When we want to change or make variations we need to know the protocol

- http at 2.0 now keeps changing protocols keep changing
- google protocol – sdp

what is the internet??

- A network of all local network that allows devices to communicate with one another
- Network of networks of interconnected isps
- Infrastructure that provided services to application option C, such as instant messaging social networks etc.
- Provides a programming interface to web simple api with send/receive messages

NETWORKING LINGO MEANINGS

- Host/end systems – a device connected to a network
- Bandwidth – transmission rate of data
- Packet switches – splits data into chunks known as packets
- DSL – digital subscriber line

Internet contains multiple protocols

- Tcp
- Ip
- http
- Skype
- 802.11

Internet standards

- RFC: request for comments
- IETF: internet engineering task force <- publish internet drafts has to go through the stages from IETF long process + rigorous testing

WHAT IS A PROTOCOL

- Protocols define Format/order of how a message is sent/received and actions taken on the message transmission,recipient ← IMPORTANT FOR ASSIGNMENT

Network edge:

- Consists of hosts that are clients and servers
- Access networks – wired/wireless connection type
- Network core – connection of routers think ISP

Dial up DSL

- Voice and data communicate over same line so dedicated to either one only!
- Useful for outback areas where low population cheaper to setup
- Used time multiplexing

ADSL (DSL – digital subscriber line)

Different frequency on same cable

Adsl depends on distance speeds are directly related from distance to supplier, this is because they use copper wiring! In optic fibre everything travels in electric pulses at speed of light so propagation delay is minimal! Also higher noise and attenuation with longer distance.

- Frequency multi-plexing so splits bandwidth for voice and other for network connection

- 0-4khz for the PSTN
- White space for splitting
- Upstream frequency
- Downstream frequency

DSL is a dedicated network no sharing its dedicated

Access net: cable network shared cancer

- Frequency division multiplexing, so channel for control/data/ and white space same as DSL

NBN fiber to the home

- Fully optical fiber from the supplier to house
- Verizon/google/nbn/fios
- 30mbps -> google fibre (1gbs)

Power over ethernet is where power is supplied from power line

- Access point ethernet from cable

Ethernet over power – power line networking

- Connect power outlets to the ethernet directly

Enterprise access networks (ethernet)

- Constantly evolved currently using switch ethernet, every device connected to cable with switch

Wireless access networks (WAN)

- Shared wireless access connects host to router at access point
- AC5300 standard shown for wireless

NETWORK CORE

- Where the routers reside
- Mesh of interconnected routers/switches
- IT IS THE CARRIER OF YOUR TRAFFIC

Two forms

- Circuit switching used in old telephones
- Packet switching current in day

Circuit switching

- Requires path pre-defined from the start--> end of where the data is going, purely circuit
- Establish circuit beforehand before transmission of data
    - FDM
        - Frequency domain multi-plexing
            - Divides the frequency between users so all users
    - TDM->SHIT
        - Time domain multi-plexing
        - 1 user at a time receives ALL the frequency but can only service one user at a time
        - Entire frequency given to user at time for fixed amount of time
    - Circuit establishment source->client has delays
        - Delay to check the source/check destination of message and if a path exists!/check if it has the capacity to reach destination
        - Termination at end
            - PROS OF CIRCUIT SWITCHING
                - Reliable in terms of data transfer -> guaranteed bandwidth!
            - CONS OF CIRCUIT SWITCHING
                - Requires a pre-defined path from source -> destination so very restrictive!
                - Whenever any router connected disconnect can possibly delete some paths
                - Whenever any network standards change has to be re-built
                - Waste of resources, if a certain amount is allocated to a user and they decide to do NOTHING with it, then that bandwidth is just wasted as guaranteed bandwidth! Waste of resources!!! No sharing!
                - Routers participating in circuit switching need to keep state maintenance

FIRST PRINCIPLE of network design KEEP THE CORE SIMPLE NO COMPLEXITY CIRCUIT SWITCHING IS COMPLEX DUE TO LAST CON LISTED. We don't want routers to remember any state or anything just the basics, eg. When we make a linked list in C we make it generic so it can have multiple uses making it complex working with strings only etc. is bad design!

Independent of data from source

Routers only contain routing tables no info from source/data

All network applications deploy on end system rely on host! No network applications go on router all on host!

Circuit switching is not feasible because it is

- Inefficient
    - Communication is very bursty -> load a page, then downtime whilst maintaining page
    - No sharing during down time so wasted resource!

    - Cannot be changed when new networking standards come
- Fixed data rate
    - We don't always want to have the same 10mb/s rate constantly as this can be seen as a waste of resources its not useful
    - Eg. We download want fastest speeds, when reading a web page or browsing web pages don't need much speed
- Connection state maintenance is difficult
    - Not scalable
    - Would waste resources maintaining state

PACKET SWITCHING

- Data is sent as chunks formatted bits as packets
- Packets have header and a payload
    - Header contains information on the destination and entire package! Has the entire control information! Think amazon shipping! Shipping + item! Item is payload, shipping info is the header has destination and information on order
        - Header contains
            - Internet address
            - Age (TTL) time to live <- how old the packet is
            - Checksum to protect header, to see if there are missing packets (sanity check)
            - Your own address!!!! Source address
- Packets travel independently, there is no defined path like in circuit switching, like in the amazon shipping example packets can choose to take different more efficient paths, like in amazon shipping not everything is delivered all at once even if it is purchased at once!

Payload is the data being sent!

Header has the instructions for how to handle the packet sort of like an api instruction set.

Core router utilise header to find the relevant path, the destination uses the header for sanity check + see who the message is from.

Advantage of packet switching is that packet loss isn't as detrimental as message switching, losing 1/5 parts is better than losing the entire message, we can always extrapolate or atleast see from the 4/5 parts delivered. Can re-transmit aswell with sanity check

When a hop receives a packet, it will use it's routing table to check if there is a path to the destination, if it does it will forward the packet to the appropriate path, else would terminate.

Packet switching timing, routers only check header for destination. Almost negligible time in processing the packet at switch can consider that overhead cost as 0. Only information needed is header. For router to forward to next hop.

Store and forward switching is when the router will wait for the entire packet to arrive before forwadding it to the next hop, so it will not only wait for the header (only information it actually needs) but it will also wait for the payload which causes a larger delay because now we are waiting for information the router is not requiring. However, this is used in public internet as it is more SAFE

and secure, if we don't wait for the entire packet this can cause an issue with the checksum, the header may corrupt/destination corrupt and we are no longer able to do our sanity check as our header/payload are separate.

Cut through switching is a newer technique where routers will only wait for the header to arrive to the router before forwarding the data to the next hop, this causes significantly less delay because the router no longer has to wait for the data to arrive in the payload, it can already send it out on its path as the only information the router uses is in the header! However, this can arise some problems and is not used in the public internet as it is a lot more dangerous store and forward is more safe as we can do sanity check at each hop, with cut through the checksum cannot be accurately calculated as its dependant on the payload too and if the payload is lagging behind the header this can cause issues!

Packets can take any route (most efficient!) we have no control on what the path for the route is for the entire message! Routers maintain NO STATE all they do is see a packet coming, read the header, and set it out to destination path based on routing table! Will not handshake with a bad header!

Statistically multiplexing

- Relies on the assumption that not all flows will burst at the same time causing overload
- Not guaranteed and has some fail cases

When we overload we want to store packets in the buffer queue to avoid overloading this is to absorb transient overload, the overflow will die down. Buffer will absorb transient overload

If the overload is not transient, the queue will be full, we cannot have infinite buffer (unfeasible) extremely costly and physically not possible. To handle persistent overload we block the queue when it is full, and we drop packets ← not so good. RTT could be different on the packets. If queue is empty, no delays quick RTT, but with congestion longer RTT

Pros of packet switching

- Doesn't maintain any state! Scalable
- We don't have to pre-define any path from source to destination so multiple paths
- Delay depends on state of router
- Overhead costs increase as the routers have to process header however, this is also true in circuit switching as the ROUTER has to process the message.
- Supports bandwidth sharing allows more users to use network no guaranteed bandwidth depends on activity and congestion on router, delay is dependant on router

Cons of packet switching

- Can drop packets
- Comes with an overhead cost

Packet switching allows more people to use network suppose we have 1mb/s link

Each user is active 10% of time -> 100kb/s split between users

Circuit switching

- Max capacity 10 users to guarantee 100kb/s

Packet switching

- Can hold infinite amount of users
- With 35 users on network, the probability that more than 10 users are active at the same time is less than .0004 -> Bernoulli trials and binomial distribution (statistical multiplexing gain)

Global internet → connecting the network of networks

Internet Structure

- Global ISP like Telstra etc that connects to access nets and all users joining are connected to the users.
- Multiple global ISPS since there is economic competition
- To connect global ISPS together we can set up
    - Peering links high speed link between ISPS private, or we can use
    - IXPS → internet exchange point
        - Saves money on infrastructure
        - Requires less hop considerabely since we don't need to hop to the peering link
    - Content providers can run their network on the IXP providing faster connection for their services.
- Regional ISP
- AARNET
    - Network for australia's academic and research

Key Properties of Links

- Bandwidth is the width of a link (like the width of pipe flowing water) instead width of link flowing data
- The longer the pipe longer delay between when pipe is opened and the water is received, similar to networking, propagation delay
- Keep efficient flow through pipe, we need to constantly pump the volume of the pipe to have an efficient link
- Bandwidth delay product gives the volume of the link, it is the bandwidth * propagation delay

Packets queue in router.

- When the buffer is full packets are dropped

Delays in the correct order

- Nodal processing delay
    - Checks bit errors - checksum
    - Determined destination
    - Normally <millisecond
- Queueing delay → difficult to predict
    - Time waiting at router before transmission
    - Dependant on router congestion
    - If we know the size of the buffer we can calculate delays in the queue
- Transmission delay
    - Packet length (size of packet) = L
    - Bandwidth of link (bps) = R
    - = L/R
- Propagation delay
    - Length of the link = d
    - Propogation speed -> how fast it moves through the link = s ← typically 2*10^8m/s
    - = d/s

For N routers in the hops, if the queueing delay is the same total delay = n * sum of all delays

Queueing delay

- If a router can process 1 packet per second
- Packets arrive at → a packets per second
- The packet length → L
- Bandwidth of link → R
- Total delay = AL/R

If the router can process 1 packet per second, and we send 1 packet per second, our queueing delay would be 0 as queue would be empty!

If the router can process 1 packet per second, and we send 10 packets in a burst every 10 seconds, the arrival rate would be the same however, there would be a queueing delay since 9 packets wait in queue while first is being processed.

THE PATTERN OF THE TRAFFIC MATTERS FOR QUEUEING DELAY!!

If aL > R → queue will full up and packets will drop

aL/R is the traffic intensity typically when aL/R = 1 → queueing delay == 0

end-to-end delay is the sum of all delays across all hops on path

throughput

the rate at which bits are transferred between sender and receiver.

Instantenous = rate given at a point in time

Average = rate over whole period of time

Bottle neck occurs when one link is smaller than others

PROTOCOL LAYERING

Keep core simple

Layering

Three design steps

- Break down the problem into tasks → divide and conquer
- Organize the tasks
- Decide who does what

What does it take to send packets across.

DECOMPISITION

- Task 1: send along a single wire
- Task 2: connect task 1 N times till it reaches end systems

Bits/packets on a wire ->physical layer

Deliver packets within local network ->datalink layer

Deliver packets across global network -> network layer

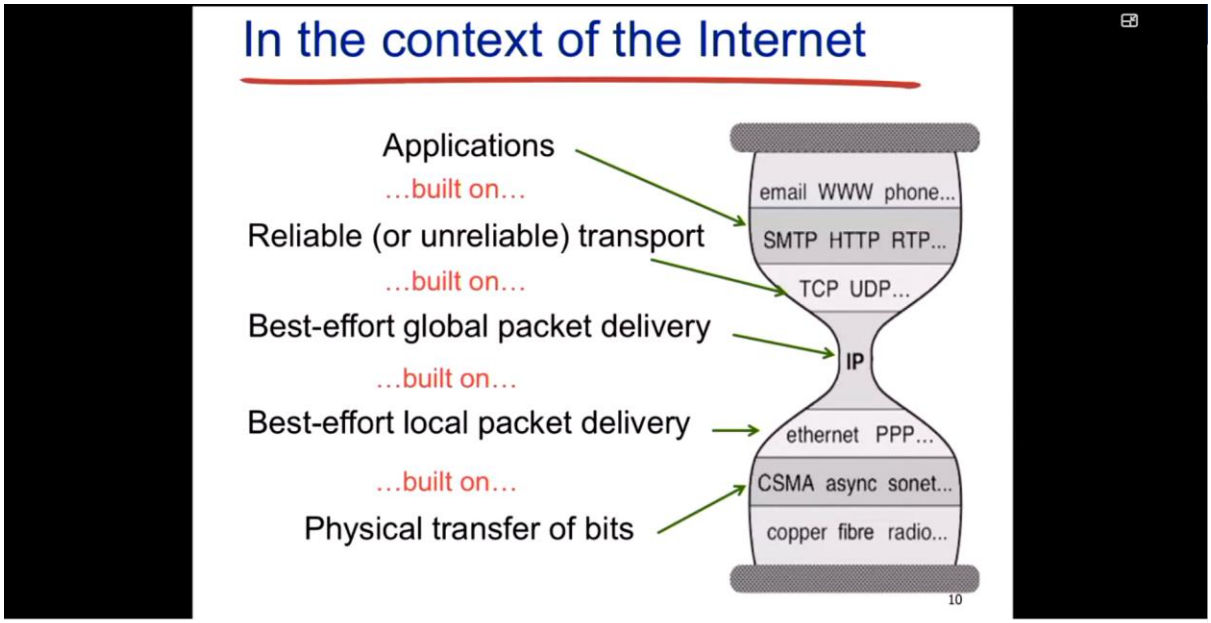
Make sure packets get to the destination -> transport layer

Process the data -> application layer

Abstraction

➔ Hiding the contents whilst providing the service

Applications are built on reliable or unreliable transport built on global packet delivery built on local packet delivery built on physical transfer of bits.



Application

- Network applications

Transport

- TCP

ONE IP LAYER unifying protocol

The routers only work in the network layer no application layers

Benefits of layering

- Abstraction ← security
- Tasks can be split up
- Modularity – easy to tell where errors are in.
- Easy to integrate different systems, existing applications don't need modifications with layering

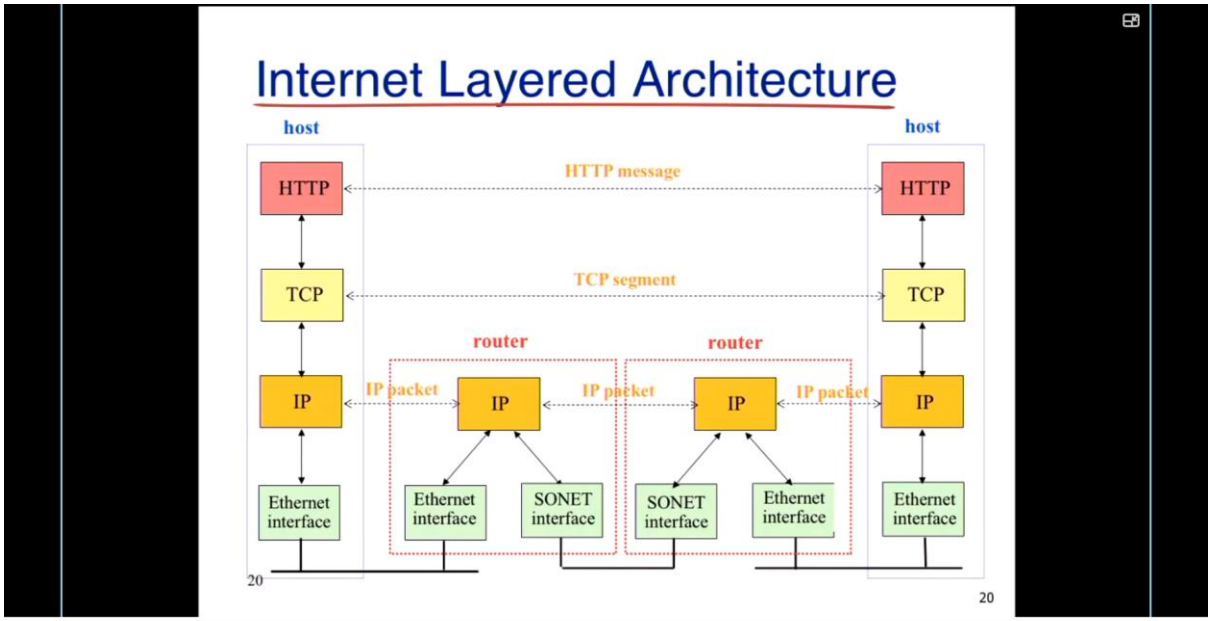
Disadvantages of layering

- Larger headers, TXP + IP + Ethernet headers are up to 54 bytes
- Worse performance overall → information hiding
- Checksum being used by multiple layers, duplication of efforts
- Layer violations when the gains too great to resists or when the network doesn't trust ends eg. Firewalls

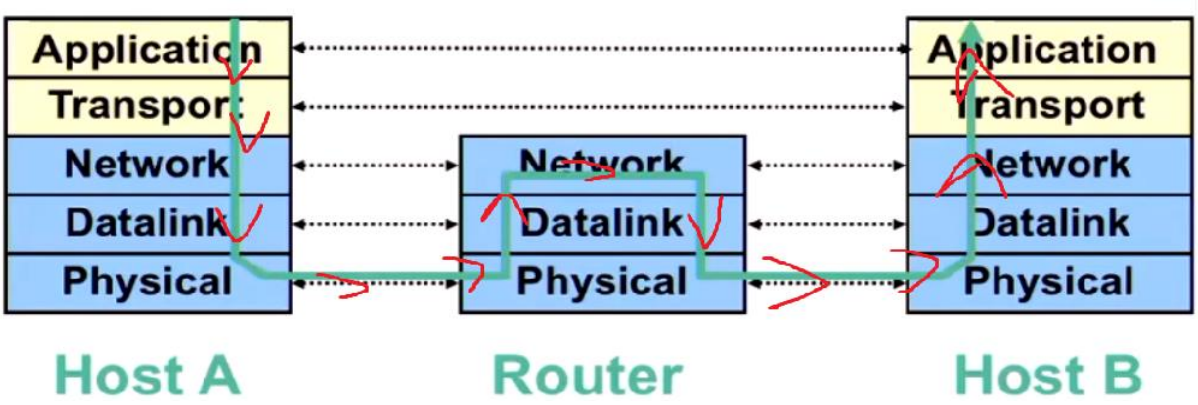
Bits arriving on the wire must make it to the application

All layers must exist on the host

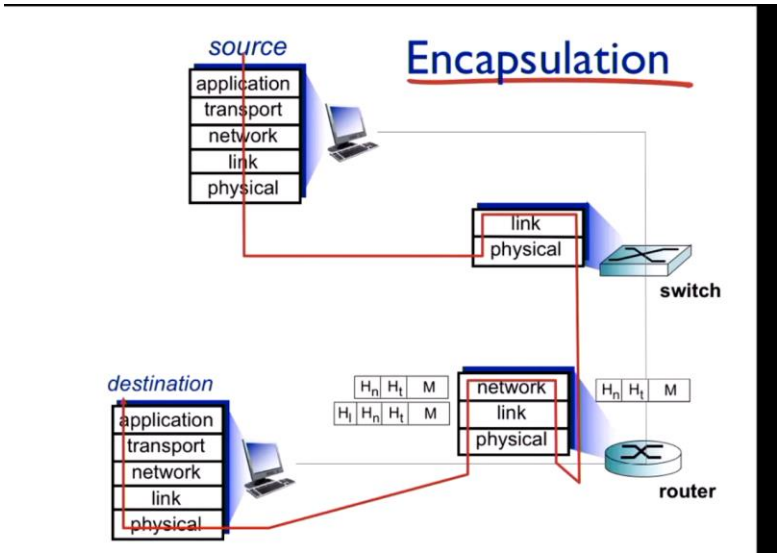
Bits arrive on wire at router, physical layer, packets must be delivered to next hop, data link layer, routers participate in global delivery, networking layers, routers don't support reliable delivery, transport layer and above are not supported



Layers interact with peer's corresponding layer



Down stack when transmitting, and up the stack when receiving



Encapsulation → LARGER OVERHEAD AS YOU GO DOWN THE STACK YOU ENCAPSULATE LEADS TO LARGER HEADERS, PDU -> concatenating to the end of the PDU

Application layer – message → M

Transport layer – segment → Ht + M adding header of the transport layer to the message

Network layer – datagram → Hn + Ht + M adding header of network layer to segment

Link Layer – frame → Hl + Hn + Ht + M adding header of link layer to the datagram

Switch then transports the frame through the link layer and physical layer to the router, note the switch has no networking layer

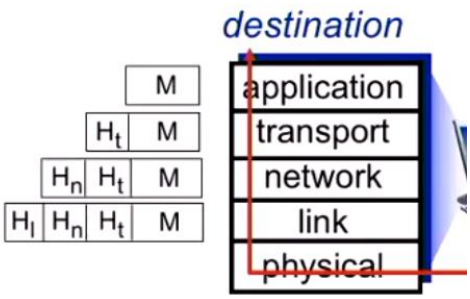
At the router we process the incoming frame

Travels to the link layer → frame → Hl + Hn + Ht + M processing the header at the link layer of the router.

Travels up to the network layer for transmission → datagram → Hn + Ht + M processing frame

Travels back down to the link layer as a new frame → Hl + Hn + Ht + M

While processing at the destination we reverse this process returning the message



Transport layer header – 20 bytes

Network header – 20 bytes

Datalink header – 18 bytes -> 14-byte for header 4 bytes for tailer

58 bytes of extra information just to carry 1-byte message

We have to add 58 bytes to the packet to send a message just for encapsulation

Application Layer

TCP → reliable

UDP → unreliable

When we create a network application, it can only be deployed on a host or end system, NOT A ROUTER OR IN NETWORK CORE → KEEP CORE SIMPLE. Test on own machine, if it works on own machine work, it will work anywhere else

IPC interprocess communication

Processes talk to each other via IPC

- We need abstractions to communicate to different machines

SOCKETS -> NEED IP ADDRESS + PORT NUMBER

Processes send/receive messages to/from its socket

Sockets can be thought of like a door that connects the application layer and the transport layer

- Sending process → pushing message out door abstraction, push it out and other part will handle for you

To identify messages processes must have an identifier, host devices have a unique 32- bit ip address

IP address only provides access to networking layer of machine, we must know which process the packet is being delivered to. This is done using port number.

We can uniquely identify a socket by knowing it's ip address and its port number to know which process to address on the machine.

UDP → SEND TO ip address + port number

TCP → SEND, already established connection

Identifier for a process consists of an ip address and a port number assosciated to the process on the host. HTTP server → port 80, mail server → port 25

To send a http message to cse.unsw.edu.au we use the ip address and port 80

Client server architecture

Server

- Exports well-defined response/request interface
- Process that sits listening for requests
- When receives a request it will carry it out
- Always on the host
- Permanent IP address
- Static port conventions, http:80 ssh: 22 email:25
- Data centres for scalability
- Communication available between other servers

Clients

- Short lived processes that make a request
- The users of the application
- Starts communication with server
- May be intermittently connected
- May have dynamic ip addresses
- Do no communicate with each other all done through medium of the server

Peer to Peer (P2P) architecture

- No always on server
- Arbitrary end systems communicate directly
- Often used for
    - File sharing (BitTorrent)
    - Games
    - Video chat/distribution
    - Distributed systems
- User downloads file off site and then they themselves become a server hosting for others once they have it

Peer to peer pros and cons

- Pros
    - Clients who discover each other can communicate without server, in client server no communication if server goes down
    - Self-stability, new peers bring new service capacity (bit torrent when you host) as well as new service demands, ← bit torrent
    - Speed: parallelism
    - Reliable communication
- Cons
    - State uncertainty, there is no shared memory or clock
    - Action uncertainty: users are in control or if mutually conflicting decisions
    - Users have free will at anyway they can walk away from the system

Application layer protocol

- Types of messages, message syntax, message semantics, rules need to be provided

There are open protocols eg. HTTP, SMTP or properiatery protcols like skype or private ones

Transport service an app requires

- Data integrity, throughput, timing and security.
- Data integrity
    - Some applications require 100% reliable data transfer
    - Other applications eg. Audio can tolerate some less
- Timing
    - Some apps require low delay to be effective eg. Games or movies
- Throughput
    - Some apps require a minimum amount of throughput (bandwidth rate) to be effective
    - Other apps can dynamically change based on throughput
- Security
    - Encryption and data integrity

Transport service requirements: common apps

application	data loss	throughput	time sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 50kbps-1Mbps video:100kbps-5Mbps	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few msecs
interactive games	loss-tolerant	few kbps up	yes, 100's msec
Chat/messaging	no loss	elastic	yes and no

38

Internet apps: application, transport protocols

application	application layer protocol	underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

40

Employ TCP through their own application layer protocol dependant on the application

Btw socket address = ip address + port address

Hyper text transfer protocol (HTTP) uses TCP

Webpage consists of objects

Objects are different file types

Web page consists of a html file which consists of several referenced objects

Each object can be addressed by a url

Request index.html as soon as establishment of connection is setup (incurs some delay making connection)

For each object on different server, another connection must be established to retrieve the object.

Uniform resource locator (URL)

Protocol://host-name[:port]/directory-path

Directory path – reflects the file system

http overview

- Request -> response system client, server paradigm

- Index.html

http is stateless

- Server maintains no information about past client requests

Protcols that maintain state are complex! Keep the core simple.

HTTP messages has two types

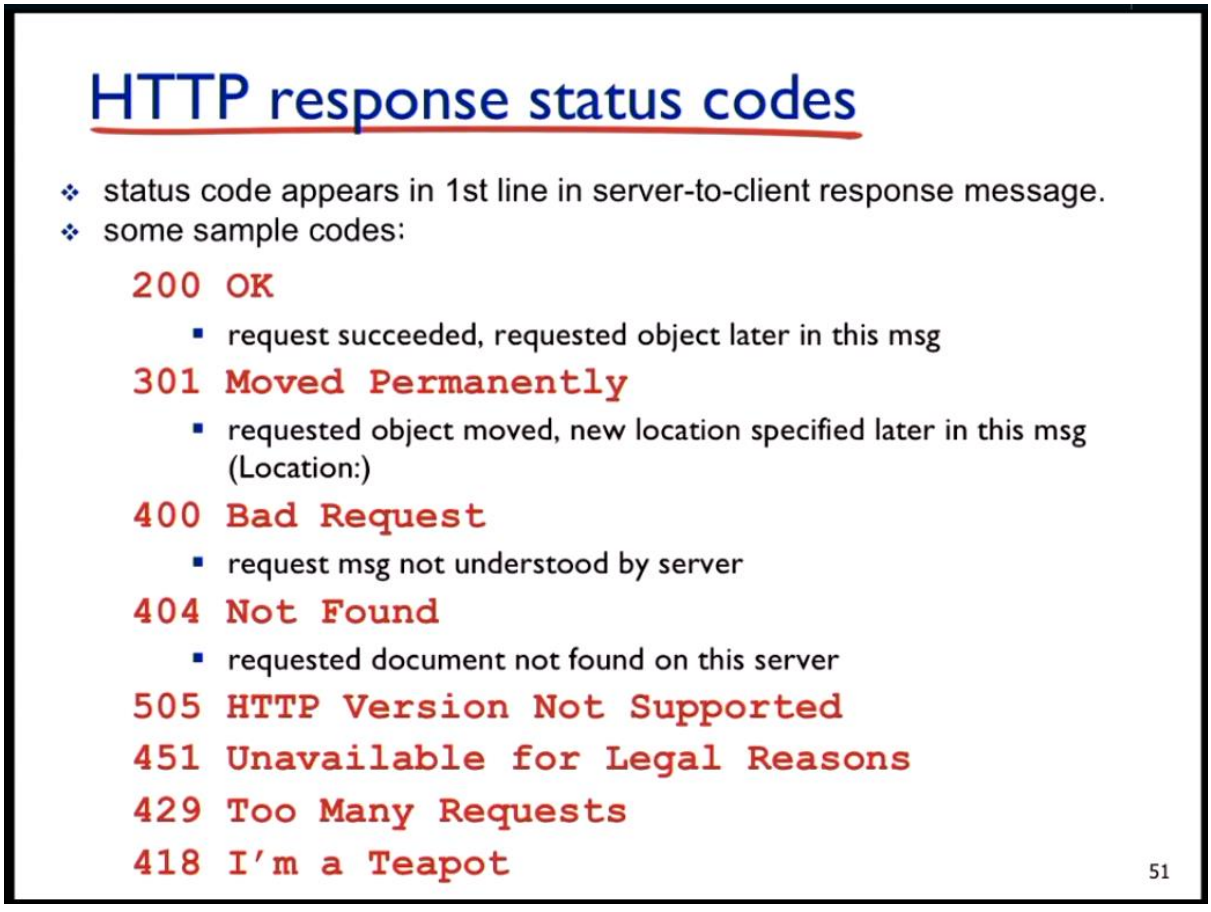
Requests/response

Request -> GET/POST

Carriage return character + line feed character (\r\n)

HTTP response

Has the response 200 ok means everything is okay, multiple different types of responses however, comes with a status code + status phrase.



LECTURE 3

Rquest Method types

HTTP/1.0:

GET → request page

POST → uploads user response to form

HEAD → ask server to leave requested object from response

HTTP/1.1

GET,POST,HEAD

PUT → upload file in entity body to path specified

Uploading form input

POST method

- Web page includes form input
- Input is uploaded to server in entity body, method="post" in url

GET method:

- Uses GET method
- Input is uploaded from url field

HTTP is a stateless protocol → doesn't remember you

Many websites use cookies to remember who you are so that you don't need to constantly authenticate user, saves your state! Eg. If you go online shopping needs to remember your cart

COOKIESS

Four components

1) Cookie header line HTTP response message
2) Cookie header line in next HTTP request

Set cookie sent by server → client

Cookie file is then saved onto client somewhere

Next time client contacts server, it will use that cookie in the header line. Server will recognise the cookie and then retrieve the information associated to that cookie.

Cookies allow for targeted advertisements

Cookies come with disadvantages too

- Sites learn a lot about you personal details included
- 3rd party cookies can follow you across multiple sites, you can turn them off but its unrealistic

3rd party cookies

- Accessing a web server
- Whenever you load a page all objects on different servers are requested → creates a cookie there, third party cookie.
- Double clicks create a cookie that can then track your browser usage
- 2007 google acquired double click → 3.1 billion dollars
- Double click allows targeted advertisements for a user
- DoubleClick → 1 pixel of unrecognisable data to create a cookie sneak 100

Performance of HTTP

- Page load time (PLT)
    - from click until user sees page
    - key measure of web performance
- depends on factors such as
    - page content/structure
    - protocols involved
    - network bandwidth and RTT

performance goals

User requires

- fast downloads
- high availability

Content provider requires

- happy users
- cost effectiveness and infrastructure

Network requires to

- avoid overload

caching and replication helps achieve these goals

How to improve PLT

Most web pages have multiple objects, eg. Images embedded

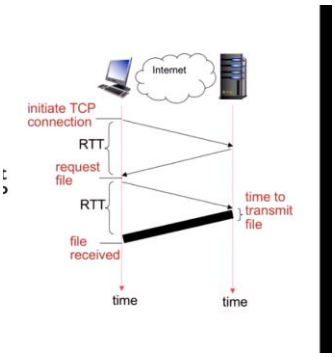
Non persistent HTTP(HTTP 1.0)

Connection fetches a single object, non persistent connection!

Done shitly

One object at a time

This creates multiple delays



HTTP response time

- ONE RTT to initiate TCP connection

- One RTT for HTTP request and response
- File transmission time
- Non-persistent HTTP response time = 2 RTT + file transmission time

HTTP 1.0 OR NON PERSISTANT HTTP

Poor PLT

Needs to set a connection for EACH object → higher delays because requires multiple TCP connections to the SAME SERVER

Sequential requests/responses even when resources are located on DIFFERENT SERVERS no parallel requesting

MULTIPLE TCP connections slow-start phases

Everytime you have to create new connection, lose momentum

Downsides of parallel HTTP connections

- Can lead to network overload
- Congestion in network, since its all bursty → sequentially queue = 0
- Assume 40 objects on one server, with primitive approach will create 40 requests on same server sequentially 1 at a time. Parallel will create 40 requests at one time, will increase network load, can be harmful

Persistent HTTP

- Server leaves TCP connection open after sending response
- Subsequent HTTP messages between same client/server use the same TCP connection → much more efficient than creating a TCP connection each time
- Allow TCP to learn more accurate RTT estimate
- Allow TCP congestion window

Persistent without pipelining

- New requests only when previous response has been received
- One RTT for each referenced object
- More efficient than non-persistant as it keeps connections open that will be re-used

With pipelining

- Default in HTTP/1.1
- Sends out multiple requests at once sort of like multi-threading
- As soon as it encouters a referenced object → send a request, as low as one RTT for all objects

Response time of one page with three embedded objects

- Initiate TCP connection → 1 RTT
- Request file -> time to transmit file → 1 RTT

How to improve PLT

- Reduce content size for transfer → compression
- Change HTTP to make better use of bandwidth → persistent connections and pipelining

- Change HTTP to avoid repeated transfer of same content → caching
- Move content closer to the client → closer servers

Caching

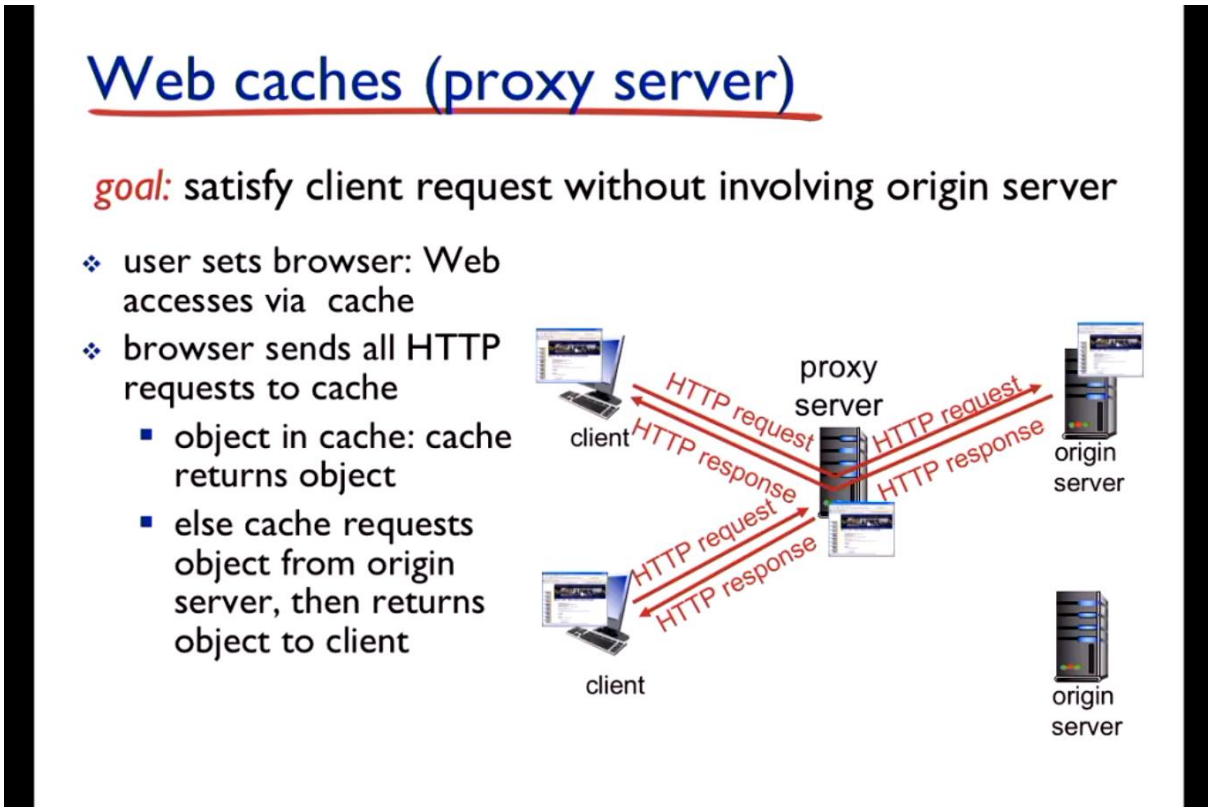
- Allows you to load a state much faster by using a reference
- Works well up to a point, however many unique request. Large overlap in content

Web caches (proxy server)

To attempt to satisfy client request without involving the original server

Client sends HTTP request, response is received through server, proxy server works through origin server, and caches the response

All new objects are cached, and existing objects in cache returns the object



Cache acts as both client and server,

When the first request is made, cache acts as a client requesting data, and then when it is received it can then act as a server for the data.

Cache is installed by ISP typically

Caching reduces response time for client requests, reduces traffic on access link and enables poor content providers to deliver content

Caching example

Assumption

Avg object size → 100K bits

Avg request rate from browsers to origin server → 15/sec

Avg data rate to browser → 1.5mbps

RTT access to origin server → 2 seconds

Access link rate: 1.54Mbps

aL/R ~= 1 → 1.5/1.54

LAN has 1gbps speed → utilisation = 1.5/1000 → 0.15%

Access link utiliszation = 99% → insane!

Total delay = internet delay + access delay + LAN delay

= 2 sec + minutes + usecs msecs

If we increase access link to 154 mbps we are reducing access link utilisation to 0.99%! reduces the delay from minutes to micro seconds, however this is a much more expensive approach!

Better option, local web caching → using 1.54 MBPS access link rate

All subsequent requests do not use access link → use LAN (local web cache)

Assume cache hit rate is 0.4

40% of requests satisfied at cache → LAN speed

60% of requests satisfied at origin → access link

Access link utilization is reduced to 60% now (add in the 40% from cache (very small)

This is even faster than 154MBPS access link as it utilises the lan speed, access link requirement down to 60%!

Likelihood of cache hits

- Distribution fo web objects requests follow a zipf-like distribution
- The probability that a document will be referenced k requests after it was last referenced is roughly 1/k. web traces exhibit excellent temporal locality

SIMILAIR TO VIDEO CONTENT CREATERS → 10% OF THE TOP POPULAR VIDEOS ACCOUNT FOR NEARLY 80% OF VIEWS WHILE REMAINING 90% ACCOUNT FOR THE REMAINING 20% OF VIEWS

More popular content is → more likely is to be cached!

To check if the cached version is still valid we can use CONDITIONAL GET REQUESTS

If modified since will make sure a resource has been modified since a given date, otherwise will send error 304 response

To make a resource uncacheable, we can use the expires header line to specify a date where we no longer cache!

Replication → clones' website to create another web server for global distribution similar to google hq in Australia. This is done with network providers CDN (content distribution network) servers. Or how league created server in Sydney for Australian players → lower RTT

CDN to improve HTTP performance

Caching and replication used as a service

CDN combination of pull caching and push replication

- Pull → direct result of clients requesting
- PUSH → expectation of high access rate

HTTPS!!!

HTTP is insecure

Basic authentication with HTTP uses base64 encoding → easily converted into plain text

HTTPS: HTTP over a connection encrypted by transport layer security

Provides

- Authentication
- Bidirectional encryption

Server push content method

Servers can push content and reduce overhead, they provide index.html and give you the 4 objects to complete the page rather than client creating multiple requests for other objects, so there is less overhead on the client, server can process the 4 objects rather than client.

Can lead to wasted resource if user requests and then closes browser

Application layer → EMAIL AND DNS

Electronic mail has three major components

- User agent for composing mail with address of mail server → normally browser/installed application on PC
- Mail servers receives mail in queue. Once it is your turn it will create a TCP connection and push out your mail to recipient mail server.
- Simple mail transfer protocol (SMTP) runs ontop of TCP (reliable as TCP is reliable)

Mail server contains

- Mailbox contains incoming messages for users
- A queue for messages (like buffer queue)
- SMTP protocol between mail servers to send email messages between different mail servers
    - Client → sender of mail
    - Server → receiver of mail
- NO INTERMEDIATE MAIL SERVERS (DIRECT COMMUNICATION BETWEEN SENDER AND RECEIVER).

- This allows for easy one way communication between sender and receiver

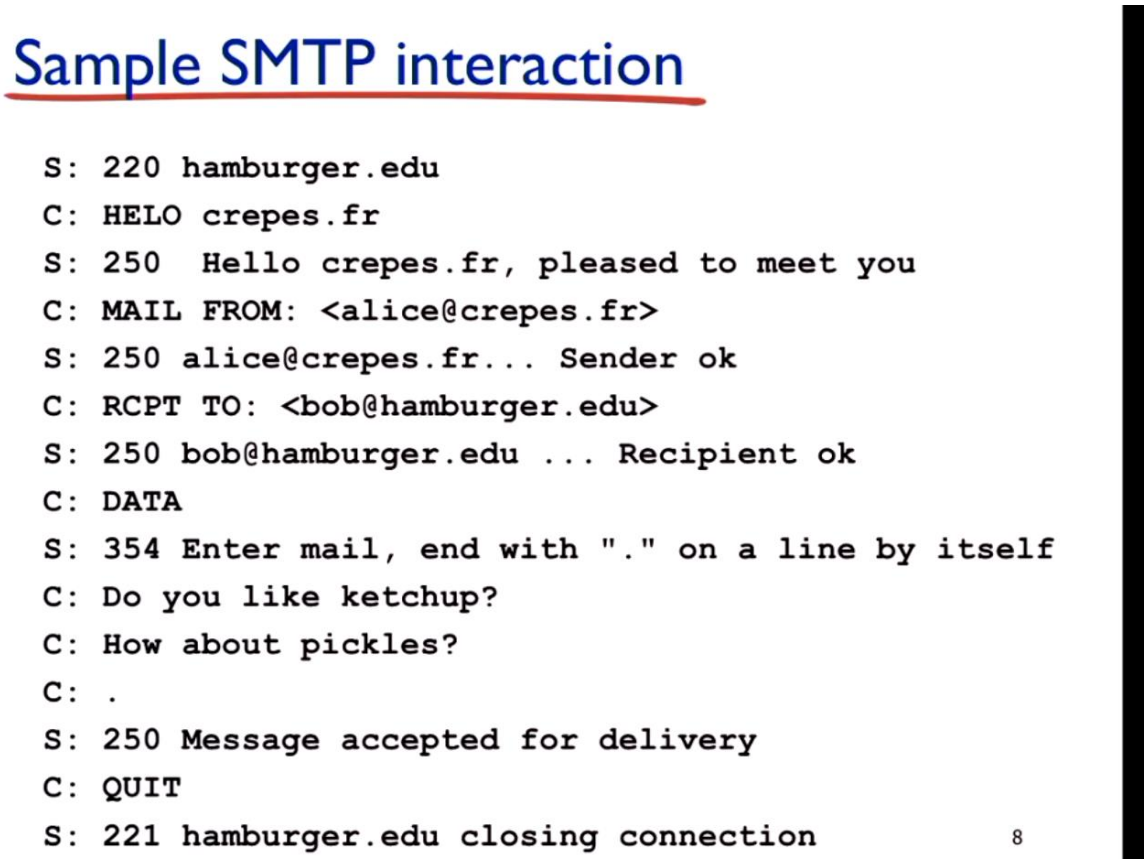
Email SMTP

- Uses sTCP for reliable transfer email message from client to server, port 25
- Direct transfer
- Three phases
    - Handshake (greeting)
    - Transfer of message
    - Closure
- Command response interaction
    - Commands: ASCII text
    - Response: status code and phrase
- ALL MESSAGES MUST BE IN 7-bit ASCII

SMTP is used to get message from user agent and transfer it to the mail box. To transfer it across we do NOT use SMTP, we use TCP to transfer message across!

S = server

C = client



PHISHING

Spear phishing

- Directed at specific individuals or companies, attackers may gather personal information
- Nigerian_prince_email.exe

Clone phishing

- Runescape phishing,
- Attempt to make a legitmate email to make you click on a link to go to their version of the wbesite to steal ur credentials

SMTP

- Uses persistent connections
- Requires message to be in 7 bit -ascii
- CRLF to determine end of message

Comparison with http

- http → PULL
- smtp → Push
- both have ascii command/response interaction/status codes
- each object in HTTP is encapsulated in its own response message
- SMTP → multiple objects sent in multipart message

Mail message format

SMTP protocol for exchanging email msgs

RFC standard for text message format

Header lines → to+from+subject, different from SMTP mail from RCP to: commands!!

Body: the "message" → only ascii

- Why do we have sender's mail server/why do we have a separate receiver's mail server

  - mail servers must be constantly on!

  - we need to transfer responsibility of delivering emails to the mail server instead

  - if we do it on ours we will have to make sure 100% of time connection is on

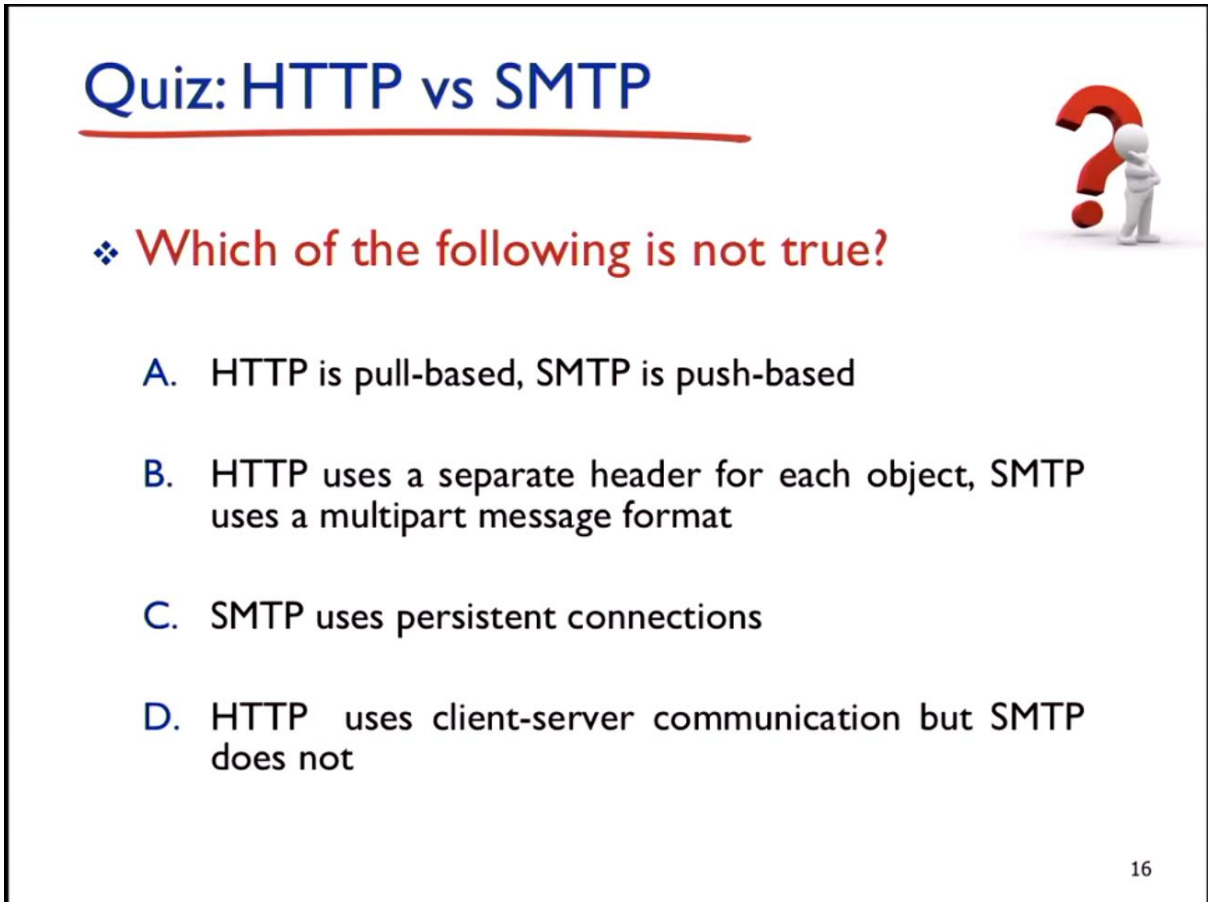
If SMTP only allows 7-bit ascii how do we send pictures/videos/files via email

- We encode these objects as 7-bit ASCII

Mail access protocols

- SMTP -> delivery/storage to receiver's server
- Mail access protocol retrieval from server
  - POP: post office protocol: authorization download → cannot create folders and sort emails
  - IMAP: Internet Mail Access Protocol: more features including manipulation of stores messages on server

- HTTP(S): Gmail,Yahoo! Mail etc.



D SMTP is still client server based

DNS – Domain Name System

We have name + student id, all records use student id and all backend work is done with student id however our name is an alias!

In web communication is based on IP addresses, 32 bit

If we were asked to get info from google, we use google's alias rather than its ip address

DNS

- Distributed database implemented in hierarchy of many name servers
- Application layer protocol

DNS takes a name and will do a query to find the corresponding IP address either via recursive or iterative query

Any protocol which requires naming will refer to DNS! Nslookup + dig will allow us to access DNS directly

Hosts.txt beginning of DNS

Maintained by the Stanford research institute

Changes were submitted to SRI by email

New versions of hosts.txt periodically FTP'd from SRI

An adiminstrator could pick names at their discretion

As the internet grew this system broke down as SRI couldn't handle the load

Names were not unique and hosts had inaccurate copies of hosts.txt

DNS CAME TO THE RESCUE DADDY DNS

Why not centralize DNS?

- Single point of failure
- Traffic volume
- Maintenance
- Distant centralized database
- ITS NOT SCALEABLE

What is DNS?

- Hostname to ip address translation
- Aliasiang for host → canonical, alias names
- Mail server aliasing
- Load distribution
    - Replicated web servers → many ip addresses correspond to one name
    - Content distribution networks (CDN): use ip address of requesting host to find best suitable server, closest least loded etc

GOALS

- No naming conflicts → UNIQUENESS → SET
- Scaleable
    - Many names
    - Frequently upodated
- Distributed and autonomous administration
    - Ability to update my own machines names
    - Don't have to track everybody's updates
- Highly available
- Fast lookups

This is achieved through hierarchy

- Three intertwined heirarchies
    - Namespace
    - Administrated
    - Storage

Heirarchial namespace

- Top is root → NEVER TO BE NAMED
- Below that TLD → top level domain names, eg. Edu com gov mil org
- Also CSTLD → country specific TLDS , eu au us fr uk
- Intermediate name servers → extras
- When writing urls go from bottom up no root included

Domains are sub trees

Name is leaf->root path

Depth of tree is arbitrary, limit 128

Naming conflicts only rise within own domainds!

Heirarchial administration

Authoritative name server (NS) berkley is administrator

All sub domains managed by berkeley

Below that eecs has its own zone where it is the authority

EACH NAME SPACE HAS AUTHORITY OVER ITS OWN ZONE!!!

A zone corresponds to an administrative authority that is responsible for that portion of the hierarchy

UCB controls names: *.berkley.edu and *.sims.berkeley.edu

EECS controls names *.eecs.berkeley.edu

Each server stores a small subset of the total DNS database

An authorative DNS server stores resource records for all DNS names in the domain it has authority for

Each server needs to know other servers that are responsible for the other portions of the hierarchy

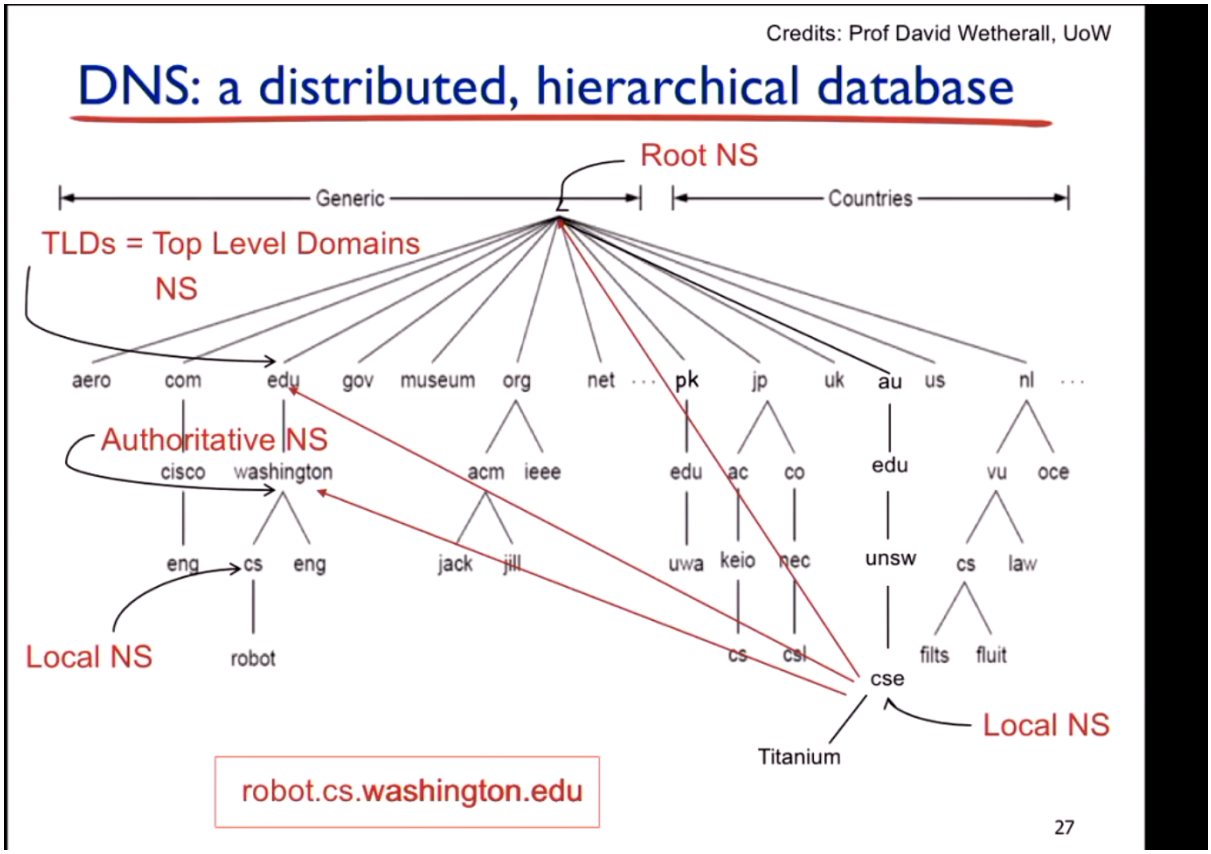
- Every server knows the root
- Root server knows about all top-level domains

First on dns query for any name server, we have name we need IP eg. Robot.cs.washington.edu

- ➔ Will ask local name server to check cache for which ip address we looking for
- ➔ If no reply we jump straight to root we do not go up the tree
- ➔ Root will check the full url, it will start sending it on the right path
- ➔ Root will send it to .edu which is the highest level under root with right extension
- ➔ Cache the entry so next time can jump straight to edu
- ➔ Cache on the way down!!
- ➔ Now .edu will send you to .washington
- ➔ Cache the entry so next time can jump straight to Washington
- ➔ Washington is the authoritative Name server
- ➔ Washington will have the mapping of all IP's under Washington and will return ip

Caching is important, it will cache robot.cs.washington.edu and now any other service requesting it will have result cached

THIS IS AN ITERATIVE QUERY WE RECOMMEND JOB IS ON CLIENT



In iterative search, only we have the cached answer since the answer travels through us

In a recursive search, the root server and .edu has the answer cached!!

Is there a single root name server? → NOOO

13 root servers!! Managed by 12 organisations geographically distributed

In reality we are distributed all across the globe

Unicasting/broadcasting, 1 source many users = broadcaster

Multicasting, send message to a subrgroup of entire course

Any casting, get hold of any member of a specific group! <← used in DNS finds the closest root server for you

All addresses mapped hard coded so it will find the first one closest to you

TLD authoritative servers

Must know how to differentiate between local name servers and authoritative servers or root name server and top level domain service and know their roles!

Top-level domain (TLD) servers:

- Responsible for
  - .com
  - .net

    - .edu
    - .aero
    - .jobs
    - .mesuems
    - All country domains
        - Uk
        - Fr
        - Ca
        - Jp
        - Au
        - Network solutions maintains servers for .com TLD
    - Educause for .edu TLD
- Authoritative DNS
    - Organization's own DNS server(s) providing authoritative host name to ip mappings for organization eg. Google
    - Can be maintained by organization or service provider
- Advantage vs disadvantage of using our own local name server vs google name server

DNS records what is stored in a DNS record

DNS – distributed database storing resource records (RR)

RR format: name, value, type, ttl

TTL → time to live refers to how long to keep a packet alive before revalidating, this is to stop a data packet from circulating indefinitely.

Type = a → ipv4 address/aaaa → ipv6 address

- Name is host name
- Value is ip address

Type = NS → name server

- Name is domain
- Value is hostname of authoritative name server for domain

Type = CNAME → canonical name

- Name is alias for some canonical name
- Value is the canonical name

Type = MX → mail server

- Value is name of mailserver associated with name

Type – PTR → reverse query

- Stores reverse DNS entries

DNS protocol and messages

- Query and reply messages both with same format

    - Message header
        - Identification, a 16-bit number for query, reply to query uses same number
        - Flags
            - Query or reply
            - Recursion desired
            - Recursion available
            - Authoritative reply

Inserting records into DNS

Example new start-up networkland

Register the domain name at DNS registrar e.g. Network solutions

Provide names, ip addresses of authoritative NS both primary and secondary

Registrar inserts two RRs into .com TLD server

- Networkutopia.com, dns1.network.utopia.com, NS
- Dns1.networkutopia.com, 212.212.212.1, A

Create authoritative server type A record for networklank and type MX record for networkload.com

Where do we insert the type A and type MX Records??

- Store in our own name server!!
- Since we are authority now

Reliability in DNS

- DNS servers are replicated
    - Name service available if atleast one version is up
    - Queries can be load-balanced between versions for performance
- Usually UDP used for queries (sometimes TCP not likely)
    - Need reliability: must implement this on top of UDP
    - Spec supports TCP but its now likely implemented
    - We try alternate servers on time out, however there is exponential backoff when retrying same servers
- Same identifier for all queries
    - We don't care which server responds or which version we just want response
    - All replies assumed same

DNS provides indirection

Addresses can change underneath

- Move www.abc.com to 4.125.91.21
- Humans/applications/users should be unaffected
- When changed we ask root, and it may have a cached copy of old ip, so we apply TTL to make sure new cached copies are validated, when cache empty we go down the chain of authority to the host we are tying to reach

Name could map to multiple ip addresses

- This allows for better load balancing
- Reduces the latency by picking closest servers

Multiple names for same address

E.g. Many services use mail, web, file transfer on same machine

And aliases allow for easy identification

Reverse DNS

- Given ip address we can look for domain name, -x option
- Special PTR record types used for reverse DNS look ups
- Reverse DNS is used in tools such as
    - Traceroute//ping
    - Receiving trace header field in SMTP
    - SMTP servers for validating ip addresses of originating servers
    - Internet forums tracking users
    - System logging/monitoring tools
    - Used in load balancing servers/content distribution to determine location of requester
    - Firewall, when a packet comes in we do reverse DNS to check name of app

Do you trust your DNS ServeR?

Alternate DNS → can be redirected unwillingly to a different website through dns this is flat out censorship

NX domain = not accessible domain

- Censorship
- Logging
    - IP addresses, websited visited, geolocation data
    - Google DNS

DNS is powerful but least protected, google DNS can log everything

ATTACKING DNS

- Ddos attacks
    - Bombard root servers with traffic
    - Not successful to date
    - Traffic filtering
    - Local DNS servers cache ips of TLD servers allowing root servers to be bypassed
- Bombard TLD server then
    - A lot more dangerous, there are a lot and less protected
- Redirect attacks
    - Man in the middle intercepting queries
    - DNS poisoning which sends bogus replies to dns server which caches
- Explot DNS for DDoS

    - Send queries with spoofed source address, the target IP
    - Requires amplification
- Look up DNS tunnelling and exfiltration

DNS cache poisoning is when you force fraudulent cache for another service, suppose you own www.trolled.com and when you perform your query one of your authority sections is google.com, google.com will be caches under your machines ip address!

The solution would be to not allow DNS servers to cache IP address mappings they have no authority over!

PEER to PEER + CDNS

Pure P2P architecture

- No always on server
- Arbitrary end systems directly communicate
- Peers are intermittently connected and change ip addresses

VOIP -> coice over IP

File dirtribution: client server cs P2P

How much time is needed to distribute file size F from one server to N peers

Peer upload/download capacity is a limited resource

From client/server

Server transmission

- Must send (upload) N file copies
- Time to send one copy: F/Us (Us = upload speed)
- Time to send N copies: NF/Us
- Client each client must download a file copy
- Dmin = min client download rate
- Client download time = F/dmin

Time to distribute file size F to N clients using client server approach

DownloadC-s >= max{NF/Us,F/dmin}

If N increases, delay increases linearly in N → more client = more delay

In case of P2P

Server transmission

- Must upload atleast one copy
- Time to send one copy : F/Us

Client

- Each client must download file copy
- Time to download = F/dmin

Clients as aggregate must download NF bits

- Max upload rate (limiting max download rate) = Us + sum of all users who have uploaded

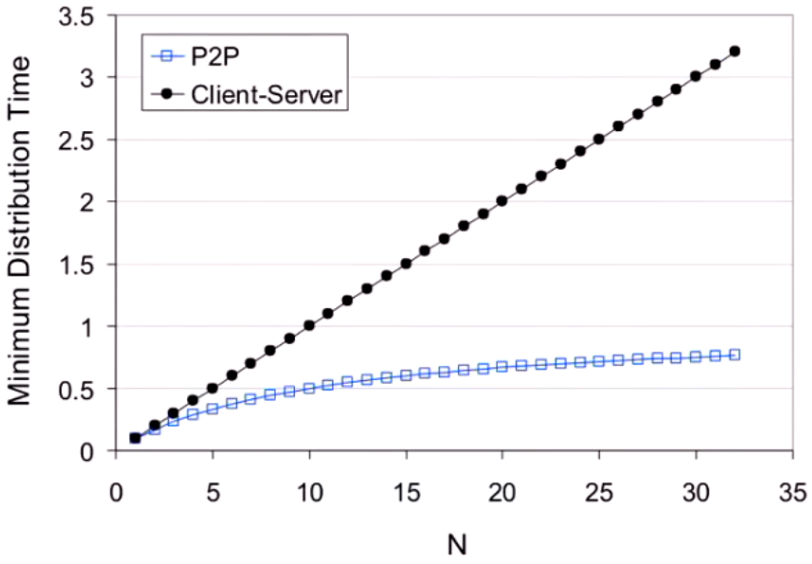
So now time to distribute file size F to N clients using peer to peer approach

DownloadP2P >= max{F/Us,F/dmin,NF/(Us + upload of all users involved)

It is still increasing linearly in N, however the more clients we add in, the more server capacity we add

Client-server vs. P2P: example

client upload rate = u, F/u = 1 hour, $u_s = 10u$



7

More users = more upload capacity means we can download faster

BitTorrent example

- File divided into 256kb chunks
- Peers in torrents send/receive file chunks

Tracker – tracks peers participating in torrent

Torrent – group of peers exchanging chunks of a file

.torrent files

Contains the address of trackers for the file, where you can find other peers

Contains a list of file chunks and their cryptographic hashes, this ensured that chunks are not modified

Peers joining torrent initially has no chunks, however accumulate them over time

Peer joining registers with tracker to get list of peers and connects to a subset of peers (neighbours)

While downloading peers upload chunks to other peers

Peer may change peers with whom it exchange chunks

Churn – peers come and go

Once a peer has entire file it may selfishly leave or altruistically remain in torrent, lets be real who the fuck seeds

Requesting, sending file chunks

Requesting chunks

- At any given time, different peers have different subsets of file chunks
- Periodically, alice asks each peer for list of chunks that they have
- Alice requests missing chunks from peers, rarest first, as it ensures you get all the packets necessary download entire file, and allows you to seed out rarest chunk

Sending chunks: tit-for-tat

- Alice sends chunks to those four peers currently sending her chunks at highest rate
    - Other peers are choked by alice
    - Re-evaluate top 4 every 10s
    - Whoever gives more gets more only top 4 get
- Every 30 seconds randomly select another peer, start sending chunks
- Optimistically unchoke this peer
- Newly chosen peer may join top 4

Free-riding quiz

Suppose todd joins a bittorrent but he doesn't want to upload any data to other peers, and todd says he can receive a complete copy of the file that is shared by the swarm, is his claim possible?

Answer: possible however it would take a very long time, he is just waiting for optimistical unchoke, since he is not giving anything he will only be receiving through the optimitical unchoke

Getting rid of the server/tracker

Distribute the tracker information using a distributed Hash Table

A DHT is a lookup structure

- Maps keys to an arbitrary value
- Works like a hash table

Distributed Hash Table (DHT)

- DHT a distributed P2P database
- Database has key,value pairs
- Distribute the pairs over the millions of peers
- A peer queries DHT with key and DHT returns the values that match the key
- Peers can also inert their key/value pairs

Challenges

- How do we assign key/value pairs to nodes
    - Convert each key to an integer
    - Assign an integer to each peer
    - Put key/value pair in the peer that is closest to the key
    - Assign key to the peer that has closest id
    - Closest is the immediate successor of the key
- CIRCULAR DHT
    - Each peer only aware of immediate successor and predecessor
    - Overlay network
    - Can only move clockwise
    - Create shortcuts
        - Each peer keeps track of ip addresses of predecessor, successor and shortcuts
        - Reduces load from 6 to 2 messages
        - Possible to design shortcuts so we get O(logN) neighbours with O(logN) messages sent in query
- How do we find them again quickly?
- What happens when nodes join/leave
    - Peer churn,
    - When a peer leaves abruptly we need to reset the DHT

We need to know how a DHT works, we need to know how does it maintain randomness and that if there are N trackers chance of you chosen is 1/N

DHT makes it so that it is very hard to pinpoint a service/torrent to a specific tracker, instead of making it based on a certain user in control of management, each peer can now be the tracker all random

Video streaming and CDNS (content distributor network)

Video traffic: major consumer of internet bandwidth

Youtube has ~1 billion users,

Netflix has ~75 million Netflix users

Our main challenge is scalability with that many users, note a megaserver is not scalable a single point of failure is never scalable

Another challenge is heterogenety, making products suited to the users

Solution is having a distributed application-level infrastructure

Multimedia video

Video sequency of images displayed at constant rate

24 frames a second

Digitial image is an array of pixels each pixel represented by bits

Coding: use redundancy within and between images to decrease the number of bits used to encode image

- Spatial is within image
- Temporal, from one image to next

CBR – constant bit rate

Video encoding at fixed rate

VBR – variable bit rate

Video encoding rates as amount of spatial, and or temporal coding changes

Examples

MPEG 1, 1.5mbps

- MPEG2 3-6mvps
- MPEG4 often used in internet <1mbps

Streaming stored video:

A video server has a server of stored videos.

Streaming multimedia: DASH

DASH: Dynamic, Adaptive Streaming over HTTP

Server:

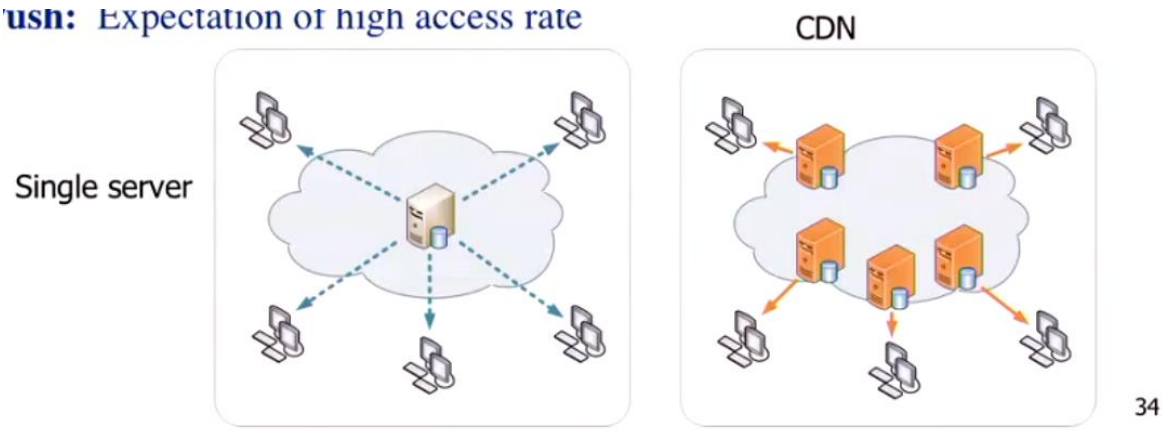
- Divides video file into multiple chunks
- Each chunk stored, encoded at different rates
- Manifest the file, providing urls for different chunks

Client:

- Periodically measures server-to-client bandwidth
- Consulting manifest, requests one chunk at a time
    - Chooses maximum coding rate suistainable given at current bandwidth
    - Can choose different coding rates at different points in time (depending on available bandwidth)
    - Intelligence at client determines when to request chunk so that buffer starvation or overflow doesn't occur, determines what encoding rate to request at (the higher the quality the more bandwith needed), and where to request chunk. Can request from url server that is close to client or has highest available bandwidth

CDN – content distribution network

A single server will never scale up



Caching and replication as a service amortise cost of infrastructure,

Goal is to bring the content closer to the user

To do this we can have large-scale distributed storage infrastructure (Servers) administed by one entity

Combination of pull (caching) and push (replication)

- Pull directs result of clients request
- Push expectation of high access rate (despacito needs to be worldwide while a content creator only seen in France can stick in France)

To check if a service are hosted by CDNs use dig and check for akami which is used for content distribution

2 options for CDN

Enter deep: put the CDN node within networks isp

- Close to users
- Used by akami at thousands of locations

Bring home

- Smaller number (10's) of larger clusters in POPs (point of presence) near access network (at internet exchange point IXP)
- Used by limelight

Enter deep

➔ Deploy node at isp

Bring home

➔ Deploy node at ixp

Enter deep gives least network delay since its already within ISP.

Google uses a mix of bring home + enter deep

Isp is 1 hop away

IXP might be multiple hops

CDNS store copies of content at CDN nodes.

Eg. Netflix stores copies of madmen

Subscriber requests content from CDN

- Directed to nearby copy retrieves content

This is an over the top approach where we disregard the underlying network that supports this network

Transport Layer

We are now moving down a layer

Network layer provides path from one end host to another, however comes with

- No guaranteed path
- No reliablility
- No guaranteed transfer rates
- Only guarantees to give best effort to move network across

Transport layer only works from an end to end mechanism, only comes to play in end system, routers do not contain transport layer, only end system acknowledges

Transport services and protocols

- Provides logical communication between app processes running on different hosts
- Transport protocols run in end systems
    - Send side: breaks app messages into segmenets that passes to network layer
    - Receive side reassembles segments into messages, passes to application layer
    - Exports services to application that network layer does nto provide

Why a transport layer?

- Reliability needed!
- Multiplexing/demultiplexing
    - Makes communication between processes at hosts
    - Transport layer can identify the correct process for request through sockets

Multiplexing/demultiplexing

Multiplexing at sender:

- Handle data from multiple sockets, add transport header (later used for demultiplexing)
- Allows for identification of which process is running

Demultiplexing at receiver

- Use header info to deliver received segments to correct socket
- Allows mapping for application/hosts

Connectionless demultiplexing

- Recall socket has host qand port number
- When creating datagram to send to UDP socket, must specify destination ip address and destination port number,
- UDP only cares for destination

-When host receives UDP segment

- Checks destination port # and directs segment to socket with that port number

Create the sockets, based on UDP.

- Only differentiate socket by port number not IP
- Multiple machines can communicate to same socket
- Server only looks at destination port number, this is only in UDP

Connection oriented demux

- TCP socket identified by a 4-tuple
    - Source IP
    - Source port number
    - Destination IP
    - Destination port number
- Demultiplex
    - Receiver uses all four values to direct segment to appropriate socket
- Server host may support many simultaneous tcp sockets
    - Each socket identified by its own 4-tuple
- Web server shave different sockets for each connecting client

In UDP identification based only on port number

In TCP identification is based on the 4-tuple

As soon as TCP handshake is over, server will create its own connection under the server

Connect call will create a separate connection for you under server process

All process go to same ip address, however each service has different port number. Destination port number = 80 are demultiplexed to different sockets

Tcp creates different sockets to communicate with each client. And the different sockets have the same port #

Servers wait at open ports for client requests

Hackers often perform port scans to determine open, closed and unreachable ports on candidate victims

Several ports are well known

- <1024 are reserved for well known apps
- Apps also use known ports
    - MSQL uses prot 1434 UDP

    - Sun Network File System (NFS) 2049 (tcp/udp)
- Hackers can exploit known flaws with these known apps

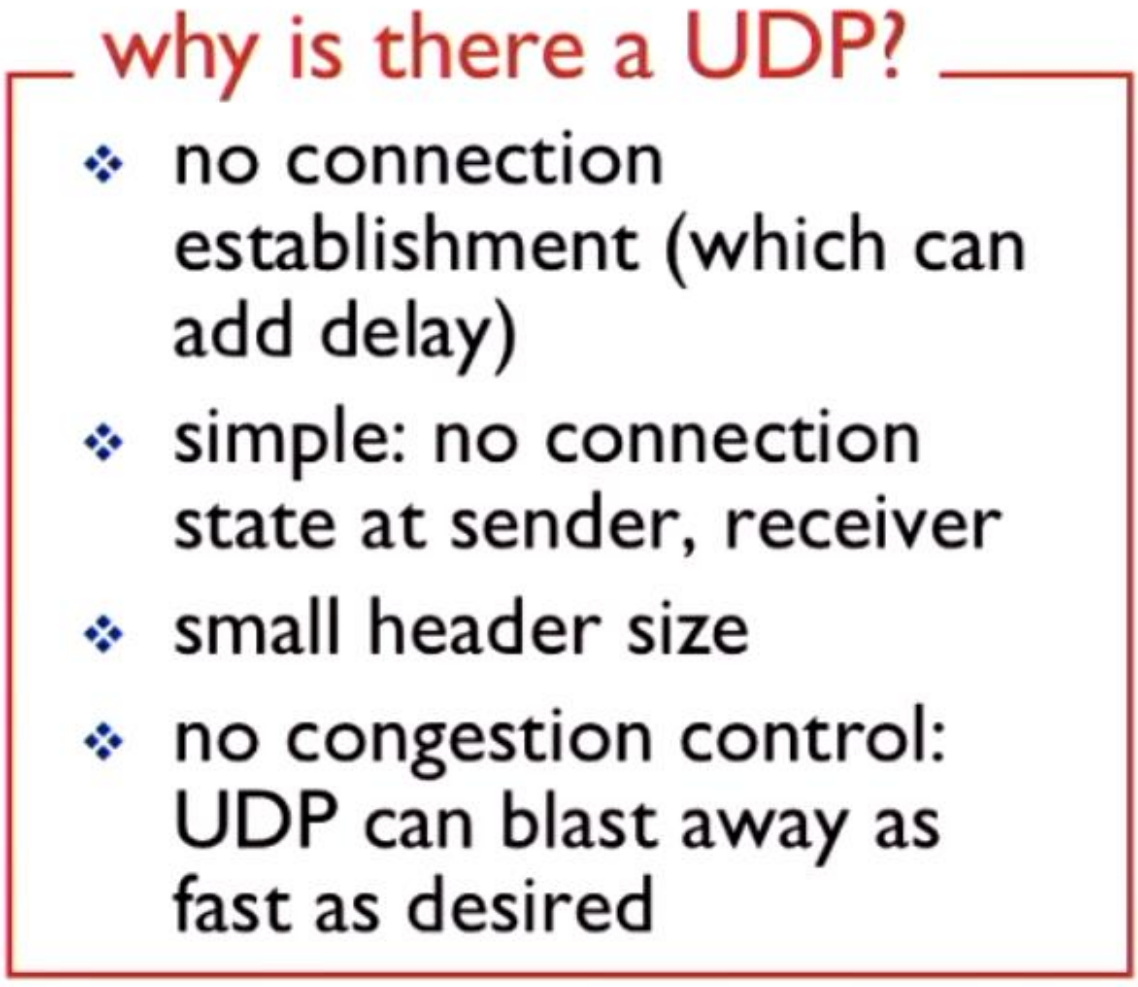
NMAp,superscan can perform port scans

UDP -> user datagram protocol

- No frills, bare bones internet protocol
- Best effort service

UDP header only requires

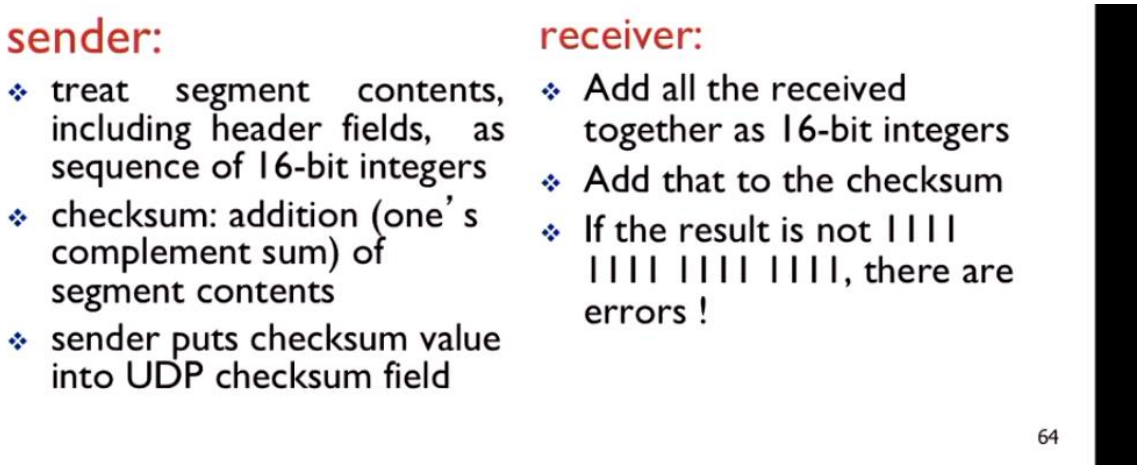
- Source port – 2 bytes
- Dest port – 2 bytes
- Length – 2 bytes
- Checksum – 2 bytes
- 8 byte total for header in UDP



UDP has a lot lower overhead than TCP, lower delay

UDP checksum

- Detect bit errors in transmitted segment
    - Router memory errors
    - Driver bugs
    - Electromagnetic interference



-
- Udp steals ip checksum method → layer violation
- But header contains the checksum itself

Mid sem 25 question, all MCQ 1 answer only

Internet checksum example

Whenever we get overflow, we take it and add it to the least significant bit, then at the end we take the one's complement of our number to get the check sum

UDP applications 8 byte header only allows multiplexing + de-multiplexing with port value

- Quick request/response (DNS, DHCP)
- Network management (SNMP)
- Routing updates (RIP)
- Voice/video chat
- Gaming especially with FPS

If udp receives a header and check-sum fails, it will simply just drop the packet. UDP offers no reliability and is a stateless protocol

Error correction is unnecessary in UDP (periodic messages)

Udp allows us to send out data even if server is down, just no reply/no error

Principles of reliable data transfer

Reliable transport

All the bad things best effort can do is

- A packet is corrupted (bit errors)
- A packet is lost
- A packet is delayed
- Packets are re-ordered
- A packet is duplicated

Reasons bit errors can occur in a packet

- Network interface card is corrupted -> send out rubbish data
- Or maybe router stores packet in buffer in queue, and when it is being read leads to bit errors

Reasons packets can be re-ordered

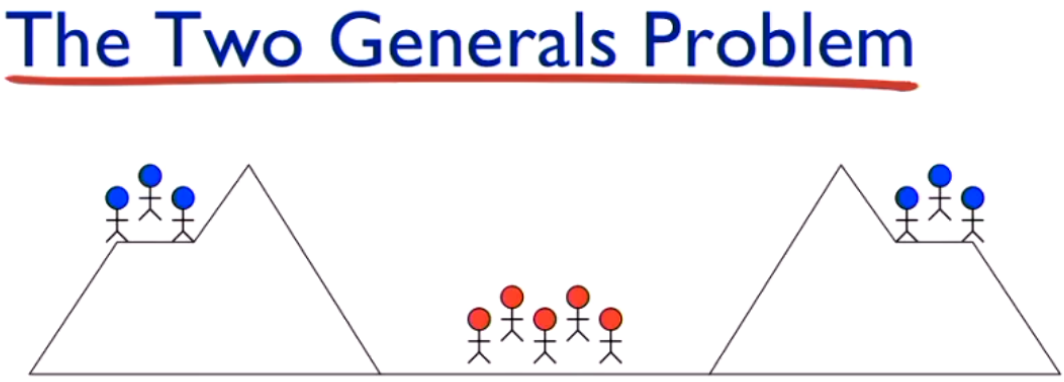
- This is dependant on the path the packet takes to reach the host, if paths are congested leads to this

Reasons a packet can be duplicated in network

- Following reliable protocol, if ack doesn't come back in time and packet is re-transmitted, however it comes back later, we get both packets

The two general problems

- Two army divions surround an enemy
- Each army is led by a single general
- Both must agree when to simultaneously attack
- If either side attacks alone there is defeat
- Generals can only communicate via messangers, and messangers may be captured



Transport layer is built on IP layer, which is unreliable, a best effort service, transport layer can provide reliability on its own building on top of the unreliable ip layer, we can implement our own reliability. We are using UDP but implementing the reliability of TCP.

tCP in a nutshell

- Hey bro I got this folder I wana transfer to XXX.com

- TCP goes, bro give it to me and don't worry about a thing
- All your packets arrive in order, with no bit corruption and will be guaranteed to arrive
- TCP is a reliable protocol

TCP is built on IP which is an unreliable best effort service

When application sends data it will assume it is a reliable protocol, so it sends an RDT (reliable data transfer), when we deal with ip then it assumes unreliable so does unreliable sends

Reliable data transfer basics

- Incrementally develop sender, receiver sides of reliable data transfer protocol (RDT)
- - consider only unidirectional data transfer
- Control info flows in both directions
- Channel will not effect ordering of packets

1 guy sends, other guy only allowed to send back acknowledgements.

RDT 1.0 reliable transfer over a reliable channel

For a channel to be perfectly reliable we require

- No bit errors
- No loss of packets

However the transport layer does nothing since the channel is assumed reliable

Rdt 2.0 channel with bit errors

- Underlying channel may flip bits in packet
    - Checksum to detect bit errors
- Receiver has to provide feedback ACKNOWLEDGEMENT BICH
- We need feedback to know current status of state
- To recover from this error we can simply re-transmit the packet, in human nature if we mess up in conversation we repeat ourselves
- Negative acknowledgements (NAKs) receiver tells sender the packet had errors
- Acknowledgement assumes packet is ok receiver says it's a okay!
- Retransmit on naks

Now we have 2 extra measures for reliability

1. Set checksum to check for errors
2. Re-transmission of packet

RDT 2.0 has a fatal flaw which can be really bad

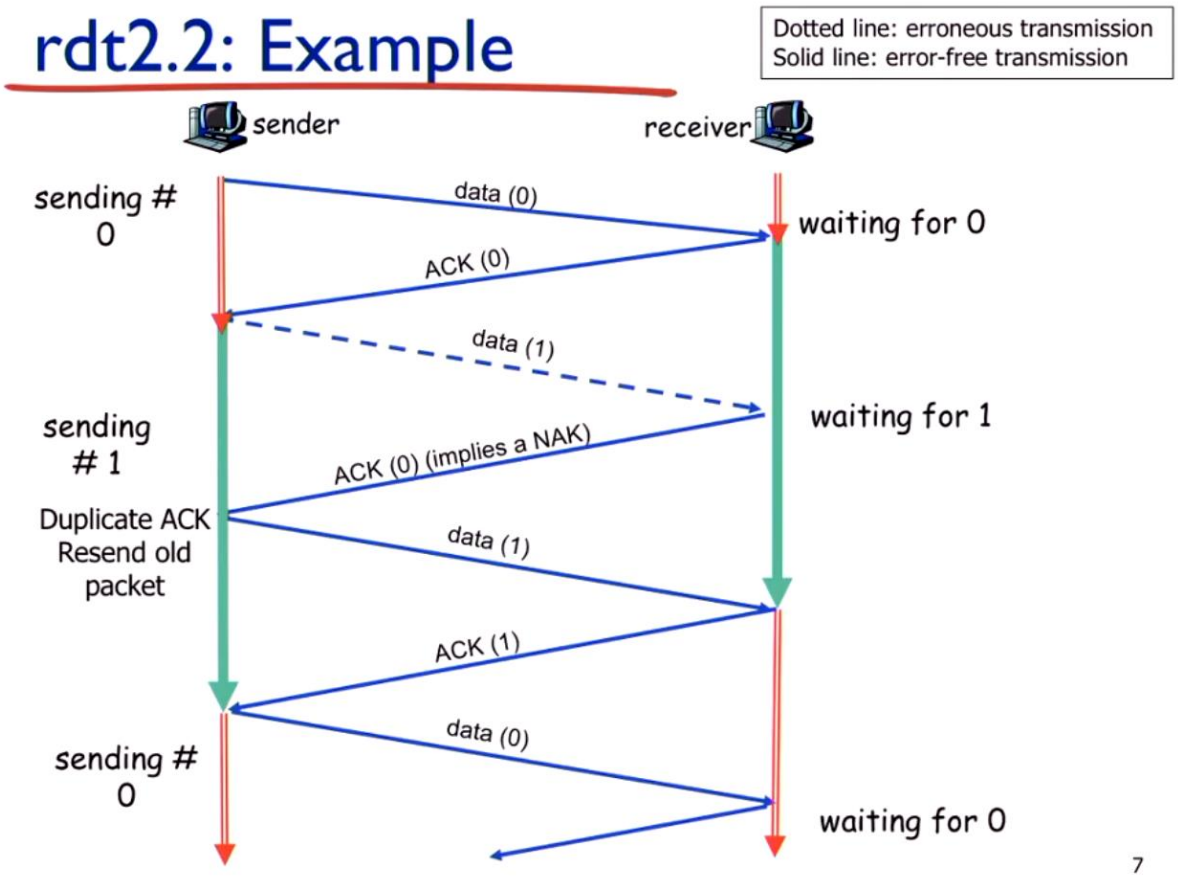
- Ack/nack response can also corrupt as above! Same chance it will corrupt!
    - Sender won't know what happened at receiver
    - Can't just retransmit possibly a duplicate, a corrupted ack could be re-transmit!
    - Put a checksum on the ack/nack, now the sender can know if the feedback is corrupted!

- Now what happens if the checksum on ack/nack fails?
    - Sender re-transmit packet if any corruption to feedback
    - Sender adds sequence number to each packet
    - Receiver discared duplicates, you can make a set where u do not add any duplicate sequence number! → RDT2.1
- This is the stop and wait protocol or alternating bit protocol
- We send a packet
- Wait for ack/nack
- And continue
- Keep transmitting till ack before we go to next packet
- Sequential protocol, we cant send bursts of packets!

RDT2.2 a nak free protocol

- Same functionality as rdt2.1 however only uses ACKS
- Instead of nak, receiver sends ack for last packet received okay
    - Receiver must explicity include seq# of packet being acked
- Duplicate ack at sender results in same action as nak, retransmit current packet

Basically now the receiver puts a sequence number,



Now if we send packet 1 and we get back ack0 we know that it was a nack since we didn't get ack1, so we transmit

All current versions of rdt assume no packet loss only assuming bit error

Rdt3.0 channels with errors AND loss

New assumption:

Underlying channel can also loose packets

- Both data/acknowledgements
- Checksum, seq#, retransmissions are helpful but we need more!!

Packets can be lost through

- Queue being full in router
- Link failure
- Network failure
- Routier failure → ip checksum failure

We need a timer!

Sender will wait a reasonable amount of time for acknowledgement

- Retransmits if no ACK within timeout period
- If packet is just delayed, retransmission is simply a dupliocate who cares
- Receiver must specifically specify seq# of packet being ack'd
- Requires a countdown timer that resets each send

Rdt 3.0 doesn't cater for packet re-ordering

Stop and wait – best care t = RTT + L/R

Rdt 3.0 has horrible performance

Performance of rdt3.0

- rdt3.0 is correct, but performance stinks
- e.g.: 1 Gbps link, 8000 bit packet and 30msec RTT:

$$D_{trans} = \frac{L}{R} = \frac{8000\ bits}{10^9\ bits/sec} = 8\ microsecs$$

- U_{sender}: *utilization* – fraction of time sender busy sending

$$U_{sender} = \frac{L/R}{RTT + L/R} = \frac{.008}{30.008} = 0.00027$$

- RTT=30 msec, 1KB pkt every 30.008 msec: 33kB/sec thruput over 1 Gbps link

- Network protocol limits use of physical resources!

Pipelined protocols

Pipelining: sender allows multiple, “in-flight”, yet to be acknowledge packets, send n-packets back to back, and wait for acknowledge for all N packets.

- Range of sequence numbers must be increased
- Buffering at sender and/or received
- Two generic forms of pipelined protocols: go-back-n, selective repeat

Stop and wait, 1 packet at a time super bad efficiency imagine 50000 packets

Pipelining can increase efficiency by massive amount N depending on burst of packets

We want (L/R)/(RTT+(L/R)) as close to 1 as possible 100% efficiency

Go-Back-N

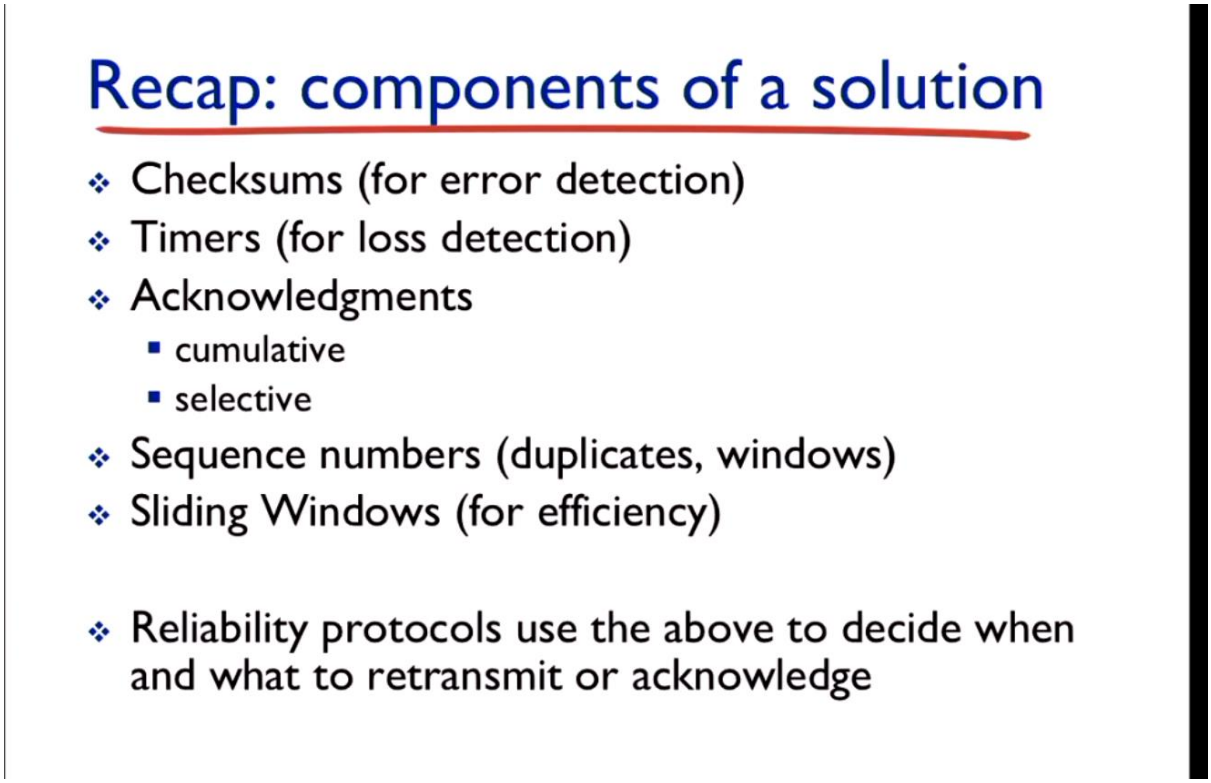
- Sender can have up to N unacked packets in pipeline
- Sender has only a single timer for oldest unacked packet, when timer expires it will re-transmit all unacked packets, timer will be on first packet in window, so packets are sent in order guaranteed! Drops all out of order packets
- No buffer at receiver, out of order packets are dropped
- Receiver only sends cumulative ack, doesn’t ack new packet if there’s a gap

Selective repeat:

- Sender can have up to N unacked packets in pipeline
- Sender has a timer on each unacked packet, when timer expires, retransmit only that unacked packet, → leads to mis-ordering!!
- Buffer at receiver, can accept the out of order packets, and will insert in order
- Receiver sends individual ack for each packet

GBN → sliding window of size N super ez

For selective repeat – window size must be less than or equal to half the size of the sequence number space



Components of a solution for reliable transport

- Checksum for bit error detection
- Timers for loss detection
- Acknowledgements in two forms
    - Cumulative GBN
    - Selective
- Sequence number for duplicates and windows
- Sliding windows for efficiency
    - GBN
    - Selective repeat

What does my boi TCP do

- Checksum exact same! Uses ip checksum. Same as udp, checksum bi-directional for acks aswell
- Sequence numbers are byte offsets!
    - Tcp numbers based on the size of packets and offset, similar to array in memory,
    - Eg. Packet size 100 10 packets,
    - Sequence 1 byte 0, sequence 2 byte 100,
    - We refer to our segmenet by the starting byte
- Segment sent when
    - Segment is full → max MSS (maximum segment size)
    - Not full, but time out while adding data to segment
- Receiver sends cumulative acks just like GBN opop

TCP segment size

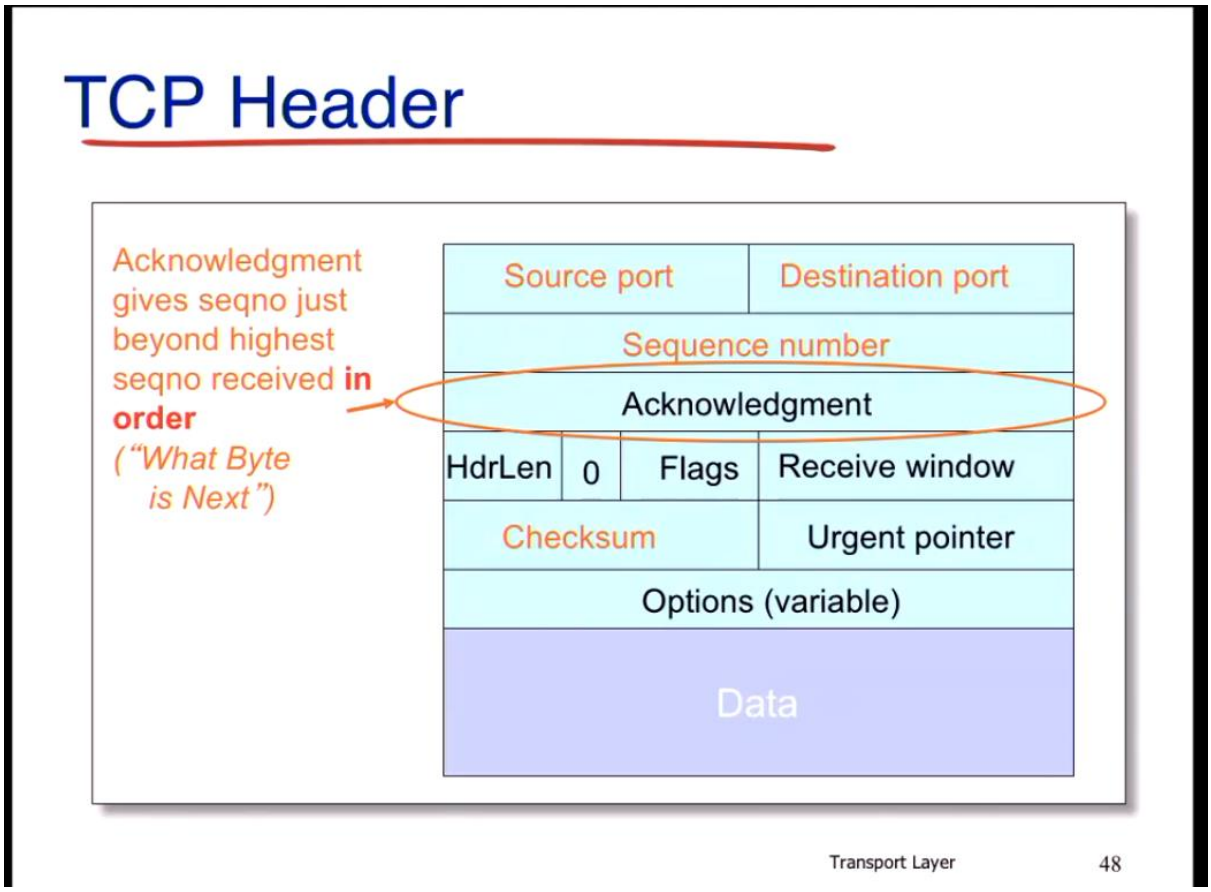
MTU (maximum transmission unit) is the total amount of data the ip layer can hand over to ethernet

(IP sitting ontop of datalink layer). If MTU is 1500 bytes, we need to take off amount for 20 byte ip header which has TCP data, then we also need to take 20 bytes off from the TCP header, then maximum segment size = 1500-20-20 = 1460 bytes

Sequence numbers

- Byte stream number off first byte in segment's data
- Sequence number = ISN+k, where ISN = initial sequence number and k = size of segment
- Now ack sequence number = next expected byte = sequence number + length of data

ACK in tcp just like GBN is the last byte received correctly



TCP Header

Piggybacking

- With piggybacking we can send a response + ack in the same message
- In M4 think of two speeding cars 1 is response 1 is ack

Receiver can buffer out of sequence packets like selective repeat

TCP is a hybrid of GBN (cumulative acks) and selective repeat (buffer out of order sequence)

Sender maintains a single retransmission timer just like GBN, and retransmits on timeout doesn't re-transmit things in the buffer

TCP round trip time, timeout

We need to adapt our timer to the network conditions, if it's too high we get long delays, if its too short, too many re-transmtis, we don't want a static timeout, we want a dynamic timeout system.

Longer than RTT which varies

estimatedRTT = (1-a)*EstimatedRTT + a*sampleRTT

- A has typical value of 0.125
- Our estimated RTT is already calculated
- Sample RTT is one sample sent out

Exponentially average, weight is assigned to history really similar to raising wam cant just raise in bursts

Estimated RTT is not enough we need more, a safety margin

DEVRTT = (1-B)*DEVrtt + B *(sampleRTT-estimatedRTT)

Typically B = 0.25

Timeout interval = estimatedRTT + 4*DevRTT ← safety margin

TCP FLOW CONTROL

There is a limited size buffer for TCP socket receiver, at each message the receiver will also inform the sender on the state of the buffer, this is called the receive buffer. While we ack we also send our state of buffer so that the sender can send in a controlled manner that doesn't overflow the buffer!

There are numerous reasons for buffer sending information to the sender

- Out of order packet delivery
- Application process data very slowly slower than it receives

At the very initialisation 3-way handshake, we will inform each side the state of the buffer.

Extreme case is when the buffer is full which results in blocking sender since queue size == 0. To escape the blockage, we can send a probe packet and wait till queue size > 0. Size = 1 byte.

Receiver will inform of the free buffer space remaining in the rwnd value at the TCP header.

CONNECTION MANAGEMENT

Initialising the connection is done through a sender/receiver handshake.

- Agree to establish connection
- Agree on the connection paramters
- Think of like a constructor in java

ISN (initial sequence number)

- Problem with starting at 0 is that it is a security issue this is why we choose a random ISN

- The security issues include
    - Allow easier brute force, connection interception if it starts at 0 all the time
    - Reduces chance that new connections with the same paramaters would conflict

2-way handshake

- Request connection
- Accept connection

This doesn't work for TCP

- If there is a delay between the accept connection and a re-transmit occurs from client → server; and later on we close the connection before the re-transmitted connection arrives, we would have a half open connection since the client would already be terminated but the connection was accepted at the server!

What we use in networking is a 3 way handshake

First a synbit is sent from client → server with the initial sequence number for establishing connection, this syn bit is a flag for checking if the connection has been created.

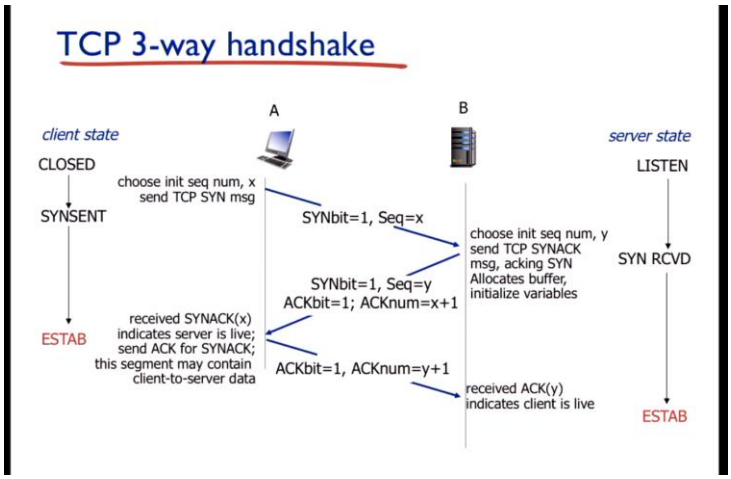
Client state goes from closed->synsent

Server will then receive the syn + header and choose its own sequence number and its own syn bit and the acknowledgement number.

Server state goes from LISTEN->SYNRCVD

The server will also now reserve buffer at this state, will initialise all its variable (ISN) and the client number.

Finally the client will know its SYN and seq was correctly acked and it will also ack the received header from the server and both states go to ESTABLISHED SEE SS BELOW



IMPORTANT THAT THE SYN AND SYNACK STATEMENT CONTAIN NO DATA ONLY UNTIL ESTABLISHED WE CAN SEND DATA, u don't tell someone wana talk and start talking before they say okay

The final client response in handshake CAN send back some data. Since establishment occurs earlier for client than the server.

Since the syn takes 1 byte of data, we will still increment our ack number for the syn! Convo transcript logs are numbered from start

SYN CONSUMES 1 SEQUENCE NUMBER SYN X → ACK X+ 1

What if the SYN Packet gets lost jfc

We will resend, but we have no sample RTT's to have a timeout, we cannot estimate our RTT.

- Use default value of 3 second for default timeout, RFC1122, RFC2988, however more recently now we use 1 second for our default timeout initially then increase to 3s for subsequent attempt (RFC 6298)

SYN Loss and WebDownloads,

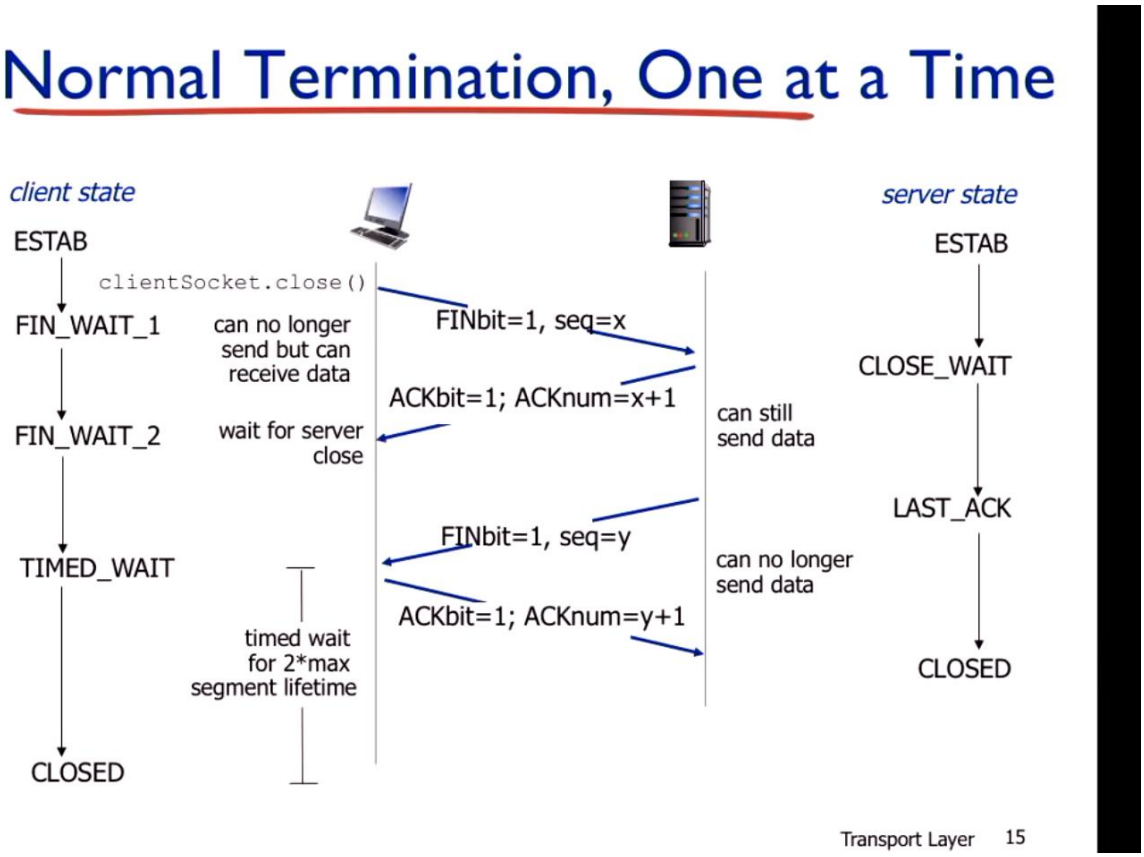
- User impatience spam refreshing actually makes page load faster since wont have to wait for timeout LUL

Closing a TCP connection

- Client and server each slode their side of connection
    - Send segment with TCP segment with fin bit turned on FIN = 1
    - This last segment can also contain some data
    - Can no longer send but CAN receive data
- We cannot send more data after our FIN segment is sent, its two independent servers so 1 can communicate even tho other 1 has FIN'd

See below diagram

For termination one at a time.

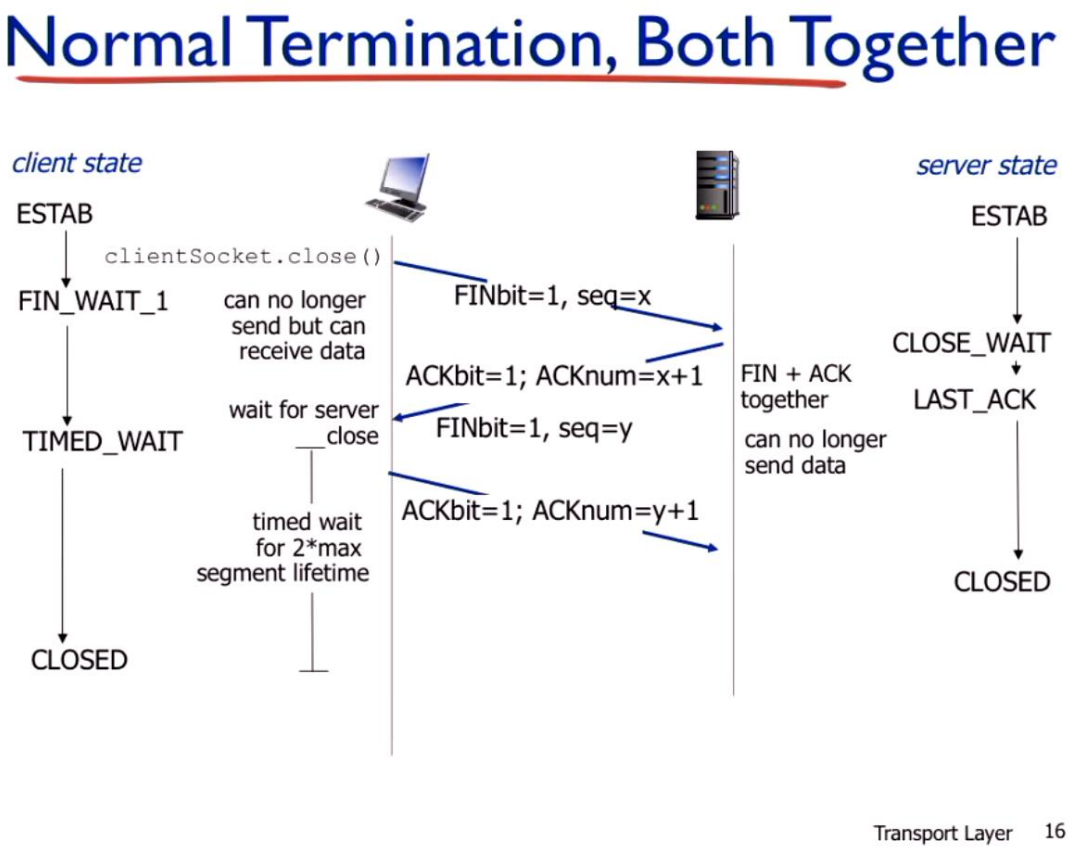


The reason we wait for 2*MSL at the timed_wait and closed is that we it's a safety window incase there is error in reliability for the final acknowledgement.

4- way handshake for close in one at a time close

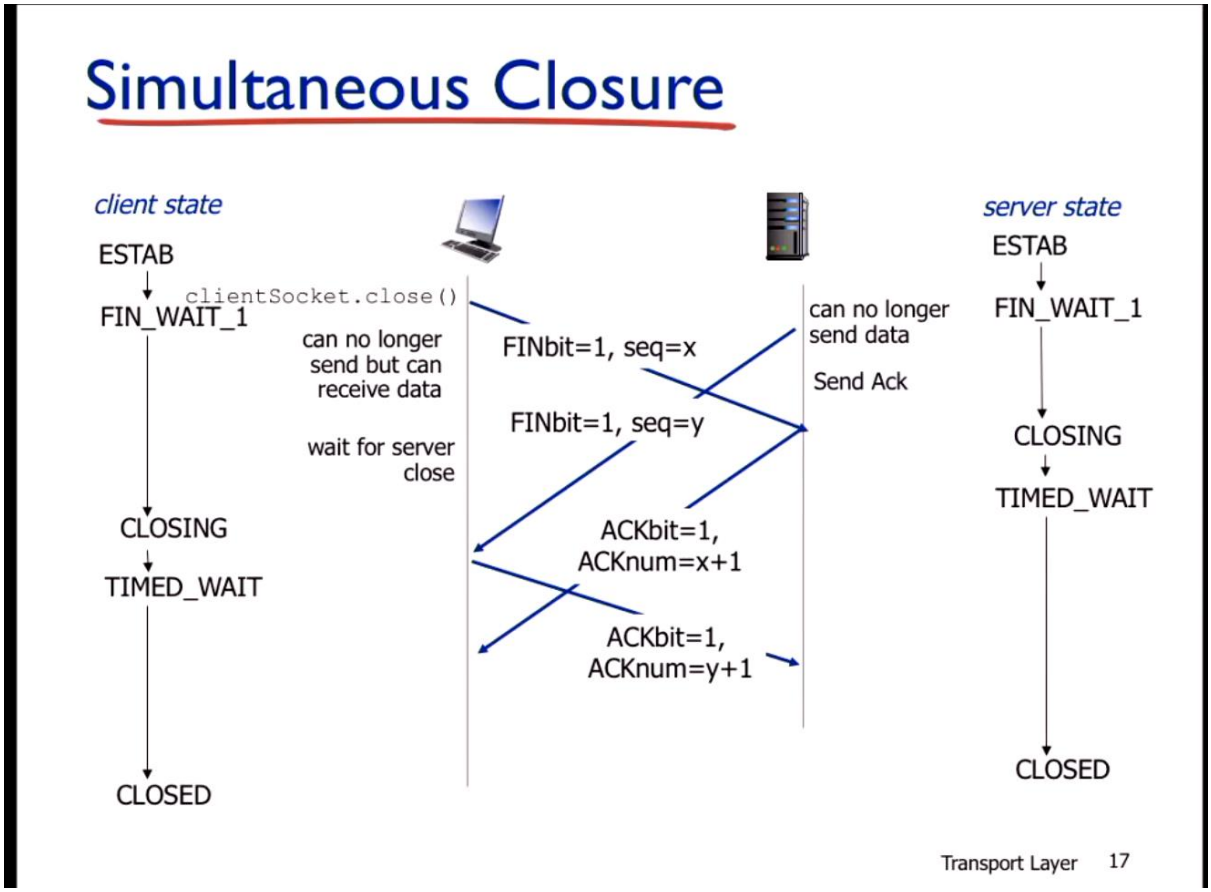
3-segment closure is when we utilise piggybacking when both terminate at the same time, on the ack it will also send its fin bit aswell

---



Simultaneous close

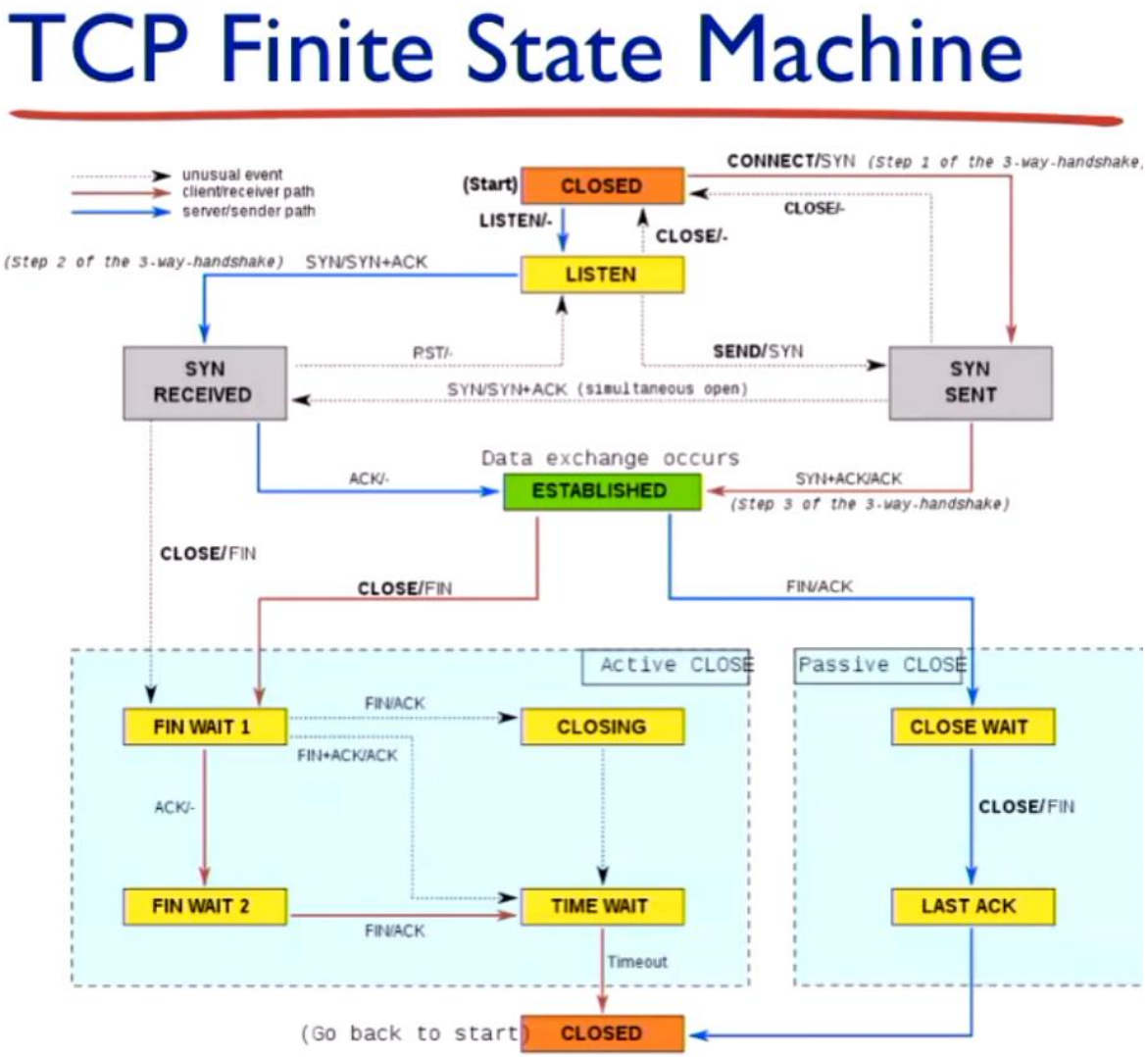
Both generate a fin at the exact same time



Abrupt termination

- Incase there is error or process crash there is a reset flag to re-establish the connection, this is done through the RESET (RST) bit

LEARN ALL THE CLOSE STATES IN RELATION TO THE FSM BELOW EXAMINABLE IMPORTANT



SYN FLOODING/ SYN ATTACK

- Creating fake fin packets
    - Destination is ip address of victim
    - Source is some spoofed ip address
- Victim host will create TCP states for each SYN creating a fuck load of half opened connections
- Acks never come back cos the villain is like “later”
- After timeout the connection state is freed, but we still waste resources
- Can cause overwhelm for the victim

To work around this we can

- Increase our connection queue size
- Decrease timeout wait (not advised)
- Firewalls: list of known bad source ip addresses (firewall on ip layer)
- TCP SYN Cookies

- DDOS ATTACK PREVENTED

This is a denial of service attack

TCP SYN Cookies

Once we receive the SYN we don't just create a connection state, we will make an ISN that is a hash of the source + destination ip address and port number of SYN Packet. Replies back with SYNACK containing init_seq, no state made and now we wait for that ack before we make state, its like a check to make sure that people don't send syn then leave and create states, brilliant

Principles of congestion control

Congestion:

- This is based on sender side in flow control its based on client side!
- Congestion is when too many sources send too much data too fast for the network to handle
- This leads to
    - Lost packets (buffer overflow at router)
    - Long delays (queue delay in router buffers)
    - A top 10 problem in life fuck congestion
- We don't want to pile onto the congestion, re-transmission because of congestion would in turn increase congestion!!
- It is a exponentially growing effect and can effect other routers involved in a path!
- Congestion will back propogate on its path if not dealt with correctly
- Congestion will
    - Increase latency
    - Increase loss rate
    - Increase re-transmission
    - Wastes capacity of traffic that is never delivered
- The knee for congestion is the point after throughput slowly increases and delays will increase fast!
- The cliff is the point after the queue is full and will drop packet, throughput = 0, no packets go through and delays approach infinity (congestion collapse)
- We aim to keep our connection rate to the knee that is safer approach
- Some more adventurous protocols try to optimise to cliff, always a tradeoff.

In 1986 rates fluctuated on and off from 32Kbps to 40bps, and the proposed solution was that senders should voluntary modify their send rate.

There are two forms of congestion control

1. End-end congestion control – hosts will measure delays/losses and base congestion off this
    a. No feedback from network
    b. Congestion inferred from end host loss/delays
    c. Approach of TCP
    d. DOES NOT WORK ON NETWORK ASSISTED CONGESTED CONTROL
2. Network assisted congestion control – routers control congestion based on queue
    a. Routers provide feedback to the hosts
    b. Single bit indicating congestion (TCP/IP ECN)
    c. Explicit rate for sender to send at

Remember our first principle KEEP THE CORE SIMPLE, don't put everything in main class, the Network Assisted Congestion Control puts more complexity onto the routers whereas end-end congestion control distributes the workload on hosts.

We want to keep the core simple!

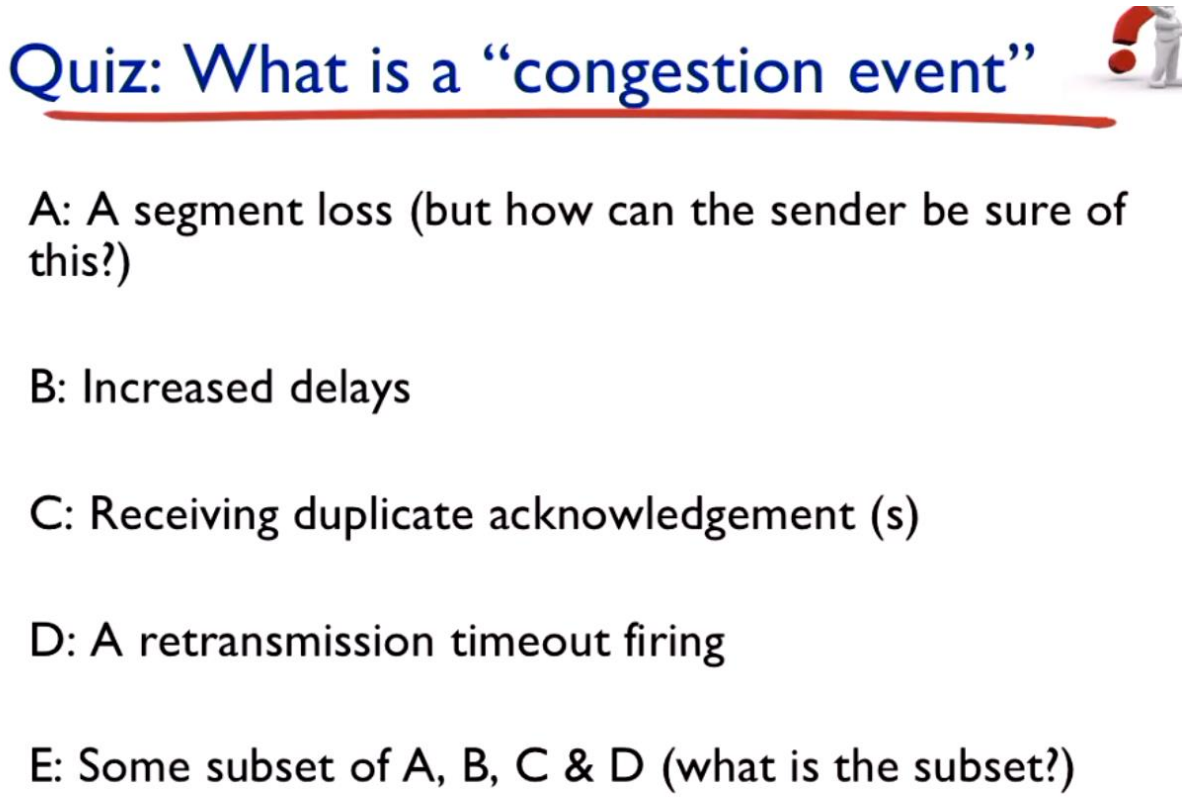
TCP Congestion Control

- TCP has a window (Window based protocol like GBN AND SR)
- The rate of congestion ~= (CWND/RTT) bytes/second and based on the result we can dynamically modify our window size
- TCP will play around with the window sample size till it finds the cliff
- We also have a flow control window (receiver window RWND)
    - How many bytes can be sent without overflowing buffer for receiver
    - Determined by the receiver report to sender
    - Sender-side window = minimum(CWND, RWND)
        - we can assume for lecture that RWND >> CWND

CWND

- how many bytes can be sent without overflowing the router
- computed by the TCP congestion control algorithm, test and increase/decrease like the COMP2121 lab 5 PWM lab

1. how does a send detect congestion
2. how does a sender adjust its sending rate
    a. modify the window size we are send



my answer I think is B (indirectly) C D (directly)

Quiz: How should we set CWND?

A: We should keep raising it until a "congestion event" then back off slightly until we notice no more events.

B: We should raise it until a "congestion event", then go back to 1 and start raising it again

C: We should raise it until a "congestion event", then go back to median value and start raising it again.

D: We should sent as fast as possible at all times.

A is dangerous, C is most suitable and B is also true mix of B and C

Fast re-transmit

If we receive 3 duplicate acks, we re-transmit our window without waiting for timeout this is to prevent un-necessary waiting

Not All Losses the Same

- Duplicate ACKs: isolated loss
  - dup ACKs indicate network capable of delivering some segments
- Timeout: much more serious
  - Not enough dup ACKs
  - Must have suffered several losses

How does the sender adjust its sending rate?

- Depends on the case

Rate adjustment

Basic structure:

- Upon receipt of ack we increase rate
- When detect loss we decrease rate – cliff! (step back from cliff)

How we input/decrease depends on the phase of congestion control we are in

- Discovering bottleneck bandwidth (at the very start of TCP connection)
- Adjusting to bandwidth variations → check for conditions, playing at the cliff area to find an optimal zone

2 zones

Zone 1 – knee

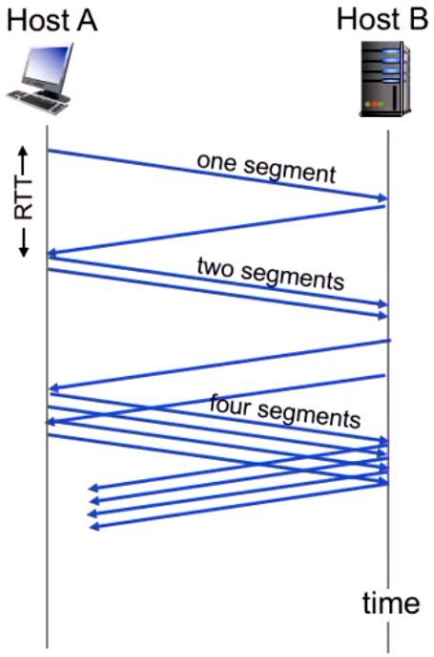
Zone 2 – cliff as seen above

Bandwidth discovery with slow start

- Estimate available bandwidth
    - Start slow for safety
    - Ramp up quickly for efficiency eg. 1, 2 ,4, 8, 16, 32 segments
    - This is a exploratory method

TCP Slow Start

- when connection begins, increase rate exponentially until first loss event:
  - initially **cwnd** = 1 MSS
  - double **cwnd** every RTT
  - Simpler implementation achieved by incrementing **cwnd** for every ACK received
- *summary:* initial rate is slow but ramps up exponentially fast



We keep going until loss, for each RTT we double the rate.

Adjusting to varying bandwidth

- Slow start gave estimate of available bandwidth
- Now we want to track variations around this current value (probing) and back off known as Congestion avoidance

TCP uses additive increase multiplicative decrease (AIMD)

AIMD

Approach is to increase transmission rate (window size), probing for usable band width until loss occurs

Additive increase (linear increase)

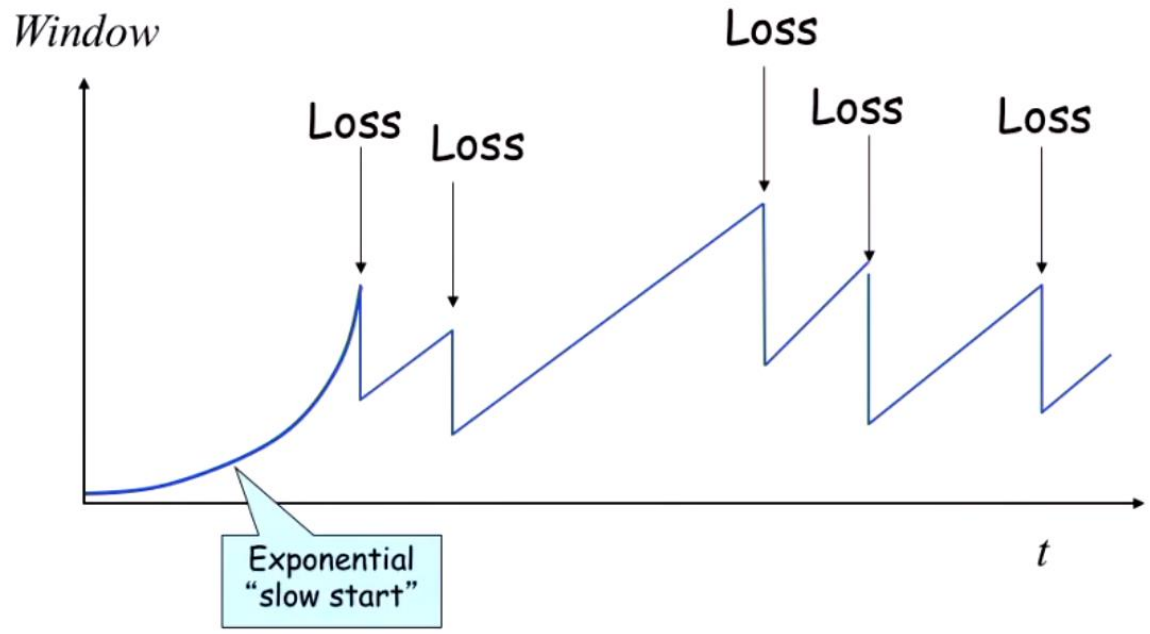
- increase CWND by 1 MSS every RTT until loss
- For each successful RTT, CWND++
- For each ack, CWND = CWND + 1/CWND

Multiplicative decrease

- Cut CWND in half after loss

Peaks in AIMD are different as bandwidth can vary.

Finally TCP window growth is shown below



Slow start vs AIMD

We have a threshold value where we switch from slow-start to congestion avoidance phase.

On timeout, slow start threshold = CWND/2

To implement we need a state at sender

CWND – initialised to a small constant

SSTHRESHOLD (initialised to large constant)

Also, dupACKcount and timer for this

SLOW START PHASE

If CWND < ssthreshold

CWND++;

CONGESTION AVOIDANCE

Else

CWND = CWND + 1/CWND;

If dupACKcount = 3

Fastretransmit();

//ssthreshold = CWND/2

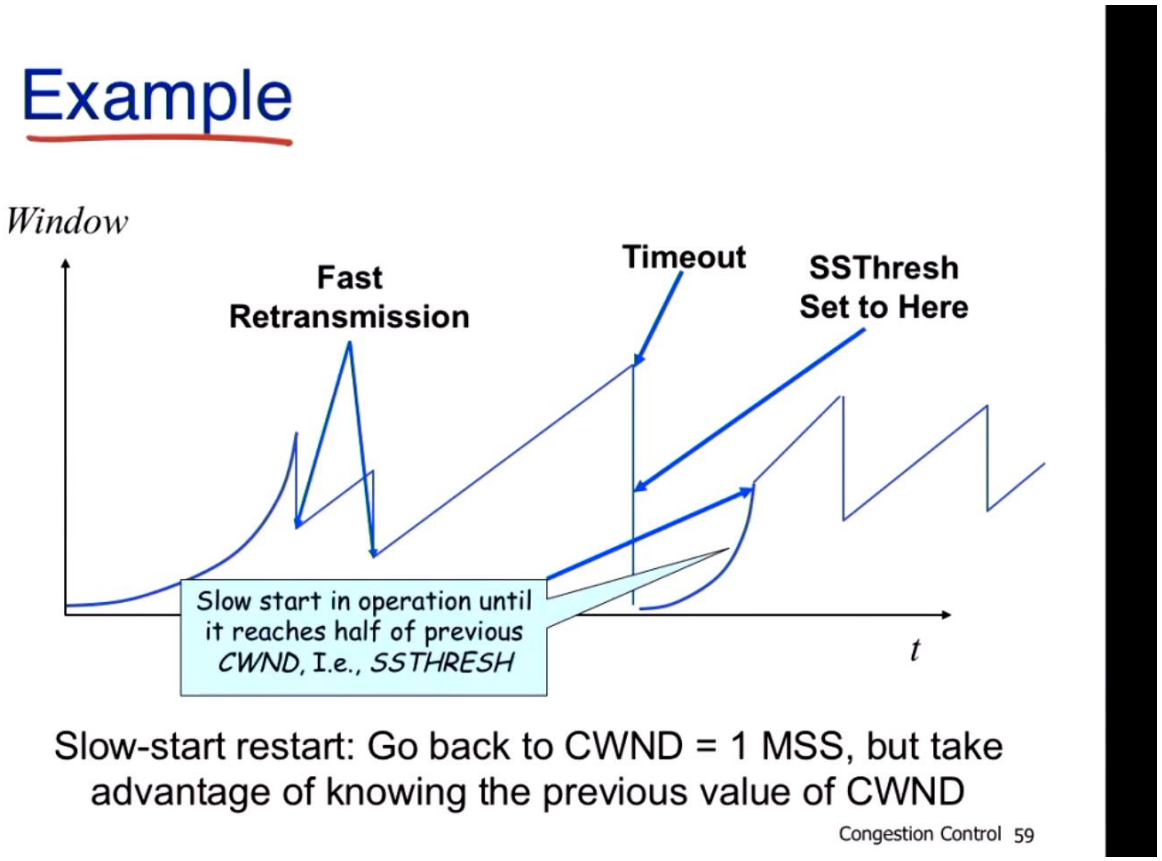
//CWND = CWND/2

Event of time out

If(timeout)

Sshtreshold = CWND/2;

CWND = 1; (reset)



On 3 dup ack, drop CWND by half and SSTHRESHOLD = half CWND

On timeout, we reset and set SSTHRESHOLD = ½

Final phase is fast recovery

- Grant the sender temporary “credit” for each dupACK so to keep packets in flight
- Id dupACKcount = 3
- SShtreshold = CWND/2
- CWND = sshtrshold + dupACKcount
- CWND = CWND + 1 for each additional duplicate ACK

- Exit fast recovery after receiving new ack then set cwnd = sshtreshold

Timeline

- ACK 101 (due to 201) cwnd=10 dup#1
- ACK 101 (due to 301) cwnd=10 dup#2
- ACK 101 (due to 401) cwnd=10 dup#3
- REXMIT 101 ssthresh=5 cwnd= 8 (5+3)
- ACK 101 (due to 501) cwnd= 9 (no xmit)
- ACK 101 (due to 601) cwnd=10 (no xmit)
- ACK 101 (due to 701) cwnd=11 (xmit 1101)
- ACK 101 (due to 801) cwnd=12 (xmit 1201)
- ACK 101 (due to 901) cwnd=13 (xmit 1301)
- ACK 101 (due to 1001) cwnd=14 (xmit 1401)
- ACK 1101 (due to 101) cwnd = 5 (xmit 1501) ← exiting fast recovery

- Improved utilisation by pushing in more data, increasing utilisation

Different versions of TCP

TCP-tahoe

- CWND = 1 on triple dup ack and time out (fast retransmit not there)

TCP-RENO

- CWND = 1 on timeout
- CWND = CWND/2 on triple dup ack

TCP-newRENO

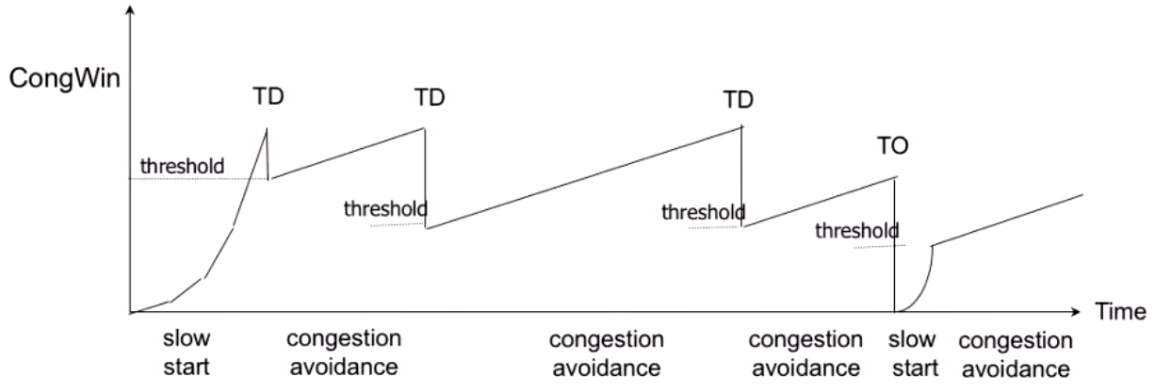
- TCP -Reno + improved fast recovery

TCPSACK (not in this course)

- Incorporates selective acknowledgements

This is TCP RENO

TCP/Reno: Big Picture



TD: Triple duplicate acknowledgements
TO: Timeout

This is not TCP new reno because on the falls, the growth is back to linear, if it was TCP new-R it would slow start after the fall and then go back to congestion avoidance

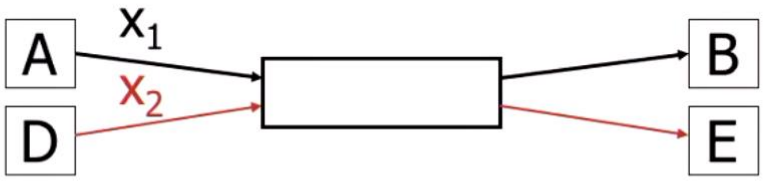
TCP Fairness (transport layer) → last TCP topic (RIP)

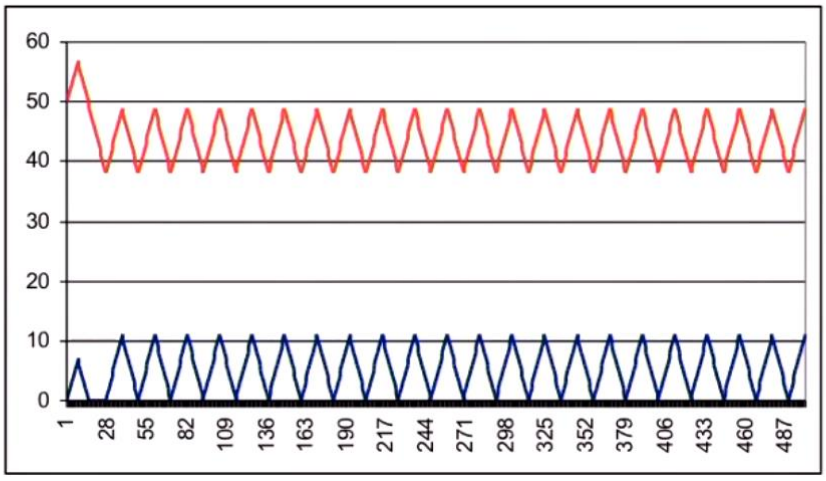
- If K TCP session share link at bottleneck of bandwidth rate of R, each should have a rate of R/K, divide bandwidth evenly

AIAD

All change is fair since movement is constant, oscillates at same slope till it hits efficiency, can never hit fair and efficient

AIAD Sharing Dynamics



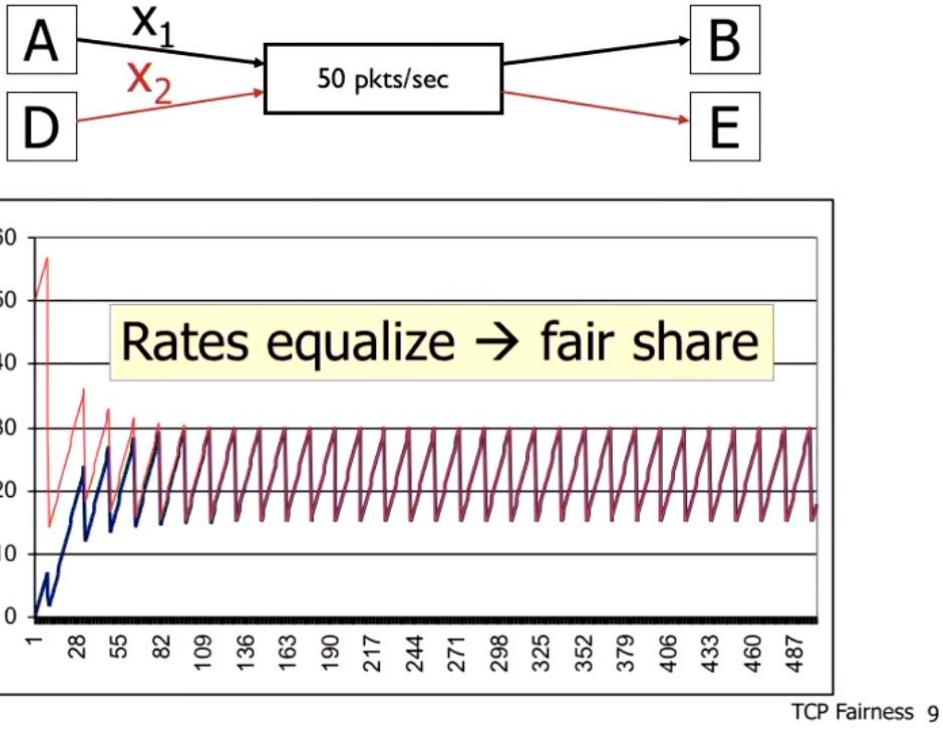


TCP Fairness 7

AIMD

Will slowly oscillate towards center of fairness + efficient

AIMD Sharing Dynamics



TCP Fairness 9

TCP throughput is inversely proportional to the RTT, and is also inversely proportional to the square root of the drop rate

TCP is unfair in the face of heterogenous RTTS

Loss not due to congestion!

TCP will confuse corruption with congestion → takes all loss as congestion and reduces congestion window which has negative effect on throughput, treats everything as congestion

Short flows also never leave slow start since there is not enough data to fully utalise TCP, for short flows its not fair, if we have 1500b file, it will slow ramp even thought most network are >1mb/s

Too few packets to trigger dupACKS on short flows

- Lead to timeout rather than fast-retransmit → latency

TCP fills up queues which leads to long delays,

- Short files wait long in queue if occupied

Three easy ways to cheat

- Increasing CWND faster than +1 MSS per RTT (customly written code)
- Opening a load of connections, it will designate bandwidth to the user based on what they are doing
    - Eg. If 1 person downloading a movie and another looking at webpages, will give more flow to movie man since it is fair looks at users as a bunch of processes
- Using a large initial CWND (slowly becoming standard with iw10 xD)

Why hasn't the internet suffered a congestion collapse yet?

- The network is robust enough to handle the few cases where some specialised people can cheat ahaha im gonna be a cheater

NOW FINALLY NETWORK LAYER

In 1968 ARPAnet (precursor to internet) used in army

Mid 1970's new networks emerged SATNet, packet radio, ethernet however they weren't linked together

1974, a protocol for packet network intercommunication which is now the foundation for the modern intern (Cerf and Kahn)

Routers forward packets from a source to destination and may cross separate networks along the way

All packets use a common Internet Protocol

- Any underlying datalink protocols
- Any high layer transport protocol

Network layer – present on every device! Except switches (datalink, physical) however there are some IP switches

Network layer transports segments from sender → receiver

On the sender side we encapsulate with the network header to create datagrams

On receiver side we deliver segments to transport layer (datagram → segment when header taken off)

Router examines header fileds in all IP datagrams passing through

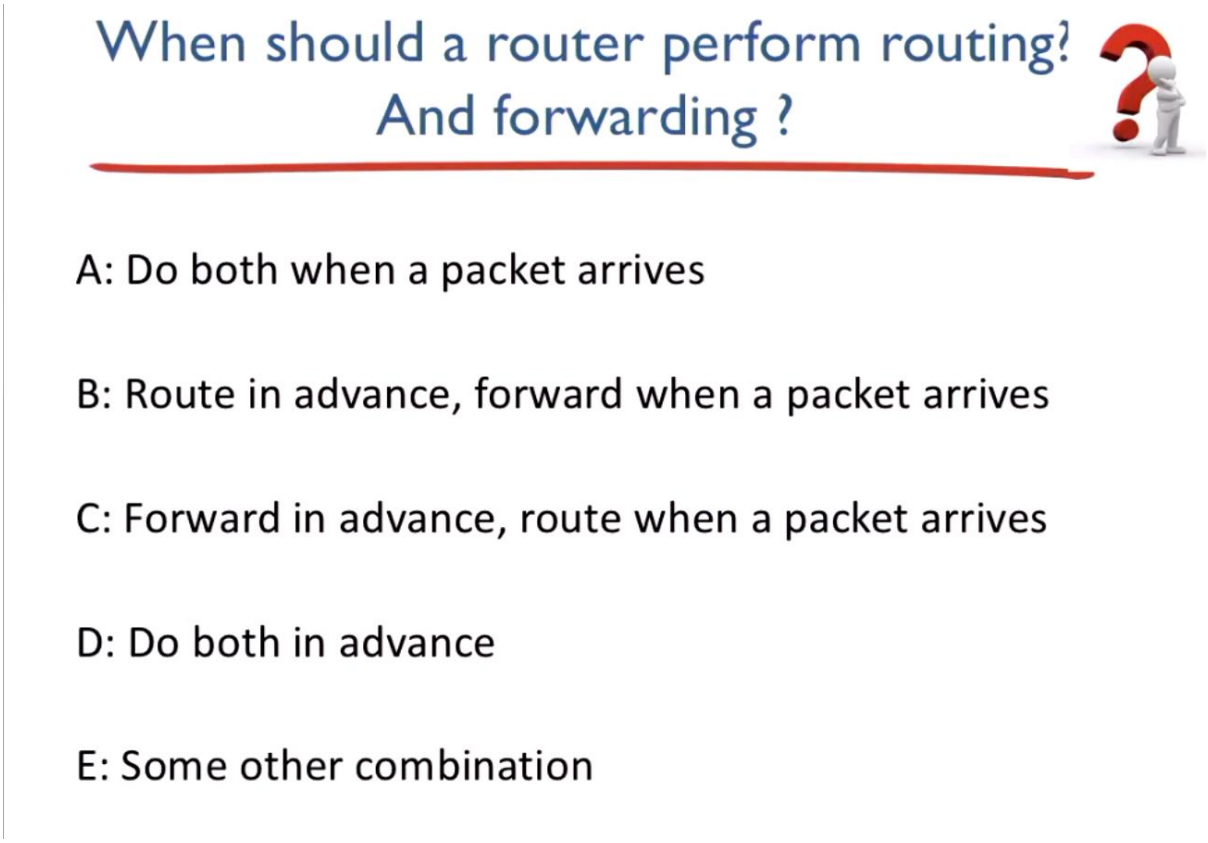
Two key network – layer function

- Forwarding
    - Move packets from router input to output
- Routing
    - Determine route taken by packet from source to destination (the algorithm using routing tables)

Example

Routing is planning trip from source to destination → put destination into google maps

Forward is the process of actually going thru (e.g. Driving) → following route in google maps



I'd say B seems most correct, A causes most delay (reactive) calculation times, C cant forward in advanced wtf its literally retarded

D is like wtf

Each router has a forwarding table

In router 2 different planes

Control plane part is where we calculate routing

→ how we get our forwarding tables

- Two approaches
  - Traditional routing algorithms implemented in router
  - Software defined networking (SDN) centralised remote servers

Dataplane part is where we do forwarding

→ sending on path

- Forward function

Per router control plane

- Individual routing algorithm components in each and every router interact in the control plane, so each router will make its own routing table and can share with its neighbours

SDNS

- A distinct typically remote controller interacts with local control agents (middle man)
- All forwarding tables are centralised and each router will have its table referenced from the control agent, allows less work in the router and router becomes simply just a forwarding device

Internet is a best effort service model (this is where transport layer comes in) TCP OP

Router architecture

- Routing processor (brain that calculates routes from routing algorithm) think CPU with ALU
- Router input/output ports
- High speed switching fabric

Routing management is done in software and operates in milliseconds, where as forwarding is done in a nanosecond timeframe, much faster to forward

input/output port have same structures but can be interswitched, exact same just changed functionality

queue available in the input port queue

switching fabric

- Memory based
- Bus based
- Cross bar

Switching rate: rate at which packets transfer from input → output

- Mesasured as multiple of line rate
- N inputs: switching rate N times line rate desriable so each line has the same speed

Switching via memory

- Traditional computers with switching under control of CPU
- Packet copied to system's memory
- Speed limited by memory bandwidth (2 bus crossings per datagram)

Switching via bus

- Datagram from input port memory → output port memory via shared bus
- Bus contention, switching speed limited by bus bandwidth

Switching via interconnection network

- Overcome bus bandwidth limitations

- Allows for concurrent packet transmission a lot faster

If fabric is slower than ports, this will lead to queuing at input port queues and maybe even loss

Output ports has a buffer for queueing, when data arrives faster at router than transmission speed

How much buffering?

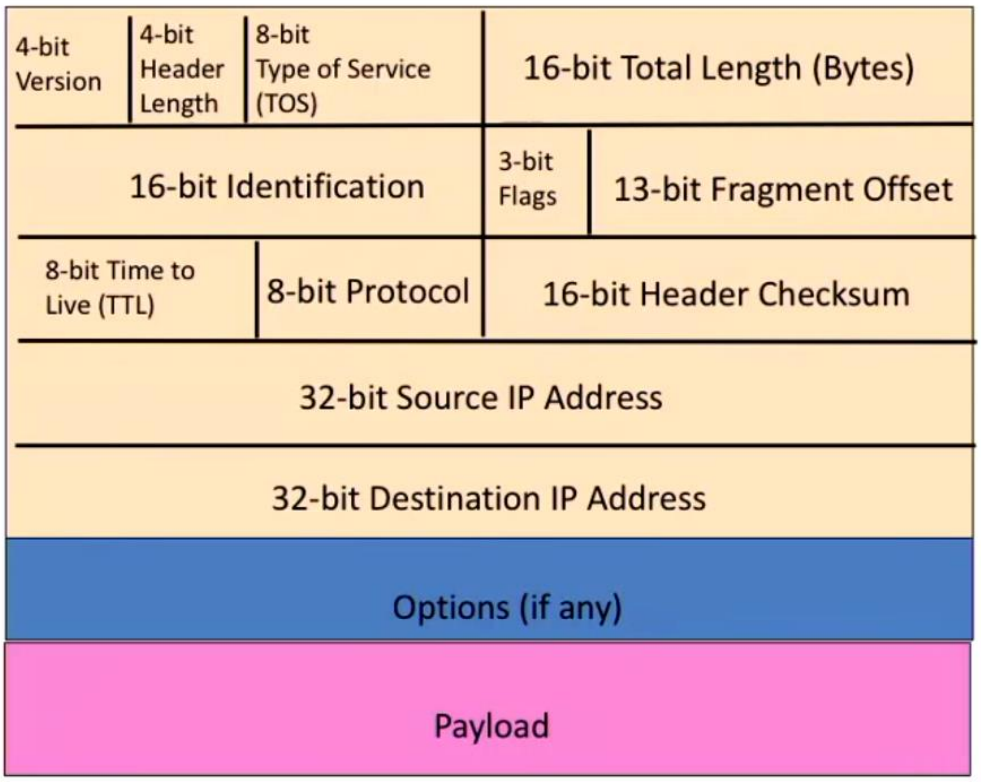
- Use bandwidth delay product, use typical RTT (250 msec)
- If our bandwidth = 10GbPs, we need to have a buffer of 0.250*10 gbs
- Or we can say with N flows we get (RTT - c)/(sqrt(n))

Scheduling mechanisms

- Scheduling chooses next packet to send out
- FIFO (queue) send in order
    - Discared policy, if packets arrives to full queue we need to know how to drop
        - Tail drop, drop arriving packet
        - Priority, drop lowest priority packet
        - Random, drop randomly (wtf bro no thx)
- Priority schedule, send highest prority queued packet
    - Each packet has its own priority from some comparator
    - Service the packets based on whichever one has highest priority (priority queue literally)
- Round robin scheduling
    - Cyclically scan class queues sending one at a time from each queue if available incase one queue empties out we wont have delays
- Weighted fair queueing (WFQ)
    - Generalised round robin
    - Each class gets weighted amount of service in each cycle, can set x amount of transmission for each queue

IP PACKET LAYOUT FOR IPv4

IP Packet Structure



4-bit version → IPv4 in this case, will change entire packet structure

4-bit header length → max of 15 → 4 byte word for each bit so max = 4*15 (size of header)

16-bit total length (bytes) = size of payload + header in total

Port numbers are maintained at the transport layers!, in IP (network) the ip addresses are used

Ip addresses → network layer

Port numbers → transport layer

32 bit source/destination ip address

8 bit protocol number, used to multiplex and de-multiplex and used to transfer to appropriate transport layer → contains the transport layer protocol (UDP/TCP etc.) look at hourglass, 6 = UDP, 17 = TCP. Identifies the protocol (multi-plexing at source) and allows for demultiplexing at the receiver

16-bit header checksum – same one's complement mechaninsm however only on header, not payload. This is because network layer is a best effort method, responsibility for payload is not in the network layer, handled in transport/application layer. Checksum ONLY APPLIED ON HEADER

8 bit time to live (TTL) – IP will consider this as a counter for number of hops maximum, eg. If val = 16 goes intor outer → hops and decrements counter so val = 15. Drops if val == 0. HOP BASED NOT TIME BASED. Will send error message back to router if it drops a packet. TTL 0 cannot be forwarded dropped at val 0, maximum number = 255. Stops routing loops if a packet gets lost (zombie packets)

If a router changes TTL on each hop, it will have to also change the checksum aswell, decrement TTL and decrement checksum aswell since we decrement TTL

Potential problems

- Header corruption
    - Solution: checksum
- Loops
    - Solution: TTL field (reactive measure to remove from loop) if a window of packets are coming around and all stuck in a loop (link will be lost forever), no new packet can be inducted into network, TTL will remove these packets
- Packet too large:
    - Solution: Fragmentation

LEARN THIS FOR EXAM!!!!!

- MTU = maximum transmission unit, maximum size of datagram than can be delivered to datalink layer(size of transport layer header + size of network layer header + payload)
- MSS = maximum segment size (size of transport layer header + payload)

If next hop MTU cannot handle the current MTU, think of moving something 1500 bits into a 1000 bit MTU. What we do is fragment.

On handshake, we will choose the MSS with the first and last hop. If MTU changes due to network hops, the routers begin fragmenting. They are then re-fragmented only at the destination, re-assembled at the destination.

The reason we re-assemble at the destination rather than at the next hop is that this will put a load on the network, we want to keep the core simple! If we do fragmentation and the three fragments take different paths we will not be able to re-fragment since they are all at different links. Each fragment will require a header on each fragment, transferring more data in than out. Increasing overhead cost.

My solution for fragmentation would be to send a arbitrary sized bit packet across all network to discover the link with the DF bit on, if ACK received, means we can use that size for our MTU, we can now slowly increment like the congestion avoidance

PATH MTU DISCOVERY

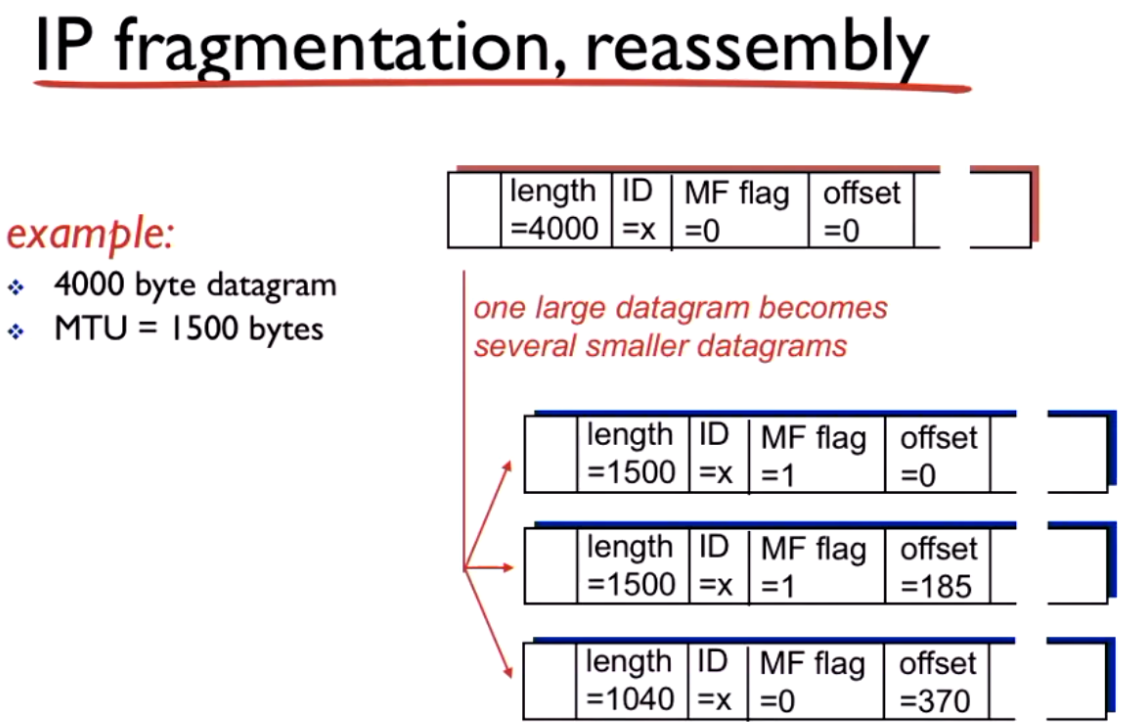
16 bit identification – to identify where the fragments belong to think of it as a reference to the group of fragments

3- bit flags (MF, DF)

- MF is more fragments to follow,
- DF is do not fragment

13-bit fragment offset

- Maximum length = 16 bit
- This is 13 bit field however, so we need to encode the 16 bit number into 13 bits
- We take total length field, and divide by 8
- 8 because 16-13 = 3, 2^3 = 8



4000-20 bytes = total size of payload, need to append 20 byte header to each fragment so MTU = 1480 + 20 bytes as max

IPv4 fragmentation procedure

- Fragmentation
  - Router breaks up datagram in size that output link can support
  - Copies IP header to pieces
  - Adjust length on pieces
  - Set offset to indicate position
  - Set MF (More fragments) flag on pieces except the last
  - Re-compute checksum
- Re-assembly
  - Receiving host uses identification field with MF and offsets to complete the datagram.

We can fragment fragments xD

IPv4 addresses

IP address: 32 bit identifier for host, router interface

Interface: connection between host/router and physical link

4 bytes dot separated eg. 192.168.0.2

Each byte has max value of 255 (2^8 – 1)

Ip limit 255.255.255.255 xD

IP address is associated with each interface, so network card or wireless network card in laptop. Each interface NEEDS an IP address to communicate with traffic.

Network ⇔ subnet (for now ahaha)

All devices have their unique ip address, and so do router interfaces

First 3 bytes of a router represents the network, and the last byte represents the byte address

Eg. 223.1.1 as network and .x could be the host address

So router has 223.1.1.x

All devices would be 223.1.1.y

Interface connecting to network must have a common network part and a unique host part

How are interfaces actually connected? → learn in linked layer

Wired ethernet interfaces connected by ethernet switches (can be hierarchical)

Wifi connects through an access point (AP).

Ip address

Network part – high order bytes

Host part low order bytes

For 223.1.1.1

Network part = 223.1.1

Host part = .1

Final total = 223.1.1.1

A network is a device interface with same network part of the ip address so 223.1.1 is a network

Can physically reach each other without intervening router

Masking

- Given an ip address how can we identify the network an ip address belongs to
- We mask by specificying ip address and the mask bits constituting the network part
- Then we perform a bit-wise AND
- Eg. 223.1.1.0 AND 225.255.255.0
- Since our host part is 0, whatever we get will be the network part of the IP
- Since our mask is 1111.1111.1111.0000
- We will get back whatever is turned on, if u and anything with OXFF u get the original, this will return our network part!

Two special reserved addresses in network address only for broadcast/network address

Broadcast address is

- Network address + host part = 255

Network address is

- Network address + host part = 0

Hence our range is from 1-254

Network address allows us to maintain a single address for a network (routing purposes).

Broadcast address allows us to access all devices through single address.

We need both these abilities in order for routing functionality.

Original internet addresses

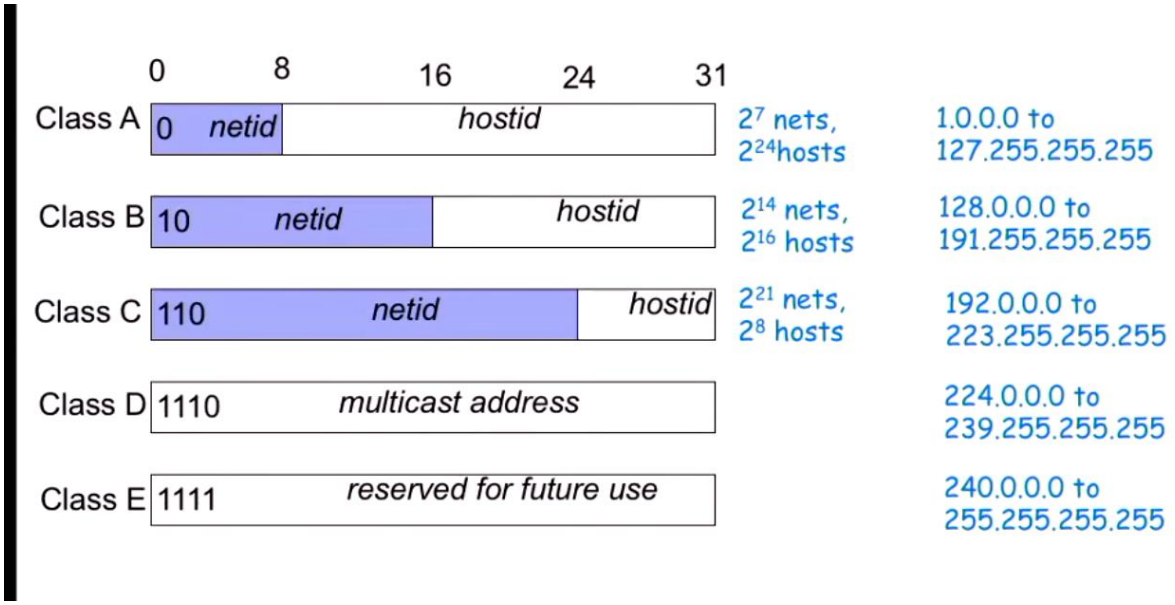
- First 8 bits network
- Rest of 24 bits = host address
- Assumed 256 networks were more than enough LOL boy were they wrong was complete opposite

Next they did class-ful addresses

Class A, network part = 8 bits, host id = 24 bits

Class B, network part 16 bits, host id = 16 bits

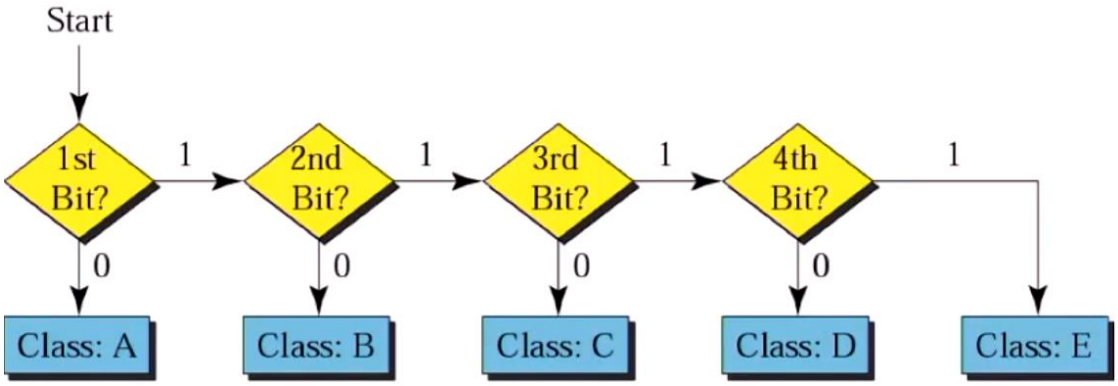
Class C network id = 24 bits, host id = 8 bits → more networks but each network limited to 255 hosts



This was used until the introduction of CIDR in 1993, this came with a problem that a network had only three preset sizes.

This mechanism made it distinguishable which class the address belongs to

- Is bit 1 = 0, if so class A
- If not is 2nd bit 0, if so class B
- If not is 3rd bit 0, if so class C and so on and so on



Under this model these questions arise

If an organization requires 6 networks each of size 30, does it have to buy 6 class C address blocks? What are the issues

Only choice available really is class C, paying for 254 host addresses for only 30 hosts, waste of money and resources. – subnet solution

An organization requires 512 addresses? How many ip addresses should it buy? What are the issues? – supernet solution

Now has to buy class B, paying for 65533 hosts addresses only 512 hosts, waste of money and resources.

Allowed no room for variety

Subnetting to the rescue

- Subnetting is the process of dividing the class A, B or C network into more manageable chunks that are suited to your networks size and structure
- Subnetting allows 3 levels of hierarchy
    - Netid
    - Subnetid
    - Hostid
- Original netid remains the same and designates the site
- Subnetting remains transparent outside the site

ISP gives you address and you manage it.

This allows us to sacrifice some host ID bits to gain network ID bits. We extend the number of bits representing the mask, which allow us to create subnetworks with 1 byte.

Quiz?

A company is granted the site address 201.70.64.0 (class C). The company needs six subnets. Design the subnets.

6 subnets need 3 bits

201.70.64.(5 bits), borrow left most 3 bits for subnet allows us to support 32 hosts, has to be inbetween cannot use the lowest + highest bit as they are reserved so now not only are we wasting 2 bits, but now we are wasting 2 addresses on each subnet so now we waste 8*2 ip addresses, since 8 subnets (even tho we only use 6) 16 ip addresses gone.

The company needs six subnets. 6 is not a power of 2. The next number that is a power of 2 is 8 (2^3). We need 3 more 1s in the subnet mask. The total number of 1s in the subnet mask is 27 (24 + 3). The mask is

11111111 11111111 11111111 11100000

or 255.255.255.224

Number of addresses in each subnet = 2^5

= 32

Network Layer 15

224 = 128+64+32

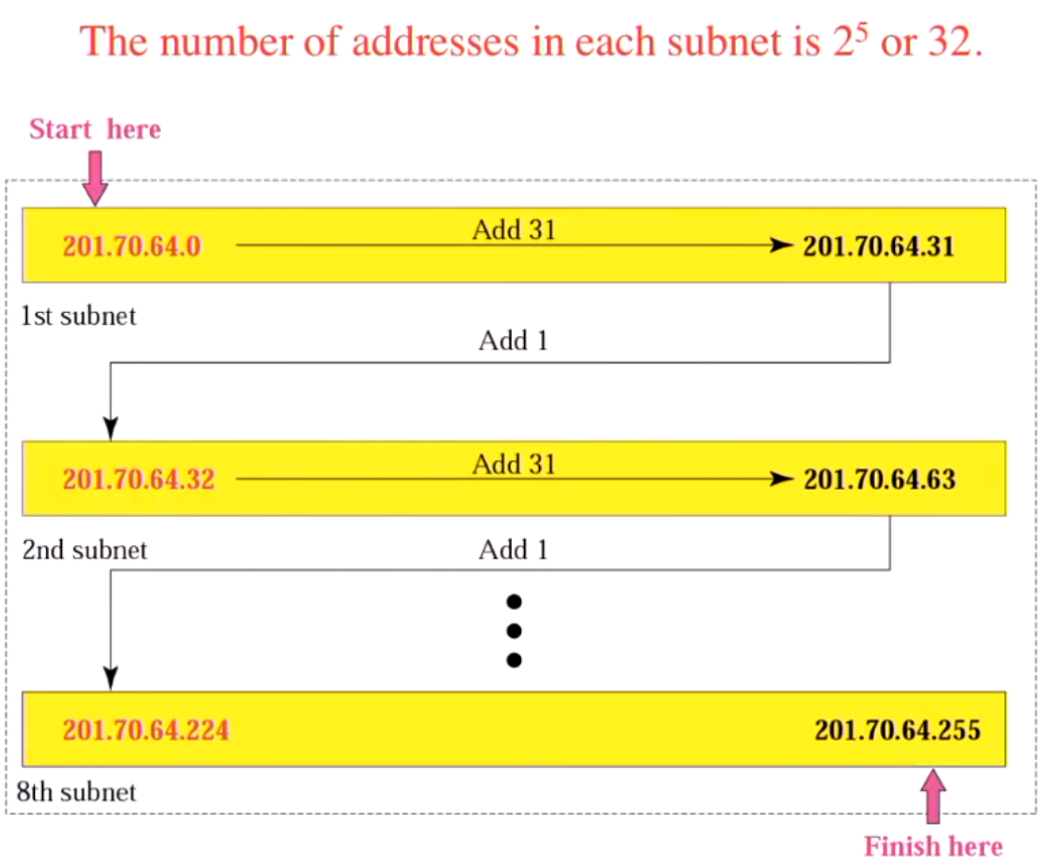
External mask for finding network = 255.255.255.0

Now internal mask = 255.255.255.224

Now in each subnet we have 5 bits left for host addresses = 32, with exception of 0 and 32 being in use already

First subnet now = 201.70.64.0 → 201.70.64.31

Second subnet now = 201.70.64.32 → 201.70.64.63 and so on and so forth!



Subnet is not visible to the ISP, ISP just gives up a block C address and we manage our subnets

Router knows internally we have subnetting and changes mask! Router will maintain our mask

Supernetting

- Traditionally one organisation one global net id
- Supernetting allows the addresses assigned to a single organization to span multiple closed prefixes
- Supernetting is one organisation, many netids
- With supernetting a single organisation is known to the rest of the world by multiple netids

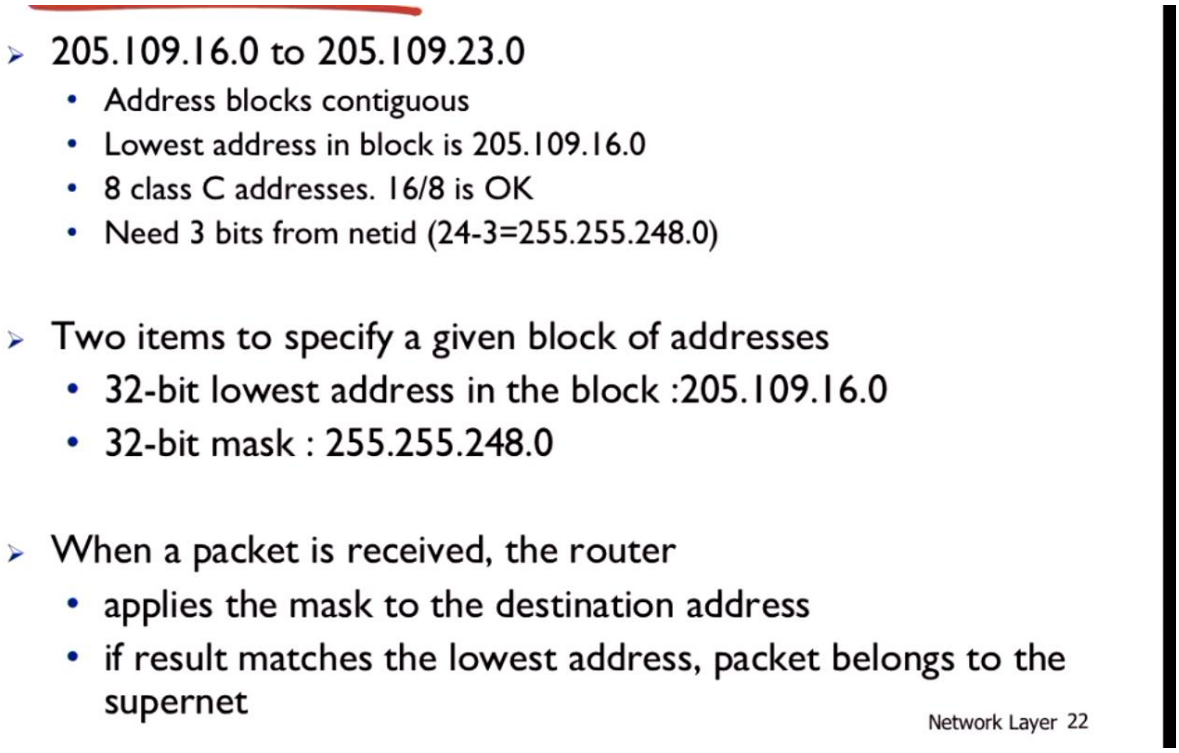
Example of supernetting

- An organisation needing 1000 addresses no longer needs class B and waste a majority of its address space, it can be allocated 4 contiguous unusuded class C addresses.
- Composite pattern!
- Can be extended into other classes
- Supernetting is the opposite of subnetting, in supernetting we take 3 bits from the network part now! Now our default mask would go to 225.225.248

Three rules for supernetting

- The number of blocks must be a power of 2 since binary
- The blocks must be contiguous so NO GAPS BETWEEN THE BLOCKS

- The third byte of the first address in the superblock must be evenly divisible by the number of blocks. In other words if the number of blocks is N, the third byte must be divisible by N



The third rule basically says the first lowest byte 16 in this case must be divisible by the number of total class C addresses we are combining if it was 205.109.17.0 would be INCORRECT COS 17 not divisible by 8

Now all we need is beginning class C address

First address in chain 205.109.16.0

And mask 255.255.248.0

248 = 11111 000

16 = 00010 000

23 = 00010 111

24 = 00011 000

17 = 00010 001

If we mask with 17, we will only get 16

If we mask with 23 it goes down to 16,

If we mask with 24, it goes down to 24

The lower 3 bits of a number will be disregarded from the mask, so we have to choose our start number to be within the upper remaining bits when we and our mask together, everything from 16-23 will collapse under to 16, as when we AND 248 with 16, it will collapse down

DID U GET THIS POINT AYE

When we apply our mask to our packet, it will fall to the lowest address in the block, it will always filter out to the correct network address

The correct network address is the lowest address in the sub/supernet or simple network.

Todays addressing CIDR

CIDR: Classless InterDomain Routing

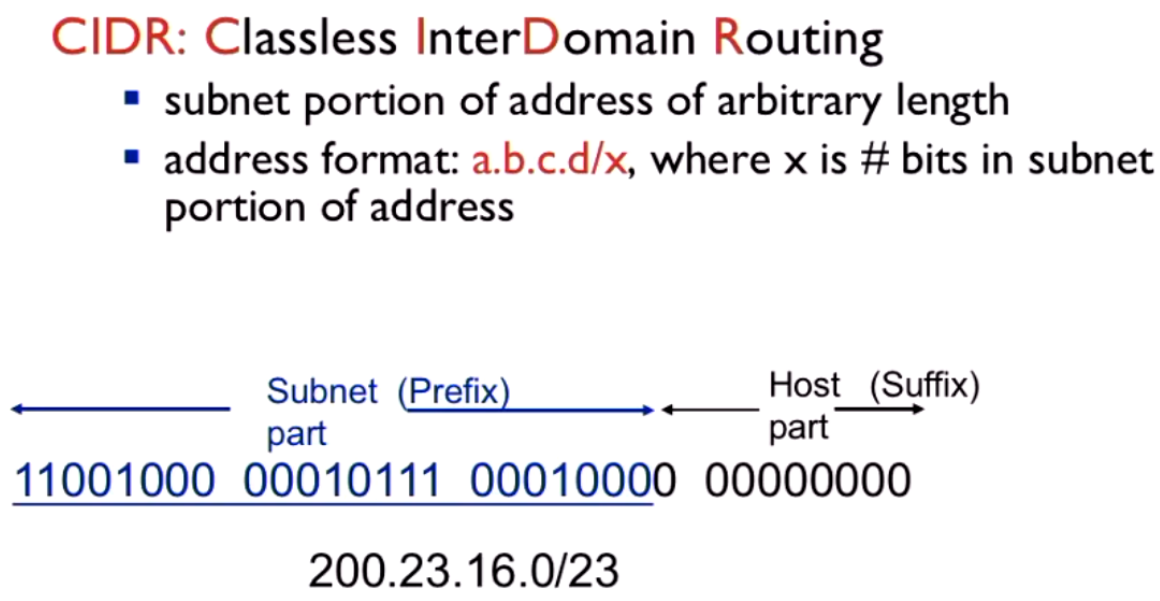
- Subnet portion of address of arbitrary length (arbitrary mask value)
- Address format: a.b.c.d/x, where x is the number of bits in the subnet portion of address

Subnet part = pre-fix

Host part = suffix

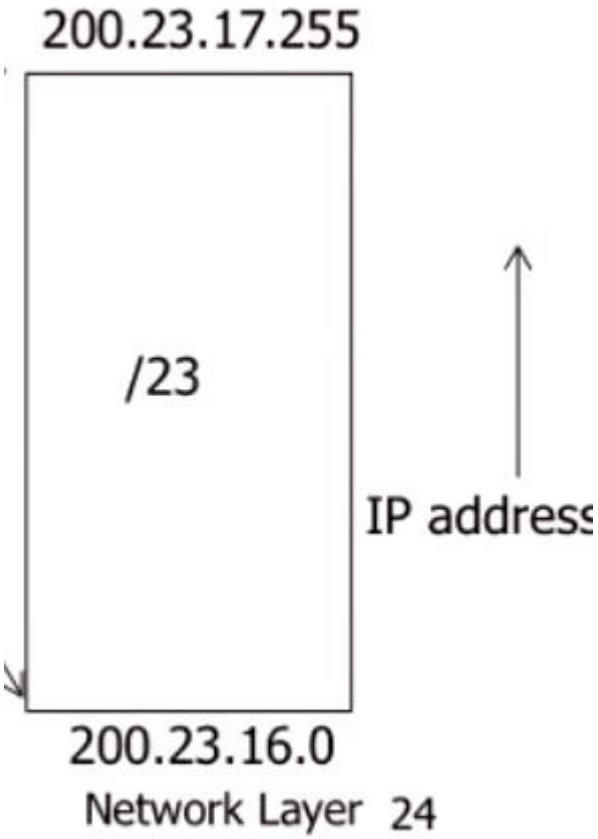
This is the slash notation.

So 200.23.16.0/23 implies that the first 23 bits are the mask and subnet, and remaining 9 bits are for host



In this example first 23 bits network part, remaining 9 bits are host part.

Max address is when all host bits turned on (also the broadcast address)



2^9 available addresses

2^9 – 2 usable addresses for hosts

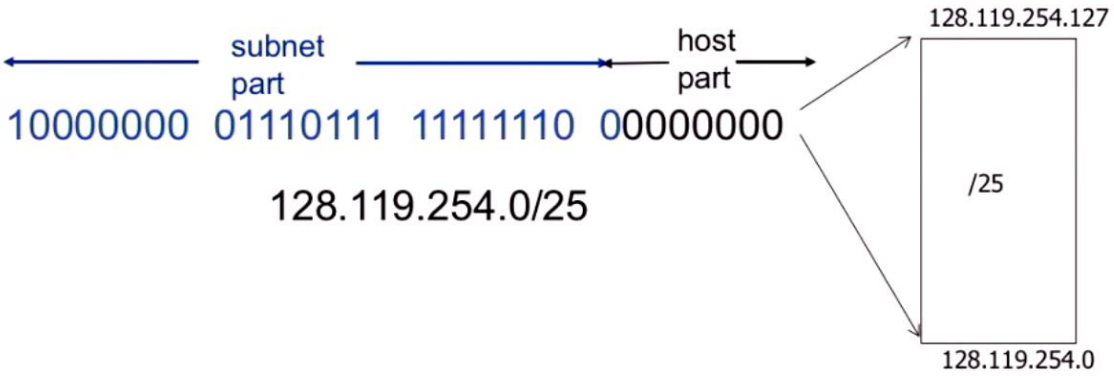
- How many IP addresses belong to the subnet 128.119.254.0/25 ? What are the IP addresses at the two end-points of this range ?

2^(32-25) = 2^7 = 128 addresses

Start address 128.119.254.0

End address 128.119.254.127

Answer: 2^7 = 128 addresses (126 are usable)



Quiz: IP Addressing

- A small organization is given a block with the beginning address and the prefix length 205.16.37.24/29 (in slash notation). What is the range of the block?

3 bits for hosts range is

205.16.37.24 → 205.16.37.31

Ez wtf

Quiz: IP Addressing

- An ISP is granted a block of addresses starting with 190.100.0.0/16. The ISP needs to distribute these addresses to three groups of customers as follows:

1. The first group has 64 customers; each needs 256 addresses.
2. The second group has 128 customers; each needs 128 addresses.
3. The third group has 128 customers; each needs 64 addresses.

Design the subblocks and give the slash notation for each subblock. Find out how many addresses are still available after these allocations.

16 bits for hosts 65535 available jesus fuck

Xxxx xxxx. Xxxx xxxx)_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Start address = 190.100.0.0

First group needs 64 subnets each need 1 byte

First group 190.100.0.0/24 → 190.100.63.255/24

Second group 190.100.64.0/25 → 190.100.127.255/25

Third group 190.100.128.0/26 → 190.100.159.255/26

Group 1 total = 64 * 256

Group 2 total = 128*128

Group 3 total = 128*64

Remaining addresses = 2^16-1 – group1 total – group2 total – group 3 total

THIS IS WHAT ISPS ACTUALLY DO

Q. how does a host get IP address?

- hard coded by system admin in a file
- DHCP: Dynamic Host Configuration Protocol:

This dynamically gets address from server “plug and play”

DHCP:

Goal is to allow host to dynamically obtain its ip address from network server when it joins network

- Can renew its lease on address in use
- Allow re-use of addresses
- Support for mobile users who want to join the network

Overview

- Host broadcasts “DHCP discover” msg
- DHCP server responds with “DHCP Offer” msg
- Host requests IP address: “DHCP request” msg
- DHCP server sends address: “DHCP ack” msg

DHCP will dynamically allocate address to devices based on number of available addresses

DHCP port client = 68

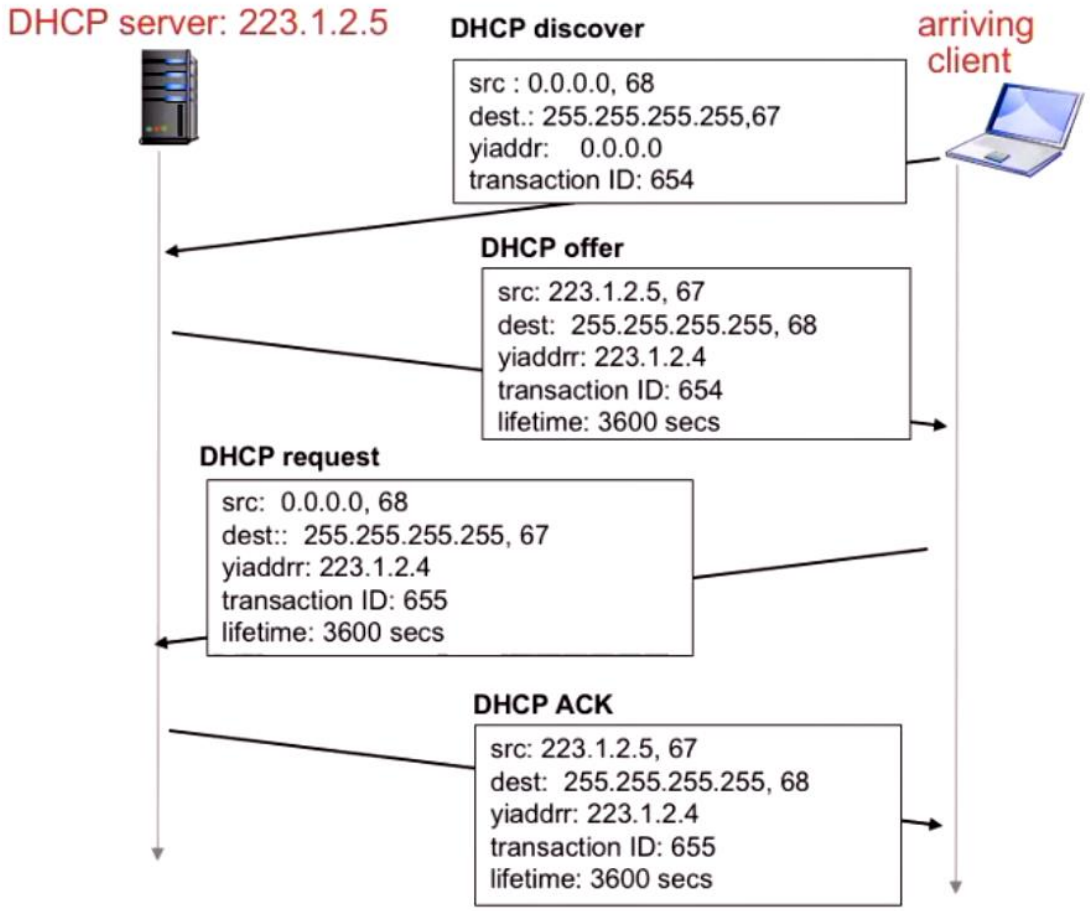
DHCP port destination = 67

New client will broadcast to 255.255.255.255

DHCP discover packet taken by DHCP server

Server will then broadcast an offer (an ip address for client)

DHCP client-server scenario



DHCP can return more than just allocated IP on a subnet

It can also return

- Address of first hop router for client
- Name and ip address of DNS server
- Network mask indicating the network vs host proportion of IP address

DHCP is an application layer protocol working ontop of UDP, providing services to IP layer.

DHCP uses UDP and port number 67 (server side) and 68 (client side)

Usually the mac address is used to identify clients

- DHCP can be configured with a registered list of acceptable MAC addresses

DHCP offer message includes ip address, length of lease, subnet mask, DNS servers, default gateway

DHCP security holes

- DoS attack by exhausting the pool of IP addresses
- Someone can mask as a DHCP server to give out fake ip addresses

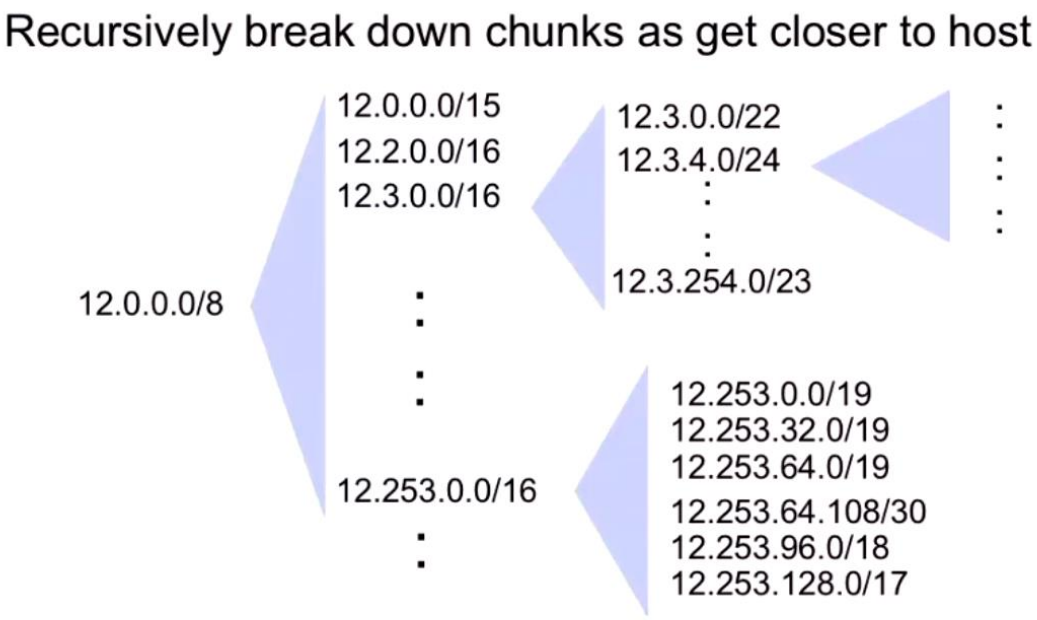
- Authentication for DHCP – RFC 3118

How does network get subnet part of IP address?

- It gets allocated portion of its providers ISP address space.

Can recursively break down chunks as we get closer to host

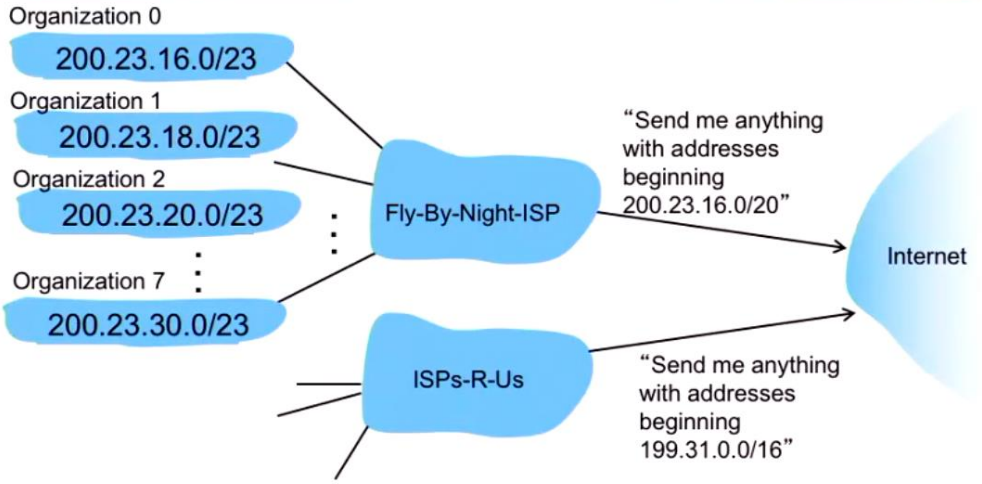
We can subdivide blocks till we reach a singular address



Hierarchical addressing

- Allows efficient advertisement of routing information, similar to DNS!!!

Quiz: What should we do if organization 1 decides to switch to ISPs-R-Us



A: Move 200.23.18.0/23 to ISPs-R-Us (and break up Fly-By-Night's/20 block).
B: Give new addresses to Organization 1 (and force them to change all their addresses)
C: Some other solution

Answer A

We use longest prefix matching since it moves over to ISPS-R-US we can say send us anything with 193.31.0.0/16 OR 200.23.18.0/23 (exact match) and since we use longest prefix matching it will go to the more specific ISPs-R-Us, in fly-by-night since it covers a range of those ISPS it takes more broad range of addresses

Longest prefix matching

- When looking for forwading table entry for given destination address we use the longest address prefix that matches the destination address

Ip addresses are allocated as blocks and have geographical significance. It is possible to map an ip address to a geographical location

how does isp get its block of addresses?

- The ICANN: Internet Corporation for Assigned Names and Numbers

ICANN gives APNIC some /8 IPv4 addresses

APNIC gives Telstra one /8, 129.0/8, so the network prefix is 1000 0001 0

Telstra gives UNSW a/16, 129.94/16, so network prefix is 1000 0001 0101 1110

UNSW gives CSE a /24, 129.197.242/24, network prefix is 1000 0001 0101 1110 1111 0010

CSE gives me a specific address 129.94.242.51

Now address is 1000 0001 0101 1110 1111 0010 0011 0011

IP address trickles down JUST LIKE DOMAIN NAMES IN DNS

IPv4 exhaustion, all registries except AFRNIC has exhausted all ipv4 addresses, maximum is 2^31 ip addresses. By the middle of 2019 AFRNIC should be exhausted

What do we do oh no

Ipv6 addresses!!

ICMP: Internet Control Message Protocol

- Protocol used by hosts and routers to communicate network level information
  - Error reporting: unreachable host/network/port
  - Echo request/reply used by ping
- Works above the IP layer
  - ICMP messages carried in IP datagrams
- ICMP message: type, code plus IP header and first 8 bytes of IP datagram payload causing error.
- Works in layer 3.5 xD

ICMP message protocols

ICMP: Internet Control Message Protocol

Type	Code	Description
0	0	echo reply(ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	3	dest port unreachable
3	4	frag needed; DF set
8	0	echo request(ping)
11	0	TTL expired
11	1	frag reassembly time exceeded
12	0	bad IP header

Traceroute and ICMP

- Source sends a series of UDP segments to destination
    - Set 1 has TTL = 1
    - Set 2 has TTL 2 and so on
    - Uses unlikely port number
- When nth set of datagram arrives to nth router
    - Router discards datagrams
    - Sends source ICMP messages type 11, code 0
    - ICMP messages includes IP address of router
    - When ICMP messages arrives, source records RTT
- Stopping criteria:
    - UDP segment eventually arrives at destination host
    - Destination returns ICMP “port unreachable” message (type 3 code 3)
    - Source stops

NAT (Netword Address Translation)

Private addresses defined in RFC 1918:

- 10.0.0.0/8 (16,777,216 hosts)
- 172.16.0.0/12(1,048,576 hosts)
- 192.168.0.0/16(65536 hosts)

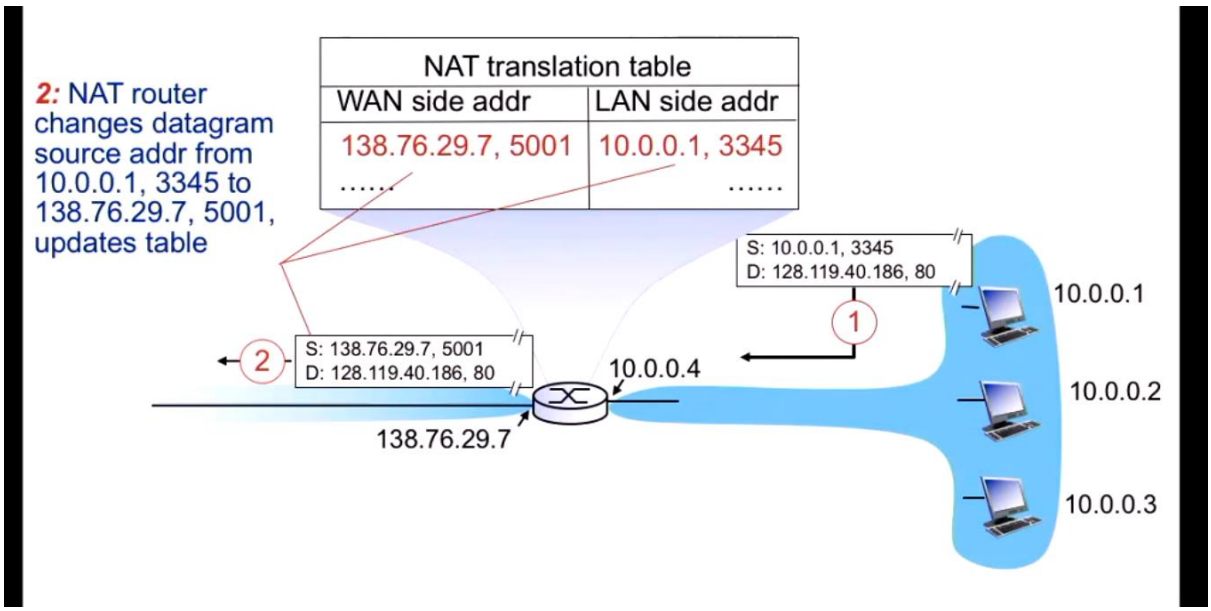
These addresses cannot be routed and anyone can use them (most typically used for NAT)

Used as a source rather than destination, do not forward to these subnets

Within a LAN we can use as destination, but router cannot forward these addresses!

We configure our router to work as a NAT box

Implementation of NAT Router



NAT Router makes these private addresses, gives them an alias so they can be transferred across.

The reason we change the port number is to prevent overlap.

The NAT router creates entries as a look up table, it acts as a middle man for the private addresses.

NAT box allows routers to reach private address if it the private address has initiated the request

Advantages of NAT

Local network uses just one IP address for outside world communication

- Range of addresses not needed from ISP: just one IP address for all devices
- Can change addresses of devices in local network without notifying outside world
- Can change ISP without changing addresses of devices in local network

NAT disadvantages

- NAT violates the architectural model of IP
    - Every IP address uniquely identifies a single node on the internet
    - Routers should only process up to network layer!!! Layer violation
- NAT changes the internet from connection less to a kind of connection-oriented network

Devices inside the local network are not explicity addressable or visible by outside world

For security – advantage (obscurity)

For communication – disadvantage, requires extra processing so more delays in communication

16 bit port-number field:

- ~65,000 simultaneous connections with a single WAN-side address!

NAT is controversial

- Routers should only process to layer 3
- Violates end-to-end argument
- NAT possibility must be taken into account by app designers eg. P2p apps
- Address shortages should have been handles by IPv6

NAT practical issues

- Nat modifies port number and IP addresses
    - Requires recalculation of TCP and IP checksum
- Some applications may embed ip address of port numbers in their payloads (FTP)
- For legacy protocols NAT must look into these packets and translate the embedded IP/port numbers
- In some cases NAT would have to change TCP sequence number
- If applications change port numbers periodically the NAT must be aware of this
- NAT traversal problems
    - How to setup a server behind a NAT router??
    - How to talk to another user behind a NAT router??

NAT traversal problem

Client wants to connect to server with address 10.0.0.1

Solution

- Inbound NAT statically configure NAT to forward incoming connection requests at a given port to server, eg. A white list!!, you advertise to the person wanting to communicate the NAT Box ip + port, and still act as a middle man

Solution

- Universal plug and play UpnP internet gateway device protocol, to allow NATed hosts to
    - Learn public ip address
    - Add/remove port mappings with lease time

Solution:

- NATed client establishes client to relay (a man in the middle)
- External client connects to relay
- And the relay acts as a bridge between the two connections! Its brilliant
- Man in the middle sending the messages
- This brought privacy issues so skype brought UDP hole punching
- Skype no longer sends but just informs both sides the information of connection so it allows communication router -> router rather than from client -> skype

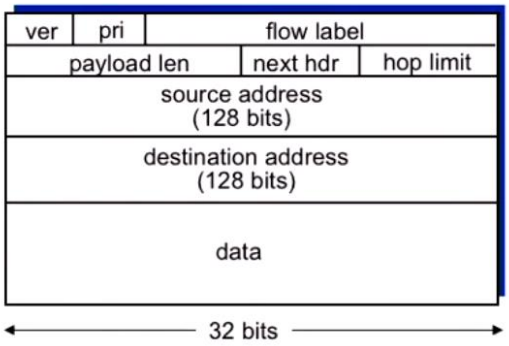
IPv6

- Initial motivation: 32-bit address space soon to be completely allocated
- Ipv6 offer solution of 128 bit addresses instead of 32 bits
- Over 10^23 addresses for every square meter on planet earth
- Additional motivation
    - Header format helps speed processing/forwarding
    - Header changes to facilitate QoS
- Ipv6 datagram format:
    - Fixed-length 40-byte header
    - No fragmentation allowed (PATH MTU DISCOVERY DONE BEFOREHAND)

With a fixed 40-byte header we need to modify our datagram

IPv6 datagram format

priority: identify priority among datagrams in flow (traffic class)
flow Label: identify datagrams in same "flow."
(concept of"flow" not well defined).
next header: identify upper layer protocol for data



Hop limit == TTL

Other changes from IPv4

- Checksum removed entirely to reduce processing time at each hop
- Options allowed but outside of header indicated by next header field
- ICMPv6: new version of ICMP
    - Additional message type eg. Packet too big
    - Multicast group management functions
- Colon hexadecimal notation instead of dotted decimal.

IPv6

Using dotted ecimal since we have 128 bits, we need to compact it into hexadecimal

So eg. 68E6:8C64:FFFF:FFFF:0:1180:96A:FFFF

Also zero compression implies zeroes inbetween

Eg. 68E6::

Transition from IPv4 to IPv6

Not all routers can be upgraded simultaneously

- No flag days
- How will network operate with both a mix of IPv4 and IPv6 routers?

Solution: tunnelling

Ipv6 datagram carried as payload in IPv4 datagram as payloads (sort of like a wrapper)

Control Plane (routing and routing algorithms) in order to build forwarding tables

Two approaches to structuring network control plane:

1. Per-router control (traditional)
2. Logically centralized control (software defined networking)

In per router control plane

- Individual routing algorithm components in each and every router interact with each other in control plane to compute routing tables

Logically centralized control plane

- A distinct typically remote controller interacts with local control agents in routers to compute forwarding tables (taking load off router) calculated in the remote controller and passed back routing table to control agent

Routing algorithm → local forwarding table

Autonomous System (AS) or domain is a region of network under a single administrative entity

Border routers which bridge between autonomous system. It has atleast 1 or 2 interfaces in different autonomous system.

Interior routers are within the current autonomous system and only communicate with routers within its own autonomous system including its border router.

Intra autonomous system routing (can only apply within its own autonomous system) solely dependant on network system administrative protocol

Inter autonomous system routing, that exchanges information between different routing systems.

Internet Routing

- Works at two levels
- Each AS runs an intra-domain routing protocol that establishes routes within its domain
    - AS – region of network under a single administrative entity
    - Link state eg. Open shortest part first (OSPF)
    - Distance vectory eg. Routing information Protocol (RIP)
    - ASes participate in an inter-domain routing protocol that establishes routes between domains

- Path vectory eg. Border gateway protocol (BGP)

(RIP) uses all costs as 1 and still works some how

Typically simple : all links are equal

Least cost paths => shortest paths (hop counts)

Network operators add policy exceptions

- Lower operational costs
- Peering agreements
- Security concerns

Routing algorithm classes

All router is based on link cost

Neighbour discovery phase to discover cost of communicating with neighbours.

Neighbour discovery is common between both the following algorithms

First we broadcast our neighbour information to all other routers in the autonomous system (diffusion phase)

Using this information we can form a topology (Graph) each router will get a view of the topology and apply their algorithm (search algorithm) to the topology (think graphs and search algorithm)

Link State (Global)

- Routers maintain cost of each link in the network
- Connectivity/cost changes flooded to all routers
- Converges quickly (less inconsistency, looping, etc).
- Limited network sizes

Distance Vector (Decentralised)

- Routers maintain next hop and cost of each destination
- Connectivity/cost changes iteratively propagate from neighbour to neighbour
- Requires multiple rounds to converge
- Scales to larger networks.

Link state routing we hear directly from the router allowing us to build a topological view of the network

In the case of distance vector we get the processed data where we utilised the information from the neighbour router to create its own forwarding table. This in turn also means any mistakes trickle down. (this doesn't occur in the first case as in link state we receive information and perform our own routing algorithm, whereas in this we only receive the outcome of the routing algorithm).

Link state routing

Each node maintains its local link state (it's connections)

Each node floods its local link state

- On receiving a new LS message a router forwards the message to all its neighbours other than the one it received the message from.

Flooding Link State Advertisements (LSA)

- Routers transmit LSA on links
    - A neighbouring router forwards out on all links except incoming links
    - Keep a copy locally so we don't forward previously seen LSA's (similar to caching
- Challenges for this
    - Packet loss
    - Out of order arrival
- Solutions for the challenges
    - ACKS and retransmits
    - Sequence numbers
    - TTL
- Uses open shortest path first! (OSPF) not RIP

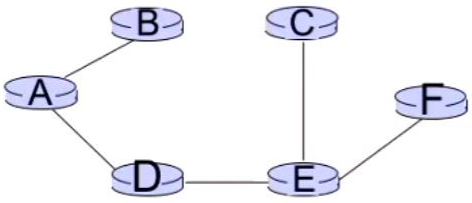
Eventually each node learns the entire network topology

- Can use djikstra's to compute shortest paths between nodes, idk why not A*

We now use Djikstra's algorithm to compute shortest part for each node.

Using our result from djikstra's we construct a shortest path tree for our forwarding table

resulting shortest-path tree from A:



Destination	Link
B	(A,B)
C	(A,D)
D	(A,D)
E	(A,D)
F	(A,D)

Issue #1: scalability

Since we broadcast to flood link state messages

We are sendings O(N*E(where N is # nodes and E is # edges

Processing complexity for djikstra is also going to be O(n^2) because we need to check all nodes w note in N' at each iteration we have O(N) iterations

We have O€ entries in LS topology database

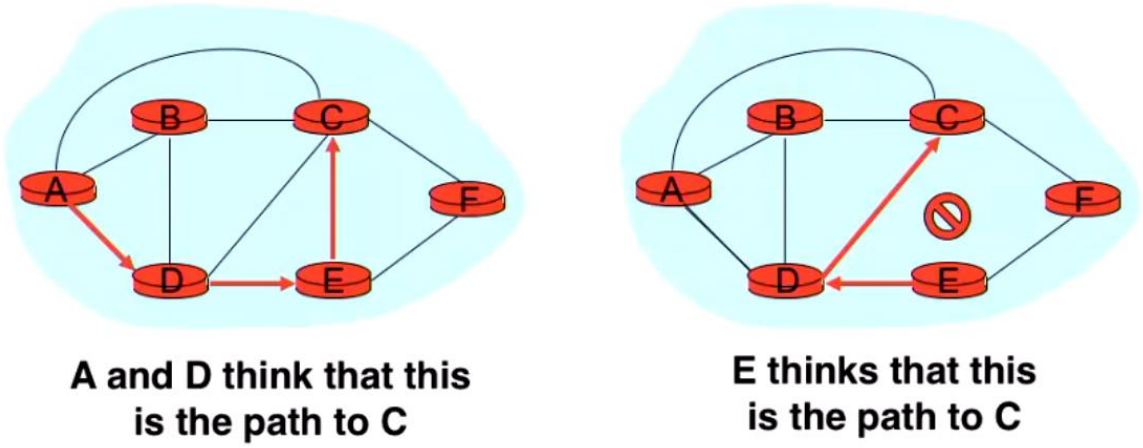
We have O(N-1) entries in the forwarding table

Issue #2: transient Disruptions

Inconsistent link-state database

Some routers know about failure before others

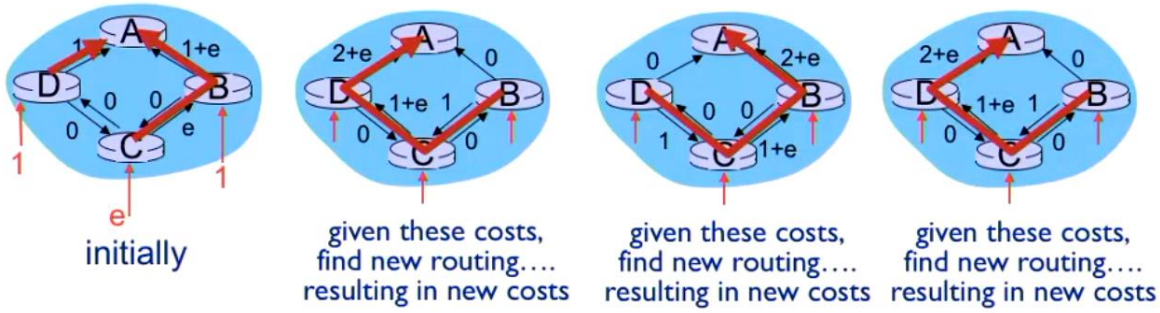
Shortest paths are no longer consistent which will cause transient forwarding loops



The loop disappears when LSA disappears

Issue #3: Oscillations

In the discovery phase we will get new costs since the change in network conditions. Once we compute it once, the network conditions change which change traffic conditions so this iteratively keeps giving inconsistent paths

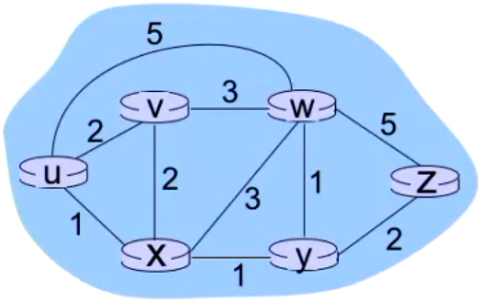


We need a stable route for routing algorithms, above is dynamic

DISTANCE VECTOR ROUTING

Uses bellman-ford equation

Bellman-Ford example



clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

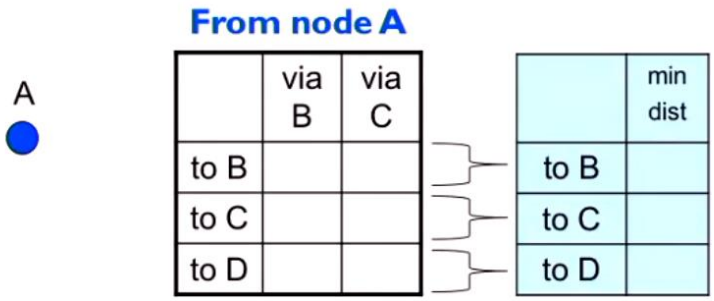
$$d_u(z) = \min \{ c(u,v) + d_v(z), \; c(u,x) + d_x(z), \; c(u,w) + d_w(z) \}$$
$$= \min \{2 + 5, \; 1 + 3, \; 5 + 3\} = 4$$

node achieving minimum is next hop in shortest path, used in forwarding table

Each router maintains its shortest distance to every destination via each of its neighbours.

We maintain two tables, one is the minimum table, other is the minimum table through the different nodes

How Distance-Vector (DV) works



Each router computes its shortest distance to every destination via any of its neighbors

40

Routers then share their distance vector table to its immediate neighbours NOT THE FIRST TABLE WITH ALL THE DISTANCES

Distance table can change without effecting our distance vector table as we take the minimum from all columns, however when our distance vector table change we must broadcast this to all our immediate neighbours, whereas if distance table changes, we don't need to broadcast unless it effects our distance vector table.

LEARN THE CALCULATIONS FOR FINAL EXAMS IT IS EXAMINABLE!!!

Each router knowns the links to its neighbours

Each router has provisional "shortest path" to every other oruter – its distance vector (DV)

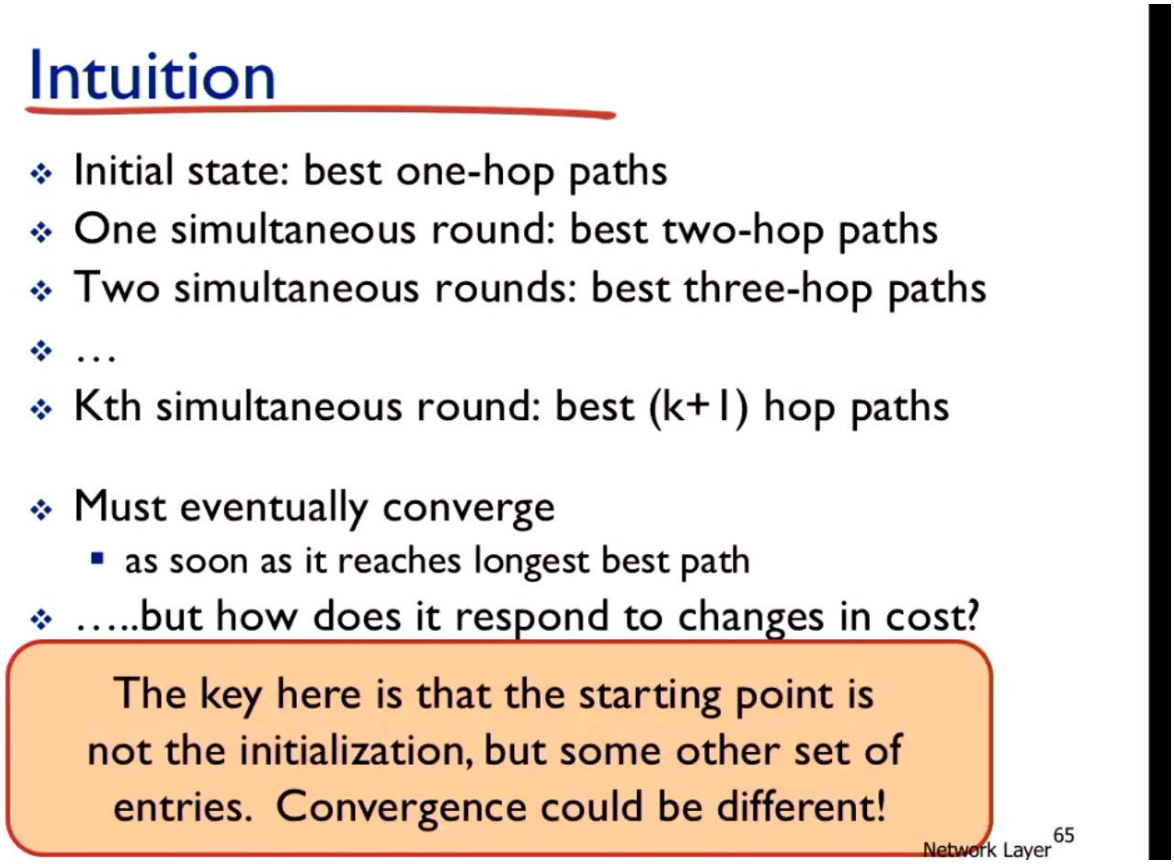
Routers exchange this DV with their neighbours

Routers look over the set of options offered by their neighbours and select the best one

Iterative process converges to set of shortest paths

Every iteration of iteration n gives optimal n-hop path

Self-terminating algorithm (as soon as no visible changes to distance vector table it terminates)



DV LINK COST CHANGES

We only tell neighbours the destination and cost involved NOT THE ROUTING PATH!!! It doesn't know there is bounce backs

DV protocol is prone to creating loops as changes in a DV table requires broadcast tto all neighbours and so on.

Good news (cost decrease) propogates quickly

Bad news (cost increase) propogates very slowly, we increment O(N) where N is the increase (counting to infinity problem)

The poisoned reverse rule

- Heuristic to avoid count to infinity
- If B routes via C to get to A
    - B tells C its B's distance to A is infinite so C wont route to A via B, just takes direct link
- Poisoned reverse rule will make sure it only tells of its neighbours not next hop router, so if A-> C = A->B->C poison reverse rule will make A-> B infinite through C in that to make sure we use our neighbours. If something changes want to look at all keep looking at past can cause problems. Depending on the neighbour we hide certain links. It removes the old "poisoned" values and turns it same cost as good news travel

There are edge cases where poisoned reverse still counts to infinity

Comparison of LS and DV algorithms

Comparison of LS and DV algorithms

message complexity

- ***LS:*** with n nodes, E links, O(nE) msgs sent
- ***DV:*** exchange between neighbors only
  - convergence time varies

speed of convergence

- ***LS:*** $O(n^2)$ algorithm requires O(nE) msgs
  - may have oscillations
- ***DV:*** convergence time varies
  - may be routing loops
  - count-to-infinity problem

robustness: what happens if router malfunctions?

LS:

- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagate thru network

For link state

- we use open shortest path OSPF
- intermediate system to intermediate system (IS-IS)

For distance vector

- we use routing information protocol RIP
- interior gateway routing protocol (IGRP-Cisco)
- border gateway protocol BGP

Quiz: Impact of link cost

A problem that can arise when a link state algorithm is used in a network where link costs equal to load is:

a) Count to infinity

b) Poisoned reverse

c) The infinite reverse

d) oscillation

D (count to infinity + poisoned + infinite reverse → distance vector) EZ

Quiz: Impact of link cost

When compensating for link cost changes in the distance vector algorithm, it can be generally said that:

a) Increased costs are propagate quickly, i.e., "bad news" travels fast

b) Decreased costs are propagated rapidly, i.e. "good news" travels fast

c) Increased costs do not converge

d) Decreased costs propagate slowly, i.e., "good news" travels slowly.

B EZ COMON

Hierarchical routing

So far we assumed all routers are identical, network is flat which is untrue in practice

We want to aggregate routers into regions known as autonomous system (AS) or simply domains

Intra-AS routing

- routing among hosts, routers in same domain
- all routers in AS must run the same protocol (Link state/DV)
- routers in different AS can run different intra-domain routing protocols
- gateway router at edge of its own AS has links to routers in other AS'es

inter-AS routing

- routing among AS'es
- gateways perform inter-domain routing as well as intra domain routing

Autonomous Systems (AS)

- as is a network under a single administrative control
- currently over 30,000 AS's
- think AT&T, France Telecom, UNSW, IBM etc.
- AS's are sometimes called domain hence the term interdomain routing.
- Each AS is assigned a unique identifier (16-bit AS number)

Internet inter-AS routing BGP

- BGP (Border Gateway Protocol): the defacto inter-domain routing protocol, the glue which holds the internet together
- BGP provides each AS a means to
    - eBGP (external BGP): obtain subnet reachability information from neighboring AS's
    - iBGP (internal BGP): propogate reachability information to all AS-internal routers
    - determine "good" routes to other networks based on reachability information and policy
- allows subnet to advertise its existence to rest of internet (broadcast "I am here")

BGP a reachability protocol

- path vector protocol, tells path but not cost
- reaches a subnet (domain) based on data received
- standard exterior routing protocol in the internet
- neither a pure distance vector protocol nor a pure link state protocol (more lenient to distance vector however)
- when a pair of AS's agree to exchange routing information, each must designate a router that will speak BGP on it's behalf. These two routers are said to become BGP peers of one another
- they are normally near the edge of the AS (hence called Border Rotuer)
- each AS can have multiple BGP routers

BGP basics

- BGP session
    - Two BGP routers exchange BGP messages over TCP connection
    - Advertising paths to different destination network prefixes (BGP is a path vector protocol)

Path attributes and BGP routes

- Advertised prefix includes BGP attributes
    - Prefix and attributes = route
- Two important attributes
    - AS-PATH: list of AS's through which prefix advertisement has passed
    - NEXT-HOP: indicates specific domain router to next hop
- Policy based routing
    - Gateway receiving route advertisement uses import policy to accept/decline path (eg. Never route through certain AS)
    - AS policy also determines whether to advertise path to other neighbouring AS's
- Policy file, to blacklist/whitelist certain AS's, we simply send packet to AS and the AS uses its own protocol (uses its own internal protocol)

BGP Dissection:

- BGP does not communicate or interpret distance metrics, even if those metrics are available
    - BGP speaker can declare that a destination is unreachable or give a list of AS's on the path to the destination
    - It cannot transmit or compare the cost of two routes unless routes come from the same AS
- If a router learns about two paths to the same network, it cannot know which path is shorter because it cannot know the cost of routes across intermediate AS
- BGP is thus a reachability protocol rather than a routing protocol
- BGP builds path based on lowest number of AS's

Link Layer and LANS

(last layer for this course physical layer is more electrical)

From macro to micro

- We are going from routing of autonomous systems, to a subnet

Hosts and routers are nodes

Communication channels that connect adjacent nodes along path of communication – link

- Wired/wireless/LAN

Layer-2 packet: frame encapsulating the datagram

Data-link layer has the responsibility of transferring datagram from one node to physically adjacent node over a link!

Datalink layer doesn't just add a header, but also a tailer!

For each link, there could be a different link layer operation protocol

Link layer context

- Datagram transferred by different link protocols over different linsk
    - Ethernet on first link, frame relay on intermediate links, 802.11 on last link
- Each link protocol provides different services
    - May or may not provide RDT over links

transportation analogy:

- trip from Princeton to Lausanne
    - limo: Princeton to JFK
    - plane: JFK to Geneva
    - train: Geneva to Lausanne
- tourist = datagram
- transport segment = communication link
- transportation mode = link layer protocol
- travel agent = routing algorithm

Link layer services

- Framing and link access
    - Encapsulates datagram into frame adding a header and tailer
    - Channel access if shared medium (eg. Wireless)
    - MAC addresses used in frame headers to identify source and destination
        - Different from IP addresses
- Reliable delivery between adjacent nodes
    - We learned how to do this already TCP BRUH OR STP XD
    - Only used on low bit error link so (fiber, or twisted pair link)
    - Wireless links: high error rates
        - Why both link-level and end-end reliability???
            - Making sure transmission is done more efficiently, if at any hop corruption, can do re-transmission there rather than waiting till end destination to send NAK
    - Flow control
        - Pacing between adjacent send and receives

    - Error detection
        - Errors caused by signal attenuation or noise
        - Receiver detects presence of errors, signals sender for re-transmission or drops the frame
    - Error correction
        - Receiver identifies and corrects bit errors without needing re-transmission (detect and fix error without re-transmit)
    - Half-duplex and full duplex
        - With half duplex, nodes at both ends of link can transmit but not at same time
        - Wireless tends to be half-duplex

Link-layer implementation location

- In each and every host
- Link layer implemented in an adaptor (network interface card on a chip)
    - Ethernet card, 802.11 card, ethernet chipset
    - Implements link and physical layer
- Attatches into host's system buses
- Combination of hardware, software and firmware

Adapators communicating

Sender side multi-plexing

- Encapsulates datagram into frame
- Adds error checking bits, rdt, flow control etc

Receiving side de-multiplexing

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

What is framing?

- Physical layer talks in terms of bits
- How do we identify frames within the sequence of bits?
- Need to frame, include a range from start → end of frame
- Framing methods
    - Byte counting
    - Byte stuffing
    - Bit stuffing
    - Timing/physical layer
    - (CHECK MY STP)

We can encapsulate the length of the frame as a byte so that we have length of frame inside the frame lol, but would it work??

If the length of the frame is corrupted, the system will collapse. Difficult to fix after errors

Byte stuffing

- Flag byte to indicate start and end of frame
- What if special byte occurs in the data within the frame
    - Replace the byte flag inside frame with an escape code
    - Have to escape the escape code too
    - Eg, if we have print “”, when we want “ inside our print we need to escape so \”
- It works but its annoying (looks for unescaped flag byte)

Byte Stuffing in PPP

- Flag Byte: 0x7E = 01111110
- ESC Byte: 0x7D = 01111101

- To stuff (unstuff) a byte, add (remove) 0x7D and XOR the byte that follows with 0x20
- Removes the Flag from the content of the frame.

- 0x7E → 0x7D {0x7E XOR 0x20}
  → 0x7D 0x5E
- 0x7D → 0x7D {0x7D XOR 0x20}
  → 0x7D 0x5D

Bit stuffing

- Can stuff at the bit level too
- Assume that the flag is 6 consecutive 1’s
- On transmit if five 1’s appear in data insert a 0
- On receive, remove the 0 that appears after 5 1’s

Framing in ethernet

- Start of frame is recognised by
    - Preamble: seven bytes with pattern 10101010
    - Start of frame delimitier (SFD): 10101011
- End of frame: absence of transition in Manchester encoded signal
- Inter-frame gap is 12 bytes (96 bits) of idle state
    - 0.96 microsec for 100mbits/ethernet
    - 0.096 microsec for 1gb/s ethernet

Preamble 7 Bytes	SFD 1 Byte	Dest MAC 6 Bytes	Source MAC 6 Bytes	Type/Length 2 Bytes	Payload 46-1500 Bytes	FCS/CRC 4 Bytes	Inter Frame Gap

LEARN FOR EXAM HOW MUCH HEADER FOR OVERALL MESSAGE -> FRAME

40+7+1+6+6+2+4 + 12 insane overhead

78 bytes of header total for 1 byte payload

Error detection

EDC = error detection and correction bits (redundancy)

D = data protected by error checking, may include header fields

- Error detection is not 100% reliable
    - Protocol may miss some errors (rarely)
    - Larger EDC fields yields better detection and correction

Error coding

Add check bits to the message bits to let some error be detected and some be corrected

Now how do we structure the code to detect many errors with few check bits and modest computation?

A simple example

- Send two copies of the same message so (101) 101
- Error if the two copies are different eg (101) 100
- How many errors cant it correct? 0
- How many errors can it detect? At most 3
- How many errors will make it fail? Specific 2 bits errors
- What is overhead? 50%

Parity bits

Suppose we want to send

- 0010110 1101100 0110010
- For every d bits (eg. D= 7) add a parity bit
- 1 if the number of ones is odd
- 0 if number of ones is even
- So parity bits would be , 1,0,1 for respectiver segments

For each block of size d bits

- Count number of 1's and compare with following parity bit

If an odd number of bits gets flipped we'll detect but cant really correct

Cost is original message/D

We can get a error pass through if we invert even number of bits, as that would pass through as still odd

Two dimensional parity

- For the same message
- Add an extra parity byte and compute parity on the columns too
- Can detect 1,2,3 and some 4-bit errors
- Can correct single bit errors

In reality

- Bit errors occur in bursts
- We're willing to trade computational complexity for space efficiency
    - Make the detection routine more complex to detect error bursts, without tons of extra data
- We need hardware to interface with network so we do computation there (makes more use)

Checksum

- Sum up data in n-bit words
- Internet checksum uses 16-bit words
- How well check sum works
    - How many errors can it detect/correct? 1 detect, 0 correct
- What have we gained compared to parity
    - Can detect all burst errors up to 16

CRC (Cyclic Redundancy Check)

- Powerful error-detection coding
- View data bits, D, as a binary number
- Choose r+1 bit pattern (generator), G
- Goal is to choose r CRC bits, R, such that
- <D,R> exactly divisible by G (modulo 2)
- Receiver knows G divides <D,R> by G. if non-zero remainder: error detected
- Can detect all burst errors less than r+1 bits
- Widely used in pratice (Ethernet, 802.11, WiFi, ATM)

CRC-32 is used in ethernet frames

0x04C11DB7 as our generator function

Do long division with our data with this generator, append remainder to frame

Multiple ACCESS protocols

- Two types of links
    - Point to point
        - PPP for dial up access
        - Point to point link between ethernet switch, host
    - Broadcast (shared wire or medium)
        - Old fashioned ethernet
        - Upstream HFC
        - 802.11 wireless LAN

Single shared broadcast channel

Two or more simultaneous transmissions by nodes:

Interference

Collision if node receives two or more signals at the same time

Multiple access protocol

- Distributed algorithm that determines how nodes share channel, i.e determined when nodes can transmit
- Communication about channel sharing must use channel itself, no out of band channel for coordination

Given: broadcast channel of rate R bps

Desiderata:

1. When one nodes wants to transmit it can send at rate R
2. When M modes want to transmit each can send at average rate R/M
3. Fully decentralized
    a. No special nodes to coordinate transmissions
    b. No synchronisation of clocks, slots
4. Simple

MAC protocols: taxonomy

- Three broad classes
    - Channel portioning
        - Divide channel into smaller pieces
        - Allocate piece to node for exclusive use
    - Random access
        - Channel not divided, allow collisions
        - Recover from collisions
    - Taking turns
        - Nodes take turns but nodes with more to send can take longer turns

Channel partitioning MAC protocols TDMA (time division multiple access, time de-multiplexing)

We can also use frequency de-multiplexing

Quiz: Does channel partitioning satisfy ideal properties ?

1. if only one node wants to transmit, it can send at rate R.
2. when M nodes want to transmit, each can send at average rate R/M (fairness)
3. fully decentralized:
   - no synchronization of clocks, slots
   - no special node to coordinate transmissions
4. simple

A. 0
B. 1
C. 2
D. 3
E. 4

(Which ones?)

With time de-multiplexing – sometimes 2,4

With frequency de-multiplexing – sometimes 2,4

Random access protocols

- When node has packet to send
  - Transmit at full channel data rate R
  - No a priori coordination among nodes
- Two or more transmitting nodes → collision
- Random acces MAC protocol specifies
  - How to detect and recover from collisions
- Examples of random access MAC protocols
  - Slotted ALOHA
  - ALOHA
  - CSMA, CSMA/CD, CSMA/CA

ALOHANET

- Norm abrahmson left Stanford in 1970 so he could surf
- Set up first data communication system for Hawaiian Islands
- Central hub at U. Hawaii, Oahu

Slotted ALOHA

assumptions

- All frames same size
- Time divided into equal size slots (time to transmit 1 frame)
- Nodes start to transmit only slot beginning

- Nodes are synchronized
- If 2 or more nodes transmit in slot, all nodes detect collision

Operation

- When node obtains fresh frame, transmits in next slot
    - If no collision: node can send new frame in next slot
    - If collision: node retransmits frame in subsequent slot with probability p until success

Pros and cons of Slotted ALOHA

Pros

- Single active node can continuously transmit at full rate of channel
- Highly decentralised, only sluots in nodes need to be in sync
- Simple

Cons

- Collisions, wasting slots
- Idle slots
- Nodes may be able to detect collisions in less time than it takes to transmit packet
- Clock syncrhonization

Slotted Aloha efficiency

Efficiency is long run fraction of successful slots, many nodes all with many frames to send

- Suppose N nodes with many frames to send each transmits in slot with probability p
- Probability that given node has success in a slot = p(1-p)^n-1
- Probability that any node has a success = Np(1-p)^n-1
- Max efficiency: find p* that maximises Np(1-p)^n-1
- For many nodes take limit of Np*(1-p*)^n-1 as N goes to infinity
- Max efficiency = 1/e = 0.37
- At best: channel used for useful transmissions 37% of the time

Pure unslotted ALOHA

- Unslotted ALOHA: simpler, no synchronization needed
- When frame first arrives
    - Transmit immediately
- Collision probability increases
    - Frame sent at t0 collides with other frames sent in [t0-1, t0+1]

Pure ALOHA efficiency

P(success by given node) = P(node transmits) ·
P(no other node transmits in $[t_0-1,t_0]$ ·
P(no other node transmits in $[t_0,t_0+1]$

$= p \cdot (1-p)^{N-1} \cdot (1-p)^{N-1}$
$= p \cdot (1-p)^{2(N-1)}$

… choosing optimum p and then letting n $\longrightarrow \infty$

$= 1/(2e) = .18$

even *worse* than slotted Aloha!

CSMA (carrier sense multiple access)

CSMA always listen before transmit

- If channel sensed idle – transmit entire frame
- If channel is busy wait
- Human analogy, don't interrupt others
- This doesn't eliminate all collissions as we cannot assume non-zero propagation delay, if 1 discovers channel idle, it transmits, this will take time to travel to all other channels, and within that time another channel senses idle and also transmits bam collision
- Allows for both transmission and listening
- If we detect collision, we wait and re-try
- It reduces but not eliminate collisions
- Biggest remaining problem
    - Collisions still take full slot

CD collision detection CSMA/CD

- Once we detect a collision, abort transmission to avoid channel wastage
- We need restriction on minimum frame size and maximum distance,
    - We need large enough frame to create a collision so it can be detected

ETHERNET CSMA/CD algorithm

- Network interface card receives datagram from network layer, creates frame
- If NIC senses channel is idle, starts frame transmission, if busy it waits until idle before transmitting
- If NIC transmits entire frame without detecting another transmission NIC is done with the frame
- If NIC detects another transmission while transmitting it aborts and sends jam signal
- After aborting, NIC enters binary backoff
    - After mth collision, NIC chooses K at random from o → 2^m-1
    - NIC waits K*512 bit times and returns to step 2
    - More collision = longer backoff as m increases → k increases → longer delay

Minimum packet size

- We enforce a minimum packet size
    - This is to make sure that we can detect collisions, if the packet is too small we transmit before we detect collision, we want our minimum frame size to last to guarantee the channel is really empty, with small frame size, we get ALOOT of collisions since we transmit without knowing if its really empty. This is because of propogation delay
- Give a host enough time to detect collisions
- In ethernet, minimum packet size = 64 bytes (two 7-byte addreses, 2 byte type, 4 byte CRC and 46 bytes of data)
- If host has less than 46 bytes to send then we add padding to make it atleast 46 bytes
- The relationship between minimum packet size and length of LAN
    - LAN length = min_frame_Size * propogation_speed / 2*bandwidth

Performance of CSMA/CD

- Time wasted in collisions
    - Proportional to distance d
- Time spend transmitting a packet
    - Packet length p divided by bandwidth b
- Rough estimate for efficiency (K is constant)
- E ~= (p/b)/((p/b)+Kd)
- For large packets, small distance E ~= 1
- As bandwidth increases, E decreases
- That is why high speed LANS are all switched

Quiz: Does CSMA/CD satisfy ideal properties

1. if only one node wants to transmit, it can send at rate R.
2. when M nodes want to transmit, each can send at average rate R/M (fairness)
3. fully decentralized:
   - no synchronization of clocks, slots
   - no special node to coordinate transmissions
4. simple

A. 0
B. 1
C. 2
D. 3
E. 4

(Which ones?)

Meets criteria, 1, 3

Taking turns MAC protocols

- Channel portioning MAC protocols
  - Share channel efficiently and fairly at high load
  - Inefficient at low load: delay in channel access, 1/N bandwidth allocated even if only 1 active node
- Random access MAC protocols
  - Efficient at low lode, single node can fully utilize channel
  - High load: collision overhead:
- Taking turns protocols
  - Best of both

Taking turns – polling

- Master node invites slave nodes to transmit in turns
- Typically used with slave devices
- Concerns
  - Polling overhead
  - Latency
  - Single point of failure in the master node

Keypad polling, scan through all 1 by 1 check if 1 wants to transfer

Taking turns – token passing

- Control token passed from one node to next sequentially
- Token message
- Concerns
    - Token overhead
    - Latency
    - Single point of failure in token

Quiz: Does taking turns satisfy ideal properti

1. if only one node wants to transmit, it can send at rate R.
2. when M nodes want to transmit, each can send at average rate R/M (fairness)
3. fully decentralized:
    - no synchronization of clocks, slots
    - no special node to coordinate transmissions
4. simple

A. 0
B. 1
C. 2
D. 3
E. 4

(Which ones?)

Satisfies 2,4 not decentralised

Switched LANS

MAC addressed and ARP

- 32-bit IP addresses
    - Network layer address for interface
    - Used in layer 3 network layer forwarding
- MAC (or LAN or physical or Ethernet) address:
    - Function is used locally to get frame from one interface to another physically connected interface (same network, in IP-addressing sense) this allows us to tell which device on the same network we are transferring to
    - 48-bit mac address for most LANs burned in the NIC ROM also sometimes can be set in software
    - Hexadecimal notation so 1A-2B-3C-4D-5E-6F

- Each adapter on LAN has a unique LAN address
- MAC address and LAN address is organised by the OUI (organiser unique identifier) (first 3-bytes for organisiation) (last 3-bytes are available for use within the LAN)

If we want to hide our mac address, we can use some software, kali-linux allows us to hide and set our own mac address.

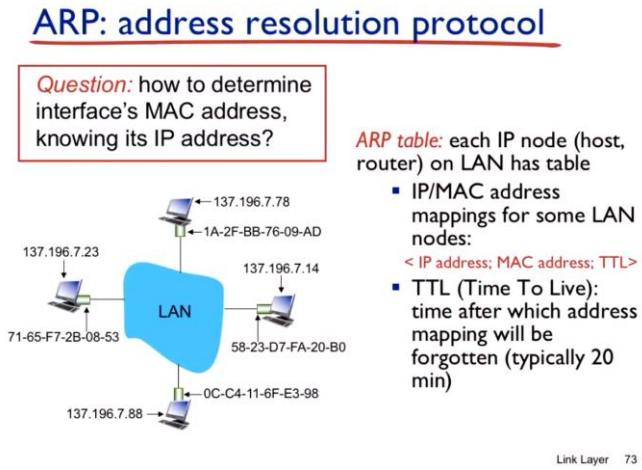
- Mac address allocation administered by IEEE
- Manufacturer buys portion of MAC address space to assure uniqueness
- MAC flat address → portability
    - Can move LAN card from one LAN to another
- IP hierarchical address is not portable
    - Address depends on IP subnet to which node is attached

MAC address vs IP address

- MAC address is used in link layer
    - Hard coded in read only memory when adapter is built
    - Flat name space of 48 bits
    - Portable and can stay the same as hosts switch
    - Used to get packet between interfaces on same network (link layer)
- IP addresses
    - Configured or learned dynamically (DHCP)
    - Hierarchical name space of 32 bits in ipv4 128 bits in ipv6
    - Not portable and depends on where host is attached
    - Used to get packet to destination IP subnet

Taking stock naming

Layer	Examples	Structure	Configuration	Resolution Service
App. Layer	www.cse.unsw.edu.au	organizational hierarchy	~ manual	
Network Layer	129.94.242.51	topological hierarchy	DHCP	DNS
Link layer	45-CC-4E-12-F0-97	vendor (flat)	hard-coded	ARP



ARP protocol for same LAN

- A wants to send datagram to B
  - B's MAC address is not in A's ARP table
- A broadcasts ARP query packet, containing B's IP address
  - Destination MAC address = FF-FF-FF-FF-FF-FF (mask)
  - All nodes on LAN receive the ARP query
- B receives ARP packet, replies to A with it's (B's) MAC address
  - Frame sent to A's MAC address (unicast)
- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (TTL timeout)
  - Soft state: information that times out(goes away) until refreshed
- ARP is plug and play
  - Nodes create their ARP tables without intervention from the net administrator

Addressing: routing to another LAN

- Send datagram from A to B via R router
- Focus on addressing at IP (datagram) and MAC layer (frame)
- Assume A knows B's IP address
  - How does A know B is not local (connected to the same LAN as A)
    - Subnet mask from DHCP
  - Assume A knows IP address of first hop router R
    - Default router discovered from DHCP
  - Assume A knows R's MAC address through ARP
- A creates IP datagram with IP source A destination B
- A creates link layer frame with R's MAC address at destination, frame contains A to B ip datagram
- MAC addresses are only visible within the same subnet, we go through R's mac address since R knows which node is B
- Frame in sent from A to R
- Frame received at R, datagram extracted and passed up to network layer
- R forwards datagram with IP source A, destination B through the forwarding table

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

Security issues: ARP cache poisoning

- Denial of service – Hacker replies back to an ARP query for a router NIC with a fake MAC address
- Man-in-the-middle attack – hacker can insert his/her machine along the path between victim machine and gateway router
- Such attacks are generally hard to launch as hacker needs physical access to the network

Solutions are

- Use switched ethernet with port security enabled (one host MAC address per switch port)
- Adopt static ARP configuration for small-size networks
- Use ARP monitoring tools such as ARPWatch
- MAC-filtering

Link Layer summary

- Principles behind datalink layer services
    - Error detection and correction
    - Sharing a broadcast channel : multiple access
    - Link layer addressing, ARP

Ethernet

- Bob Metcalfe, Xerox PARC, visits Hawaii and gets the idea for ethernet
    - First widely used LAN technology
    - Simpler and cheap than token LANS and ATMS
    - Kept up with the speed race of 10mbs-10gbps
- Bus
    - Popular in mid 90s
        - All nodes in same collision domain share same bus (can collide)
        - CSMA/CD for media access control because shared!
- Star:
    - Prevails today
        - Active switch in the center (central hub)
        - Each spoke runs a separate ethernet protocol (nodes do not collide)
        - No sharing, no CSMA/CD

Ethernet frame structure

- Sending adapter encapsulates ip datagrams or other network layer protocol in ethernet frame
- Preamble
    - Start of frame is recognized by a preamble
        - Seven byte with pattern 10101010
        - Start of frame delimiter (SFD) 10101011
    - Used to synchronize receiver and sender clock rates
    - Inter frame gap is 127 bytes of idle state

- Ethernet frame
- addresses 6 byte source and destination MAC Addresses
    - If adapter receives frame with matching destination address, or with broadcast address (eg ARP packet) it will pass data in frame to the network layer protocol
    - Otherwise adapter ignores frame
- Type
    - Indicates higher layer protocol (mostly ip)
- CRC: cyclic redundancy check at receiver
    - Error detected: frame is dropped

Ethernet: unreliable connectionless protocol

- Connectionless
    - No handshaking between sending and receiving NICs
- Unreliable
    - Receiving NIC doesn't send acks or nacks to the destination NIC
    - Data dropped in frames recovered only if initial sender uses a reliable data transfer such as tcp otherwise drop = loss
- Ethernets MAC protocol: unslotted CSMA/CD with binary backoff

Ethernet standards: link and physical layers

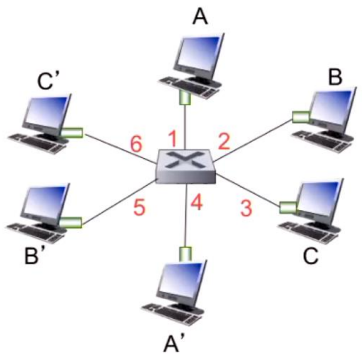
- Many different Ethernet standards
    - Common MAC protocol and frame format
    - Different speeds: 2mbps, 10mbps, 100mbps, 1gbps, 10gbps, 40gbps, 100gbps
    - Different physical layer media: fibre and cable

Ethernet switch

- Link-layer device takes an active role
    - Store and forward the ethernet frames
    - Examine incoming frame's MAC address, selectively forwarding frames to one or more outgoing links when frame is to be forwarded on segment, uses CSMA/CD to access segment
- Transparent
    - Hosts are unaware of presence of switches
- Plug and play, self learning
    - Switches do not need to be configured

Switch: multiple simultaneous transmissions

- Hosts have dedicated, direct connection to switch
- Switches buffer packets
- Ethernet protocol used on each incoming link but no collisions: full duplex
    - Each link is its own collision domain
- Switching A-to-A' and B-to-B' can transmit simultaneously without collisions
- A switch with six interfance



switch with six interfaces
(1,2,3,4,5,6)

Switch forwarding table

- How does switch know A' is reachable via interface 4, B' reachable via interface 5 etc etc.
- Each switch uses backward learning, each switch has a switch table and each entry consist of
    - MAC address of host interface to reach host and time stamp
    - Looks like a routing table
    - The switch forwarding table is made through backward learning

Switch self-learning

- Switches learn which host can be reachable through which interfaces
    - When frame is received the switch learns the location of the sender and incoming LAN segment
    - Records sender/location pair in switch table
- Once it receives it will check the source for the mac address + interface, and will be able to link the mac address to the interface
- It will then send the received frame's interface as a broadcast and see's which one replies
- Only the source will reply so it will know that the source was where to link

Switch: frame filtering/forwarding

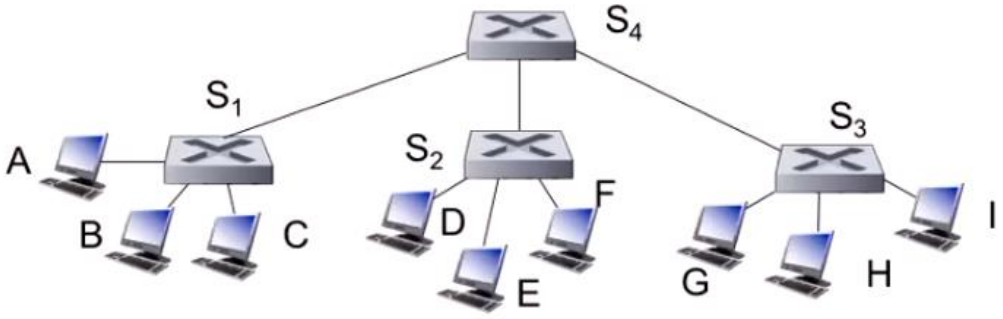
when frame received at switch:

1. record incoming link, MAC address of sending host
2. index switch table using MAC destination address
3. if entry found for destination
   then {
   if destination on segment from which frame arrived
   then drop frame
   else forward frame on interface indicated by entry
   }
   else flood /* forward on all interfaces except arriving interface */

Inter-connecting switches

- Switches can be connected together in a hierarchy similar to DNS
- Subnets are deployed in this way
-

❖ switches can be connected together



If a wants to learn where g is, A will broadcast a frame with src as A destination G

Goes through s1->s4->s2,s3

Only G will unicast back to A

Each switch will know the entry of A since it is broadcasted

Spanning tree protocol

- Can we have loops in the topology
- For redundancy
- We want LAN switches to just work with any topology
    - A simple plug and play
- Spanning tree will shut down specific links which can cause loops
- All switches run the algorithm in parallel initially empty
- Send messages to find out the links to other switches
- Can adapt to link changes/switch failures
- Each switch initially believes it is the root
- Each switch sends periodic updates to neighbors address, address of root, distance in hops to root
- Switches favour ports with shorter distance to root
- We shut down some links even if can shut down direct links for an OVERALL better performance

Switches vs routers

- Both are store and forward
- Routers are network layered device and examine network layer headers
- Switches are link layer devices examining link layer headers
- Both have forwarding tables
- Routers compute tables using routing algorithms and ip addresses
- Switches learn forwarding table using flooding and learning and MAC addresses
- Switches require often broadcasting

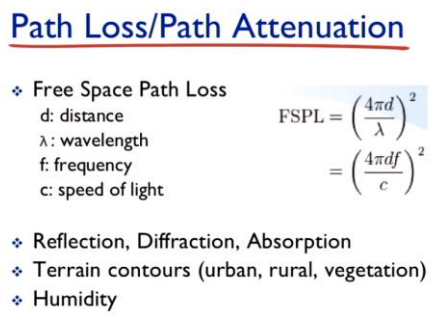
Security issues

- In a switched LAN once the switch table entries are established frames are not broadcast
    - Sniffing frames is harder than pure broadcast LANs
    - Note: attacker can still sniff broadcast frames and frames for which there are no entries (as they are broadcast)
- Switch poisoning
    - Attacker can fill up switch table with rubbish entries by sending large amount of frames with some bogus source MAC address
- Since switch table is full genuine packets frequently need to be broadcast as previous entries have been wiped out

Wireless Networks

- Wireless 101
    - Lambda - Wavelength is the distance between 2 peaks
    - C = speed of light
    - F – frequency = c/lambda
- Elements of a wireless network
    - Wireless hosts so any device
    - Base station, access points or cell towers
    - Wireless link, used to connect mobiles to base station, used as backbone link
    - Infrastructure mode, base station connects mobiles
    - Non infrastructure mode, ad hoc mode, each node acts as a router

- Wireless network taxonomy
    - Single hop or multiple hops
- Wireless link characteristics
    - Important differences from wired link
        - Decreased signal strength, radio signal attenuates as it propgates through matter
        - Interference from other sources
        - Multipath propogation, reflection off objects ground arriving at destination at different times
        - Makes communication more difficult



The loss is at a rate of 1/d^2 where d = distance

Signal to noise ratio (SNR)

Only thing we have control over is signal strength

- Larger SNR = easier to extract signal from noise (very good)
- SNR vs BER (bit error rate) tradeoffs
    - Given physical layer, increase power → increase SNR → decrease BER
    - Given SNR: choose physical layer that meets BER requirement to give the highest throughput
        - SNR may change with mobility: dynampically adapt physical layer (modulation technique, rate)

Hidden terminal problem

- 3 nodes A B C
- A B can hear each other
- B C can hear each other
- A cannot hear C which means AC and unaware of their interference fat B
- Carrier sense will be ineffective
- Signal attenuation

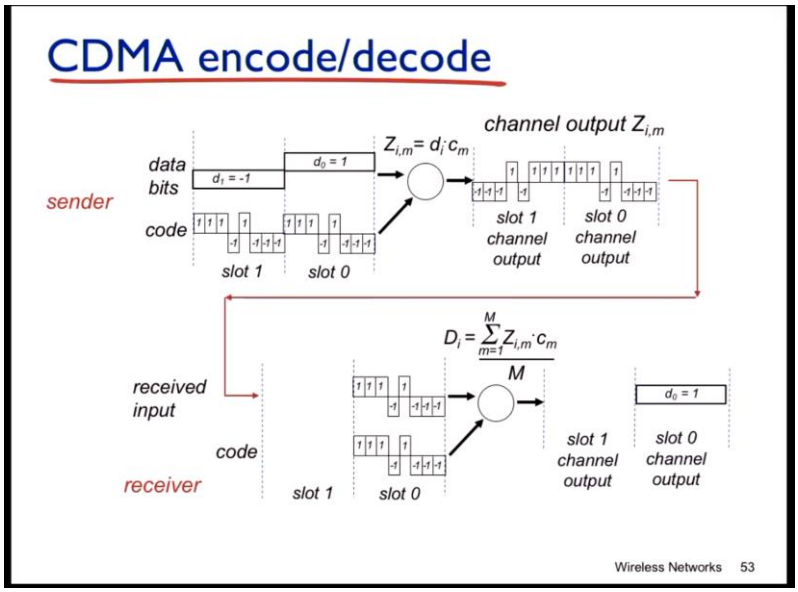
Exposed Terminals

- 4 nodes A B C D,
- Node B sends packet to A, C hears this decides not to send a packet to D since it senses busy channel
- Carrier sense prevents a successful transmission

- Prevents transmissions

CDMA (code division multiple Access)

- Unique code assigned to each user: code set portioning
    - All users share same frequency but each user has own chipping sequence to encode data
    - Allows multiple users to co-exist and transmit simultaneously with minimal interference if we have orthogonal codes
    - Encoded signal = original data * chipping sequence
    - Decoding: inner product of encoded signal and chipping sequence



802.11 LAN architecture

- wireless host communicates with base station

- base station is the access point

- basic service set (BSS) aka the cell, in infrastructure mode contains

    - wireless hosts

    - access point: base station

    - ad hoc mode (hosts only)

802.11 channel assosciation

- AP admin chooses frequency for AP
- Interference is possibe: channel can be same as that chosen by neighboring AP
- Host must assosciate with an AP
    - Scans channels listening for beacon frames containing AP's name SSID and MAC address
    - Selects AP to assosciate with
    - May perform authentication

    - Will typically run DHCP to get IP addresses in AP's subnet

802.11 passive/active scanning

- Passive scanning
    - Beacon frames sent from APs
    - Assosciation request frame sent H1 to selected AP
    - Assosciation response frame sent from selected AP to H1
- Active scanning
    - Probe request frame broadcast from H1
    - Probe response frames sent from AP's
    - Assosciation Request frame sent H1 to selected AP
    - Assosciation response frame sent from selected AP to H1

IEEE 802.11 multiple access

- Avoid collisions: 2* nodes transmitting at the same time
- 802.11 CSMA – sense before transmitting
    - Don't collide with ongoing transmission by other node
- 802.11: no collision detection
    - Difficult to receive sense collisions when transmitting due to weak received signals (fading)
    - Cant sense all collisions in any case: hidden terminal or fading
    - Our goal is to avoid collisions: CSMA/CA (collision avoidance)

Multiple Access: key points

- No concept of a global collision
    - Different receivers hear different signals
    - Different senders reach different receivers
- Collisions only occur at the receiver NOT THE SENDER
- The goal of the protocol is to detect if the receiver can hear the sender and tell the senders who might interfere with the receiver to shut up

CSMA/CA

Distributed coordination function (DCF)

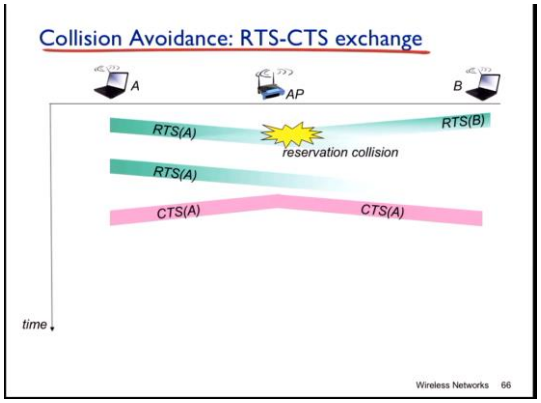
- On sender side
    - 1. Sense if channel is idle for DIFS then transmit entire frame (no CD)
    - If sense channel is busy then start random backoff time, and the timer counts down while channel is idle, and transmit when timer expires, if no ACK increase interval and repeat
- On receiver side
    - If frame received ok
        - Return ACK after SIFS (ACK needed due to hidden terminal problem

DIFS = DCF inter frame space

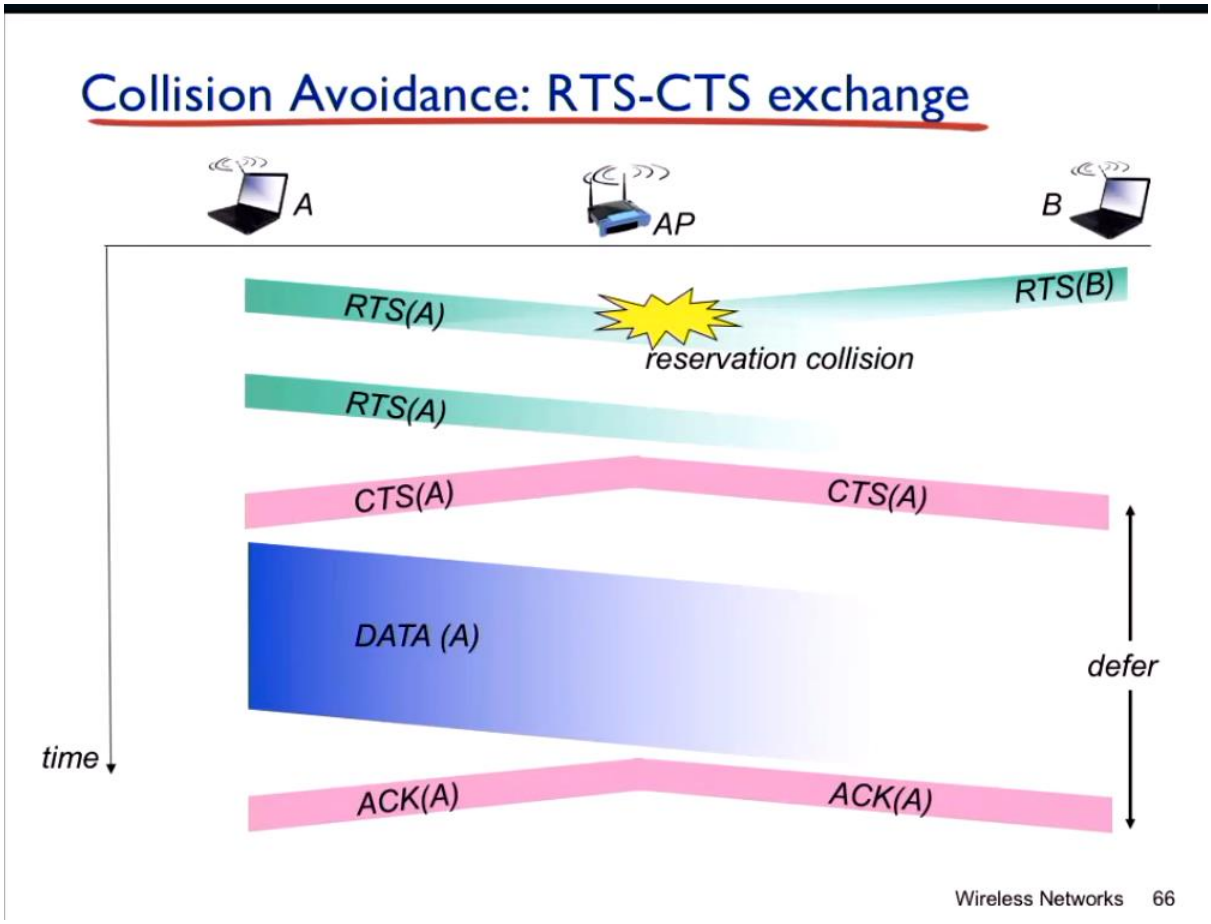
SIFS = short inter frame space

- Allow the sender to reserve channel rather than random access of data frames

- Sender first transmits small request to send (RTS) packets to BS using CSMA
    - RTS may still collide but they are very short
- BS broadcasts clear to send CTS in response to RTS
- CTS heard by all nodes
    - Sender transmits data frame
    - Other stations defer transmissions
- We avoid data frame collisions completely using small reservation packets think of restraunt bookings to avoid collisions

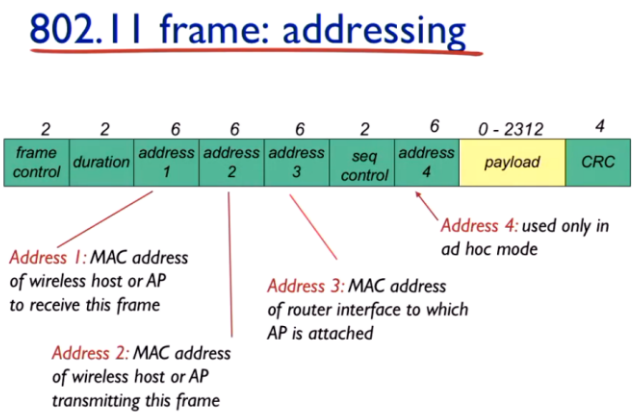


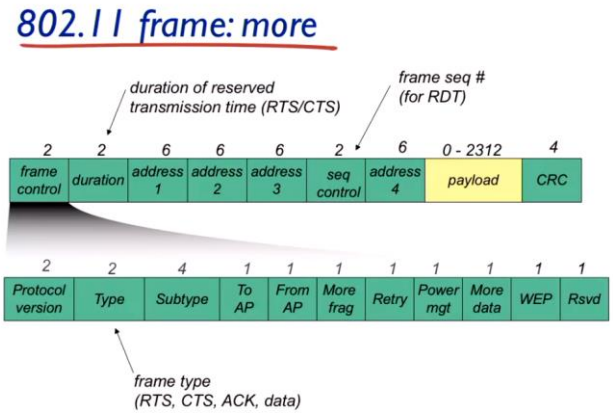
B will hear the CTSA which also has a given time, and when A is done transmitting data, it will broadcast a ACK A so it's free game after



RTS,CTS allows for solution to both hidden and exposed terminal problem

802.11 frame addressing





Day in the life of a web request

- Open a browser and type address of web server
- First we need an IP address so we use DHCP to get its own IP address and DHCP runs over UDP which runs over IP which runs over ethernet finally sent over physical layer
- Ethernet frame broadcast on LAN received at router running DHCP server
- DHCP performs an offer to offer an ip address to the original laptop
- The host will go up the stack and then sends a DHCP request again,
- DHCP will send back an ACK, and the ack contains ip, first hop router etc.
- (DHCP broadcasts because it does not know where the request is coming from)
- Before sending HTTP request, we need to perform DNS Request, and before that we also need to use ARP to know mac address
- ARP used to get address of first hop router, so we can perform our DNS query
- TCP connection between yourself and google to send HTTP request

Network Security

Confidentiality – only sender and receiver should be able to read the message

- Sender encrypts
- Reader decrypts

Authentication

- Sender and receiver must be able to confirm the identity of who is who

Message integrity

- Sender and receiver want to ensure the message has not been modified without being detected and also the receiver knows that the sender is who sent the message

Access and availability

- Services must be accessible and available to users

What can bad guys do

- Eavesdrop – interception of messages
- Insert messages
- Modify messages
- Impersonation by faking the source address in packet or any fields in the packets
- Hijacking takeover, ongoing connection is overtaken by removing sender or receiver and replacing with themselves
- Denial of service (prevent a service from being used eg. DDOS) overloading resources SEND THE PACKETS

Steganography

- Covered writing
- It conceals the existence of the message
    - Character marking – selected letters of printed text are over-written in pencil
    - Invisible ink – writing something down which leaves no trace but using chemicals or heat or uv light can uncover the messages
    - Pin punctures – small pin punctures on selected letters are ordinarily not visible unless paper is held in front of a light
    - Typewriter correction ribbon – used between lines typed with a black ribbon, result is only seen under strong light
- By obscuring the messages we can hide it and the receiver can read the message using a special technique
- Security by obscurity
- Subset stegnography
    - For example hiding message in last word of each line in the paragraph or the 2nd word on each line
- Null Cipher
- First letter of each word becomes the letter for encryption for example

- ROY G BIV = red orange yellow green blue indigo violet
- Can apply offset to null Cipher
- Photo steganography – we change the least significant bit of a byte with almost no visible change to the image allowing us to hide a message inside the bytes of the image (there are 256 variations of grey)
- This can also be used with RGB colored images (each R G B is a byte)
    - Advantages are that it does not change the size of the file still same number of bytes
    - Disadvantages are that conversion may lead to loss in hidden message
- Using TCP/IP packets as a cover
- We can embed text directly into the headers
    - IP Identification can be modified too for UDP
    - The sequence number can be a message when converted and modified easily

Cryptography

- M → plaintext message
- Ka(m) → ciphertext, encrypted message with key Ka
- M → Kb(Ka(m)) the message decrypted with key Kb

Symmetric key cryptography

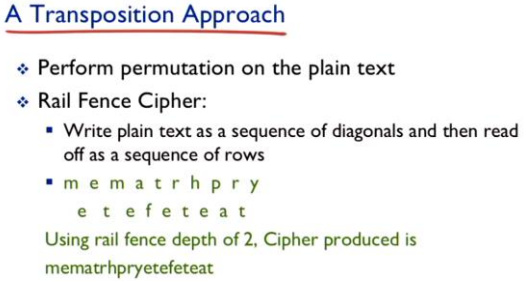
- Ks is a key shared that both encrypts and decrypts the messages
- Both sender and receiver need to use the same key so we need to make sure they use the same KEY AND THE INTRUDER DOES NOT KNOW THE KEY

We can use

- Substitution cipher
    - Substituting one thing for another
    - Monoalphabetic cipher, substitute one letter for another
    - Ceaser cipher, replace each letter of the alphabet with the letter standing three places further down the alphabet
    - Encryption key would be replacementCharacter = (character+3) mod 26
    - The shift can be changed +3 can be changed to + k
    - Disadvantage
        - With only 25 possible keys it is very easy to brute force the key value
        - Cipher can be any permutation of the 26 alphabet characters
        - To counter this we can map the 26 letters to a set of 26 encryption letters to increase number of possible keys to 26! Rather than just 25 which is much more powerful of than traditional ceaser cipher

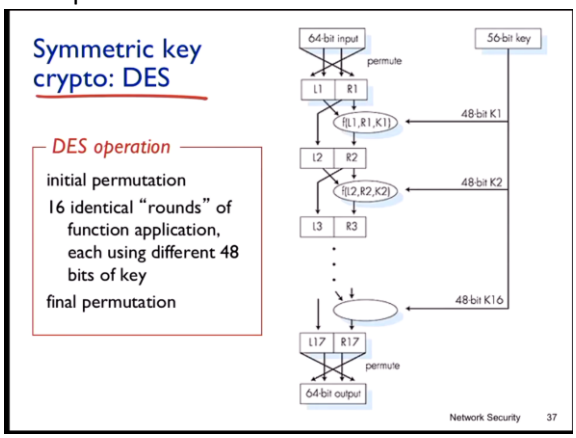
Breaking encryption scheme

- Cipher-text only attack
    - Trudy has ciphertext she can analyse
- Two approaches for cipher-text-only attack
    - Brute force (search based analysis)
    - Statistical analysis
- Known-plaintext attack
    - Trudy has part of the plaintext corresponding to ciphertext

- Eg. In monoalphabetic cipher Trudy determines pairings for letters a b c d e f g h i
- Approach is a chosen plaintext attack
  - Trudy can get ciphertext for the chosen plaintext
- Statistical analysis
  - Letter analysis ETAOIN SHRDLU
  - These are the most frequent letters that occur in the alphabet
  - Monoalphabetic ciphers can be broken easily because they reflect the same frequency of the original letter analysis
  - We can also identify 2 letter, 3 letter words or even single letter words
- Encrypting diagrams
  - Playfair cipher
    - Treats diagrams in the plaintext as single units and translates them into ciphertext diagrams
    - We use a 5x5 matrix of letters using the alphabet
    - Then we use a keyword to highlight in the matrix beginning
    - Then we encrypt two letters at a time by moving based on where the letters lie on the remaining matrix
- Polyalphabet ciphers
  - We can use n substitution ciphers, M1, M2, M...Mn
  - A cycling pattern so for example we set our cycle to 3 and the key becomes
    - M1,M3,M2,M1,M3,M2…. And so on
    - The goal is to make sure repeat letters try to get different value to break statistical analysis
- Vigen'ere Cipher
  - 26 substituion ciphers with shifts of 0 through to 25
  - Cycling pattern
    - Assume key consists of m sequence of letters
    - Since we are cycling through, there are multiple ciphertext letters for each plaintext letter and this allows for further breaking of statistical analysis
- A transposition approach
  - Perform permutation on the plain text
  - Rail Fence Cipher
    - Write plaintext as a sequence of diagonals and then read off as a sequence of rows
    - Now we can read it in column form of depth n to get th emessage
    - 

  - We can also write plaintext in a rectangle, row by row and then read the message off column by column but permute the order of the column
  - The column order is now the key of the algorithm

        - These still have same letter frequency however we can apply multiple transpositions to deal with that
    - Stream ciphers
        - Encrypt byte stream one bit at a time
        - Combine each bit of keystream with bit of plaintext to get bit of ciphertext
        - We exclusive or between message and keystream, and the receiver will perform the exclusive or with the keystream and that to receive the original message
        - RC4 is a popular stream cipher
        - Used in WEP for 802.11
        - Can be used in SSL
    - Block Ciphers
        - Break plaintext message into equal-sized blocks
        - Encrypt the block as a whole
        - 1-1 mapping is used to map k bit blocks of plaintext to k bit block of ciphertext
        - There are k! permutation of keys available where k is the number of bits in each block
        - To prevent brute force attacks
            - Choose large K(64,128,etc)
            - This creates maintenance and indexing problems as we need 2^k indexs in the table and search can be expensive
            - To counter this we use subtables to help with the encryption
            - We pass N/t bits into each separate table where t is the number of tables and N number of bits
            - We then scramble the 64-bit accumulation of each section of bits
            - Then we pass that as our output
            - This is used in DES AES or 3DES

- Symmetric key cryptography: DES
  - DES stands for Data Encryption Standard
  - US encryption standard
  - 56-bit symmetric key, 64-bit plaintext input
  - Block cipher with cipher block chaining
  - DES suffers from security issues
    - 56-bit-key encrypted phase can be brute forced in less than a day
    - No known good analytic attack
  - Making DES more secure
    - 3DES: encrypt 3 times with 3 different keys (yeah bro just add more obscurity)
  - DES operation
  - 

- AES
  - Advanced encryption standards
  - Symmetric key NIST standard replaced DES
  - Processes data in 128 bit blocks
  - 128 192 or 256 bits
  - Brute force decryption try each key taking 1 second on DES, takes 149 trillion years of AES
- Cipher Block Chaining (CBC)
  - If input block repeated it will produce same cipher text
  - Suppose we have (ACK ____) always appearing in a response
  - Since we know ACK always starts at the message we can use this to our advantage to decrypt
  - What we can do is use random numbers, and XOR the input with the number to get a new message, each time sending the random number to the host, however this will double the message size as we constantly need to send our random number
  - Generate our own random numbers
  - Have encryption of current block dependant on the result of the previous block
  - C(i) = ks(m(i) EOR c(i-1))
  - M(i) = ks(c(i)) EOR c(i-1)
  - To encrypt the first block we use an initialization vector: random block c(0)
  - We don't need to hide this
  - Change IV For each session or message

        - Guarantees that even if the same message is sent repeatedly the ciphertext will be completely different each time
    - In DES cipherblock chaining, each plaintext block is EOR'd with the previous ciphertext block before being encrypted, the reason we use EOR over AND or an OR is that Exclusive OR applied twice will return the original value EOR has no bias to distribution of 0's and 1's aswell!

Public Key Cryptography

- Requires sender and received to know a shared secret key
- But how do we decide on what key to use if both sender and receiver have never met
- To solve this we use RSA public key cryptography
- Sender and receiver don't share a secret key
- Public encryption key which can be known to everyone and all
- A private decryption key known only to the receiver
- We use public keys to decrypt encrypted messages that were encrypted with OUR public key.
- Keys are generated in such a manner that even if you know the public key you cant find out the private key!
- However using RSA we can use other peoples public RSA key to pose as someone else, say we have bob alice and Trudy, we can use bobs public key with trudys private key to pose as alice, and then set up a hijack
- Since everyone knows public keys we need authentication

Public key encryption algorithms

- We need a public and private key K+ and K- such that
    - K-(k+(m)) = m
- Given public key K+ it should be impossible to compute private key K-
- RSA (rivest shamir Adelson algorithm)
- Messages are just a bit pattern and we can represent bit patterns as a integer
- So encrypting a message is the same as a number (think of casting bits)
- Example m = 1001 0001 can be represented as 145 in binary
- To encrypt m we encrypt 145 to give a new number which can be decrypted to give 145

RSA algorithm

1. Choose two large prime numbers P and Q (1024 bits each or more)
2. Compute n = pq, z= (p-1)(q-1)
3. Choose e where (e<n) that has no common factors with z (e and z are relatively prime)
4. Choose d such that ed-1 is exactly divisible by z, ed mod z = 1
5. Public key is (n,e) private key is (n,d)
6. Now we can compute encryption and decryption
7. To encrypt message m, we say c = m^e mod n
8. To decrypt ciphertext c, we say m = c^d mod n

EVEN THOUGH WE KNOW N WE CANNOT RELATE E TO D

ITS MAGIC because m = (m^e mod n)^d mod n

A property of RSA encryption is that it does not matter which key is applied first!

In the alice and bob case, if bob sent the message first we would know its from bob as it will have both his private and public key, which acts as a signature,

Suppose we know a public key how hard can it be to determine the private key, well we would need to find factors of n without knowing the two factors of p and q which is insanely difficult almost impossible especially since numbers chosen are enormous

RSA in practice: session keys

- RSA is computationally intensive
- DES is atleast 100x faster than RSA
- Use public key crypto to establish secure connection then establish second key symmetric session for encrypting data
- So we use RSA to exchange a symmetric key, once we have our symmetric key we can use symmetric key cryptography with no one else knowing what our symmetric key is
- This solves the main issue of delivering a symmetric key between to hosts securely!
- This is the best of both worlds!

Authentication

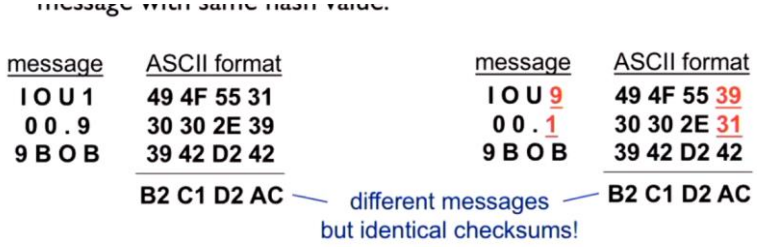
- Protocol AP1.0: sender says “I AM SENDER”
    - Failure scenario
        - In a network sender and receiver can’t see each other so
        - If anyone else says “I AM SENDER” the receiver will believe it
- Protocol AP2.0: sender says “I AM SENDER” in an IP packet containing the sender IP source
    - Failure scenario
        - Someone can create a packet spoofing the sender ip source field
- Protocol AP3.0: sender says “I AM SENDER” and sends the secret password to prove it
    - Failure scenario
        - Someone can perform playback attack, recording the secret initiation with the password and later use it for the receiver
- Protocol AP3.1: sender says “I AM SENDER” and instead sends an encrypted secret password to prove it
    - Failure scenario
        - Still fails to playback attack in this case the attacker wont know the password only the encrypted password however the receiver will only be seeking the encrypted password
- Protocol AP4.0
    - Avoids playback attack
    - Uses a NONCE: a random number only used once in the communication
    - To prove the sender is “live”, the receiver sends the sender a Nonce, the sender must then return the nonce encrypted with the shared secret key
    - The attacker cannot send back the encrypted nonce as the attacker cannot use the same key shared between sender and receiver
    - Attacker cannot use the same encrypted key as a nonce is only used once, in another communication we use a different nonce

- Protocol AP5.0
    - AP4.0 requires a shard symmetric key
    - Can we authenticate using public key cryptography
    - AP 5.0 uses nonce, public key cryptography
    - Now that we are using public key cryptography we can use our private key to send our nonce encrypted with the private key and send over the public key
    - Failure scenario
        - Man in the middle attack
        - We intercept the message and send back the nonce encrypted with the attacker private key, and make sure communication to the two look normal, then the attacker sends their public key rather than the senders public key and all of the sender's replys are dropped by the man in the middle. This way we can send the message encrypted to the senders public key without anything looking suspicious, this allows us to get messages from the receiver encrypted with the attacker public key, and the attack can send the encrypted message with the senders public key
        - Full control attack which leaves no trace, as both sender/receiver will have communicated and seen the communication without any idea of anyone in the middle (FBI WEBCAM I KNOW UR THERE).
        - This attack is also very difficult to detect as both sender and receiver received and sent all messages however the attacker also received every message sent

Message Integrity

- Digital signatures
    - Cryptographic technique analogous to hand-written signatures
    - Sender digitially signs document, establishing they are the document creater/owner
    - Verifiable and non-forgeable, receipent can prove to someone that the sender and no one else has signed the document
- Simple digital signatures for message m
    - Sender signs message by encrypting with their private key creating a signed messaged
    - Send both the plaintext and cyphertext, we are using this primarily for digital signature
    - We use encryption for verification
    - However this doubles the size of message sent
    - Since the cipher text is encrypted with the private key, anyone can decrypt with the public key to see clearly who created the signature since only "1 person has access to private key"
    - We can verify that whoever signed with the public key is the owner/creator
        - And no one else signed
    - It is non-repudiation
        - You can take the message and the signature to court and prove who signed the message
- This is too expensive as public private encryption is extremely expensive, so we do not forward the signature of the complete message, rather than sending the message encrypted with the private key essentially doubling amount sent in total, we use message digests (a hash function), this will reduce the message to a smaller space of a fixed size, and then we

encrypt the hash, think of it like hashing, you get one big message, can use a hash function to reduce the message, and the encrypt the hash function with the private key, and this process can be reversed
    - The hash returns the message digest (the fingerprint)
    - We want a hash function such that the message hash is unique so no collisions!
- Internet checksum is possible but its very poor, since a re-ordered message can have the exact same checksum even though the messages are different



    -
- Hash function algorithms
    - MD5 hash function widely used (RFC 1321)
        - Computes 128 bit message digest in 4 step process
        - Arbitrary 128-bit string appears difficult to construct message whose MD5 hash is equal to the message
    - SHA-1 also used
        - US standard
        - 160-bit message digest
    - SHA-2 and SHA-3 are recent standard and provide better security since SHA-1 and MD5 can easily be cracked
- Signed message digest procedure
    - Send the digitially signed message
    - First we pass the message through the hash to obtain the digest (fingerprint)
    - Then we can apply the private key as an authentication to prove only you could have been the one to sign the message
    - Send both the message and the encrypted digest as evidence for integrity
    - Essentially we are sending the message AND proof of the message being sent by us
        - ON THE RECEIVER END
    - We receive the message and the encrypted hash
    - First we use the public key to decrypt the encrypted hash
    - Then we apply the same hash function to the message received
    - Then we want to compare the decrypted hash to the computed hash to make sure they are equal
    - If not (FRAUD??)

- NOW BACK TO THE MAN IN THE MIDDLE ATTACK
    - Public key certification (to prevent the whole attack since the receiver assumes the attackers public key is indeed the senders public key the attack continues, if there is a way to verify that the public key received on the receiver has come from the sender than we can prevfent this attack)

    - Say for instance this situation is given

    - CA (certification authority) binds the public key to a particular entity E
    - The entity which is either a person or router registers its public key with the certification authority
        - E provides proof of identity to the CA
        - CA creates certificate binding the entity to it's public key
            - It will take the public key and bind it to the CA's private key for authentication purposes
            - This is called signed by the certification authority
        - The CA will then advertise its more secure public key rather than your own key
        - Certificate containing E's public key is digitally signed by CA-CA saying "this is E's public key"
        - Now when anyone wants the entity E's public key it would
            - Get the certificate from either E or something else
            - Apply the CA's public key to the certificate of E to get the public key for E
            - This is an authorised way to get a public key!! Security for public key!
        - Certificates contain
            - Serial number (unique to whoever issued the certificate)
            - Information about the owner, algorithm, key value which is omitted and the fingerprint (signature from CA)
    - Certificate summary
        - Primary standard is X.509 (RFC 2459)
        - Certificate details above
        - Public key infrastructure (PKI)
            - Certificates and certification authorities large overhead.

Quiz

- Suppose Bob wants to send Alice a digital signature for the message *m*. To create the digital signature
  - A) Bob applies a hash function to *m* and encrypts the result with his private key
  - B) Bob applies a hash function to *m* and encrypts the result with Alice's public key
  - C) Bob encrypts *m* with his private key and then applies a hash function to the result
  - D) Bob applies a hash function to *m* and encrypts the result with his public key

A easyyyyy

Quiz

- Suppose a CA creates Bob's certificate, which binds Bob's public key to Bob. This certificate is signed with
  - A) Bob's private key
  - B) Bob's public key
  - C) The CA's private key
  - D) The CA's public key

C easy wtf