

# Networking

---

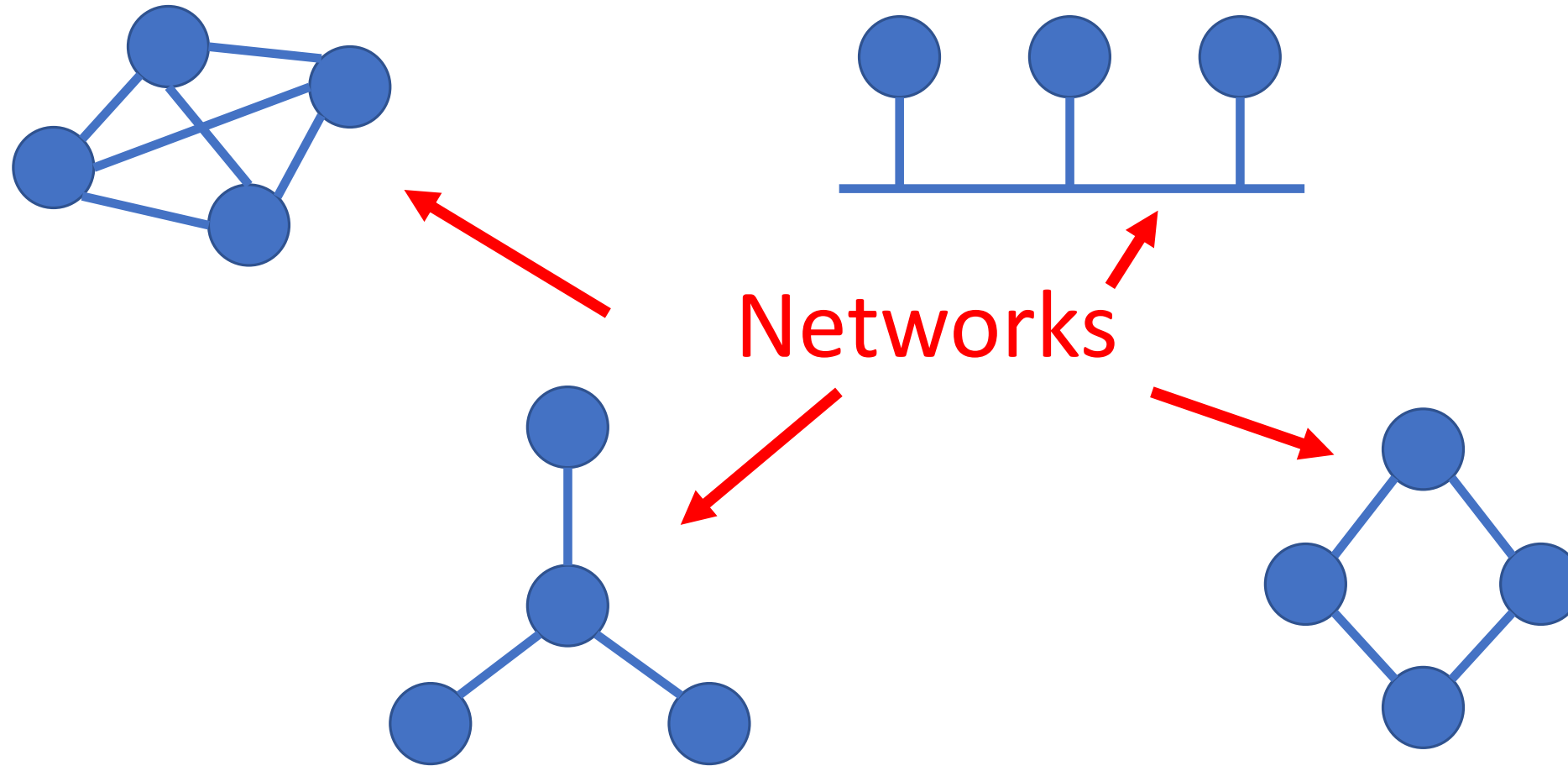
OSI MODEL, TOPOLOGIES, IPV4, IPV6, TCP, UDP

# Motivation

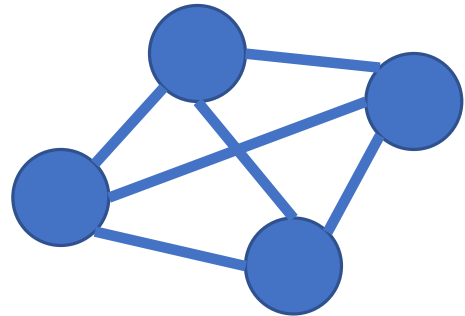
---

- Resource sharing across computing devices
- Need inter-operability to communicate
- Layered model
- "Competing" standards
  - OSI Model
  - Tanenbaum's TCP/IP 5-layer model
- In reality, these standards are equivalent, they just group protocols differently

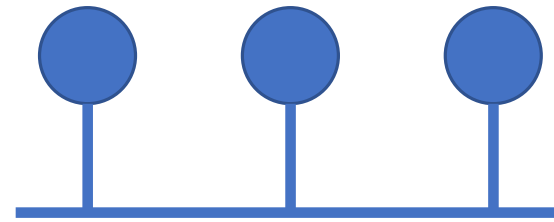
# Basic Concepts



# Basic Concepts

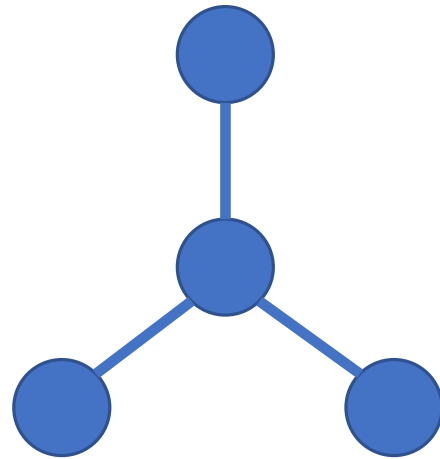


mesh

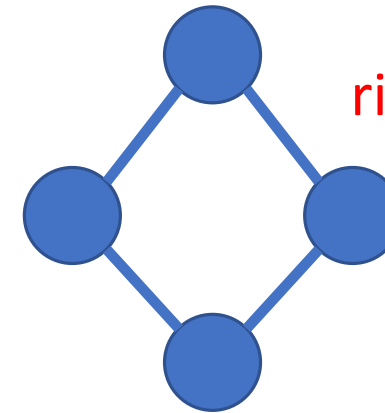


bus

## Topology

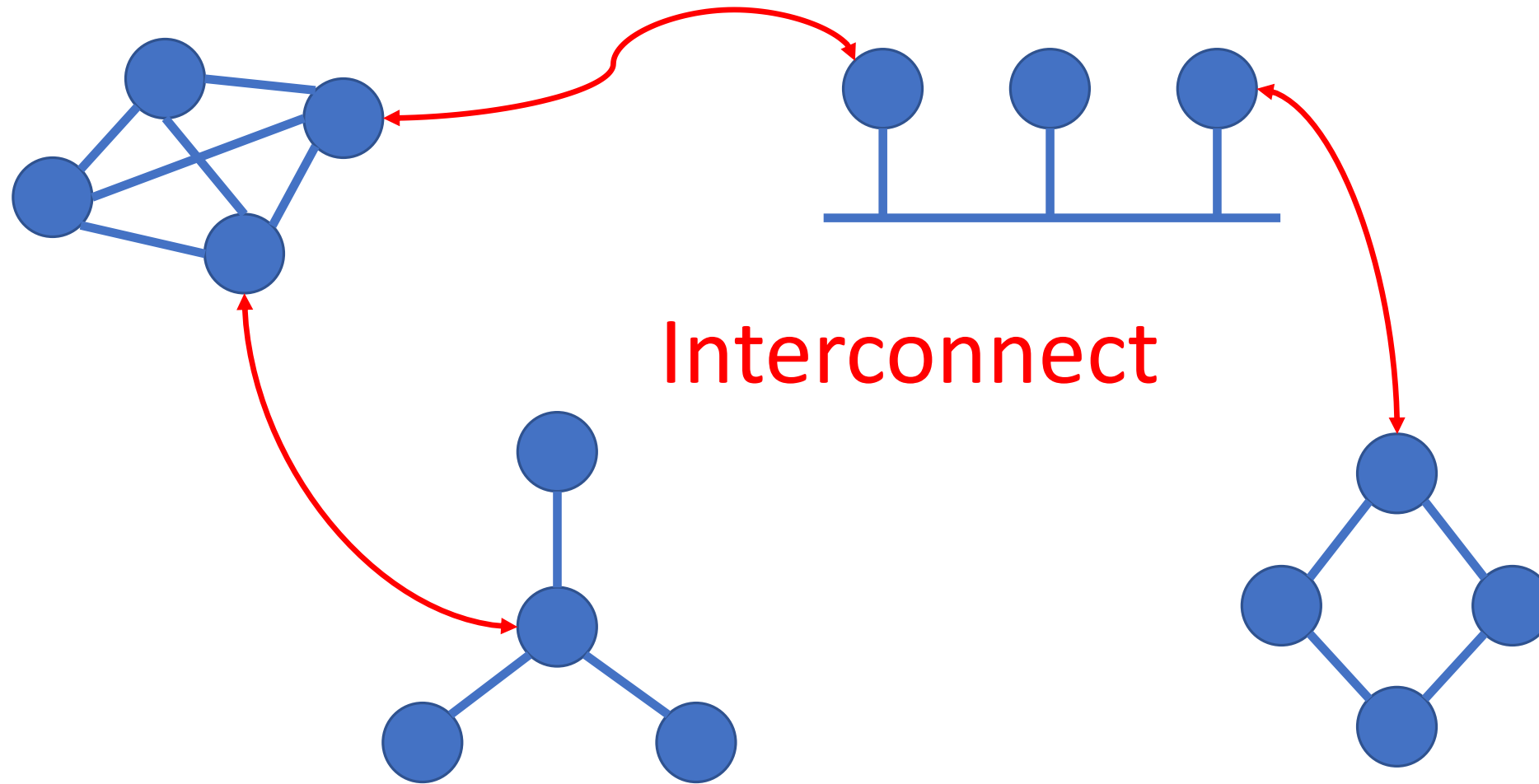


star



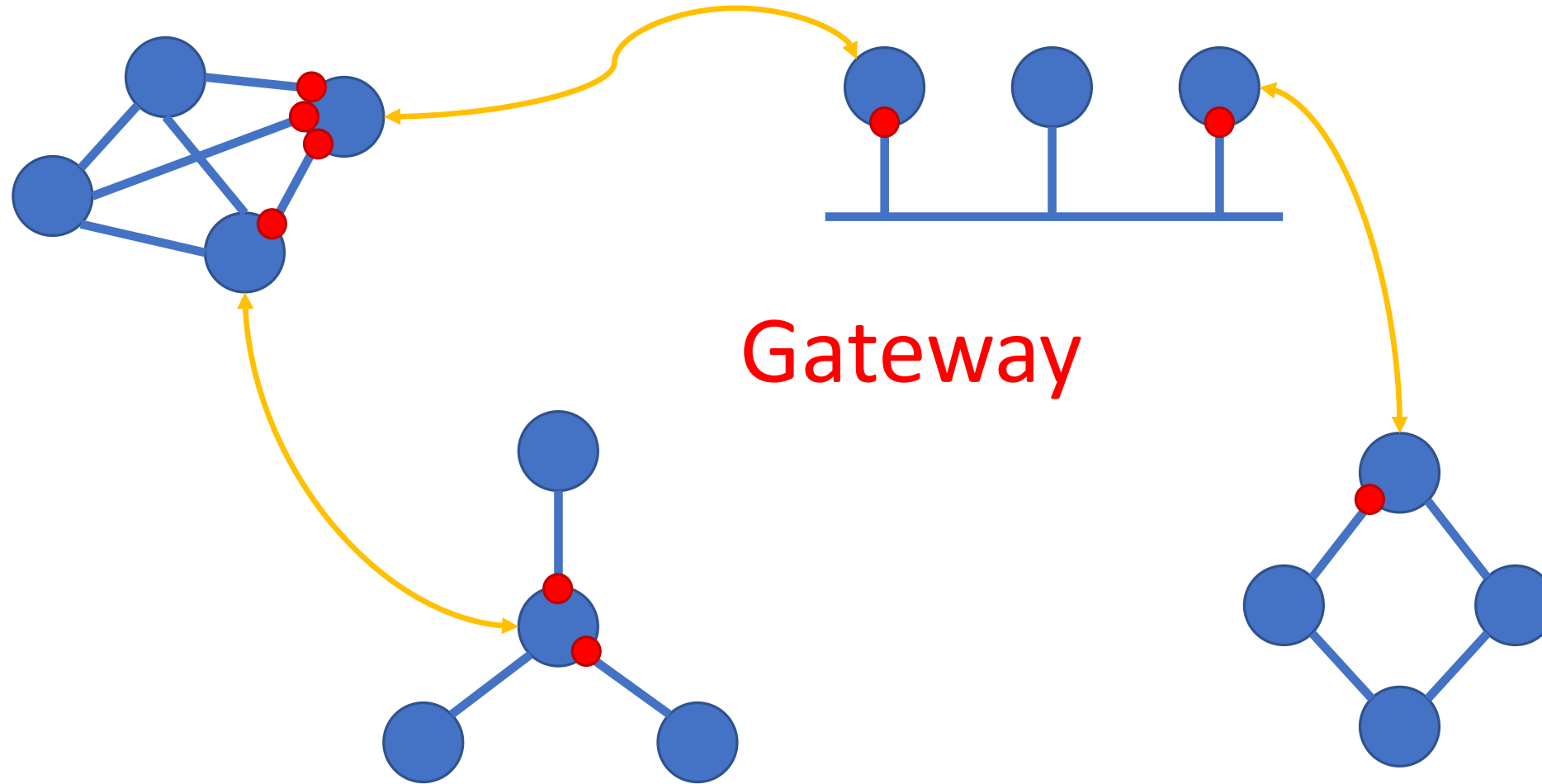
ring

# Basic Concepts

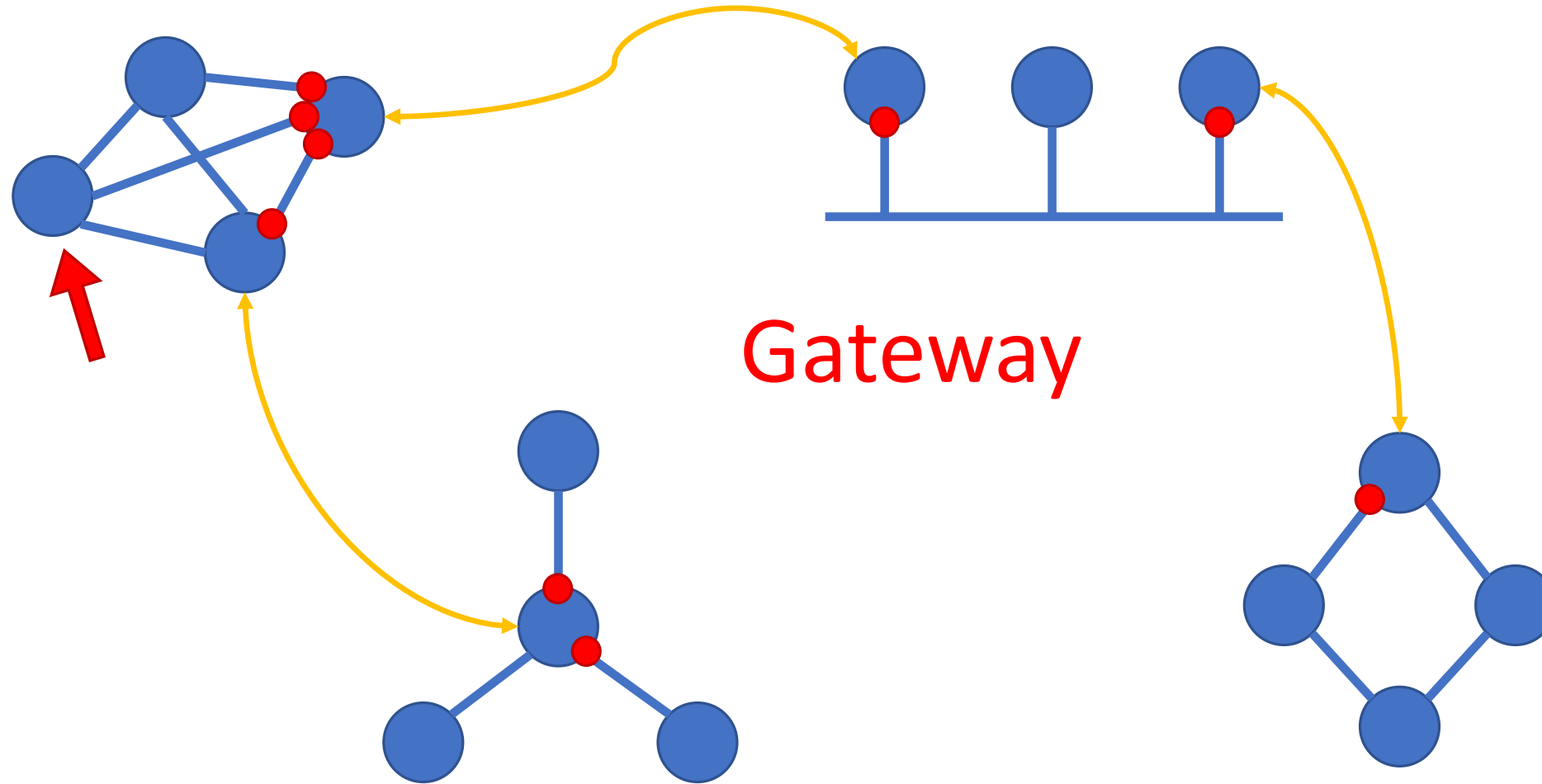


Interconnect

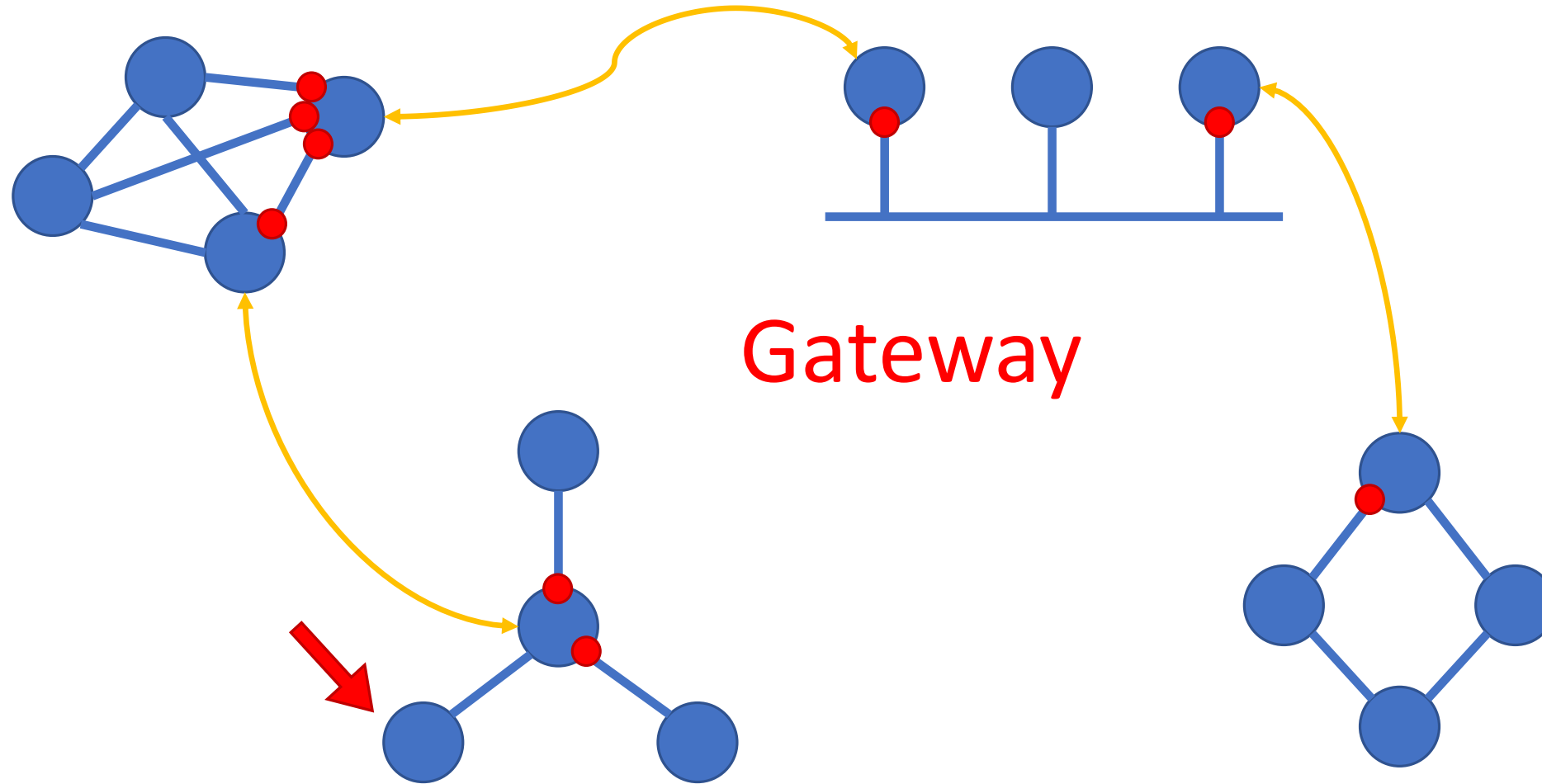
# Basic Concepts



# Basic Concepts

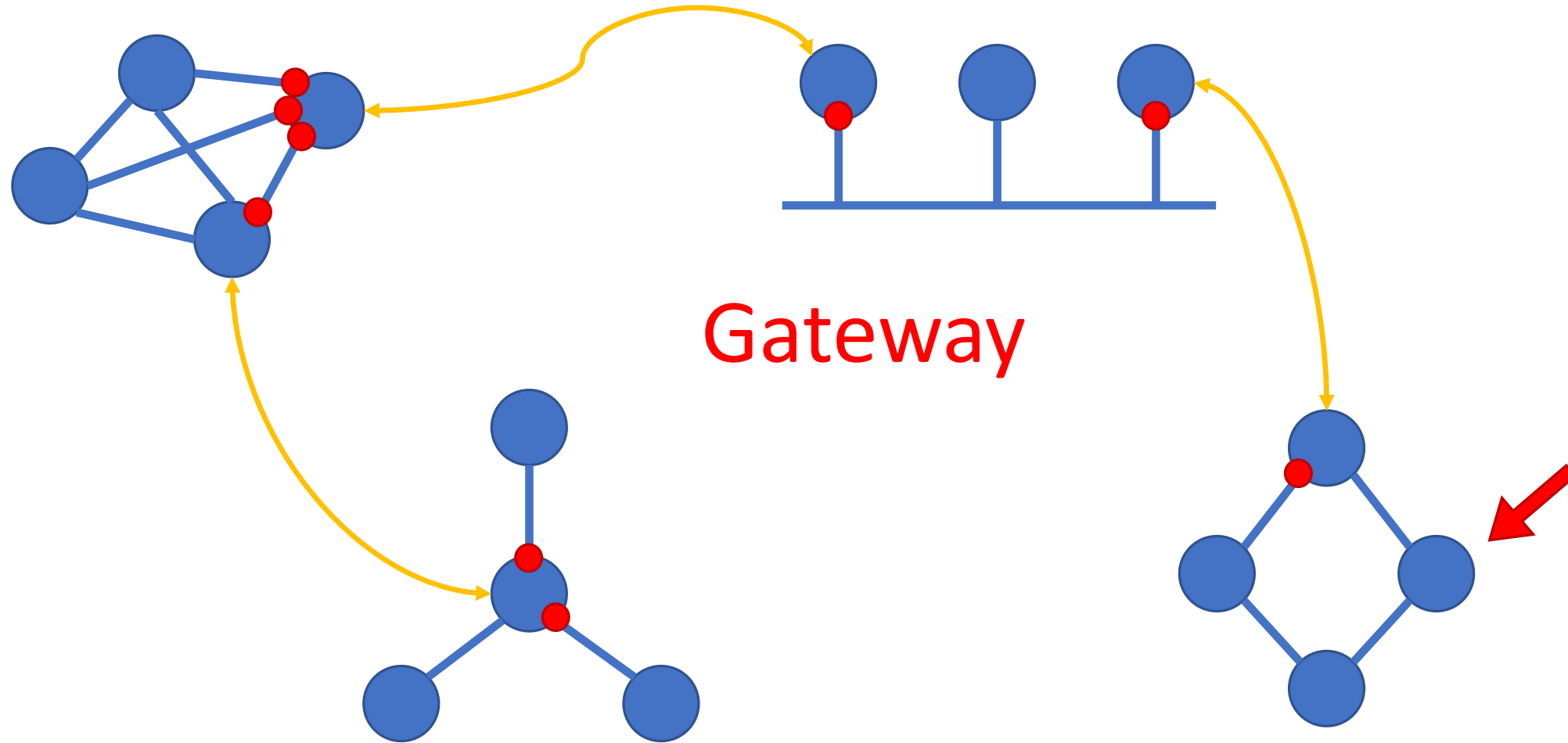


# Basic Concepts

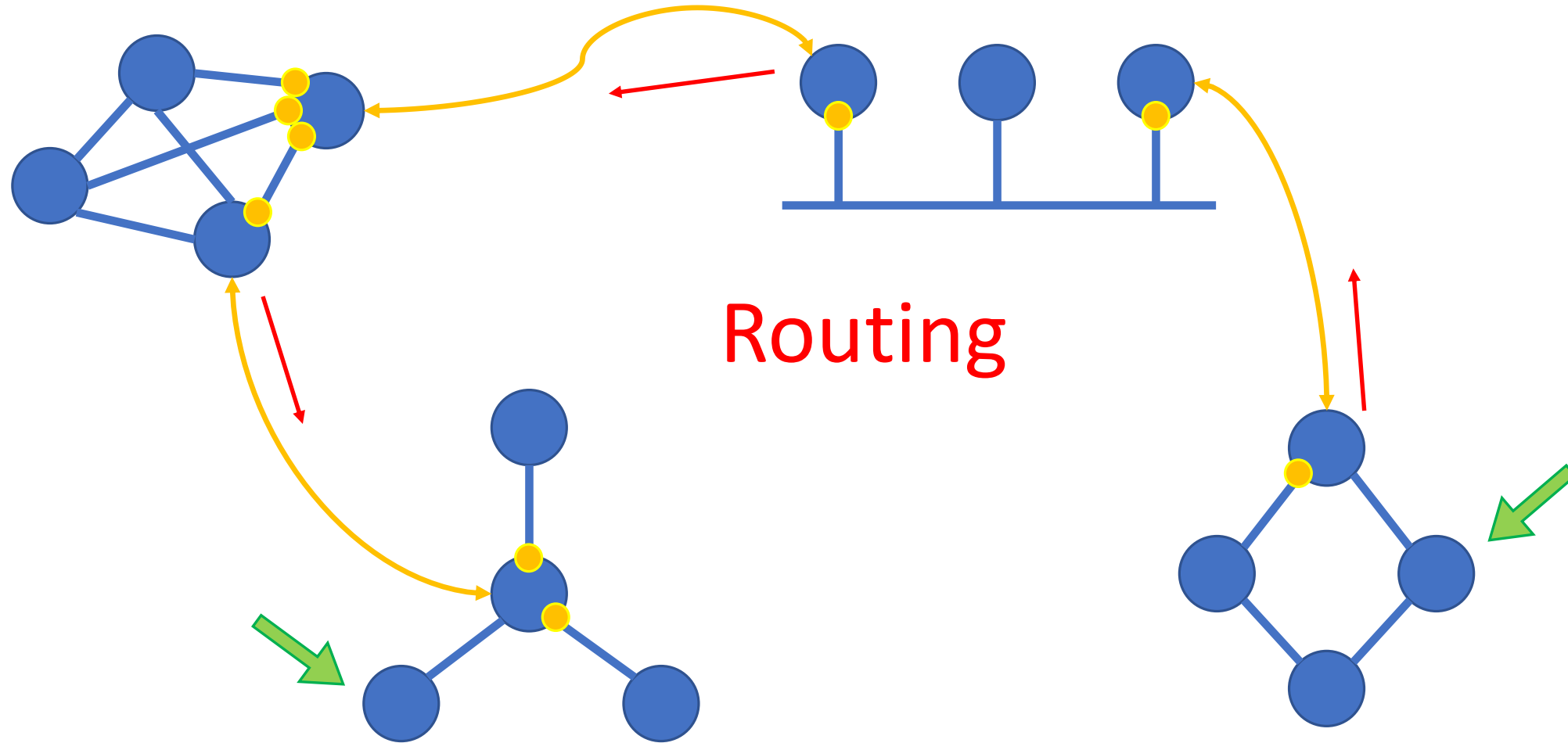




# Basic Concepts



# Basic Concepts



# OSI Model

---

- Operating System Interconnection model (OSI model)
- Developed by the International Organization for Standardization (ISO)
- ISO/IEC 7498-1
- Seven layers
- Protocols operate on certain layers
  - Some protocols operate across layers

# Tanenbaum's TCP/IP 5-Layer Model

- Similar to OSI model in nature
- Eliminates some protocols from lower layers

OSI Model	Tanenbaum Model
Application	Application
Presentation	
Session	
Transport	Transport
Network	Internet
Data Link	Data Link
Physical	Physical

# OSI Model – Application Layer

---

- Layer 7 of the OSI model
- Protocols on L7 allow applications to communicate with each other
- Need predefined set of rules
- Some L7 protocols make assumptions on protocols on lower layers
  - Possible to implement L7 protocols ignoring these assumptions as long as bridges in lower layers are provided

# OSI Model – Application Layer

Protocol	Name	Description
HTTP	Hypertext Transfer Protocol	Allows bidirectional transfer of data between a client and a server. Most commonly used by web browsers.
FTP	File Transfer Protocol	Allows bidirectional data transfer between clients and servers. Optimized to transfer individual files.
IRC	Internet Relay Chat	Allows bidirectional text data streams to be sent across clients and servers. Used in some chat services.
SSH	Secure Shell	Allows for secure remote access.
SMTP	Simple Mail Transfer Protocol	Used to send e-mails to a server.
POP	Post Office Protocol	Used to receive e-mails from a server.
IMAP	Internet Message Access Protocol	Used by e-mail clients to retrieve messages from a server.
XMPP	Extensible Messaging and Presence Protocol	Used in some chat applications to allow client-to-client communications over a distributed network.

# OSI Model – Presentation Layer

---

- Layer 6 of the OSI model
- Protocols in L6 establish context between L7 and L5
- L6 is the last layer at which the programmer considers data structures and presentation

# OSI Model – Presentation Layer

Protocol	Name	Description
Telnet	Teletype Network	Provides a bidirectional, text oriented communication facility using virtual terminals
NCP	NetWare Core Protocol	Allows for clock synchronization, file, directory access, messaging and remote command execution
LPP	Lightweight Presentation Protocol	Used to provide ISO presentation services to L5 of the OSI model
AFP	Apple Filing Protocol	Part of the Apple File Service, allowing remote access of directories and files.



# OSI Model – Session Layer

---

- Layer 5 of the OSI model
- Protocols in L5 provide authentication, authorization, checkpointing, and recovery for network devices
- Programmers may request services from this protocol

# OSI Model – Session Layer

Protocol	Name	Description
PPTP	Point-to-Point Tunneling Protocol	Method for implementing virtual private networks, now obsolete. Has many known security issues.
RPC	Remote Procedure Call	Used for distributed computing to request execution of remote subroutines and obtain the result of the computation as if executed in the local machine.
NetBIOS	Network Basic Input Output System	Non-routable protocol that allows applications to communicate over a network, providing name services, datagram distribution services (stateless), and session services (stateful).
L2TP	Layer 2 Tunneling Protocol	Used to support virtual private networks by providing a tunneling mechanism.
H.245	Call Control Protocol for Multimedia Applications	Used for the line transmission of non-telephone signals.
SOCKS	Secure Sockets	Allows for the exchange of network packets between a client and a server over a proxy server.
SSL	Secure Sockets Layer	Provides communication security over a computer network. Superseded by TLS.
TLS	Transport Layer Security	Modern version of SSL. Provides communication security over a computer network.

# OSI Model – Transport Layer

---

- Layer 4 of the OSI model
- Protocols in L4 provide the means of transferring datagrams from a source to a destination
  - Possible to transfer datagrams over various networks
  - Can be connection oriented (stateful) or connectionless (stateless)
- Maintain quality of services
- Protocols may include services that add
  - Reliability
  - Flow control
  - Congestion avoidance
  - Multiplexing

# OSI Model – Transport Layer

Protocol	Name	Description
TCP	Transmission Control Protocol	Stream-oriented protocol that provides reliable, ordered, and error-checked delivery over an IP network.
UDP	User Datagram Protocol	Datagram-oriented (message oriented) protocol that provides error-checking but does not guarantee delivery, ordering, or datagram duplication.
RDP	Reliable Data Protocol	Connection-oriented protocol that provides facilities for remote loading, debugging, and bulk transfer of images and data, but does not guarantee ordering.
ATP	AppleTalk Transmission Protocol	Used in old Apple machines as the backbone for AppleTalk services.
SST	Structured Stream Transport Protocol	Experimental protocol that provides reliable, ordered, and error-checked delivery over IP networks with enhanced stream management.

# OSI Model – Network Layer

---

- Layer 3 of the OSI model
- Protocols in L3 provide facilities to transfer data sequences (datagrams) from one node to another node in a different network
  - Examine the destination of the datagram and send it through the proper communications channel
  - This is called *packet routing*
  - Networks that are traversed as part of routing are called *hops*
- Devices that operate at this layer are called *routers*

# OSI Model – Network Layer

Protocol	Name	Description
IP	Internet Protocol	Allows routing between networks in the Internet protocol suite.
IPX	Internetwork Packet Exchange	Allows routing between networks in the IPX/SPX protocol suite.
IPsec	Internet Protocol Security	Authenticates and encrypts packets of data sent over an IPv4 network.
ICMP	Internet Control Message Protocol	Used to send error messages and operational information.
RIP	Routing Information Protocol	Distance-vector routing protocol used to propagate routing tables across routers.
OSPF	Open Shortest Path First Protocol	Protocol used to propagate routing tables using a link state routing algorithm.
EIGRP	Enhanced Interior Gateway Routing Protocol	Allows for automatic routing decision and configuration of routing tables.
BGP	Border Gateway Protocol	Allows for propagation of routing and reachability information.

# OSI Model – Data Link Layer

---

- Layer 2 of the OSI model
- Protocols in L2 provide facilities for internode transfers, linking two *directly connected* nodes
- Can detect and may correct errors that occur at L1
- Manipulating packets at this layer is called *packet switching*
  - Devices that work at L2 are called *switches*

# OSI Model – Data Link Layer

Protocol	Name	Description
STP	Spanning Tree Protocol	Builds a loop free topology for Ethernet networks allowing for backup links and eliminating broadcast radiation caused by loops.
IEEE 802.3	Ethernet	Portions of Ethernet are in L2, such as the parts that deal with media access controls.
IEEE 802.11	Wi-Fi	Portions of Wi-Fi are in L2, such as the parts that deal with media access controls.



# OSI Model – Physical Layer

---

- Layer 1 of the OSI model
- Protocols in L1 define the signaling necessary to transmit information between nodes and networks
  - Electrical characteristics
  - Signal frequencies
  - Transport mediums
  - Layout of pins
- Concern is *how* data is transmitted, not *what* data is transmitted
- Devices in this layer include
  - Cables
  - Hubs
  - Repeaters

# OSI Model – Physical Layer

Protocol	Name	Description
IEEE 802.3	Ethernet	Portions of IEEE 802.3 reside at this layer, such as those dealing with cabling and modulation.
IEEE 802.11	Wi-Fi	Portions of IEEE 802.11 reside at this layer, such as those dealing with frequencies used and modulation schemes.
IEEE 802.15.1	Bluetooth	Portions of this protocol suite reside at this layer, such as those dealing with frequencies used and modulation schemes.
IEEE 802.15.4	ZigBee, et al	Portions of this protocol suite reside at this layer, such as those dealing with frequencies used and modulation schemes.

# Tanenbaum's TCP/IP 5-Layer Model

- Similar to OSI model in nature
- Eliminates some protocols from lower layers

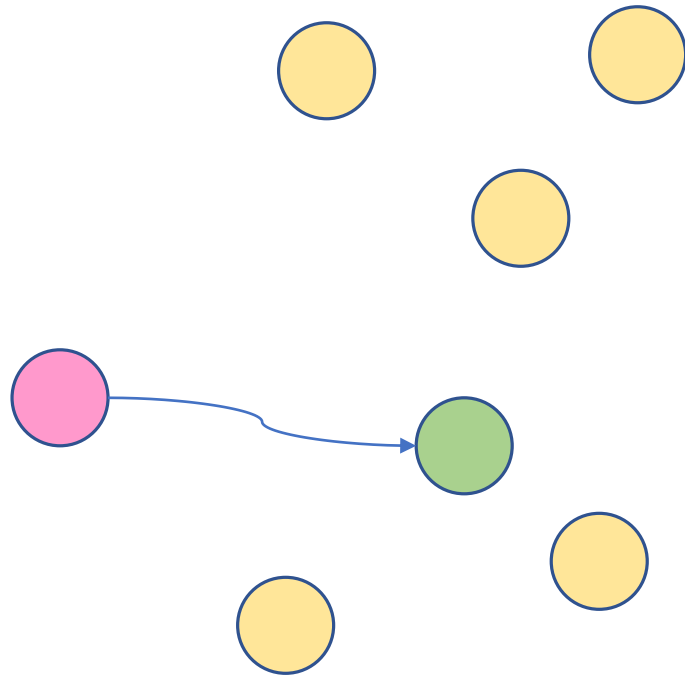
OSI Model	Tanenbaum Model
Application	Application
Presentation	
Session	
Transport	Transport
Network	Internet
Data Link	Data Link
Physical	Physical



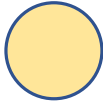
# Internet Protocol

---

- Most commonly used Network Layer protocol
- Connectionless protocol
- Operates in a best effort delivery model
  - Delivery is NOT guaranteed
  - Ordered delivery is NOT guaranteed
  - Does NOT avoid duplicate packets
- Higher layer protocols need to make up for this if required

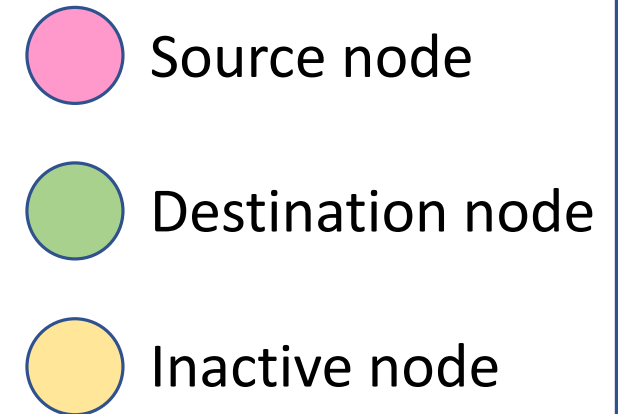
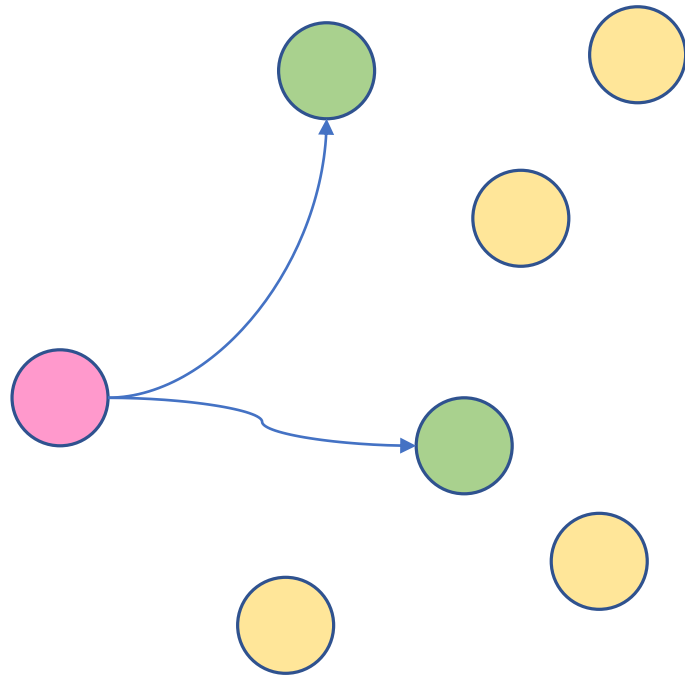
# Types of IP Traffic



-  Source node
-  Destination node
-  Inactive node

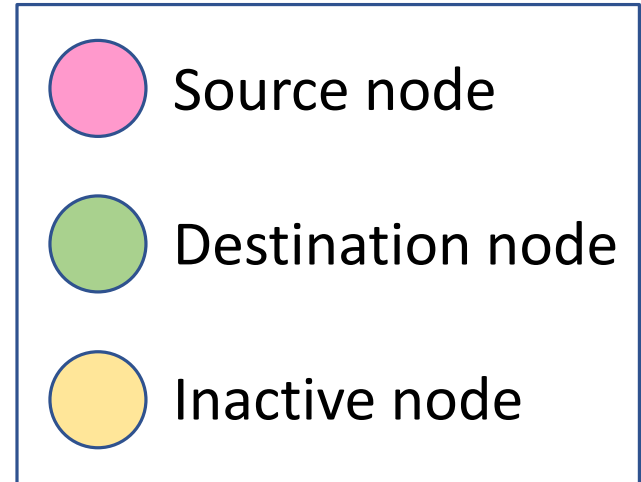
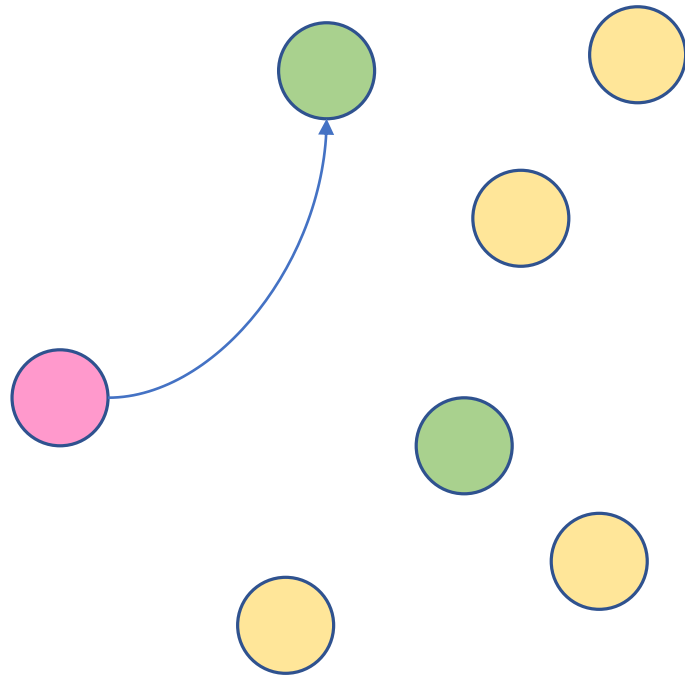
**Unicast** transmissions refer to one-to-one transmissions from one point in a network to another point in [possibly another] network.

# Types of IP Traffic



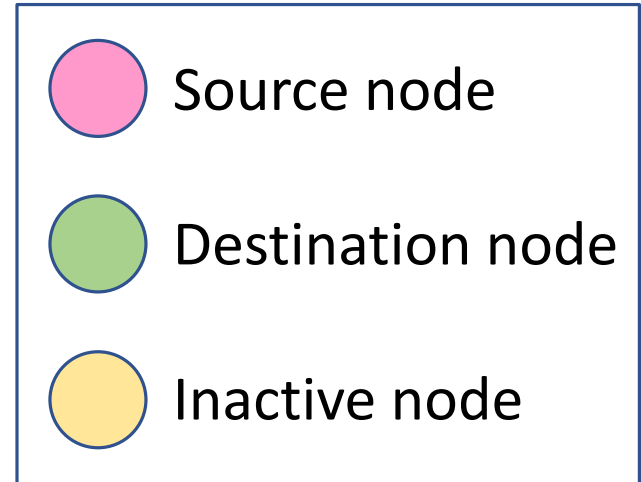
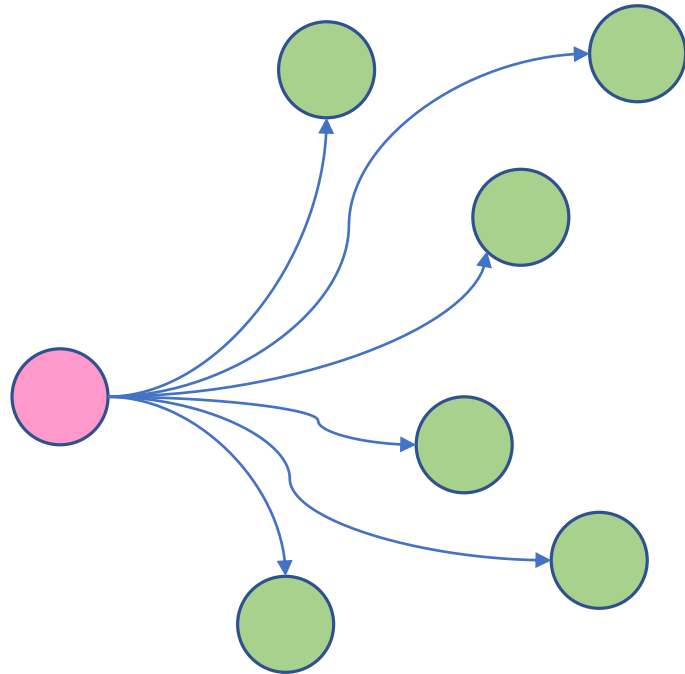
**Multicast** transmissions refer to one-to-many or many-to-many model. Packets are routed simultaneously in a single transmission to many recipients.

# Types of IP Traffic



**Anycast** transmissions use a one-to-one-of-many approach. Packets are routed to any single member of a set of receivers that are defined by the same destination address.

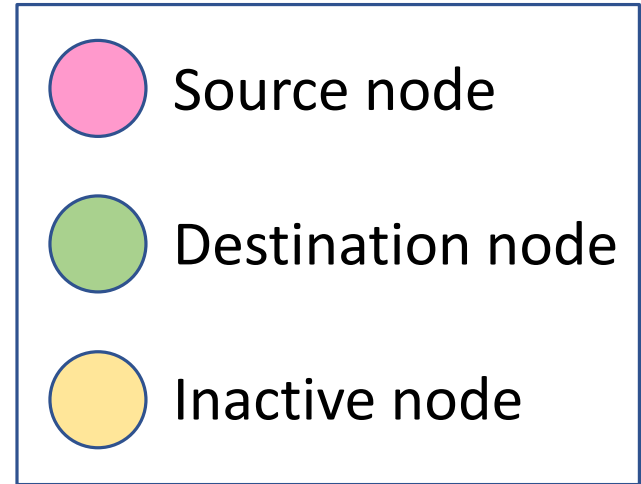
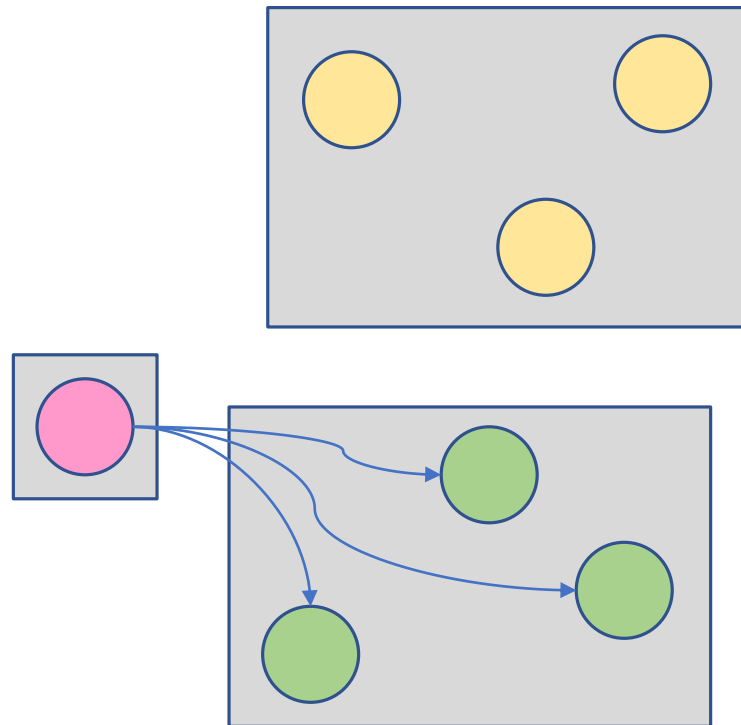
# Types of IP Traffic



**Broadcast** transmissions refer to a one-to-all transmission. The network automatically replicates datagrams (packets) as needed in order to reach all recipients.



# Types of IP Traffic



**Geocast** deliver packets to a group of destinations defined by their geographical location.

# Internet Protocol version 4

---

- Most widely deployed version of the Internet Protocol
- IPv4 addresses consist of four octets
  - Four groups of 8 bits
  - Separated by periods (periods are not part of the IP address itself)
- IPv4 address range:
  - 0.0.0.0 – 255.255.255.255
  - 4,394,967,296 addresses in total
  - Not all addresses are actually available
- IP address divided into two parts:
  - Network identifier (prefixes the IP address)
  - Host identifier (identity of the host within the network)

# Address Distribution Schemes on IPv4

---

- Classful network
  - Divides the address space for IPv4 into
- Classless Inter-Domain Routing (CIDR, say *sider*)
  - Subverts exhaustion problem in classful addressing
  - Slows growth of routing tables

# Classful Addressing on IPv4

Class	Leading Bits	Size of NNB	Size of Rest Bitfield	# of Networks	Addresses per Network	Total Addresses
A	0	8	24	128	$2^{24}$	$2^{31}$
B	10	16	16	$2^{14}$	$2^{16}$	$2^{30}$
C	110	24	8	$2^{21}$	$2^8$	$2^{20}$
D	1110	-	-	-	-	$2^{28}$
E	1111	-	-	-	-	$2^{28}$

Class	Start Address	End Address
A	0.0.0.0	127.255.255.255
B	128.0.0.0	191.255.255.255
C	192.0.0.0	223.255.255.255
D	224.0.0.0	239.255.255.255
E	240.0.0.0	255.255.255.255

This scheme is obviously deficient when it comes to IP address distributions.

# Classless Inter-Domain Routing on IPv4

---

- Overcomes exhaustion problem in classful addressing by introducing *variable-length subnet masking* (VLSM)
  - Allows network to be divided into variously sized subnets
  - Advantage of sizing the network into more appropriate needs
- Addresses in CIDR consist of
  - An IP address
  - A routing prefix

# Classless Inter-Domain Routing on IPv4

---

## Notation

192.168.001.100/24

IP address, which can be a node address, network address, or broadcast address

# Classless Inter-Domain Routing on IPv4

---

## Notation

192.168.001.100/24

The / (slash) character

# Classless Inter-Domain Routing on IPv4

---

## Notation

192.168.001.100 / 24

A decimal number which is the count of leading 1 bits in the routing mask. Often called the network mask, netmask, or subnet mask.



# Classless Inter-Domain Routing on IPv4

---

- Determining if two addresses reside in the same network
  - Routing mask must be the same on both addresses
  - The bitwise AND of the address and the routing mask yield the same result

# Classless Inter-Domain Routing on IPv4

- Determining if two addresses reside in the same network
  - Routing mask must be the same on both addresses
  - The bitwise AND of the address and the routing mask yield the same result

**192.168.1.100/24**

192.168.0001.100

& 255.255.255.000

---

192.168.0001.100

**192.168.1.200/24**

192.168.0001.200

& 255.255.255.000

---

192.168.0001.000

Network address  
match. Addresses are  
on the same network.

# Classless Inter-Domain Routing on IPv4

- Determining if two addresses reside in the same network
  - Routing mask must be the same on both addresses
  - The bitwise AND of the address and the routing mask yield the same result

**192.168.2.100/24**

192.168.0000.0000

& 255.255.255.0000

---

192.168.0000.0000

Network address do not match. Addresses are not on the same network.

**192.168.1.200/24**

192.168.0001.2000

& 255.255.255.0000

---

192.168.0001.0000

# Classless Inter-Domain Routing on IPv4

- Determining if two addresses reside in the same network
  - Routing mask must be the same on both addresses
  - The bitwise AND of the address and the routing mask yield the same result

**192.168.1.100/25**

192.168.0001.100  
& 255.255.0000.0000  
-----  
192.168.0001.0000

Routing masks do not match. Addresses are not on the same network.

**192.168.1.200/24**

192.168.0001.200  
& 255.255.255.0000  
-----  
192.168.0001.0000

# Classless Inter-Domain Routing on IPv4

---

- Not all addresses are available in IPv4, among them
  - *Network Address*, which is defined to be the lowest address on a network. This is also called the *routing prefix*.
  - *Broadcast Address*, which is defined to be the highest address on a network
  - These addresses are reserved: reserved addresses can not be assigned to any node in the network
  
- Certain IP blocks are also not available
  - IPs in these blocks can not appear in public networks
  - Some of these blocks have special purposes

# Classless Inter-Domain Routing on IPv4

CIDR	# of addresses	Usage	Purpose
0.0.0.0/8	16,777,216	Software	For broadcast messages to the current host (source only)
10.0.0.0/8	16,777,216	Private network	Local communications within a private network
100.64.0.0/10	4,194,304	Private network	For communications between a service provider and subscribers when using Network Address Translation
127.0.0.0/8	16,777,216	Host	Loopback address to the local host (no place like 127.0.0.1)
169.254.0.0/16	65,536	Subnet	Link-local addresses between two hosts configured using Automatic Private IP Addressing (APIPA)
172.16.0.0/12	1,048,576	Private network	Local communications within a private network
192.0.0.0/24	256	Private network	For the IANA IPv4 Special Address Registry
192.0.2.0/24	256	Documentation	TEST-NET-1; for use in documentation and examples
192.88.99.0/24	256	Internet	Before deprecation, used by 6to4 anycast

# Classless Inter-Domain Routing on IPv4

CIDR	# of addresses	Usage	Purpose
192.168.0.0/16	65,536	Private network	Local communications within a private network
198.18.0.0/15	131,072	Private network	Testing of inter-network communications between two separate subnets
198.51.100.0/24	256	Documentation	TEST-NET-2; for use in documentation and examples
203.0.113.0/24	256	Documentation	TEST-NET-3; for use in documentation and examples
224.0.0.0/4	268,435,456	Internet	Multicast
240.0.0.0/4	268,435,456	Internet	Experimental, reserved for future use
255.255.255.255 /32	1	Subnet	Reserved for the <i>limited broadcast</i> destination address

# IPv4 Packet Header

---

- Contains 14 fields:
  - 13 fields are required
  - 1 field is an enumeration of optional options
- Fields are stored in Big Endian format



# IPv4 Packet Header

	0								1								2								3							
	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
<b>0</b>	Version				IHL				DSCP				ECN				Total Length															
<b>4</b>	Identification																Flags				Fragment Offset											
<b>8</b>	TTL								Protocol								Checksum															
<b>12</b>	Source																															
<b>16</b>	Destination																															
<b>20</b>	Options																															
<b>24</b>																																
<b>28</b>																																
<b>32</b>																																

# IPv4 Packet Header Fields

Field	Name	Description
Version	Version	IP packet version. For IPv4, this is always 4.
IHL	Internet Header Length	Number of 32 bit words in the header. Minimum value is 5. If options are used, value will be greater than 5.
DSCP	Differentiated Service Code Point	Used to provide quality of service for packet, e.g. give low latency service to critical network traffic while providing best-effort service to non-critical traffic.
ECN	Explicit Congestion Notification	Allows end-to-end notification of network congestion without dropping packets. Used only when both endpoints and network support it. Endpoints must agree to use it.
Total Length	Total Length	Total packet length in bytes, including header and data. Minimum value is 20, which is a header, no options and data.
Identification	Identification	Used for uniquely identifying the group of fragments in a single IP datagram.

# IPv4 Packet Header Fields

Field	Name	Description								
Flags	Flags	<p>Bitfield used to control or identify the fragment.</p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Reserved, must be zero.</td> </tr> <tr> <td>1</td> <td>Do not fragment (DF) packet if set. If fragmentation is required, drop packet.</td> </tr> <tr> <td>2</td> <td>More fragments (MF). For fragmented packets, MF is set except for the last one.</td> </tr> </tbody> </table>	Bit	Description	0	Reserved, must be zero.	1	Do not fragment (DF) packet if set. If fragmentation is required, drop packet.	2	More fragments (MF). For fragmented packets, MF is set except for the last one.
Bit	Description									
0	Reserved, must be zero.									
1	Do not fragment (DF) packet if set. If fragmentation is required, drop packet.									
2	More fragments (MF). For fragmented packets, MF is set except for the last one.									
Fragment offset	Fragment offset	Offset of the fragment relative to the start of the original unfragmented packet. Measured in 8 byte blocks.								
TTL	Time To Live	Amount of time the packet has to live. Decrement by one on each routing hop. If it reaches zero, packet is dropped and router sends an ICMP Time Exceeded Message to the sender.								

# IPv4 Packet Header Fields

Field	Name	Description
Protocol	Protocol	Used to define the protocol contained in the data portion of the packet. The Internet Assigned Numbers Authority (IANA) maintains the list of protocol numbers. For example, TCP is 6, ICMP is 1, and UDP is 17.
Checksum	Header Checksum	Header checksum computed as the 16 bit one's complement of the one's complement sum of all 16 bit words in the header. When computing the checksum, the value of the checksum field is 0. Each time a router decreases the TTL of a packet, a new checksum is computed.
Source	Source	Source address of IP packet. If using a NAT/PAT setup, source address may be changed.
Destination	Destination	Destination address of IP packet. If using a NAT/PAT setup, destination address may be changed.

# IPv4 Packet Header

---

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01
  
```

# IPv4 Packet Header

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01
  
```

Packet Version (4)

# IPv4 Packet Header

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01

```

Header Length (5 words)

# IPv4 Packet Header

---

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01
  
```



# IPv4 Packet Header

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01
  
```

— Differentiated Service Code Point (00000 – default)

# IPv4 Packet Header

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01
  
```

Explicit Congestion Notification (00 – Not ECN Capable Transport)

# IPv4 Packet Header

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01

```

Total Length (70 bytes)

# IPv4 Packet Header

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01
  
```

Identification (0x12df)



# IPv4 Packet Header

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01
  
```

Flags (DF = 0, MF = 0)

# IPv4 Packet Header

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01
  
```


 Fragment Offset (packet is not fragmented, so 0)

# IPv4 Packet Header

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01
  
```

← Time to live (128 hops maximum)

# IPv4 Packet Header

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01
  
```

Protocol (17 – UDP)





# IPv4 Packet Header

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01
  
```

Checksum 

# IPv4 Packet – Computing Checksum

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01
  
```

```

sum = 0x4500 + 0x0046 + 0x12df + 0x0000 + 0x8011 + 0x0000 + 0x0ae3
      + 0x38b9 + 0x80e3 + 0x1efe
      = 0x1bbb3
checksum = ~( (sum & 0xffff) + (sum >> 16) )
          = 0x444b
  
```

# IPv4 Packet – Verifying Checksum

```

45 00 00 46 12 df 00 00 80 11 00 00 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01
  
```

```

sum = 0x4500 + 0x0046 + 0x12df + 0x0000 + 0x8011 + 0x444b + 0x0ae3
      + 0x38b9 + 0x80e3 + 0x1efe
      = 0x1fffe
checksum = ~( (sum & 0xffff) + (sum >> 16) )
          = 0x0000
  
```

# IPv4 Packet Header

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01
  
```

Source IP address (10.227.56.185)

# IPv4 Packet Header

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01

```

Destination IP (128.227.30.254)

# IPv4 Packet

---

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01
  
```

- **From** 10.227.56.185
- **To** 128.227.30.254
- **Contains** UDP packet
- Valid IPv4 packet (version 4, checksum valid)
- No flags or QoS

# Internet Protocol version 6

---

- Newer version of Internet Protocol seeing growing deployment
- IPv6 addresses consist of eight groups of two bytes
  - Separated by colons (colons are not part of the IP address itself)
  - Multiple consecutive groups of zeros can be represented by double colon ::
- IPv6 address range:
  - 0:0:0:0:0:0:0:0 – ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
  - $2^{128}$  addresses in total ( $\approx 4.477 \times 10^{28}$  addresses per person)
  - Not all addresses are actually available
- Does not support broadcast traffic

# IPv6 Packet Header

	0								1								2								3							
	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
<b>0</b>	Version				Traffic Class				Flow Label																							
<b>4</b>	Payload Length																Next Header				Hop Limit											
<b>8</b>	Source																															
<b>12</b>																																
<b>16</b>																																
<b>20</b>																																
<b>24</b>	Destination																															
<b>28</b>																																
<b>32</b>																																
<b>36</b>																																



# IPv6 Packet Header Fields

Field	Description
Version	IP packet version. For IPv6, this is always 6.
Traffic Class	This field is a combination of the DSCP and ECN fields from IPv4. The 6 most significant bits are for DSCP and the remaining two bits for ECN.
Flow Label	If set, this field is used to hint routers to allow packets to be routed using the same path when multiple outbound paths are available.
Payload Length	The size of the payload in bytes, including headers any extension headers.
Next Header	Specifies the type of header contained in the data portion of the packet. Equivalent to the Protocol field in IPv4.
Hop Limit	Same as modern use of TTL field in IPv4.
Source address	The IPv6 address of the source node.
Destination address	The IPv6 address of the target node.

# Limitations of IP

---

- Does not guarantee data delivery
    - Packets can be lost (TTL reaches 0, corrupted headers)
  - Does not guarantee data ordering
  - Does not guarantee integrity of payload
- How can this be solved?
- This is the job of the Transport Layer

# Transmission Control Protocol

---

- Provides means to establish reliable, ordered, and error-checked delivery of data over two network nodes
- Runs exclusively over the IP protocol (hence the common TCP/IP name)
- Uses client-server model
  - One node connects to the other
  - Node initiating connection is called *client*
  - Node receiving the connection is called *server*
  - Server must actively listen for connections
- Data is transmitted in *segments* that make up a *stream*

# TCP Segment Header

	0								1								2								3							
	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	Source Port																Destination Port															
4	Sequence Number																															
8	ACK number																															
12	NS	Reserved		Data Offset				FIN	SYN	RST	PSH	ACK	URG	ECE	CWR	Window Size																
16	Checksum																URG Pointer															
20	Options																															
...																																

# TCP Segment Header

Field	Description
Source Port	Identification of the sending internet socket
Destination Port	Identification of the receiving internet socket
Sequence Number	If the SYN flag is set, this field contains the <i>initial</i> sequence number. If the SYN flag is clear, this field contains the <i>accumulated</i> sequence number for the session.
ACK Number	If the ACK flag is set, this field contains the <i>next sequence number that the sender is expecting</i> . This acknowledges the receipt of all previous bytes.
NS	ECN-nonce, adding concealment protection
Reserved	Should be set to 0.
Data offset	Number of words in the header size, minimum size is 5 words and maximum is 15.
FIN	Indicates last segment from sender.
SYN	Synchronize sequence numbers. Only used in opening a connection.
RST	Reset connection
PSH	Push function to push the buffered data to receiving application.

# TCP Segment Header

Field	Description
ACK	Indicates that the ACK Number field is significant. All segments after SYN should have this field set.
URG	URG Pointer field is valid.
ECE	ECN-Echo: If SYN is set, then the peer is ECN capable. If SYN is clear, indicates network congestion (or possibly impending congestion) to the TCP sender. Flag is set this way if IP packet with Congestion Experienced flag (ECN=11) set in the IP header is was received.
CWR	Congestion Window Reduced. Set by sending host to indicate that it received the TCP segment with ECE flag and has responded with the congestion control mechanism.
Window Size	The size of the receive window specifying the number of window size units (usually in bytes).
Checksum	16-bit checksum in the header, payload, and pseudoheader.
URG Pointer	Urgent Pointer: If URG is set, this field contains an offset from the sequence number indicating the last urgent data byte.
Options	Additional options to the segment.

# TCP Protocol Operation

---

- Stateful system
- Divided into three phases
  - **Connection Establishment:** Using three way handshake
  - **Data Transfer:** Using data transfer and congestion control mechanism
  - **Connection Termination:** Using four way handshake. Releases all allocated resources.

# TCP Connection Establishment

---

- Server *must* be actively listening for incoming connections
  - Bind an *internet socket* (port) to an interface and open it for listening
  - Service may bind to multiple ports on any number of interfaces
  - No service or combination of services may bind to duplicate ports on same interface
- Internet sockets are represented by integer numbers
- Certain application layer protocols are always bound to the same internet socket
  - This is by convention
  - In UNIX and family, ports below 1024 can only be opened with SUID permissions



# Common TCP Internet Socket

Port	Protocol	Description
20	FTP	Active mode data transmission in File Transfer Protocol.
21	FTP	Standard listening port in FTP, passive mode data transmission.
22	SSH	Secure Shell protocol.
23	Telnet	Telnet remote login.
25	SMTP	Plaintext Simple Mail Transfer Protocol.
80	HTTP	Plaintext Hypertext Transfer Protocol (unencrypted web servers)
110	POP	Plaintext Post Office Protocol (unencrypted read access to e-mail servers)
143	IMAP	Plaintext Internet Message Access Protocol
194	IRC	Plaintext Internet Relay Chat
443	HTTPS	Secure Hypertext Transfer Protocol (encrypted web servers)

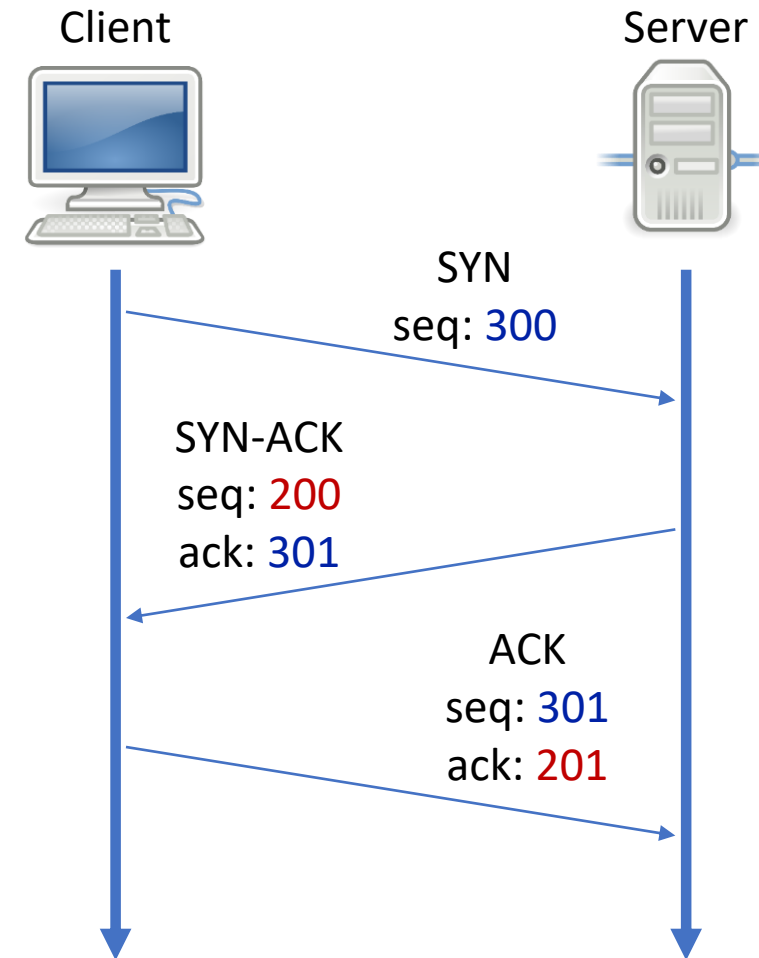
# TCP Connection Establishment

---

- TCP uses a three way handshake
- Server binds to port in an interface (performs *passive open*)
- Client is free to connect to this port
- Client initiates connection by performing an *active open*
- Follows three way handshake

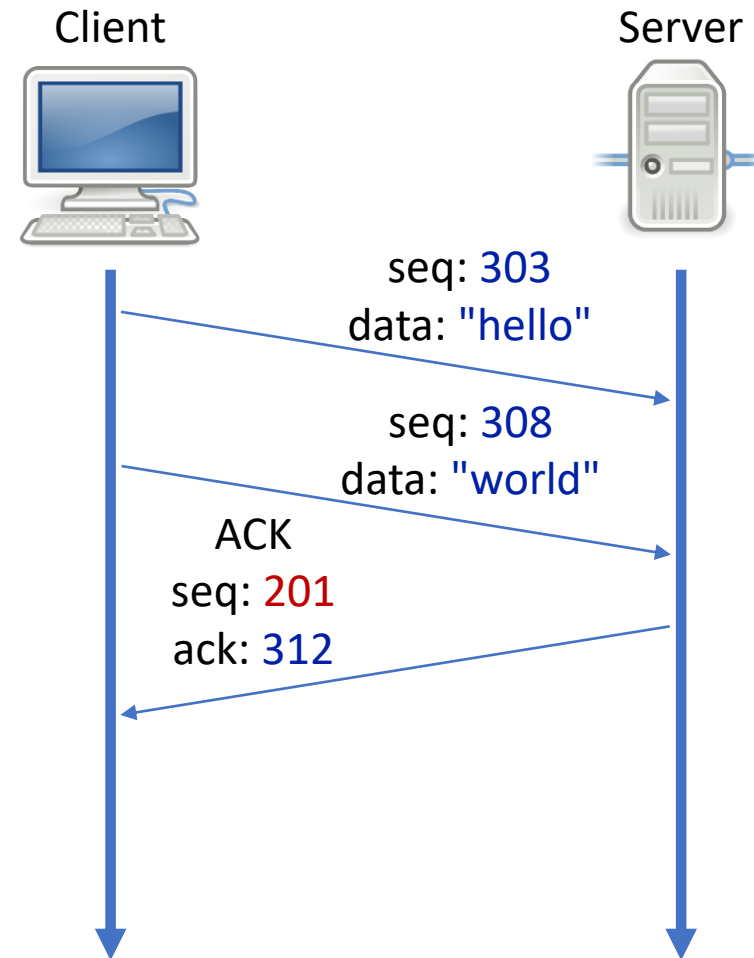
# TCP Three Way Handshake

- Client sends SYN segment to server using random segment sequence number  $n$
- Server responds with SYN-ACK segment, number is set to  $n + 1$ . Server chooses its own sequence number  $m$ .
- Client sends an ACK segment to server, sequence number  $n + 1$ , and acknowledgement number is  $m + 1$ .



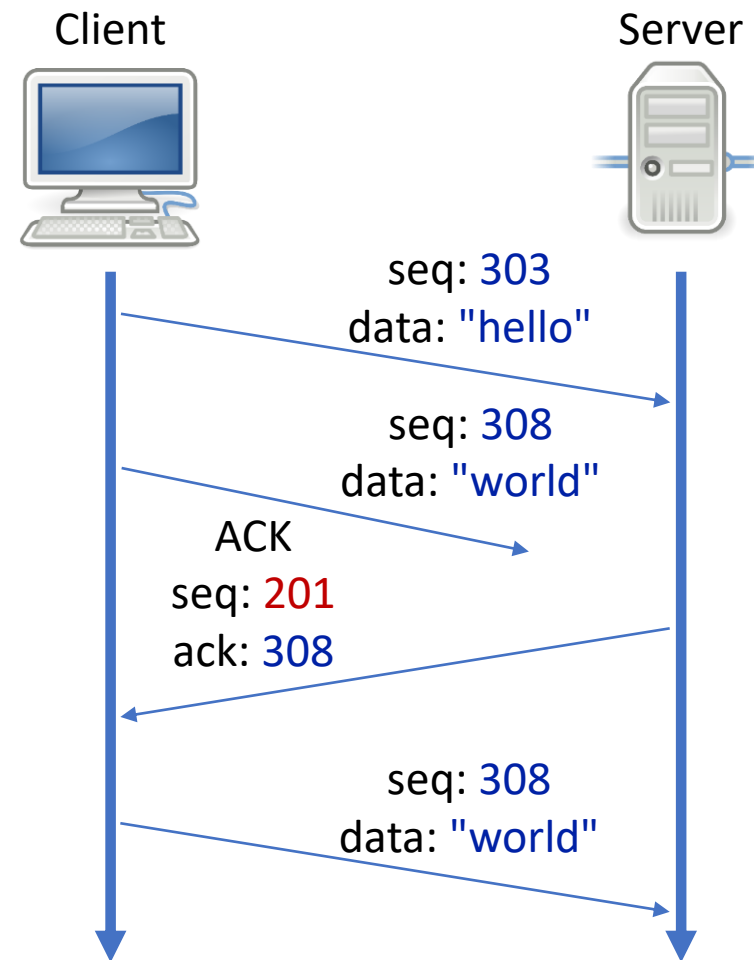
# TCP Data Transfer

- Sending end transmits multiple segments with increasing SYN numbers
- Receiving end replies with ACK segments and increasing SYN numbers
  - ACK number given by SEND\_SYN\_SEQ + highest octect sequence number it has correctly received



# TCP Data Transfer

- Lost segments are detected by not receiving ACK segment from receiver.
  - Retransmit after timeout.
- If ACK is lost, sender transmits copy of segment. Receiver ACKs it and ignores it.
- What about duplicate ACKs received? What does this mean?



# TCP Congestion Control

---

- Four algorithms are currently used
  - Slow Start
  - Congestion Avoidance
  - Fast Retransmit
  - Fast Recovery

# TCP Congestion Control

---

- Slow Start (Sender-based flow control)
- Works using return rate of ACK segments replied by receiver
  - Rate at which acknowledgements are returned by receiver determine the rate which the sender can transmit data
- Algorithm:
  - Initialize congestion window to one segment (maximum segment size initialized by the receiver during the connection establishment)
  - Increase congestion window by one per ACK returned by receiver
- This is an exponential growth algorithm (1 → 2 → 4 → 8...)

# TCP Congestion Control

---

- Once network is saturated (congestion window too large) or network conditions change dropping packets sender goes into *congestion avoidance*
- Sender sends transmission window to one half of the current window size, but at least two segments
- If congestion was indicated by a timeout, the congestion window is reset to one segment: we go back to *slow start*
  - *Slow start* is only used up to the transmission window, increase linearly afterwards
- If congestion was indicated by duplicate ACKs, the *Fast Retransmit* and *Fast Recovery* algorithms are used



# TCP Congestion Control

---

- Two possible causes on duplicate ACK
  - TCP segment was lost
  - TCP segments were received out of order
- Sender does not know which condition
  - In case of out of order: sender should get an ACK for the latest expected segment relatively soon
  - Typically, no more than one or two duplicate ACKs are received in out of order condition
  - More than two is a good indication that a segment was lost
- When three or more duplicate ACKs are received, sender does not wait for timer to expire before retransmitting the segment: *Fast Retransmit*

# TCP Congestion Control

---

- On *Fast Retransmit* sender has implicit knowledge that data is still going to receiver
- This indicates that the lost segment was a rare event
- Instead of going abruptly to slow start, we increment linearly as in Congestion Avoidance mode

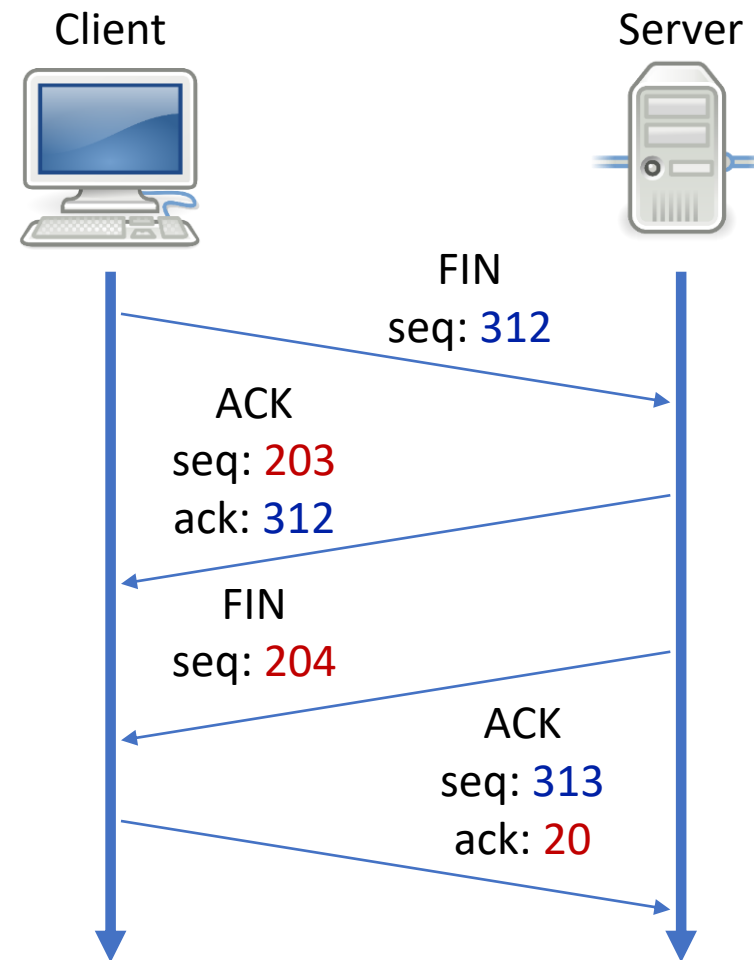
# TCP Transmission Termination

---

- Connection can be terminated by either endpoint
- Four way handshake

# TCP Connection Termination

- Can be terminated by either endpoint
- Uses four way handshake
  - Endpoint wishing to terminate (A) sends FIN segment
  - Other endpoint (B) sends ACK
  - (B) sends own FIN segment
  - (A) replies with ACK segment



# Security Issues with TCP

---

- No guarantees on the integrity of the data being sent
  - Attacker can intercept TCP packet, change payload and checksums
- No guarantees on authenticity of endpoints
  - Attacker can intercept TCP packets and reply

# More TCP Security Issues

---

- **Assumption:** attacker can spoof IP packets
- Attacker repeatedly send SYN segments to server, and reply with the corresponding ACK (initiate three way handshakes)
- Attacker does not keep track of connections established this way (drop connection without four way handshake)
- Server is forced to consume resources keeping track of bogus connections
- **Denial of Service:** *SYN flood*

# More TCP Security Issues

---

- Dealing with SYN floods
  - Unfortunately, not that many good solutions
  - Not trivial to differentiate malicious connections from benign connections
- On SYN flood attacker does not transmit data
  - Lower timeout on server? Can result in some benign connections being dropped due to network latency
- Add more memory and processing power on server to deal with connection?
  - Too expensive
  - Distributed Denial of Services
- Content Distribution Networks (CDNs)
  - Provide large computational backbone to handle many connections

# More TCP Security Issues

---

- **Assumption:** Attacker can eavesdrop on communications channel and is able to redirect packets
- Attacker forges a false TCP segment that looks like the next segment in the stream
- When receiving host sends ACK for extra segment
- Synchronization is lost, ACK numbers start to differ
- **Connection Hijack:** Attacker now carries on with the connection to receiver



# More TCP Security Issues

---

- **Assumption:** Attacker can eavesdrop in the communications channel and is able to predict the size of the next segment to be sent
- Attacker computes next sequence number
- Attacker injects segment computed sequence number and payload size of the next expected sequence
- When next *legitimate* segment is received, it is silently dropped as duplicate
- **TCP veto:** Receiver is tricked into accepting a malicious payload

# More on TCP Security Issues

---

An entity that can easily perform this type of attacks:

## Your Internet Service Provider.

All your internet traffic goes through them. They can shape your connection as they please.

*That's a nice TCP link to that YouTube Video you've got there... It'd be a shame if something happened to it...*

# User Datagram Protocol

---

- Provides means to establish a *connectionless* communication model with little or no overhead over two nodes
- Runs exclusively over the IP protocol (hence the common UDP/IP name)
- No client-server model
  - No handshaking dialog
  - No guarantee delivery
  - No ordering
  - No duplicate detection
- Application is directly exposed to any unreliability on the underlying network

# User Datagram Protocol

---

- Advantages over TCP
  - **Transaction Oriented:** makes it suitable for query-response protocols
  - **Datagram Oriented:** Suitable for modeling other protocols, does not depend on a previous state
  - **Excellent in Unidirectional Setting:** Suitable for broadcast of information
  - **Simplicity:** Suitable for bootstrapping other network services without a full protocol stack
  - **Stateless:** Because no state is kept, it can work with a large number of clients.
  - **No Retransmission Delays:** Suitable for Real-Time applications
- Disadvantages over TCP
  - **Application Layer Error Recovery:** If error recovery is needed, it must be implemented at the application layer.

# User Datagram Protocol

---

- Transmits data in *datagrams*
  - self-contained, independent entity of data carrying sufficient information to be routed from the source to the destination without reliance on earlier exchanges
- Provides application multiplexing, like TCP, using internet sockets
- Provides header and payload verification via checksums

# UDP Datagram Header



Field	Description
Source Port	Identification of the sending internet socket, when meaningful. Assumed to be the port to reply to if needed. If not used, this field should be set to 0. If the source is the client, the port number is likely to be an ephemeral port (a port that is only utilized for the current UDP session).
Destination Port	Identification of the receiving internet socket. This field is required.
Length	The number of bytes in the UDP header and data. The minimum length is 8 bytes (the size of the UDP header).
Checksum	Contains a checksum of the header and the accompanying data. The field is optional in IPv4 and set to zero if unused. The field is mandatory in IPv6.

# Common UDP Internet Socket

Port	Protocol	Description
53	DNS	Domain Name System protocol (resolve hostnames)
67	BOOTP	Bootstrap Protocol server, also used by DHCP (Dynamic Host Configuration Protocol)
68	BOOOTP	Bootstrap Protocol client, also used by DHCP
69	TFTP	Trivial File Transfer Protocol, used in some bootloaders to download firmware images
123	NTP	Network Time Protocol, network latency aware clock synchronization protocol
161	SNMP	Simple Network Management Protocol, used for management of devices in a network

# Recall the IPv4 Packet Header

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01
  
```

Protocol (17 – UDP)



# UDP Datagram Header

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01
  
```


**Source Port (53470)**

# UDP Datagram Header

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01
  
```


**Destination Port (53)**

# UDP Datagram Header

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01
  
```

Datagram Length (header + data = 50 bytes) 

# UDP Datagram Header

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 92 46 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01
  
```

Checksum

# UDP Header – Computing Checksum

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 00 00 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01
  
```

Create pseudo-IP header containing:

- Source Address
- Destination Address
- Padded UDP protocol number
- UDP packet size

# UDP Header – Computing Checksum

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 00 00 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01

```

```

0a e3 38 b9 80 e3 1e fe 00 11 00 32

```

- Perform 16 bit one's complement sum of pseudo-IP header and UDP header + data

# UDP Header – Computing Checksum

```

45 00 00 46 12 df 00 00 80 11 44 4b 0a e3 38 b9
80 e3 1e fe d0 de 00 35 00 32 00 00 81 eb 01 00
00 01 00 00 00 00 00 00 08 6e 6f 72 6d 61 6e 64
79 03 63 64 6e 07 6d 6f 7a 69 6c 6c 61 03 6e 65
74 00 00 01 00 01
  
```

```

sum = 0x0ae3 + 0x38b9 + 0x80e3 + 0x1efe + 0x0011 + 0x0032 + 0xd0de
      + 0x0035 + 0x0032 + 0x0000 + 0x81eb + 0x0100 + 0x0001 + 0x0000
      + 0x0000 + 0x0000 + 0x086e + 0x6f72 + 0x6d61 + 0x6e64 + 0x7903
      + 0x6364 + 0x6e07 + 0x6d6f + 0x7a69 + 0x6c6c + 0x6103 + 0x6e65
      + 0x7400 + 0x0001 + 0x0001
      = 0x76db2
  
```

```
checksum = ~(sum & 0xffff) + (sum >> 16) = 0x9246
```

# Networking

---

PHYSICAL LAYER, TI CC3100

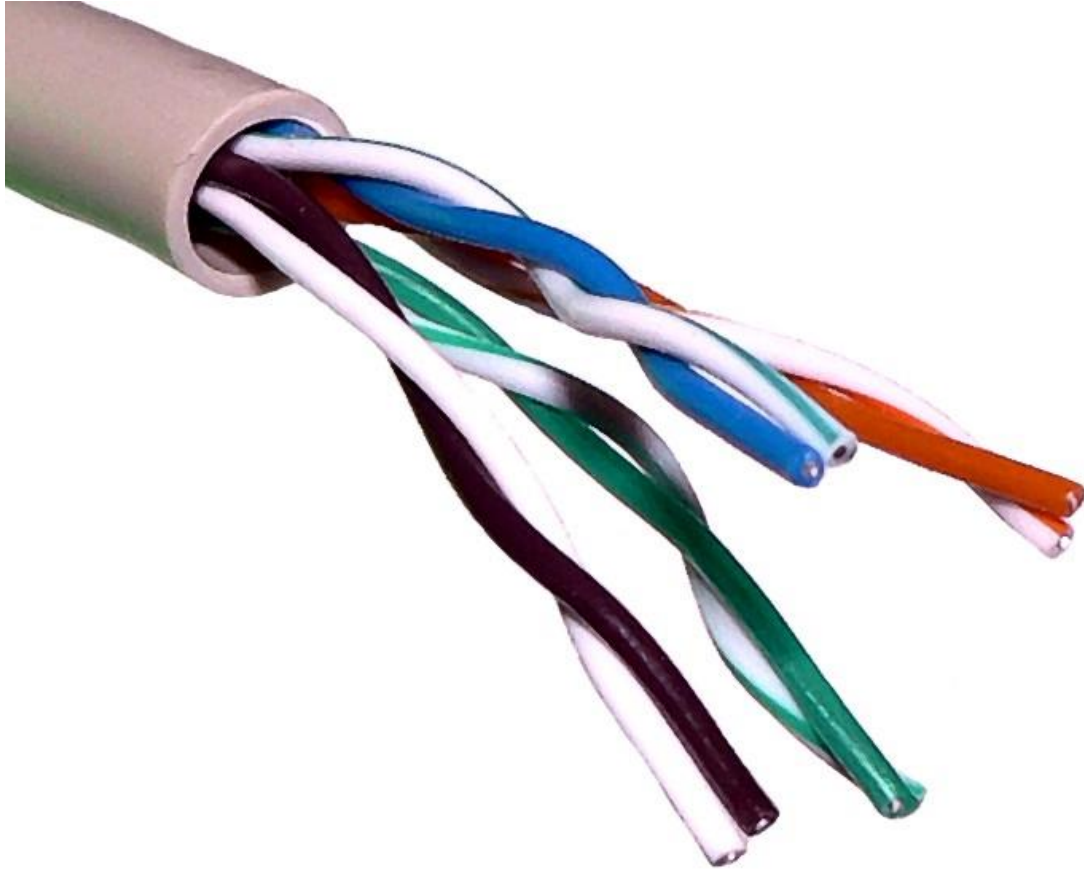


# Physical Layer

---

- Medium used to transmit and receive data
  - Copper Wire
  - Fiber Optic
  - Radio Waves

# Physical Layer – Ethernet



- Copper wire cabling
  - Currently using Category 5e and Category 6 cables
- Twisted-pair cabling
  - Transmit differential signal
  - Reduce noise
- Cable may be shielded
  - Better noise reduction
  - Provide common ground on cable

# Differential Signal Transmission

- Send same signal using opposite polarities over two different wires

$$S_+ = 1 + \sin t$$

$$S_- = -S_+ = -1 - \sin t$$

- Add noise signature to signal

$$N(t) = \text{rnd}(t)$$

- Resulting signal becomes

$$S_{N+} = 1 + \sin t + \text{rnd}(t)$$

$$S_{N-} = -1 - \sin t + \text{rnd}(t)$$

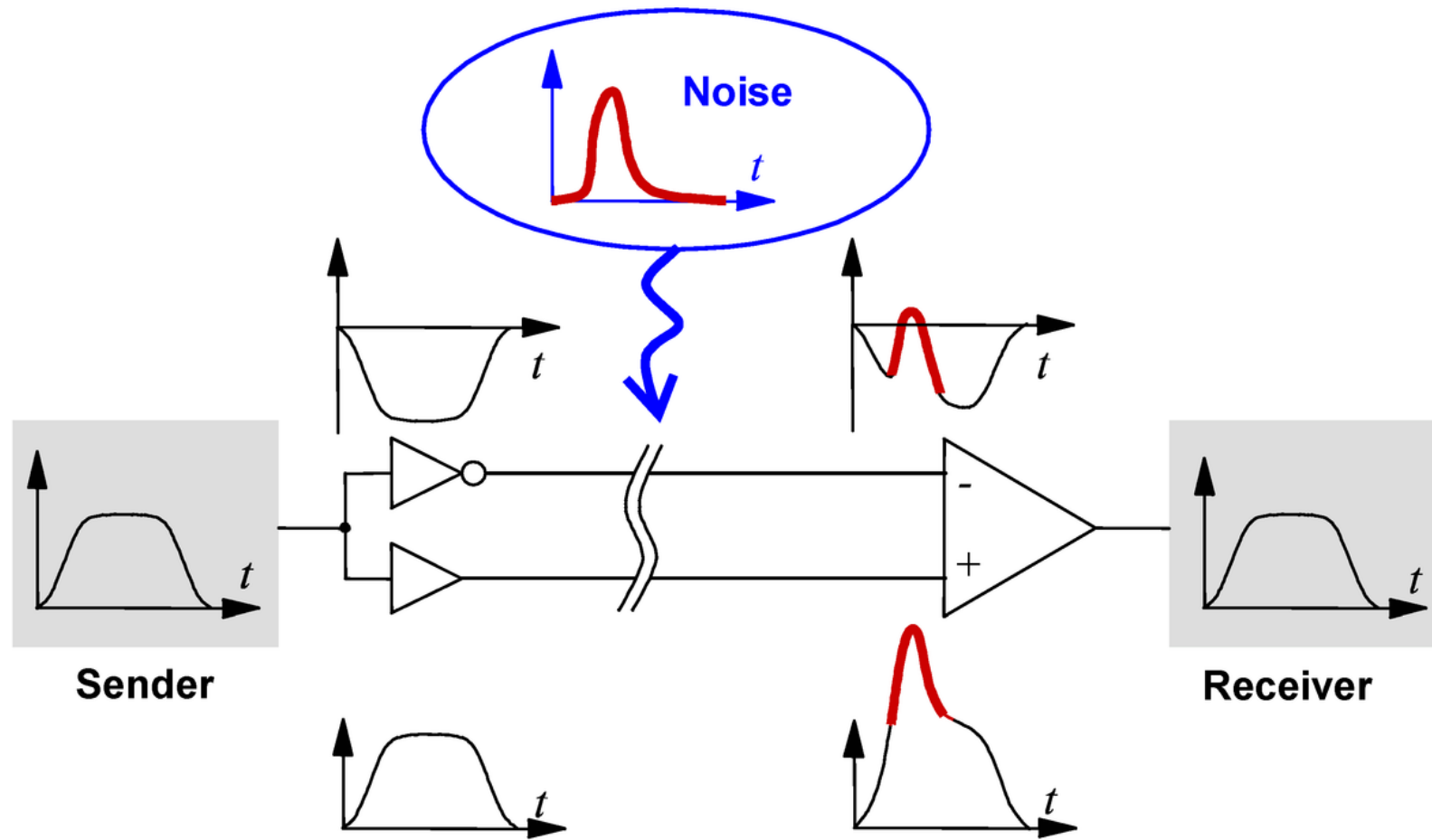
- Receiving end computes  $S_{N+} - S_{N-}$

$$S_{N+} - S_{N-} = 1 + \sin t + \text{rnd}(t) + 1 + \sin t - \text{rnd}(t)$$

$$= 2 + 2\sin t$$

- Noise term is cancelled

# Differential Signal Transmission



# Twisted Pairing

---

- Issue with differential signaling is that the noise must affect both wires with the same magnitude
- This is where twisted pairing comes in
  - Twisted pair places both wires in a net equally close to the noise source
  - End result is that the noise signature cancels out
- Other electromagnetic effects are also at play

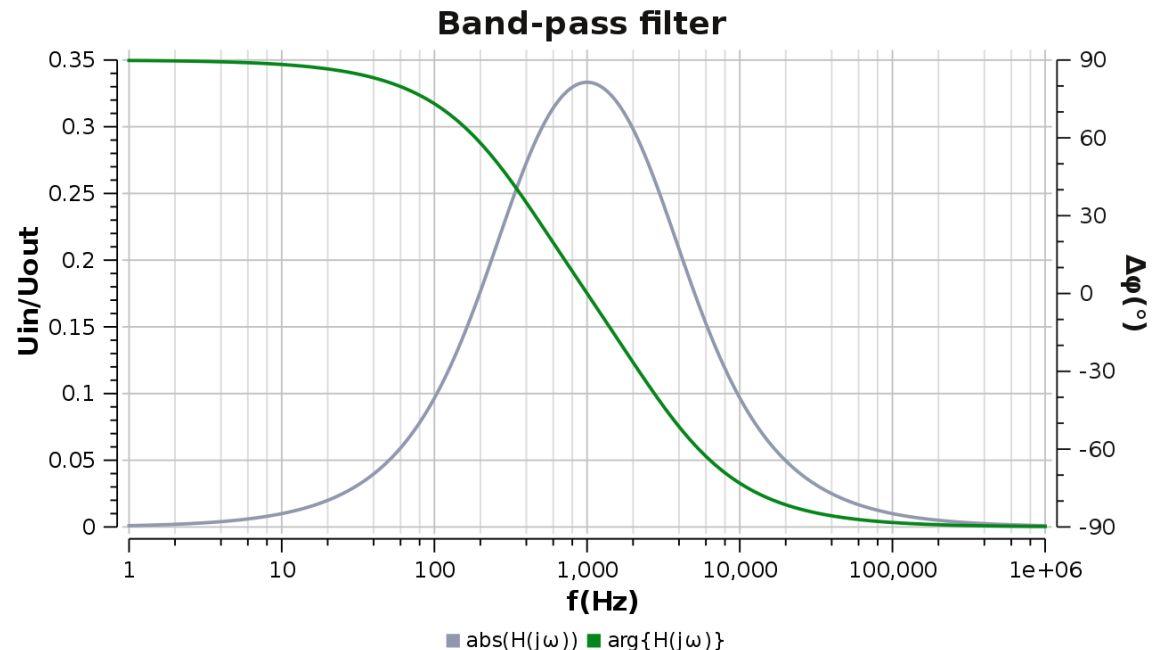
# Physical Layer – Wireless

---

- Use radio waves as carrier
- Different styles of modulation
  - Amplitude Modulation (AM)
  - Frequency Modulation (FM)
  - Phase Modulation (PM)
  - Phase-shift keying (PSK)
  - Quadrature Amplitude Modulation (QAM)
  - Minimum-shift keying (MSK)
- Wireless signals use two waves
  - A carrier wave
  - A data wave
- The operation between the waves depends on the modulation scheme

# Physical Layer – Wireless

- Tuning is done based on the carrier wave
- Receiver uses a band pass filter to retrieve a portion of the EM spectrum
- Demodulate to retrieve data wave



# Amplitude Modulation

---

- Carrier wave described by

$$C(t) = A_c \sin(\omega_c t + \phi_c)$$

- Data wave described by  $a_m(t) \in [-1,1]$

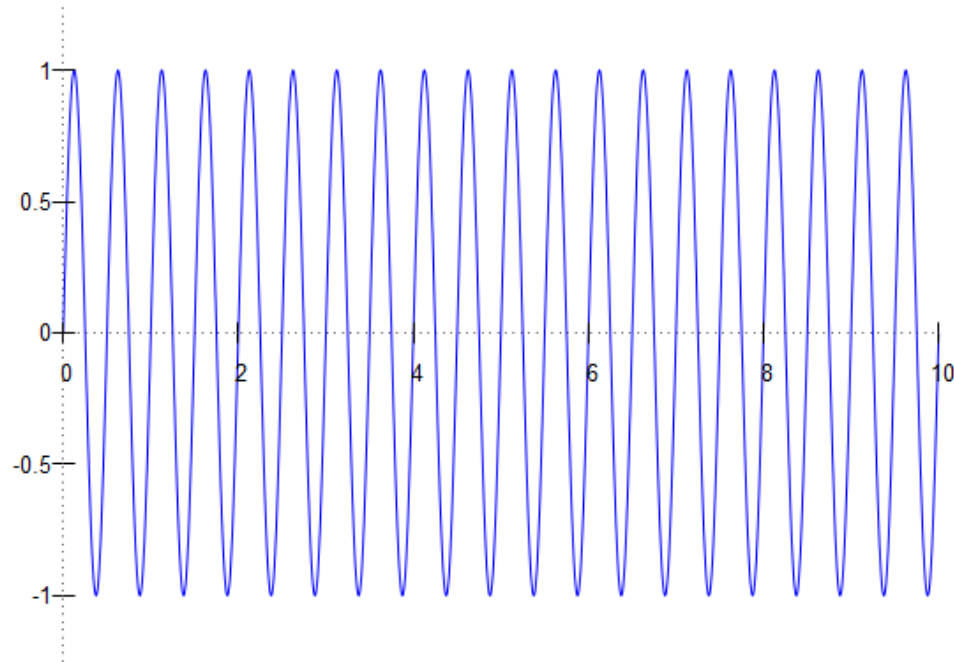
- Transmit

$$x_\alpha(t) = [1 + \alpha a_m(t)] \cdot C(t)$$

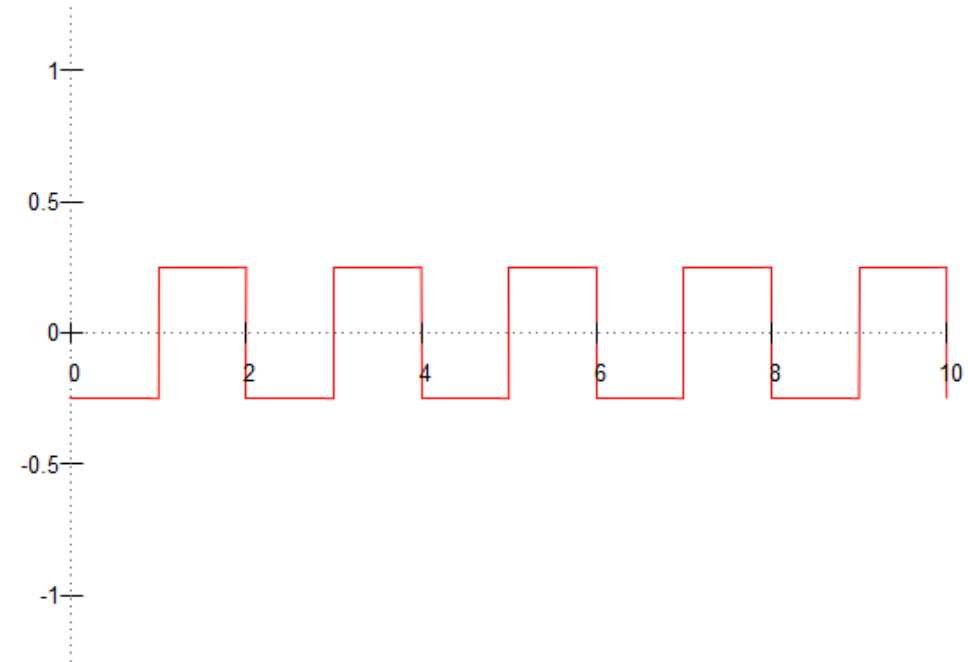
Where  $\alpha \in [0,1]$  is the modulation index



# Amplitude modulation



Carrier Wave



Data Wave

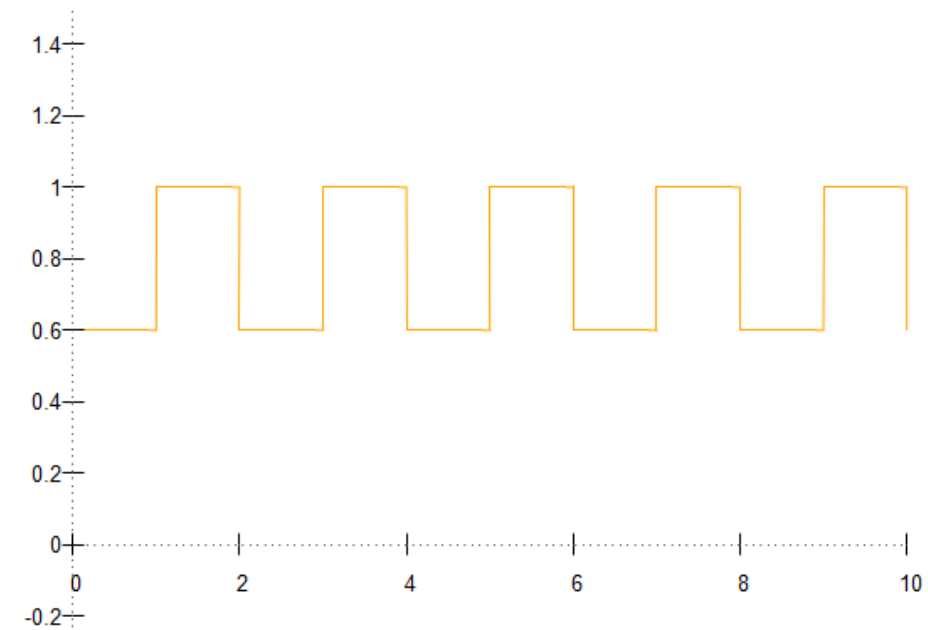
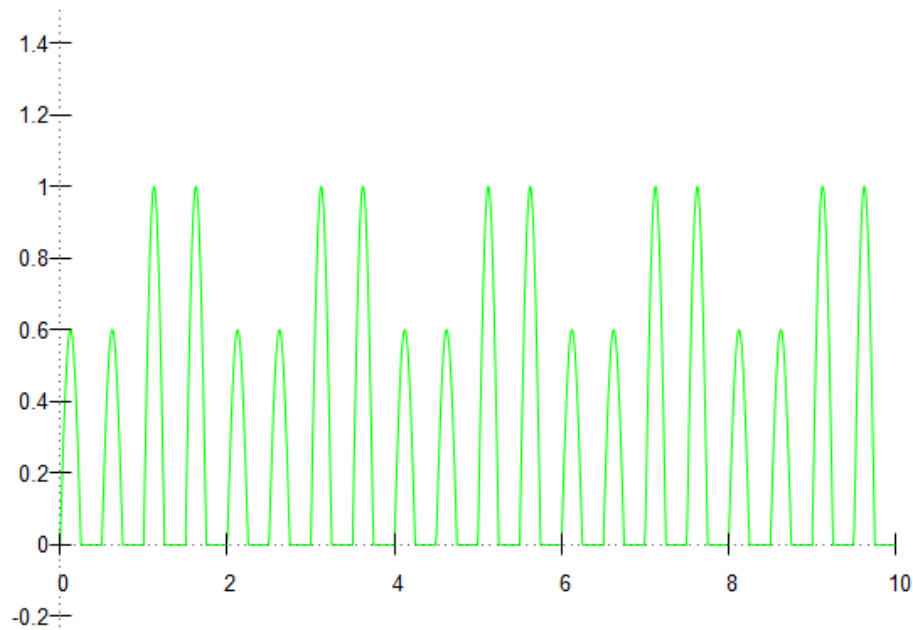
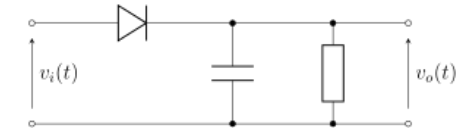
# Amplitude modulation

---



# Demodulating AM

- Rectify waveform
- Filter high frequency component using a low pass filter
- Remove DC component if required using a high pass filter



# Frequency Modulation

---

- Carrier wave described by

$$C(t, \omega_c) = A_c \sin(\omega_c t + \phi_c)$$

- Data wave described by  $a_m(t) \in [-1, 1]$

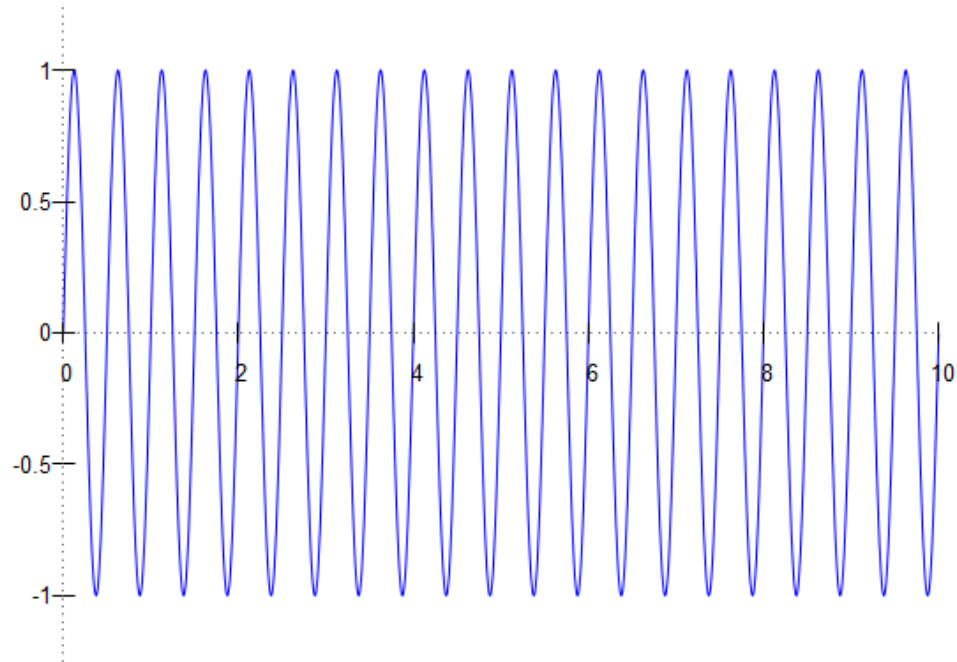
- Compute a new frequency for the carrier wave

$$\omega'_c(t) = \omega_c + \frac{\Delta f}{\max(a_m(t))} a_m(t)$$

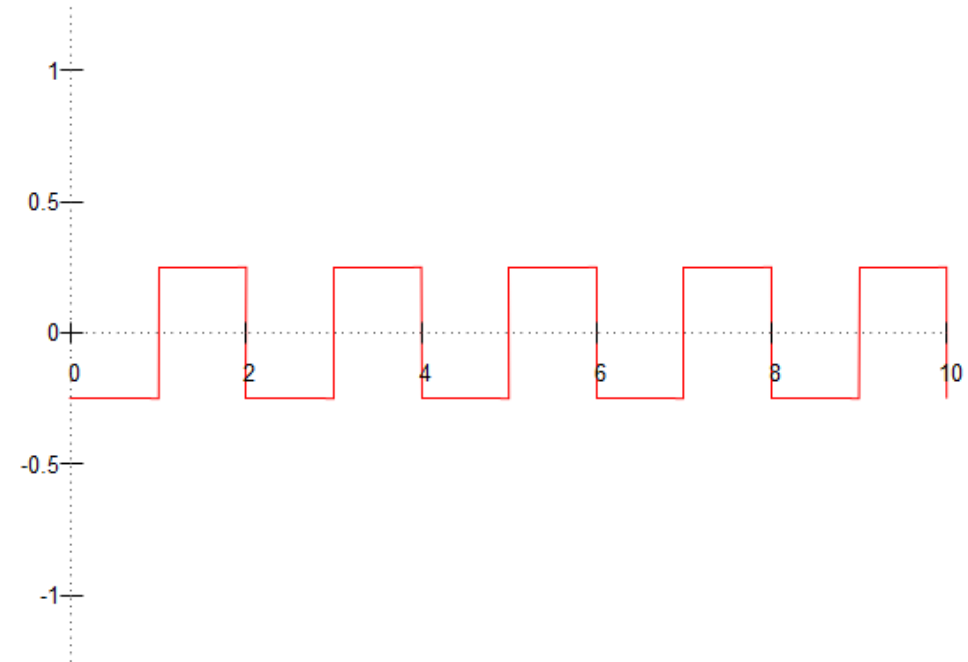
- Transmit

$$x(t) = C(t, \omega'_c(t))$$

# Frequency modulation



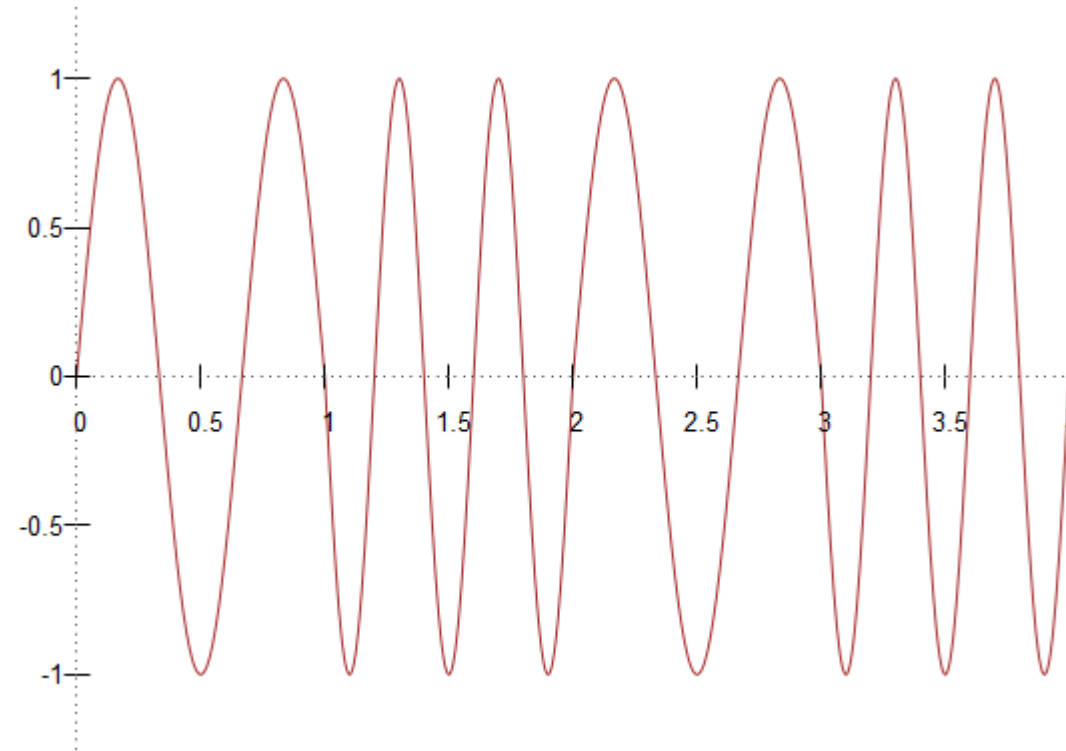
Carrier Wave



Data Wave

# Frequency Modulation

---



# Demodulating FM

---

- Slope FM Detector
- Radio detector
- Foster-Seeley FM detector
- Phase-locked loop FM demodulator
- Quadrature FM demodulator
- Coincidence FM demodulator

# Phase Modulation

---

- Carrier wave described by

$$C(t, \phi_c) = A_c \sin(\omega t + \phi_c)$$

- Data wave described by  $a_m(t) \in [-1, 1]$

- Compute a new phase for the carrier wave

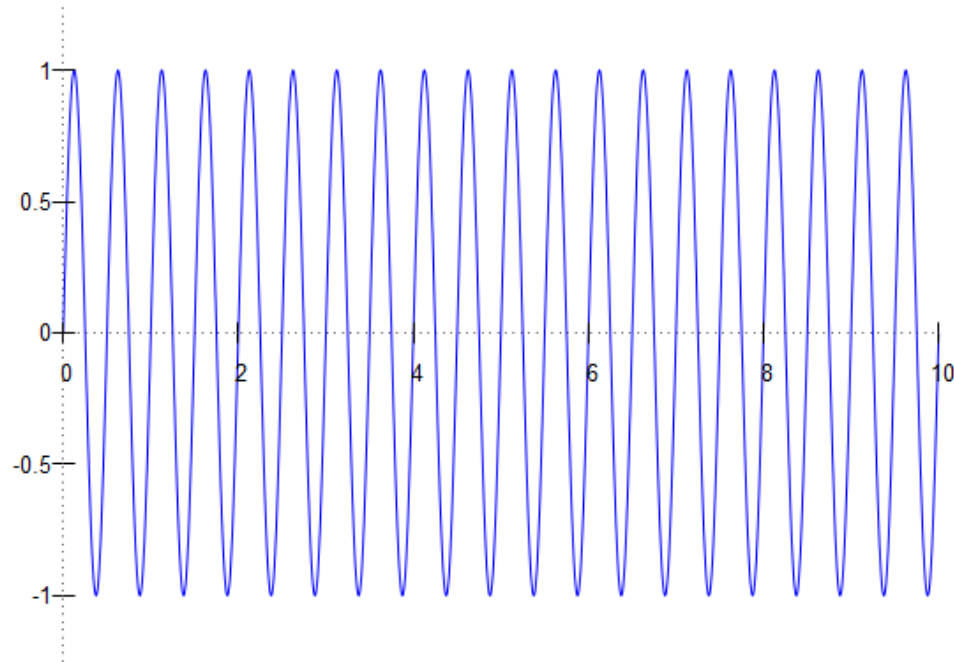
$$\phi'_c = \phi_c + \frac{\Delta\Phi}{\max(a_m)} a_m(t)$$

- Transmit

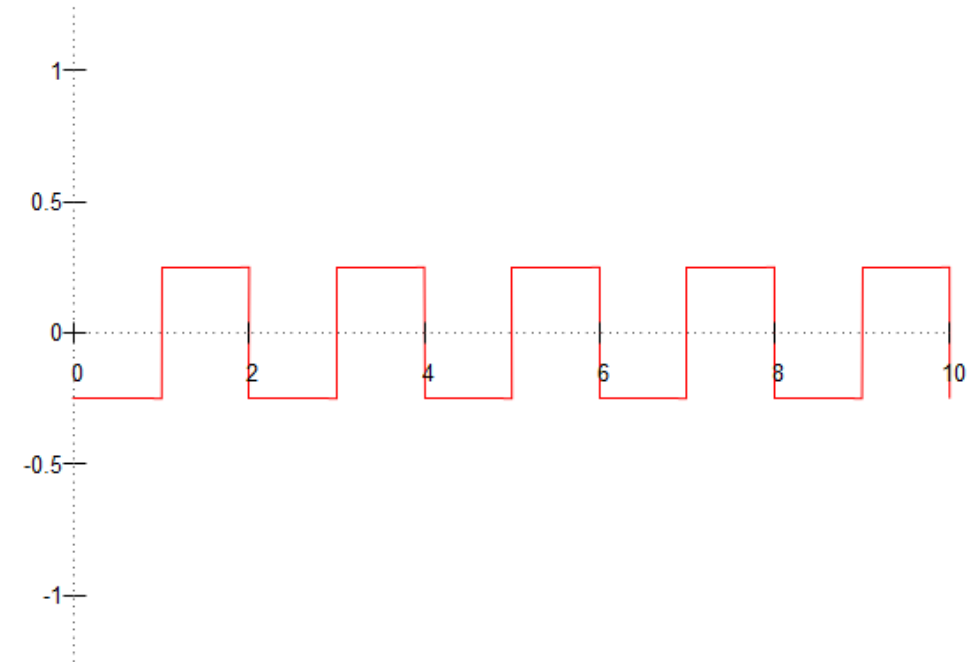
$$x(t) = C(t, \phi'_c(t))$$



# Phase Modulation



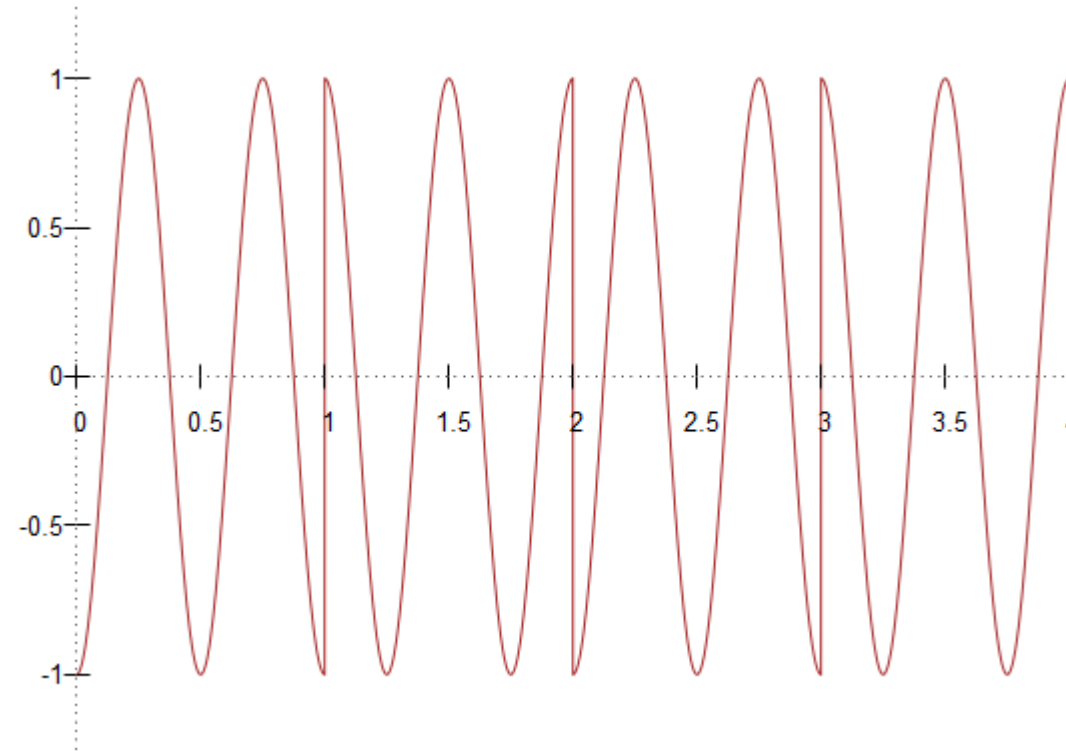
Carrier Wave



Data Wave

# Phase Modulation

---



# Demodulating PM

Recall the received signal is

$$x(t) = C(t, \phi'_c(t)) = A_c \sin(\omega_c t + \phi'_c(t))$$

Also, recall the product of two sinusoids is given by

$$\sin \theta \cdot \sin \varphi = \frac{1}{2} (\cos(\theta - \varphi) - \cos(\theta + \varphi))$$

Then

$$\begin{aligned} & \sin(\omega_c t + \phi_c) \cdot \sin(\omega_c t + \phi'_c) \\ &= \frac{1}{2} [\cos(\omega_c t + \phi_c - (\omega_c t + \phi'_c)) - \cos(\omega_c t + \phi_c + \omega_c t + \phi'_c)] \\ &= \frac{1}{2} [\cos(\phi_c - \phi'_c) - \cos(2\omega_c t + (\phi_c + \phi'_c))] \end{aligned}$$

We can then integrate this waveform over each period to reconstruct the signal.

# Quadrature Amplitude Modulation

---

- Requires changing the phase and amplitude of a carrier sine wave
- Generate two waves: In-phase wave and the quadrature-phase wave

$$I = A_c \cos \varphi$$

$$Q = A_c \sin \varphi$$

- Rewrite carrier wave in terms of  $I$  and  $Q$

$$C(t) = A_c \sin(\omega t + \phi) = A_c \cos\left(\omega t + \phi - \frac{\pi}{2}\right) = A_c \cos(\omega t + \varphi)$$

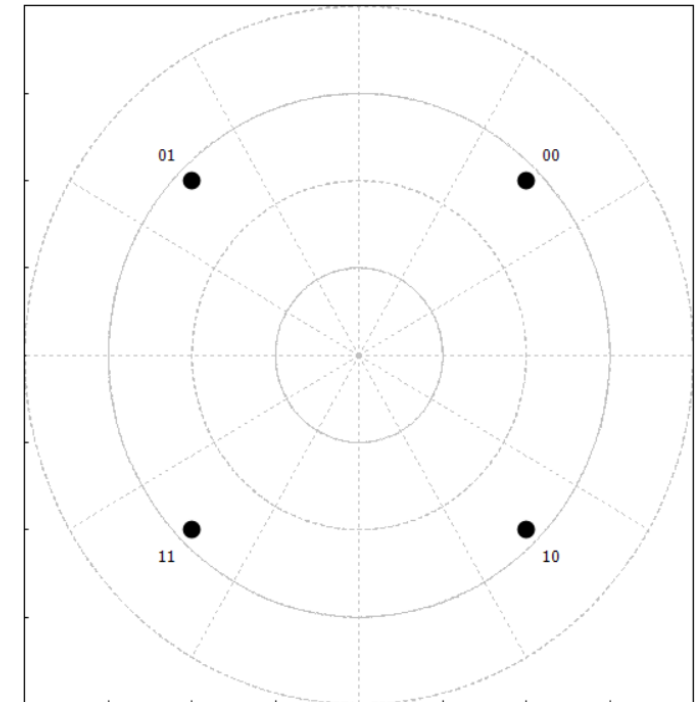
$$C(t) = I \cos(\omega t) - Q \sin(\omega t)$$

- The phase of the carrier wave can be changed by changing the amplitude of  $I$  and  $Q$

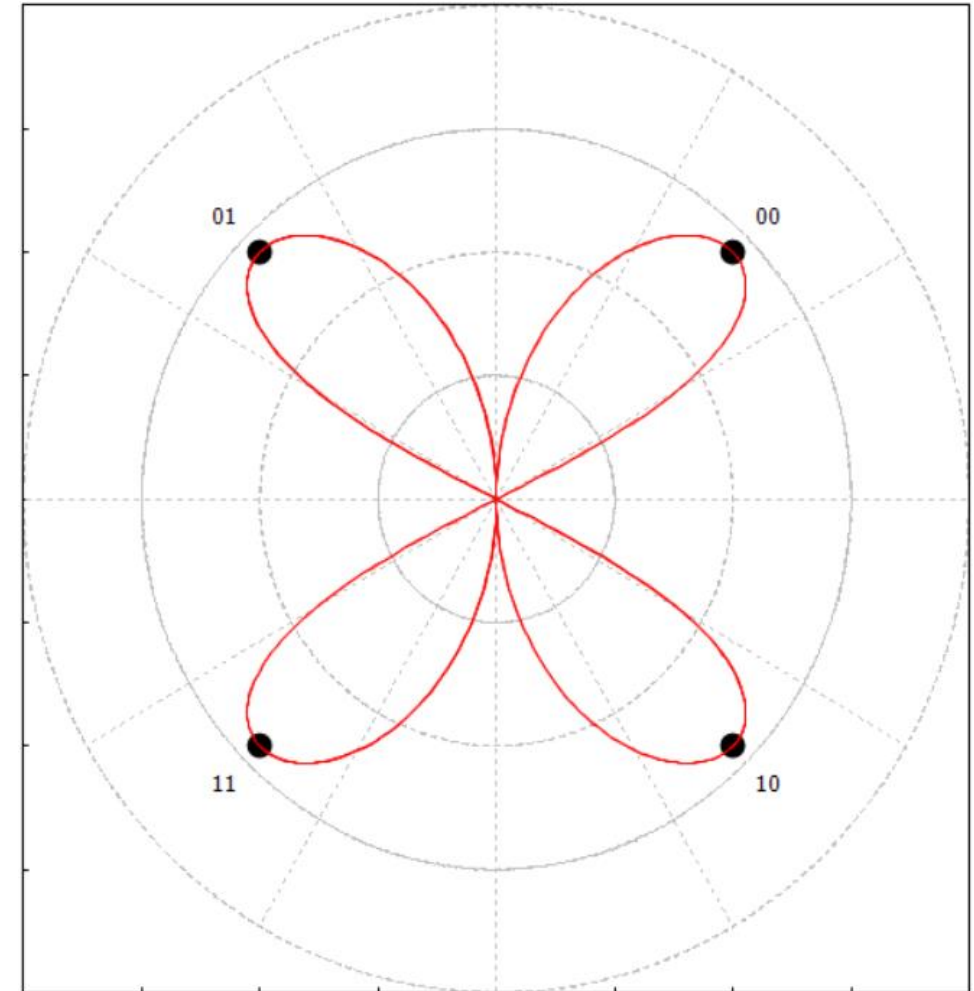
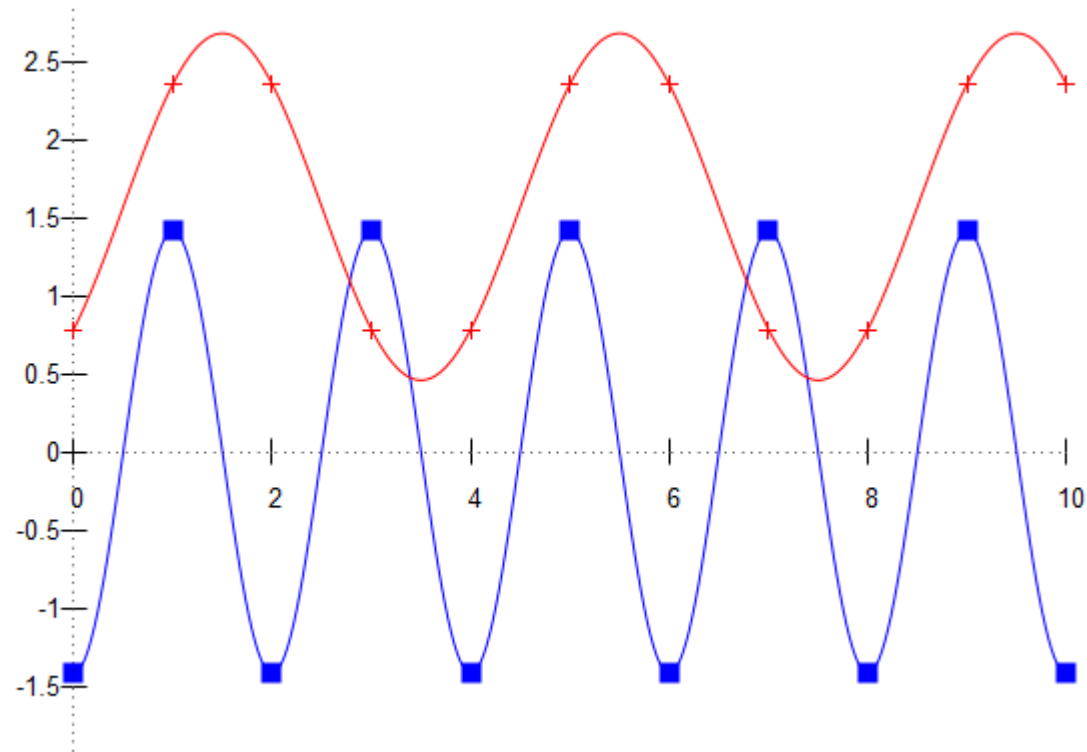
# Quadrature Amplitude Modulation

- We encode data as a function of both *phase* and *amplitude*
- Say we wish to send the bitstring 11011000
  - Determine how many points we want to use (4-QAM, 16-QAM)
  - Split bitstring into encoded point: 11, 01, 10, 00
  - Generate I and Q so that the points are covered

Data	Phase	Amplitude
00	$\pi/4$	1
01	$3\pi/4$	1
11	$\pi/4$	-1
10	$3\pi/4$	-1

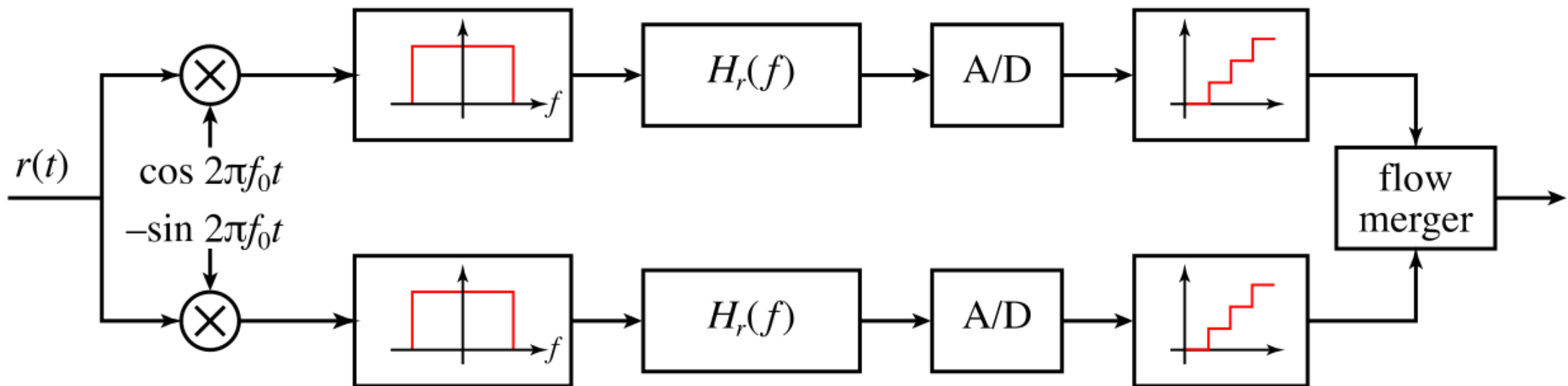


# Quadrature Amplitude Modulation



# Demodulating QAM

- Signal processing is necessary
- We will not go over the math in class



# Physical Layer – IEEE 802.11

- IEEE 802.11 working group defines wireless communication for local area networks
- Standards deal with both physical layer and data link layer

Standard	Release Date	Carrier Frequency (GHz)	Bandwidth (MHz)	Modulation	Maximum Data Rate
802.11	1997	2.4	20	DSSS, FHSS	2 Mbps
802.11b	1999	2.4	20	DSSS	11 Mbps
802.11a	1999	5	20	OFDM	54 Mbps
802.11g	2003	2.4	20	DSSS, OFDM	54 Mbps
802.11n	2009	2.4, 5	20, 40	OFDM	600 Mbps
802.11ac	2013	5	40, 80, 160	OFDM	6.93 Gbps

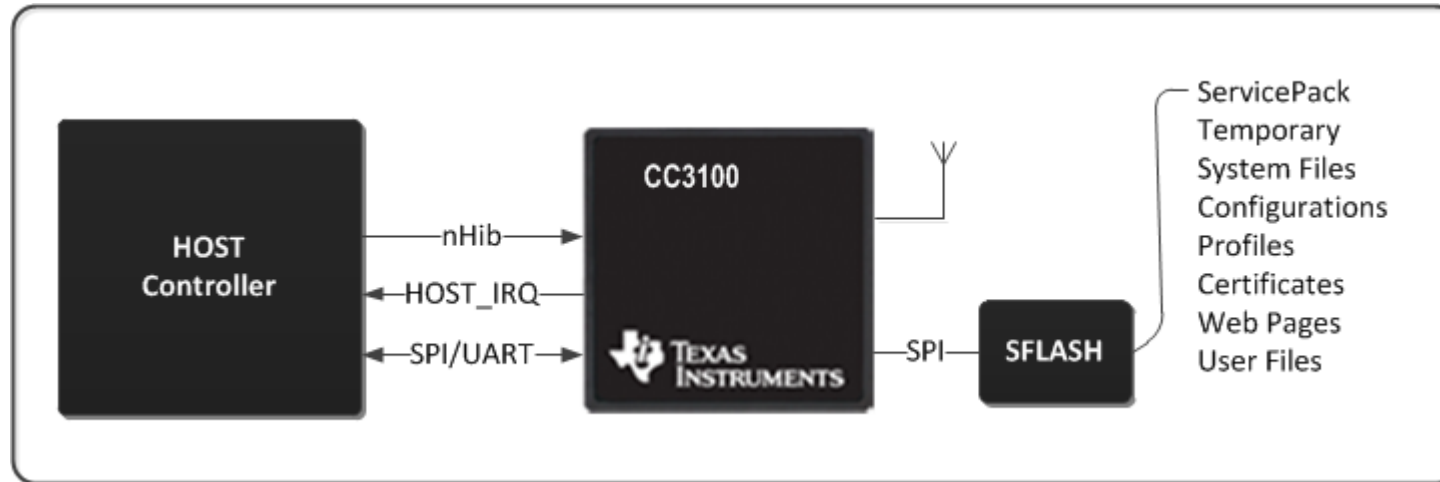


# Texas Instruments CC3100

---

- Single chip solution for wireless networking on 802.11 protocols
- Wi-Fi network processor and power management system
- Contains ARM microcontroller to offload Wi-Fi and IP from external microcontroller
  - Your microcontroller sends commands to the CC3100 for it to perform actions
- Full TCP/IP stack implemented
- Cryptography engine with 256 bit AES for TLS and SSL
- SPI or UART interface

# Texas Instruments CC3100



# Texas Instruments CC3100

- TI provides library of functions to interface with the CC3100

API	Description
<code>sl_WlanConnect()</code>	Connect to an access point
<code>sl_WlanProfileAdd()</code>	Stores a wireless profile on the device
<code>sl_WlanProfileDel()</code>	Deletes a stored profile on the device
<code>sl_WlanProfileget()</code>	Retrieves information regarding a stored profile
<code>sl_WlanPolicySet()</code>	Sets the connection policy on how to treat access point
<code>sl_WlanEvtHdlr()</code>	Handles asynchronous WLAN events
<code>sl_NetAppEvtHdlr()</code>	Handles asynchronous IP events

# Texas Instruments CC3100

---

```

/* connecting to an access point */
const char* const SSID = "my_ssid";
const char password[] = {0x1c, 0xce, 0xdb, 0x00, 0xda, 0xca, 0xfe};
SlSecParams_t sec_params = {
    .Key = password,
    .KeyLen = sizeof(password),
    .Type = SL_SEC_TYPE_WPA_WPA2
};
int ret = sl_WlanConnect(SSID, strlen(SSID), 0, &sec_params, 0);

```

# Texas Instruments CC3100

---

```

/* transmit datagrams (UDP) */
SlSockAddrIn_t addr = {
    .sin_family = SL_AF_INET
};
addr.sin_port = sl_Htons(80);
addr.sin_addr.sl_addr = sl_Htonl(0x0ae338b9);
int soc_fildes = sl_Socket(SL_AF_INET, SL_SOCKET_DGRAM);

ret = sl_SendTo(soc_fildes, data, data_len, 0, &addr, sizeof(addr));
  
```

---

# IEEE 802.11

---

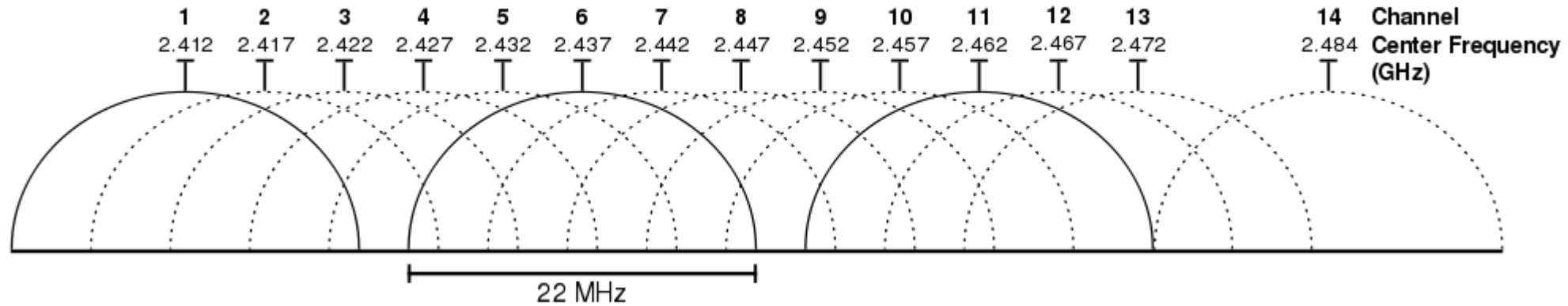
PHYSICAL AND DATA LINK LAYER PROTOCOL, SECURITY

# 802.11 Topology

---

- Physical Layer: radio waves (2.4/5 GHz Spectrum)
- Can work with two topologies
  - Star: central AP managing all connections
  - Mesh: no central authority

# 802.11 Spectrum





# 802.11 MAC Frame Format

	0								1								2								3							
	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	Frame control																Duration/Connection ID															
4	Address 1																															
8	Address 1																Address 2															
12	Address 2																															
16	Address 3																															
20	Address 3																Sequence Control															
24	Address 4																															
28	Address 4																Frame Body															
...	Frame Body																															
...	CRC																															

# 802.11 MAC Frame Format

Field	Description
Frame Control	Indicates whether frame is a control frame, management frame, or data frame. Provides control information, which includes whether a frame is coming or going to the Distribution System (DS, or access point), any fragmentation information, and privacy information.
Duration or Connection ID	If used as a duration field, indicates the time in microseconds the channel will be allocated for successful transmission of the frame. Otherwise, contains association or connection identifier.
Addresses	The Media Access Control (MAC) addresses involved in the transaction.
Frame body	Up to 2312 bytes are transmitted as part of the frame body. The frame body may be empty.
CRC	Cycling Redundancy Check performed on the frame to ensure integrity.

# 802.11 Frame Control Field

	0								1							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	Protocol Version		Type		Subtype				To DS	From DS	More Frag	Retry	Pwr Mgt	More Data	Prot	Order

Field	Description
Protocol Version	Always 0 for standard 802.11
Type	Frame type: data, management, control
Subtype	Frame subtype
To DS	If set, indicates that frame is for DS
From DS	If set, indicates frame is coming from DS
Retry	Set in case of retransmission
More Frag	Frame is followed by more fragments.
Pwr Mgt	If set, station goes into Power Save mode

# 802.11 Frame Control Field

0	0								1							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	Protocol Version		Type		Subtype				To DS	From DS	More Frag	Retry	Pwr Mgt	More Data	Prot	Order

Field	Description
More Data	If set, AP has more buffered data for station in Power Save mode.
Prot	Protected Frame. Frame is protected by link layer security protocols (WEP, WPA, WPA2)
Order	Frames and fragments can be transmitted in order at the cost of additional processing by both the sending and receiving MAC. When strict ordering delivery is employed, this bit is set to 1.

# 802.11 Frame Types – Selected

Type		Subtype				Class	Subtype Name
5	4	3	2	1	0		
0	0	0	0	0	0	Management Frame	Association Request
0	0	0	0	0	1	Management Frame	Association Response
0	0	0	0	1	0	Management Frame	Reassociation Request
0	0	0	0	1	1	Management Frame	Reassociation Response
0	0	1	0	0	0	Management Frame	Beacon
0	0	1	0	1	0	Management Frame	Dissociation
0	1	1	0	1	1	Control Frame	Request to send
0	1	1	1	0	0	Control Frame	Clear to send
1	0	0	0	0	0	Data Frame	Data
1	0	1	0	0	0	Data Frame	QoS Data (802.11e)

# 802.11 Frame Control – DS Bits

	To DS = 0	To DS = 1
From DS = 0	All Management and Control Frames. Data frames with an IBSS (never infrastructure data frames).	Data frames transmitted from a wireless station in an infrastructure network.
From DS = 1	Data frames received for a wireless station in an infrastructure network.	Data frames on a wireless bridge.

# 802.11 AP Association

---

- Wireless station listens for beacon frames to find AP (Passive Scanning)
- Wireless station sends Probe Request frames to solicit responses from a network with a given name on every channel (Active Scanning)
- Once AP is found, three connection states
  - Not authenticated or associated
  - Authenticated but not yet associated
  - Authenticated and Associated
- Series of management frames are exchanged (Layer 2) to get to the last state

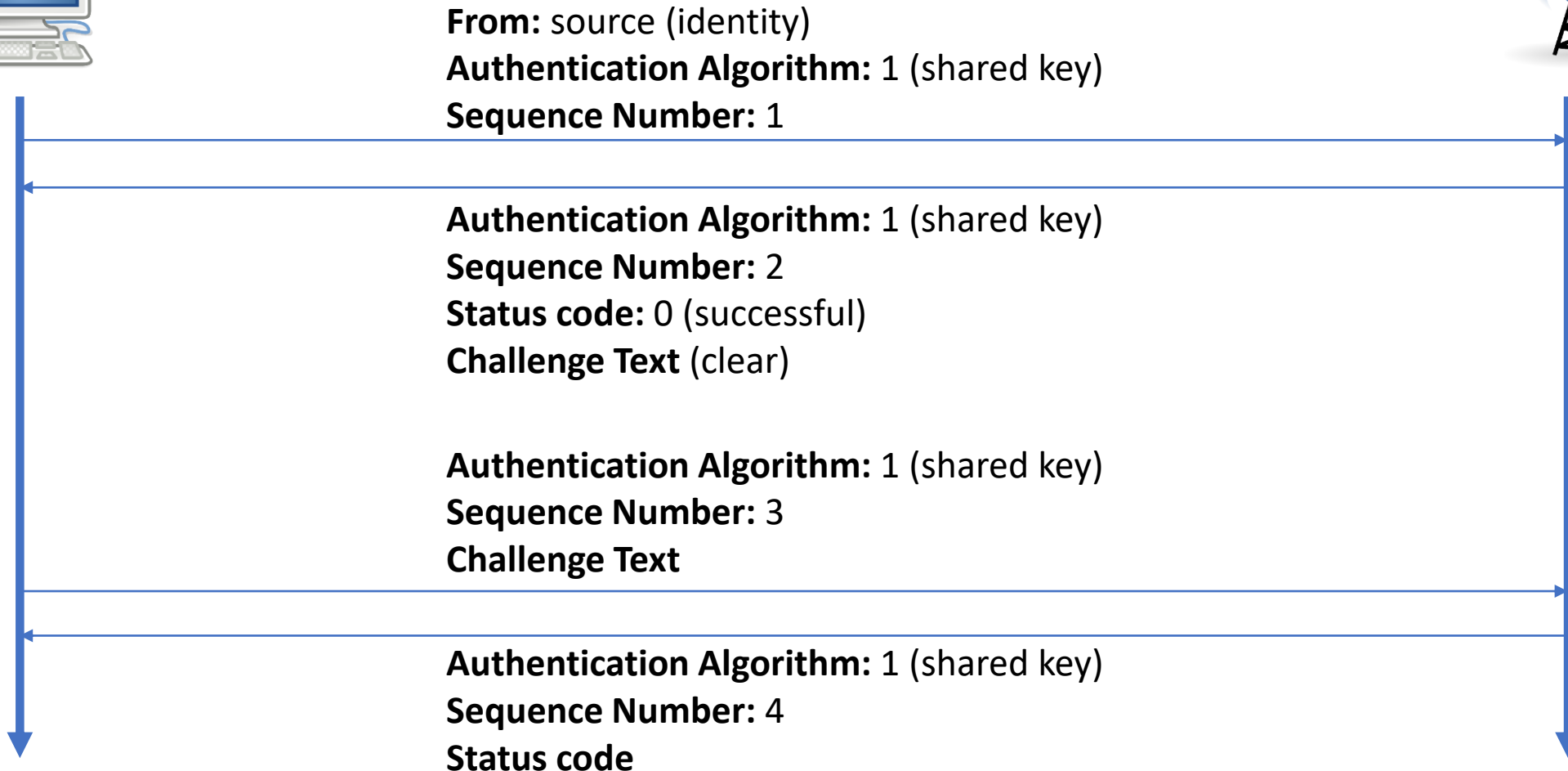
# 802.11 AP Association

---

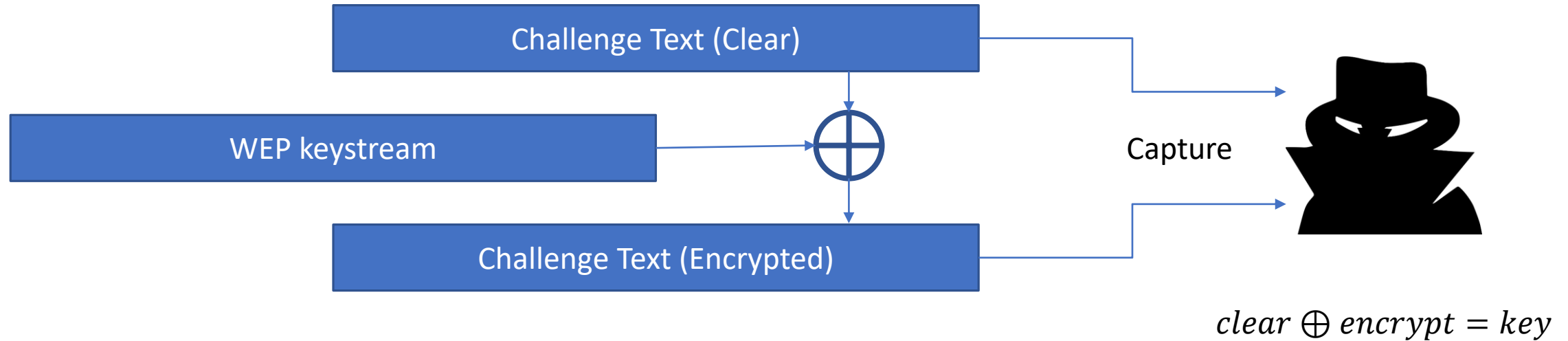
- Wireless station performs low-level 802.11 authentication with AP
  - Open Authentication: AP accepts wireless station without verifying identity
  - Shared-key Authentication: uses WEP
- Mobile station sends association request
  - If station has not authenticated, it receives Deauthentication frame as response
- If association request is granted, AP responds with status code 0 and the Association ID
  - Unsuccessful association include only a status code
- AP can now process frames for the mobile station



# 802.11 Shared-key Authentication



# 802.11 Shared-key Authentication



# Other Issues with WEP

---

- Reuse of keystream
  - CRC is not cryptographically secure as a compression function
  - Manual key management
  - 40 bit shared secret
  - Reuse of IVs
  - Known attacks against RC4
    - Possible to do key recovery
- Do not deploy WEP networks



# 802.11 Disconnecting from AP

---

- Any side can send a Deauthentication Frame when all communications are terminated
  - When dissociated, a wireless station can still be authenticated to the AP
- Any side can terminate the association by sending a Disassociation Frame
  - A station can send a Disassociation Frame without deauthenticating (e.g. in the case of roaming)
- For proper connection termination, both frames should be sent
- These frames are sent in clear text (management frame control fields)
- Attacker can inject them into network!

# 802.11 Denial of Service

---

- Attacker obtains MAC address of AP and victim station(s)
  - Information is readily available, sent in clear text
- Attacker sends spoofed deauthentication request to AP
- Victim station(s) attempt to communicate with AP
- AP sends Disassociation frame to victim station(s)
- Victim(s) evicted from network
- Mitigation: none, this is part of the protocol

# 802.11 Denial of Service

---

- Attacker can then listen for new handshake
  - Can brute force WPA/WPA2 key
- Attacker can force victim station(s) to connect to evil twin AP
  - Can now fully control station(s) traffic
  - Opens avenues for MITM, traffic shaping...