CS436

Austin Xia

April 1, 2022

Contents

List of Figures

List of Tables

1 internet introduction

billions of connected computint devices
host = end system
network apps is running at internet's **edge**
Packet switches: forward packets (chunks of data), routers/switches
Communication links: fiber/copper,radio, satellite; transmission rate: bandwidth
networks: collection of devices, routers, links managed by an organization

1.1 protocols

protocols control sending/receiving of messages
Internet standards: RFC, IETF(Internet Engineering Task Force)
all communication activity in Internet governed by protocols
it's like human saying hello

2 network edge

in edge we have end system/host, client/servers
servers often in data center
host hosts internet applications, clients and servers

2.1 access network

wired/wireless communication links.
computer use access network to connect to network

- cable-based access: frequency division multiplexing (FDM): different channels transmitted in different frequency bands
- HFC: hybrid fiber coax
- digital subscriber line (DSL): use existing telephone line
- Home network: access point, router.firewall.NAT, cable/DSL modem combined, and to/from head-end/central office
- wireless access networks: shared wireless access network coneects end system to router: via base station aka access point
- enterprise networks: mix of wired/wrireless
- datacenter networks: high-bandwidth links, connect thousands of servers together and to internet

2.2 host: send packets of data

- take application message
- breaks into chunks, **packets** of length **L**
- transmit packet into access network at **transmission rate R**. link transmission rate === **capacity/link bandwidth**

packet transmission delay = time needed to transmit L-bit packet into link = $\frac{L}{R}$

2.3 links: physical layers

- bit: propogates between transmitter/receivers pairs
- physical link: lies between transmitter and receivers
- guided media: signals propagate in solid media: copper/fiber/coax
- unguided media: propagate freely like radio
- Twisted pair(TP): two insulated copper wires.
- coaxial cable: 2 concentric copper conductors, bidirectional, broadband
- fiber optic cable: glass fiber carrying light pulses, each pulse a bit, highspeed, low-error-rate
- wireless radio: environment effect (reflection/obstruction/interference); signals carried in various bands in electromagnetic spectrum
  radio link types: wifi, bluetooth, terrestrial microwave (point-to-point), satellite

3 network core

network core is a mesh of interconnected routers **packet-switching**: host break application-layer messages into packets
network forwards packet from one router to next. across links on path from source to destination

3.1 two key netowrk-core functions

- forwarding: aka switching, move arriving packets from router's input link to output link
- routing: global action: determine source-destination path taken by packets. there is routing algorithms

3.2 packet-switching: store-and-forward

packet transmission delay: takes L/R seconds to transmit (push out) L-bit packet into link at R bps
store and forward: entire packet must arrive at router begore it can be transmitted to next link

3.3 packet-switching: queueing

packet queuing and loss:
queueing occurs when work arrives faster than it can be serviced
packets will be queueing, waiting to be transmitted, packets can be dropped(lost) if memory (buffer) in router fills up

3.4 circuit switching: alternative to packet switching

end-end recources allocated to, reserved for 'call' between source and destination.

- no sharing

- idle if not used
- used in traditional telephone networks
- FDM frequency division multiplexing: optical electromagnetic frequency divided into frequency bands, each cell allocate its own band
- TDM time division multiplexing: time divided into slots. each cell allocated periodic slots. transmit at maxium rate only during its time slots

3.5 packet switching vs circuit switching

circuit-switching support fixed number of users.
packet switching support more users with small probablity of lossing packages

3.6 internet structure: network of network

- host connect to internet via access ISPs
- acess ISPs are interconnected so any 2 host can send packets to each other
- resulting network of networks is complex and is driven by economics/national-policies

access net **may be** connected to regional ISP, regional ISP connect to ISP are interconnected via IXP (internet exchange point)
at center is well connected networks like **tire-1 commercial ISP like AT&T** or **content provider network** (like facebook) that connect its data centers to internet, bypassing tier-1, regional ISP

3.7 performance: loss, delay, throughput

packet delay/loss occurs when packets **queue** in router buffers. loss when memory to hold queued packets fills up
transmission delay and queueing delay
packet delay = $d_{nodal} = d_{proccess} + d_{queue} + d_{trans} + d_{propagate/travel}$
d_{proc}: nodal proccessing

- check bit errors
- determine output link
- typecially ¡ microsecs

transmission delay is packet-length/transmission-rate propogate delay is length of pysical link / proprogate speed
queueing delay

- a: average packet arrival rate
- L: package length (bit)
- R: link bandwidth (bit transmission rate)
- traffic-intensity: $\frac{La}{R}$ arrivial rate / service rate

  - ≈ 0 small delay
  - $\rightarrow 1$ large delay
  - > 1 average infinite delay

real internet delay: traceroute: 1st packet reach 1st probe, 2nd receive 2nd probe
throughput:
rate (bit/time) at which bits are sent form sender to receiver

- instananious
- average

throughput is upperbounded with smallest throughput link

3.8 security

internet not designed with much security
packet sniffing:

- broadcase media
- read/records all packets passing by

denial of service (DOS): attackers make resource (server, bandwidth) unavailable to legitmate traffic by overwhelming recource with bogus traffic

- select target
- break into host around the network (botnet)
- send packets to target from compromised hosts

controled hosts are bots. bad guy in control is botmaster
lines of defense

- authentication: cellular networks provide hardware identity via SIM card. no such hardware exisit in traditional internet
- confidentiality: via encryption
- integrity checks: digital signature prevent/detect tampering. sigh the payload such that none can imitate the signature
- access restrictions: password-protected VPNs. create another layer of network with limited access
- firewalls: specialized middleboxes in access and core networks
  - off-by-default: filter coming packets to restrict senders, receivers, applications
  - tecting/reacting to DOS attack
-

3.9 why layering

approach to designing/discussing complex systems

- explicit structure allows identification, relationship of system's pieces: layered reference model
- modularization eases maintenance, updating of system: change in layer's service implementation: transparent to rest of system
- application: support network applications: http support web, imap support mail, DNS support all application
- transport: process-process data transfer: TCP, UDP, hides all hardware/software underneath
- network: routing of datagram from source to destination: IP, routing protocols
- link: data transfer between neighbouring network elements. transmitter-receiver. end point of same physical medium. from one hop to next. like Ethernet, wifi, PPP
- pysical: pushing bits on the wire

3.10 stuff

first router is edge router, it get things into internet
access networks: digital subscriber line (DSL): use telephone line to central office DSLAM
data goes to internet, voice goes to telephone
access networks: home networks
cable/DSL modem to router/firewall/NAT to wifi wireless access point
this is combined into a single box

4 application layer

4.1 principles of network applications

some network apps: wocial networking, web, text message, e-mail, streaming
creating a network app:
write programs that run on different machines
no need to write software for network-core devices
cline-server paradigm server:

- always on
- permanent IP address
- often in data center, for scaling

client:

- contact, communicate with server
- may be intermittently connected
- may have dynamic IP address

- do not communicate directly with each other

peer-peer architecture

- no always-on server
- arbitrary end system directly communicate
- peers request service from other peers, provide service in return to other peers (so new peers bring new service capacity, as well as new sercie demands)
- peers are intermittently connected and change IP address. (so requires complex management)

process communicating process: program running within a host

- client process: process that initiates communication
- server process: process that waits to be connected
- in P2P, client and server run on same machine

socket process sends/receives messages to/from its socket
sending socket/ receiving socket
addressing processes to receive messages, process must have identifier
host device has unique 32-bit ipAddress
identifier includes both IP address and **port number**
application-layer protocol defines

- types of messages exchanged. (request/response)
- message syntax: what fields in messages, how they delineated
- message semantics: meaning of info in fields
- rules: when/how process send/response to messages

open protocols

- defined in RFC, every has access to protocol definition
- anyone can writes

app needs

- reliability (no loss/loss-tolerent)
- timing
- throughput

internet transport protocols services

- TCP:
  - reliable transport between sending and receiving protocols,
  - flow control: sender won't overwhelm receiver.
  - cogestion control:throttle sender when network overloaded

  - connection-oriented: setup required between client and server processes
  - does not provide: timing, minium throughput guarantee, security
- UDP:
  - unreliable data transfer between sending receiving process
  - does not provide: reliability, flow control, congestion control, timing, throughput guarantee, security, connection setup
- TLS transport layer security: provides encryped TCP connections, data integrity, end-point authentication

note: you can implement security in application layer, even transport layer no security

4.2 web and http

web and http

- web page consists of objects
- objects can be html file, image etc
- web page consists of base html-file, which includes several referenced objects, each addressable by url

http:

- web's application-layer protocol
- client: brower that requests receives (using http protocol) and displays web object; server: web server sends (using http protocol) objects in response to request
- use TCP, client intiates TCP connection to server
- server accepts tcp connection
- http messages (application-layer protocol messages) exhanged between browser (http client) and web server (http server)
- tcp closed
- http is statelss: server maintains no info about past client requests

non-persistent http vs persistent http non-persistent:

- tcp connection opened
- at most 1 object sent over tcp connection
- tcp connection closed

persistent http:

- tcp connection opened
- multiple objects sent in one connection
- tcp connection closed

example

- 1a client initiate tcp connection at port 80
- 1b server waiting for tcp connction at port 80 accepts connection
- 2 client send http request message
- 3 server receives message, send stuff
- 4 http server close tcp connection
- 5 client receive response containing html file, parse html finds 10 referenced jpg file 5
- 6 repeat 1-5 10 times

thus 11 objects in total, 2 RTT per object + transmission time.
non-persistent http: response time RTT: round trip time: time for a small packet to travel from clinet to server and back
http response time (per object):

- one RTT to initiate tcp connection
- one rtt for http request and first few bytes of http respone to return
- object/file transmission time

issues:

- 2 rtt per objects
- os overhead for each tcp connection
- browsers often open multiple parallel tcp connections to fetch referenced objects in parallel

persistent http(http 1.1):

- server leaves connection open after sending response
- subsequent http messages between same client/server sent over open connection
- clinet sends requests as soon as it encouters a referenced object
- as little as 1 rtt for all reference objects

http request message

- 2 types of http messages: request, response
- http request message: ASCII

http1.1 introduced multiple, pipelined GETs over single TCP connection

- server responds in-order (first come first server) to get requests
- with fcfs, first object may block other objects
- loss recovery retransmitting lost tcp segments stalls objects transmission

http/2: key goal: decreased delay in multi-object http requests
increase flixbility at server in sending objects to clinets

- methods, status code, most header fields unchanged
- transmission order of requested objects based on client-specified objects priority
- push unreuqested objects to clients
- divide objects into frames, schedule frames to mitigate HOL blocking

http/2 to http/3 http/2 over tcp connection means:

- recovery form packet loss still stalls all object transmission: browers have incentive to parallel tcp connection to reduece stalling
- no security
- HTTP/3 adds security, per object error and congestion-control over UDP

maintaining user/server state: cookies cookies has 4 components:

- cookie header line of http repsonse message
- cookie header line in next http request message
- cookie file kept on user's host managed by user's browser
- back-end database at web site

cookies can be used for

- authorization
- shopping cart
- recommendation
- user session state (web email)

web caches goal: satisfy clinet requests without involving origin server
user configure browser to point to a local Web Cache
browser sends all http request to cache, if object in cache, cache returns object to client
web cache act as server for original requesting client, client to origin server
server tells cache object's allowable caching in response header

4.3 email smtp imap

- user agents
- mail servers
- simple mail transfer protocol: SMTP

user agent:

- mail reader

- composing editing reading mail messages
- outgoing incoming messages stored on server

mail server:

- mailbox contains incoming messages for user
- message queue of outgoing (to be sent) mail messages
- SMTP protocol between mail servers to send email messages

SMTP RFC

- use tcp to reliably transfer email from client(mail server initiating connection) to server

4.4 dns domain name system

- distributed database implemented in hierarchy of many name servers
- application-layer protocol (complexity at edge, not core)
- services:
  - hostname-to-IP-address translation
  - host aliasing
  - mail server aliasing
  - load distribution

root dns, top level domain (.edu, .org DNS serverse), authorizative(yahoo.com dns servers)
example: to find www.amazon.com, client queries root server to find .com dns server, queries .com dns server to find amazon.com DNS server, query amazon.com DNS server to get IP address for www.amazon.com

4.4.1 first layer

dns: root name servers

- official, contact of last resort by name servers that cannot resolve name
- ICANN internet corporation for assigned names and numbers manages root DNS domain
- 13 logical root name servers worldside each server replicated many times

4.4.2 second layer

top level domain, and authoritative servers responsible for .com, .org .ca, .cn etc.
network solutions: authoritative registry for .com, .net TLD
Educase: .edu TLD

4.4.3 third layer

authoritative DNS servers organization's own server providing hostname to IP mapping for organization's named host
can be maintained by organization or service provider
DNS name resolution

- iterated query: contacted server replies with name of server to contact
- recursive query: puts burden of name resolution on contacted name server

dns records, RR distributed database storing resource record (RR)
RR format:

- type=A name is hostname value is ipAddress
- type=CNAME, name is alias for real name, value is canonical name (www.ibm.com is really servereast.backup2.ibm.com)
- type=NS, name is domain (foo.com) value is hostname of authoritative name server for this domain
- type=MX, value is name of SMTP mail server with name

DNS query and reply message have same format
header: indentification 16 bit, flags: query or reply, recursion desired, recursion available, reply is authoritative
questions: name type fields for a query
answers: RRs in response to query
authority: records for authoritative servers
addition info:

4.5 p2p app

- no always-on server
- arbitrary end system directly communicate
- peer request service from other peers, provide service in return to other peers
- intermittently connected and change IP address
- example: bit torrent

using client server, time to distribute F to N clients is

$$D_{c-s} \geq max\{NF/u_s, F/d_{min}\}$$

for p2p,

$$D_{P2P} \geq max\{F/u_s, F/d_{min}, NF/(u_s + \sum u_i)\}$$

bitTorrent
files divided into 256KB chunks
peers in swarm send receive file chunks
tracker tracks peers participating in swarm
swarm group of users exchanging files
churn peers may come and go

4.6 video streaming and content distribution

dash server:

- divides file into multiple chunks
- each chunk encoded at multiple different rates
- different rate encodings stored in different files
- files replicated in various CDN nodes
- **manifest file**: privides URL for different chunks

client:

- periodically estimates server-to-client bandwidth
- consulting manifest, requesting one chunk at a time (choose rate from different server)

client intelligence: when, what encoding rate, where
steaming video = encoding + DASH + playout buffering
CDN content distribution networks

- enter deep: push CDN servers into many access networks close to users
- bring home: smaller number of large clusters in POPs near access nets

CDN stores copies of content at CDN nodes
returns manifest to subscriber
OTT: over the top, internet host-host communication as a service, use dash and internet instead of directly streaming

5 transport layer

5.1 transport layer services

- provide logical communication between application processes running on different host
- transport protocols actions in end systems
  - sender: break applciation messages into segments, passes to network layer
  - receiver: reassembers segments into messages, passes to application layer
- 2 transport layer protocols available: TCP and UDP

network layer: logical communication between hosts
transport layer: logical communication between **processes**
actions:
sender:

- is passed application layer message
- determines segment header fields value

- creates segment
- passes segment to IP

receiver:

- receives segment from IP
- check header value
- extracts applciation-layer messages
- demultiplexes message up to application via socket

TCP:

- reliable, in order
- congestion control
- flow control
- connection setup

UDP:

- unreliable unordered
- no-frills extension of best-effort IP

5.2 multiplexing and demultiplexing

demultiplexing at receiver: user header info to deliver received segments to correct socket
multiplexing at sender: handle data from multiple sockets, add transport header
demultiplexing work by:
host receives **IP datagram**, which has source IP and destination IP
each datagram carries one **transport-layer segment**, each segment has source destination port number
host uses IP addresses and port numbers to direct segment to appropriate socket

5.2.1 connectionless demultiplexing vs connection-oriented demultiplexing

tcp socket identified by

- source IP
- source port
- dest IP
- dest port

demux: receiver uses all 4 value to direct segment to appropriate socket
where as UDP doesn't care about source IP and source port

5.3 connectionless transport UDP

segments may be lost or delivered out of order
conectionless: no handshaking between sender/receiver; each segment handled independently of others
we can add flow/congestion control or reliability at application layer
UDP checksum: detect errors
got 2 port number (16 bit int)
wraparound(addition), then sum(add 1 if wraparound is 17 bit), then checksum (flip bit in sum)

5.4 connection oriented transport TCP

- point-to-point (one sender one receiver)
- reliable, in-order (no message boundaries)
- full duplex data: bi-directional data flow in same connection
- cumulative ACKs
- pipelining (TCP congestion and flow ontrol set window size)
- connection oriented: handshaking initializes sender, receiver state before exchange
- flow controlled: sender wont overwhelm receiver

5.4.1 tcp segment structure

- sequence number: counting bytes of data into bytestream (not segment)
- acknowledge number: seq number of next expected byte
- receive window: bytes receiver willing to accept
- (TCP) options and applicaiton data are of variable length
- C,E are congestion notification
- RSF: RST, SYN, FIN: connection management
- there is also header length and checksum

tcp sender:

- event: data received from application
  - create segment with seq #
  - seq # is bytestream number of first data byte in segment
  - start timer if not already running (think of timer as for oldest unACKed segment), expiration interval: timeoutInterval
- event: timeout
  - retransmit segment that caused timeout
  - restart timer

- event: ACK received
  - if ACK acknowledges previously unACKed segments: updates what is known to be ACKed
  - start timer if there are still unACKed segments

TCP fast retransmit:
if sender receives 3 additional ACKs for same data, resend unACKed segment with smallest seq#, because most likely, unACKed segment is lost

5.5 tcp congestion control

tcp sender limits transmission: lastByteSent-lastByteAcked $\leq$ cwnd
cwnd is dynamically adjusted
tcp rate $\approx \frac{cwnd}{RTT}$ bytes/sec
tcp slow start: initial cwnd = 1 MSS
double cwnd each RTT

6 network layer

- transport segment from sending to receiving host
- on sending side encapsulates segments into datagrams
- on receiving side, delivers xegments to transport layer
- network layer protocols in **every** host, router
- router examines header fields in all IP datagrams passing through it

forwarding: move packets from router's input to appropriate router output
routing: determine route taken by packets from source to dest
local forwarding table: determines local forwarding (header value 0100 into output link 3)
this uses longest prefix matching
IP address: 32 bits
interface: connection between host/router and physical link
routers typically has multiple interfaces, hosts typically has 1 or 2 interfaces, each interface has an IP address

6.1 CIDR

CIDR: classless interdomain routing
a.b.c.d/x where x is number of bits in subnet portion of address
200.23.16.0/23 means 23 bits fixed, 32-23 bits free
subnet part - high order bits,
host part - low order bits
what is a subnet? device interfaces with same subnet part of IP address, can physically reach each other without intervening router

6.2 dhcp

dhcp dynamic host configuration protocol. dynamically get address from server
goal: allow host dynamically obtain its IP address from network server when it joins network

- can renew its lease on address in use
- allows reuse of address
- support for mobile users who want to join network

overview:

- host broadcases "DHCP discover" msg [optional]
- dhcp server responds with "DHCP offer" msg [optional]
- host requests IP address: "DHCP request" msg
- DHCP server sends address "DHCP ack" msg

DHCP can return more than just allocated IP address on subnet

- address of first-hop router for client
- name and IP address of DNS server
- network mask (indicating network versus host portion of address)

how does network get subnet part of IP addr? gets allocated portion of its provider ISP's address space
how does ISP get block of address? from ICANN (internet corportation for assigned names and numbers)

6.3 NAT

NAT: network address translation
datagrams with source or destination in this network has 10.0.0/24 address for source, destination as well
datagrams leaving local network have same single source NAT IP address 138.76.29.7 different source port number
local network uses just 1 IP address as for as outside world is concerned

- range of address not needed from ISP, just 1 address for all devices
- can change addresses of devices in local network without notifying outside world
- can change ISP withou changing addresses of devices in local network
- devices inside local net not explicitly addressable

implementation: NAT router

- outgoing datagrams: replace source IP address, port number to NAT IP address, new port number
- remember every source IP address, port number to NAT IP address, new port number
- incoming datagrams: replace NAT IP address, new port number to corresponding stuff stored in NAT table

port number is 16 bit long
NAT is controversal

- routers should only process up to layer 3
- violates end-to-end argument (P2P must consider NAT)
- address shortage should be solved by IPv6

6.4 IP gragmentation reassembly

network links have MTU (max transfer size), largest possible link-level frame
large IP datagram divided within net

- one datagram becomes several datagrams
- reassembled only at final destination
- IP header bits used to identify order related

fragflag: if 1, more fragment coming
offset: what bit is it now

7 routing

routing algorithms:

7.1 dijsktra

Initialization: $N = \{u\}$ for all nodes v if v adjacent to u, $d(v) = c(u, v)$
else $D(v) = \infty$
Loop:
find w not in N st D(w) is a minimum
add w to N
updated D(v) for all v adjacent to w, not in N

$$D(v) = min(D(v), D(w)_c(w, v))$$

until all nodes are in N

7.2 distance vector altorithm

each node knows cost to each neighbor, and maintains its neighbors distance vectors
when x receives new DV estimate from neighbor, it udpates its own dv using
$D_x(y) \rightarrow min_v\{c(x, v) + D_v(y)\}$
administrative autonomy:
internet=networks of networks,
each network admin may want to control routing in its own network

7.3 hierarchical routing

aggregate routers into regions "autonomous system" (AS)
routers in same AS run same routing protocol
gateway router: at edge of its own AS, has link to router in another AS
forwarding table configured by both intra-and inter- AS routing algorithms

8 link layer

hosts, routers are **nodes**
communication channels that connect adjacent nodes along communication path: links
there is wired links, wireless links, and LANs
layer-2 packet: frame, encapsulates datagram
data-link layer has responsibility of transferring datagram from one node to physically adjacent node over a link
datagram transferred by different link protocols over different links (ethernet, then frame relay, then 802.11)

8.1 link layer services

- framing, link access
  - encapsulate datagram into frame, adding header/tailer
  - channel access if shared medium
  - mac address used in frame headers to identify source/destination
- reliable dilivery between adjacent nodes
  - like in transport layer
  - seldom used on low bit-error link
  - wireless link has high error rates
- error detection
  - errors caused by signal attenuation, noise
  - receiver detects presence of errors (signals sender for retransmission or drops frame)
- error correction: receiver identifies and corrects bit error without resorting to retransmission

link layer is implemented in

- each and every host
- in adaptor/NIC or on a chip
- attaches into host's system buses
- combination of hardware software, firmware

adaptors communicating
sending side:

- encapsulates datagram in frame

- adds error checking bits, rdt, flow control, etc

receiving side

- looks for errors, rdt, flow control, etc
- extracts datagram, passes to upper layer at receiving side

multiple access protocols

- single shared broadcast channel
- ≥ 2 simutaneous transmissions by nodes: interference, collision if nodes receives ≥ 2 signals

distributed algorithm that determines how nodes share channel

8.2 mac protocols: taxonomy

3 broad classes:

- channel partitioning (divide channel into smaller pieces, allocate peice to ndoe for exclusive use)
- random access (channel not divided, allow collision and recover from it)
- taking turns (take turns but nodes with more to send take longer turns)

TDMA time division multiple access
FDMA
random access protocols: CSMA
carrier sense multiple access (CSMA): listen before transmit, if idle, transmit entire frame, else defer
carrier sense multiple access, collision detection (CSMA/CD): listen before transmit, if idle, transmit, else defer
after aborting, nic enters exponential backoff: after mth collision, nIC chooses K at random from $\{0, 1, 2...\}$ NIC waits $K * 512$ bit times
polling (take turns)
master invites slave nodes to transmit in turn, typically used with dumb slave devices
concern: polling overhead, latency, single point of failure(master)
token passing
control token passed from one node to next sequentially
there is token message
concern: token overhead, latency, single point of failure(token)

8.3 LAN addresses and ARP

LAN address is MAC address, is allocated by IEEE
manufacturer buys MAC address, like social security number, IP address is postal code
MAC flat address gets portability, can move LAN card from one LAN to another
ARP table: each IP node (host, router) on LAN has table, it has IP/MAC address mappings for some LAN nodes <IP address; MAC address; TTL>
ARP protocol: same LAN A wants to send datagram to B
B's MAC address not in A's ARP table

A boradcast ARP query packet, containing B's IP address (dest MAC address = FF-FF-FF-FF-FF-FF), all nodes on LAN receive ARP query

B receives ARP packet, replies to A with its MAC address (sent to A's MAC address)

A caches IP-to-MAC mapping in its ARP until timeout. soft state: information that times out unless refreshed

ARP is plug and play: nodes create their ARP tables without intervention from net administrator

8.4 addressing: routing to another LAN

send datagram from A to B via R

- focus on addressing - at IP and MAC layer
- assume A knows B IP address
- asuume A knows IP address of first hop router R
- assume A knows R's MAC address

A creates IP datagram with IP source A destination B

A creates link layer frame with R's MAC address as dest, frame contains A-B IP datagram

frame sent A to R, received at R, datagram removed, passed up to IP

R forwards datagram with IP source A destination B

R creates link-layer frame with B's MAC address as dest, frame contains A-B IP datagram

ethernet physical topology:

bus: all nodes in same collision domain (all devices connected to one cable)

start: active switch in center, each spoke runs a separate ethernet protocol (nodes do not collide)

8.5 ethernet frame structure

sending adapter encapsulate IP datagram in ethernet frame

preamble, destMacAddr, sourceAddr, type, data, CRC

preamble is 7 bytes used to synchronize receiver, sender clock rates

type is usually IP, could be AppleTalk

CRC cyclic redundancy check at receiver

8.6 ethernet: unreliable connectionless

connectionless: no handshaking between NIC

unreliable: no ack

uses CSMA/CD

8.7 ethernet switch

link-layer device: takes an active role

store, forward ethernet frames

examine encoming mac address, selectively forward frame to one-or-more outgoing links when frame is to be forwarded on segment, uses CSMA/CD to access segment

transparent: host are unaware of presense of switches

plug-and-play: do not need to be configured

8.8 switch: multiple simultaneous transmission

hosts have dedicated, direct connection to switch
switch buffer packets
ethernet protocol used on each incoming link, but no collisions; full duplex each link is its own collision domain

8.9 switch forwarding table

each switch has a swtich table like a routing table
switch learns which host through which interface

- record incoming link with mac address of sending host
- index switch table using MAC destination address
- if entry found: if destination on segment from which frame arrive, drop frame else forward frame on interface indicated by entry else: flood (forward all interfaces except arriving interface)

8.10 switch vs router

both store-and-forward

- routers: network-layer device (examine network layer header)
- swtich: link-layer device (examine link layer header)

both have forwarding table

- routers: compute using routing algorithm, IP addresses
- switches: learn forwarding using flooding learning MAC address

8.11 a day in the life

- connecting laptop needs to get their IP address, addr of first-hop router, of DNS server: use **DHCP**
- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.3 ethernet,
- ether frame braodcast (dest:FFFFFFFFFFFF) on LAN, received at router running DHCP server
- thernet demuxed to IP demuxed, UDP dumuxed to DHCP (de-multiplexing)
- DHCP server formulates DHCP ACK containing client IP address, IP address of first-hop router
- encapsulation at DHCP server, frame forwarded (switch learning) through LAN, demultiplexing at client
- DHCP client receives DHCP ack reply
- **client now knows IP address, name,address of DNS server, IP address of first hop router**
- before sending HTTP request, need IP address of google
- DNS query created, encapsulated in IP, in Ethernet. to send frame to router, need MAC address of router interfaceL (ARP)

- ARP query broadcast, received by router, which replies with ARP reply giving MAC address of router interface
- client now knows MAC address of first hop router, so can now send frame containing DNS query
- IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router
- IP datagram forwarded from campus network into comcast network, routed to DNS server
- demux at DNS server
- DNS reply to clinet with IP address of google
- to send http request, client opens tcp socket to web server
- tcp syn segment (step 1 in 3-way handshake) inter-domain routed to web server
- web server responds with tcp synack (step 2)
- tcp connection established
- http request sent to tcp socket
- ip datagram containing http request routed to google
- web server respondes with http reply
- ip datagram routed back to client