OSI model

↓

open systems interconnect

reference model → some common design for devices to connect together and exchange data.

↓

this was standardized in 1984 by researchers

↓

modern models borrow ideas from this model → TCP/IP model widely used today

host → any device

address → address of a computer in the network.

↓

network address

↳ this is a more specific name

models have protocols

↓

commonly agreed → rules

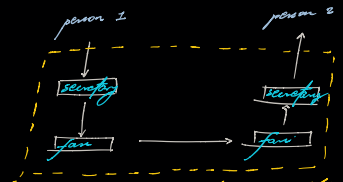
↓

helps heterogeneous devices to communicate

Textbook

↳ # classic Tanenbaum

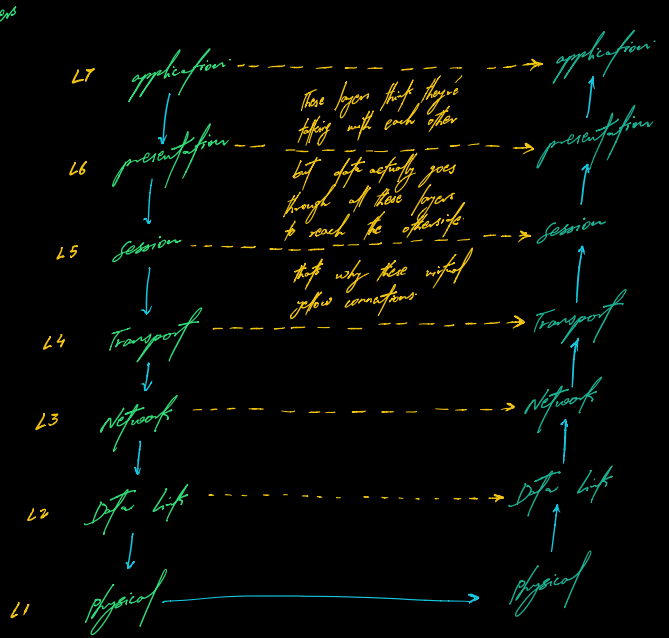
models are composed of layers of abstraction



↳ for the two people they don't need to know what happens inside the box.

secretaries don't need to know what the fax machines do internally

7 layers



Application layer

Browser → HTTP, HTTPS, FTP

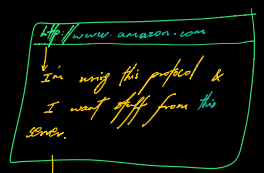
Outlook → SMTP

Skype → Skype protocol → proprietary

Remote Desktop → Remote Desktop Protocol

or TELNET for unix systems

These are some common protocols of layer 7 (application layer)
This is how they talk with each other.



normally we also mention what webpage we want but in this case we are just asking for a default webpage "index.html"

Presentation Layer (L6)

limited to English chars → more chars in more languages

Translation of ASCII/unicode to binary

Data Compression → we don't give a fuck if it's big/small, less is better.

Encryption
↳ this is what HTTPS uses
↳ SSL Secure Socket Layers

Most modern browsers manage top 3 layers: application, presentation and session layer

Session Layer (L5)

Establish, manage & terminate connections

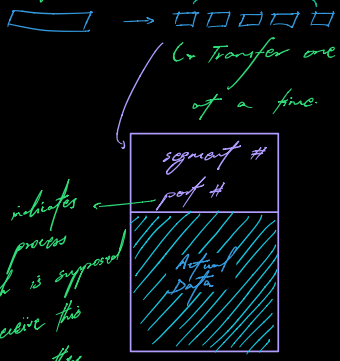
Authorization & Authentication.

verify permissions → whether to let a particular host access something.

verify identity

L4 # Transport Layer (TCP / UDP)

Segmentation



→ size of chunk depends on TCP/UDP etc

→ also give numbers

(→ Transfer one chunk at a time.

so transport layer on the other side can put the chunks back together.

indicates the process which is supposed to receive this on the other side

flow control → devices can handle different throughputs depending → so we need this layer to handle that

Error Control → does checksum and asks for packets again if checksum doesn't add up.

TCP	vs	UDP
slow		fast
lossless		lossy → were okay with losing some bits for more speed.
ACK for each packet		no feedback
webpage, email		video calls, streaming
connection oriented		connection less

Network Layer (L3)

segment / packet from before + src + dest address

logical addressing → IP addr → 0.0.0.0 to 255.255.255.255 } → 4 byte address

Routing maybe → not going to write much; this thing will turn up later on in much more detail. → the video didn't much here.

domain name → DNS → IP of destination

(DNS) ↓ it is just a hashtable

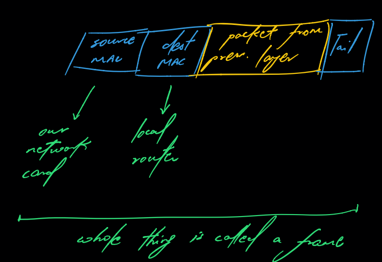
in Wireshark we can capture random packets in our network and we can check:

- source port
- dest port
- checksum
- individual bytes
- sequence no
- ack no
- etc

Data Link Layer (L2)

wireshark first asks where the packets should be captured from → wifi, RJ45, Thunderbolt, etc.

physical address → MAC address



media access control (↑ mediums)
↳ prevents collisions with devices using the same

error detection → detect errors before data leaves router

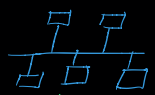
Physical Layer (L1)

translate from bits to signals & vice versa

Now we will study each layer in detail

Data Link Layer

① Access to media → fiber optic
→ wifi
→ Twisted cable

② media access control → 
controlling who has access to media

③ Error Detection & correction

④ OPTIONAL flow control & Acknowledgement → manage differences in throughput capacity

DLL → logical link subLayer → flow control, Ack & error notification
media Access Control subLayer (MAC address)
↳ controls who gets access to media
↳ is digital addressing

Hardware used
network Interface Card
wifi card
Layer 2 switch:
↳ Layer 3 switches are called routers but modern routers act as both L2 & L3 switch

src → switch → dst
↳ routes & also amplifies signal if necessary

Delays
① Transmission Delay → Time taken to place bits on line
② Propagation Delay → Time taken for bits to actually travel on the line.

Transmission Delay

measured ← B → bandwidth in bps → rate of placing in $10, 10^2, 10^3$, etc not 1024, etc.

L → # of bits in packet bits on line

↓ measured in 2 powers → so 1Mb means 1024

1 Kb = 1000

$T_t = \frac{L}{B}$ seconds

Propagation Delay

D → length of wire

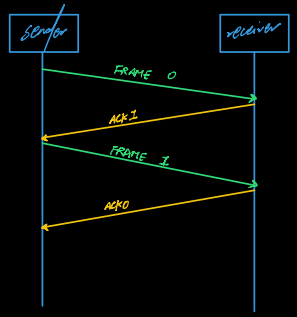
v → velocity → normally maximum or assume 3×10^8 m/s

$T_p = \frac{D}{v}$ seconds

Total delay = $T_t + T_p$

we will measure the efficiencies of many of the protocols using these 2 metrics primarily.

Stop and Wait ARQ (Automatic Repeat Request)



$L = T_t + T_p$

+ $T_{processing}$ → very small → can be ignored

+ $T_t + T_p$

$\therefore L = T_t + 2T_p$ seconds

efficiency η = useful time / total time

$= \frac{T_t}{T_t + 2T_p} = \frac{1}{1 + 2a}$

where $a = \frac{T_p}{T_t}$

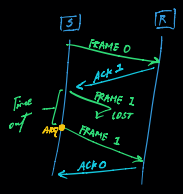
Just like η we have throughput or bandwidth utilization / effective bandwidth.

$= \eta \cdot B$ → bandwidth of medium.

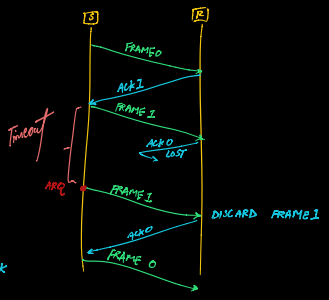
$\frac{L \cdot B}{T_t + 2T_p}$ ← $\frac{\text{bits transferred}}{\text{time taken}}$ ← $\frac{L \text{ bits}}{T_t + 2T_p \text{ seconds}}$

when frame is lost, sender waits for some time and sends frame again.

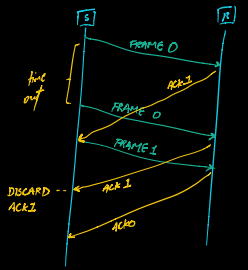
this extra time = T_o = timeout to the process



LOST ACK

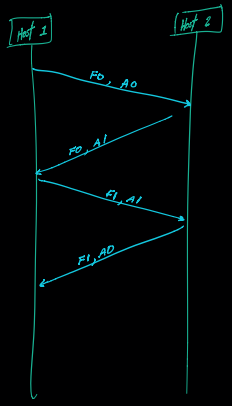


STOP AND WAIT delayed ACK



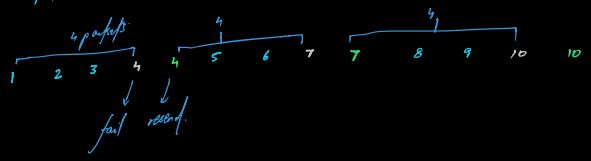
STOP AND WAIT ARQ (piggybacking)

Here we're talking about communications both ways.



The ACK is so small it makes sense to tag it along with other frames.

In stop & wait ARQ we send 10 packets of which every 4th packet is lost. How many transmissions are needed?



if 1000 packets are to be sent using stop & wait ARQ, error probability = 80%

Transmissions needed = ?

$$n + np + (np)p + (np^2)\cdot p \ldots$$

p packets lost so send again

$$= n + np + np^2 + np^3 + \ldots$$

→ geometric progression.

$$= n\left(\frac{1}{1-p}\right)$$

$$\therefore = 1000\left(\frac{1}{1-0.8}\right)$$

$$= 5000$$

Capacity of a channel/medium
↳ max no. of bits on the wire any time

= Bandwidth * Propagation Delay

= (BW * T_p) bits

$T_p = \frac{l}{v}$

cap $\propto$ BW

cap $\propto T_p \propto l$
↓
length of wire

example

BW = 2.1 mbps

T_p = 1 sec

Size of frames = 10^3 bits

$\therefore$ capacity = $2.1 \times 10^6 \times 1 = 2.1 \times 10^6$ bits

$= 2.1 \times 10^3$ frames

$= 2100$ frames

for full duplex = 4200 frames

η = no. of packets at any time / capacity

↳ this thing = 1

we can prove this is equal to $\frac{1}{1+2a}$

this is only for stop & wait ARQ

if we use stop & wait ARQ,

That the equation is a general purpose eq.

$\therefore \eta = \frac{1}{4200}$

in S & wait ARQ

we have to wait for $T_t + 2T_p$ time

before we get any ACK

if we use this $T_t + 2T_p$ time for

stuffing more packets in the channel

we can stuff $\frac{T_t + 2T_p}{T_t}$ → the time we have left.

T_t → time to stuff one packet

$\therefore$ $1 + 2\left(\frac{T_p}{T_t}\right)$ packets in the channel → capacity

$$2 = \frac{\#\text{ of packets at a time}}{\text{capacity}} = \frac{1}{1 + 2\left(\frac{T_p}{T_t}\right)}$$

$\therefore$ $$2 = \frac{1}{1 + 2a}$$

Stop & wait sucks

So we use sliding window

0 1 2 3 0 1 2 3 0 1 2 3

R only sends one ACK for every four packets.

0 1 2 3 0 1 2 3 0 1 2 3

R can send ACK anytime, doesn't have to wait for all packets from S

0 1 2 3 0 1 2 3 0 1 2 3