hosts = end systems = client + server
what is internet? system of million connected computing devices providing communication links within diff media (fiber, copper wire, radio, satellite)
- diff transmission rates (bandwidth (bps) = 10^3 kbps to 100 Mbps to 10 Gbps
- router/switches = forward packets thru
packet switches network
- an infrastructure that enables distributed services/apps (web email, games, file sharing)

units: speed (bps) - kilo(K) = 10^3 = 1000
- mega(m) = 10^6 = 1,000,000
- giga(G) = 10^9 = 1,000,000,000

capacity/size (bytes B) - kilo(K) = 2^{10} = 1024
- mega(M) = 2^{20} = 1048576
- giga(G) = 2^{30} = 1073741824

Time(s) - millim) = 10^{-3} = 0.001
- micro(u) = 10^{-6} = 0.000001

8 bits = 1 Byte

access technologies:

home access	enterprise access	wide area network
DSL	wifi	3G, 4G, 5G, LTE
dialup modem	ethernet	
wifi	telephone line	
cable internet	FTTH (fiber to the home)	
Twisted copper wire	Twisted copper wire	

approaches to moving data thru network:

circuit switching	packet switching
dedicated circuit path per call used by all data	data sent in packets
end-end resources reserved for call b/w source to dest.	resources not reserved; on demand and may have to queue to access comm. links; forwards packets from router to next on path from source to dest.
ex: telephone networks	ex: internet
when network establishes a circuit connection, it reserves a constant R in networks link for duration of connection R = transmission rate	packet sent into network w/o reservation so if one link is congested bc packets are being transmitted at the same time then packet will have to wait in buffer at sending side of trans link and suffer delays
R is reserved for sender-receiver connection; sender can transfer data to receiver at guaranteed constant rate; guaranteed performance	internet will make best effort to deliver packets in timely manner but it cannot make any guarantees; no performance guaranty.
R of 100 Kbps must be reserved for all users; trans capacity of a link is 1 mbps so reserved for 10 users	trans. rate: L/R; packets of L bits onto link at R bps; if 35 users are allowed to send packets of a then how probab. of more than 10 users R be active at the same time
segments sit idle if not used on call; no sharing of resources	link trans capacity is shared on packet-by-packet basis among users
each user is allocated a time slot per frame	allows more than 3 times the number of users

4 sources of delay at each hop along the path from src to dest.
① propagation ② transmission ③ processing ④ queuing

$d_{nop} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$

Trans. delay: Time to transmit packet onto link = L/R
L (size of packet)/R (Trans speed); units in sec(s) or millisec(ms)
- when last bit of packet is transmitted

Prop. delay: time a packet takes to travel across a link = d/s
d (distance from one end to other)/s (prop. speed) = 2.5×10^8 m/s
- units in sec(s) or millisec(ms); when first bit of packet will reach the other end of link; depends on physical medium (fiber, coastal cable, air, etc)

end-end delay: Time it takes for entire packet to reach receiver
= $d_{trans} + d_{prop}$ = L/R + d/s in seconds (adds $d_{proc} + d_{queue}$ if needed)

proc delay: checks for bit errors;

queue delay: Time spent waiting for transmission; depends on congestion
- units in micro(u) or millisec(m); packets can get lost if buffer is full
- if packet arrival rate (La) to link exceeds trans. rate capacity (R) then packets will queue; La/R ~ 0 = avg small queue delay
La/R → 1 = avg large queue delay; La/R > 1 = avg delay infinite

Protocol: defines the format, order of messages sent + received among network entities, action taken on msg trans + receipt.

five layer TCP/IP ① application layer, ② transport, ③ network ④ link ⑤ physical layer

protocol stack: ① link layer, ② physical layer

seven layer OSI reference model: addition of ⑥ presentation + ⑦ session

app layer: used to send data over multiple end systems
- HTTP, FTP, SMTP (email), DNS; referred to as 'message'

transport: host-host data transfer
- TCP, UDP; referred to as 'segment'

network: routing of packets (datagrams) from src. to dest.
- IP protocol, routing protocols; refer to as 'IP layer'

link: moves packet from one node to another in route
- reliable delivery; ethernet, wifi, cable access network DOCSIS protocol (not TCP); referred to as 'frames'

physical: moves individual bits within from from 1 node to next
- link dependent protocols, depends on physical medium

Presentation: allows apps to interpret meaning of data;
- data compression, encryption, description

session: Supports synchronization and delimiting of data exchanged; includes building, checkpoint + recovery scheme

encapsulation: (process from app layer to link layer)

throughput: min {Rs, Rc, R/m}; rate at which hosts exchange bits
bottleneck: link on end-end path that constraints end-end throughput
- link with smallest capacity

Circuit Switching: physical path is obtained and reserved to a single connection bw two endpoints for duration of connection

Socket: 'endpoint' in connection for sending/receiving data bw two programs running on network; used w set of request or 'function calls'

Port: uniquely identify diff apps/processes running on single computer and enable to share a single physical connection to packet-switched network

DNS: domain naming system for computers, services, resources connected to Internet

Proxy: machine to satisfy client HTTP request w/o contacting origin server if nearby clients send their HTTP request to this machine

Trace route: comp. network diagnostic tool for displaying route/path + measuring transit delays of packets across an IP network

Pipelining: sender is allowed to send multiple packets w/o waiting for ACK if network sources send data at constant bit rate, Circuit Switching more desirable because there is no statistical multiplexing gains to be had, by using circuit each connection will get constant amount of bandwidth

App Architecture: designed by app dev who decides how app is structured over end systems. Client-Server Architecture: server that services requests for client

Server: always on host, permanent IP Address; server farms for scaling

Client: comm. w server; may be intermittently connected; dynamic IP addr. do not communicate directly w each other

Data Center: virtual server to keep up with client requests

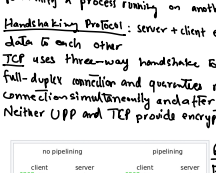
P2P Architecture: min or no reliance on dedicated servers in data center
- peers: laptop, desktops; peers communicate w/o passing thru server; cost effective
- not always on server; arbitrary end systems directly communicate
- peers intermittently connected and change IP address; highly scalable

Hybrid of Client-Server + P2P: Napster: File Transfer P2P
File search centered at
- Peer registry content at central server
- Peer query same central server to locate content

Processes: communication by sending/receiving messages across comp. network
to identify a process running on another host: IP address + port # (↳ 32 bit)

Handshaking Protocol: server + client exchange control info before sending data to each other

TCP uses three-way handshake to create reliable connections; has full-duplex connection and guarantees reliable data transfer over connection simultaneously and after server-client finish, comm. Terminates
Neither UDP and TCP provide encryption; TCP with (SSL) secure socket layer provides encryption, security etc



Pipelining: Sender can send multiple packets without waiting for ACKs
- range of seq. #s must be increased
- sender + receiver may have to buffer more than one packet

Two approaches for pipeline error recovery: Go Back N → selective repeat

Go-Back-N Protocol: link layer protocol that uses sliding window method for reliable + sequential delivery of data frames.
GBN can have performance issues, when window size * bandwidth, delay are large, many packets can be in pipeline
A single packet error can cause GBN to retransmit large number of packets
GBN can allow a max num of packets, it must wait one or more packet to be ACK before proceeding so if packet 2 is lost then rest after one out of order and dropped.

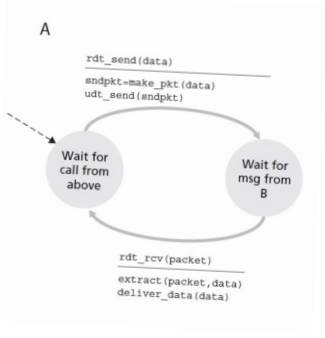
multiplexing & demultiplexing: extending host-to-host delivery service provided by the network layer to process-to-process delivery service for applications running on the host
demultiplexing: the job of delivering the data in a transport layer segment to the correct socket
multiplexing: the job of gathering data chunks at the source host from different sockets, encapsulating each data chunk with header info to create segments and passing those segments to network layer
transport layer in middle host must demultiplex segments arriving from network layer to different processes by directing the arriving segments data to the corresponding process's socket
transport layer in middle must also gather outgoing data from these sockets, form transport layer segments, and pass these segments down to network layer

UDP:
offers unreliable data transfer service, packets sent by client or server could get lost
connectionless services to apps; no frill service that provides no reliability, no flow control, no congestion control
has no congestion control; can result in high loss rate bw sender and receiver
traffic is unregulated; organizations can block udp traffic for security reasons
finer application level control over what data is sent: when application process passes data to udp, it packages the data in segment and quickly passes it to network layer
socket identified by two tuple: dest. ip address and dest. port #
no connection establishment delay to transfer data; dns rather runs over udp
quick udp internet connection (quic) protocol uses udp as transport protocol and implements reliability;
reliability can be built into an app itself
does not maintain connection state and does not track any parameters that tcp does
a server for particular app can typically support more active clients if app runs on udp than tcp
8 bytes of header overhead
udp is used to carry network management data that needs to often in stressed state
developers sometimes choose udp over tcp for multimedia apps and internet phone
remote file server (nfs), network management (snmp), name translation (dns) typically use udp as transport protocol
uses internet checksum for error detection

TCP: offers reliable transfer service, resends segment until receipt of segment is acknowledged by dest. regardless of how long it takes
offers connection oriented service to apps; guaranteed delivery of app layer messages to dest. and flow control
breaks long messages into shorter segments/packets and provides congestion control
tcp congestion control can prevent any tcp connection from swamping links/routers bw communicating host during excessive traffic; perfect for streaming media
transport when udp is blocked
congestion control mechanism over transport layer tcp sender when one or more links bw source and dest. become congested
socket identified by four-tuple: source ip address, source port #, dest. ip address, dest. port #
has connection establishment delay before transferring data; dns would be slower on tcp
reliability is important for http web pages with text so http uses tcp and web pages load slower
maintains connection state in end systems
connection state includes: receive/send buffers, congestion control, sequence/acknowledgement number parameters
20 bytes of header overhead
congestion controlled data transfer can be difficulty to achieve for network management data
real time apps such as internet phone, video conferencing poorly react to congestion control; sometimes developers use tcp over udp for streaming media and internet phone depending on transmission constraints
electronic mail (smtp), remote terminal access (telnet), web (http), file transfer (ftp) use tcp as transfer protocol
transport layer multiplexing requires source port number field (unique identifiers) and destination port number field (special fields that indicates socket to which segment is to be delivered)
transport layer demultiplexing: socket in host must be assigned port #s, when segments arrives from network to host, transport layer checks dest. port # (4 for tcp, 2 for udp) and directs segment to appropriate socket
each port number = 16 bit # (0-65535)
well known port #s: from 0-1023, restricted, reserved for use by HTTP (80) and FTP (21)

web caching:

Reliable Data Transfer:
unidirectional data transfer: data transfer in only one direction from sender to receiver
bidirectional: data transfer in both directions
all transmitted packets are received in order in which they were sent
reliable data transfer over reliable channel rdt1.0:
there are separate finite state machine (fsm) for sender and receiver
-sending side accepts data from upper layer and makes packet with data and sends packet into channel
    -rdt_send(data); packet = make_pkt(data);
udt_send(packet
-receiving side receives a packet from underlying channel, extracts data from packet and delivers data to upper layer
    -rdt_rcv(packet); extract(packet, data);
deliver_data(data)
in this protocol all packet flow is from sender to receiver, and not vice versa
reliable data transfer over channel with some errors
rdt2.0: also known as stop-and-wait protocol
- sending side has two states: (1) waits for data to be passed down and (2) wait for ACK/NAK from receiver (page 249)
    - when sender is in state of waiting for ACK/NAK, it cannot get more data from upper layer
- receiving side responds with ACK or NAK on packet arrival
    - if ACK, sender protocol goes in state of (1) waiting for data to be passed down from upper layer
    - if NAK, sender protocol retransmits the last packet to receiver and waits for feedback from receiver
Automatic Repeat Request (ARQ)s an error-control mechanism for data transmission which uses positive (ACK) or negative (NAK) acknowledgements and timeouts to achieve reliable data transmission over an unreliable communication link
- three additional protocols in ARQ to handle errors: error detection, receiver feedback, retransmission
    - feedback: ACK and NAK are one bit long - NAK: 0, ACK: 1
    - retransmission: packet w error at receiver is retransmitted by sender
- three approaches for corrupted receiver feedback:
    - corrupted ACK/NAK message dictation bw sender and receiver
    - add enough checksums to let sender detect errors and recover
    - sender resends current data packet but this time with a unique sequence number for each packet
- protocol rdt2.1: uses ACK/NAK feedback, if out-of-order packet is received then ACK, if corrupted packet is received then NAK
    - if sender gets two ACKs for same packet, sender will know that the receiver did not receive packet correctly
- protocol rdt2.2: checksum, ACK packets, retransmission, seq nums, etc
    - receiver must include sequence number of packet being acknowledged by including ACK, 0 or ACK, 1
    - sender must check the sequence number of packet being acknowledged by including 0 or 1 in isACK() argument
- **reliable data transfer over lossy channel with some errors rdt3.0:** channel can lose packets
- concerns to address: how to detect packet loss and what to do?
sender does not know if packet is lost, ACK is lost, or either is simply delayed: sender implements time based retransmission mechanism so timer will timeout/expire and sender can (1) start a timer for packets being sent or resent (2) respond to timer interruption (3) stop the timer
if packet is having large delay when transmitting (not lost), then as in rdt2.2, receiver will assign seq num to handle duplicate packets



Total amount of time to get IP add: $RTT_1 + RTT_2 + \dots RTT_n$
once IP add is known, RTT_0 elapses to set up TCP, another RTT_0 elapses to request and receive object: $2RTT_0 + RTT_1 + RTT_2 + \dots RTT_n$
Non-persistent HTTP w. no parallel TCP (w. 8 small obj):
$2RTT_0 + RTT_1 + RTT_2 + \dots RTT_n + 8 \times 2 \times RTT_0$
Persistent HTTP conn. w. pipelining: $RTT_1 + \dots + RTT_n + 2RTT_0 + RTT_0$
default mode of HTTP.
Persistent HTTP conn. w/o pipelining: $RTT_1 + \dots + RTT_n + 2RTT_0 + 8RTT_0$
w/o parallel connections

access network - network that physically connects an end system to the first router on a path from from the end system to any other end system
home access: DSL (digital subscriber line), cable
access in enterprise: ethernet and wifi
LAN (local area network) as access network in corporate/university campuses
wide-area wireless access: 3G and LTE
in packet switching, resources for communication are not reserved and are rather used on first come first serve basis which means that some message may have to wait
In circuit switching, the resources needed to provide communication between two end systems are reserved for the entire of the communication session. Anytime two hosts want to communicate, the network creates a dedicated end-to-end connection.
store and forward transmission
frequency-division multiplexing
time-division multiplexing
circuit switching could be wasteful once the link is no longer in use.
packing switching can be more efficient and simpler to implement and offers better sharing of the link capacity. However, it can be bad for real time services because of its unpredictable end-to-end delays.
$L / R \rightarrow (N + P - 1) L / R$
types of delays
processing delay = time required to examine packet's header and determine where to direct it
queuing delay
the time a packet may wait before getting transmitted onto the link
transmission delay
time required to push or transmit all of the packet into the link
L/R - packet length / transmission rate
propagation delay
time required for the packet to traverse the link from one end system to another
d / s - distance between end systems / propagation speed
traffic intensity La / R
a is the average rate at which packets arrive at the queue
> 1 means rate at which bits arrive is greater than rate at which bits can be transmitted from the queue
throughput
instantaneous - rate at which B is receiving the file at any instant
average - F / T - file length / time for B to receive all bits
rate at which sending process can deliver bits to receiving process
Rs if Rs < Rc
Rc if Rc < Rs
bottleneck link - min {Rc, Rs}
protocol layering
application - transport - network - link - physical
application - network applications and their application-layer protocols
application in one end uses these protocols to send packets to application in another end system
transport - transports application layer messages between application end points
TCP and UDP
TCP - connection oriented, guaranteed delivery, congestion control
UDP - connectionless, no reliability, no flow control, no congestion control
segments
network
moving network layer packets as datagrams between two hosts
datagrams
link layer
routes a datagram through series of routers between source and destination
frames
physical layer
move individual bits within frame from one node to the next
link dependent and depend on actual transmission medium of link
encapsulation
each layer adds header information when sending and extracts a header when receiving
application-layer message - packet of information with application in one end system to another application
transport segments - transport-layer packet with header
network datagram - network-layer packets with header
link-frame - link-layer packets with header

Application
socket - software interface through which a process sends messages into and receives messages from the network
interface between application layer and transport layer within a host
also referred as API between application and network
need address of host and identifier that specifies the receiving process in the host
host identified by IP address - 32 bit quantity unique to hosts
port number to specify the process in the host
reliable data transfer - guaranteed data delivery
unreliable data transfer may be acceptable for loss tolerant applications
application with throughput requirements are bandwidth sensitive applications, elastic are otherwise
TCP - client and server exchange transport-layer control information with each other first before messages are sent. This is called handshaking which alerts the client and server allowing them to prepare for packets to come
after the handshaking, a TCP connection is said to exist between the sockets
full duplex connection so both processes can send messages at the same time
UDP - connectionless so no handshaking, no reliable data transfer
A handshaking protocol is used by client and server to exchange transport layer control information with each other before sending messages. This establishes the TCP connection.
HTTP uses TCP so there is no concern of lost data or recovering from loss or the reordering of data within the network
HTTP is stateless because it maintains no information about the clients
non-persistent - series of requests can be made back to back over the same TCP connection
sending base file with 10 objects - 11 TCP connections
connection must be maintained, each connection has buffers and variables which could be bad
persistent - requests over separate connections
HTTP uses persistent as default
round-trip time RTT - time it takes for a small packet to travel from client to serve and back to client
web caching - network entity that satisfies HTTP requests on behalf of an origin web server

client establishes TCP connection with web cache and sends HTTP request for the object
web cache checks for a local copy of the object and returns to the client if there is one
otherwise, web cache opens TCP connection to the origin server and request for the object
origin server sends object to web cache
web cache stores a copy and sends a copy to client over existing TCP connection
web caching reduces response time especially if bottleneck bandwidth between client and origin server is less than bottleneck between client anc cache
cache can deliver object rapidly to client if it has a copy
web caching can reduce traffic
DNS - directory serve that translates hostnames to IP addresses
domain name system
process
suser machine runs client side of DNS
client extracts host name from URL and passes hostname to client side of DNS
DNS client sends a query containing hostname to DNS server
DNS client receives a reply with the IP address of the host