
SAPIENZA
UNIVERSITÀ DI ROMA


Northeastern

October 10th 2018

# Networking concepts review

Credits to Marco Spaziani Brunella and Manuel Campo

**Network Infrastructures course**

**Speaker**: Ludovico Ferranti

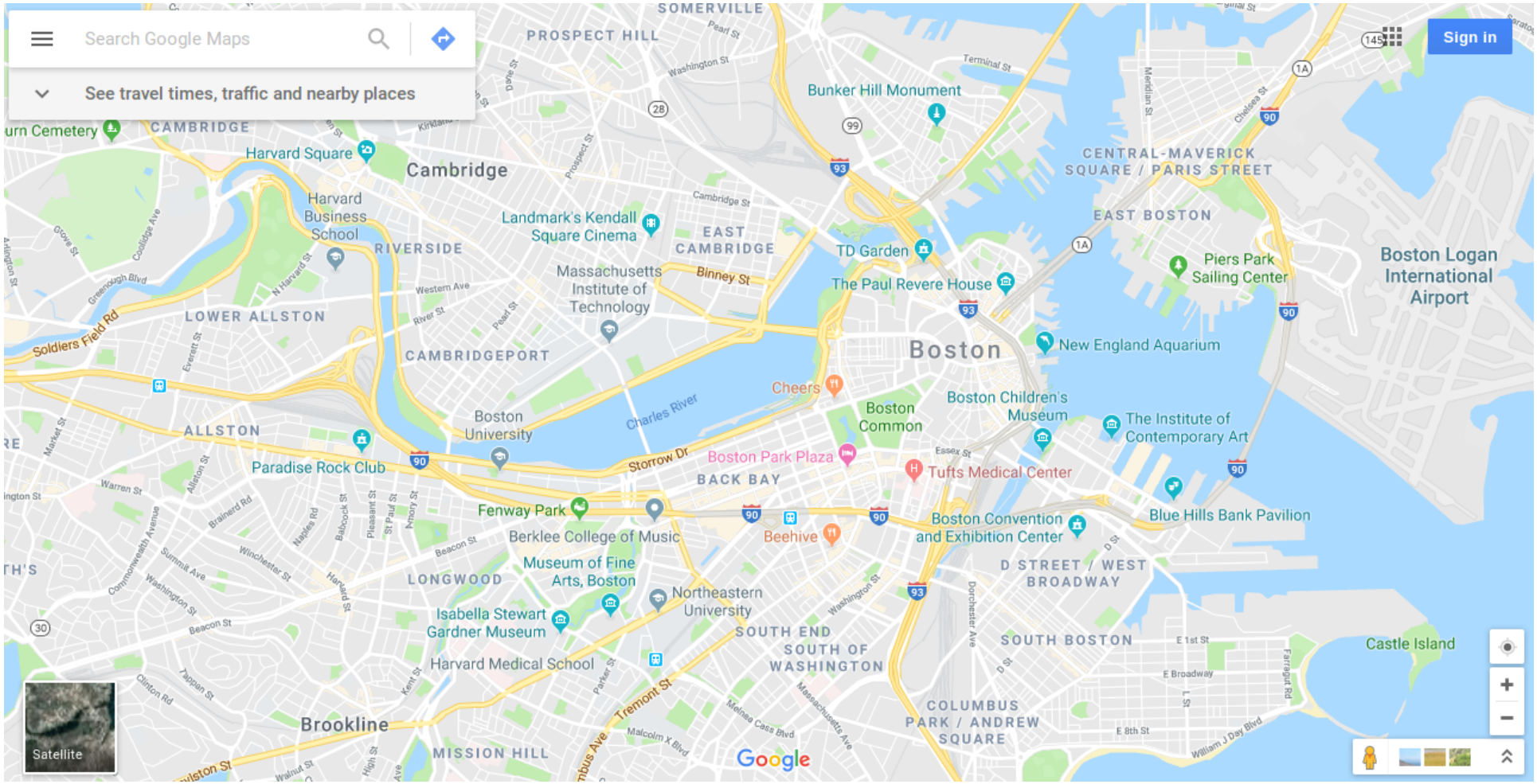
# Who's talking?

- M.Sc. in Artificial Intelligence and Robotics at Sapienza – University of Rome

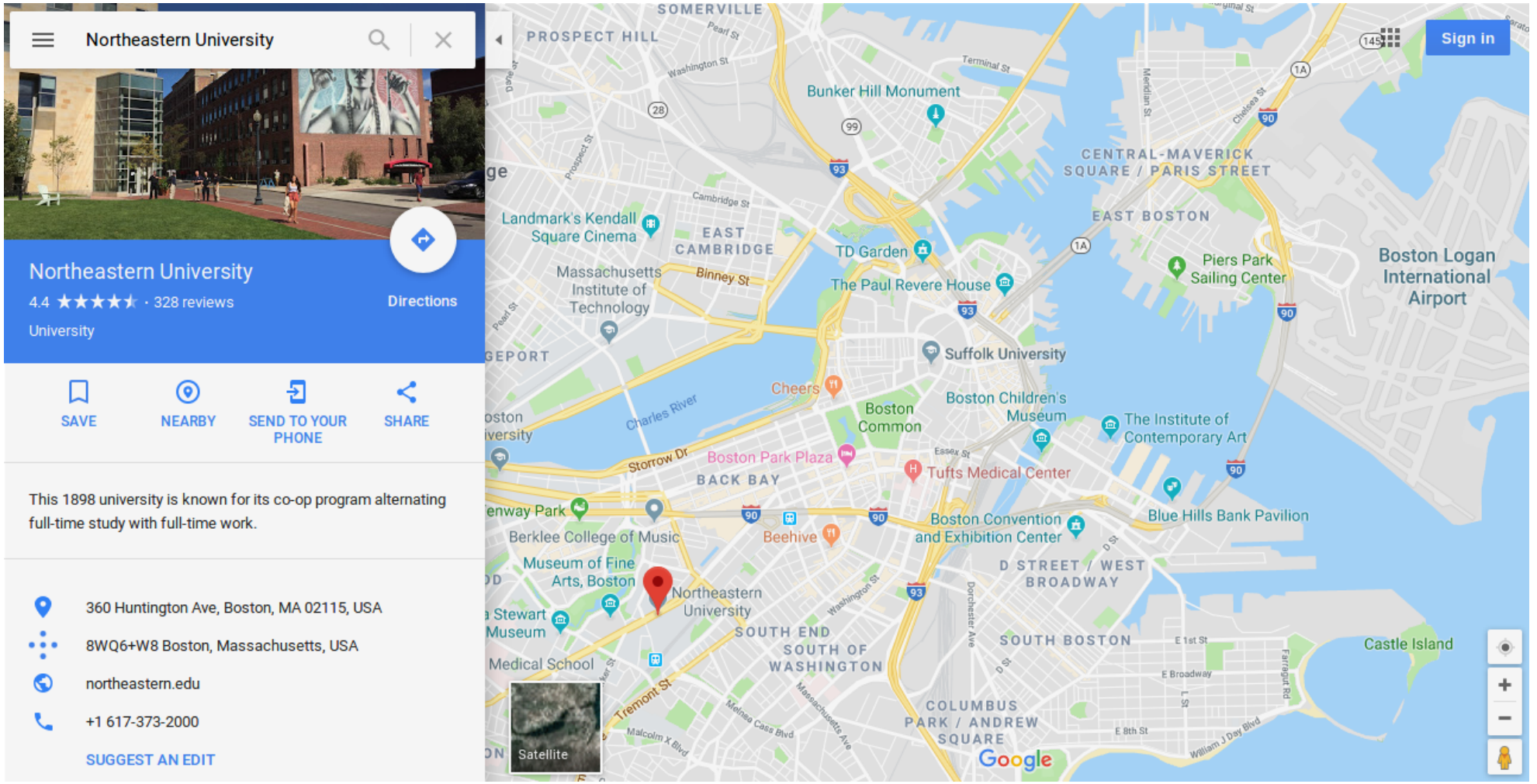
- 3rd year of double Phd Program at Sapienza University of Rome (ICT) and Northeastern University – Boston MA (ECE)

- Research area: UAVs, Mobile Robotics, Disaster recovery networks

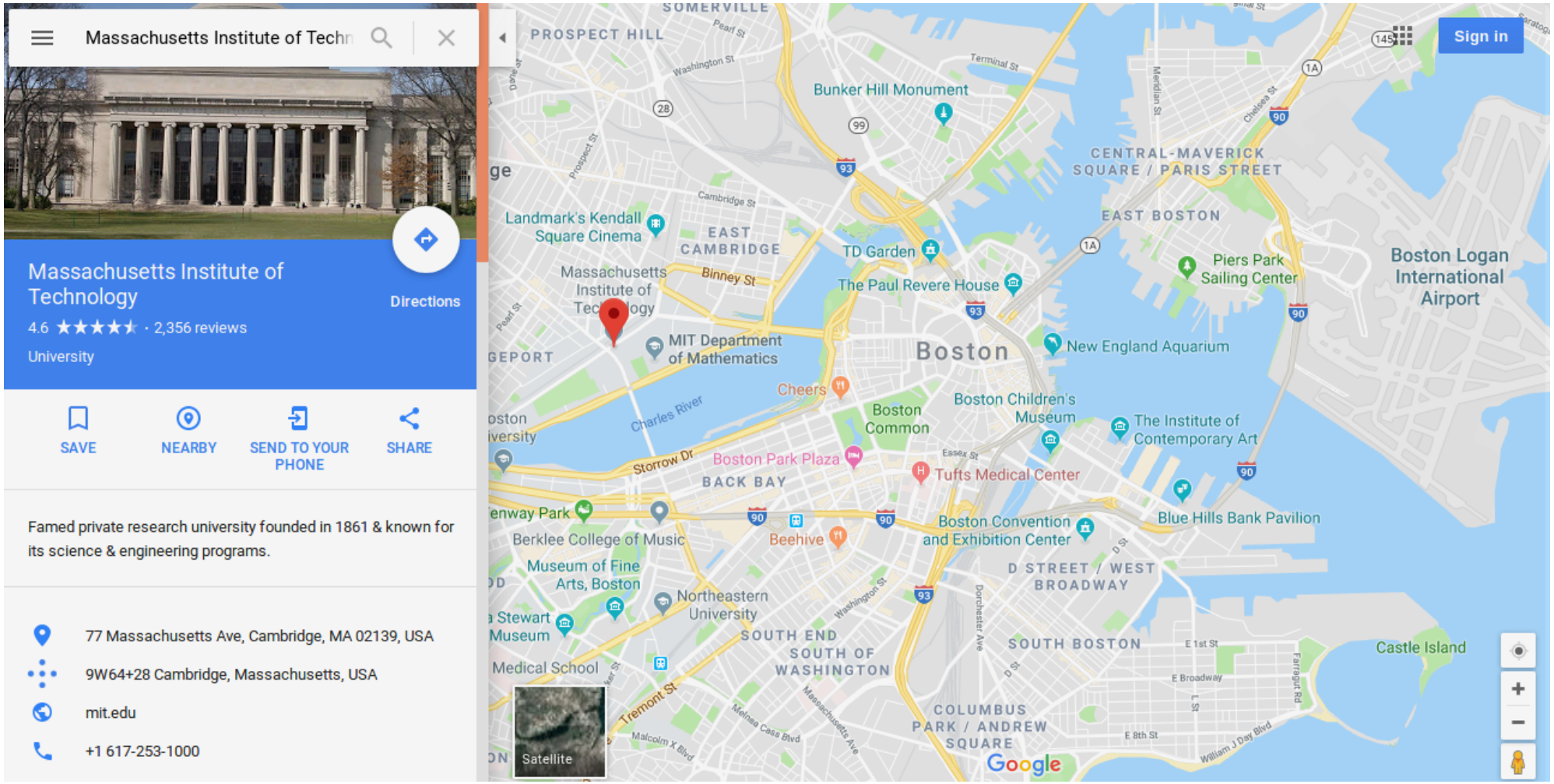
# Boston


Search Google Maps
See travel times, traffic and nearby places
Sign in
SOMERVILLE
PROSPECT HILL
Bunker Hill Monument
CAMBRIDGE
Harvard Square
Cambridge
Harvard Business School
RIVERSIDE
Landmark's Kendall Square Cinema
EAST CAMBRIDGE
Massachusetts Institute of Technology
Binney St
TD Garden
The Paul Revere House
CENTRAL-MAVERICK SQUARE / PARIS STREET
EAST BOSTON
Piers Park Sailing Center
Boston Logan International Airport
LOWER ALLSTON
Soldiers Field Rd
CAMBRIDGEPORT
Boston
New England Aquarium
Cheers
Boston Common
Boston Children's Museum
The Institute of Contemporary Art
Charles River
Boston University
ALLSTON
Paradise Rock Club
Storrow Dr
Boston Park Plaza
Tufts Medical Center
BACK BAY
Fenway Park
Berklee College of Music
Beehive
Boston Convention and Exhibition Center
Blue Hills Bank Pavilion
D STREET / WEST BROADWAY
LONGWOOD
Museum of Fine Arts, Boston
Isabella Stewart Gardner Museum
Northeastern University
SOUTH END
SOUTH OF WASHINGTON
SOUTH BOSTON
Castle Island
Harvard Medical School
Brookline
MISSION HILL
COLUMBUS PARK / ANDREW SQUARE
Tremont St
Satellite
Google

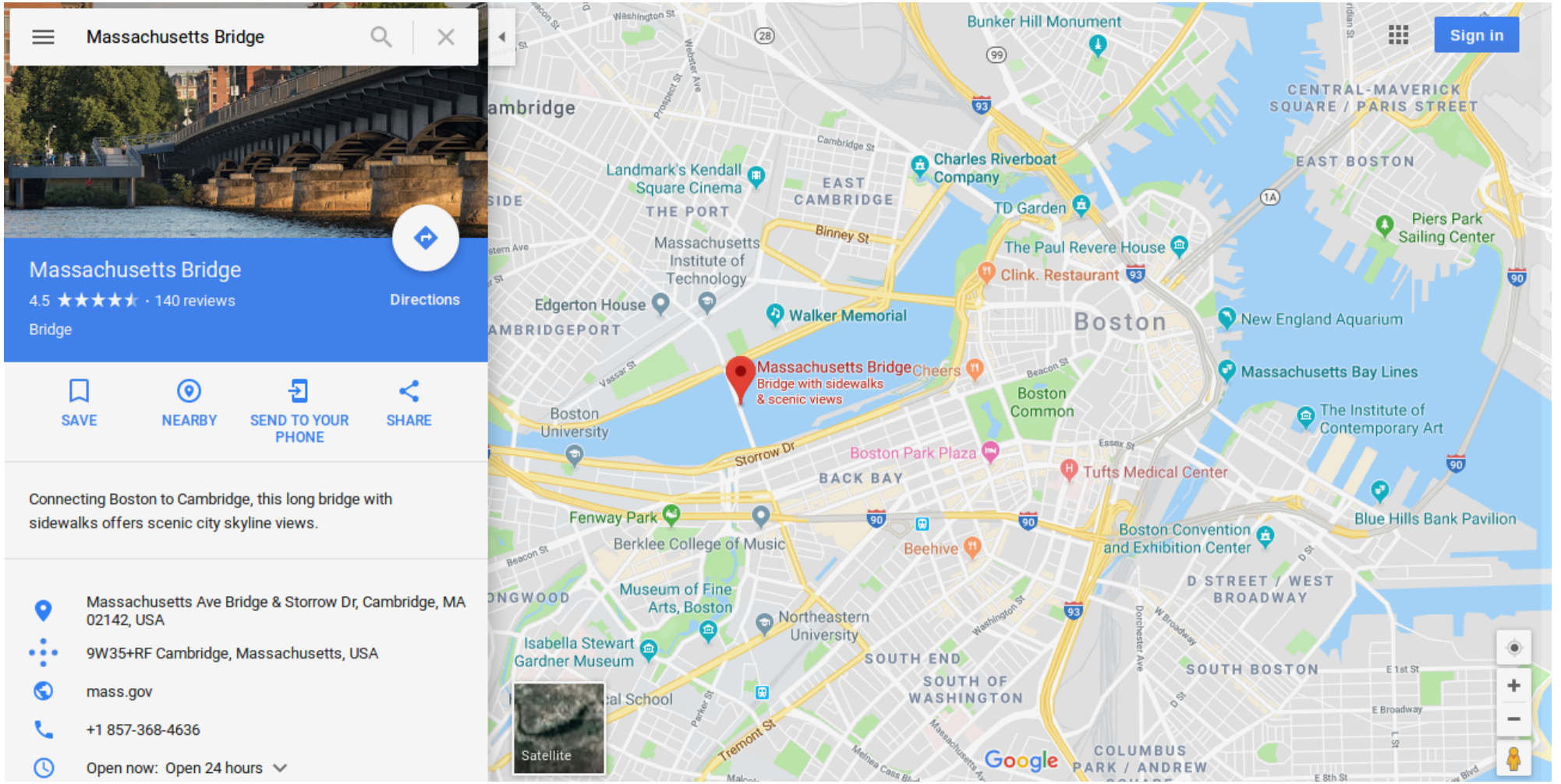
# Northeastern University



# MIT

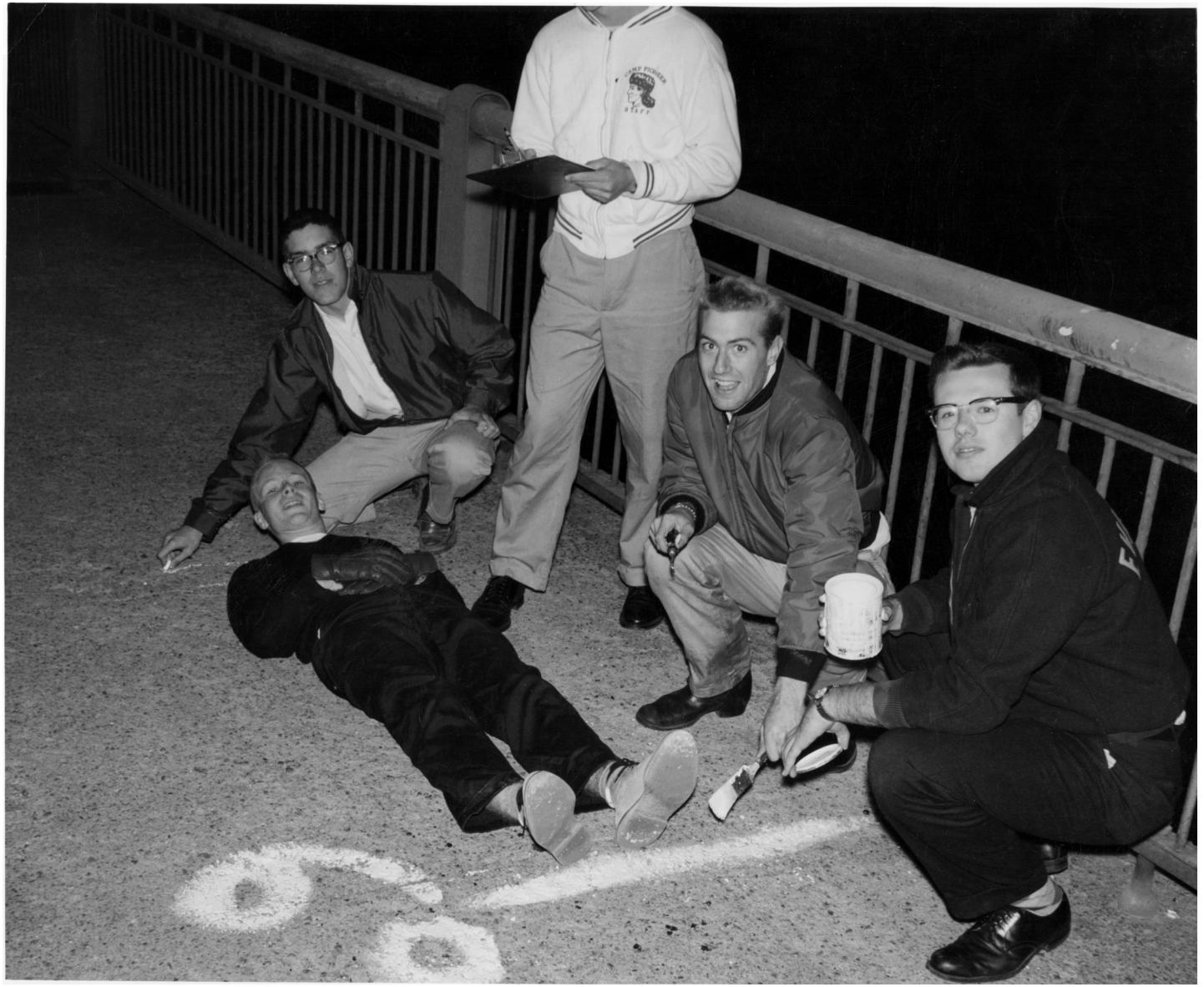

Massachusetts Institute of Techn
Massachusetts Institute of Technology
4.6 · 2,356 reviews
University
Directions
SAVE
NEARBY
SEND TO YOUR PHONE
SHARE
Famed private research university founded in 1861 & known for its science & engineering programs.
77 Massachusetts Ave, Cambridge, MA 02139, USA
9W64+28 Cambridge, Massachusetts, USA
mit.edu
+1 617-253-1000
Sign in
Bunker Hill Monument
PROSPECT HILL
SOMERVILLE
CENTRAL-MAVERICK SQUARE / PARIS STREET
EAST BOSTON
Landmark's Kendall Square Cinema
EAST CAMBRIDGE
Massachusetts Institute of Technology
Binney St
TD Garden
The Paul Revere House
Piers Park Sailing Center
Boston Logan International Airport
MIT Department of Mathematics
Boston
New England Aquarium
Cheers
Charles River
Boston Common
Boston Children's Museum
The Institute of Contemporary Art
Storrow Dr
Boston Park Plaza
BACK BAY
Tufts Medical Center
Fenway Park
Berklee College of Music
Beehive
Boston Convention and Exhibition Center
Blue Hills Bank Pavilion
Museum of Fine Arts, Boston
Northeastern University
D STREET / WEST BROADWAY
Stewart Museum
SOUTH END
SOUTH OF WASHINGTON
SOUTH BOSTON
Castle Island
Medical School
Tremont St
COLUMBUS PARK / ANDREW SQUARE
Satellite
Google

# Mass Ave Bridge



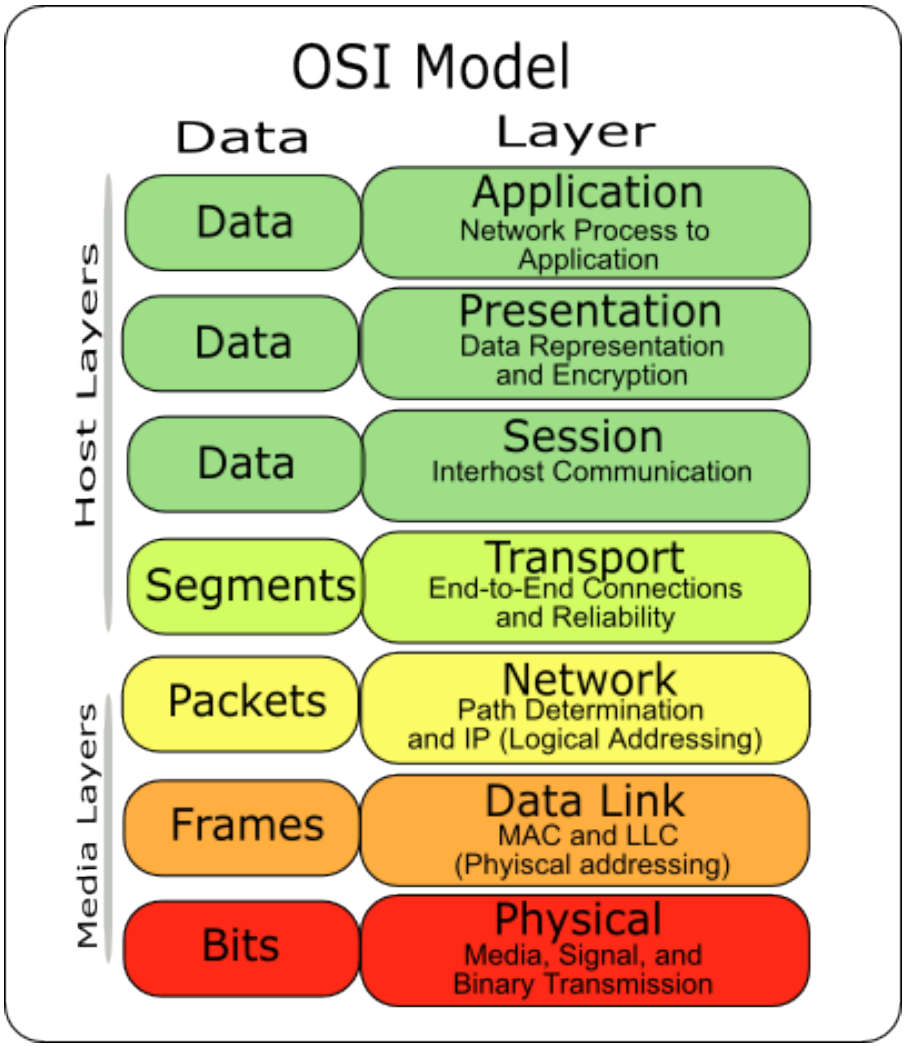
# Mass Bridge


# Oliver R. Smoot



# International Standard Organization


# Open System Interconnection




- OSI model resumes an earlier proposal of the ISO, so the acronym used is **ISO/OSI**.
- ISO/OSI model divides network project in 7 different levels, called **LAYERs.**
- Each layer has its features and it is able to use services of underlying layer transparently and provides its services to the upper one.
- ISO/OSI model defines the rules and the protocols of each layer.

# How to remember layers?

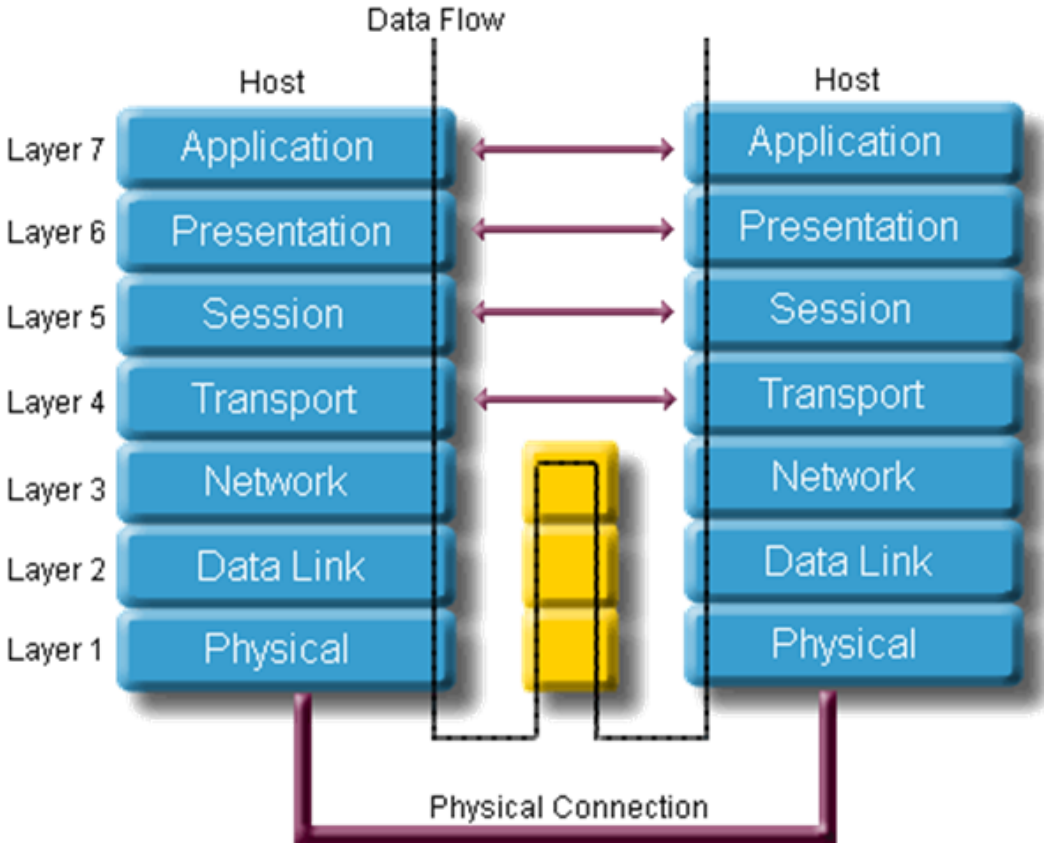
Sentences to help remember the layers of the OSI Reference Model:

**A**ll **P**eople **S**eem **T**o **N**eed **D**ata **P**rocessing.

Or

**P**lease **D**o **N**ot **T**hrow **S**ausage **P**izza **A**way.

# ISO/OSI Model




- Each level of a nodes is able to communicate virtually with the equivalent level of the other nodes, even if only the lower layers (Physical level) are actually in contact with each other.
- A single node can be source or final destination of information.
- **Network Nodes** had only the first three layers.


# ISO/OSI Model


**Application layer** – The spot where users actually communicate to the computer.

**Presentation layer** – Presents data to the Application layer and is responsible for data translation and code formatting.

**Session layer** – Responsible for setting up, managing, and then tearing down sessions between Presentation layer entities.

**Transport layer** – Segments and reassembles data into a data stream.

**Network layer** – Manages devices addressing, tracks the location of devices on the network, and determines the best way to move data.

**Data Link layer** – Provides the physical transmission of the data and handles error notification, network topology, and flow control.

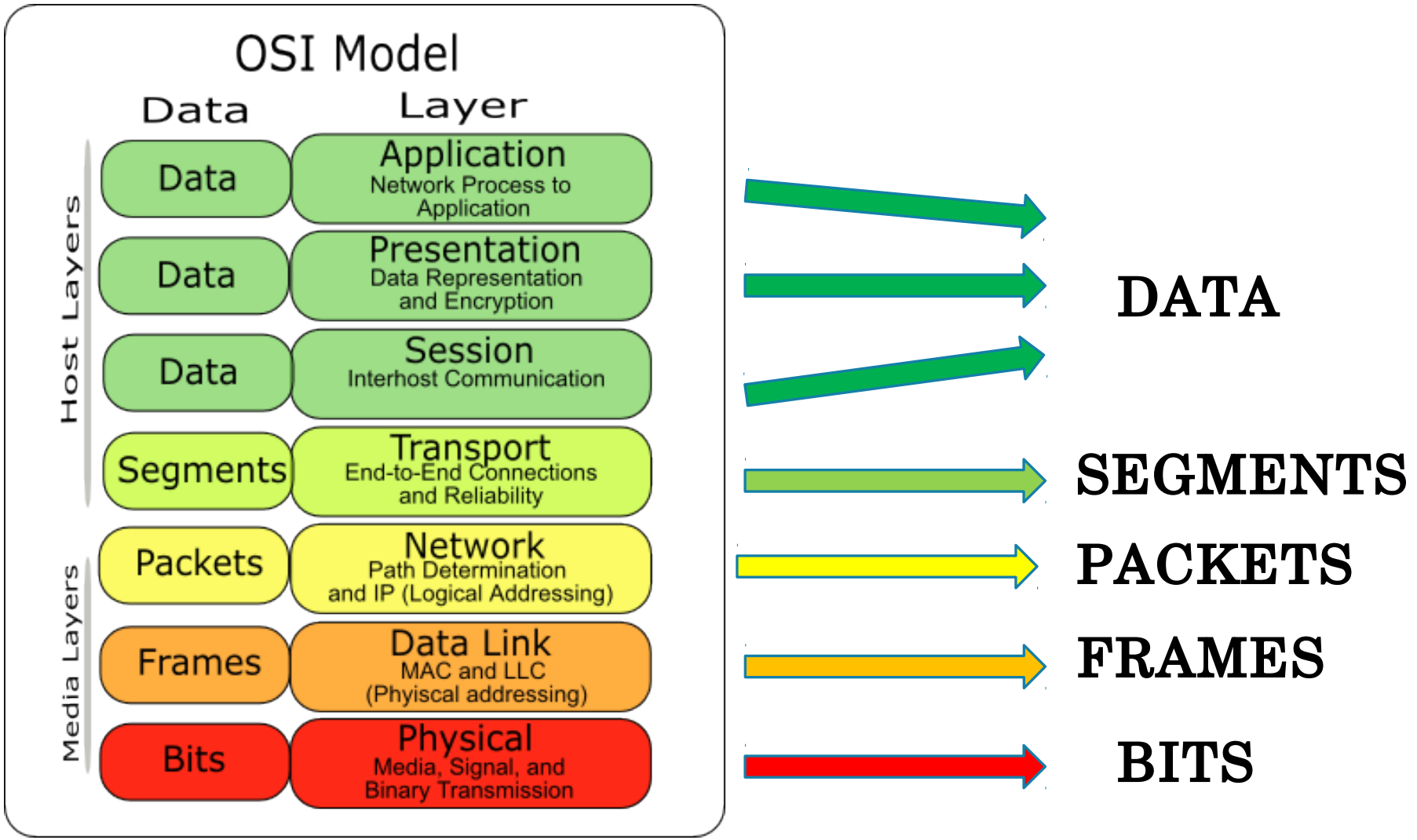
**Physical layer** – Sends and receives bits.

# Data unit

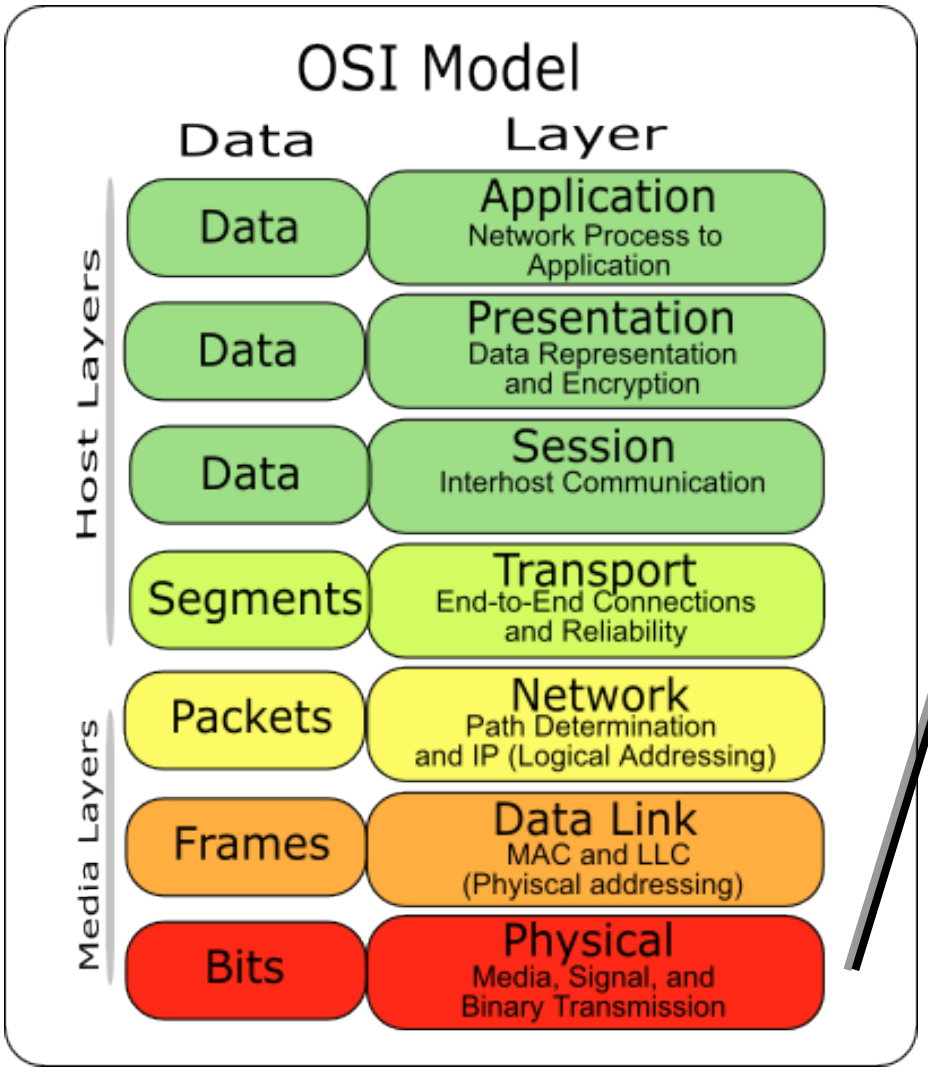
- Each layer of the OSI model at the source must communicate with its peer layer at the destination.
- During the protocols at each layer exchange packets of information called protocol data units (PDUs) between peer layers.

- **Bits** are sent between **physical layer** peers.
- **Frames** are sent between **data link layer** peers.
- **Packets** are sent between **network layer** peers.
- **Segments** are sent between **transport layer** peers.

# Data unit


OSI Model
Data
Layer
Host Layers
Media Layers
Data
Application
Network Process to Application
Data
Presentation
Data Representation and Encryption
Data
Session
Interhost Communication
Segments
Transport
End-to-End Connections and Reliability
Packets
Network
Path Determination and IP (Logical Addressing)
Frames
Data Link
MAC and LLC (Phyiscal addressing)
Bits
Physical
Media, Signal, and Binary Transmission
DATA
SEGMENTS
PACKETS
FRAMES
BITS

# PHY




**Physical Layer (PHY):** does two things: it sends bits and receives bits.

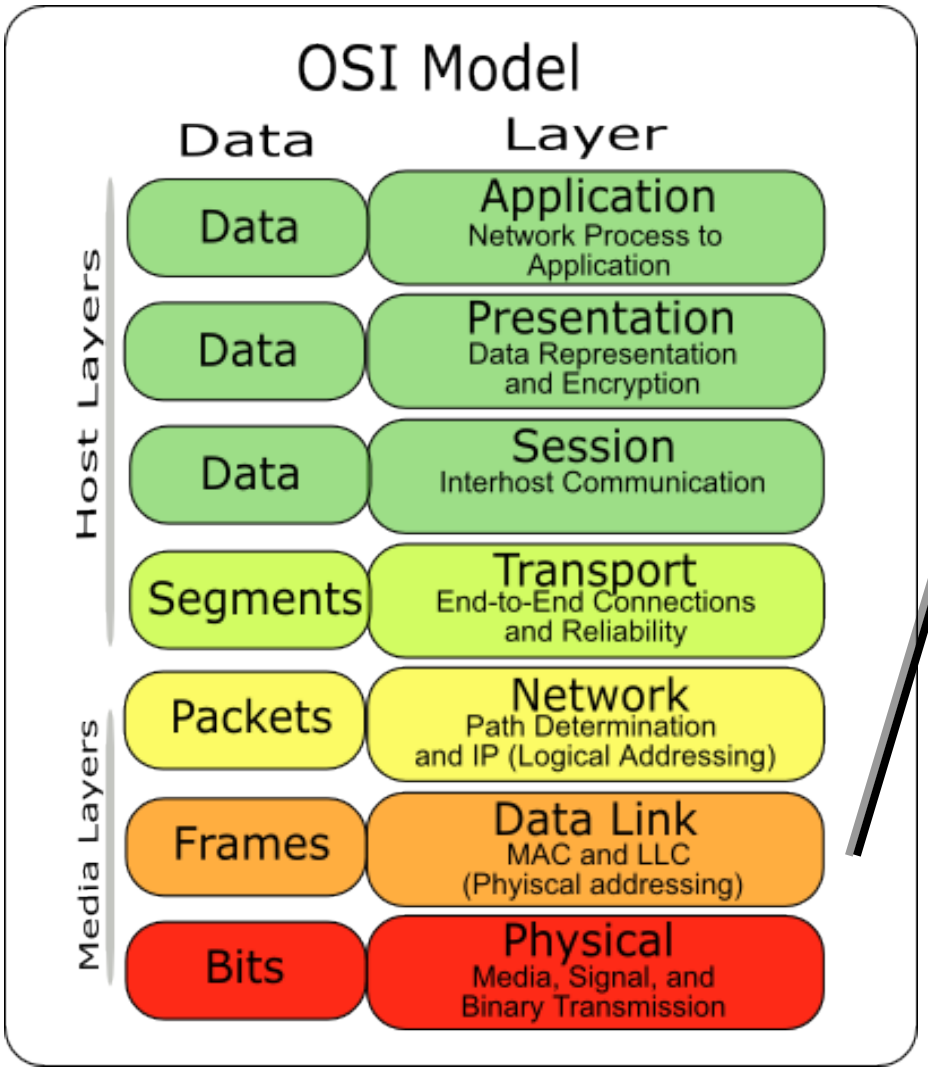
- Defines the electrical, mechanical, procedural and functional specifications for:

1. **Activating**
2. **Maintaining**
3. **Deactivating**

the physical link between end systems.

- The Physical layer sends ***Bits*** between peer physical layers.
- **Devices**: **modem, repeater, hub, bus**

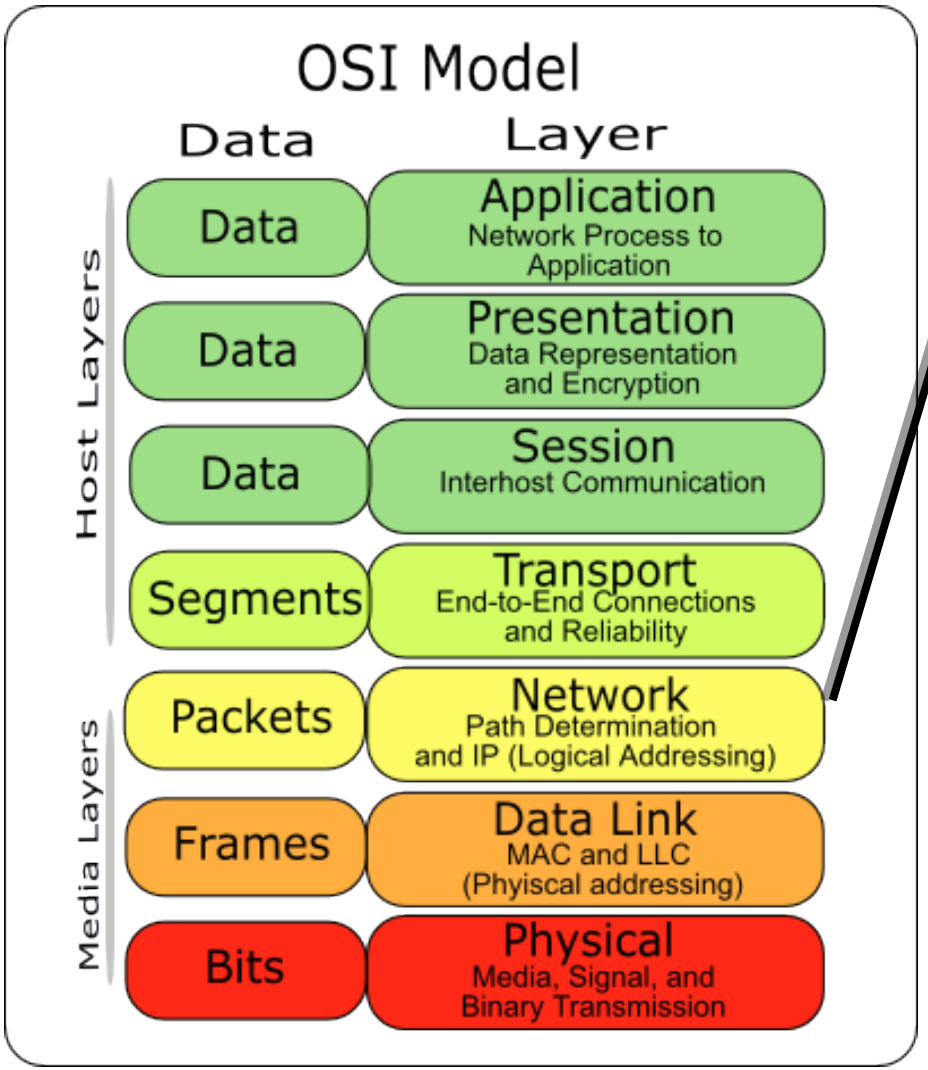
# DLL




- **Data Link Layer (DLL):** transfers data (frames) between *network entities*. DLL *detects and corrects errors* that may occur in the PHY, manages network topology, and flow control.
- DLL ensure that messages are delivered to the proper device on a LAN using hardware addresses, and translates messages from the Network layer into bits for the Physical layer to transmit.
- DLL protocols: Ethernet, Point-to-Point Protocol (PPP).

- **Devices: switch, bridge**

# Network Layer



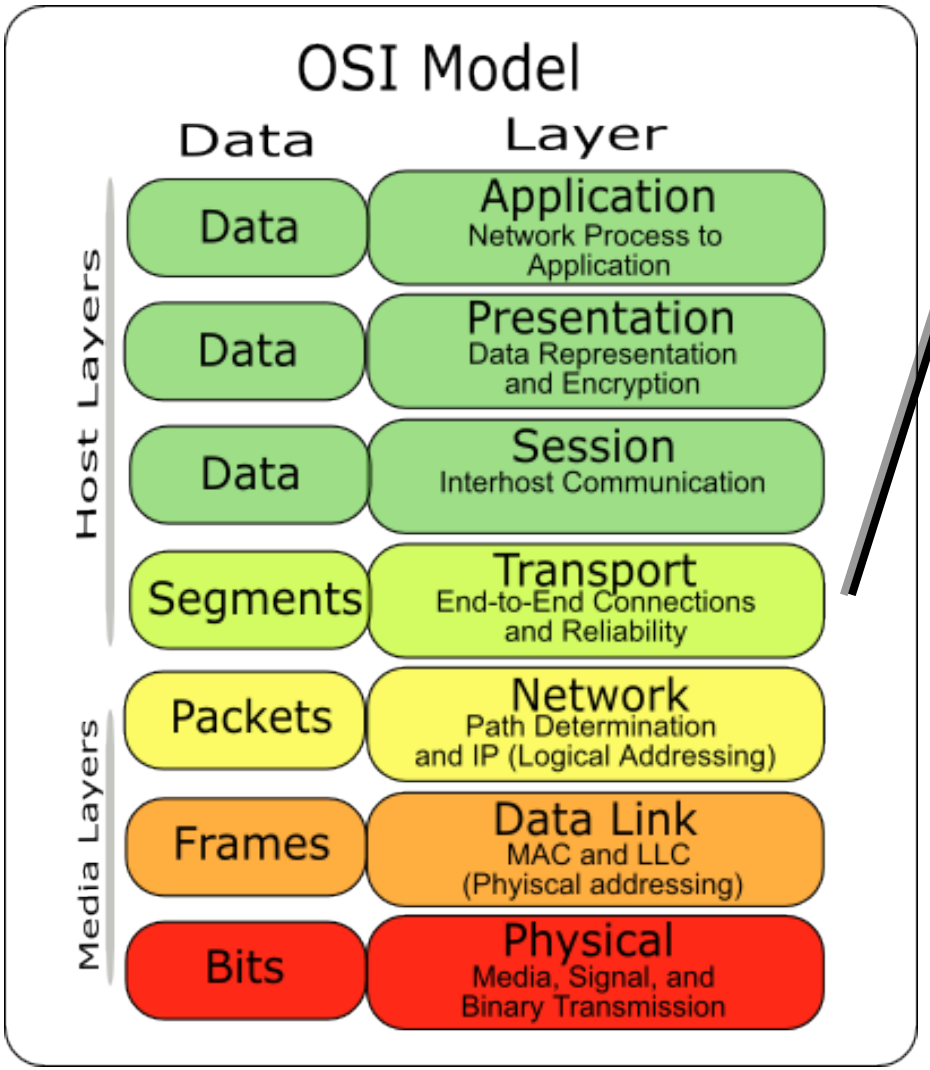

- **Network Layer:** Establishes the connection between two end nodes and provides path selection. This is where routing take place

- Functions are:

1. *Connectionless communication*: for example IP protocol. Connection-oriented protocols are at higher layers
2. *Host addressing*: every host must have a unique address (IP address)
3. *Message forwarding between different networks*

- **Devices**: **Router, gateway**
- **Protocols: IP, ICMP, ARP**

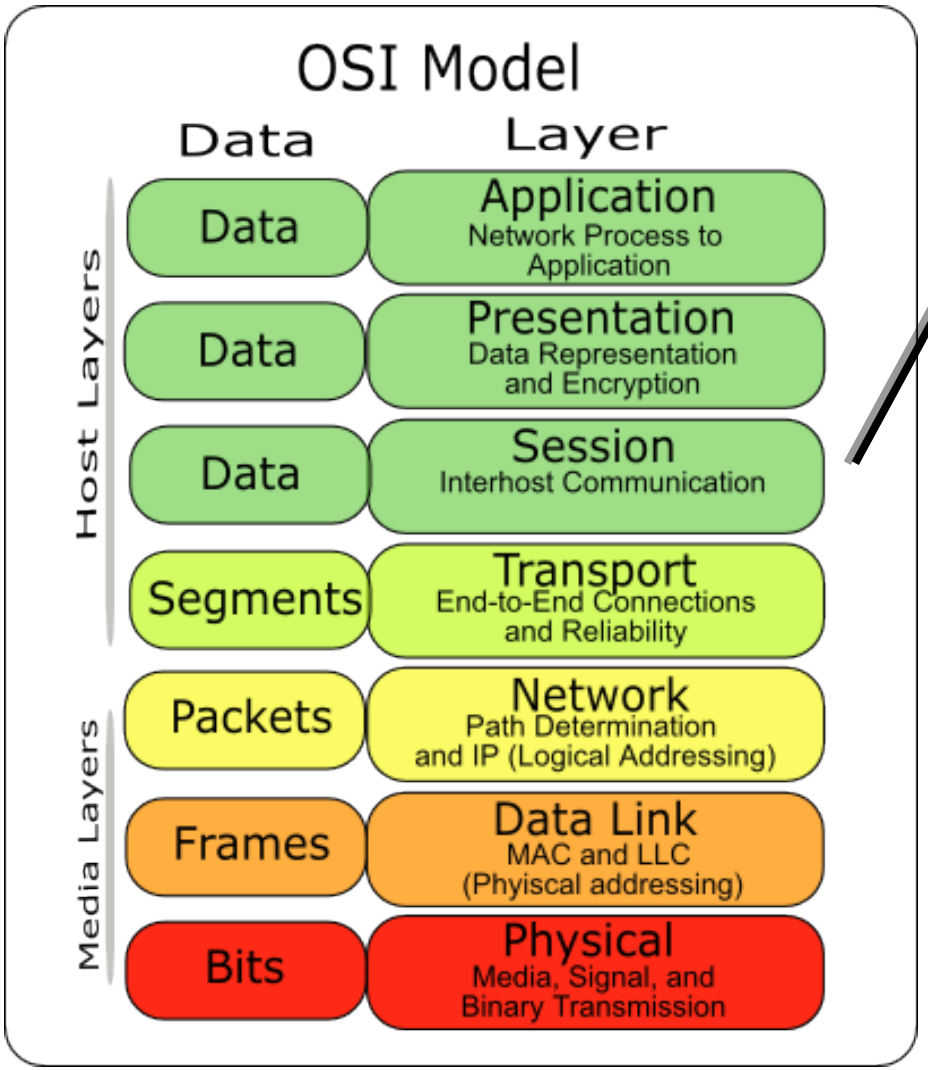
# Transport Layer




**Transport Layer:** organizes data into a data stream.

- This layer reassemble data from upper-layer applications and unite it onto the same data stream.
- The Transport layer can use positive acknowledgement and retransmission to ensure reliable delivery.
- **Protocols: TCP, UDP**

# Session Layer




**Session Layer:** is responsible for:

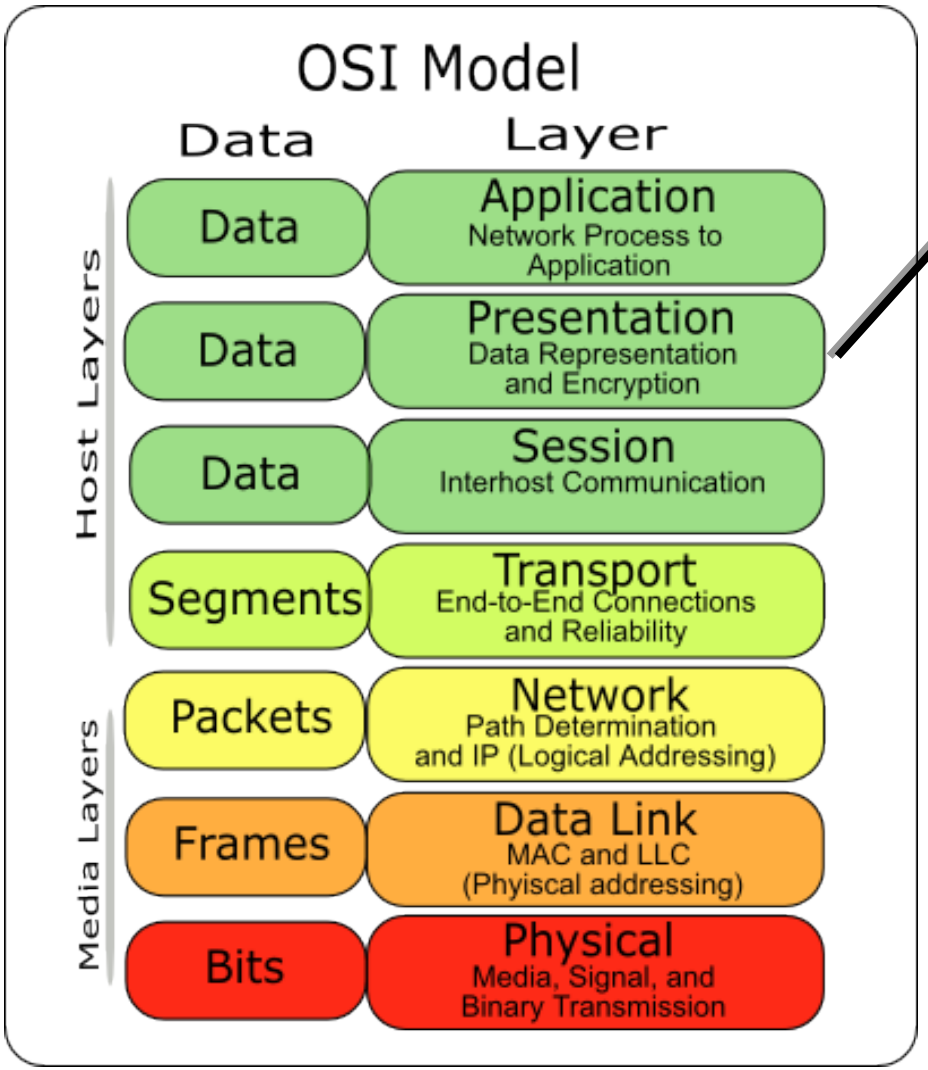
Setting up

Managing

Tearing down

sessions between Presentation layer entities.

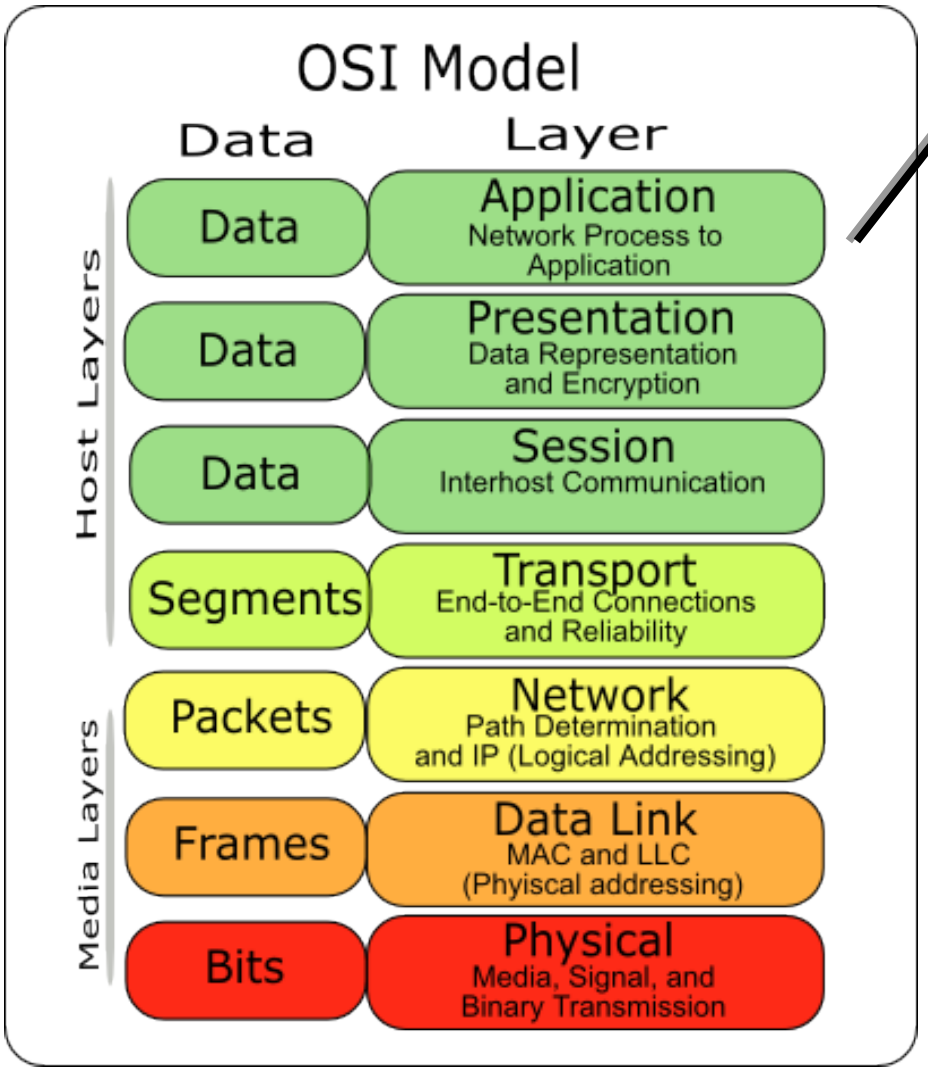
# Presentation Layer




**Presentation Layer:** presents data to the Application layer and is responsible for **data translation** and **code formatting**.

- This layer is essentially a translator and provides coding and conversion functions.
- A successful data-transfer technique is to adapt the data into a standard format before transmission.

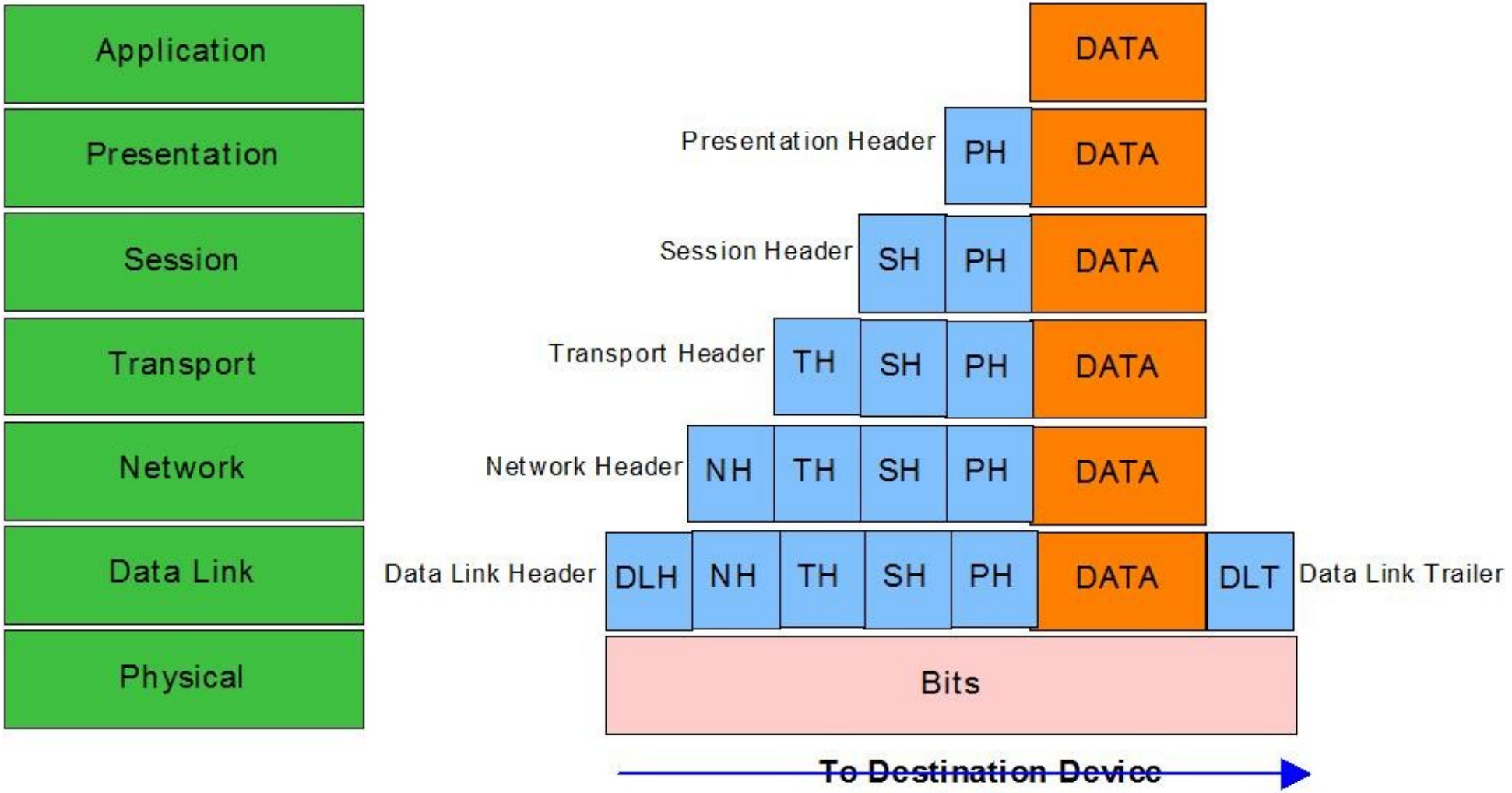
# Application Layer



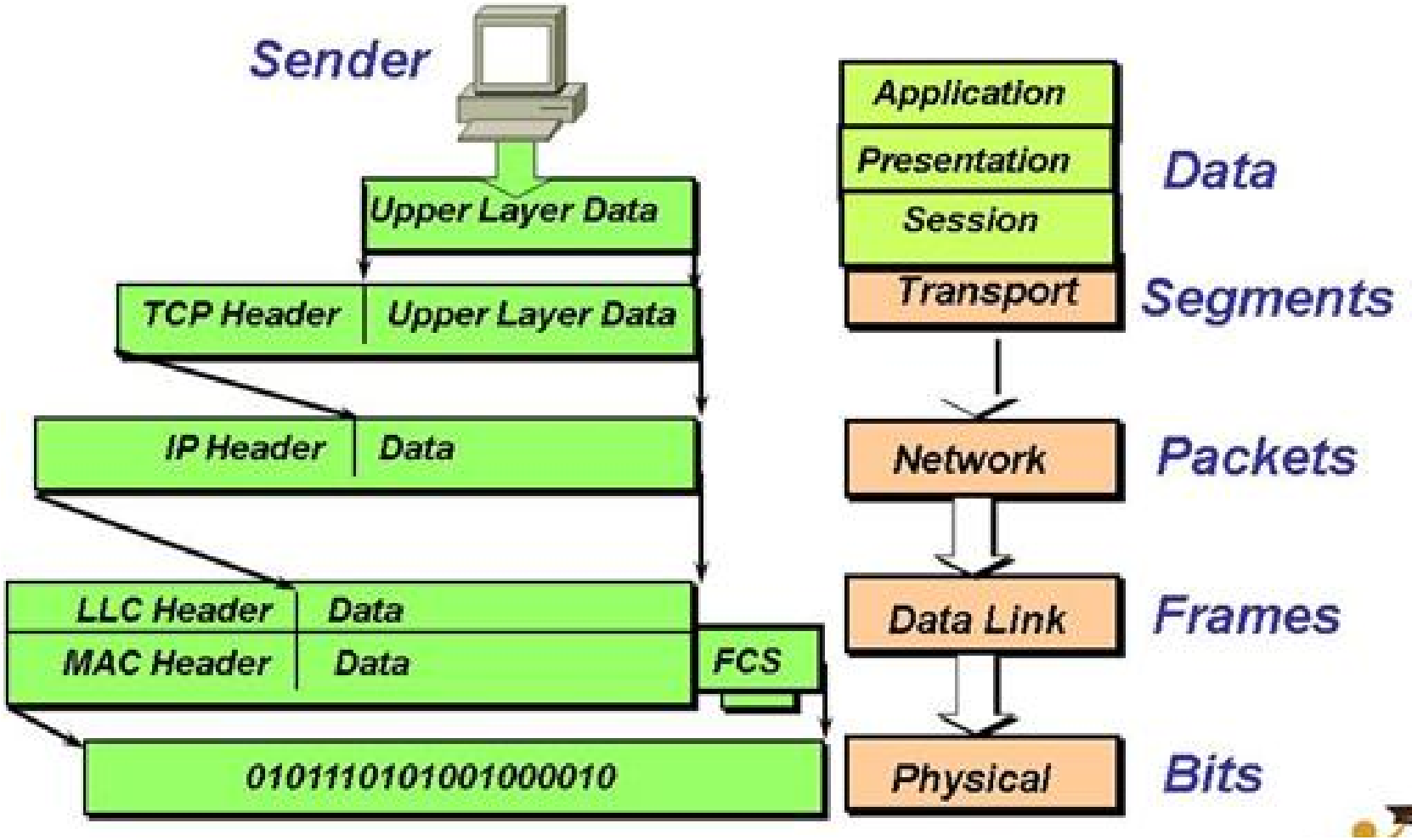

**Application Layer:** marks the spot where users actually communicate to the computer.

- This layer determine if sufficient resources for the communication exist.
- A vast array of protocols combine at the DoD.
  model's Process/Application layer to integrate the various activities and duties spanning the focus of the OSI's corresponding top three layers (Application, Presentation, and Session).

# Data encapsulation



# Data encapsulation



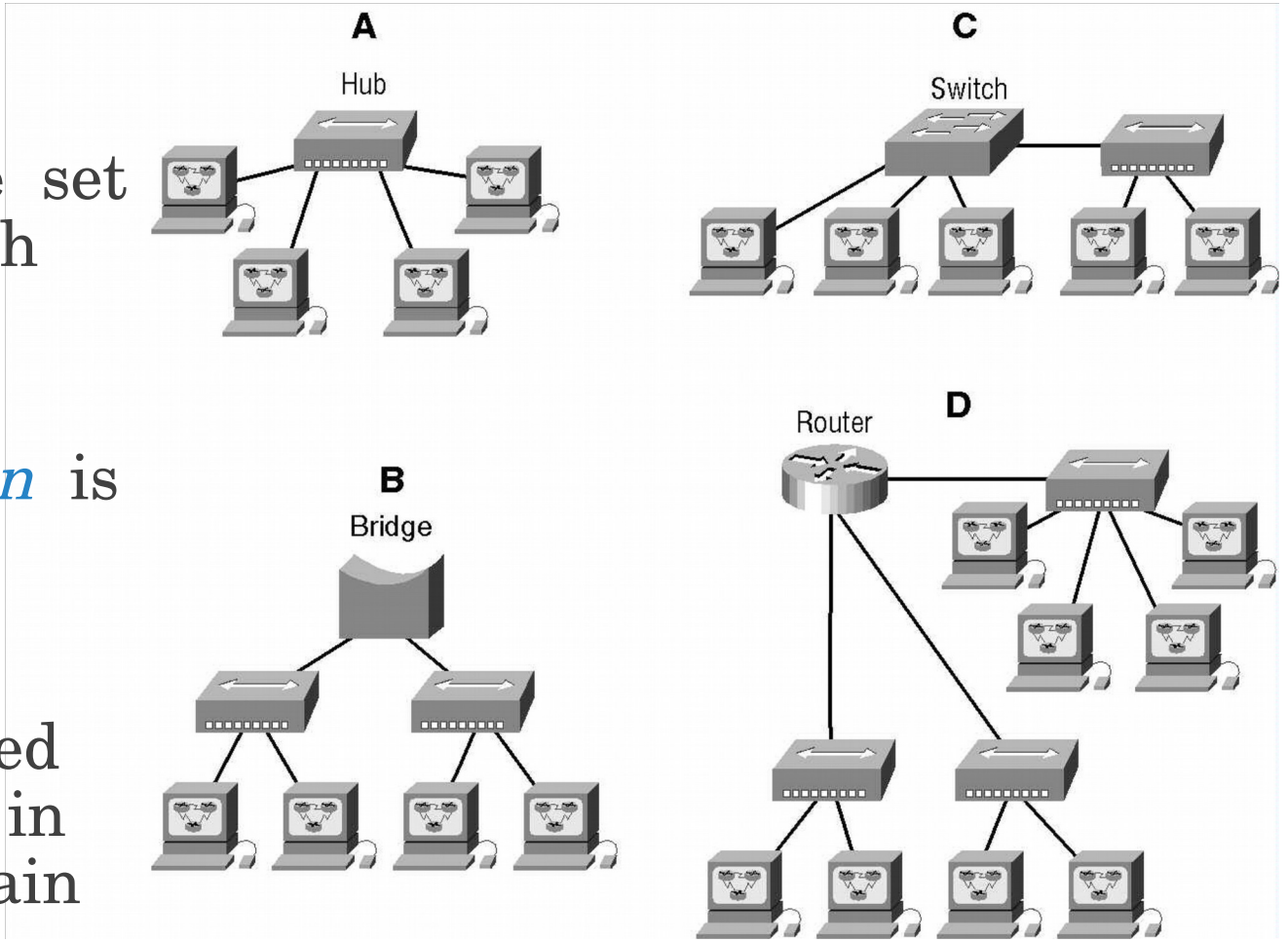
# Hub, switch and router

| HUB | SWITCH | ROUTER |
|---|---|---|
| Layer 1 | Layer 2 | Layer 3 |
| | Connencts different devices to form a network | Connects two or more different networks |
| Broadcasts to all connected devices | Transmits data to only devices able to listen that message | |
| Critical BW wastage | Minimal BW wastage | |
| Half-duplex device | Full-Duplex device | |
| | Principle of MAC addresses | Principle of IP addresses |
| | | HW makes use of routing algorithms to compute the best possible path for routing data packets across different networks |

# Collision vs Broadcast domain

The term *collision domain* defines the set of devices for which their ***frames*** could collide

A *broadcast domain* is a set of NICs for which a *broadcast frame* sent by one NIC will be received by all other NICs in the broadcast domain



# Collision vs Broadcast domain

- **HUB**: **Continues** Collision domain, **Continues** Broadcast domains

Ethernet Hub

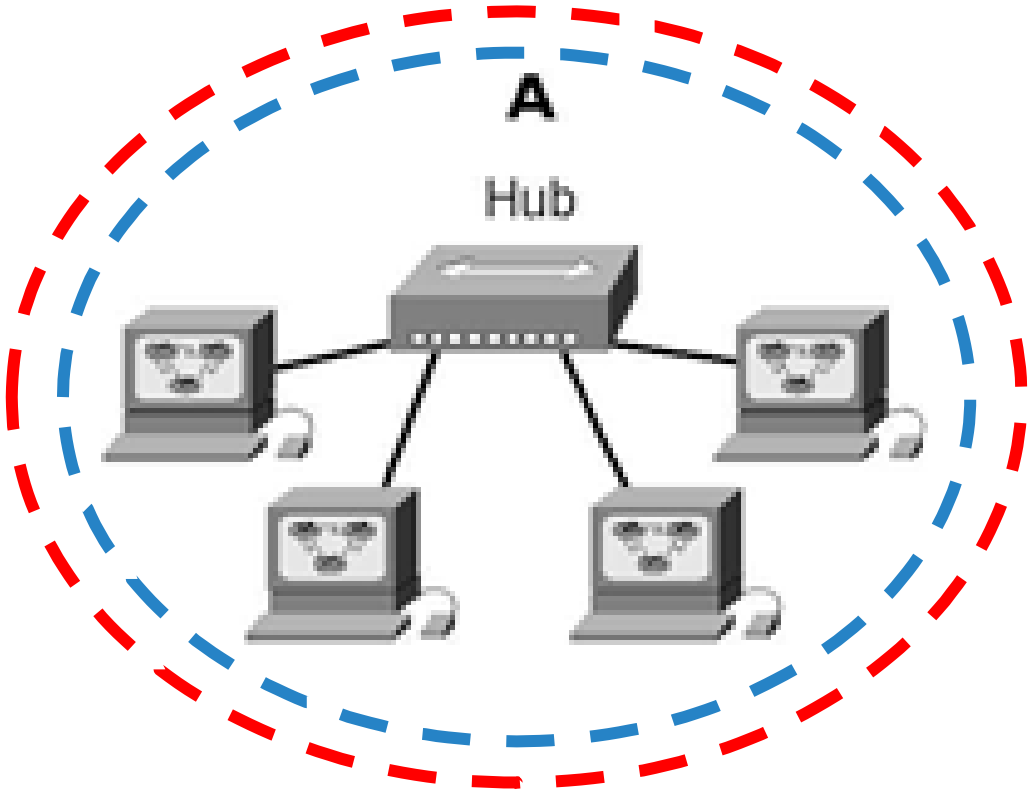
- **SWITCH**: **End** Collision domain, **Continues** Broadcast domains

Ethernet Switch

- **ROUTER**: **End** Collision domain, **End** Broadcast domains

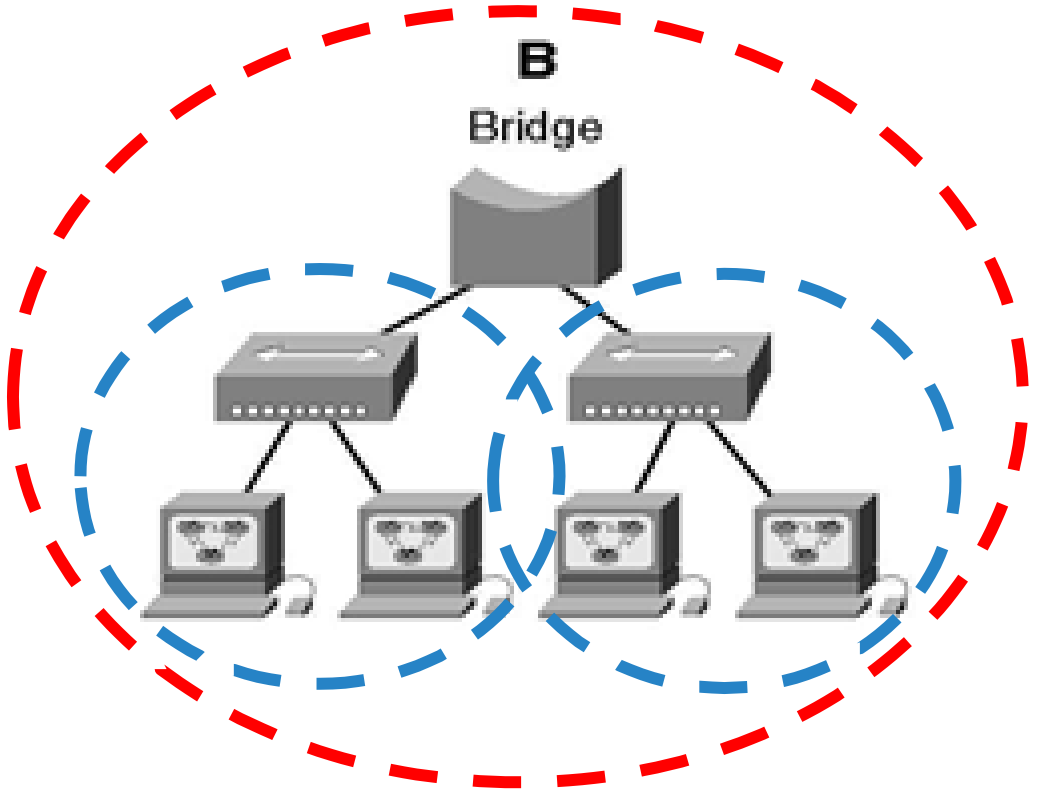
Router

# Collision vs Broadcast domain




- 1 ***collision domain***
- 1 ***broadcast domain***
- A collision domain is also a broadcast domain, but it is not the opposite.
- In a star topology with an hub, two domains always coincide

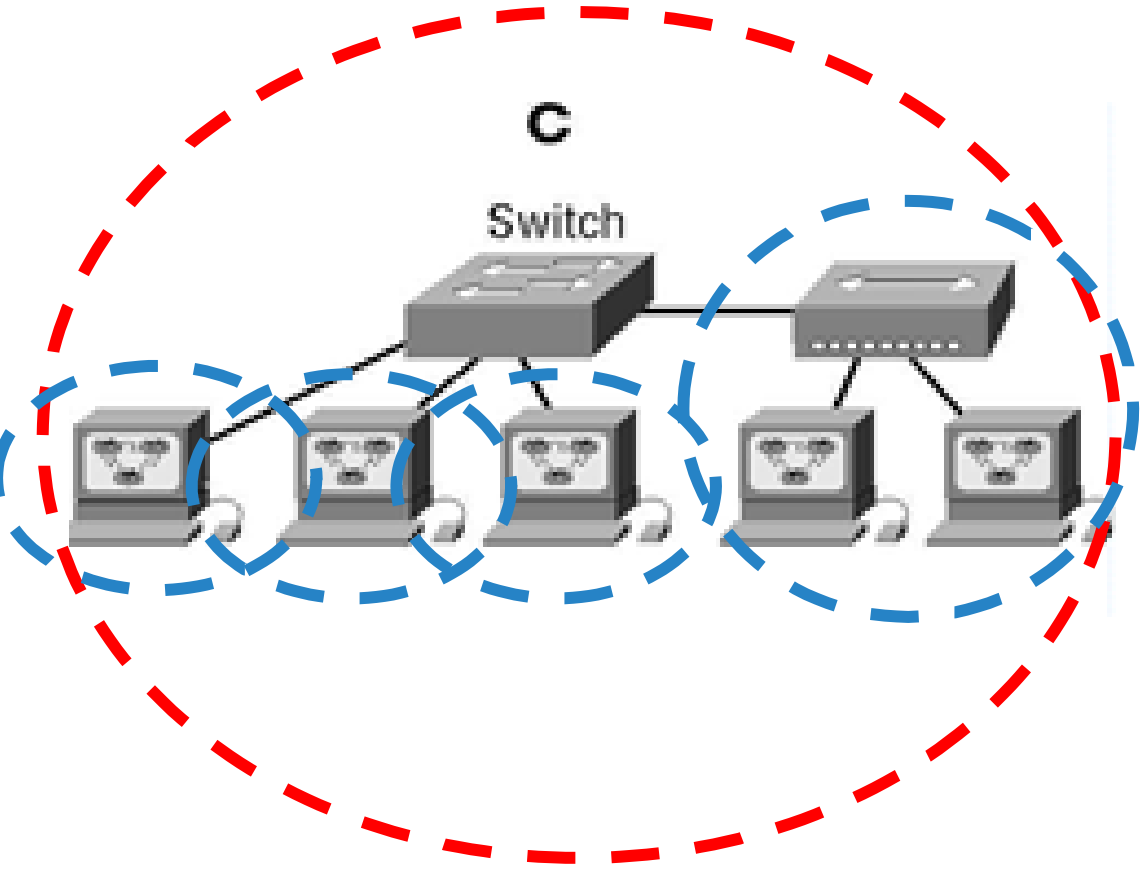
# Collision vs Broadcast domain




- 2 ***collision domains***
- 1 ***broadcast domain***

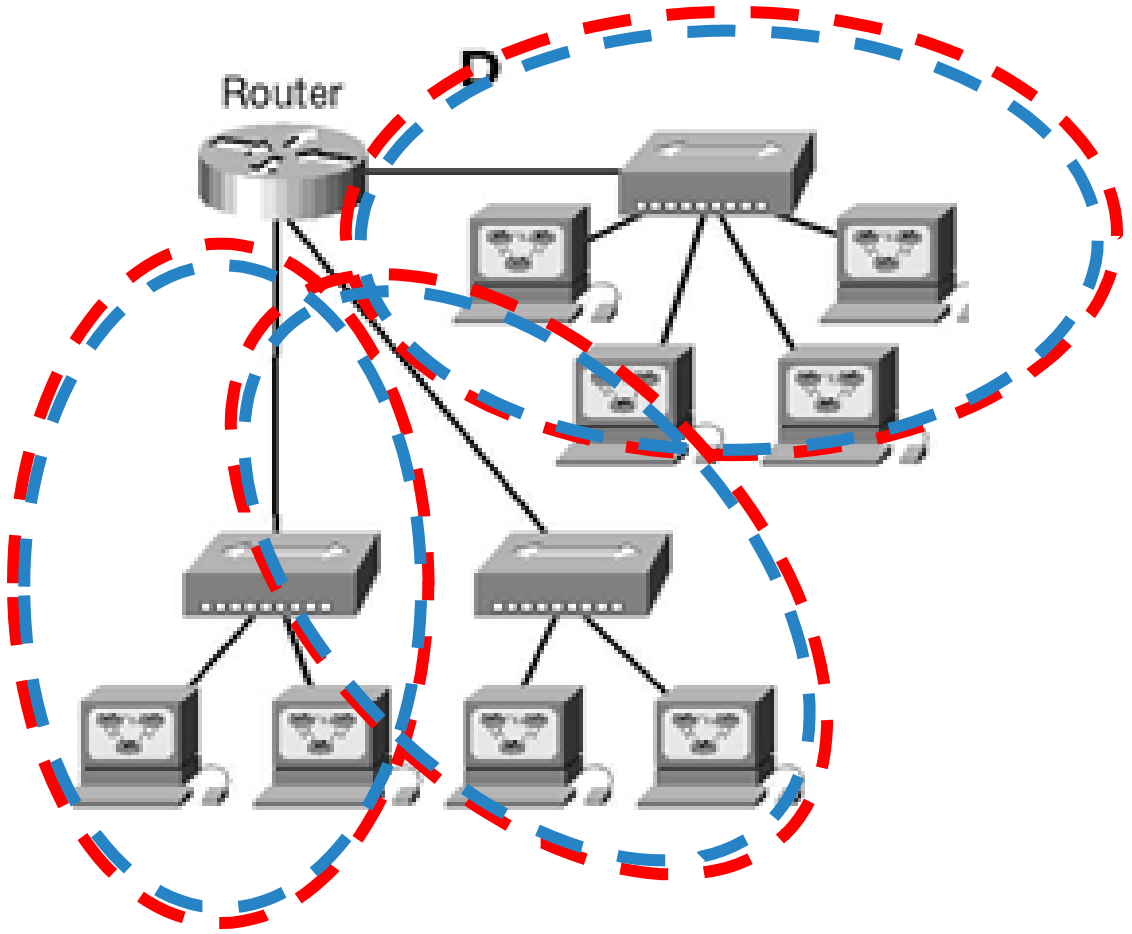
NB: 1. Bridge has a smaller # of ports than Switch
2. Bridge connects two different topology, while Switch connects a single topology

# Collision vs Broadcast domain




- 4 ***collision domains***
- 1 ***broadcast domain***

# Collision vs Broadcast domain




- 3 ***collision domains***
- 3 ***broadcast domains***


# Collision vs Broadcast domain


## HOW MANY BROADCAST DOMAINS?

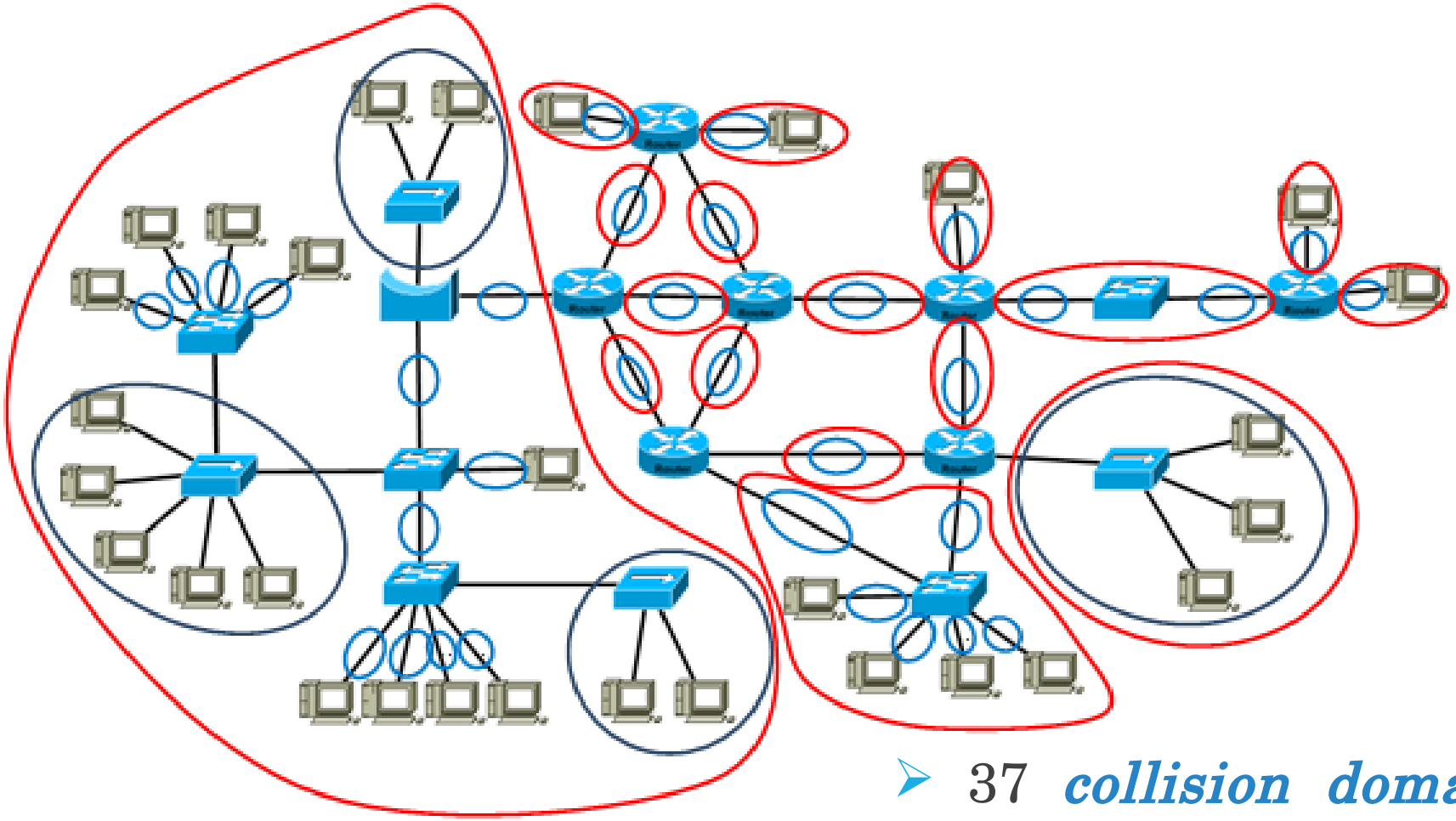
# of Broadcast domains = # of Network = # links attached to routers

## HOW MANY COLLISION DOMAINS?

# of Collision domains = # of links attached to switches but ALSO to routers

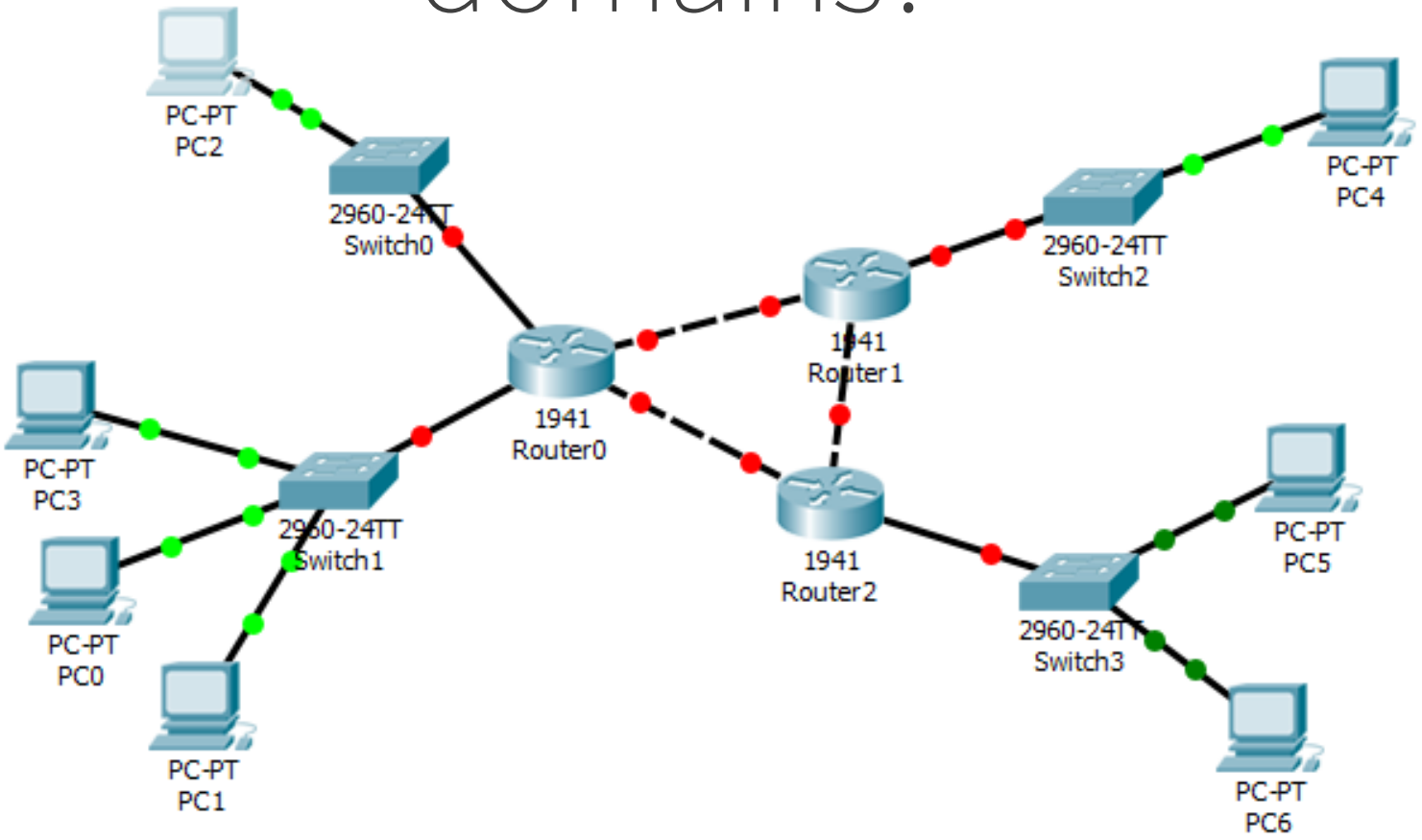
**A link that connects two routers is both a collision domain and a broadcast domain**

# Collision vs Broadcast domain

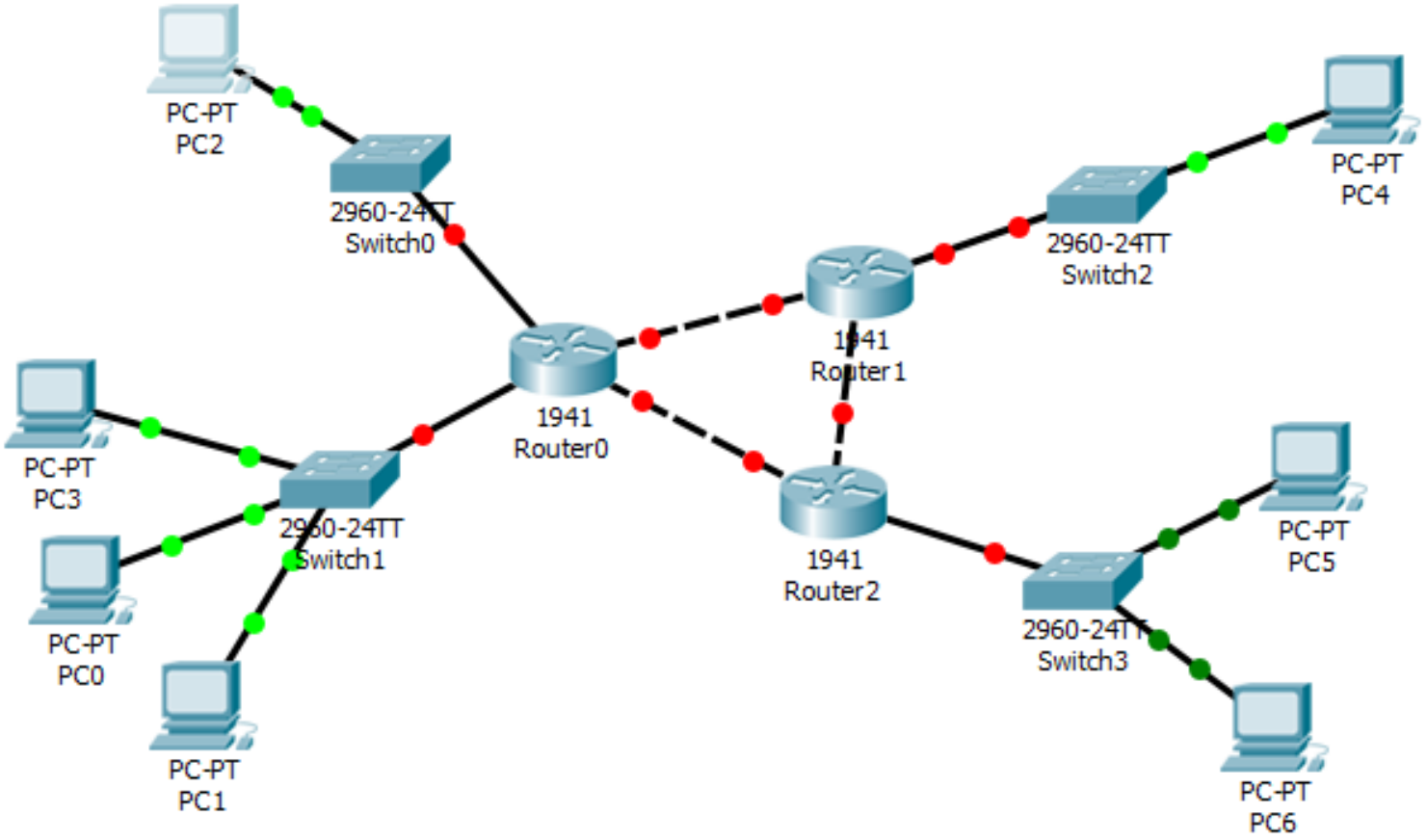


- 37 ***collision domains***
- 17 ***broadcast domains***

# How many collision/broadcast domains?

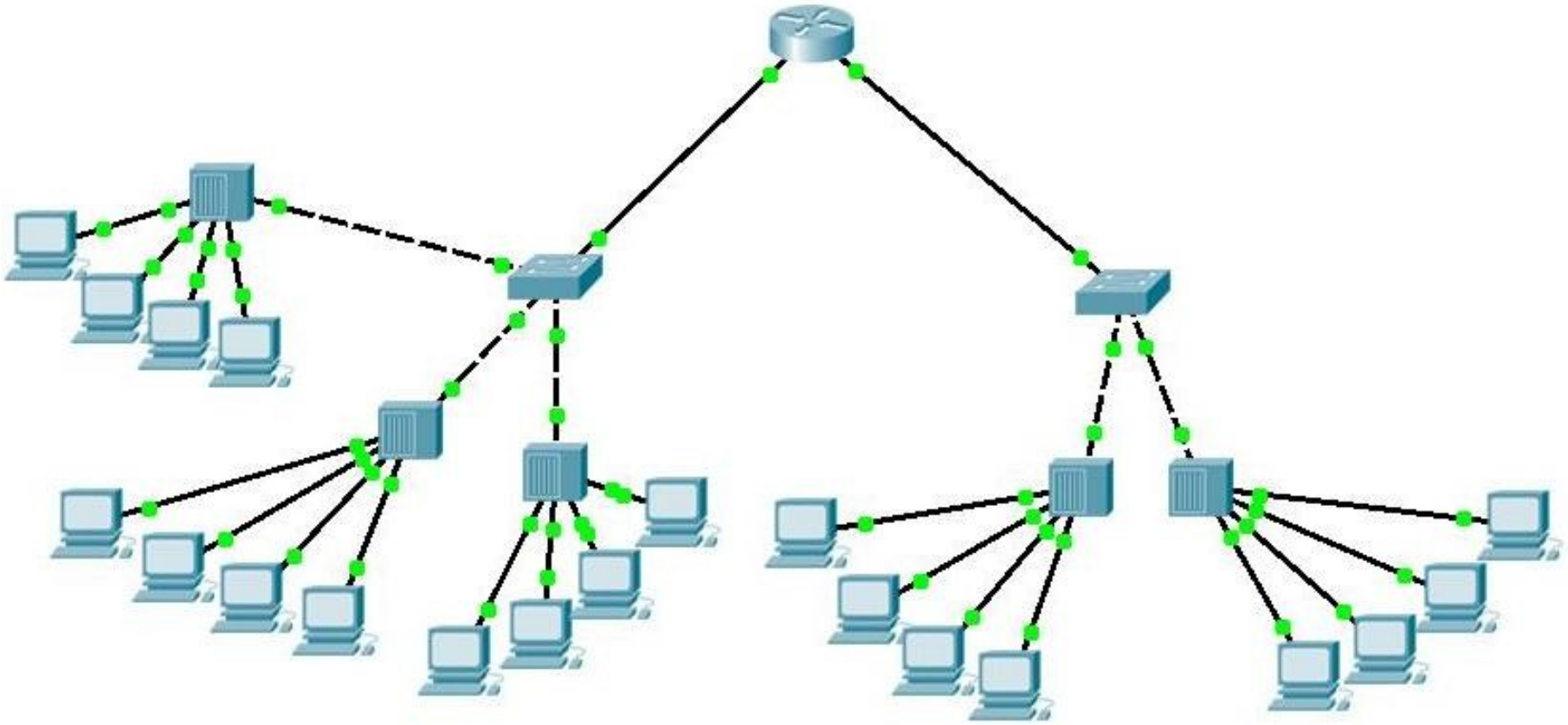


# How many collision/broadcast domains?

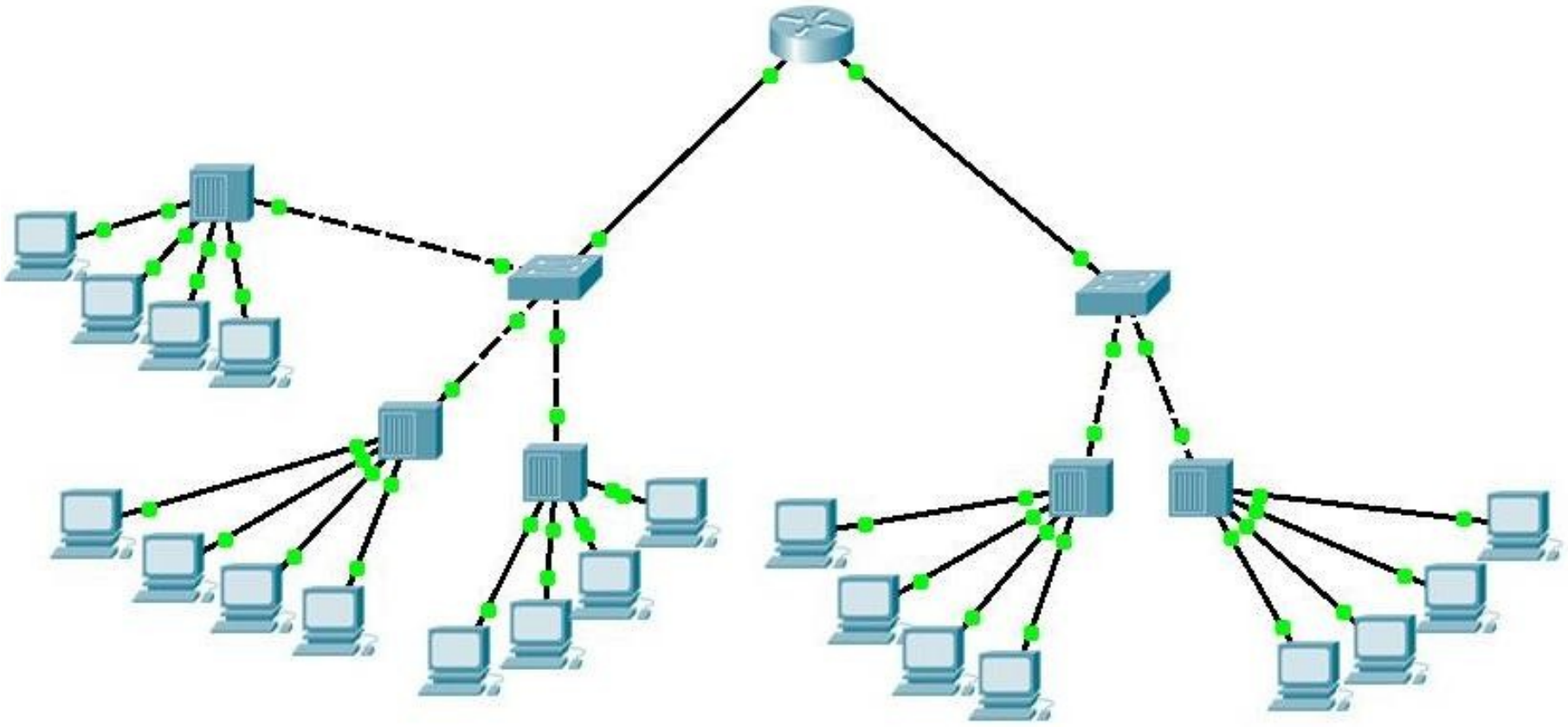



- 14 ***collision domains***
- 7 ***broadcast domains***

# How many collision/broadcast domains?

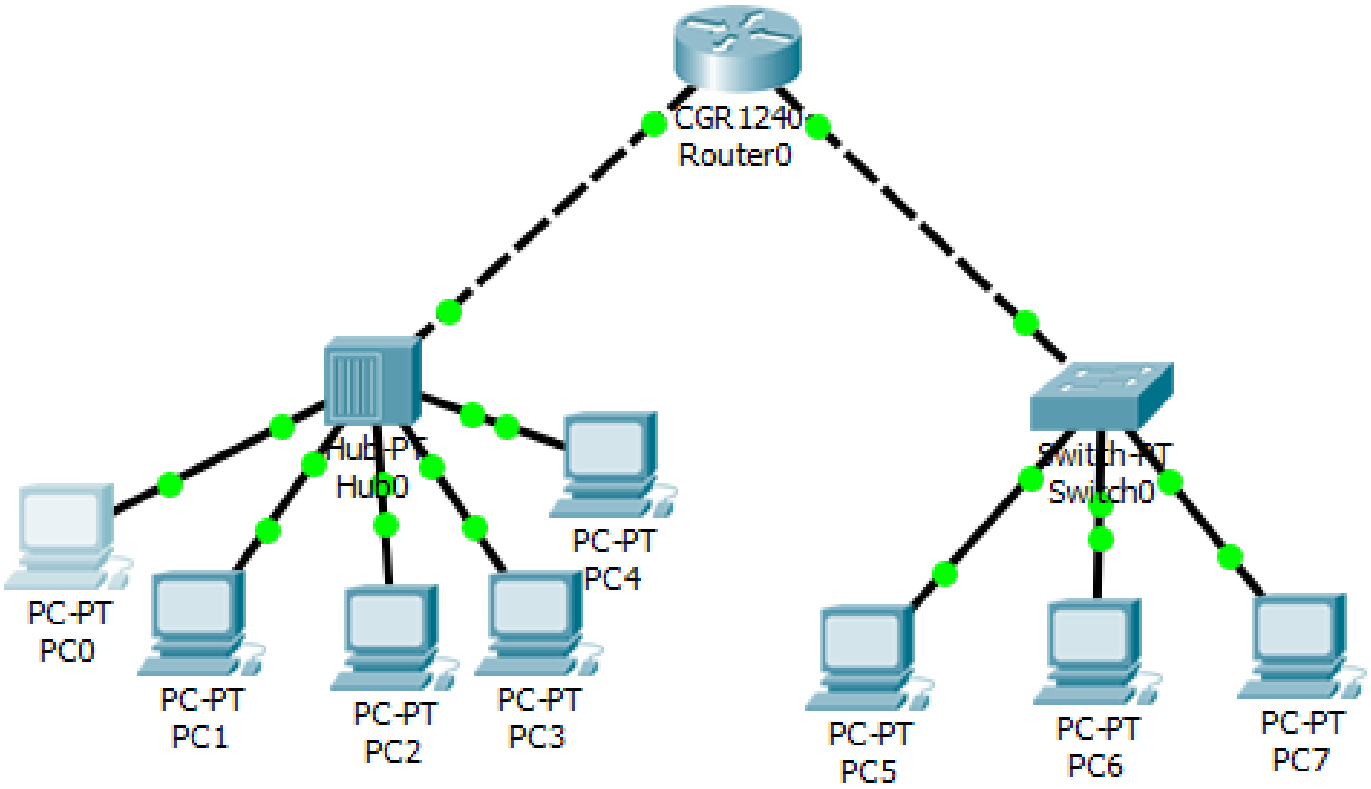


# How many collision/broadcast domains?

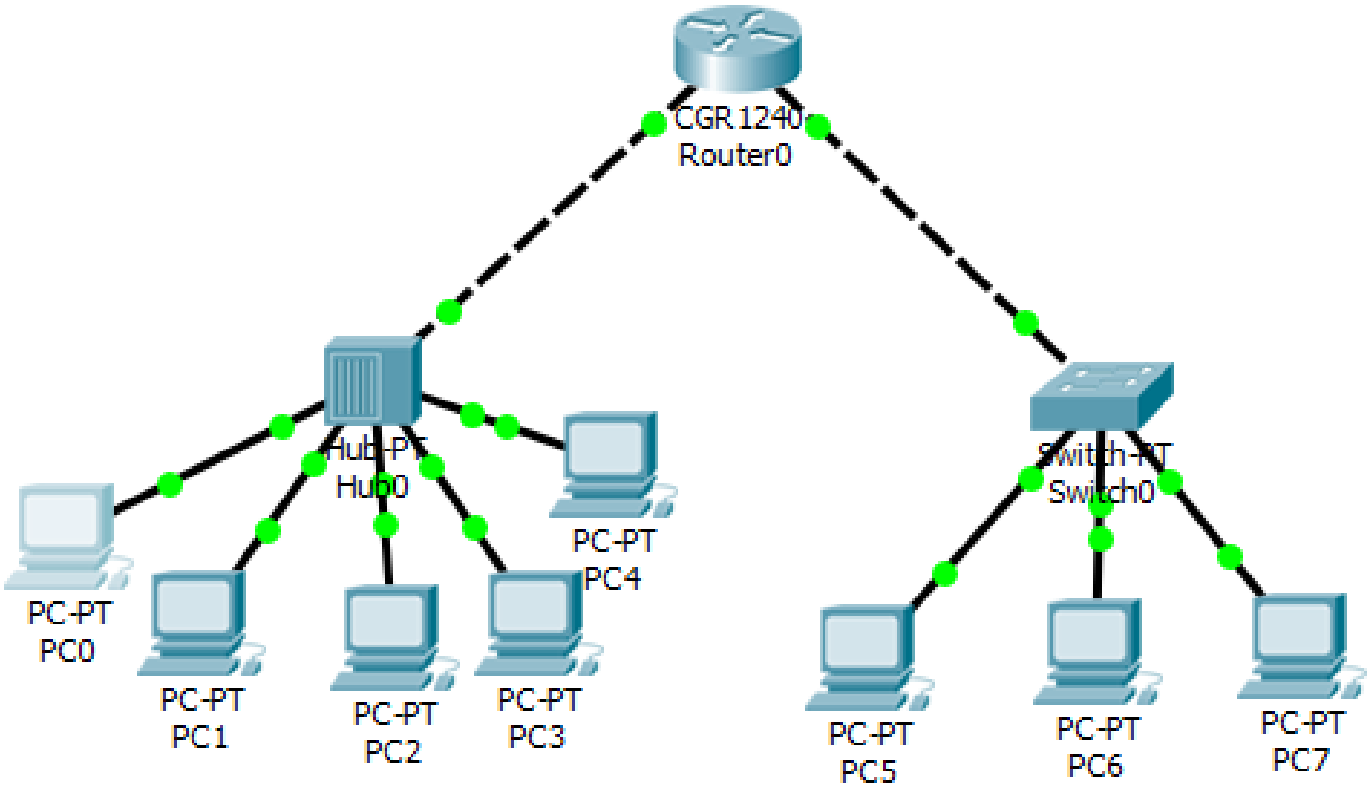


- 7 ***collision domains***
- 2 ***broadcast domains***

# How many collision/broadcast domains?



# How many collision/broadcast domains?




- 5 ***collision domains***
- 2 ***broadcast domains***



# Transport Layer Protocols: TCP and UDP


# Transport Layer


- Transport-layer (with its protocols) provides for **logical communication** between application processes running on different hosts
- There are two distinct transport-layer protocols available to the application layer: TCP and UDP
- **TCP** (Transmission Control Protocol) provides a reliable, connection-oriented service to the invoking application.
- **UDP** (User Datagram Protocol) provides an unreliable, connectionless service to the invoking application.
- Both protocols use the **segments** as transport layer packet


# UDP – User Data Protocol

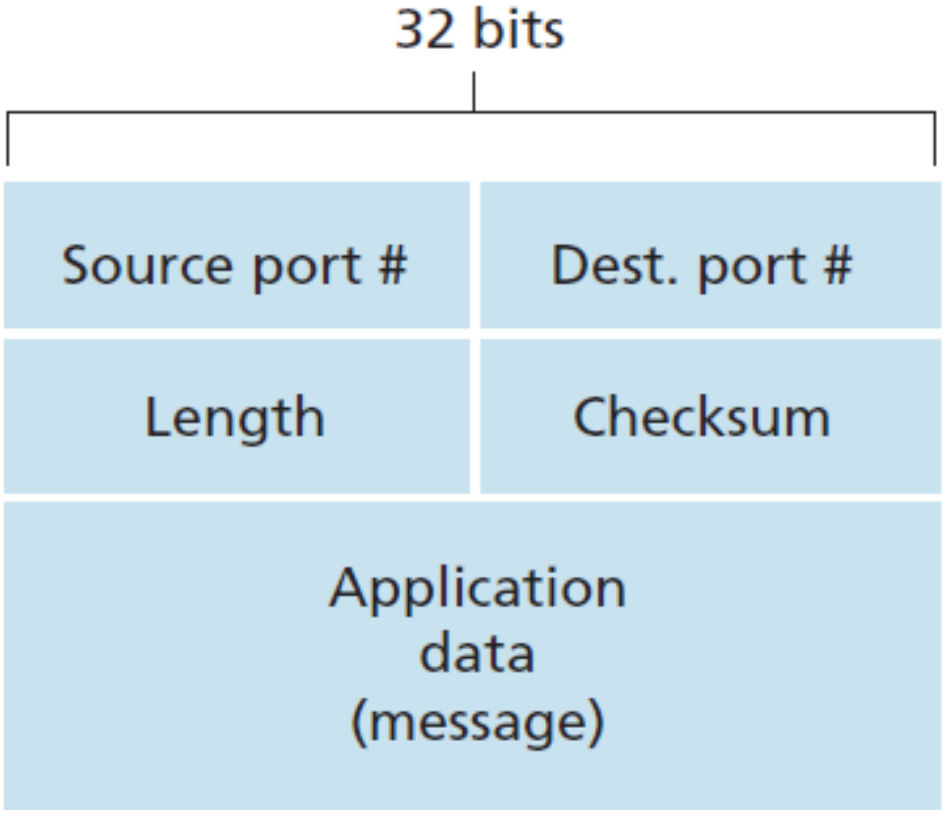

- UDP does the minimum that a transport protocol needs to do
- It had a multiplexing/demultiplexing function, some light error checking, **but it adds nothing to IP**.
- If the application developer chooses UDP, then the application is almost directly talking with IP.
- With UDP there is no handshaking between sending and receiving transport-layer entities before sending a segment. For this reason, UDP is said to be ***connectionless***.


# UDP – How it works?


1. UDP takes messages from the application process, attaches **source and destination port number fields** for the multiplexing/demultiplexing service, adds two other small fields (**checksum** and **Length**), and passes the resulting segment to the **network layer**.
2. The network layer **encapsulates** the transport-layer segment into an IP datagram and then sends the IP packet to the receiving host.
3. If the IP packet arrives at the receiving host, UDP uses the **destination port number** to deliver the segment's data to the correct **application process**.

# UDP Segment Structure

- Each port number is a 16-bit number, ranging from 0 to 65535.
- The port numbers ranging from 0 to 1023 are called **well-known port numbers** and are restricted, reserved for use by well-known application protocols such as HTTP (80)
- The **port numbers** allow the destination host to pass the application data to the correct process running on the destination end system
- **Length**: specifies the number of bytes in the UDP segment
- **Checksum**: it provides for error detection



# TCP- Transmission Control Protocol

- **Connection-oriented**: if an application process wants to start sending data to another one, two processes must first make a "3-way-handshake" [SYN – SYN ACK - ACK]: once the connection is established data transfer can begin
- **Reliable transport**: TCP manages message acknowledgment and retransmissions in case of lost parts
- **Error detection:** errors are detected with checksum field, and if a packet is erroneous, it is not acknowledged by the receiver (Positive Acknowledgement with Retransmission (PAR))
- **Congestion control:** There is an algorithm blocking the sending of packets IN CASE OF LOSS EVENTS


# TCP Segment Structure

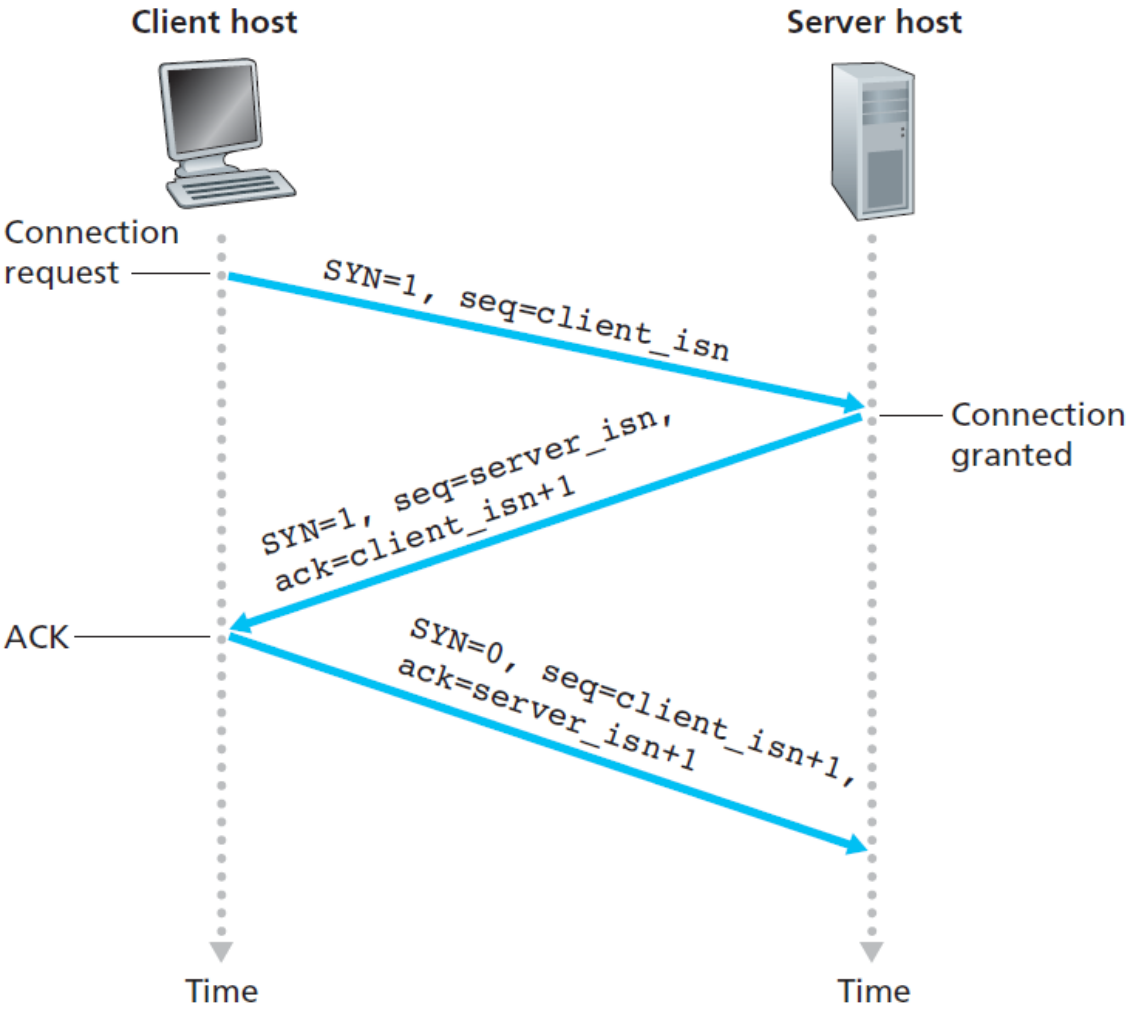

- As with UDP, the header includes **source and destination port numbers** (used for multiplexing/demultiplexing data from/to upper-layer applications) and a **checksum** field
- **Sequence number (32 bits)** and **Acknowledgment number(32 bits)** used by the TCP in order to implement a reliable data transfer
- **Receive windows (16 bits)** used for flow control (indicates the number of bytes that a receiver is able to receive
- **Header length field (4 bits)** specifies the lenght of the header
- **Flags(1 bit per flag):** SYN, FIN, RST used for connection setup and tear down, ACK used to recognize an ack


# TCP Connection Management

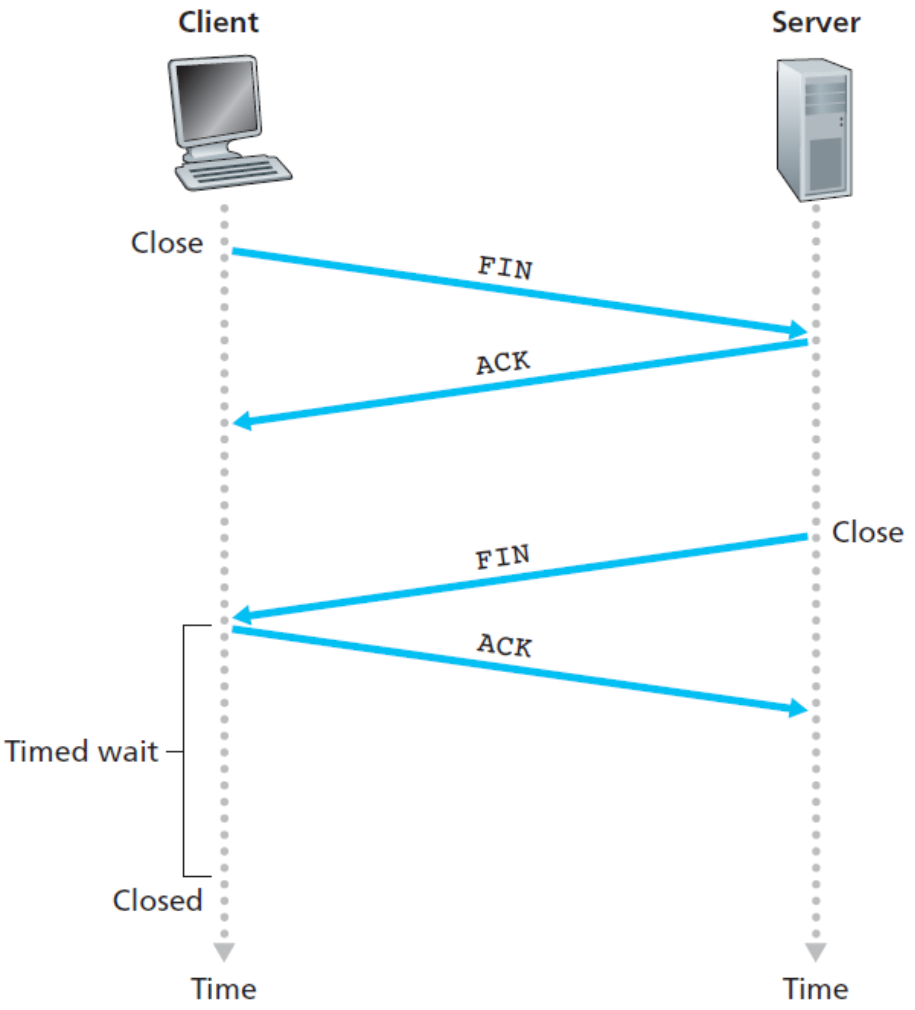

## How a TCP connection is established?

- Let's suppose a process running in one host (**client**) wants to initiate a connection with another process in another host (**server**)
- The client application process informs the client TCP that it wants to establish a connection to a process in the server
- The TCP client starts to establish a connection with the TCP server

# Three Way Handshake



# Closing a TCP connection




# TCP vs UDP


| FEATURES | TCP | UDP |
|---|---|---|
| Header size | 20-60 Bytes | 8 Bytes |
| Connection Oriented | YES | NO |
| Reliable Transport | YES | NO |
| Ordered Delivery | YES | NO |
| Checksum | YES | Optional |
| Flow Control | YES | NO |
| Congestion Control | YES | NO |

# Memes of course

You vs the guy she tells you not to worry about.

| UDP | TCP |
| --- | --- |
| Unreliable | Reliable |
| Connectionless | Connection-oriented |
| No windowing or retransmission | Segment retransmission and flow control through windowing |
| No sequencing | Segment sequencing |
| No acknowledgement | Acknowledge segments |

# Memes of course

**Kirk Bater**
@KirkBater

Follow

This image is a TCP/IP Joke. This tweet is a UDP joke. I don't care if you get it.

**Thread**
iamkirkbater and jkjustjoshing

**iamkirkbater** Aug 23rd, 2017 at 9:37 AM
in #www

Do you want to hear a joke about TCP/IP?

7

7 replies

**jkjustjoshing** 5 months ago
Yes, I'd like to hear a joke about TCP/IP

**iamkirkbater** 5 months ago
Are you ready to hear the joke about TCP/IP?

**jkjustjoshing** 5 months ago
I am ready to hear the joke about TCP/IP

**iamkirkbater** 5 months ago
Here is a joke about TCP/IP.

**iamkirkbater** 5 months ago
Did you receive the joke about TCP/IP?

**jkjustjoshing** 5 months ago
I have received the joke about TCP/IP.

**iamkirkbater** 5 months ago
Excellent. You have received the joke about TCP/IP. Goodbye.


# Causes and costs of Congestion


Why is important to avoid a network congestion?

1. **Large queuing delays** are reached as the packet arrival rate nears the link capacity.
2. The sender must perform retransmissions in order to compensate for dropped (lost) packets due to **buffer overflow**
3. **Unneeded retransmissions** by the sender may cause a router to use its link bandwidth to forward unneeded copies of a packet.
4. When a packet is dropped along a path, there is a **waste of transmission capacity**.


# Congestion control


We have to answer to following 3 questions:

1. How TCP sender limits the rate?
2. How TCP sender perceives that there is a congestion?
3. What algorithm should the sender use to change its send rate?


# How TCP sender limits the rate?


- Each side of a TCP connection consists of a receive buffer, a send buffer, and several variables
- The TCP congestion-control mechanism operating at the sender **keeps track** of an additional variable, the **congestion window** (cwnd)
- The **cwnd** is characterized by a certain number of MSSs (Maximum Segment Size) that the sender sends each RTT
- Sender sends with rate **CWND/RTT**
- If there is a congestion, CWND (and the rate) decreases


# How TCP sender perceives that there is a congestion?


**Loss event**: occurrence of either a timeout or the receipt of three duplicate ACKs from the receiver

- What happens in case of network congestion?
  1. **Router buffers can overflows,**
  2. Datagrams arriving to buffers are **dropped**.
  3. There will be no response (ACK) from the receiver

**Following events imply congestion:**

1. Lost segment (timeout occurrence)
2. Multiple ACK


# What algorithm should the sender use to change its send rate?


**TCP congestion-control algorithm** having three major components:

1. **Slow Start**
2. **Congestion Avoidance**
3. **Fast Recovery**


# Slow Start


- When a TCP connection begins, the value of **cwnd** is initialized to

  cwnd = 1 MSS

- It results in an initial sending rate of a MSS/RTT.
- cwnd value **increases by 1** MSS **every** time a transmitted **segment** is first **acknowledged**
- TCP sender would like to find the amount of available bandwidth quickly!!!
- After 1 RTT we will have cwnd=2
- After 2 RTT we will have cwnd=4
- After 3 RTT we will have cwnd=8…

**When should this exponential growth end?**


# Congestion Avoidance


- Rather than doubling the value of cwnd every RTT, TCP adopts a more conservative approach and increases the value of cwnd by just a single MSS every RTT
- A common approach is to increase cwnd by MSS bytes (MSS/cwnd) whenever a new acknowledgment arrives.
- If we have a cwnd = 10, for each arriving ACK, the cwnd will increase by 1/10 of MSS. When all ACK will arrive, cwnd is increased by 1 MSS
- There is a linear increase, unlike the exponential one of slow start

**When should congestion avoidance's linear increase end?**

When a timeout or a triple ACK occours…


# Fast Recovery


The behaviour is different depending on the entity of the loss event:

1. Timeout
2. Triple ACK received

Both events are sympton of congestion, but the first one is more serious

3. In case of timeout the action is drastic:
   - ssthresh = cwnd/2
   - cwnd = 1
   - Slow Start mode
4. In case of triple ACK:
   - Cwnd=cwnd/2 + 3MSS
   - Ssthresh = cwnd/2
   - Fast Recovery mode

# Example of congestion control