Lecture 2
Network Fundamentals

Print version of the lecture in *CPS633 Computer Security*

presented on Week of September 10, 2019

by Ali Miri ©2019from Department of Computer Science at Ryerson University

2.1

Learning Objectives

- To become familiar to communication protocol stacks,
- To better understand computer different stack layer protocols such as TCP, UDP, ICMP, and IP
- To know different types network devices and their roles
- To discuss some of the inherent security of vulnerabilities of the current TCP/IP model in use

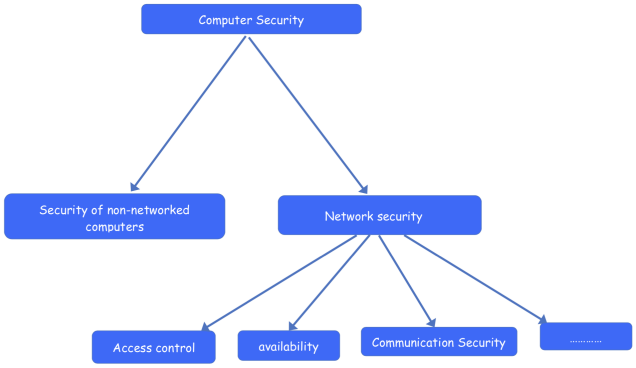
2.2

Contents

2.3

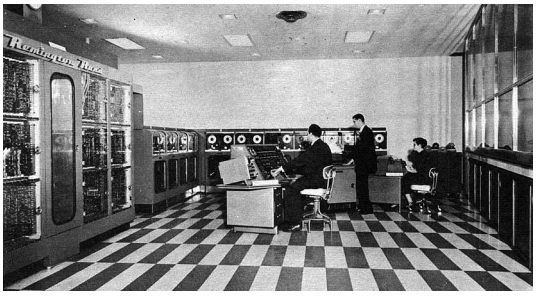
1 Preliminaries

1.1 Concepts



2.4

Need for Network Security



- Need for a (remote) user/machine to connect to a machine?

2.5

1.2 M2M Communication

Requirements

Need to consider:

2.6

Requirements

Need to develop communication protocols. Why?

2.7

2 Models

2.1 IP Model

DARPA Model

- Work in early 1970's, supported by the Defense Advanced Research Projects Agency (DARPA) led to
  - **Transmission Control Protocol(TCP)**
  - **Internet Protocol(IP)**
- Later these protocols were used to produce what is known as **TCP/IP model** (or more correctly *IP model*)

- It used *packet-switching* (vs. circuit-switching)
- Each packet has two major pieces: the *packet header* and the *data*
- *Packets:* Data units processed through a *multilayer Protocol*, commonly referred to as a *stack layer*. Data units are referred to by different names, depending on the layer.
- *Packet headers:* Later!

2.8

2.2 OSI Model

OSI Model

- Later, work by ISO (and in parallel by International Telecommunication Union) and industry resulted in the *Open Systems Interconnection model (OSI model)* $\Rightarrow$ *ISO 7498* and *ITU-T X.200*
- OSI model defines abstract layers for different functions of a communications system, and then develops protocols for each layer
- Each layer can establish a logical connection with a peer-layer on different computer, using the information contained in the packet. Packets also contain information to allow layers to interact with layers above and below $\rightarrow$ **Encapsulation**
- *Encapsulation:* Each stack layer adds its own header, and possibly a footer to data unit received before passing it on to the next layer, and (eventually) to the peer-layer.

2.9

OSI Model

OSI model				
Layer			Protocol data unit (PDU)	Function[6]
Host layers	7	Application	Data	High-level APIs, including resource sharing, remote file access
	6	Presentation		Translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption
	5	Session		Managing communication sessions, i.e. continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes
	4	Transport	Segment, Datagram	Reliable transmission of data segments between points on a network, including segmentation, acknowledgement and multiplexing
Media layers	3	Network	Packet	Structuring and managing a multi-node network, including addressing, routing and traffic control
	2	Data link	Frame	Reliable transmission of data frames between two nodes connected by a physical layer
	1	Physical	Symbol	Transmission and reception of raw bit streams over a physical medium

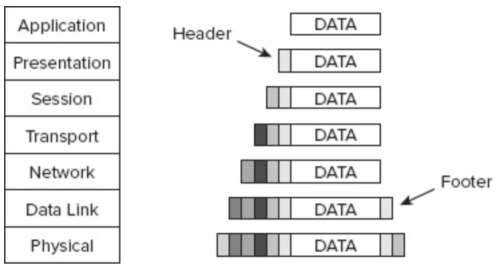
source:

https://en.wikipedia.org/wiki/OSI_model

2.10

2.3 Encapsulation

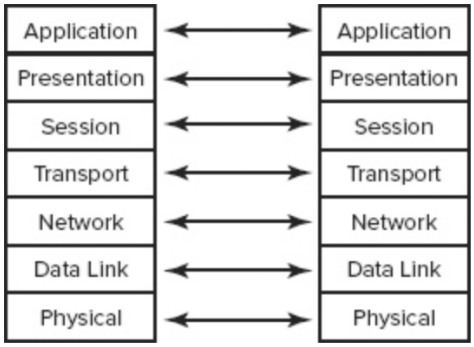
OSI Encapsulation



One Layer's Header (and Footer) is Another Layer's Data

2.11

OSI Encapsulation



Peer Layer Logical Channels

2.12

OSI Encapsulation

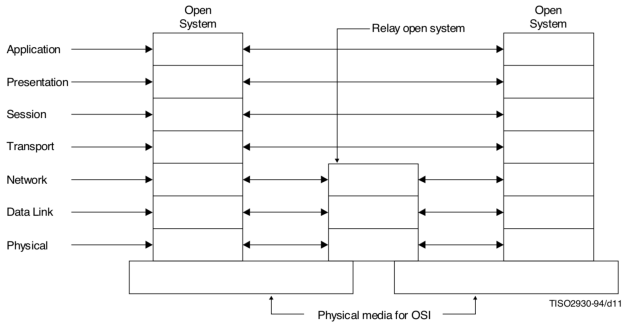


Figure 12 – Communication involving relay open systems

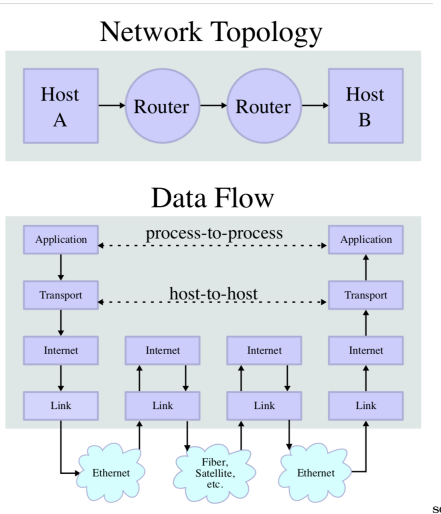
source:

`https://www.itu.int/rec/T-REC-X.200-199407-I`

2.13

TCP/IP Model

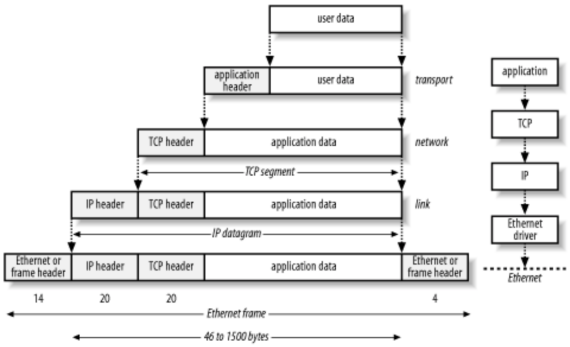
How about the IP model?



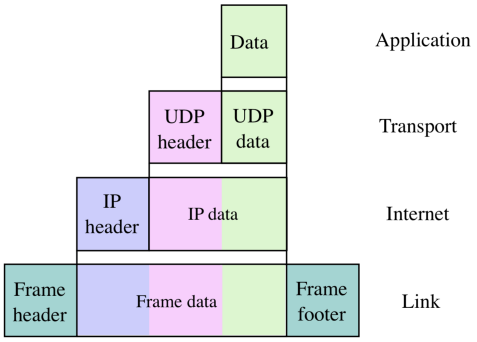
source: `https://en.wikipedia.org/wiki/Internet_protocol_suite`

2.14

TCP/IP Encapsulation



TCP/IP Encapsulation



Multilayer Protocols

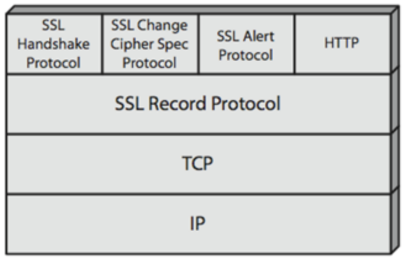
Some final comments on multilayer protocols

- It supports a large range of applications in higher level, and is easy to implement
  - For example, exchange of *http* over wired *Ethernet* can look like

    [Ethernet [IP [TCP [HTTP]]]]

- It may be possible to add additional (over-)layers of encapsulation.
  - For example, in the example above, we could introduce SSL/TLS encryption by adding a a new encapsulation between HTTP and TCP:

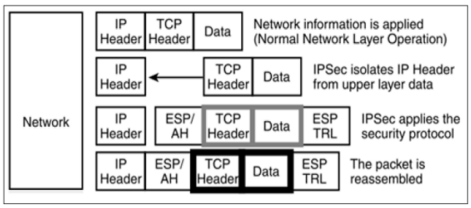
    [Ethernet [IP [TCP [SSL [HTTP]]]]]



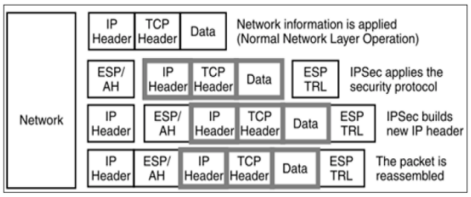
Multilayer Protocols

- or, add additional protection by introducing VPN with IPSec Network layer encryption

[Ethernet [IPSec [IP [TCP [SSL [HTTP]]]]]]



or



source:https://www.tutorialspoint.com/network_security/network_security_layer.htm

• This capability however can also be (mis-)used to create covert channels, or bypass and firewalls.

2.18

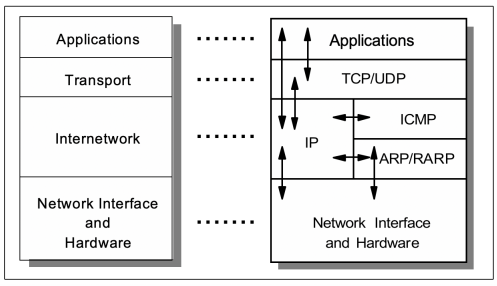
Wrap-up

Quick wrap-up

- The TPC/IP is the popular model in use, whereas the OSI model is used for analysis and development
- Both models are platform-independent protocol based on open standards, but also can utilize a fair amount of resources.
- Neither was designed with security in mind (!) → many security vulnerabilities to exploit!
- The two models sometimes are used interchangeably, so it is important to know which model is being addressed in various contexts ← (naming can help)

2.19

- For the reminder, we mostly focus on *TCP, UDP*, and *IP, ICMP, ARP/RARP*



The TCP/IP protocol stack - Each layer represents a package of functions

Source: IBM TCP/IP Tutorial and Technical Overview, 2001

2.20

3 Transport Protocols

3.1 TCP

Aside

Definitions: *Ports*

- An integer (an address number) that both ends of communication link must agree to use when transferring data ← this process is referred to as *listening on* or *binding to* that port
- The integer range from 0 to 65535, with port numbers 0 to 1023 reserved for privileged services and designated as *well-known* ports. Ports 1024 to 49151 are typically known as the *registered software* ports. Ports 49152 to 65535 are known as the *random*, *dynamic*, or *ephemeral* ports - this (recommended by IANA) is not universally followed by all OSs.
- Only one process per machine can listen on the same port number

2.21

Aside

Addressing:

- Every network device must have an address on the network
- Every physical Network Interface Card (NIC) has a unique 48-bit (or 64-bit) factory burned number, referred to as Media Access Control (MAC) address ← No two devices can have the same MAC address
- Every software process that communicates must be addressable
- A network device/node can use multiple type of addresses
  - MAC address (00:00:0C:1A:2B:C3)
  - IP address (141.117.126.20)
  - Machine name
  - Organizational/domain name

2.22

- Two main protocols at the Transport layer are TCP and UDP.
- When setting a data exchange, a number of questions may need to be considered:
  - What type of data is being transmitted? (How big of data, use, data loss impact, $\cdots$)
  - How reliable is the communication medium, and do we need guarantee that the data got to its destination?
  - Who does what, and the amount of acceptable overhead?

2.23

Two Scenarios

Two communication scenarios:

(a) `Someone right now walks to this class and start talking!! He hasn't checked to see if the class is taking a place, or giving a warning to the class that he is about to come. He hasn't even check to make sure that he has brought the right notes. Different parts of his talk also may be out of order, or never delivered. He also does not get an acknowledgement for (any prat of) his talk from the class (or the instructor).`

Sounds bad? But, there can be advantages!

2.24

Two Scenarios

(b) `The talk is important to the class - it has to be delivered correctly and in proper order. The class is noisy. The speaker will announce his plan to come to a given class, and the class/instructor will acknowledge. There are additional checks to make sure that different pieces of the talk are delivered, and are in the right order, and each piece delivered gets its own acknowledgement. This way a full, complete talk is given. The speaker even let the instructor know when he wants to end the talk, and gets an acknowledgement.`

Better? Sure, but there are overhead costs! 2.25

TCP

Transmission Control Protocol (TCP) - used with most commonly used transport protocols

- A connection-oriented reliable protocol ← it used three-way handshakes to establish connections
- It supports full-duplex communications
- It used the handshake process to establish communication '*sessions*' ← it is session-oriented.
- It has a large overhead.

2.26

3.1.1 Headers

TCP Headers

TCP headers are relatively complex (as compared with UDP's, discussed next). They are typically between 20-60 bytes long, and divided into the following field in order:

Size in Bits	Field
16	Source port
16	Destination port
32	Sequence number
4	Data offset
4	Reserved for future use
8	Flags
16	Window size
16	Checksum
16	Urgent pointer
Variable	Various options; must be a multiple of 32 bits

TCP Header Construction

2.27

TCP Header Construction

TCP Header

Offsets	Octet	0								1								2								3			
Octet	Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size														
16	128	Checksum																	Urgent pointer (if URG set)														
20 ...	160 ...	Options (if *data offset* > 5. Padded at the end with "0" bytes if necessary.) ...																															

source:

https://en.wikipedia.org/wiki/Transmission_Control_Protocol

2.28

TCP Headers Flags

Bit	Name	Description
CWR	Congestion Window Reduced	Used to manage transmission over congested links
ECE	ECN-Ech (Explicit Notification)	Used to manage transmission over congested links
URG	Urgent	Indicates urgent data
ACK	Acknowledgement	Acknowledges synchronization or shutdown reques
PSH	Push	Indicates need to push data immediately to application
RST	Reset	Causes immediate disconnect of TCP session
SYN	Synchronization	Requests synchronization with new sequencing numbers
FIN	Finish	Requests graceful shutdown of TCP session

TCP Header Flags

2.29

TCP Headers Flags

- Easy to set flags: just identify the corresponding bit in the header/flag, and turn it on or off by setting it to a 0 or 1 (the representation flag can be set to binary or hex).
- The first two flags are used for QoS and congestion control, and are rarely used.
- The last 6 are commonly used: **U**RG, **A**CK, **P**SH, **R**ST, **S**YN, and **F**IN.
- One suggested phrase to remember the order is to use "***U**nskilled **A**ttackers **P**ester **R**eal **S**ecurity **F**olk*"!

2.30

3.2 UDP

UDP

UDP: *User Datagram Protocol* or *Unreliable Datagram Protocol* (also called *connectionless protocol*) - 1980

- It is a connectionless "best-effort" communications protocol
- It does not check to see if the receiving machine is ready to receive data, or let it know that the data is about to arrive. Also, it does not check whether the data was received, or in the right order.
- Its size can be up to 65,535 bytes long, although typically much smaller
- It has no error detection or correction
- It does not used a pre-established session ← once the sender has send the UDP fragment, it forgets about it!
- It has a smaller overhead than TCP, and it used in many applications such as DNS, NTP, DHCP, streaming video, voice over IP, online game, etc., where they are query-based, or those with some loss of data can be tolerated.

2.31

3.2.1 Headers

UDP Headers

UDP Header

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port														
4	32	Length																Checksum														

source:https://en.wikipedia.org/wiki/User_Datagram_Protocol

2.32

4 IP Protocols

- Network layer protocols are needed when exchanges data between peers not on the same network.
- These protocols focus on various functions needed, such as
  - Discovering paths from source to destination ← *routing*
  - Node/path Node availabilities
  - Dynamic address configuration
  - Packet addressing
  - Resolution between network layer addresses and lower level addresses
  - (Add on) security services
  - ...

2.33

Examples of network layer protocols:

- Internet Control Message Protocol (ICMP)
- Address Resolution Protocol (ARP) and Reverse Address Resolution Protocol (RARP)
- Routing Information Protocol (RIP)
- Open Shortest Path First (OSPF)
- Border Gateway Protocol (BGP)
- Internet Group Management Protocol (IGMP)
- Internet Protocol (IP)
- Internet Protocol Security (IPSec)
- Internetwork Packet Exchange (IPX)
- Network Address Translation (NAT)
- Simple Key Management for Internet Protocols (SKIP)

2.34

4.1 IP

Internet Protocol (IP)

You tell me how to get there (IP), and I will get us there (TCP, or UDP)!

Internet Protocol (IP):

- Provides route addressing for data packets
- Is a connectionless, unreliable datagram service ← employ TCP on IP to gain reliable and controlled communication session
- Does not guarantee that packets will be delivered, flow control, or error recovery

2.35

4.1.1 Addressing

IP Addressing

IP Addresses (also known as *Internet addresses*
(We will mostly focus on IPv4)

- size: 32-bit unsigned binary value, typically represented as a *dotted decimal format* (example `23.145.76.1`)
- Used by the IP protocol to uniquely identify a host
- Each IP packet contains a *source IP address* and a *destination IP address*
- The destination IP address needs to be translated/mapped to a physical device

2.36

4.1.2 Routing

IP Routing

IP routing: The process of sending packets from a host on one network to another host on a different remote network

- A device can simultaneously function as both a normal host and a router.
- Usually done by routers, and based on routing tables that determines a next hop address to which the packet should be forwarded.
- Most networks have one or more *default gateways*: routers that hosts use to communicate with other hosts on remote networks

2.37

4.1.3 Headers

IP Packet Headers

- size: 20 bytes for IPv4 (40 bytes for IPv6), with the max size not to exceed 60 bytes (no max. value for IPv6)

IPv4 Header Format

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP						ECN		Total Length															
4	32	Identification																Flags			Fragment Offset											
8	64	Time To Live								Protocol								Header Checksum															
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															
24	192																																
28	224																																
32	256																																

source:https://en.wikipedia.org/wiki/IPv4

2.38

IP Packet Headers

Time-to-Live (TTL):

- Originally was introduced to avoid cycles (in routing)
- In IPv4, It was designed to specify time (in second) that datagram is allowed to travel
- Most routers process the packet in less than a second, however based on the protocol, each router subtracts 1 from the value of TTL, before forwarding it to the next hop.
- When TTL is equal 0, it is considered that it has traveled a closed loop, and it is discarded $\rightarrow$ essentially TTL can be used as a hop counter.
- The initial value of TTL is set by the higher level protocol that created the packet.

2.39

IP Packet Headers

Protocol Number: This field indicates the higher level protocol to which IP should deliver the data in this datagram.

List of some common protocol numbers

- 1: Internet Control Message Protocol (ICMP)
- 2: Internet Group Management Protocol (IGMP)
- 4: IP (IP encapsulation)
- 6: Transmission Control Protocol (TCP)
- 17: User Datagram Protocol (UDP)

For full list of protocols, see `https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml`

2.40

4.1.4 Vulnerabilities

TCP/IP Vulnerabilities

TCP/IP has numerous security vulnerabilities:

- Sniffing
- Spoofing
- Man-in-the-middle attacks
- SYN flood and various DoS attacks
- Fragment attacks
- ...

2.41

4.2 ICMP

ICMP

Internet Control Message Protocol (ICMP) is used to determine the health of a network or a specific link

- It used IP as if ICMP were a higher level protocol, i.e ICMP messages are encapsulated in IP datagrams
- The ICMP header starts after the IP header and is identified by IP protocol number '1'
- All ICMP packets have an 8-byte header and variable-sized data section. The first 4 bytes of the header have fixed format, while the last 4 bytes depend on the type/code of that ICMP packet

ICMP Header Format

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Type								Code								Checksum															
4	32	Rest of Header																															

source:`https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol`

2.42

4.2.1 Types

ICMP Types/Codes

2.43

4.2.2 Ping

Ping

Two simple and widely used applications based on ICMP are *Ping* and *Traceroute*.

- Ping sends IP packets to a specified destination host and measures the round trip time to receive a response.
- It can indicate '*reachability*', and time to do so.
- *Echo*, and *Echo reply* ICMP type are used by Ping to detect whether a host is active in a network.

2.44

Table 1: Common ICMP type field values

Type	Function
0	Echo Reply
3	Destination Unreachable
5	Redirect
8	Echo
11	Time Exceeded
30	Traceroute

source:https://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml

4.2.3 Traceroute

Traceroute

Traceroute is used to determine the route IP datagrams follow through the network.

Assumption: All hosts on the same network must go through the same external router

- It shows a path a packet can take to reach its destination. It does that by sending a series of all going to the same destination but with TTL values starting at 1. This will result in an ICMP *'Destination Unreachable* or *''Time Exceeded* message, until one the packets if finally is delivered to the destination host. → Traceroute keep track of all routers in the path, which can be used to map a network
- On Unix and Mac OS, it uses UDP packets, whereas on Windows it uses ICMP packets

2.45

4.2.4 ARP

Address Resolution Protocol (ARP)

Address Resolution Protocol (ARP) and *Reverse Address Resolution Protocol (RARP)* are used to interoperability of logical and physical addressing schemes. In a single physical network. Higher level protocols address destination hosts in the form of a symbolic address, typically IP addresses. ARP and RARP provide translations between these two.

ARP (and RARP similarly) achieves its translation task by checking a lookup table - '*ARP cache*' to find physical address to match to an IP address. If the address is not found in the ARP cache, it sends a broadcast message, called 'ARP request' to find a match.

This process can be abused by what is known as *ARP cache poisoning*, where an attacker inserts incorrect information into the ARP cache.

2.46

5 devices

Network Devices

Layers 1, 2 and 3 network devices:

- **Hubs:** A layer 1 device that repeats data it receives on one port to all other systems on its other port → Because the way it operates, it makes is really easy to 'sniff' the traffic!

- **Bridge:** A layer 2 device that connect two physical segments of a network, i e it create a table that maps MAC addresses to network segments. It sends the message to ALL devices on the segment Bridge (similar to a hub) will forward frames anyway, even if it is not in its table!
- **Switch:** A mixture of hub and bridge (a layer 2 device), keeps track of MAC addresses attached to each of it's port. Like a hub it can retransmit data to multiple ports. By keeping MAC addresses it has capabilities of a bridge. → Make sniffing less effective

2.47

Network Devices

Routers: user IP addresses to forward packets (layer3)

- it blocks traffic to unknown addresses, and block broadcast traffic
- It adds header and trailer (bridges or switches do not)
- It always has to queue traffic before forwarding (bridges and switches do not have to)
- Uses different network addressees off all ports (bridges and switches use the same network address for all ports!
- it builds tables based on network addresses (bridges and switches do it based on MAC addresses)

2.48