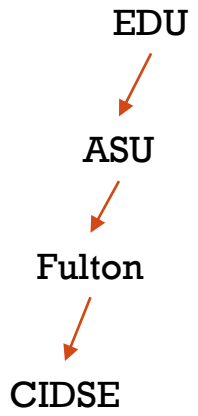
- **Change your hostname - /etc/hostname**

DOMAIN NAME SPACE



DOMAIN NAME VS IP ADDRESS

- IP Addresses
  - Not user friendly → 104.16.51.14
  - Reorganizing a network may change all IP addresses
- Domain Names
  - User Friendly → asu.edu
  - Hierarchical structure → **Tree**
    - **fulton.asu.edu**
    - cidse.fulton.asu.edu
  - Names can be used to obtain more information
  - Network services should use names instead of addresses



NAME DOMAINS

- Functional domains
  - .com, .edu, .gov, .net, .tv, .etc
- Geographic domains
  - .fr, .uk, .us, etc.
  - .ks, .mo, .nb, .ca, etc.
  - Hierarchical structure → cidse.fulton.asu.edu
    - machine.subdomain.2nd-level-domain.top-level-domain

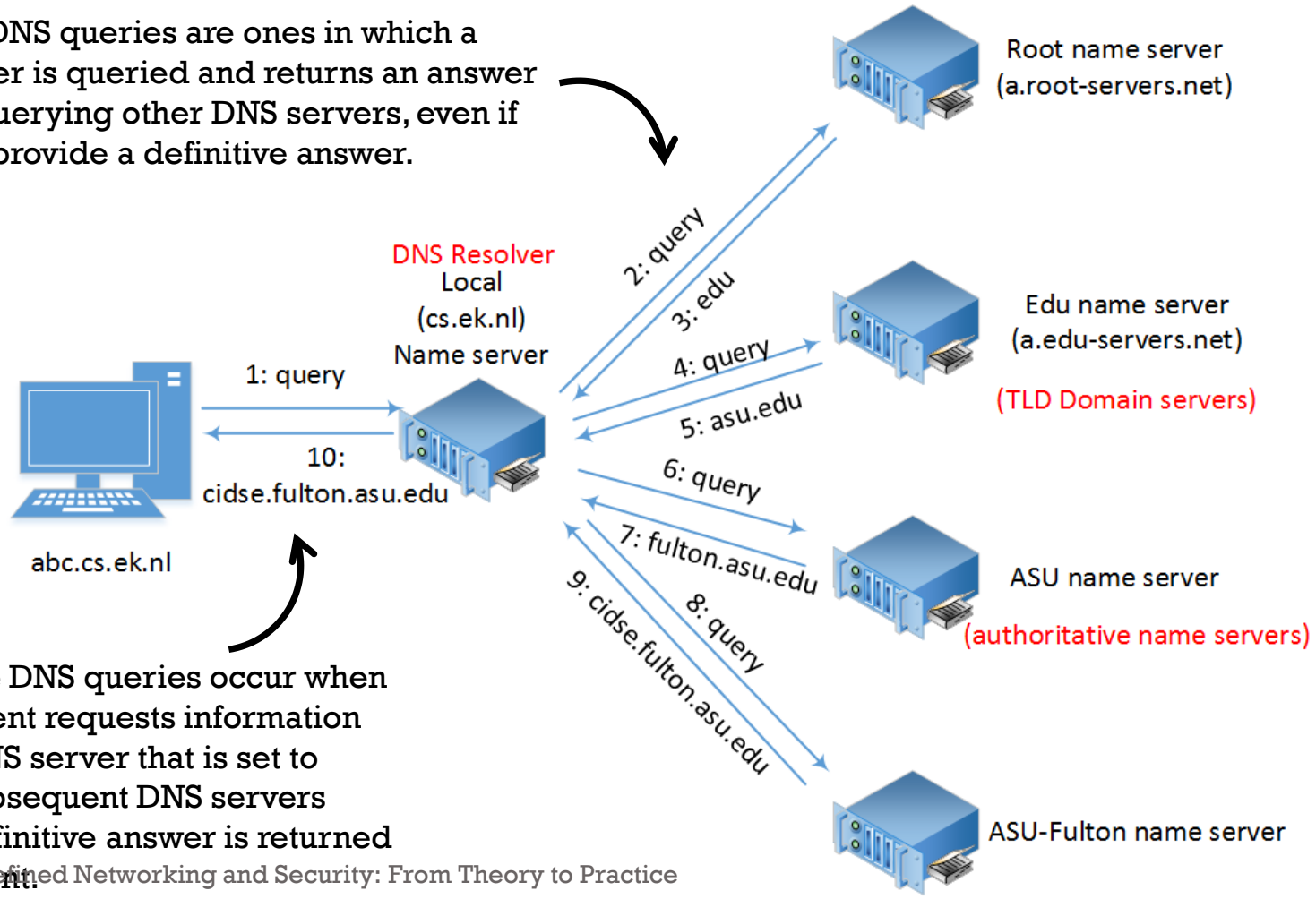
ROOT NAME SERVERS

Root Name Servers	Location(s)	Historical Name
A.ROOT-SERVERS.NET	Dulles, VA, USA	ns.internic.net
B.ROOT-SERVERS.NET	Marina Del Rey, CA, USA	ns1.isi.edu
C.ROOT-SERVERS.NET	Herndon, VA, USA; Los Angeles, CA, USA	c.psi.net
D.ROOT-SERVERS.NET	College Park, MD, USA	terp.umd.edu
E.ROOT-SERVERS.NET	Mountain View, CA, USA	ns.nasa.gov
F.ROOT-SERVERS.NET	Auckland, New Zealand; Sao Paulo, Brazil; Hong Kong, China; Johannesburg, South Africa; Los Angeles, CA, USA; New York, NY, USA; Madrid, Spain; Palo Alto, CA, USA; Rome, Italy; Seoul, Korea; San Francisco, CA, USA; San Jose, CA, USA; Ottawa, ON, Canada	ns.isc.org
G.ROOT-SERVERS.NET	Vienna, VA, USA	ns.nic.ddn.mil
H.ROOT-SERVERS.NET	Aberdeen, MD, USA	aos.arl.army.mil
I.ROOT-SERVERS.NET	Stockholm, Sweden Helsinki, Finland	nic.nordu.net
J.ROOT-SERVERS.NET	Dulles, VA, USA; Mountain View, CA, USA; Sterling, VA, USA; Seattle, WA, USA; Atlanta, GA, USA; Los Angeles, CA, USA; Amsterdam, The Netherlands	
K.ROOT-SERVERS.NET	London, UK; Amsterdam, The Netherlands	
L.ROOT-SERVERS.NET	Los Angeles, CA, USA	
M.ROOT-SERVERS.NET	Tokyo, Japan	

- What are they?
  - Sources for the top level domain servers.
- Where are they?
  - 13 different root name servers spread across the Internet
- What do they do?
  - Provide access to authoritative name servers

DNS LOOKUP

- Iterative DNS queries are ones in which a DNS server is queried and returns an answer without querying other DNS servers, even if it cannot provide a definitive answer.



- Recursive DNS queries occur when a DNS client requests information from a DNS server that is set to query subsequent DNS servers until a definitive answer is returned to the client.

- **DNS resolver** responsible for initiating and sequencing the queries that translating a domain name into an IP address.

- **Authoritative Name Servers** manage a Zone and either provide IP address lookup information themselves, or delegate the lookup of zone/sub-zone information to other DNS name servers.

NAME SERVERS WITH BIND9

- **BIND (Berkley Internet Naming Daemon)**
  - A software service to provide Domain Name Service
- Installation – `sudo apt-get install bind9`
- Restart service – `sudo /etc/init.d/bind9 restart`
- Main configuration files:
  - /etc/bind/named.conf (primary config file, you don't need to change it)
  - /etc/bind/named.conf.options (setup forwarders)
  - /etc/bind/named.conf.local (define forward and reverse zone file)

```
[...]

zone "example.com" {
     type master;
     file "/etc/bind/db.example.com";
};

[...]
```

```
zone "1.168.192.in-addr.arpa" {
        type master;
        notify no;
        file "/etc/bind/db.192";
};
```


FORWARD RESOLUTION SERVICE

- /etc/bind/db.example.com
- Map from a specific host name to its IP address.

```
;
; BIND data file for local loopback interface
;
$TTL    604800
@       IN      SOA     ns.example.com. root.example.com. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      ns.example.com.
ns      IN      A       192.168.1.10

;also list other computers
box     IN      A       192.168.1.21
```

BIND9 Server Howto, https://help.ubuntu.com/community/BIND9ServerHowto

DOMAIN RESOURCE RECORDS (RR)

Type	Meaning	Value
SOA	Start of authority	Parameters for this zone
A	IPv4 address of a host	32-Bit integer
AAAA	IPv6 address of a host	128-Bit integer
MX	Mail exchange	Priority, domain willing to accept email
NS	Name server	Name of a server for this domain
CNAME	Canonical name	Domain name
PTR	Pointer	Alias for an IP address
SPF	Sender policy framework	Text encoding of mail sending policy
SRV	Service	Host that provides it
TXT	Text	Descriptive ASCII text

REVERSE RESOLUTION SERVICE

- /etc/bind/db.192
- Map from a specific host name to its IP address.

Edit **/etc/bind/named.conf.local** and add the following:

```
zone "1.168.192.in-addr.arpa" {
        type master;
        notify no;
        file "/etc/bind/db.192";
};
```

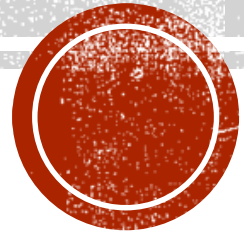
```
;
; BIND reverse data file for local loopback interface
;
$TTL    604800
@       IN      SOA     ns.example.com. root.example.com. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      ns.
10      IN      PTR     ns.example.com.

; also list other computers
21      IN      PTR     box.example.com.
```

BIND9 Server Howto, https://help.ubuntu.com/community/BIND9ServerHowto

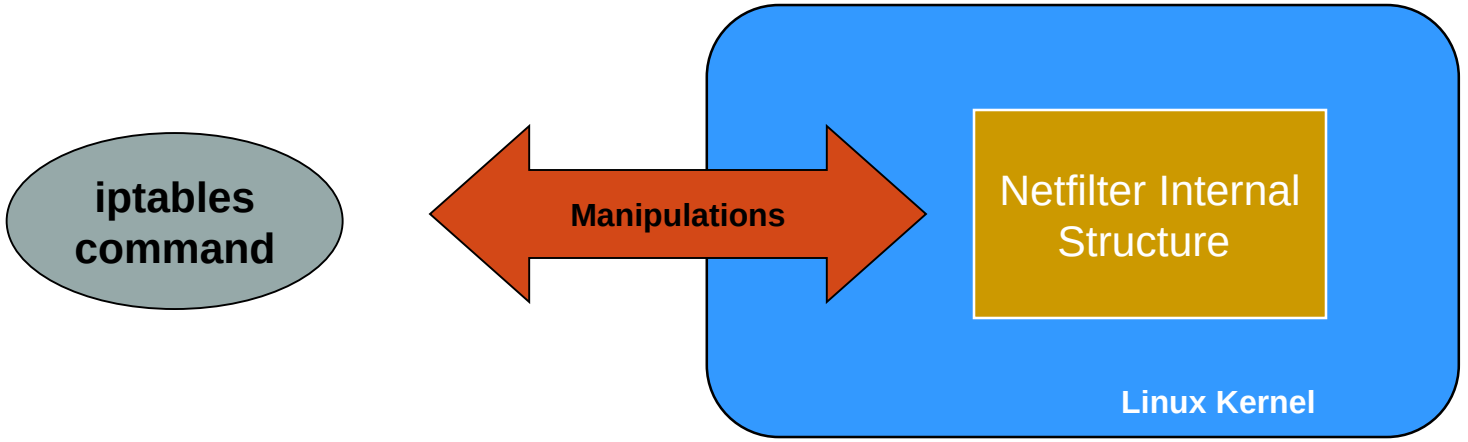
COMPUTER NETWORKING SERVICES

iptables



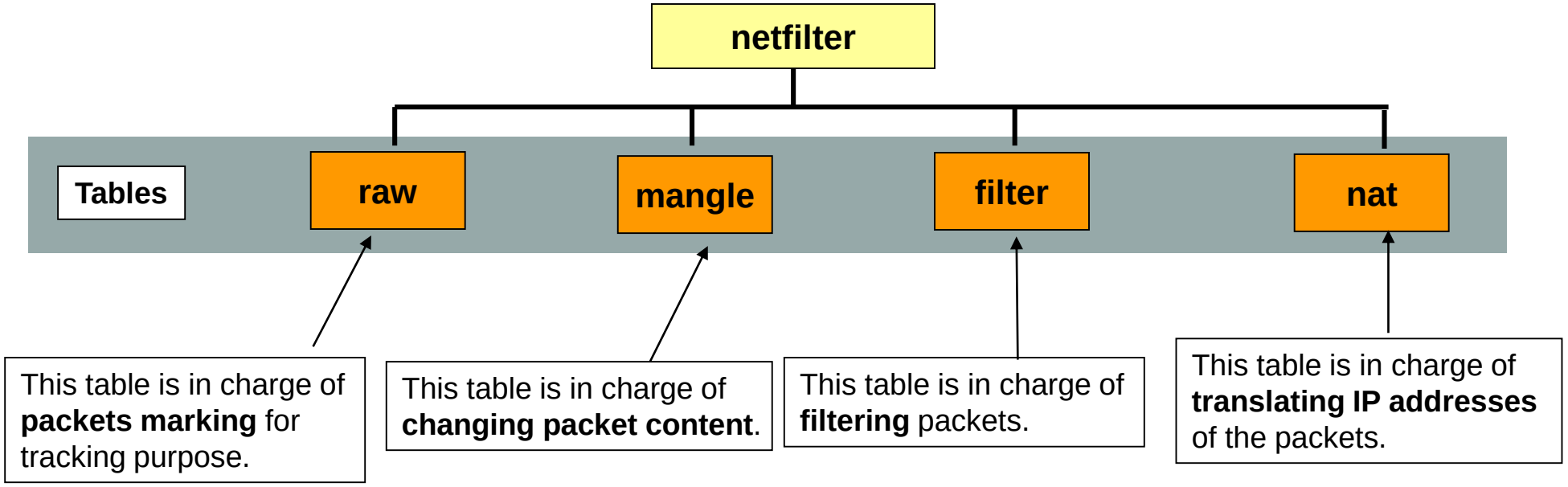
IPTABLES

- Actually, **iptables** is a user-level program that controls the kernel-level network module called **netfilter**.



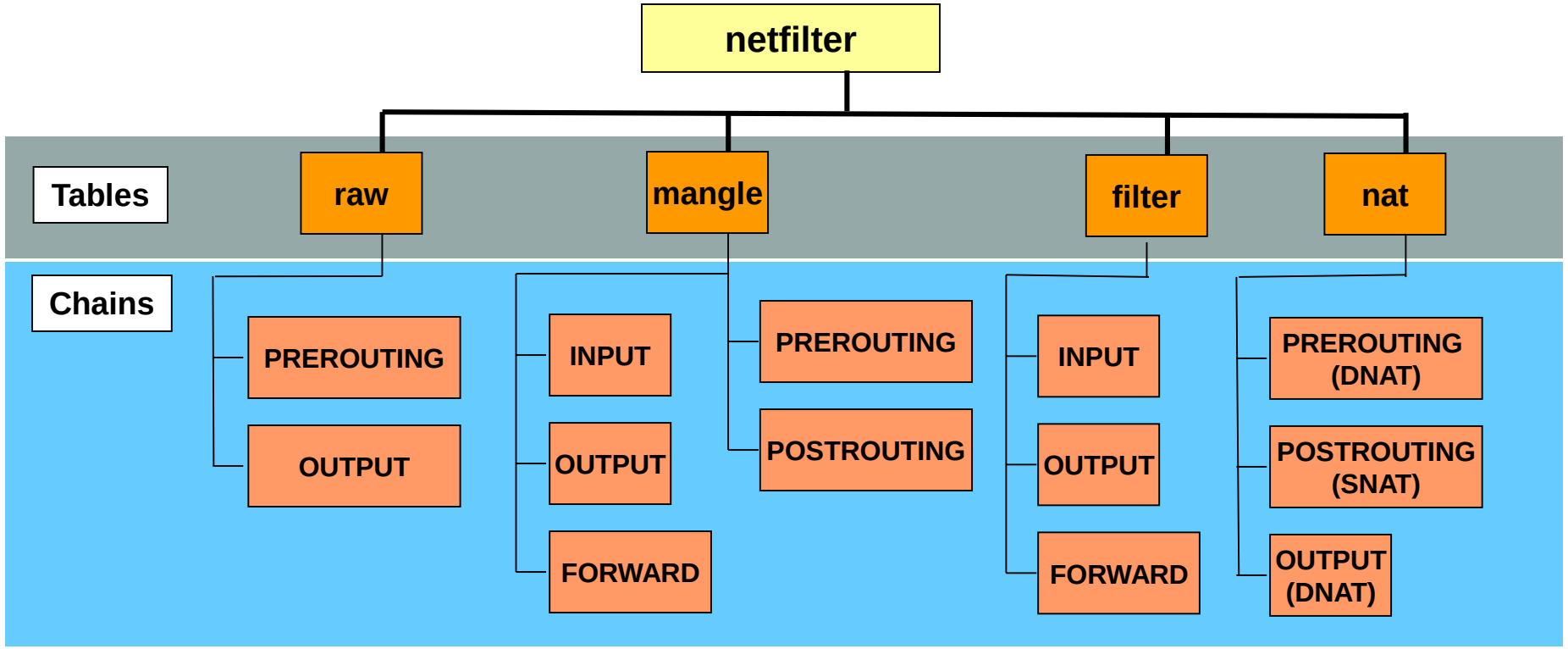
IPTABLES – TABLES AND CHAINS

- Each function provided by the netfilter architecture is presented as a **table**.

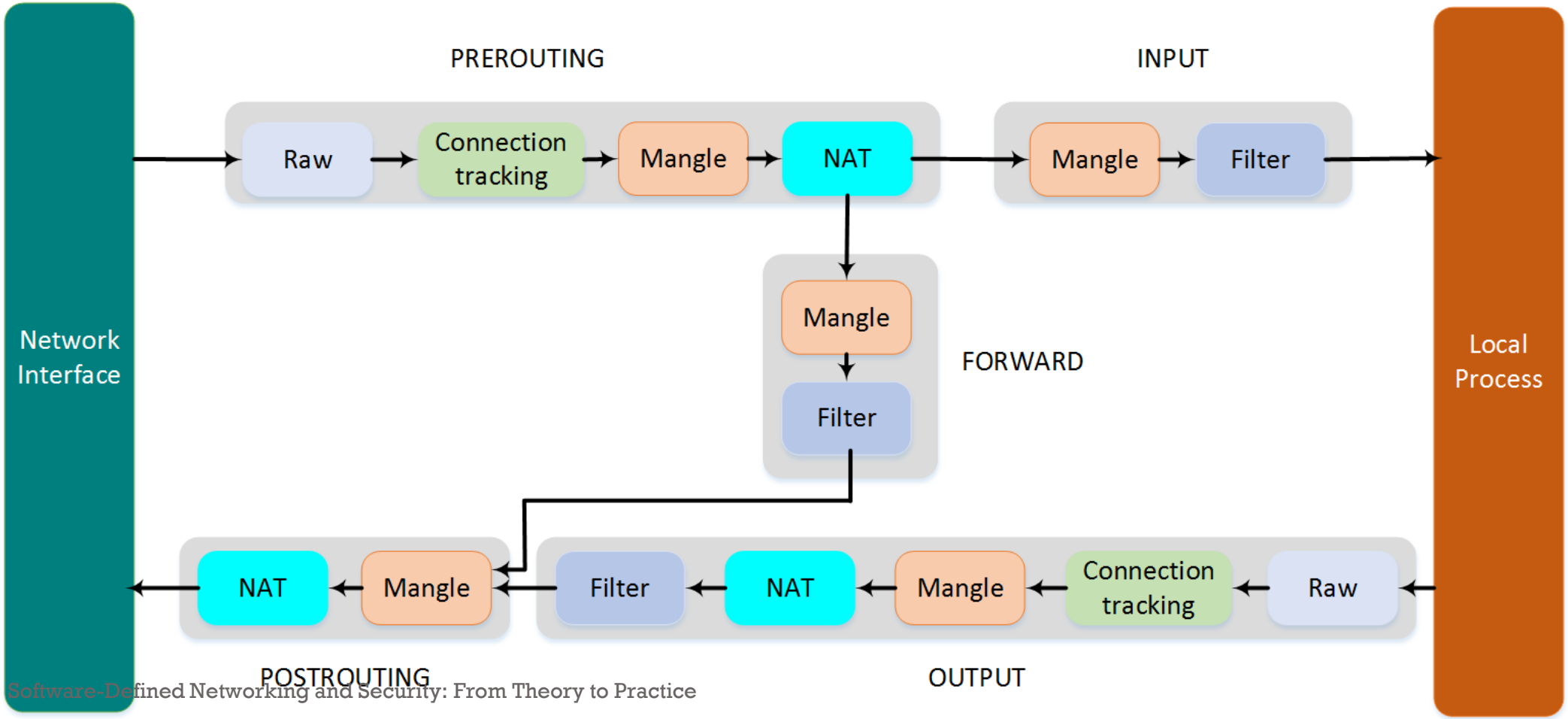


IPTABLES – TABLES AND CHAINS

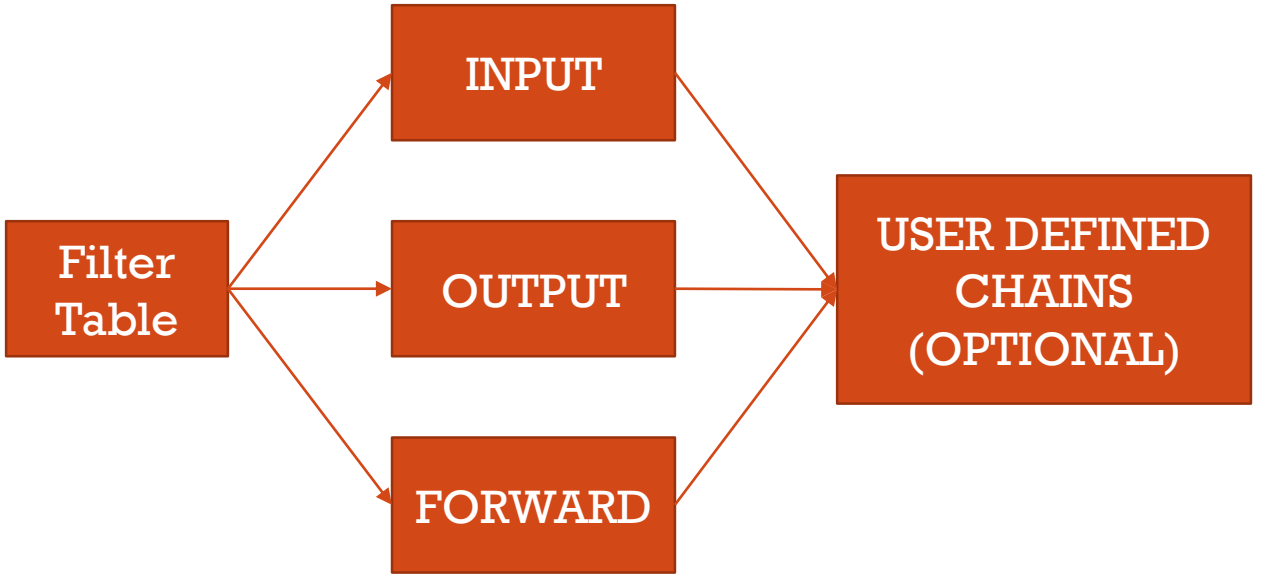
- Under each table, there are a set of **chains**.
  - Under each chain, you can assign a set of **rules**.



PACKET PROCESSING



FILTER ACTIONS

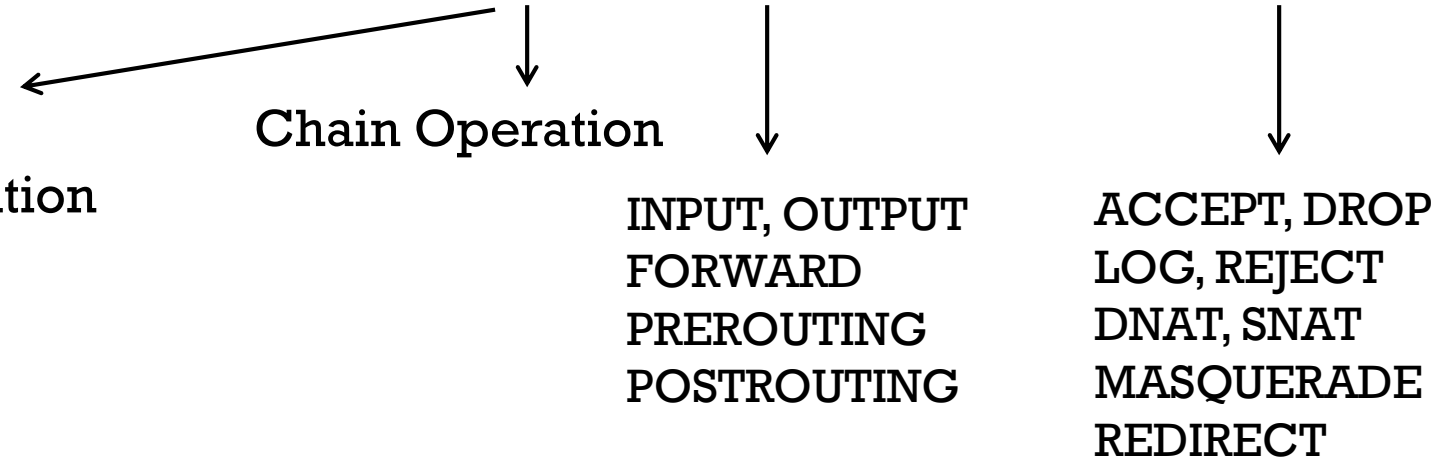


EXAMPLES

- INPUT
  - iptables -A INPUT -s 0/0 -i eth0 -d 192.168.1.1 -p TCP -j ACCEPT
  - iptables -A INPUT -p icmp --icmp-type echo-reply -j ICMP-RULE
- FORWARD
  - iptables -A FORWARD -s 0/0 -i eth0 -d 192.168.1.58 -o eth1 -p TCP -sport 1024:65535 -dport 80 -j ACCEPT
- OUTPUT
  - iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT

IPTABLES COMMAND STRUCTURE

```
iptables -t <table> operation chain specs -j target
```



- *Specs:* specify packet characteristics for matching
  - If the packet does not match the specification the packet is handed off to the next rule in the chain
  - If the packet meets the specification then the rule is passed to the `target`

RULE OPERATIONS

- `-I` Add a rule to the head of a chain
- **`-A`** Appends a rule to the tail of a chain
- `-D` Deletes a rule that matches the specifiers
- `-R` Replaces a rule in a chain

Syntax

```
iptables -t table -OP chain specifiers
iptables -t table -OP chain line# specifiers
iptables -OP chain specifiers
```


CHAIN OPERATIONS

- Listing a chain

```
iptables -t <table> -L chain
iptables -L chain
iptables -L
```

- Flushing a chain

Deletes all rules associated with a chain

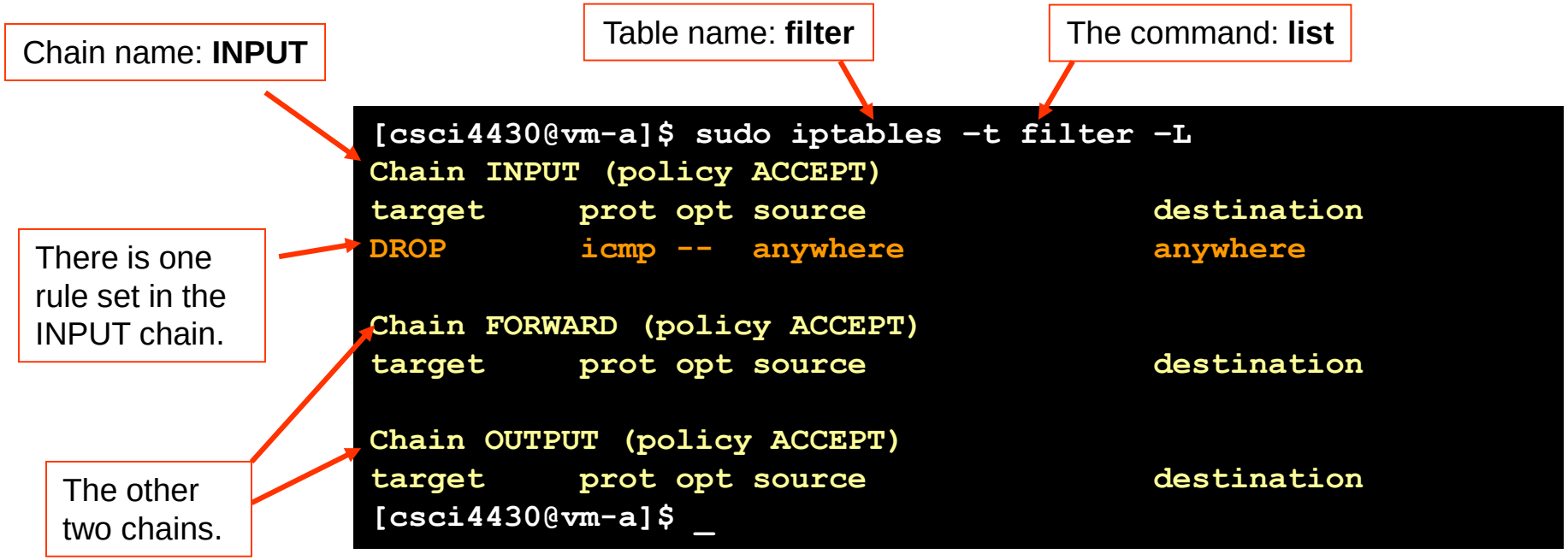
```
iptables -t table -F chain
iptables -F chain
iptables -F
```

- Setting the default policy of a chain (filter)

```
iptables -P chain policy
```

policy → *DROP, ACCEPT, REJECT, …*

IPTABLES – TABLES AND CHAINS



The rule in the INPUT chain means:

When a packet with ICMP payload passes through the **INPUT hook**, **DROP** that packets, no matter it is **from anywhere** and **to anywhere**.

IPTABLES – TABLES AND CHAINS

```
[csci4430@vm-a]$ sudo iptables -t filter -A INPUT --protocol icmp --jump DROP
[csci4430@vm-a]$ sudo iptables -t filter -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[csci4430@vm-a]$ _
```

This entry shows that a new rule is added to the INPUT chain of the filter table successfully.

Add a new rule to the INPUT chain.

The **protocol** of the packets in which this rule is interested is **ICMP**.

If a packet
(1) passes through the INPUT hook, and
(2) is an ICMP packet,

then the packet **jumps to the target DROP** – **to discard the packet**.

NAT RULES SET UP

- Your private network can "access" CUHK network and itself only.

```
[cs@vm-a]$ sudo iptables -t nat -A POSTROUTING \
               -s 10.0.0.0/24 -d 137.189.0.0/16 \
               -j MASQUERADE
```

- Your private network can only **use SSH** to reach the outside world!

```
[cs@vm-a]$ sudo iptables -t nat -A POSTROUTING \
               -p tcp -d ! 10.0.0.0/24 --dport 22 \
               -j MASQUERADE
```


EXCISE: FILTER ICMP PACKETS

- Please do the following task:
  1. Make a SSH connection to VM a and VM b with putty
  2. Flush all existing rules at VM a
     - `sudo iptables -t filter -F`
     - `sudo iptables -t nat -F`
     - `sudo iptables -t mangle -F`
  3. Ping VM a at VM b in normal case
     - `ping -c 1 10.0.<vm_group_id>.1`
  4. Check /proc/sys/net/ipv4/ip_forward at VM a
     - `cat /proc/sys/net/ipv4/ip_forward`
     - If the value is not '1', `sudo nano /proc/sys/net/ipv4/ip_forward`
  5. Filter ICMP packet from VM b to VM a at VM a
     - `sudo iptables -A INPUT -p icmp -s 10.0.0.2 -d 10.0.0.1 -j DROP`
  6. Ping VM a at VM b again, you will find it can not work
     - `ping -c 1 10.0.0.1`
  7. Capture the result of ping at VM b, and save it as "capture1.png"

Chain Operations (for User Chain)

- Creating a user chain

```
iptables -t table -N chain
iptables -N chain
```

- Deleting a user chain

```
iptables -t table -X chain
iptables -X chain
iptables -X
```

- Renaming a user chain

```
iptables -t table -E old new
iptables -E old new
```


MORE IPTABLES COMMANDS

SPECS: PACKET CHARACTERISTICS

- Protocol
- Source IP
- Destination IP
- Input Interface
- Output interface
- Frag flag

- TCP Datagrams
  - Src port
  - Dst port
  - Flags
  - TCP options
- UDP Datagrams
  - Src port
  - Dst port
- ICMP Messages
  - Type and code

PROTOCOL FIELD

- Syntax: **`-p | --protocol [!] [<protocol>]`**
- Protocol name: `tcp,udp,icmp`
- Protocol number as list in `/etc/protocols`
  - all (0), icmp (1), tcp (6), udp (17), …
- Examples
  - `--protocol 0`
  - `-p tcp,udp`
  - `-p ! udp`
  - `--protocol icmp`

ICMP TYPE AND CODE

RFC 792

- **Syntax: `-p icmp –icmp-type <type>`**
- **Type (code):**
  - `echo-request (0)`
  - `echo-reply (8)`
  - `destination-unreachable (3)`
  - `source-quench (4)`
  - `time-exceeded (10)`
  - `redirect (5)`

SOURCE/DESTINATION IP ADDRESS

- Syntax:

```
-s | --source | --src [!] <address>[/mask]
-d | --destination | --dst [!] <address>[/mask]
```

- Examples
  - `-s 1.2.3.4`
  - `-s 192.168.0.1/255.255.255.0`
    - IP address/network mask (Specifies a range of IP addresses)
  - `-s 192.168.0.0/24`
    - Specifies a range of addresses (192.168.0.0 – 192.168.0.255)
  - `-s ! 10.0.0.0/8`
    - Everything except `10.0.0.0-10.255.255.255`

INTERFACE

- `-i` <Input interface>
  - Only in INPUT, FORWARD, PREROUTING chains
    - `-i eth0`
    - `-i ! eth0` except eth0
    - `-i eth+` all ethernet interfaces
    - `-i lo` loop back interface
- `-o` <Output interface>
  - Only in OUTPUT, FORWARD, POSTROUTING chains

FRAGMENT

- Specifies second and additional fragmented packets. The negated version of this specifies unfragmented packets.
- Syntax:

  `[!] -f | --fragment`

- Examples
  - `-f` frag flag is set
  - `! -f` frag flag is not set

PORT SPECS

- Syntax

  --source-port || --sport [!] <port>[:<port>]

  --destination-port || --dport [!] <port>[:<port>]

- examples
  - `-p tcp --sport 80`
  - `-p udp --dport 53`
  - `-p tcp,udp --sport 0:1023`
  - `-p tcp,udp --sport 1024`
  - `-p tcp,udp --dport 1024:`

TCP FLAGS

- `-p tcp –tcp-flags SYN,ACK,FIN SYN`
  - **Tests SYN, ACK, FIN flags to see if the SYN bit is the only flag set**
  - **Possible flags**
    - `ACK`
    - `FIN`
    - `RST`
    - `PSH`
    - `SYN`
    - `URG`

SYN

- Tests `tcp` packets for `SYN` to be set and `ACK` and `FIN` not set
  - `-p tcp --syn`

    Filters all packets requesting `tcp` connection (new connection)
  - `-p tcp ! --syn`

CONNECTION STATE

- `-m state --state <state-specifier>`
- State-specifiers
  - `NEW` Associated with a new connection request
  - `ESTABLISHED` Associated with an established connection
    - `RELATED` Associated with a new secondary connection request related to an established connection (`ftp, icmp`)
  - `INVALID` Associated with a bad connection or is malformed
- examples:
  - iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
  - iptables -A OUTPUT --out-interface <interface> -p udp -s $IPADDR --source-port 53 -d $NAME_SERVER --destination-port 53 -m state --state NEW,RELATED -j ACCEPT

RATE LIMITS

- **-m limit --limit *<rate/unit>***
- Rate/unit
  - `rate:` Packets per unit time
  - `Unit:` `Second, minute, hour, day`
- **-m limit --time-burst *number* --limit *<rate/unit>***
  - `number` – max permitted burst before rate limit is applied
- Examples
  - iptables -A INPUT -i eth0 -p icmp --icmp-type echo-request -m limit
    --limit 1/second -j ACCEPT
  - iptables -A INPUT -i eth0 -p icmp --icmp-type echo-request -j DROP

TARGETS/ACTIONS

- Target types
  - Firewall actions – filter table chains & user defined
    - `ACCEPT, DROP, REJECT, LOG, RETURN`
  - NAT support
    - `DNAT, MASQ, REDIRECT, SNAT`
  - Uncommon targets
    - `MARK, MIRROR, QUEUE, TOS, TTL, ULOG`

FIREWALL ACTIONS

- `iptables operation specification` **`-j target`**
  - If the packet does not match the specification the packet is handed off to the **next rule** in the chain
  - If the packet meets the specification then the rule is passed to the **`target`**

FIREWALL ACTIONS

- `-j ACCEPT`
  - Lets the packet satisfying the specification **pass** to the next chain in the packet path
- `-j DROP`
  - The packet satisfying the specification is **dropped with no error packet sent to the sender**

FIREWALL ACTIONS

- `-j REJECT`
  - The packet satisfying the specification is dropped **with an error packet sent to the sender**
  - `-j REJECT` default error is **port unreachable**
  - `-j REJECT --reject-with` *flag*
    - *icmp-net-unreachable*
    - *icmp-host-unreachable*
    - *icmp-port-unreachable*
    - *Icmp-proto-unreachable*
    - *icmp-net-prohibited*
    - *icmp-host-prohibited*
    - *tcp-reset*
    - *Sends a tcp packet with the `RST` bit set*

FIREWALL ACTIONS

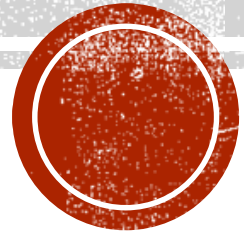
- `-j LOG`
  - Causes the packet satisfying the specification to be logged using the **Syslog** facility
  - `--log-prefix "IPT description of entry"`
    - IPT identifies the source of the log entry, i.e. Iptables
    - Description within quotes is limited to 29 characters
  - `--log-ip-options`
  - `--log-level`
  - `--log-tcp-options`
  - `--log-tcp-sequence`
  - To log a dropped packet a log rule must precede the dropping rule

FIREWALL ACTIONS

- `-j user-chain-name`
  - Lets the packet satisfying the specification **pass to the named user chain**
- `-j RETURN`
  - Used in the user chain to **return** to the calling chain

COMPUTER NETWORKING SERVICES

Iptables/NAT



NETWORK ADDRESS TRANSLATION (NAT)

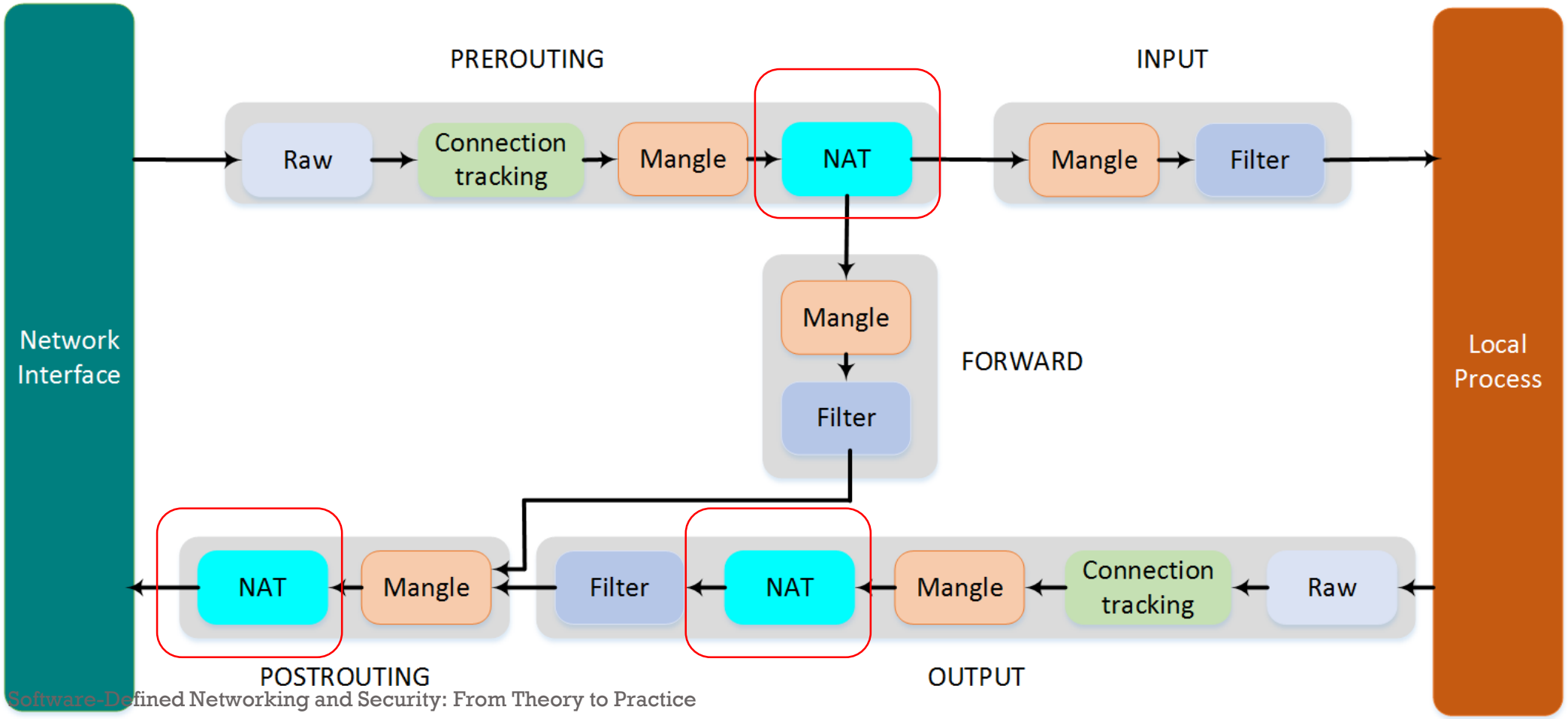
- Some important characteristics of how most organizations use the internet
  - Most hosts are client
  - Few hosts access the internet simultaneously
  - Internet communications are routed
- Network Address Translation
  - RFC 1631, in May 1994
  - A basic implementation of NAT involves
    - Using one of the private addresses for local networks
    - Assigned one or more public IP addresses
  - The word 'translator' refers to the device that implements NAT

PRIVATE ADDRESS SPACE

- Private addresses space defined by RFC1918
  - 24-bit block (Class A)
    - 10.0.0.0/8
  - 20-bit block (16 contiguous Class B)
    - 172.16.0.0/12 ~ 172.31.0.0/12
  - 16-bit block (256 contiguous Class C)
    - 192.168.0.0/16 ~ 192.168.255.0/16

- Operation consideration
  - Router should set up filters for both inbound and outbound private network traffic

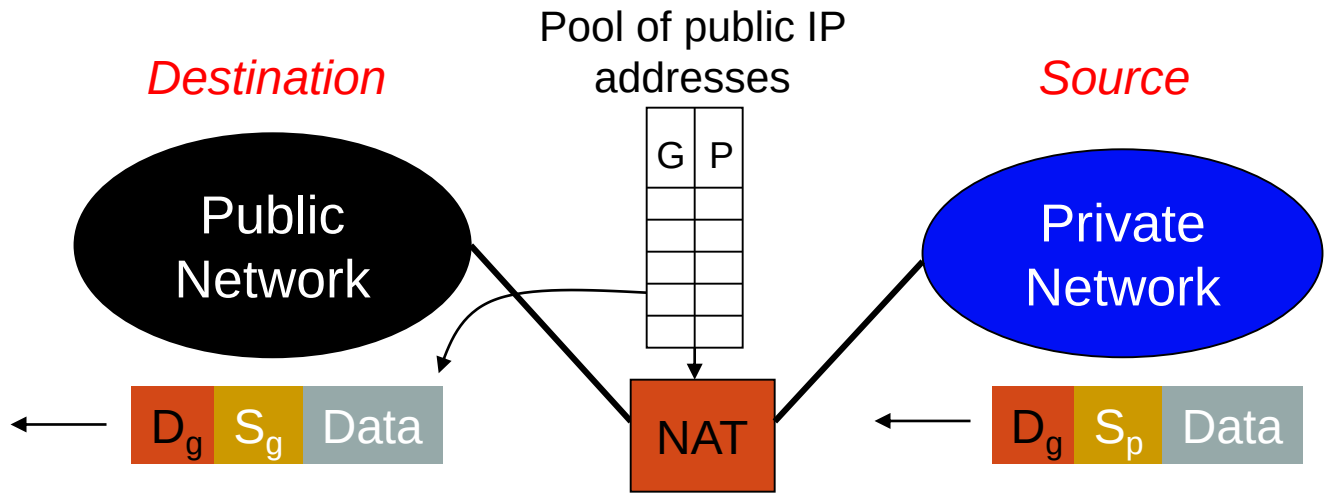
NAT – IPTABLES' VIEW



NAT ACTIONS – IPTABLES' VIEW

- DNAT: rewrite the **destination IP** address of the packet
  - Iptables -t nat -A PREROUTING -p TCP -i eth1 -d $EXT_IP --dport 80 -j DNAT --to-destination $INT_HTTP_IP
- SNAT: rewrite the **source IP** address of the packet
  - iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source $INT_IP
- MASQ: rewrite the **source IP** address of the packet to the IP address of *firewall's outgoing interface*
  - iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

NAT ILLUSTRATION



- **Private IP Addresses**
  - 10.0.0.0/8
  - 172.16.0.0/12
  - 192.168.0.0/16

- **Operation: Source (S) wants to talk to Destination (D):**
  - Create S_g-S_p mapping
  - Replace S_p with S_g for outgoing packets
  - Replace S_g with S_p for incoming packets

NAT ADDRESS MAPPINGS

- Each time a NAT router encounters an IP datagram
  - It must translate addresses
  - BUT, how does it know what to translate, and what to use for the translated addresses

- Translation table
  - Maps the inside local address to the inside global address
  - Also contains mappings between outside global address and outside local address for inbound translations

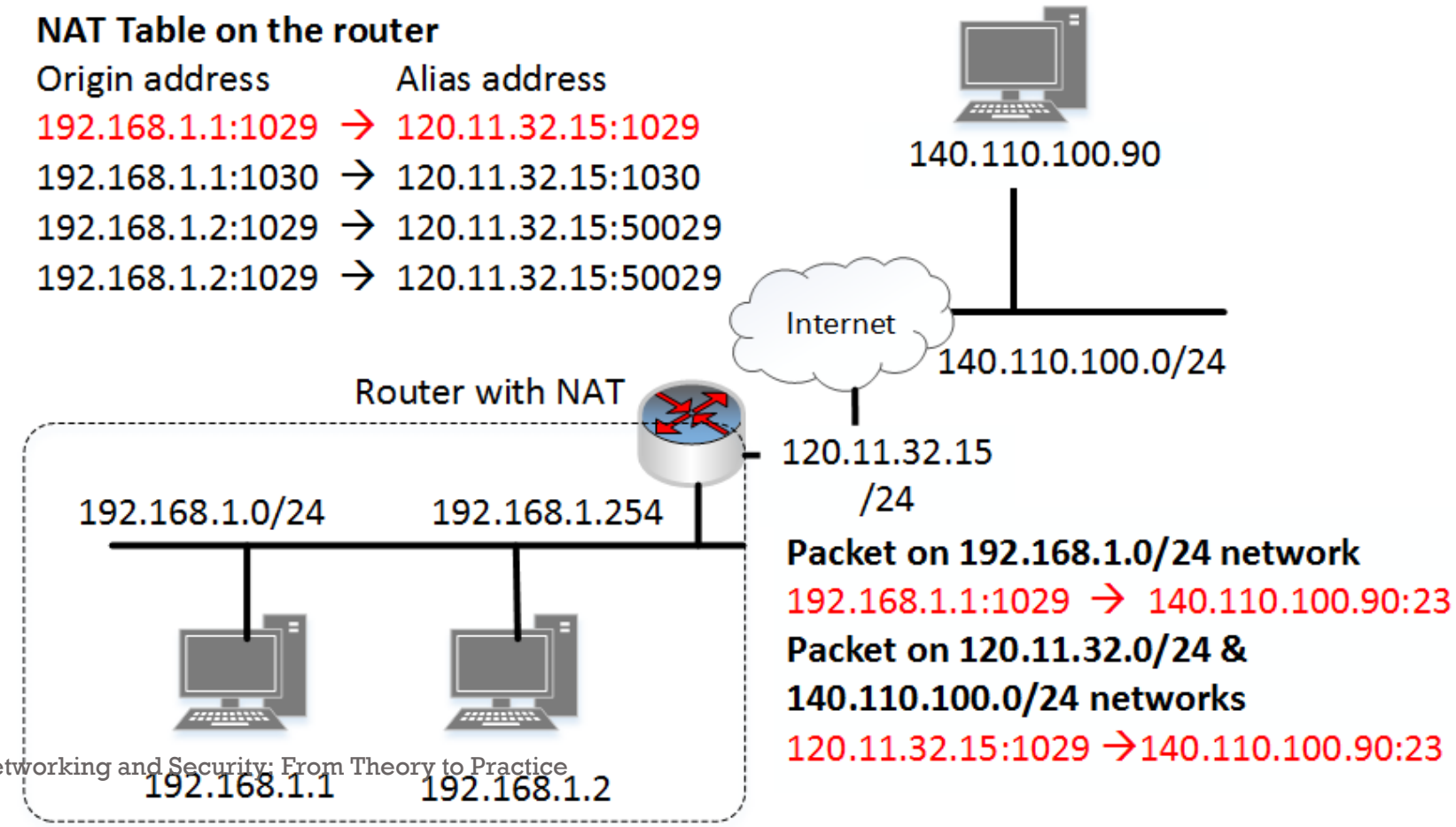
- Two address mappings
  - Static mappings
    - Allow the inside host with an inside local address to always use a inside global address
  - Dynamic mappings
    - Allow a pool of inside global addresses to be shared by a large number of inside hosts

NAT PORT-BASED OPERATION

- NAT Port-Based Operation
  - Overloaded operation
  - Network Address Port Translation (NAPT)/Port Address Translation (PAT)
  - Both traditional NAT and bidirectional NAT work by swapping inside network and outside network addresses
    - One-to-one mapping between inside local address and inside global address
    - Use dynamic address assignment to allow a large number of private hosts to share a small number of registered public addresses
- Using ports to multiplex private addresses
  - Also translate port addresses
  - Allow 250 hosts on the private network to use only 20 IP address
  - Overloading of an inside global address

NAT PORT-BASED OPERATION

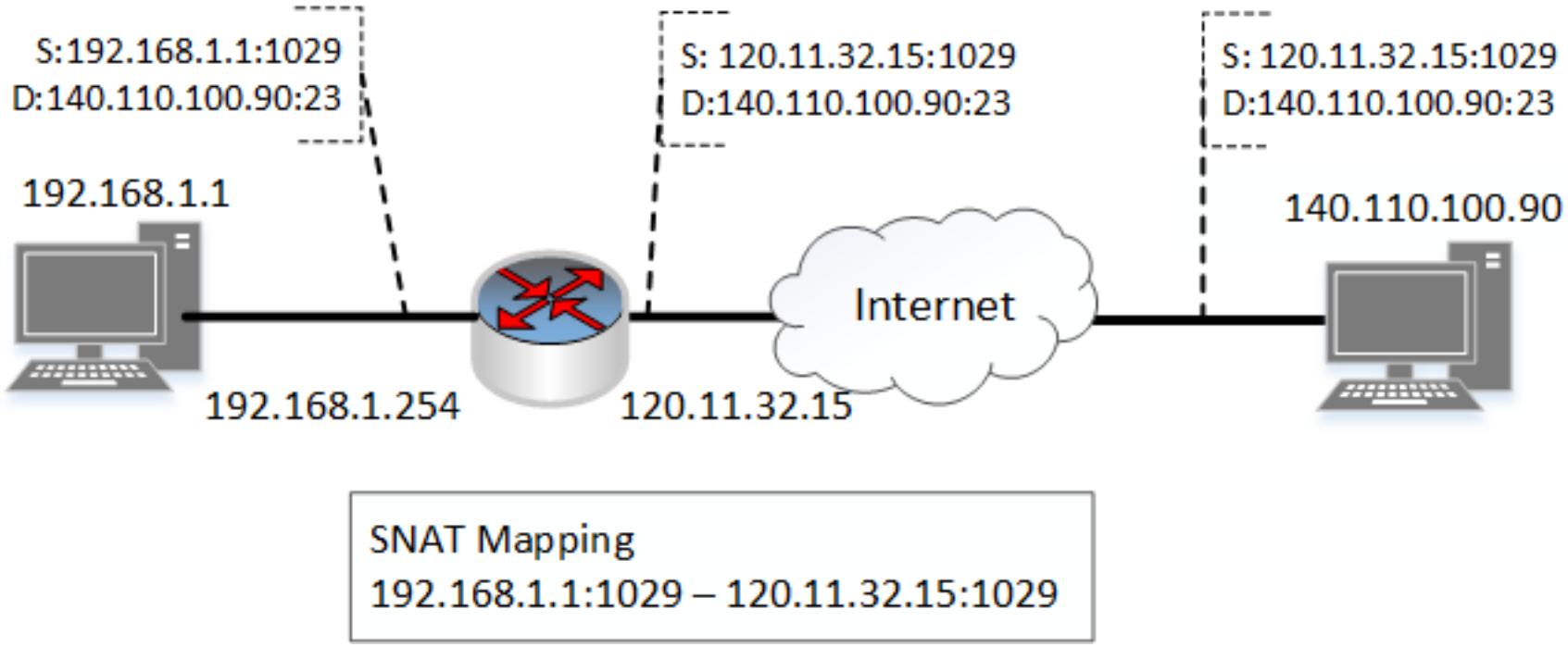
- NAT example:



NAT OVERLAPPING OPERATION

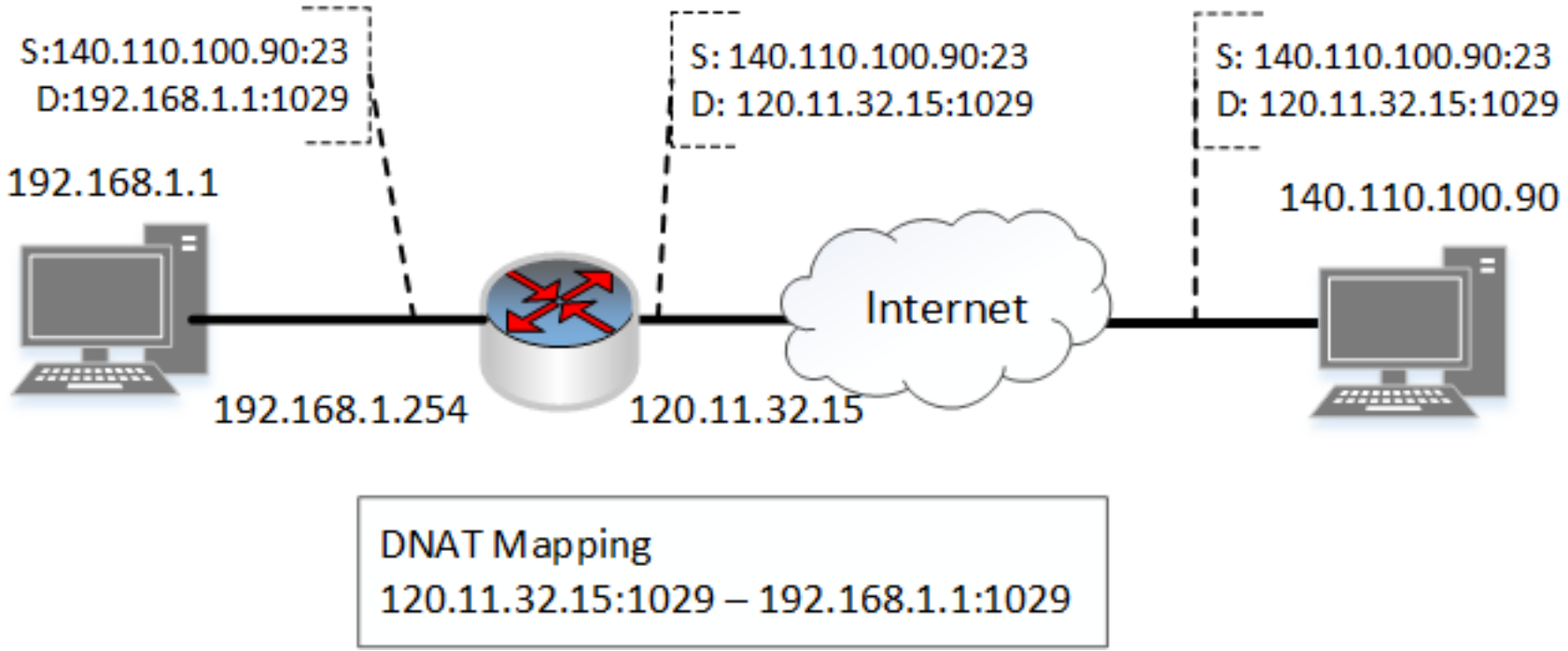
- NAT Overlapping Operation
  - Twice NAT Operation
  - The previous three versions of NAT are normally used to connect a network using private, non-routable addresses to the public internet
    - No overlap between the address spaces of the inside and outside network
- Cases with overlapping private and public address blocks
  - Private network to private network connections
  - Invalid assignment of public address space to private network
- Dealing with overlapping blocks by using NAT twice
  - Translate both the source and destination address on each transition
  - Rely on use of the DNS
    - Let the inside network send requests to the overlapping network in a way that can be uniquely identified

SNAT



- SNAT & DNAT
  - S: Source D: Destination
  - SNAT
    - Rewrite the source IP and/or Port.
    - The rewritten packet looks like one sent by the NAT server.

DNAT



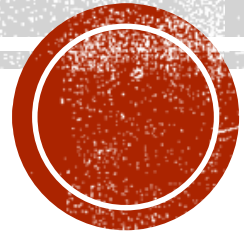
- DNAT
  - Rewrite the destination IP and/or Port.
  - The rewritten packet will be redirect to another IP address when it pass through NAT server.
- Both SNAT and DNAT are usually used together in coordination for two-way communication.

NAT QUESTION

- How to convert a public network to a NATed private network?
- How packets traversal through multiple NATed networks?

COMPUTER NETWORK ROUTING

Routing, Routing Table



CHECK NETWORKING SETUP

- Try these commands and what they do
  - route
  - netstate
  - ifconfig
  - ping
  - traceroute
  - nslookup
  - dig
  - host
  - arp

Reference site: https://gist.github.com/miglen/70765e663c48ae0544da08c07006791f

http://www.tldp.org/LDP/GNU-Linux-Tools-Summary/html/c8319.htm

CITE THIS WORK

```
@book{huang2018software,
title={Software-Defined Networking and Security: From Theory to Practice},
author={Huang, Dijiang and Chowdhary, Ankur and Pisharody, Sandeep},
year={2018},
publisher={CRC Press}}
```