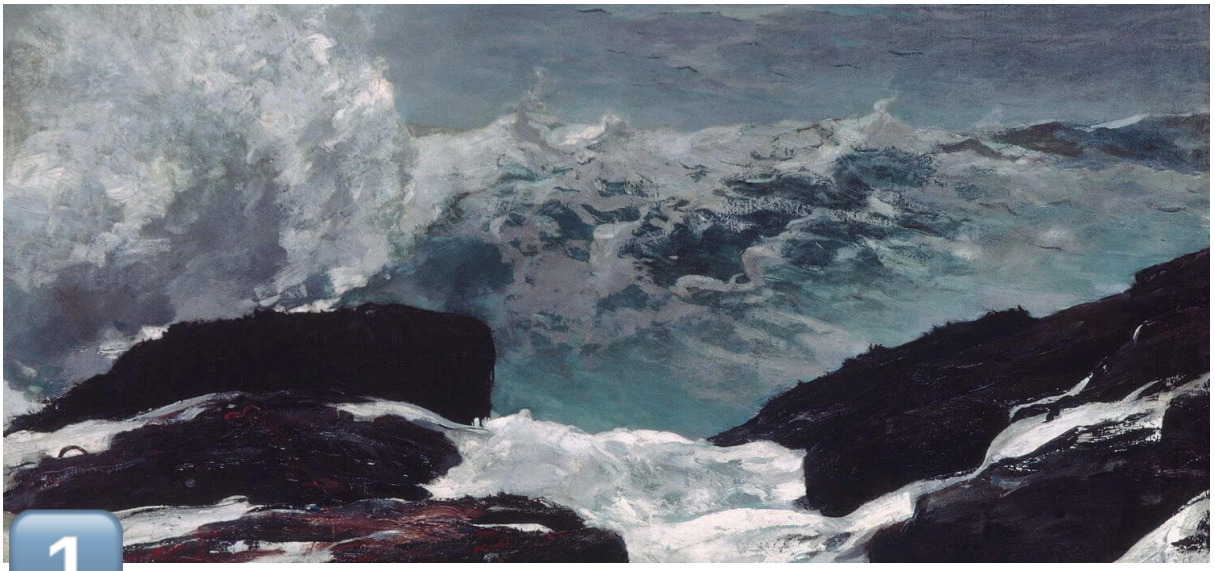
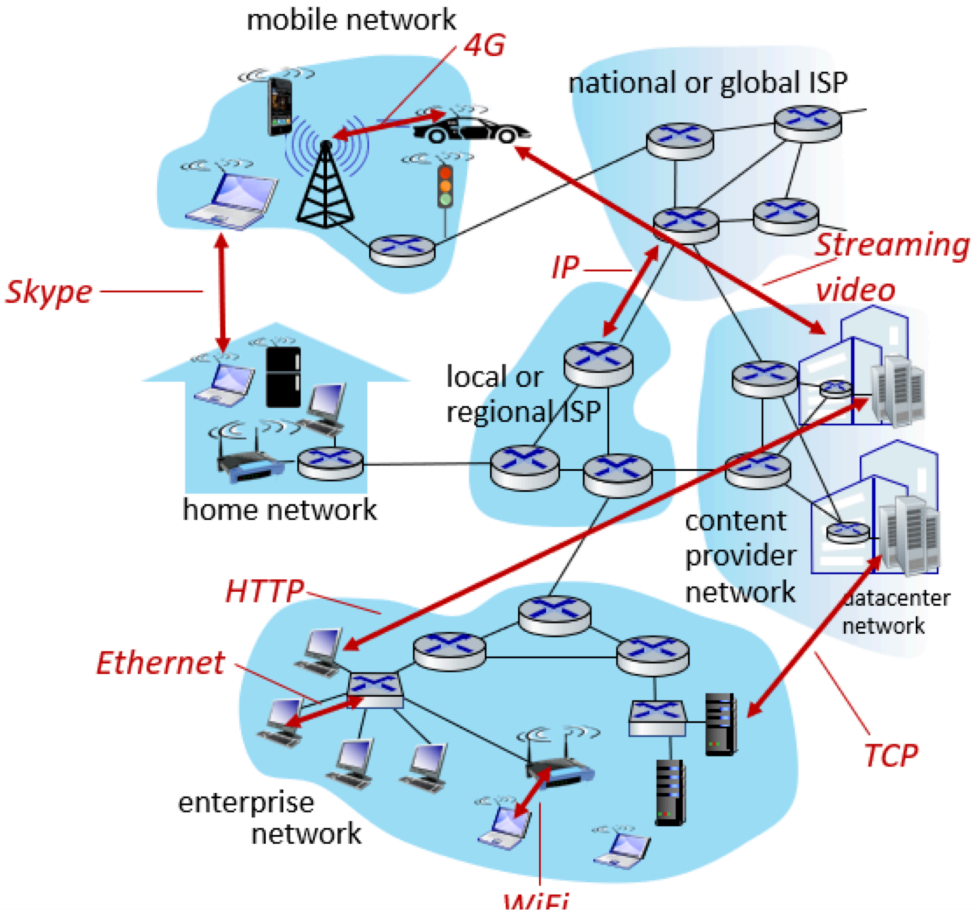


Unit 1 - Internet Architecture and Applications

- Internet: Introduction and terminology
  - Internet
    - Internet is a computer network that interconnects billions of computing devices throughout the world
    - It provides services to distributed applications (eg., email, web, streaming)
    - It is often referred as *network of networks*



▼ Computer Network

- A computer network is like a graph of end-systems (hosts) connected by communication links and packet switches

▼ Hosts

- Run applications that generate or receive data in the form of packets

▼ Route/Path

- A sequence of packet switches + communication links

▼ Administration

- Typically managed by one entity (eg., ISP, enterprise IT)

▼ Access to Internet

- Hosts connect via *internet service providers* (ISPs)

▼ History of the internet

- 1960s: Birth → ARPANET, packet switching.

- 1970s: TCP/IP, email.
- 1980s: Internet takes shape → DNS, ISPs.
- 1990s: World Wide Web, browsers.
- 2000s: Broadband, social media.
- Today: IoT, AI, cloud services.

▼ Some important abbreviations

- IANA: Internet Assigned Numbers Authority.
- ICANN: Internet Corporation for Assigned Names and Numbers.
- IETF: Internet Engineering Task Force (standards).
- IAB: Internet Architecture Board
- TCP: Transmission Control Protocol

▼ Internet Service Provider

- ISP (*Internet Service Provider*) is a business entity or company which provides internet access to end-systems in return for subscription fee

▼ PoPs

- The place where end-systems connect to ISPs are called *Point-of-Presence* (PoP)
- PoP contains routers, switches, MPLS, Communication links
  - eg: Airtel, Jio, BSNL, AT&T, Verizon

▼ PoPs of various ISPs maybe interconnected the following methods

▼ Multihoming

- When an ISP connects to two or more upstream ISPs instead of just one
- It is reliable because if one provider goes down, traffic can be routed through the other and balances load by splitting traffic to avoid congestion

▼ Peering

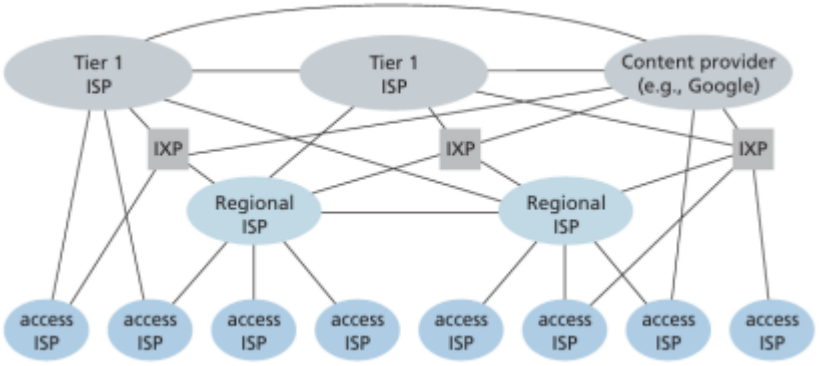
- When two ISPs (usually of same tier) agree to connect directly and exchange traffic without charging each other
- It saves money by avoiding any 3rd party ISP and improves performance (shorter path)

▼ Internet Exchange Point

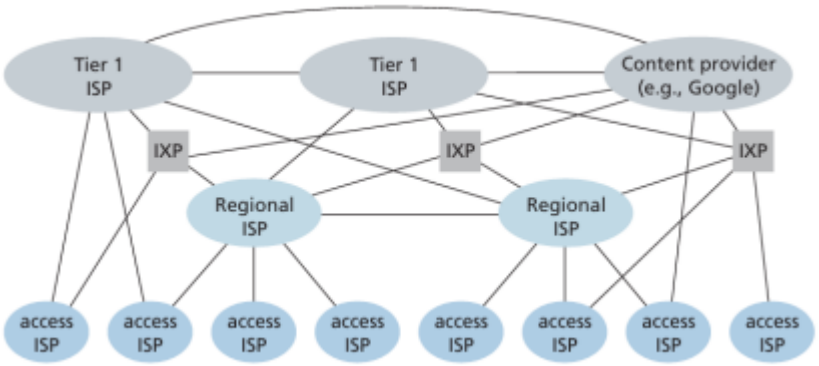
- A physical facility where multiple ISPs meet and exchange traffic usually located in data centers in big cities
- ISPs bring their routers there and interconnect with many other ISPs at once
- This reduces cost (no need for dozes of separate connections) and reduces latency (traffic stays local instead of going through international paths)

▼ ISP Hierarchy

- Tier 1 ISPs: Global backbone providers (AT&T, Sprint)
- Regional ISPs: Cover specific regions/countries (Airtel, BSNL, Vodafone)
- Access ISPs: Last-mile provides (Local broadband provider)



▼ Revenue Generation



- The flow of money generally goes upward in the hierarchy, while flow of traffic can go in any direction

▼ End Users → Access ISPs

- The customer pays to *Access ISPs* like Jio, Airtel, BSNL etc.,
- These ISPs provide the *last-mile connection* (fiber, DSL cable, mobile etc.,)
- Usually these are subscription based model which is a steady revenue source

▼ Access ISPs → Regional ISPs

- *Small/Local ISPs* don't have global connections so they pay *Regional ISPs* for upstream connectivity
- *Regional ISPs* aggregate traffic from many access providers

▼ Regional ISPs → Tier-1 ISPs

- *Regional ISPs* pay *Tier-1 ISPs* like AT&T, Tata Communications etc.,
- *Tier-1 ISPs* own massive international backbones and submarine cables
- *Tier-1 ISPs* don't pay anyone because they interconnect with each other in settlement-free arrangements

▼ Direct Deals with Content Providers

- Big content providers like Google, Netflix, Amazon, Facebook generate massive traffic
- Instead of paying middlemen, they often:

        - Place servers inside ISPs' data centers (They are called CDNs or Content Delivery Network)
        - Negotiate direct peering agreements with ISPs
    - ▼ Peering & IXPs
        - When 2 ISPs peer, they usually don't pay each other (free exchange of traffic)
        - but sometimes, if traffic is unequal, one side will have to pay
        - IXPs lower costs because instead of paying multiple providers, ISPs can exchange locally
- ▼ Internet & Distributed Applications
    - ▼ Distributed Applications
        - Distributed application is an application that runs on multiple computers (hosts) across the internet and works together
        - For example, you open YouTube in your browser which makes one part and then YouTube's servers are the other part thereby exchanging messages over the internet
        - So internet isn't an app by itself, but it provides the platform that apps use to talk to each other
    - ▼ Role of Protocols
        - The rules for communication that are followed by distributed applications to work are called protocols
        - Protocols define :
            - Format (syntax) : How data looks
              Example : HTTP request looks like `GET /index.html HTTP/1.1`
            - Meaning (semantics) : What each part means
              Example : `GET` means fetch this resource
            - Timing : When messages are sent and in what order.
              Example : Client must send request before server replies
        - Different apps may need different kinds of services like Reliability (TCP) for apps like web browsing, email, file transfer etc., Low delay (UDP) for apps like video calls, gaming etc.,

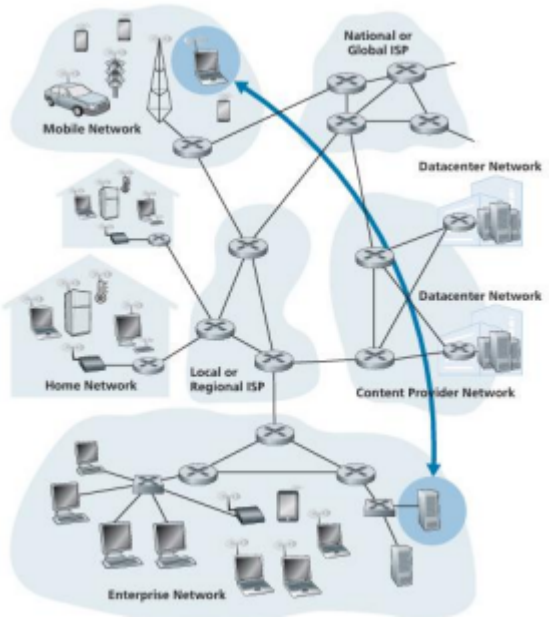
Throughput/Rate Guarantees for apps needing a steady bit rate (video streaming services etc.,)

▼ Socket Interface

- Applications don't directly talk to the network hardware, instead they use a socket - like a door which connects the app to the internet
- A socket is defined by *IP Address + Port Number*
  - For example, Visiting google → socket = `142.250.190.14:443` (IP + HTTPS port)
- App developer controls what goes through the socket and which transport protocol to use

▼ Network Edge

- The network edge is where end devices (hosts) connect to the internet
- These devices can be clients (laptops, phones, PCs, IoT devices) or servers (websites, cloud services, data centers)

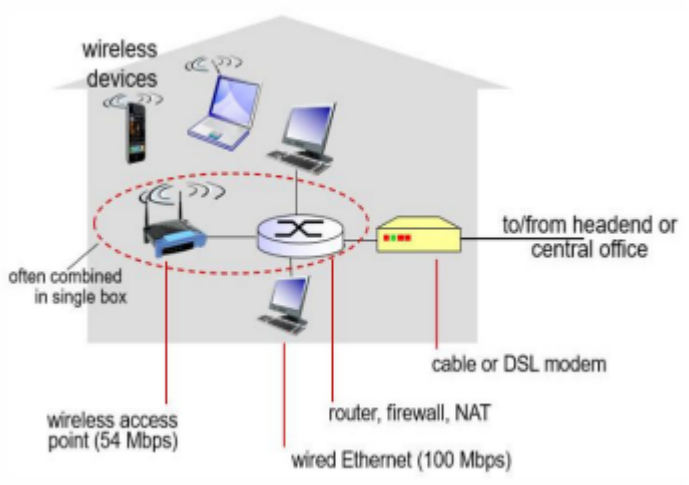


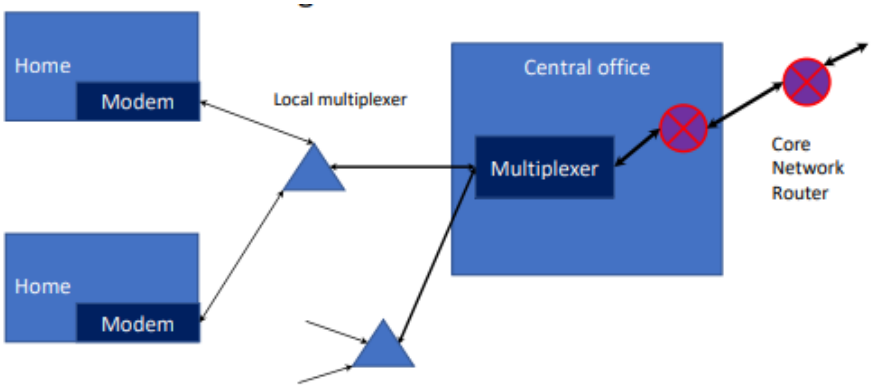
▼ Types of Access Networks

▼ Home Networks

- Usually connect via ISP broadband (DSL, cable, fiber)

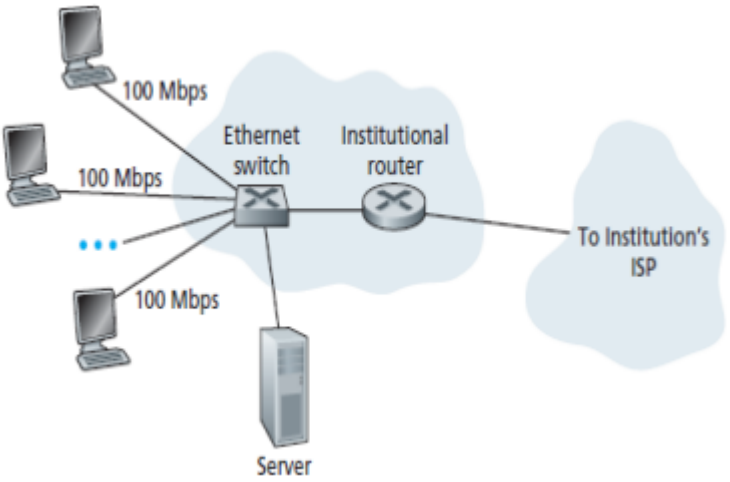
- A home router connects multiple devices using Wi-Fi or Ethernet





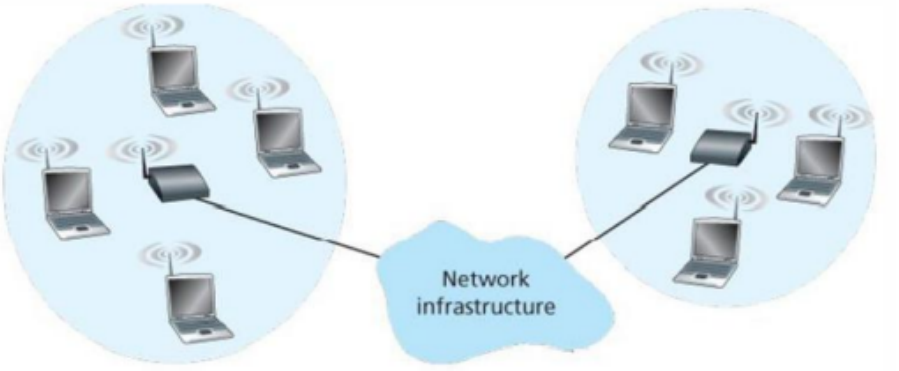
▼ Enterprise Networks

- Used in universities, companies, hospitals
- Usually large LANs with hundreds/thousands of devices
- Often use Ethernet switches inside + firewalls/security systems at the edge



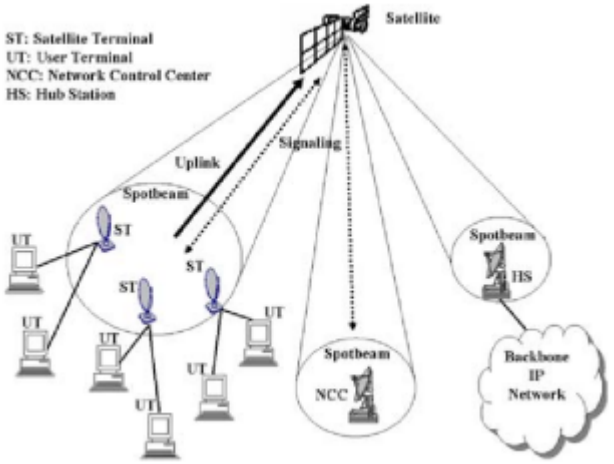
▼ Mobile/Cellular Networks

- Devices connect via 4G/5G towers
- The tower connects to the mobile operator's core, which links to the internet



▼ Satellite & Wireless Access

- Used in rural/remote areas
- Starlink (SpaceX) is a modern example of satellite internet



▼ Nomenclature under different access ISP architectures

Feature	DSL based access Network	Cable TV based access network	FTTH Based access network
Modem	DSL modem	Cable modem	Optical modem
Local Multiplexer	Splitter	Fiber node	Optical network terminator
Central office	DSL access multiplexer (DSLAM)	Cable modem terminating system (CMTS)	Optical line terminator (OLT)
Downlink rates	12 Mbps [ITU 1999] and 24 Mbps [ITU 2003]	DOCSIS 2.0 standard 42.8 Mbps	100 Mbps (cable length based)
Uplink rates	1.8 Mbps [ITU 1999] and 2.5 Mbps [2003]	DOCSIS 2.0 standard 30.7 Mbps	30 Mbps (cable length based)

▼ Types of Link media

- The physical medium that carries data from your device to the ISP:
  - *Twisted Pair* - Used in DSL Telephone lines (Copper Wires)
  - *Coaxial Cable* - Used in cable internet (ACT, Hathway)
  - *Fiber Optic* - Fastest and most reliable, carries light signals
  - *Wireless* - Wi-Fi , 4G,5G, microwave, satellite links

▼ Host Roles at the Edge

- *Client* - Requests Data/Services (browser fetching a webpage)

- *Server* - Provides data/services (Google's servers sending the webpage)
- Sometimes devices can be both (P2P file sharing)

▼ Network Core

- Network Core or "The backbone network"
- Once data leaves the edge, it enters the core of the internet
- The core is made of powerful routers and high-speed links that move data across cities, countries and continents

▼ Two Main Switching Methods

▼ Circuit Switching

- A dedicated path is established between sender and receiver before communication starts
  - Example - Traditional telephone networks
- Pro : It provides guaranteed bandwidth and delay
- Con : Inefficient because if the line is idle, resources are wasted
- In summary, it is like reserving the whole train for yo

▼ Packet Switching

- Data is broken into packets. Each packet finds its own path through the network
- Routers forward packets one by one using the best available route
- Pro : Efficient use of network, supports many users
- Con : Can cause delays (packets may queue or take different routes)
- In summary, it is like putting your luggage into trains shared by others

▼ Types of switching inside routers

- *Store and Forward* : A router must receive the entire packet before forwarding

- *Cut-Through* : Starts forwarding as soon as it reads the destination address (Less common)

▼ Internet’s Core Structure

- The core is built from ISPs interconnected at different levels (tier 1, regional, local)
- Uses redundant paths (so if one link fails, packets can reroute)
- The core provides scalability and reliability by not depending on one single path or provider

▼ Routing vs Forwarding

- *Forwarding* : Moving a packet one hop (like a local delivery)
- *Routing* : Choosing the path the packet will take (like planning the entire journey)
- Routers having routing tables built using algorithms (ex : shortest path)

▼ Delay, Loss and Throughput in packet-switched networks

- When packets travel through the internet, they don’t always arrive instantly because of several factors like delay, packet loss, speed of data flow etc.,

▼ The delays that occur in a packet switched transmission are

- *Processing Delay*
  - Time for a router to examine the packet’s header and decide where to forward it
  - Usually it is very small
- *Queuing Delay*
  - Time spent waiting in the router’s queue if many packets arrive at once
  - Depends on traffic load
  - It can be zero also if there is no congestion or very large if there is a lot of congestion
- *Transmission Delay*

    - The time taken to push all the packet's bits onto the link
    - $d_{trans} = \frac{L}{R}$
      L : packet length (bits)
      R : link transmission rate (bps)
  - *Propagation Delay*
    - Time taken for a signal to travel through the medium (fiber, copper, etc.,)
    - $d_{prop} = \frac{d}{s}$
      d : distance
      s : propagation speed ($2 * 10^7 m/s$ in fiber)
  - Total nodal delay $d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$

▼ Packet Loss

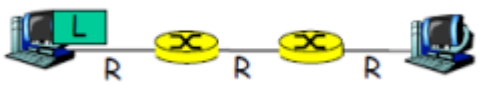
- Happens when queues overflow at a router (buffer is full)
- Dropped packets need to be retransmitted by the sender (if using TCP)
- It is very common in congested networks

▼ Throughput

- Rate at which bits are delivered from sender to receiver
- 2 Types
  - *Instantaneous throughput* - at a given moment
  - *Average throughput* - over a long period
- Bottleneck Link
  - The slowest link on the path determines the throughput

▼ Questions

▼ For the scenario given below, assume the queuing delay, propagation delay and processing delay to be negligible. Suppose packet length L = 7.5 Mb and link rate R = 1.5 Mbps. Calculate the end-to-end delay



- Given L = 7.5 Mb and R = 1.5 Mbps
  But there are 3 links
  so $d_{trans} = \frac{7.5*10^6}{1.5*10^6} = 5s$
  then $d_{end-to-end} = 3 * 5 = 15s$

▼ Suppose there is a 10 Mbps microwave link between a geostationary satellite and its base station on Earth. Every minute the satellite takes a digital photo and sends it to the base station. Assume a propagation speed of $2.4 \times 10^8 m/s$.
What is the propagation delay of the link?
What is the bandwidth-delay product, R · dprop?
Note : R. dprop → Represents the maximum number of bits in the channel
Let x denote the size of the photo. What is the minimum value of x for the microwave link to be continuously transmitting?

- Propagation delay = $\frac{(36000km)}{(2.4*10^8 m/s)} = 0.15s$
- $R.d_{prop} = (10 * 10^6) * (0.15) = 1.5Mb$
- In 1 minute, $R * T = (10 * 10^6) * 60 = 600Mb$

▼ Consider the figure below where transmission delay is the only significant delay. Each link is 2Mbps. Suppose the number of links N is 3. Calculate the end to end delay for the two cases given below. Note that each switch is a store and forward switch.
If message of size 8 Mb is transferred without segmentation.
If the message is segmented into 800 packets of 10 kb length.

- Given message size = 8 Mb
  R = 2 Mbps
  N = 3
  since no segmentation, $d_{trans} = \frac{L}{R} = \frac{8*10^6}{2*10^6} = 4s$
  and since 3 links, $d_{end-to-end} = N * d_{trans} = 3 * 4 = 12s$
- Each packet is segmented into 10 Kb size,
  $\frac{L}{R} = \frac{10000}{2*10^6} = 0.005s$
  Then we use the formula $d_{end-to-end} = N * \frac{L}{R} + (P - 1) * \frac{L}{R}$
  $d_{end-to-end} = 3 * 5ms + (800 - 1) * 5ms = 802 * 5ms = 4.01s$

▼ Protocol Layers and Service Models

▼ Use of Layers

- To manage the complexity of internet, we divide networking into layers, each with a specific role
- Each layer provides services to the layer above, and uses services from the layer below

▼ Internet's 5 Layer Model

▼ Application Layer

- Applications running on hosts generate/receive data
- Data is referred as a message
- A process initiates communication with another by sending a query/request
- Message is formatted according to application layer protocol
- Applications can have QoS (Quality of Service) requirements

▼ Transport Layer

- Responsible for providing QoS for messages
- Performs multiplexing at sender and demultiplexing at receiver
- Maps each message to corresponding process
- Appends new header to each message which is called segment

▼ Network Layer

- Fragments the segments into packets and moves them hop-by-hop (router to router)
- Uses source and destination IP addresses
- Path between source host and destination host is discovered
- Appends new header to each packet which is called datagram

▼ Link Layer

- Pushes the packets onto a link using link layer protocols
- Can forward frames using MAC address
- Appends a new header to packets which is called frame
- Provides synchronization at receiver

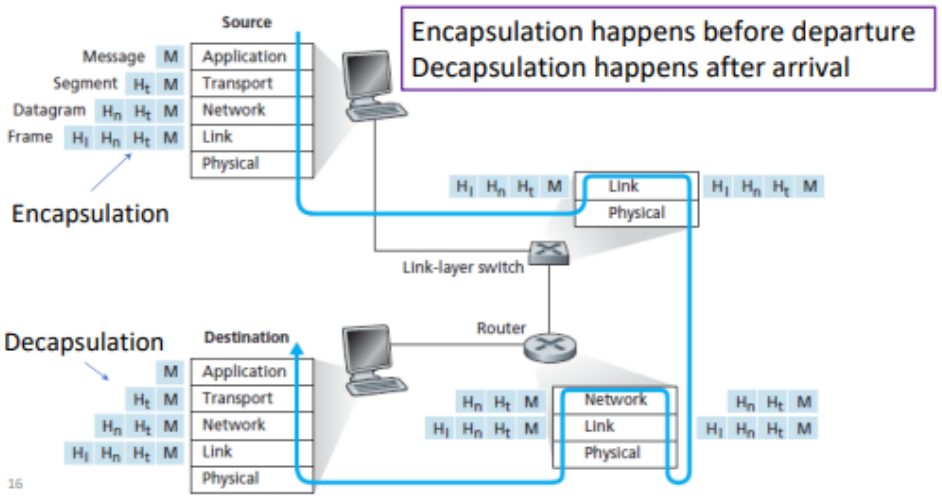
- Transforms a raw transmission facility into a line that appears free to undetected transmission errors by masking real errors so network layer
- The sender breaks up the input data into data frames and transmits them sequentially. If the service is reliable, receiver confirms correct receipt of each frame by sending back an *acknowledgement frame*
- An issue that arises in link later is how to keep a fast transmitter from drowning a slow receiver in data. Some traffic regulation mechanism must be used to let the transmitter know when the receiver can accept more data

▼ Physical Layer

- Provides physical interface between host and link
  Example : modem and ethernet card, wireless adapter
- Transmits raw bits over a communication channel
- Converts binary data into signals
- Performs modulation and demodulation
- Performs transmission, reception and filtering of signals

▼ Encapsulation and Decapsulation

- When a message goes down the stack, each layer adds its own header and it becomes a new data unit
- Application data → TCP segment → IP packet → Link frame → Bits



▼ Principles of Network Applications

- Applications are programs that run at the edge of the network (on clients and servers)
- Example : Web browsers, WhatsApp, Zoom, Email clients, Netflix etc.,
- They communicate across the internet using sockets and protocols

▼ Application Architecture

- Application architecture dictates how the application developer views the interaction between applications running on the end-systems
- 2 Main Styles

▼ Client Server Architecture

- *Server* : Always on, has fixed IP, provides services (ex : Google server etc.,)
- *Client* : Initiates communication, requests services
- example : web browsing, email

▼ Peer-to-Peer Architecture

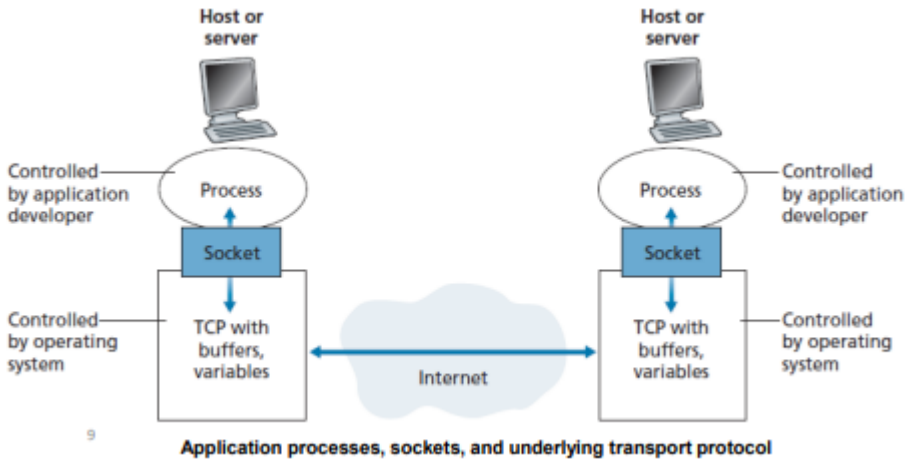
- No always-on server; *Peers* (end systems) communicate directly
- Each peer can act as both client and server
- Scales better, reduces cost
- example : BitTorrent, Skype (older versions)

- Many modern apps use a hybrid (client-server + P2P)
  - example : Zoom has servers for coordination, but video streams may go peer-to-peer

▼ Processes Communication

- An application in one host is called a process
- Processes communicate across the internet by exchanging messages
- A process sends/receives through a socket

- Identified by : IP address + port number



Application processes, sockets, and underlying transport protocol

▼ Transport Services need by applications

- Different apps need different levels of services:
  - *Reliable data transfer* for no loss, in-order delivery. Needed by web, email, file transfer
  - *Low latency* is needed for gaming, voice, video conferencing
  - *Throughput* (bandwidth) needed by large file downloads, video streaming
  - *Security* for confidentiality, authentication, integrity like HTTPS

Application	Data Loss	Throughput	Time-Sensitive
File transfer/download	No loss	Elastic	No
E-mail	No loss	Elastic	No
Web documents	No loss	Elastic (few kbps)	No
Internet telephony/ Video conferencing	Loss-tolerant	Audio: few kbps–1 Mbps Video: 10 kbps–5 Mbps	Yes: 100s of msec
Streaming stored audio/video	Loss-tolerant	Same as above	Yes: few seconds
Interactive games	Loss-tolerant	Few kbps–10 kbps	Yes: 100s of msec
Smartphone messaging	No loss	Elastic	Yes and no

▼ Transport Protocol Choices

▼ TCP (Transmission Control Protocol)

- Connection-oriented

- Reliable, in-order delivery
- Congestion control, flow control
- Used by HTTP, SMTP, FTP

▼ UDP (User Datagram Protocol)

- Connectionless, No guarantees
- Low overhead, faster
- Used by DNS, video streaming, online gaming

▼ Application Layer Protocols

- They define how applications exchange messages
  - HTTP (web), SMTP/IMAP/POP3 (email), DNS (name resolution), FTP (file transfer)
- Protocols define
  - Message syntax (fields, structures)
  - Semantics (meaning of fields)
  - Rules (who speaks first, when to respond)

▼ Web and HTTP

▼ Web

- Web page consists of *objects* each of which can be stored on different web servers
- Object can be HTML file, JPEG image, Java applet, audio file etc.,
- Web page consists of a base HTML file which includes several referenced objects, each addressable by a *URL*

  `www.pes.edu/ec/pic.gif`

  www.pes.edu → host name
  ec/pic.gif → path name

▼ HTTP

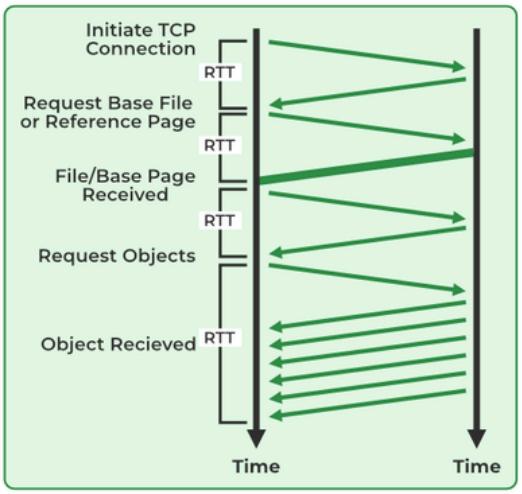
- HTTP stands for Hyper Text Transfer Protocol
- It is the application-layer protocol of the web
- It defines how browsers and servers communicate

- Uses TCP as transport protocol (default port : 80 for HTTP or 443 for HTTPS)
  Client initiates TCP connection and Server accepts TCP connection
  Once HTTP messages are exchanged between browser and web server, TCP connection is closed
- HTTP is stateless : server maintains no past requests from clients
  Every request is treated independently but we add cookies later to maintain state

▼ HTTP Connections

▼ Non-Persistent HTTP (HTTP/1.0)

- Each request/response uses a new TCP connection
- At most one object sent over TCP connection
- TCP connection closed
- Downloading multiple objects require multiple connections

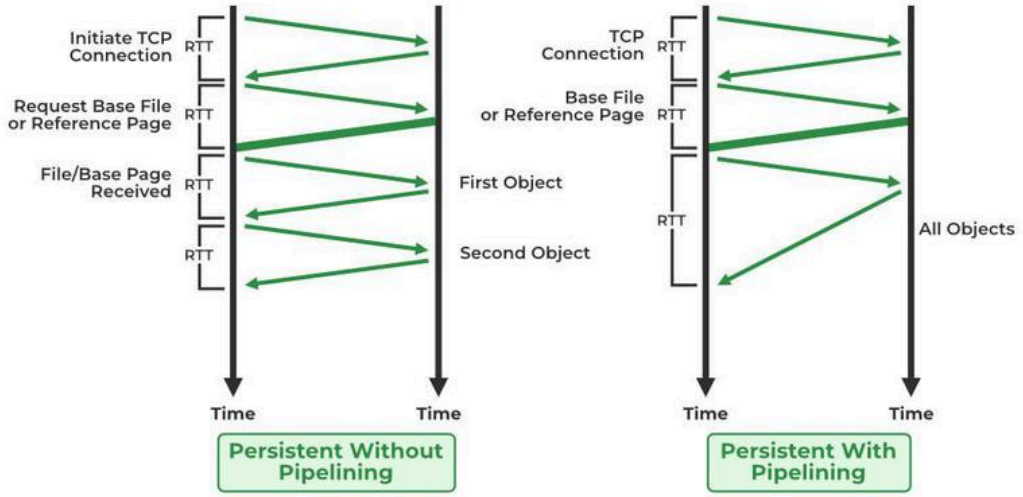


Non-Persistent & Parallel Connections

▼ Persistent HTTP (HTTP/1.1 and later)

- TCP connection opened to a server
- Multiple objects can be sent over single TCP connection
- TCP connection closed
- Faster, less overhead
- Server closes connection after some specified time of inactivity

- In HTTP/1.1 → Allows up to 6 parallel TCP connections
  In HTTP/2 → Includes multiplexing, message prioritization and server pushing



▼ RTT in HTTP

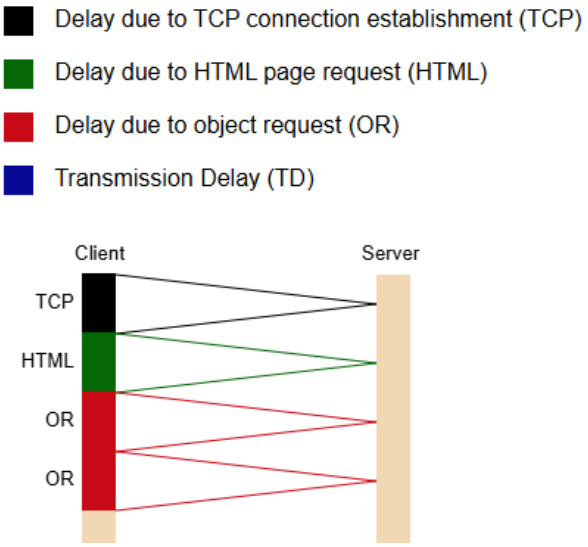
- RTT stands for Round Trip Time
- It is the time for a packet to go from client to server and then back to client
- In HTTP, one RTT for the TCP handshake and one more RTT for HTTP request + first byte of response
  So, Total = 2 RTT + transmission time

▼ Questions

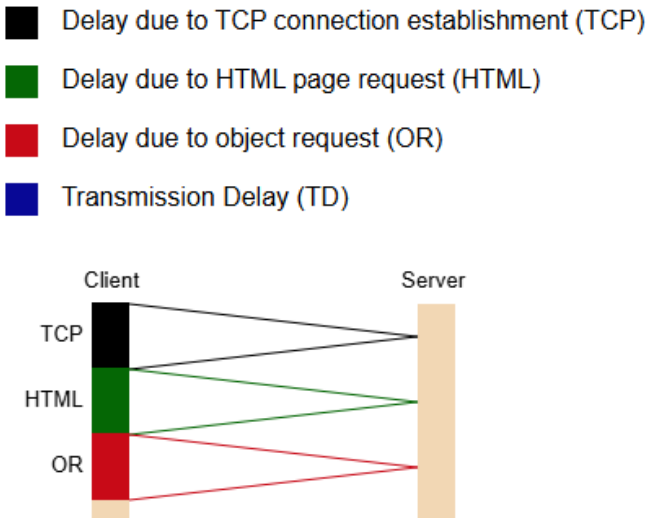
▼ Consider accessing the webpage ww.someSchool.edu/someDepartment/Schoolpage.html which contains two embedded objects. Suppose the Web server and client are connected by a long link of rate R. Let RTT denote the two way propagation delay. Suppose the length (bits) of the webpage and two objects are L1, L2 and L3 respectively.
Suppose the HTTP request message is of negligible length and can be piggybacked with acknowledgements.
Calculate separately, the total access delay under a persistent TCP connection and non-persistent TCP connections. Show the timing diagram

- For persistent + serial,
  TCP setup for main page = $1RTT$

Request + Response for main page = $1RTT + \frac{L_1}{R}$
Request + Response for object 1 = $1RTT + \frac{L_2}{R}$
Request + Response for object 2 = $1RTT + \frac{L_2}{R}$
Total = $4RTT + \frac{L_1+L_2+L_3}{R}$

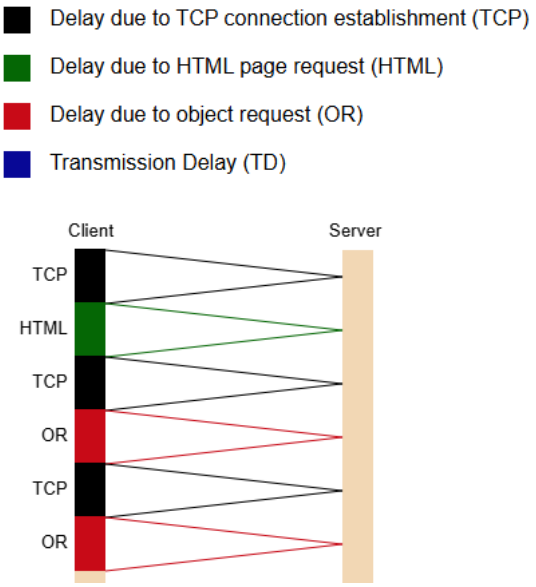


- For persistent + parallel,
  TCP setup for main page = $1RTT$
  Request + response for main page = $1RTT + \frac{L_1}{R}$
  Client sends requests for both objects together, so depends on whichever takes more time, Request = $1RTT + max(\frac{L_2}{R}, \frac{L_3}{R})$
  Total = $3RTT + \frac{L_1}{R} + max(\frac{L_2}{R}, \frac{L_3}{R})$

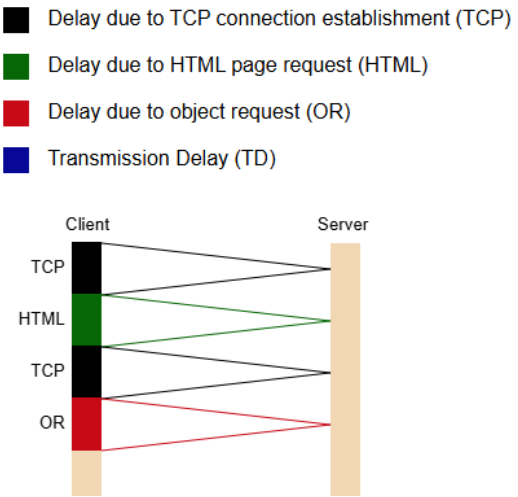


- For non-persistent + serial,
  For each object a new TCP connection is made and costs an

extra RTT
Request for main page = $2RTT + \frac{L_1}{R}$
Request for object 1 = $2RTT + \frac{L_2}{R}$
Request for object 2 = $2RTT + \frac{L_3}{R}$
Total = $6RTT + \frac{L_1+L_2+L_3}{R}$



- For non-persistent + parallel,
  TCP setup + request/response = $2RTT + \frac{L_1}{R}$
  Client opens TCP connection for both objects in parallel = $2RTT + max(\frac{L_2}{R} + \frac{L_3}{R})$
  Total = $4RTT + \frac{L_1}{R} + max(\frac{L_2}{R} + \frac{L_3}{R})$



▼ Common HTTP Methods

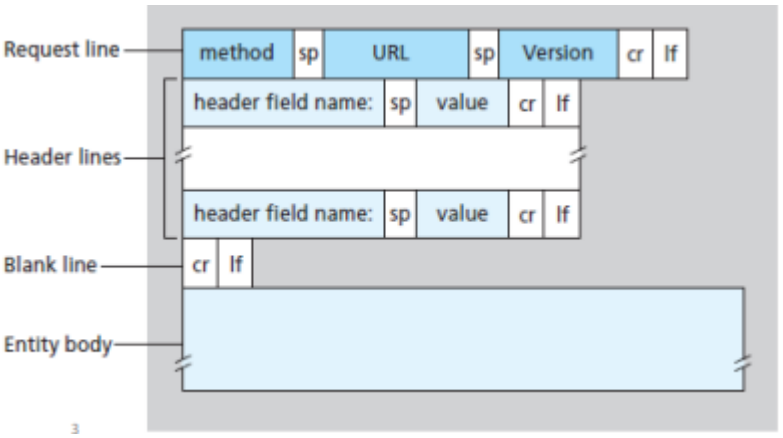
- **GET** → request data (most common)

- **POST** → send data (forms, uploads)
- **HEAD** → request headers only
- **PUT** → upload/replace data
- **DELETE** → remove resource

▼ HTTP Message Format

▼ HTTP Request Message (By client)

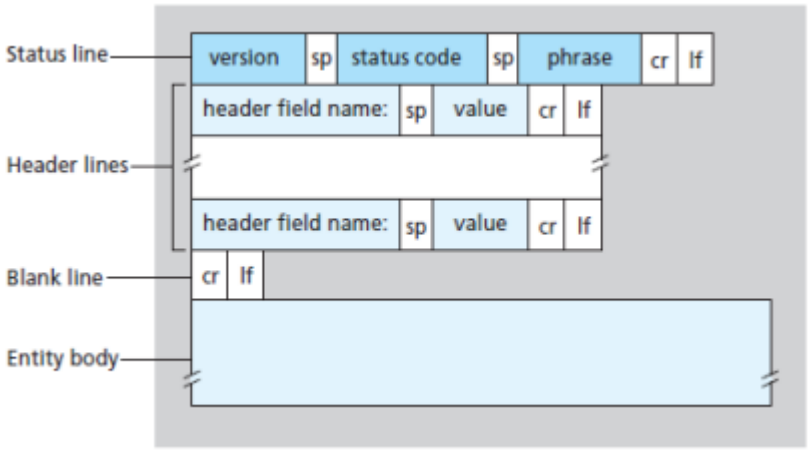
```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Chrome/119
Accept-Language: en
```



▼ HTTP Response Message (By server)

```
HTTP/1.1 200 OK
Date: Tue, 05 Sep 2023
Server: Apache/2.2
Content-Type: text/html
Content-Length: 1024

<html> ... actual webpage ... </html>
```



▼ HTTP Response Status Codes

- Status code appears in 1st line in server-to-client response message
- `200 OK` - Request succeeded, requested object later in this message
- `301 Moved Permanently` - Requested object moved, new location specified later in this message (in Location: field)
- `400 Bad Request` - Request message not understood by server
- `404 Not Found` - Requested document not found on this server
- `505 HTTP Version Not Supported`

▼ Commonly these are the codes

- 1xx = Informational
- 2xx = Success
- 3xx = Redirection
- 4xx = Client Error
- 5xx = Server Error

▼ Questions

▼ Consider the figure below, where a client is sending an HTTP GET message to a web server, gaia.cs.umass.edu .
Suppose the client-to-server HTTP GET message is the following:
GET /kurose_ross_sandbox/interactive/quotation10.htm HTTP/1.1
Host: gaia.cs.umass.edu
Accept: text/plain, text/html, text/xml, image/gif, image/png, audio/mp4, audio/basic, video/mpeg, video/wmv,

Accept-Language: en-us, en-gb;q=0.1, en;q=0.2, fr, fr-ch, zh, cs
If-Modified-Since: Thu, 17 Jul 2025 08:55:22 -0700
User Agent: Mozilla/5.0 (Windows NT 6.1; WOW64)
AppleWebKit/535.19 (KHTML, like Gecko)
Chrome/18.0.1025.168 Safari/535.19

▼ What is the name of the file that is being retrieved in this GET message?

- quotation10.htm

▼ What version of HTTP is the client running?

- HTTP/1.1

▼ True or False: The client will accept html files

- True

▼ True or False: The client will accept jpeg images

- False

▼ What is the client's preferred version of English?

- American English

▼ What is the client's least preferred version of English?

- British English. q = 0.1 for en-gb

▼ True or False: The client will accept the German language

- False. De is not present

▼ True or False: The client already has a cached copy of the file

- True. Check for “If-Modified-Since”

▼ Consider the figure below, where the server is sending a HTTP RESPONSE message back the client.
Suppose the server-to-client HTTP RESPONSE message is the following:
HTTP/1.1 200 OK
Date: Thu, 17 Jul 2025 15:37:38 +0000
Server: Apache/2.2.3 (CentOS)
Last-Modified: Thu, 17 Jul 2025 16:00:38 +0000
ETag:17dc6-a5c-bf716880.

Content-Length: 4230
Keep-Alive: timeout=48, max=77
Connection: Keep-alive
Content-type: image/html

▼ Is the response message using HTTP 1.0 or HTTP 1.1?

- HTTP 1.1

▼ Was the server able to send the document successfully? Yes or No

- Code is 200 OK, so Yes

▼ How big is the document in bytes?

- 4230 bytes

▼ Is the connection persistent or nonpersistent?

- Persistent

▼ What is the type of file being sent by the server in response?

- image.html

▼ What is the name of the server and its version? Write your answer as server/x.y.z

- Apache/2.2.3

▼ Will the ETag change if the resource content at this particular resource location changes? Yes or No

- Yes. Etag is a string that uniquely identifies a resource. If resource is updated, Etag will change

▼ Cookies and Web Caching

- Each HTTP request is independent because the server doesn't remember past requests (Stateless)
- But many applications need state (Like shopping carts, logins, preferences)

▼ Cookies

- Cookies are small pieces of data stored at the client side (in browser)

- They allow the server to keep track of users across multiple requests

▼ How they work

1. When a client logs in, the server generates an ID and sends in the response header `Set-Cookie: ID=12345`
2. Client stores it locally in browser's cookie file
3. For every future request to that site, the browser automatically includes `Cookie: ID=12345`
4. Server uses that ID to fetch your info

▼ Web Caching

- If every client fetched objects directly from the original server, it would be slow and expensive
- Web cache or Proxy server is a network device between client and server that keeps copies of recently requested web objects
- Next time another client requests the same object, cache responds directly (faster)
- This reduces response time (data is closer to client), reduces traffic on access link (saves bandwidth) and reduces load on original server

▼ How it works

- Browser sends request and cache checks if it has a fresh copy
- If it has, cache sends directly but if it doesn't cache forwards request to origin server, then saves the response for later

▼ HTTP headers for caching

- `Cache-Control: max-age=3600` → Object is valid for 1 hour
- `Expires: <date>` → After this time, cache must fetch a new copy
- `If-Modified-Since` → Client asks server if object has changed. If unchanged, server replies "304 Not Modified" and saves bandwidth

- In summary, cookies help HTTP maintain state (user sessions, logins, shopping cart) whereas web caching stores frequently requested objects closer to users to reduce delay and bandwidth usage

▼ DNS

- DNS stands for Domain Name System
- It is easier to find a particular server using a name instead of their IP address
  so, DNS is like the phonebook of the internet which maps the names to their respective IP address

▼ DNS Services

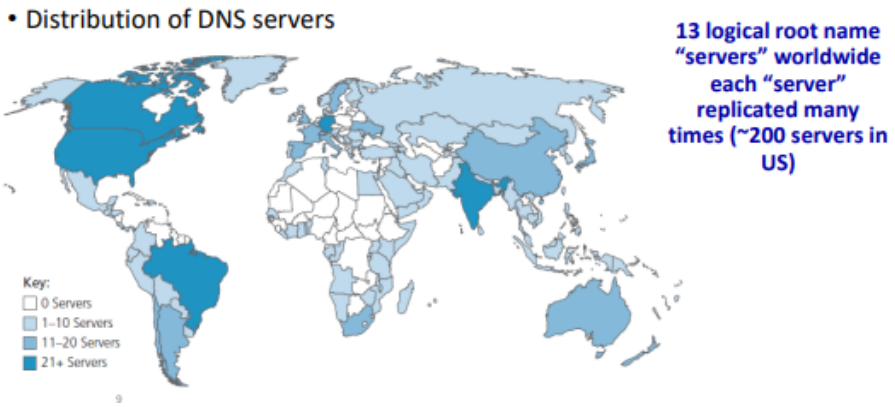
- Hostname ↔ IP address translation (main job)
- *Host aliasing*: A server may have multiple names (for example, `www.youtube.com` is an alias for `yt3.ggpht.com`
- *Main server aliasing*: Email uses DNS to map `@gmail.com` → Google's mail servers
- *Load distribution*: One hostname can map to multiple IPs, so client requests are spread across servers

▼ DNS Hierarchy

- DNS is not a single server, it is a hierarchical distributed database

▼ Root DNS servers

- Around 600 servers worldwide



- Point to *Top-Level Domain* (TLD servers)

▼ TLD servers

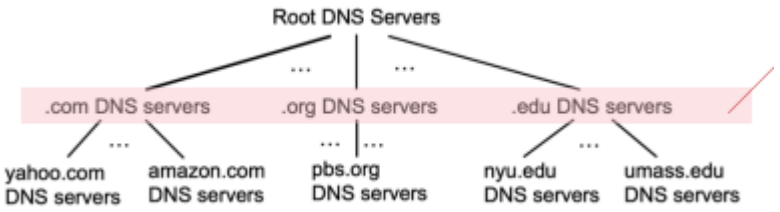
- Responsible for top-level domains like `.com` , `.org` , `.edu` , `.in`
- Point to authoritative DNS servers

▼ Authoritative DNS servers

- Store the actual mapping for specific domains
  - For example, `www.pes.edu` 's authoritative server at the university holds the mapping

▼ Local DNS Resolver

- When you type a website, your ISP's local DNS checks its cache or else forwards the request up the hierarchy



1st layer : Root DNS servers
2nd layer : TLD servers
3rd layer : Authoritative DNS servers

▼ DNS Query Process

- Assume we type `www.pes.edu` in the browser

1. Browser sends a query to *Local DNS* (resolver, often at your ISP)
2. If not cached, Local DNS queries the *Root Sever*
3. Root server replies "Ask the `.edu` TLD server"
4. *TLD* replies "Ask the authoritative server for pes.edu"
5. *Authoritative server* replies with IP address
6. Local DNS gives IP to browser, which then contacts the web browser

▼ DNS Caching

- Every DNS server caches mappings for efficiency
- Reduces load and speeds up lookups

- Entries have a TTL (Time-To-Live) after which they expire

▼ DNS Message Format

- Uses UDP (port 53) usually, sometimes TCP
- Query and Response have similar structure
  - Header (ID, Flags)
  - Question (name being asked)
  - Answer (IP mapping)
  - Authority (other authoritative servers)
  - Additional (extra info, like mail servers)

▼ DNS Resource Records (RR)

- A DNS database is made of Resource Records
- Each RR has this format : `Name, Value, Type, TTL`

▼ A Record (Address Record)

- Maps a hostname → IPv4 address
- Example : `( www.iitm.ac.in , 14.139.160.3, A, 86400)`
  meaning “www.iitm.ac.in” has IPv4 = 14.139.160.3 valid for 1 day

▼ AAAA Record (Quad A)

- Maps a hostname → IPv6 address
- Example: `( www.google.com , 2404:6800:4007:80a::200e, AAAA, 300)`

▼ CNAME (Canonical Name)

- Maps a hostname to another hostname (aliasing)
- Example: `( www.youtube.com , yt3.ggpht.com , CNAME, 400)`
  meaning “www.youtube.com” is just an alias for “yt3.ggpht.com”

▼ MX Record (Mail Exchange)

- Maps a domain to mail server
- Example: `( iitm.ac.in , mail.iitm.ac.in , MX, 600)`
  meaning send mail for `@iitm.ac.in` users to “mail.iitm.ac.in”

▼ NS Record (Name Server)

- Maps a domain to authoritative DNS server
- Example: `( iitm.ac.in , ns1.iitm.ac.in , NS, 172800)` meaning the authoritative server for “iitm.ac.in” is “ns1.iitm.ac.in”

▼ Other useful Records

- PTR (Pointer) : IP → Hostname (reverse DNS lookup)
- SOA (Start of Authority) : Metadata about the zone (primary server, contact info)
- TXT : Stores text info (often for verification or SPF email records)

▼ Summary

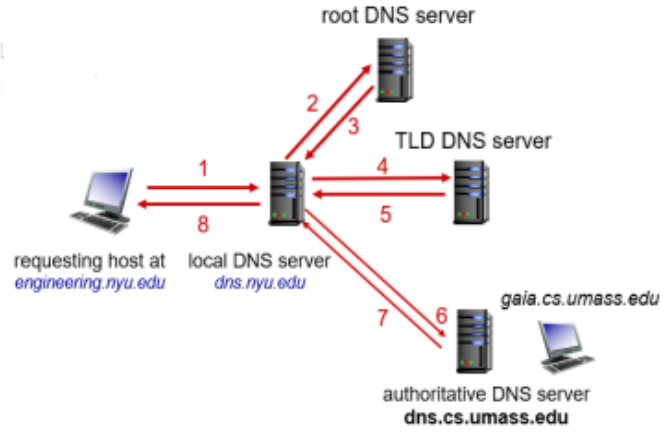
- **A / AAAA** → Hostname → IP.
- **CNAME** → Hostname → another hostname.
- **MX** → Domain → mail server.
- **NS** → Domain → authoritative name server.
- **PTR** → IP → Hostname.

▼ Iterative vs Recursive DNS Queries

▼ Iterative

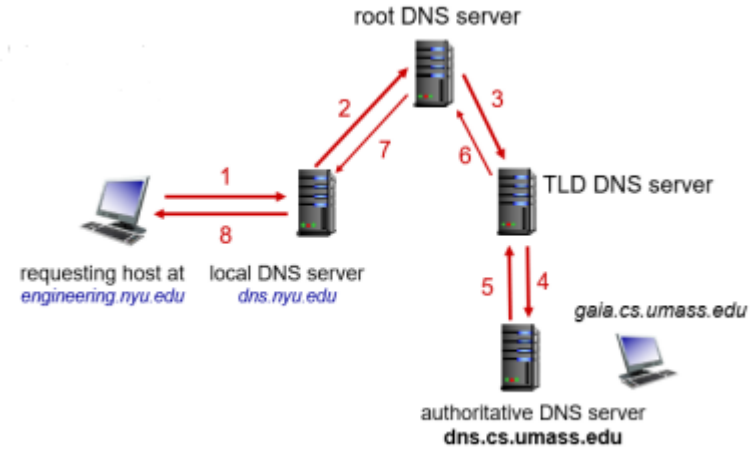
- The DNS server replies with best information it has
- If it doesn’t know the answer, it returns a referral (another DNS server’s address) instead of fetching the result
- Then the client must query that referred server
- This keeps going on until the client finds the final answer
- Example :
  - Client → Local DNS → Root : “Where is www.iitm.ac.in?”
  - Root : “Ask the “.in” TLD server”
  - Client → TLD: “Where is www.iitm.ac.in?”
  - TLD : “Ask the authoritative server for iitm.ac.in”
  - Client → Authoritative: “Where is www.iitm.ac.in”?

    - Authoritative : “Here is the IP: 14.139.160.3”
- Control is with client



▼ Recursive

- DNS server does all the work for the client
- If the server doesn’t know the answer, it queries other DNS servers on behalf of the client
- Finally it returns the actual IP to the client
- Example :
    - Client → Local DNS: “Where is iitm.ac.in?”
    - Local DNS queries Root → gets referral to “.in”
    - Local DNS queries “.in” → gets referral to iitm.ac.in authoritative
    - Local DNS queries authoritative → gets IP
    - Local DNS → Client: “Here is the IP: 14.139.160.3”
- Control is with server



▼ DNS Caching and Vulnerabilities

▼ DNS Caching

- To avoid repeating the whole root DNS → TLD → authoritative DNS process every time, DNS servers cache the results
- Each DNS record comes a TTL (Time-To-Live) which decides how long it stays valid in the cache
  - Example : when you type `www.iitm.ac.in` , Local DNS resolver fetches the IP once. If you or anyone else on that network requests it again before TTL expires → resolved instantly from cache (no extra queries)
- Caching occurs in the following places
  - Browser cache (short-term, per tab/session)
  - OS cache (system-wide, flushed when DNS service starts)
  - Local DNS resolved (ex: ISP's DNS)
  - Even recursive DNS servers cache for many clients
- Benefits
  - Faster response time for users
  - Reduces internet traffic
  - Reduces load on authoritative servers

▼ DNS Vulnerabilities

- DNS Spoofing/Cache Poisoning

        - Attacker injects a fake IP address into a DNS resolver's cache
        - Then when users try to visit a site, they are redirected to another malicious server
    - DNS Amplification Attack (DDoS)
        - Attackers send small DNS queries with a spoofed source IP (victim's IP)
        - Open DNS servers reply with much larger responses
        - This results to victim getting flooded with traffic (amplification)
    - Man in the middle attack
        - Attacker intercepts DNS queries and replies with false records
    - Solutions & Protections
        - *DNSSEC* - (DNS Security Extensions) add digital signatures to DNS responses to prevent tampering
        - *Randomizing source port + Query ID* - Make spoofing harder
        - *Using trusted resolvers* - Like Google DNS `8.8.8.8` and Cloudflare `1.1.1.1`

▼ Electronic Mail in the Internet

▼ Three Major Components

▼ User Agents (UA)

- Software that lets users compose, read and send emails
- Examples : Gmail, Apple mail, Outlook, Thunderbird

▼ Mail Servers

- Each organization has mail servers (ex : gmail.com, iitm.ac.in)
- They handle sending and receiving of mail
- Each server has a mailbox for users and a message queue for outgoing mail

▼ Protocols

- Used to send and receive mail
- SMTP, POP3, IMAP, HTTP

▼ How Email Works

- If the task was to send a mail from bobthegreat@gmail.com to unknownoodles@iitm.ac.in

1. bobthegreat's user agent hands the message to Gmail's outgoing mail server
2. Gmail server uses SMTP to contact IITM's mail server and transfers the message
3. The mail is stored in unknownoodles' mailbox at IITM's server
4. unknownoodles uses his user agent to retrieve the mail using POP3/IMAP/HTTP

▼ SMTP (Simple Mail Transfer Protocol)

- It is used for sending messages from client → server and server → client
- Runs on TCP, port 25
- It is a push protocol (sender pushes mail to receiver server)
- Works with plain text commands like `HELO`, `MAIL FROM`, `RCPT TO`, `DATA`, `QUIT`
- Example :

```
S: 220 iitm.ac.in SMTP server ready
C: HELO gmail.com
C: MAIL FROM:<bobthegreat@gmail.com>
C: RCPT TO:<unknownoodles@iitm.ac.in>
C: DATA
C: (message body)
C: -
C: QUIT
S: 221 Goodbye
```

- MIME or Multipurpose Internet Mail Extensions is what makes attachments possible in email. It converts the binary file (like PDF,

image, MP3) into a text-based encoding (like Base64) so it can be safely carries over SMTP, which was originally designed only for ASCII text

▼ POP3 vs IMAP vs HTTP

Protocol	Use	Key Feature
POP3 (Post Office Protocol v3)	Retrieve mail	Downloads mail from server to client, often deletes from server. Simple but limited
IMAP (Internet Message Access Protocol)	Retrieve mail	Keeps mail on server, allows folders, multiple devices to sync. More powerful
HTTP	Web-based email	You access mail via web browser using HTTP/HTTPS, no separate client needed

▼ Video streaming

- Video is a sequence of images, typically being displayed at a constant rate and streaming means playing video as it downloads, rather than waiting for the whole file
- An uncompressed, digitally encoded image consists of an array of pixels where each pixel is encoded into number of bits to represent luminance and color

Video Quality	Bit Rate	Resolution
SD	800-1000 Kbps	480p
HD	1.2-2 Mbps	720p
FHD	1.9-4.5 Mbps	1080p
UHD	10 Mbps	2160p

- This needs
  - high throughput (lots of bits/sec)
  - low jitter (smooth playback)
  - low delay (so playback doesn’t freeze)

▼ Types of Streaming

- Streaming stored video
  - example : Netflix, YouTube
  - Video is pre-recorded and stored on servers
  - Can use buffering, adaptive bitrate, caching
- Live streaming
  - example : JioHotstar, Twitch
  - Video is generated in real-time and transmitted almost immediately
  - More sensitive to delay

- Streaming challenges include
  - Variable bandwidth (Unstable internet)
  - Packet loss (Especially with UDP)
  - Playout Delay (Buffering is needed so playback doesn't freeze)
- To solve these challenges we use *DASH*

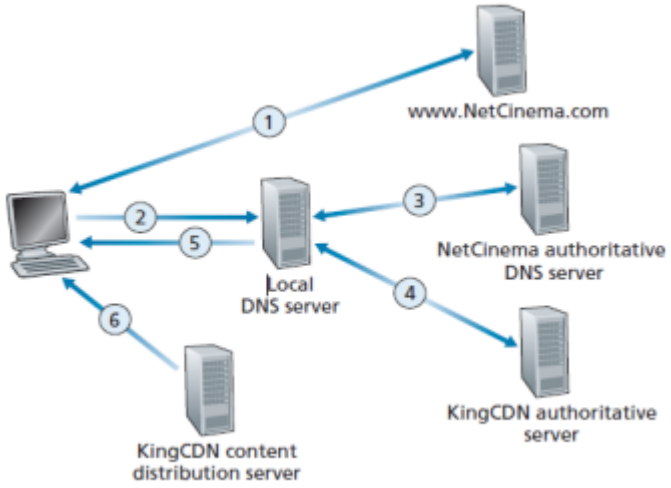
▼ DASH

- Stands for Dynamic Adaptive Streaming over HTTP
- The core idea is to chop the video into chucks (few seconds each)
- Each chunk is stored at multiple quality levels (240p, 480p, 720p, 1080p etc.,) which can vary based on response to changes in available bandwidth at the client
- Before playback, client gets a manifest file (MPD) that lists chunks, bitrates available, URLs to fetch them
  Client then downloads the MPD and pulls one chunk at a time using HTTP GET
- The main advantages of of DASH are
  - Adaptation to bandwidth fluctuations
  - Works over HTTP → No need special streaming protocols
  - Allows CDN caching (chunks are just HTTP objects)

▼ Content Distribution Networks (CDN)

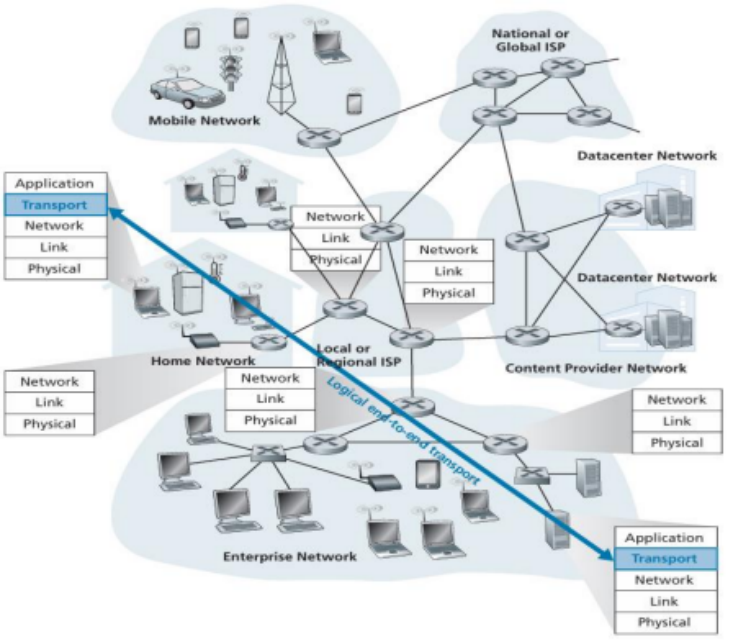
- The objective of DASH is to ensure quality of experience for the client after a server is chosen
- The object of CDN is to maintain the videos closer to the clients and resolve server assignment for video streaming
- CDNs distribute video content across geographically spread servers
- When a client requests a video, DNS redirects them to nearest CDN node
- Example : Netflix Open Connect, Google CDN
- Benefits : Lower latency, Load balancing, Scalability
- When a streaming service decides to use CDN, they need to decide where to put their servers. There are 2 approaches
  - ▼ Enter Deep
    - CDN servers are placed deep inside ISPs' networks
    - Goal is to be as close as possible to end-users to reduce latency and congestion
    - Example : Netflix Open Connect appliances installed in local ISPs
    - This has very low latency, less traffic on backbone networks
    - But it is really expensive, and must deploy and maintain many small servers
  - ▼ Bring Home
    - CDN servers are placed at a small number of large PoPs (Points of Presence) near Internet Exchange Points (IXPs)
    - Users' traffic is redirected to these PoPs
    - This reduces the number of servers and hence easy to manage
    - But it is not as close to end-users which means higher latency compared to Enter Deep
- When a client tries to access a video on the content providers webpage, the DNS servers help locate the server cluster under CDN

and locate appropriate server
Following this, TCP connection is established to the server and then HTTP based DASH takes over during streaming
This is seen in the below example of a client trying to access a movie from Netcinemas which is distributed by KingCDN



▼ Transport Layer

- Transport Layer provides a logical communication between processes (not just hosts)
  - Network layer: Host-to-host delivery (IP addresses)
  - Transport layer: Process-to-process delivery (Uses port numbers)



▼ Main Services

- Process-to-process delivery → via port numbers (ex : HTTP = 80, SMTP = 25)
- Reliable data transfer → error detection, retransmission (TCP)
- Flow control → Prevents sender from overwhelming receiver
- Congestion control → Prevents too much traffic in the network
- Multiplexing/Demultiplexing → Allow many processes to share one network connection

▼ Sockets

- Software interface between application layer and transport layer
- An application (like chrome, Gmail) sends/receives data via a socket
- It is identified by a 4-tuple : `source IP, source port, destination IP, destination port`
- For example : Assuming you are connecting to www.google.com and your laptop might be using some port 50000
  then for google → (IP: 142.250.183.100, Port 80)
  and for your laptop → (IP: 192.69.69.69, Port 50000)
  so socket = (192.69.69.69, 50000, 142.250.183.100, 80)

▼ Multiplexing and Demultiplexing

▼ Multiplexing

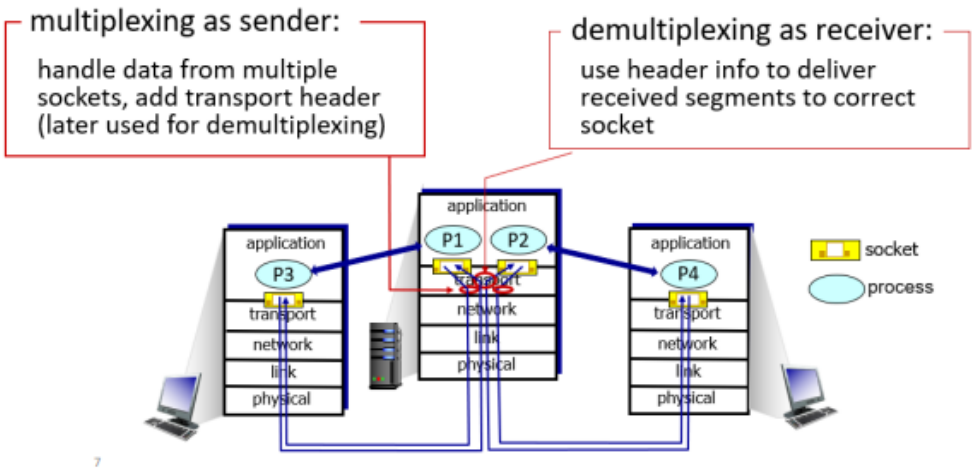
- Many processes share the same network connection
- Transport layer adds headers with port numbers to each segment
- Then all the segments are interleaved and handed down to the network layer which only sees source IP address (host-level delivery)

▼ Demultiplexing

- Network layer delivers all packets to transport layer
- Transport layer uses port numbers in headers to deliver segments to the correct application process

- Example :
  You are browsing (HTTP) and checking email (SMTP) at the same

time. Both use the same IP but different port numbers
Transport layer demultiplexes them and delivers to the right application



▼ Connectionless transport (UDP)

- UDP stands for User Datagram Protocol
- UDP is defined in RFC 768
- It is simple to implement
- The main reason UDP exists is because some applications prefer speed over reliability like DNS, VoIP, video streaming, online gaming
- These apps care about low latency than perfect delivery

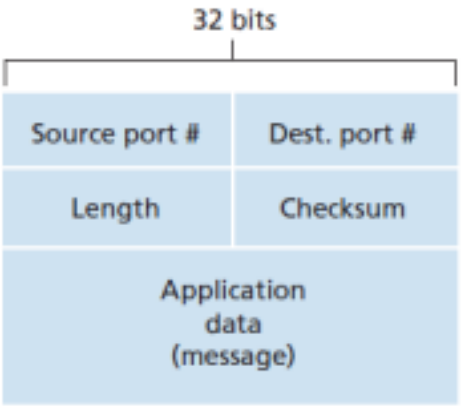
▼ Characteristics

- Connectionless - No handshake before sending
- Unreliable - No guarantee of delivery, ordering or duplicate protection
- Lightweight - Just adds a small header (8 bytes)
- Best effort - IP delivers packets, UDP just adds port numbers for demux

▼ UDP Segment Structure

- The UDP segment is very simple
  - Source port (16 bits) → Identifies sending process
  - Destination port (16 bits) → Identifies receiving process

- Length (16 bits) → Total length of UDP segment (Header + data length)
- Checksum (16 bits) → Error detection (optional)
- Application data (variable size) → The actual message from the application (payload). Could be DNS query, voice packet, video chunk etc.,



▼ Example of UDP in Practice

- DNS → hostname lookups (www.iitm.ac.in) . Need speed
- VoIP → Small voice packets, tolerate name loss (Skype, WhatsApp calls)
- Streaming → better to skip a frame than pause (Live video, Zoom)
- Online gaming → Real-time updates, late packets are useless (PUBG, Valorant)
- DHCP → Quick request/response (IP Address assignment)

▼ Checksum

- Even though IP has its own header checksum, UDP adds another one for end-to-end error detection (across payload too)
- This helps detect bit errors in transmission

▼ Computation of checksum

- UDP takes the entire segment (header + data)
- Then adds it in 16-bit words using 1's complement addition
- Then takes the 1's complement of that sum

- Result = checksum field (16 bits)

▼ Verification

- Receiver does the same calculation across header + data + checksum
- If the result = all 1's , then no error
- If not, error detected and entire segment is discarded

▼ Key notes

- UDP checksum is option (field can be all 0's)
- but most implementations keep it enabled for safety
- Unlike TCP, UDP doesn't retransmit on error, the application must decide what to do

▼ Question

▼ Given the following 16-bit words, find if there is error

```
Word 1: 0110011001100000
Word 2: 0101010101010101
Word 3: 1000111100001100
```

- Step 1 : Add first 2 words
  0110011001100000
  +0101010101010101
  ——————————————-
  1011101110110101
- Step 2 : Add 3rd word
  1011101110110101
  +1000111100001100
  ——————————————
  0100101011000010 (there was carry, so wrap around and add to LSB)
- Step 3 : Take 1's complement of sum
  0100101011000010 → 1011010100111101
- Step 4 : Verification at Receiver
  Receiver adds all words plus checksum

If result = 1111111111111111 , then no error
but here its not the case, so error detected