489

Lecture 1

- The internet consists of many **end-systems** (devices with IP)
- **Switches** connect many ends: router(for core) and link-layer switches(for access networks)
- **Link** connect between them (fibers and lines)
- Internet managed by many parties
    - Cable company, etc
    - **Internet Service Provider** who own physical infrastructure to provide service.
- Transport
    - **Data** transfer through **path**

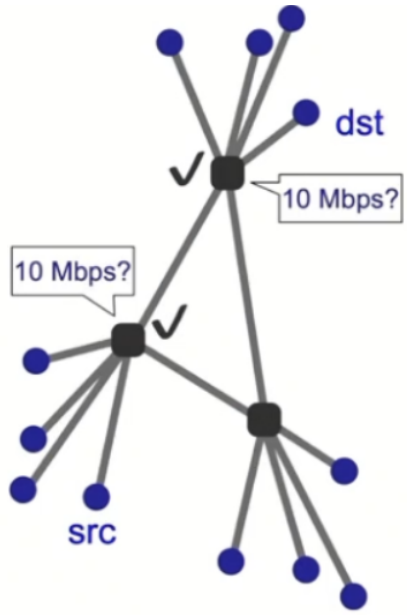
A federated system

- The internet ties together different networks by **IP protocol**
- Switched networks: end-systems and networks connected by switches instead of directly connecting them -- we want to **scale**, directly connecting N nodes need N^2 links
- End systems attached to the internet provide a **socket** interface to deliver data to a specific dest program running on another end system

Network sharing

- congestion control
- 2 ways of sharing
    - Circuit switching (**reservation**)
        - Resource reserve per connection
        - Admission control: per connection
        - Cons: you need to tell how many resources are needed forehead
            - Inefficient since same amount for both busy and leisure time
            - Complexity+delay of circuit setup/teardown
            - switch fails -> its switch fails

- Pros:
  - quality ensure (predictable performance)
  - Simple/fast switching (once circuit established)
  - 1. src sends reservation request to dst
    2. Switches create circuit *after* admission control
    3. src sends data
    4. src sends teardown request

    - When the network establish the circuit(connection), it also reserves a constant transmission rate in the network's links for the duration of connection
    - Example: if a link has 4 circuits, and each link between adjacent switches has a transmission rate of 1 Mbps, then each end-to-end circuit-switch connection gets 250 kbps of dedicated transmission rate.
- Packet switching via statistical multiplexing
  - Packets treated independently, on-demand
  - Admission control: per packet
  - Each packet contains dst, treated independently
  - With buffers to absolve transient overloads
  - Pros:
    - Efficient use of resources
    - Simpler to implement
    - Robust
  - Cons:
    - unpredictable performance
    - requires buffer management and congestion control

        - **allowing more demands than the network can handle**

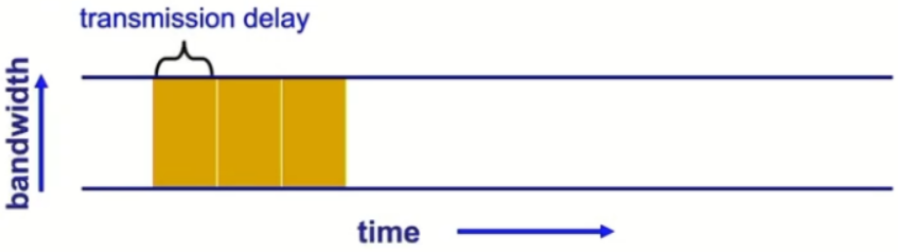
Store-and-forward Transmission

- packet switch must receive the entire packet before it can begin to transmit the first bit of the packet onto the outbound link. The router cannot transmit the bits it has received; instead it must first buffer/store the packet's bits. Forward after the router has received all of the packet's bits
- Let’s now consider the general case of sending one packet from source to destination over a path consisting of N links each of rate R (thus, there are N-1 routers between source and destination).
  Applying the same logic as above, we see that the end-to-end delay is:
- $$end\text{-}to\text{-}end = NLR$$

Forwarding Table and Routing Protocols

- Each router has a **forwarding table** that maps destination addresses to router's outbound links.
- Routing protocols are used to automatically set the forwarding tables.

Performance metrics

- Delay
    - How long does it take to send a packet from its source to destination
    - Four components
        - **Transmission delay**: how long the sender take to send the package (push all bits of a packet into a link, **packet can be transmitted only after all packets have arrived**)
            - Different links transmit data at different rates: transmission rate
            - **Packet size(L) / transmission rate of the link(R)**
            - Pipe view of a link:
                - 

            - transmission delay decrease as bandwidth increase

        - **Propagation delay**: how long the package take to traverse the network (physical distance)
            - Link bandwidth: number of bits sent/received per unit time -- bits/sec, bps
            - delay x bandwidth -- propagation delay: max# of flying bit on link
            - **Link length(d) / Propagation speed of link(s)**
        - **Queuing delay**: Buffering, before the package leave the switch
            - Buffer has limited compacity: result in dropping
            - Depends on traffic pattern:
                - average queuing delay
                - variance of queuing delay
                - probability delay exceeds a threshold value
            - Basic terminology:
                - **A**rrival process: how packets arrive
                    - Average rate A
                - Average time packets **w**ait in the queue
                    - Waiting time
                - average number of packets waiting in the queue
                    - the **l**ength of queue
            - Little's Law
                - L: A x W, L is easy to compute while W is hard
        - **Processing delay**: the switch decides how to send the package(examine the packet's header), usually negligible
            - Strongly influences a router's maximum throughput, which is the maximum rate at which a router can forward packets.
    - First two depends on link properties, next two due to traffic mix and switch internals
- Loss: what fraction of the packets sent to a destination is dropped
    - Smaller size benefit from this: resending large package is wasteful
- Throughput: At what rate is the destination receiving data from the source
    - Transfer time = file size / transmission rate + propagation delay*(ignored)
    - Throughput ≈ transmission rate
    - Average throughput = min {R, R1, R2..}

- RTT: time for a packet to go from a source to a destination and to come back
    - Average end-to-end delay: RTT/2

Lecture 2

Protocol Layering

- Each protocol belongs to one of the **layers**, the **services** that a layer offers to are called **service model** of a layer
- **Protocol**: an agreement between parties on how to communicate (on same layer)
    - define **syntax** of communication
        - **header** -> instruction on accessing payload
    - **Semantics**
- Each layer provides its service by
    - Performing certain actions within that layer
    - Using the services of the layer directly below it

Structure

Applications, L7

- SMTP, HTTP, DNS, NTP

Transport Layer, L4

- Reliable or unreliable
- TCP, UDP

Network Layer, L3

- Best-effort global packet delivery
- IP
    - Single protocols gives nice abstraction and simple work

Link Layer, L2

- Best-effort local packet delivery
- Ethernet, FDDI, PPP

Physical Layer, L1

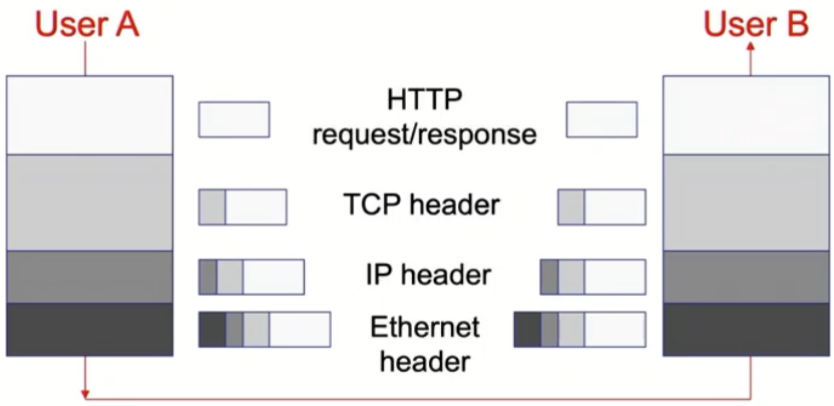
- Optical, Copper, Radio, PSTN

OSI Layers

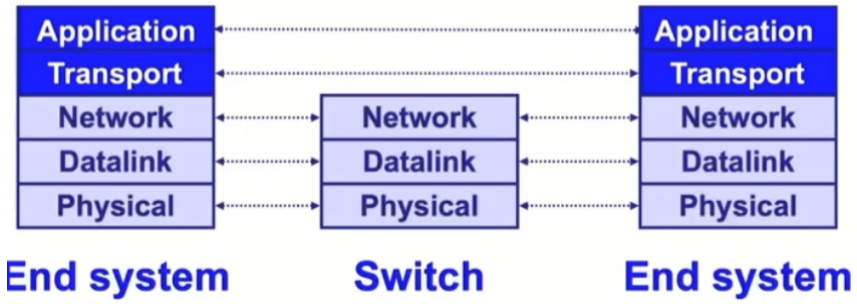
- Open Systems Interconnection model
- L5, Session, L6, Presentation. These are often implemented as part of the application layer.

Layer encapsulation

-

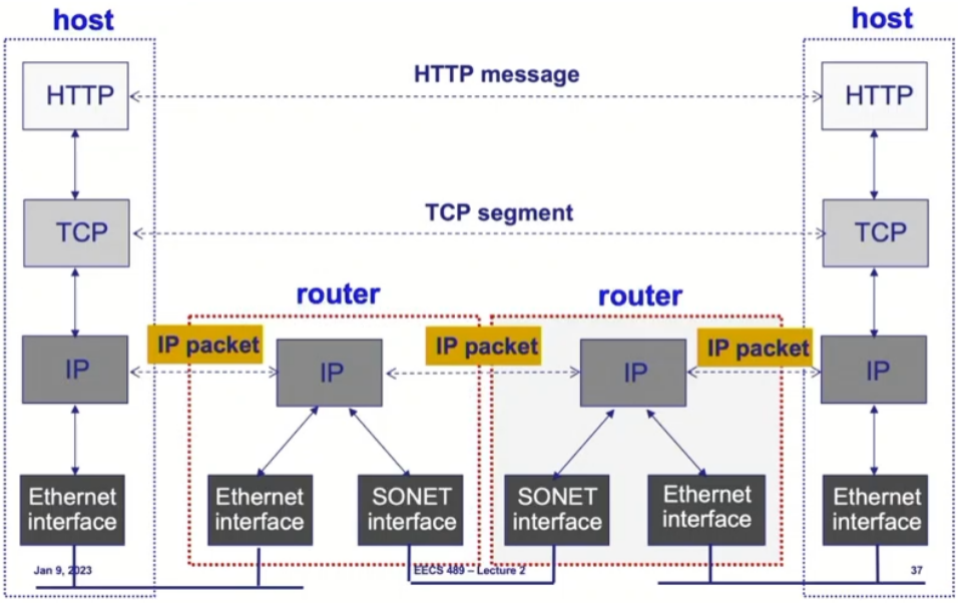


Implementation



- End systems
  - Bits arrive on wire, must make it up to application.
  - All layers must exist at host.
- Network system
  - We don't care about application -- unless for security reason
  - We only care about L1, L2, L3, does not support reliable delivery

        - Switches implement only L1, L2
        - Routers have L1, L2, L3. Switches do what routers do but don't participate in global delivery
- Logical communication
    - A layer interact with its peers corresponding layer
- Physical communication
    - Communication goes down to physical network
        - Then up to relevant layer
- A protocol-centric diagram



Pros and Cons of layering

- Pros
    - Reduce complexity
    - Improve flexibility
- Cons
    - Higher overheads
    - Cross-layer information often useful
- Implication of hourglass
    - Single network protocol
    - Allows arbitrary networks to interoperate
        - Any network that supports IP can exchange packets
    - Decouples applications from low-level networking technologies
        - Applications function on all networks

    - Support simultaneous innovations above and below IP
    - But changing IP itself is hard: IPv4 -> IPv6
        - adoption is hard, translation
        - we are running out of IPv4 address

Placing network functionality

- End-to-end arguments
    - Dumb network and smart end systems
    - Functions that can be completely and correctly implement only with the knowledge of application end host, should not be pushed into the network
        - If you add too much application specific functionality into the network its actually hard to evolve in network
        - We don't want to keep updating
    - Fate sharing: fail together or don't fail at all

HTTP and the Web

History

- First HTTP implementation - 1990
- HTTP/0.9 -- Simple GET
- HTTP/1.0 -- Client & Server
- HTTP/1.1 -- Performance and security optimization
- HTTP/2 -- Server push, Latency optimizations via request multiplexing over a single TCP connection, binary protocol instead of test
- HTTP/3 -- solves head-of-line blocking problem in multiplexing over single TCP
    - Built on top of QUIC, which is a user-space congestion control protocol on UDP, also reliable

Web components

- Infrastructure
    - Clients
    - Servers
- Content:
    - URL: naming content

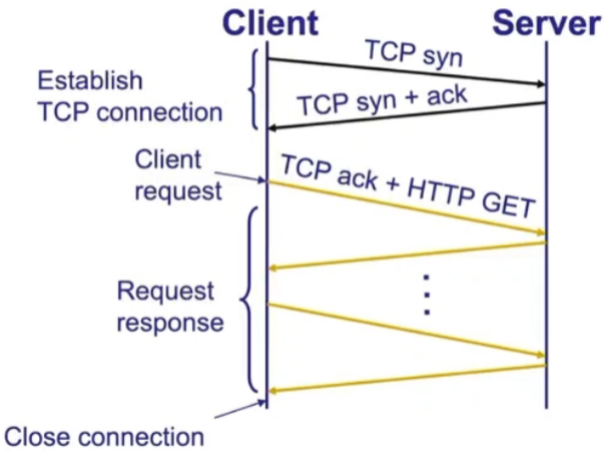
    - HTML: formatting content
- Protocol for exchanging information: HTTP

URL: Uniform Record Locator

- Protocol: //host-name[:port]/path/resource
    - protocol://host-name[:port]/directory-path/resource
        - protocol: http, ftp, https, smtp, rtsp, *etc.*
        - host-name: DNS name, IP address
        - port: defaults to protocol's standard port
            - *E.g.*, http: 80, https: 443
        - directory path: hierarchical, reflecting file system
        - resource: Identifies the desired resource
- Extend the idea of hierarchical hostnames to include anything in a file system
- Extend to program execution as well - query/js

HTTP: Hyper Text Transfer Protocol

- Client-server architecture
    - Server is always on and known
    - Clients initiate contact to server
- Synchronous request/reply protocol
    - Runs over TCP port 80
- Stateless: servers wont track state for its communication
    - Request itself is within all information needed
- ASCII format (before HTTP 2)

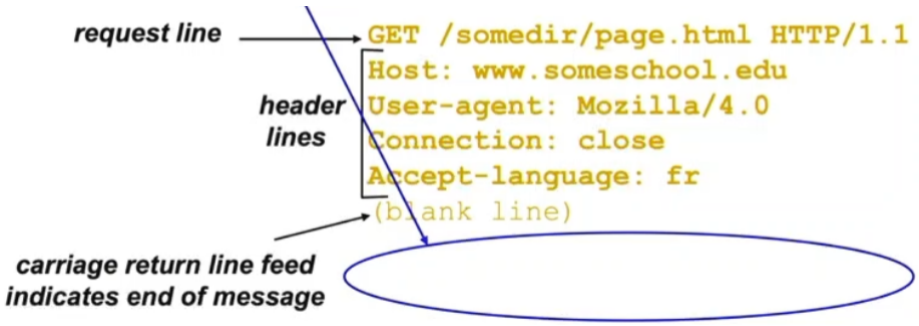


Method types (HTTP 1.1)

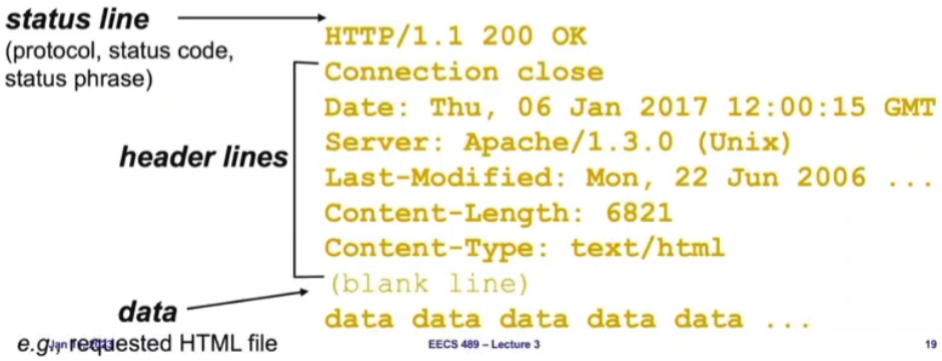
- GET, HEAD
- POST (send information)
- PUT (upload file)
- DELETE (delete file)

Client-to-server communication

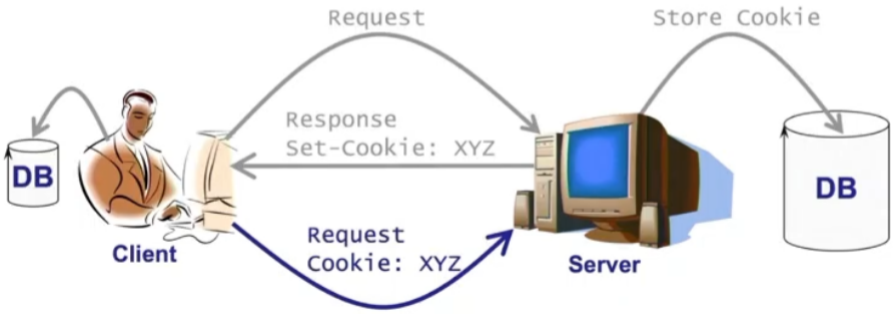
- HTTP Request message
  - Request line: method, resource, and protocol version
  - Request headers: provide info or modify request
  - Body: optional data



- HTTO Response Message
  - Status line: protocol version, status code, status phrase
  - Response headers: provide information
  - Body: optional data



- HTTP is stateless
  - Each request-response treated independently
    - Servers not required to retain state
  - Pros: Improves scalability on the server-side
    - Failure handling is easier
    - Can handle higher rate of requests

        - Order of requests doesn't matter
    - Cons: Some applications need persistent state
        - Need to uniquely identify user or store temporary info
- How to keep state
    - Cookies:
        - Client-side state maintenance
            - Client store cookie and send cookie to server
        - Provide Authentication
        - privacy concern
        - 

    - Group-based identifiers: federate learning

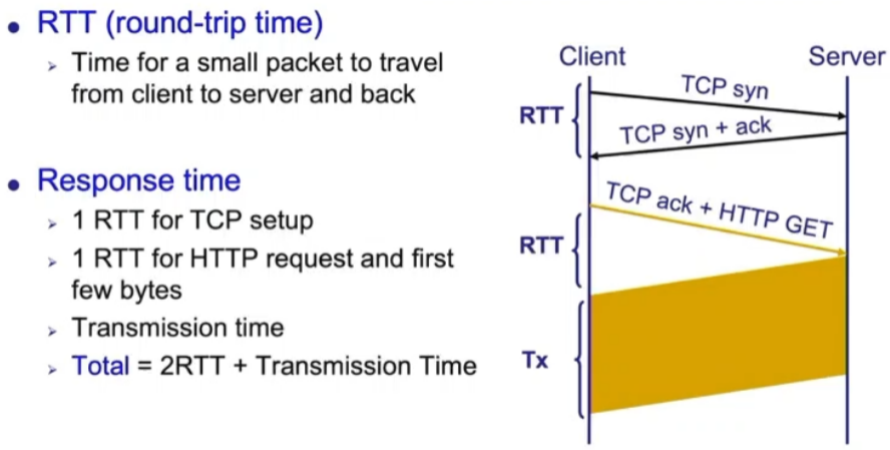
Performance goals

- User
    - Fast download -- improve networking protocols, HTTP/TCP, etc
    - High availability -- caching and replication
- Content provider
    - Happy users -- caching and replication
    - Cost -- CDNs, datacenters
- Network
    - Avoid overloading

HTTP performance

- Most web pages have multiple objects
- Retrieve one item at a time
- New TCP (handshake) connection per object -- high overhead

Object request response time



Non-persistent connections (default in HTTP 1.0)

- 2RTT + transmission time for each object in HTML file and one more for HTML file itself
- Improve by 3 "P"s

Concurrent request and responses (HTTP 1.2) bandwidth and latency

- Use multiple connections in parallel
- Does not necessary maintain order of response
- **Network might be overload**

Persistent connections (HTTP 1.1) latency

- Maintain TCP connections across multiple requests
  - Including transfers subsequent to current page
  - Client or server can tear down connection
- Pros
  - Avoid overhead of connection set-up/tear-down
  - Allow underlying layers to learn about RTT and bandwidth characteristics

Pipelined requests & responses latency

- Batch requests and response to reduce the number of packets
  - Multiple requests can be contained in one TCP segment
- Data are sent in a FIFO manner
  - Can lead to head-of-line(HOL) blocking if first of is really large
  - Not supported by default by major browsers circa 2015
- Solution

- **Priority and preemption**
- by TCP 3.0

Scorecard: Getting n small objects

- Time dominated by latency

- One-at-a-time: ~2n RTT
- m concurrent: ~2[n/m] RTT
- Persistent: ~ (n+1) RTT
- Pipelined: ~2 RTT
- Pipelined and Persistent: ~2 RTT first time; RTT later for another n from the same site

Scorecard: Getting n large objects each of size F

- Time dominated by TCP throughput B_C (<= B_L), where link bandwidth is referred by B_L

- One-at-a-time: ~ nF/B_C
- m concurrent: ~ $nF/(mB_C)$
  - Assuming each TCP connection gets the same throughput and mB_C <= B_L
- Pipelined and/or persistent: ~ nF/B_C
  - The only thing that helps is higher throughput

Caching: exploits locality of reference

- How
  - Modifier to GET requests:
    - If-modified-since: returns "not modified" if resource not modified since specified time
    - Client specifies this time in request
    - Server compare this against "last modified" time of resource
    - Server returns "Not Modified" if resource has not changed, or a latest version
  - Response header:
    - Expires : lifetime of caching resource
    - No-cache: ignore cache and always fetch from server

- Where: clients transfer same information
    - Client (browser)
    - Forward proxies
        - Cache documents close to clients
            - Reduce network traffic and decrease latency
            - By ISP or enterprises
    - Reverse proxies
        - Cache documents close to server
            - Decrease server load
            - By content provider
    - Content Distribution Network

CDN

- Replication
    - Replicate popular websites across many machines
        - Spreads load across servers
        - Place content closer to clients
        - Helps when content isn't cacheable
- CDN
    - Caching and replication as a service
    - Large-scale distributed storage infrastructure administered by one entity
        - Akamai
    - Combination of caching and replication
        - **Pull**: Direct result of client's requests (caching)
        - **Push**: Expectation of high access rate (replication) base on access of popular pattern
    - Cost-effective content delivery
        - Many sites hosted on shared physical infrastructure
            - Efficiency of multiplexing
            - Human operator costs
            - Economies of scale
        - e.g. CDNs, Web hosting companies, Cloud infrastructure

    - Akamai
        - creates new domain names for each client
            - a128.g.akamai.net for cnn.com
        - Client content provider modifies content so that embedded URLs reference new domains
            - e.g., http://www.cnn.com/image-of-the-day.gif becomes http://a128.g.akamai.net/image-of-the-day.gif
        - Requests now sent to CDN's infrastructure
- Why direct clients to particular replicas
    - Balancing load across server replicas
    - Pairing clients with nearby servers to decrease latency and overall bandwidth usage

DNS

Internet names & addresses

- Machine address:
    - Router-usable labels for machines
    - Conforms to network structure
- Machine names:
    - Human-readable labels
    - Conforms to organization structure
- The DNS is how we map from one to another -- directory

Why?

- Convenience
    - Easier to remember
- Provide level of indirection
    - Decouple names from address -- if an address changed later
    - Many uses beyond just name a host

At first people store all host into a file.

In 1983, the first stable operational DNS implementation

Goals:

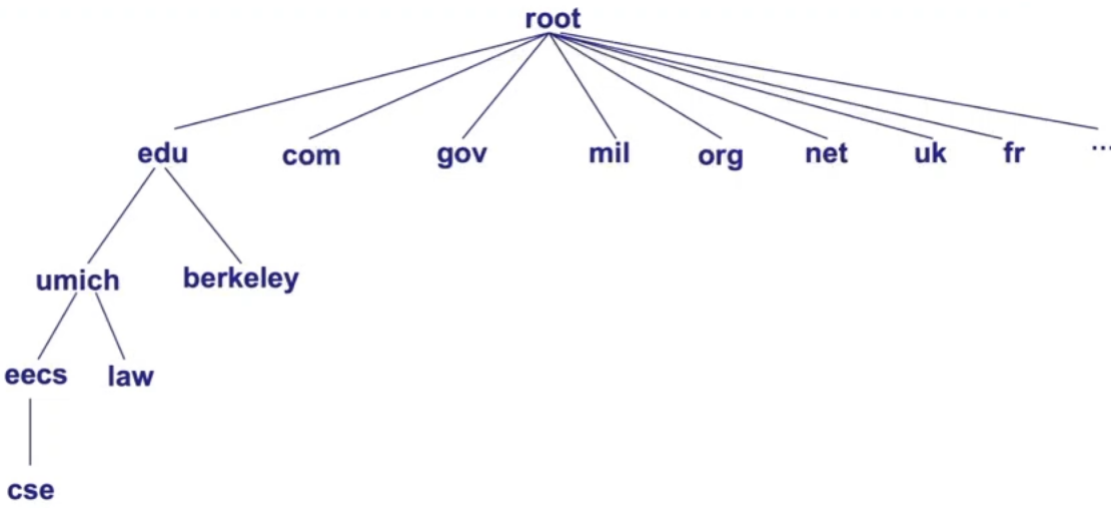
- Uniqueness: no naming conflicts
- Scalable
    - Many names and frequent updates (secondary)
- Distributed, autonomous administration
    - Ability to update its own name
    - Don't have to track other's updates
- Highly available
- Lookups are fast (caching)
- Perfect consistency is a non-goal

How:

- Partition the namespace
- Distribute administration for each partition
- Distribute name resolution for each partition

Hierarchy

- Three intertwined hierarchies
    - Hierarchical namespace
    - Hierarchically administered
    - Hierarchy of servers



- Root doesn't change
    - first-level name doesn't change, but ip could change

- Domains are subtrees
- Name is leaf-to-root path
- Depth of three is limited 128
- Name collision is trivially avoid

ICAN/IANA

- Responsible of managing first 2 level: an administrative authority that is responsible for that portion of the hierarchy

Server hierarchy

- Top of hierarchy: Root servers'
    - Location hardwired into other servers
- Next level: Top-level domain (TLD) servers
    - .com, .edu
    - Managed professionally
- Bottom level: Authoritative DNS servers
    - Actually store the name-to-address mapping
    - Maintained by the corresponding administrative authority
- Each server stores a subset of total DNS database
- An authoritative DNS server stores "resource records" for all DNS names in the domain that it has authority for
- Each server needs to know other servers responsible for other portion of the hierarchy
    - Every server knows the root
    - Root knows all top-level domains

DNS Root

- 13 DNS root servers, each has many replication

DNS records

- DNS servers store resource records (RRs)
    - RR is (name, value, type, TTL)
- Type = A-> Address
    - name = hostname

    - value = IP address
- Type = NS -> Name Server
    - name = domain
    - value = name of DNS server for domain
- Type = CNAME -> Canonical Name
    - name = alias name for some real name
    - value = canonical name
- Type = MX -> Mail eXchanger
    - name = domain in email address
    - value = name(s) of mail server(s)

Inserting Resource Records

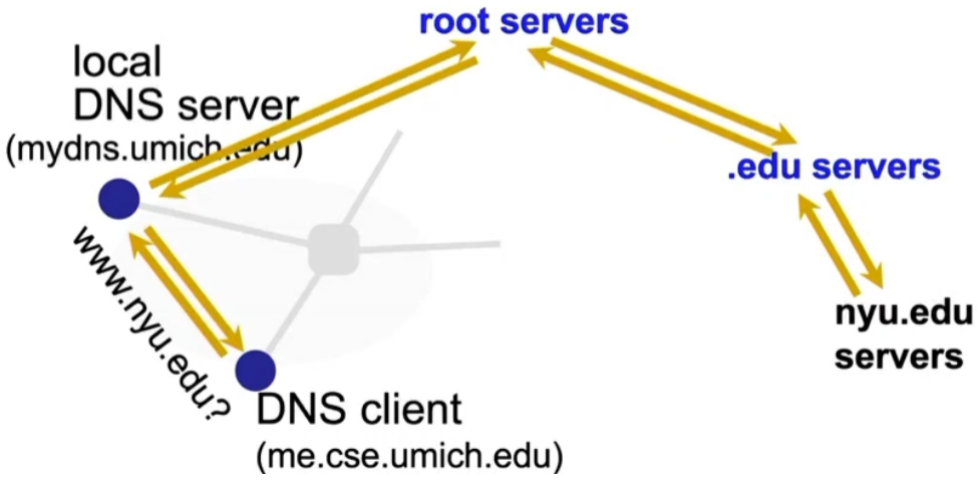
- Register
    - Provide registrar with names and IP address of your authoritative name servers
    - Register insert RR pairs into TLD server
        - »(foobar.com, dns1.foobar.com, NS)
          »(dns1.foobar.com, 212.44.9.129, A)
    - Store resource records in your server
        - e.g., type A record for www.foobar.com
        - e.g., type MX record for foobar.com
    -

Using DNS

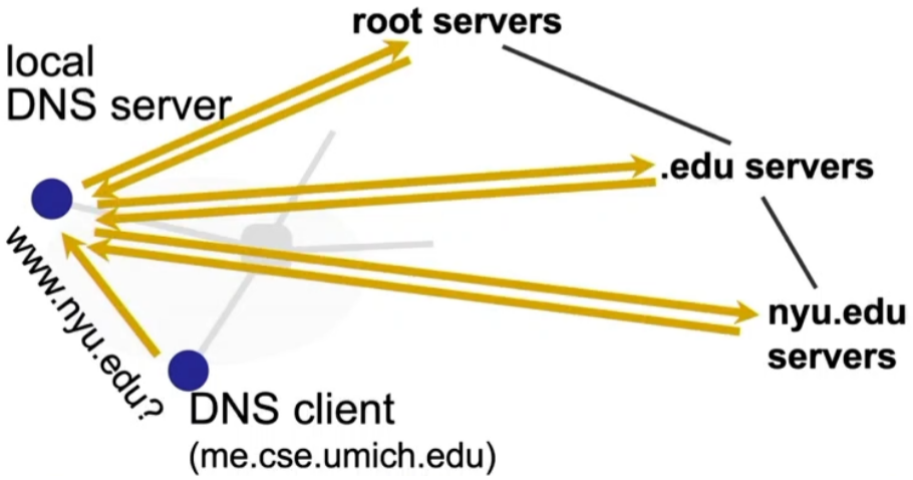
- Two components
    - Local DNS servers
    - Resolver software on hosts
- Local DNS server
    - Client configured with default server's address or learn it via a host configuration protocol
- Client application
    - Obtain DNS name from URL
    - Do getnameinfo() to trigger DNS request to its local DNS server

```
dig nyu.edu

; <<>> DiG 9.10.6 <<>> nyu.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 47443
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;nyu.edu.                       IN      A

;; ANSWER SECTION:
nyu.edu.                60      IN      A       216.165.47.10

;; Query time: 39 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Fri Sep 10 08:21:43 EDT 2021
;; MSG SIZE  rcvd: 52
```

Recursive: do it for you



Iterative: who to ask next



- Usually used since it reduce overhead on behalf of other server

DNS protocol

- Query and Reply message; both with the same message format
  - Header: identifier, flag, etc.
  - Plus resource records
- Client-server interaction on UDP Port 53
  - Spec supports TCP too, but not always implemented
  - DNS package is small and not worth TCP handshake

Reliability

- Replicated DNS servers (primary/secondary)
  - Name service available if at least one replica is up
  - Queries can be load-balanced between replicas
- Usually, UDP used for queries
  - We can implement reliability on UDP
- Try alternate servers on timeout
  - Exponential backoff when retrying same server: help reduce the load since you are not sure if a server is not responding or the network is just slow
- Same identifier for all queries -- don't care which server is responding

DNS Caching (to look up fast)

- Performing all these queries takes time
  - Up to 1-second latency before starting download
- Caching can greatly reduce overhead
  - The top-level server very rarely change
  - Popular sites visited often
  - Local DNS server often has the information cached
- How DNS caching works
  - DNS server cache responses to queries
  - Reponses include a "TTL" field
  - Server deletes cached entry after TTL expires
- Negative caching
  - Remember things that do not work
  - These can take a long time to fail the first name

    - But real fast next time
    - Useful but optional feature
- Administrative delegation and hierarchy enables:
    - Easy unique naming
    - "Fate sharing" for network failures: if bottleneck reach then service down
    - Reasonable trust model --now DNS is not secure
    - Caching increasing scalability and performance

DNS provide indirection

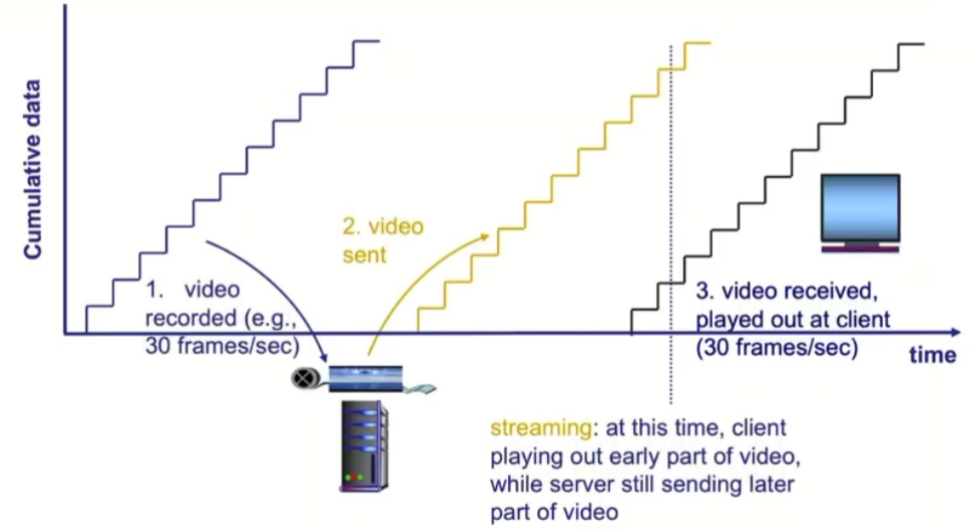
- Addresses can change underneath
- Name could map to multiple IP address -- CDN
- Multiple names for same address -- alias

Video Streaming(stored video)

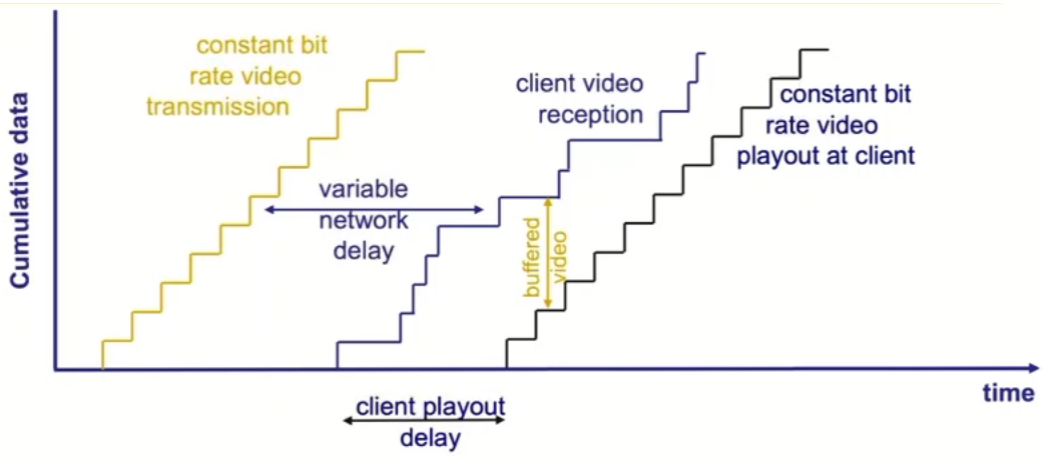
- Dominates the global internet traffic landscape
- The video medium
    - Video is a sequence of image/frames displayed at constant rate (moving pictures)
    - Digital image is an array of pixels, each pixel represented by bits
    - Compression is key
    - Same video can be compressed to multiple quality levels -- **to adopt condition**

HTTP Streaming

- Video is stored at an HTTP server with a URL
- Clients send a GET request for the URL
- Server sends the video file as a stream
- Client first buffers for a while -- minimize interruptions later
- One the buffer reaches a threshold
    - The video plays in the foreground
    - more frame are downloaded in background



- Challenges
  - Absorb network delay variations
  - Handle user interactions
  - Handle packet loss, retransmission



Client-side buffering and playout delay: compensate for network-added delay, delay jitter

- Issues
  - Same bitrate for all clients
    - Clients can have very different network conditions/change over time
  - Cannot dynamically adopt to conditions

DASH: Dynamic Adaptive Streaming over HTTP

- Keep multiple resolutions for the same video
  - Stored in a manifest file in the HTTP server
- Clients asks for the manifest file first to learn about the options

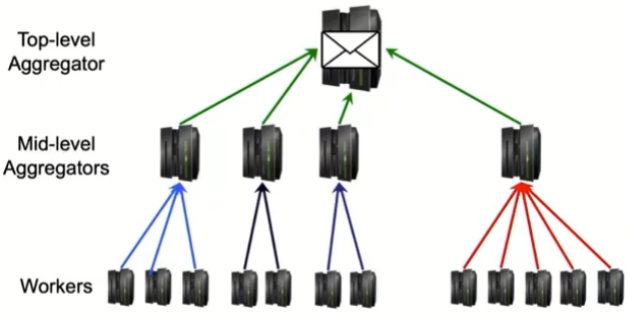
- Asks for chunks at a time and measures available bandwidth while they are downloaded
    - Low bandwidth -> switch to lower bitrate
    - High bandwidth -> switch to higher bitrate

Datacenter applications

- Implications
    - Scale
        - Need scalable designs
        - Low cost designs - use commodity technology
        - High utilization
            - Contrast: avg. utilization on internet link often 30$
        - Tolerate frequent failure - replication
            - Large number of low cost components
        - Automate
    - Service model: clouds/multi-tenancy
        - Performance guarantees
        - Isolation guarantees
        - Portability

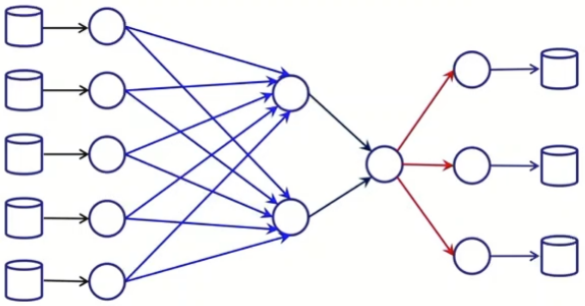
Applications

- Parallelism
- Two common paradigms
    - Partition-aggregate



        - End-to-end response time
            - RTT = O(10) to 100 milliseconds, less than 200 millisecond
            - Depends on the server

- Map-reduce



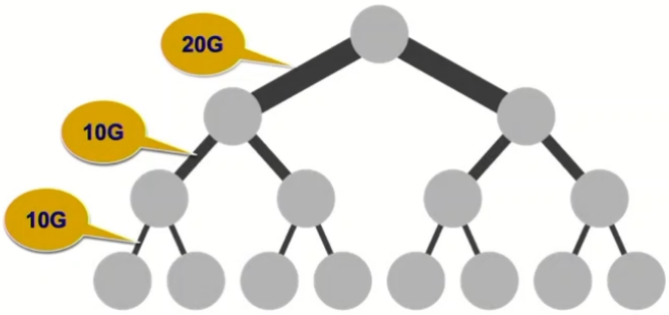
Datacenter networks

- East-West traffic: between servers, communication within "big data" computations, traffic may shift on small timescales
- Datacenter traffic characteristics
  - Most flows are small
  - Most bytes come from large flows
- Applicants want
  - High bandwidth
  - Low latency

High bandwidth

- Ideally, each server can talk to any other server at its full access link rate
- Conceptually: datacenter network as one giants switch
  - Expensive and impractical
- Bisection bandwidth
  - Partition a network into 2 equal parts
  - Minimum bandwidth between partitions is the bisection bandwidth
  - Full bisection bandwidth: bisection bandwidth in an N node network is N/2 times the bandwidth of a single link
    - Nodes of any two halves can communicate at full speed with each other

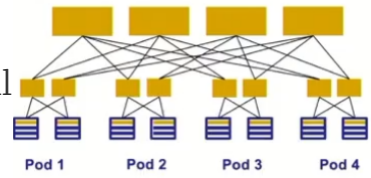
- Achieve full bisection
  - Scale up: make links fatter toward the core of the network
  - Problem: expensive
  - Solution: over-subscribe/better topologies



Oversubscription

- Not enough bandwidth
  - Oversubscription: less bandwidth in the ToR-Agg links than all the servers bandwidth in the rack
  - Oversubscription ratio: Ratio between bandwidth underneath and bandwidth above
- Not enough paths between server pairs
  - Load balancing issues
  - Failure recovery issues

Better topologies

- Add number of switch if one fail 

- Clos topology
  - Multi-stage network'
  - k pods, each pod has 2 layers of k/2 switches
  - All links have same bandwidth
  - At most $k^3/4$ machines
- Challenges in scale-out designs
  - Topology offer high bisection bandwidth
  - All other system components must be able to exploit this available capacity

        - Routing must use all path
        - Transport protocol must fill all pipes.

Transport Layer Basics

Why transport layer

- IP capture hosts, but end-to-end communication happens between applications
    - Need a way to decide which packet go to which application
- IP provides a weak service model (best-effort)
    - Packets can be corrupted, delayed, dropped, reordered, duplicated
    - No guidance on how much traffic to send and when
    - Dealing with this is tedious for application developers

Multiplexing & demultiplexing

Mux

- Gather and combining data chunks at the source from different applications and delivering to the network layer
- On the sender side

Demux

- Delivering correct data to corresponding sockets from a multiplexed stream
- On the receiver side

Role of the transport layer

- Communication between process
- Provide common end-to-end services for app layer (optional)
    - Reliable delivery
    - Well-paced data delivery
        - Too fast overwhelm the network
- TCP and UDP are common transport protocols
    - Also SCTP, MPTCP, SST, RDP, DDCP, ...
- UDP is a minimalist transport protocol
    - Only provides mux/demux capabilities

- TCP offers a reliable, in-order, byte stream abstraction
    - With congestion control, but no performance guarantees.

QUIC transport protocol

- Built on top of UDP
- QUIC packets are encrypted individually
- Faster connection setup by reusing the negotiated parameters from a previous connection
- Many other benefits: extensibility, reduced sensitivity to packet loss

Applications and sockets

Socket

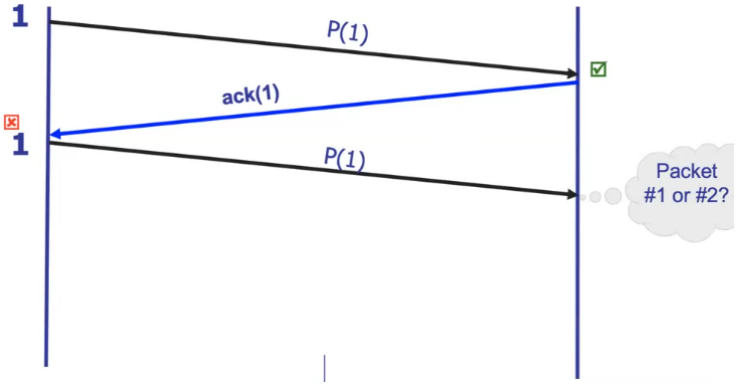
- Software abstraction for an application process to exchange network messages with OS
- UDP: SOCK_DGRAM
- TCP: SOCK_STREAM

Ports

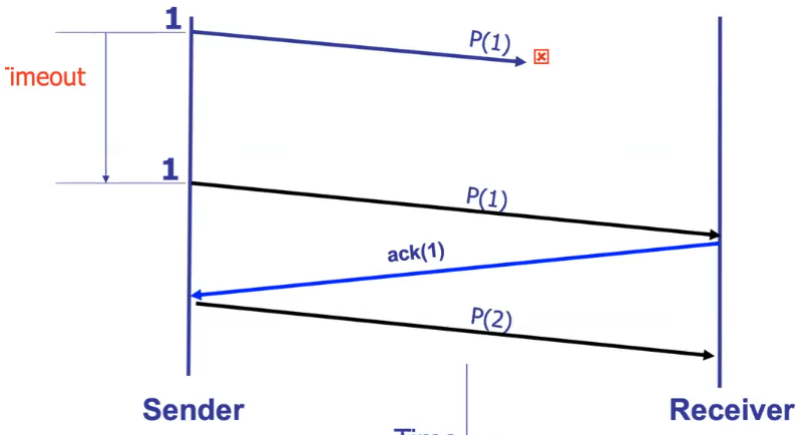
- 16-bit numbers that help distinguish apps
    - Packets carry src/dst port no in transport header
    - Well-known(1-1023) and ephemeral ports
- OS stores mapping between sockets and ports
    - Port in packets and sockets in OS
    - For UDP ports
        - OS stores (local port, local IP address)
    - For TCP ports
        - OS stores (local port, local IP, remote port, remote IP)
        - Because TCP is connection oriented, we need to store the identifier of connection

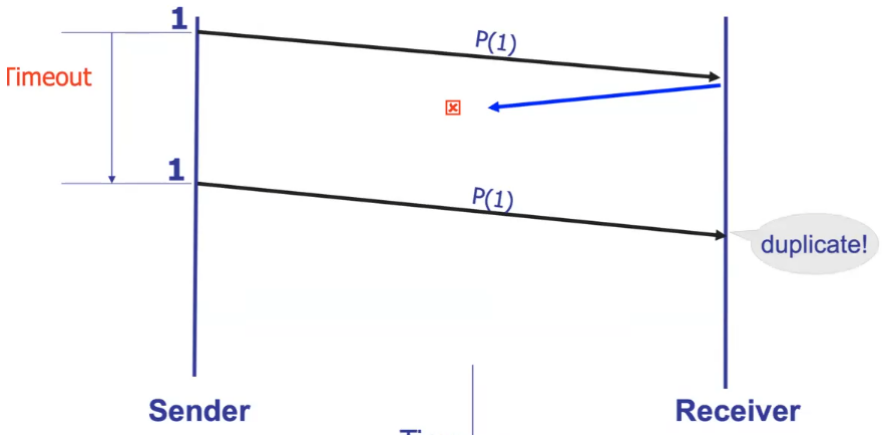
Reliable Transport

Packet corruption: what if ACK/NACK corrupted



Packet loss



- Timer-driven loss detection
  - Set timer when packet is send; retransmit on timeout
- If ACK is loss
  - 

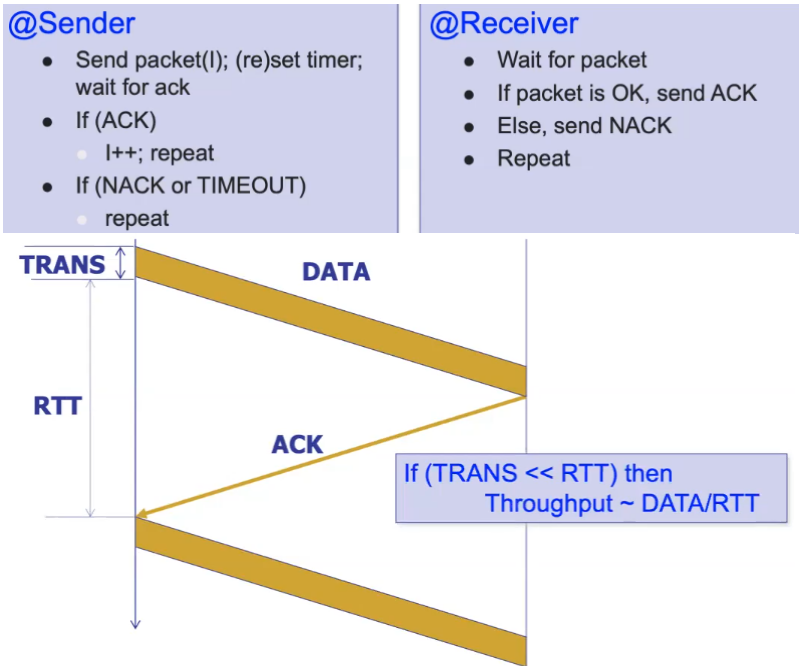
  - The receiver can know by sequence number, and remove this packet, and send back ACK (1)

Dealing with delay (could lead by timer-driven loss detection)

- Solve same as packet loss

Designing Solutions

"Stop and Wait"

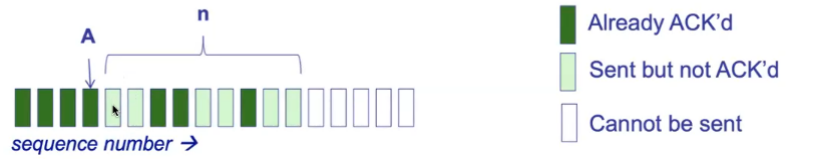


- Correct reliable, but extremely inefficient

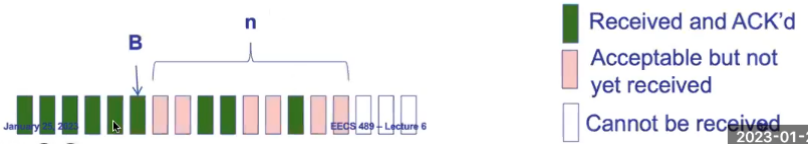
"Sliding Window" -- packets in flight

- **Left edge of window**
  - **Sender: beginning of unacknowledged data**
  - **Receiver: beginning of expected data**
- Window = set of adjacent sequence numbers
- Send up to n packets at a time
  - Sender can send packets in its window
  - Receiver can accept packets in its window
  - Window of acceptable packets "slides" on successful reception/acknowledgement
  - Window contains all packets that might still be in transit

- Let A be the last ack'd packet of sender without gap; then window of sender = {A+1, A+2, …, A+n}

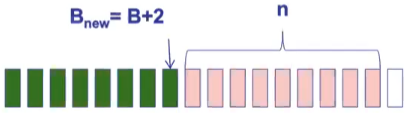


- Let B be the last received packet without gap by receiver, then window of receiver = {B+1,…, B+n}



  - Throughput of sliding window
    - $min(n * Data/RTT, LinkBandwidth)$
    - if n is too large you are sending too much to the internet and they can get dropped
  - ACK for sliding window
    - Cumulative ack
      - At receiver

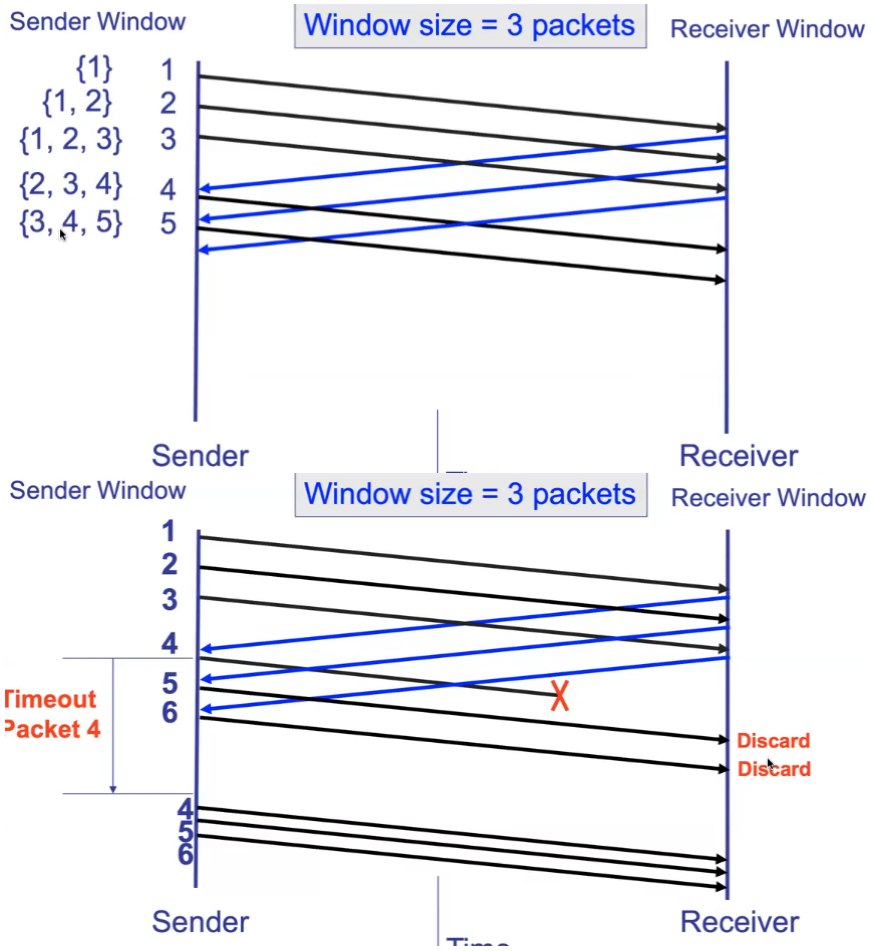
      - After receiving B+1, B+2



        - Receiver sends ACK(B + 3) = Bnew+1
    - Selective ACK: individually ack correctly received packets
      - Require bookkeeping

Protocols

- Resending packets
  - **Go-Back-N**
    - Sender transmit up to n unacknowledged packets
    - Receiver only accept packets in order
      - Discard out of order packets
    - Receiver use cumulative ack

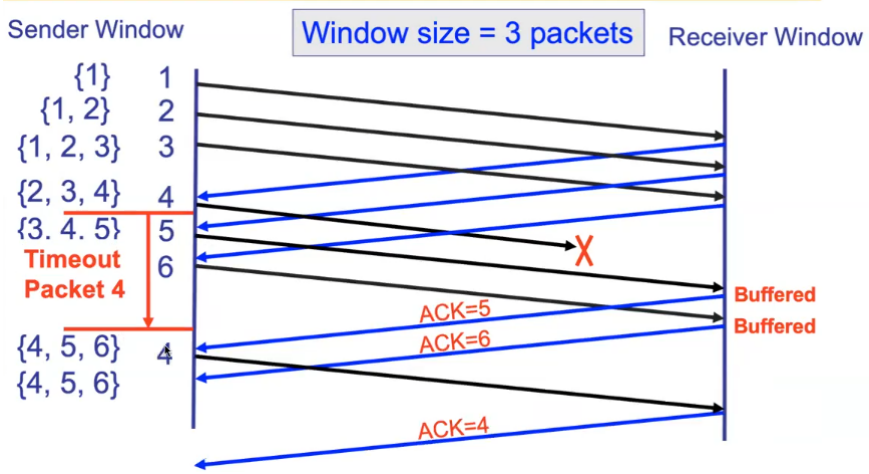
- sender set timer for 1st outstanding ack (A + 1)
    - If timeout, retransmit A+ 1..A+n
- 

-

- **Selective Repeat**
    - Sender: transmit up to unacknowledged packets

Assume packet k is lost, k+1 is not
- Receiver: indicates packet k+1 correctly received
- Sender: retransmit only packet k on timeout

Efficient in retransmissions but complex book-keeping
- Need a timer per packet



- When GBD better
  - error rate is low, or waste resource
- SR better
  - error rate is high, or complex

Components of a solution

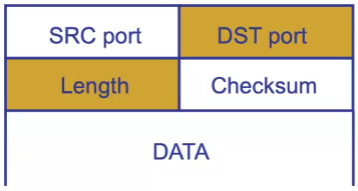
- Checksums (for error detection)
- Timers (for loss detection)
- Acknowledgments
  - Cumulative
  - Selective
- Sequence numbers (duplicates, windows)
- Sliding windows (for efficiency)
- Reliability protocols use the above to decide when and what to retransmit or acknowledge

Summary

- Transport layer allows applications to communicate with each other
- Provides unreliable and reliable mechanisms
- Possible to build reliable transport over unreliable medium

UDP: User Datagram Protocol

- Lightweight communication between processes
  - Avoid overhead and delays of order & reliability
- UDP described in RFC
  - Dest IP address and port to support demultiplexing

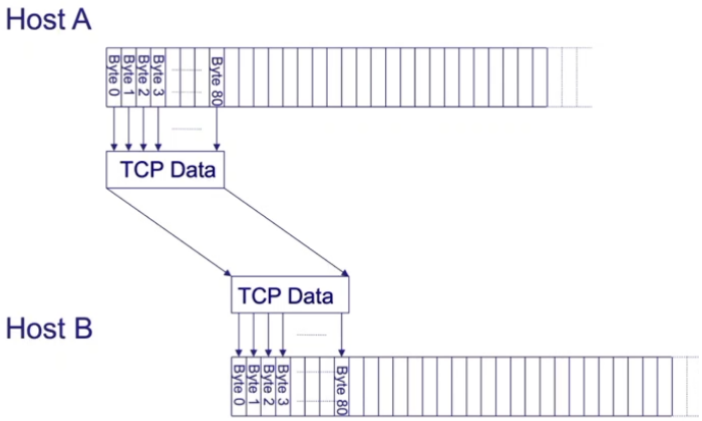


  - Optional error checking -- checksum = 0 means do not verify
  - Source port also optional, but useful to respond back to sender in some cases

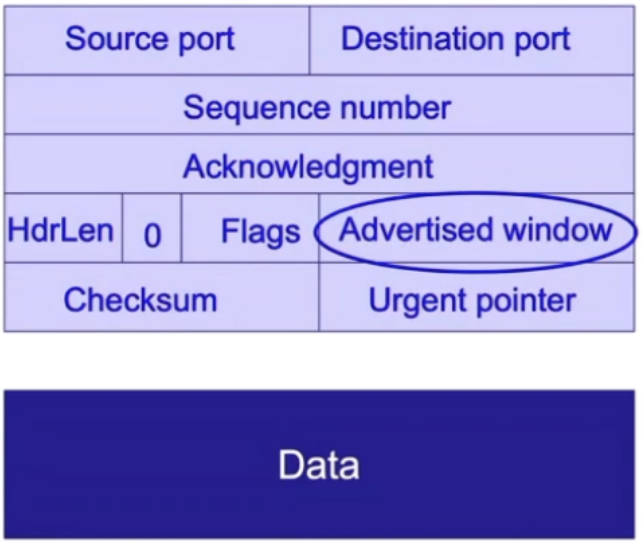
TCP

- delivers a reliable, in-order, byte stream
- Reliable: TCP resend lost packets
  - Until it give up and shut down connection
- Byte stream: assume there is an incoming stream of data

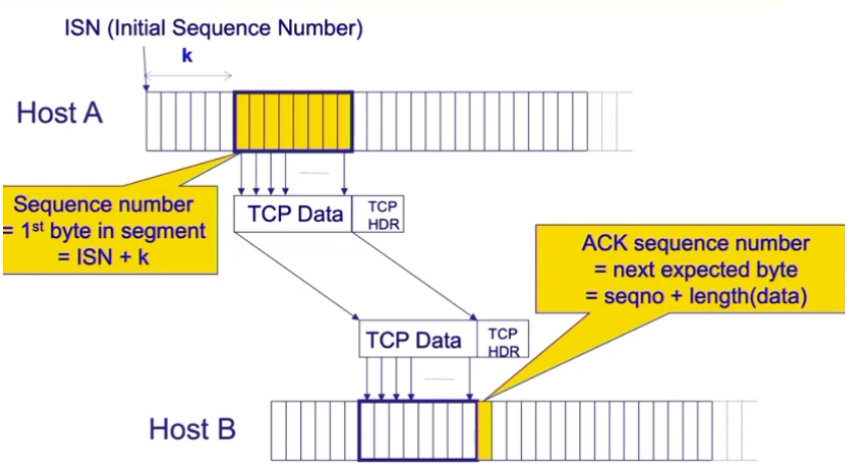
- Most of what we've seen
  - Checksums
  - Sequence numbers are byte offsets
  - Sender and receiver maintain a sliding window
  - Receiver sends cumulative acknowledgements (like GBN)
    - »Sender maintains a single retransmission timer
  - Receivers buffer out-of-sequence packets (like SR)
- Few more: fast retransmit, timeout estimation algorithms etc.



TCP header



- Source/Dest port: for multiplex and demux
- Checksum: computed over pseudo-header and data, optional, if 0 don't check
- Sequence number: **byte offsets** since TCP is byte stream
  - 1st byte in segment = ISN(initial sequence number) + k

- 

  - Sender: seqno=X, length=B
  - Receiver: ACK=X+B
  - Sender: seqno=X+B, length=B
  - Receiver: ACK=X+2B
  - Sender: seqno=X+2B, length=B

  - Seqno of next packet is same as last ACK field

- ACK is for the data received - bidirectional communication
- Header Length: number of 4-byte words; 5: No options
- Flag: SYN, ACK, FIN, RST, RSH, URG

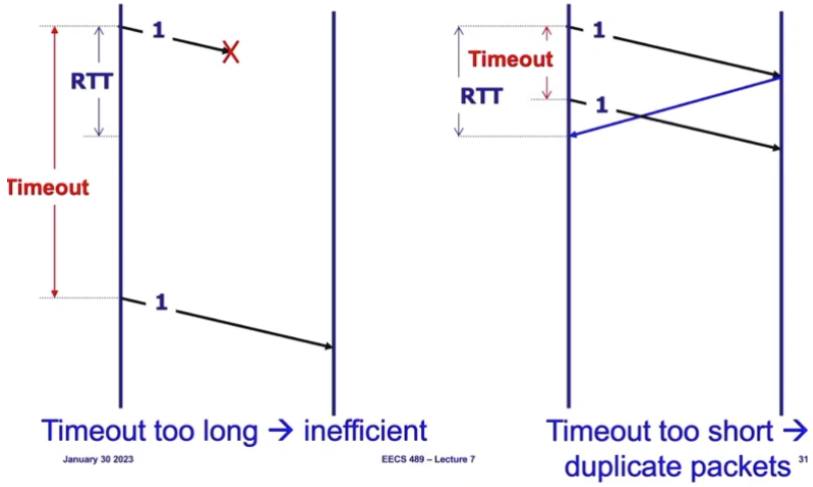
Fast retransmit: duplicate ACKs trigger early retransmission

- Duplicate ACKs are a sign of an isolated loss
- Trigger retransmission upon receiving k loss
  - Default by 3

Loss with cumulative ACKs

- Send missing packet and move sliding window by the number of dup ACKs
  - Speed up but can cause error
- **Send missing packet and wait for ACK**
  - Slow but stable
  - **TCP take this**

Retransmission timeout

- If the sender hasn't received an ACK by timeout, retransmit the first packet in the window.
- 

- Solution: make timeout proportional to RTT -- but how we measure RTT

RTT estimation

- $$\text{EstimatedRTT} = (1-\alpha)*\text{EstimatedRTT} + \alpha*\text{SampleRTT}$$
- Problem: we cannot distinguish between real ACK and ACK of the retransmitted packet

Karn/Partidge algorithm

- Don't use SampleRTT from retransmissions
  - Once retransmitted, ignore that segment in the future
- Computes EstimatedRTT using α = 0.125
- Timeout value (RTO) = 2 × EstimatedRTT
  - Employs exponential backoff
    - »Every time RTO timer expires, set RTO ← 2·RTO
      - (Up to maximum ≥ 60 sec)
    - »Every time new measurement comes in (= successful original transmission), collapse RTO back to 2 × EstimatedRTT

Jacobson/Karuis algorithm

- Problem: need to better capture variability in RTT
  - Directly measure deviation

- Deviation = | SampleRTT – EstimatedRTT |
- DevRTT: exponential average of Deviation

- RTO = EstimatedRTT + 4 x DevRTT

TCP segments

- if too small: large overhead
- too large: penalty for loss
- send if segment full
- or time out 

- IP packet
  - No bigger than **Maximum Transmission Unit** (MTU)
- TCP packet
  - IP packet with a TCP header and data inside
  - TCP header >= 20 bytes
- TCP segment
  - No more than **Maximum Segment Size** (MSS) bytes
  - MSS = MTU - (IP header) - (TCP header)

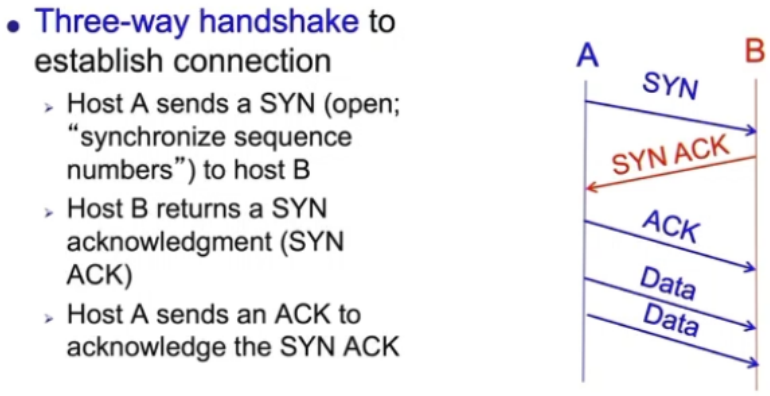
TCP connection establishment

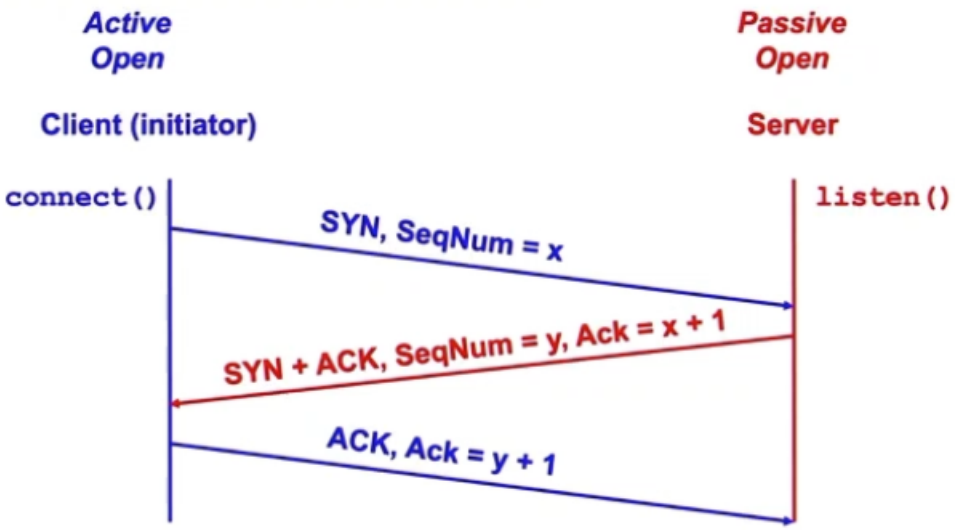
Initial Sequence Number (ISN)

- Hard to guess
- Sequence number for very first byte
- Host exchange ISNs when establishing connection

Establishing Connection

- Three-way handshake to establish connection



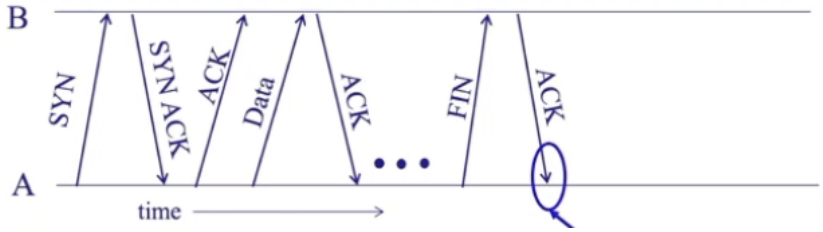


- What if SYN lost

- Suppose the SYN packet gets lost
  - Packet dropped by the network or server is busy
- Eventually, no SYN-ACK arrives
  - Sender retransmits the SYN on timeout
- How should the TCP sender set the timer?
  - Sender has no idea how far away the receiver is
  - Hard to guess a reasonable length of time to wait
  - SHOULD (RFCs 1122 & 2988) use default of 3 seconds
    - »Some implementations instead use 6 seconds
- User clicks on a hypertext link
  - Browser creates a socket and does a "connect"
  - The "connect" triggers the OS to transmit a SYN
- If the SYN is lost…
  - 3-6 seconds of delay: can be very long
  - User may become impatient and can retry
- User triggers an "abort" of the "connect"
  - Browser creates a new socket and another "connect"
  - Can be effective in some cases

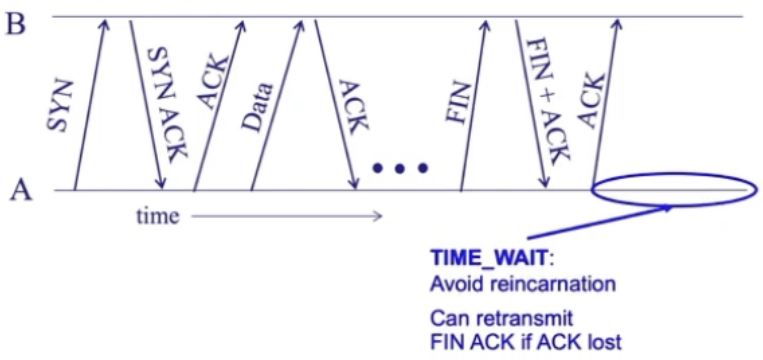
Tear down

- Normal termination, one side at a time
  - FIN to close and receive remaining bytes



    - Half close
  - Other host ACK to confirm
  - Close A's side but not B's
    - Until B sends a FIN
    - A acks
  - After exchanging FIN & ACK: TIME_WAIT
    - **Avoid reincarnation**
    - B will retransmit FIN if ACK lost

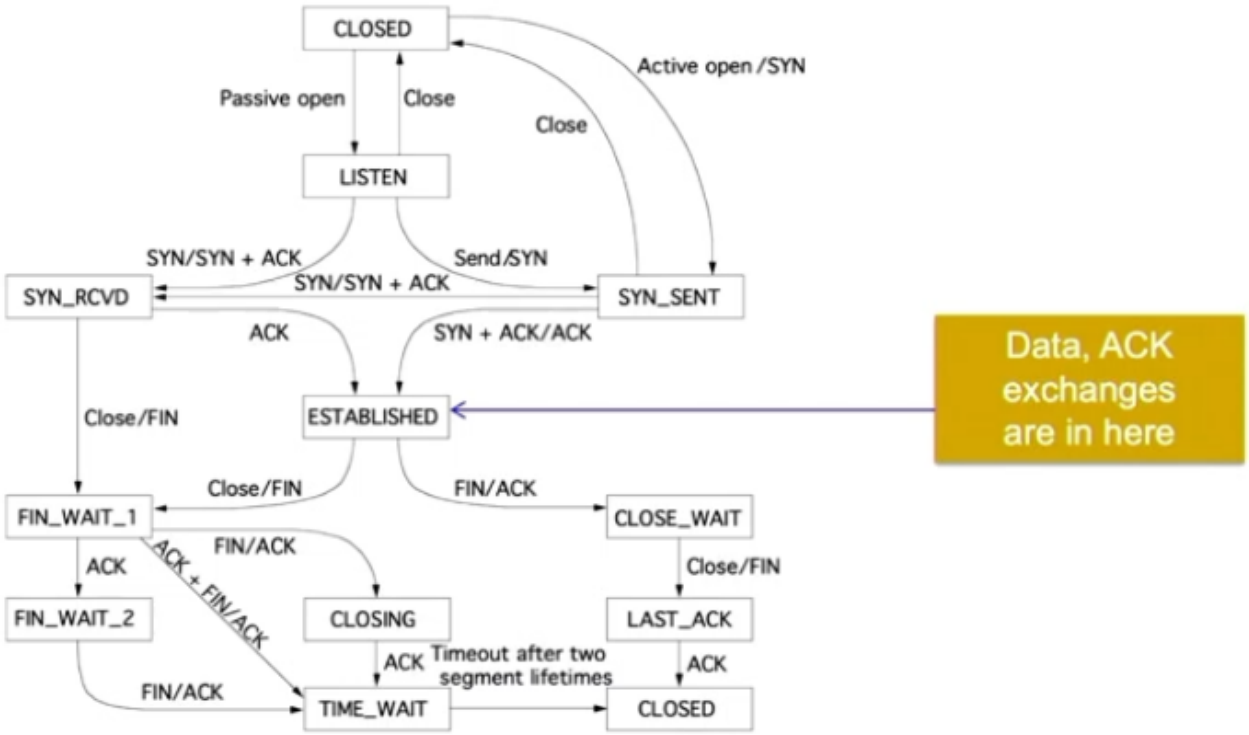
    - Closed
- Normal termination, both together
    -



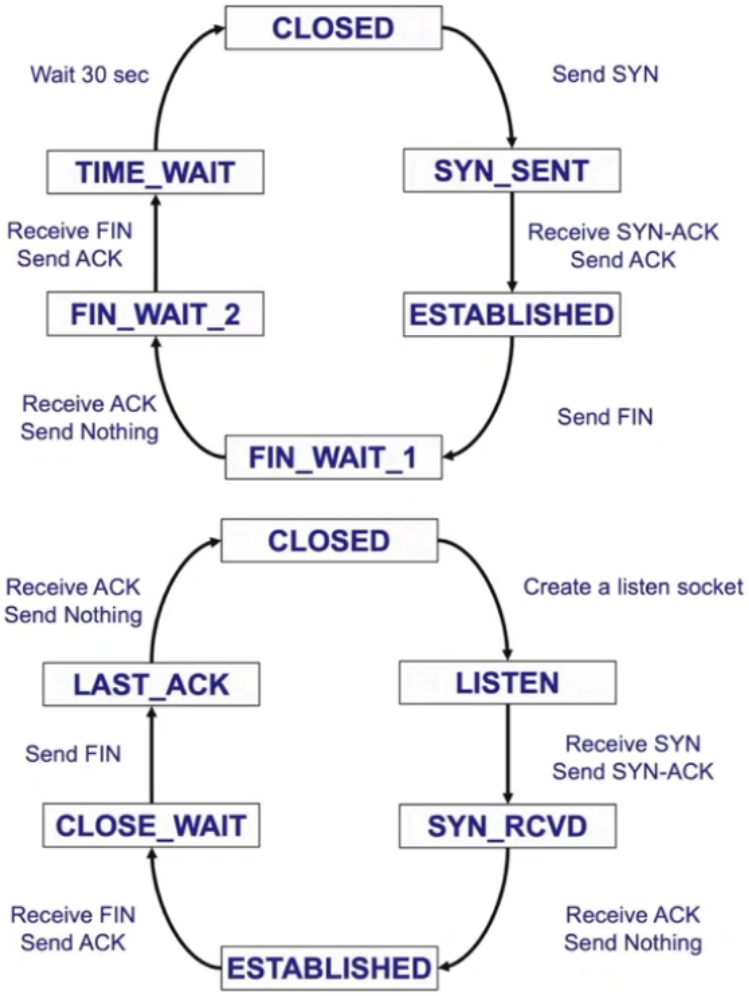
        - Same as before, but B sets FIN with their ack of A's FIN

- Abrupt termination
    - A send RESET(RST) to B
    - B does not ack the RST
        - If RST lost anything in flight will be lost

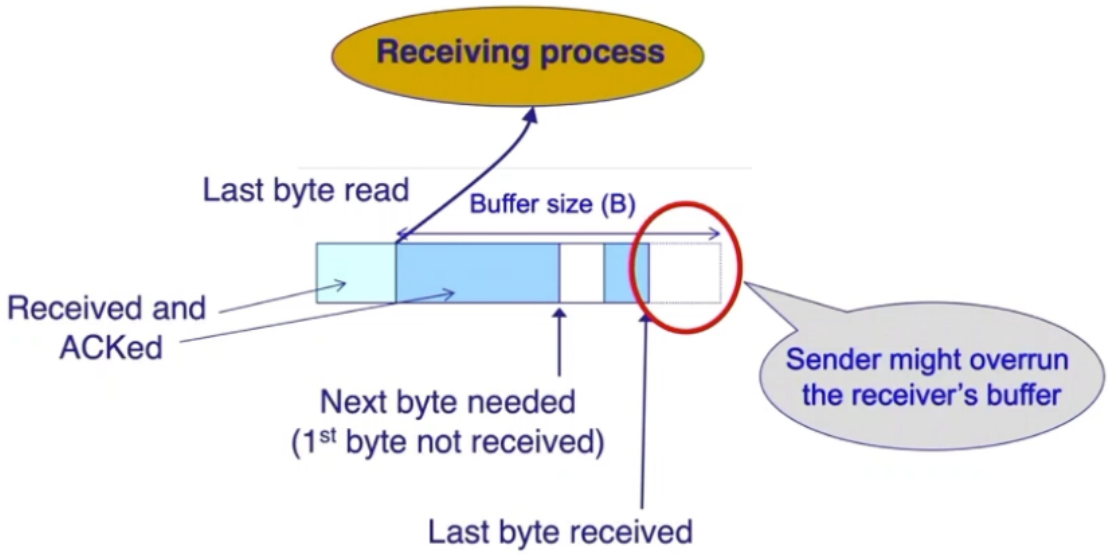
TCP flow control



Client/Server lifecycle



Why flow control



Solution: Advertise window(flow control)

- Receiver uses an "**Advertised Window"(RWND)** to prevent sender from overflowing its window
  - Receiver indicates value of RWND in ACKs
  - Sender ensures that the total number of bytes in flight <= RWND
- RWND = Buffer_size - (last_byte_received - last_byte_read)

Sliding window with flow control

- Sender: window advances when new data ACK'd
- Receiver: window advances as receiving process consumes data
- Receiver advertises to the sender where the receiver window currently ends ("righthand edge")
  - Sender agrees not to exceed this amount
- UDP does not have flow control
  - Data can be lost due to buffer overflow
- Sender can send no faster than RWND/RTT bytes/sec
- Receiver only advertises more space when it has consumed old arriving data
- What happens when RWND=0?
  - Sender keeps probing with one data bytes

- In original TCP design, that was the sole protocol mechanism controlling sender's rate
  - What's missing?

- If RWND = 0: sender try to see if the connection still alive
- Missing: sender side not ensured ↓

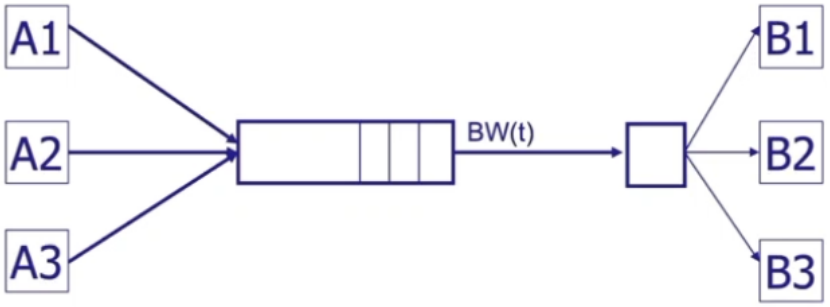
TCP congestion control

- If two packets arrive at a router at the same time
    - Router will transmit one and buffer/drop another
- Internet traffic is burst
- Root cause: statistical multiplexing -- share resource
- "Congestion collapse" sending rate only limited by flow control so that keep sending, network is overwhelmed
    - Fix: extend TCP's existing window-based protocol but adapt the window size in response to congestion

Design consideration

- How do we know the network is congested
    - Implicit or explicit signal from router
- Who takes care of congestion
    - End host -- may receive some help from the network
- How do we handle congestion
    - Continuous adaption
- Three issues of consider
    - Discovering the bottleneck next bandwidth
    - Adjusting to variations in bandwidth
    - Sharing

- Two Issues:
  - Adjust total sending rate to match bandwidth
  - Allocation of bandwidth between flows

- 

  (0) Send without care
  (1) Reservations
  (2) Pricing
  (3) Dynamic Adjustment
  - Hosts infer level of congestion; adjust
  - Network reports congestion level to hosts; hosts adjust
  - Combinations of the above
  - Simple to implement but suboptimal, messy dynamics
- Generality of dynamic adjustment has proven to be very powerful

Summarize

- Each TCP connection has a window
  - Controls number of packets in flight
- Sending rate
  - Window/RTT
- Very window size to control sending rate

Windows to remember

- Congestion Window: CWND
  - Computed by sender using congestion control algo
  - Bytes that can be sent without overflowing routers

- Flow control window: RWND
    - Bytes that can be sent without overflowing receiver
    - Determined by the receiver and report to the sender
- Sender-side window: min{CWND, RWND}, CWND in units of MSS(amount of payload data in a TCP packet)

How the sender adjust its sending rate

- Finding available bottleneck bandwidth
- Adjusting to bandwidth variations
- Sharing bandwidth

Detecting congestion

- Packet delays
    - Difficult for noisy signals
- Routers tell end hosts when they're congested
- Packet loss
    - Could be due to corruption

Not all loss are same

- Duplicate ACKs: isolated loss
    - Still getting ACKs
- Timeout: much more serious
    - Not enough dupacks
    - Must have suffer several losses
- Will adjust rate differently for each case

Rate adjustment

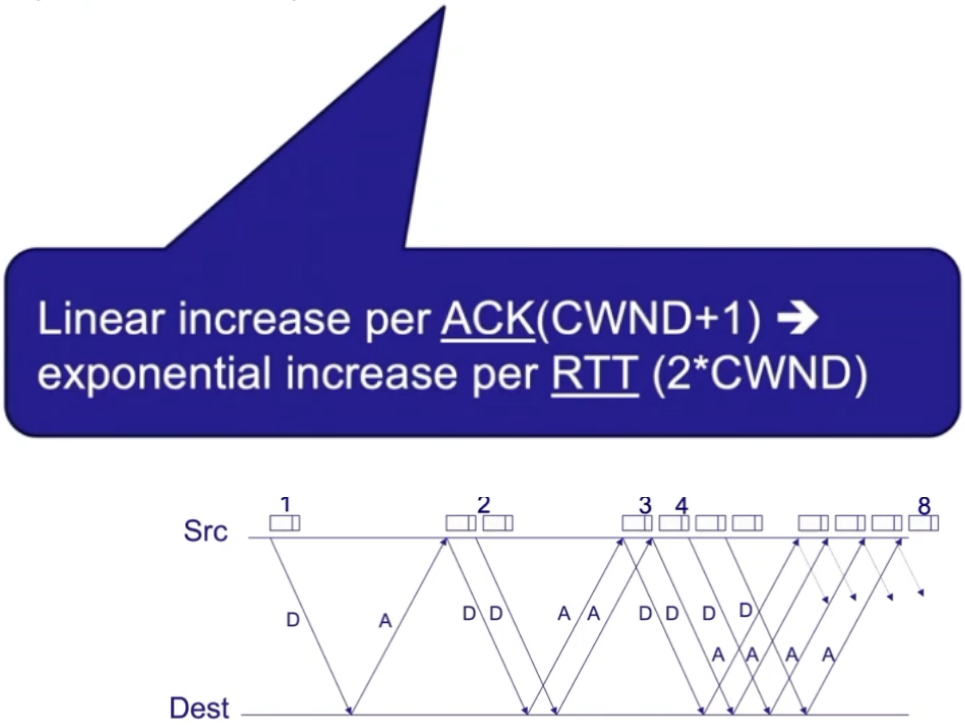
- Upon receipt of ACK or new data: increase rate
- Upon detection of loss: decrease rate

Bandwidth discovery with "Slow Start"

- Goal: estimate available bandwidth
    - Start slow
    - Ramp up quickly

- Sender starts at a slow rate, but increases exponentially until first loss
- Start with a small congestion window
  - Initially, CWND = 1
  - So, initial sending rate is MSS/RTT
- Double the CWND for each RTT with no loss

- For each RTT: double CWND
  - i.e., for each ACK, CWND += 1

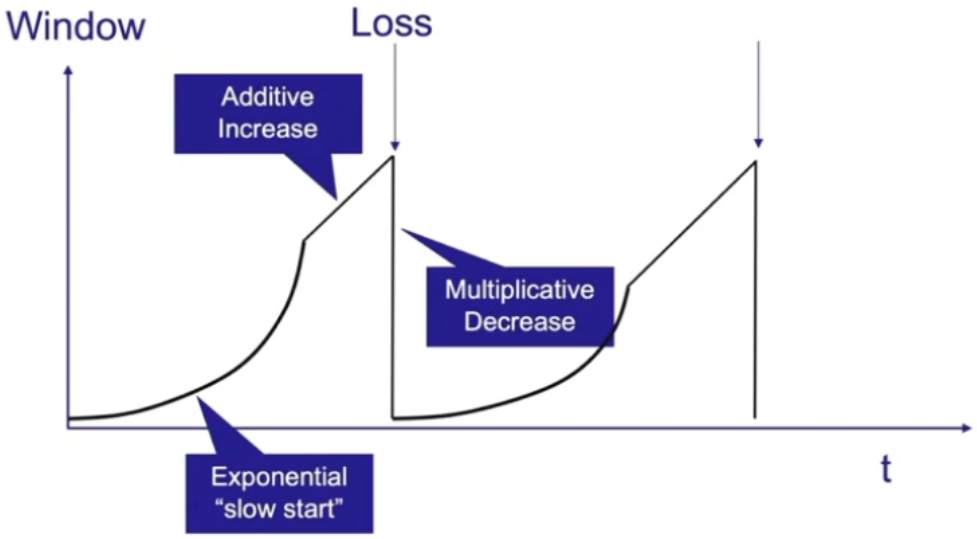


- Stop if CWND > ssthresh(inited to a large value)
  - stop rapid growth and focus on maintenance
  - repeat probing and backoff
- **TCP use "Additive Increase Multiplicative Decrease" (AIMD)**

AIMD (best way)

- Additive increase
  - For each ACK, CWND = CWND + 1/CWND
- Multiplicative decrease
  - On packet loss, divide ssthresh in hash and slow start
    - ssthreash = CWND/2

- CWND = 1
- limited slow start



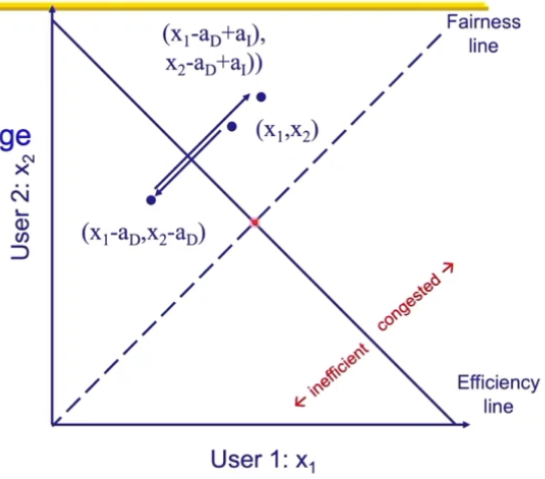
- Recall the three issues
  - Finding available bottleneck bandwidth
  - Adjusting to bandwidth variations
  - Sharing bandwidth

- Two goals for bandwidth sharing
  - Efficiency: High utilization of link bandwidth
  - Fairness: Each flow gets equal share

- AIAD: gentle increase, gentle decrease
- MIAD: drastic increase, gentle decrease
- MIMD: drastic increase and decrease

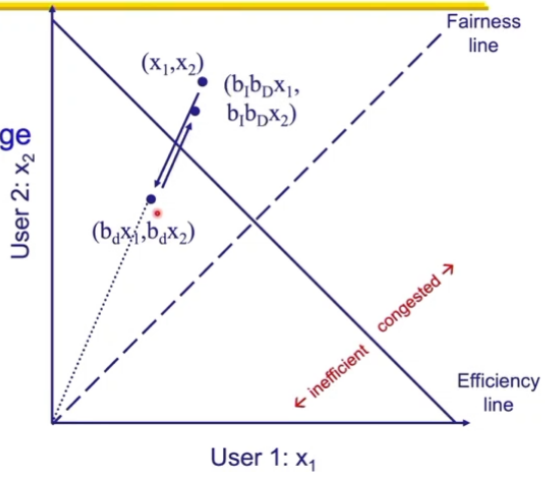
AIAD

- Increase: $x + a_I$
- Decrease: $x - a_D$
- Does not converge to fairness



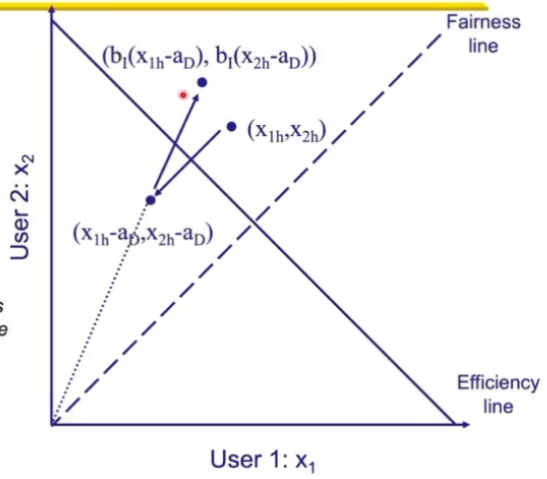
MIMD

- Increase: $x*b_I$
- Decrease: $x*b_D$
- Does not converge to fairness

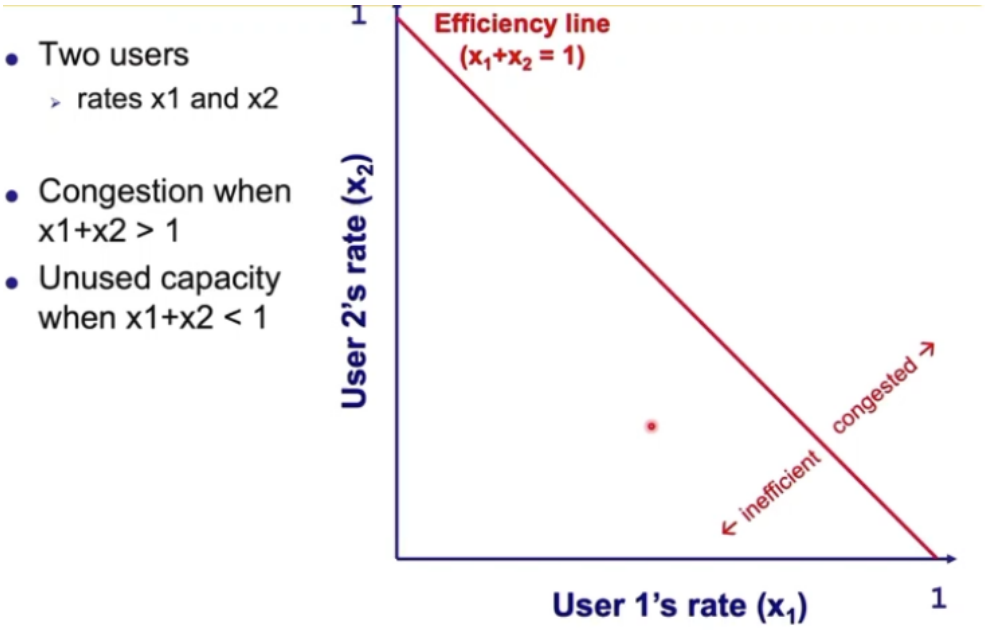


MIAD

- Increase: $x*b_I$
- Decrease: $x - a_D$
- Does not converge to fairness
- Does not converge to efficiency
- *"Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks"*
  -- Chiu and Jain



Model of congestion control



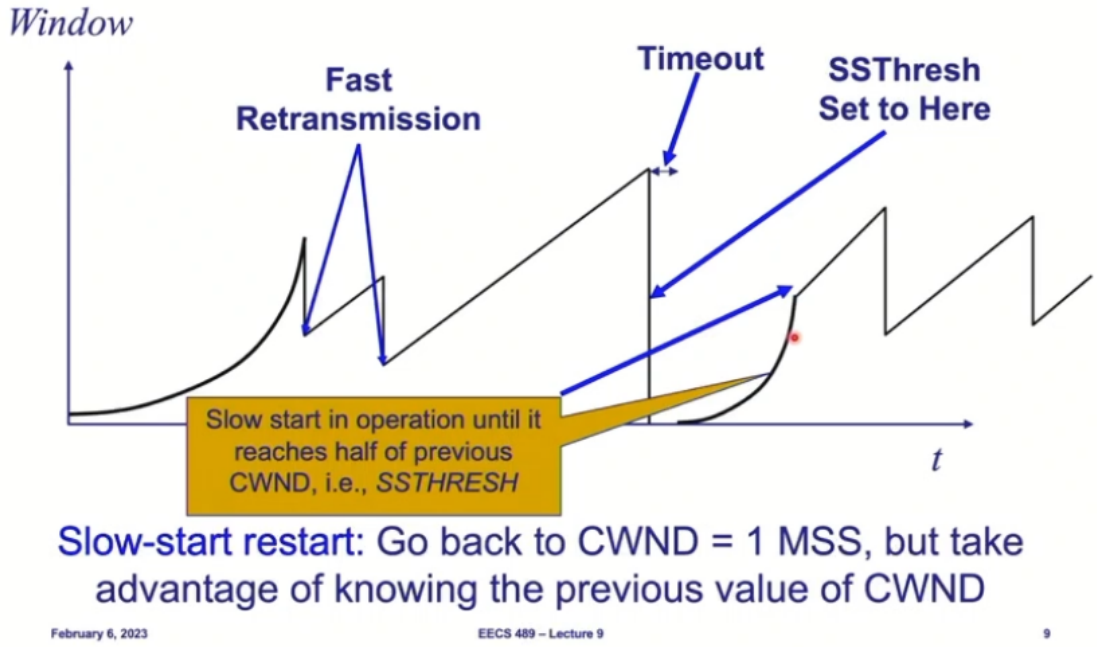
Congestion Control implementation

- States at sender
  - CWND: init to small constant
  - ssthresh: init to large contant
  - dupACKcount and timer
- Events
  - ACK - new data
    - If CWND < ssthresh (slow start)
      - CWND += 1

        CWND packets per RTT
        Hence, after one RTT with no drops:
        CWND = 2xCWND
    - Else (Congestion avoidance)
      - CWND = CWND + 1/CWND

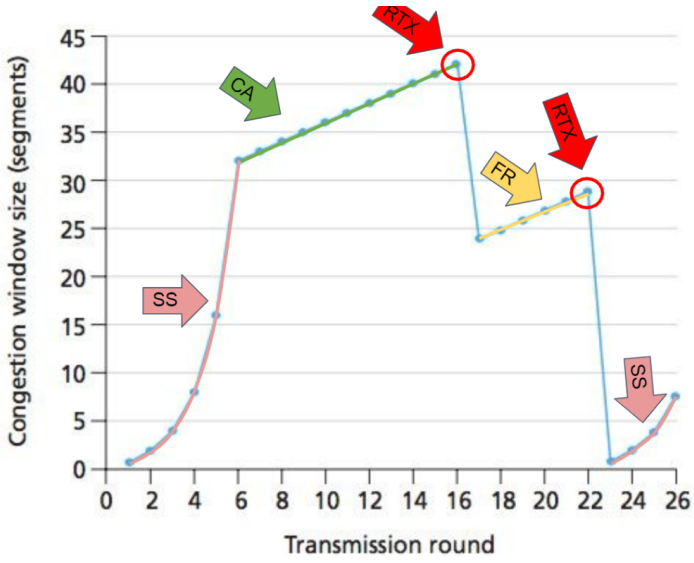
        CWND packets per RTT
        Hence, after one RTT with no drops:
        CWND = CWND + 1
  - dupACK - 3 for fast retransmitting

        - dupACK++
        - if count = 3
            - ssthersh = CWND / 2
            - CWND = CWND / 2
    - Timeout
        - ssthresh <- CWND / 2
        - CWND <- 1



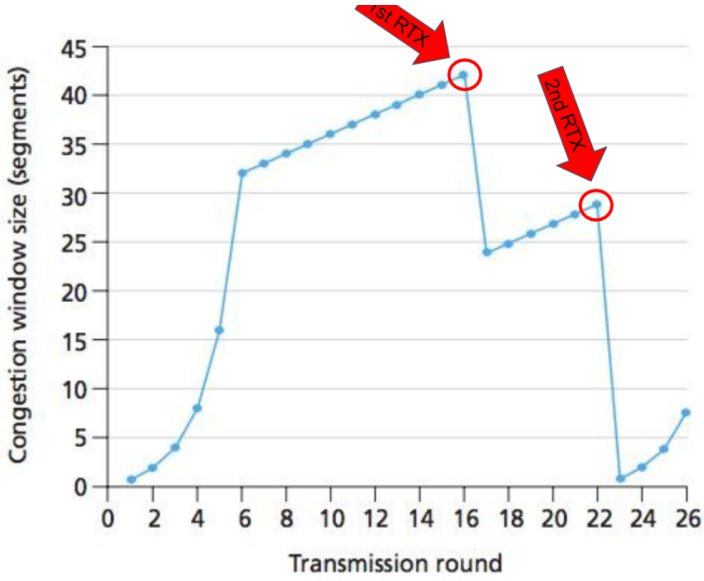
Identify:

- Slow Start (SS)
- Congestion Avoidance (CA)
- Fast Recovery (FR)
- Retransmission (RTX)



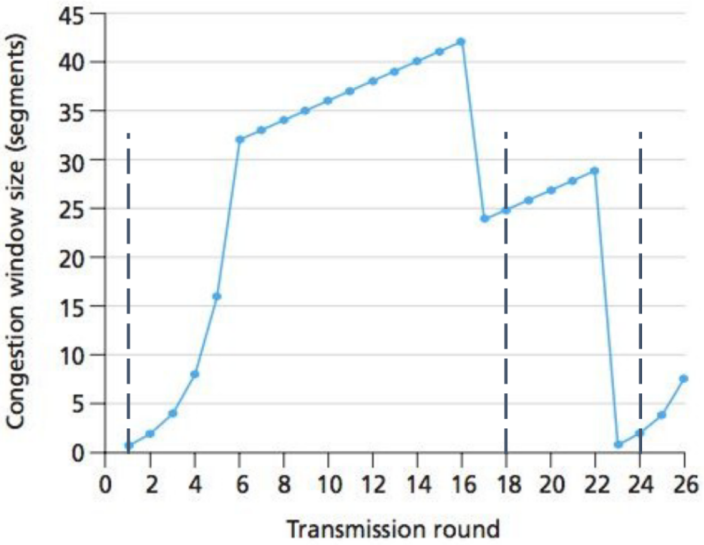
What triggers the first retransmission? How about the second?

First: Duplicate ACK
Second:Timeout

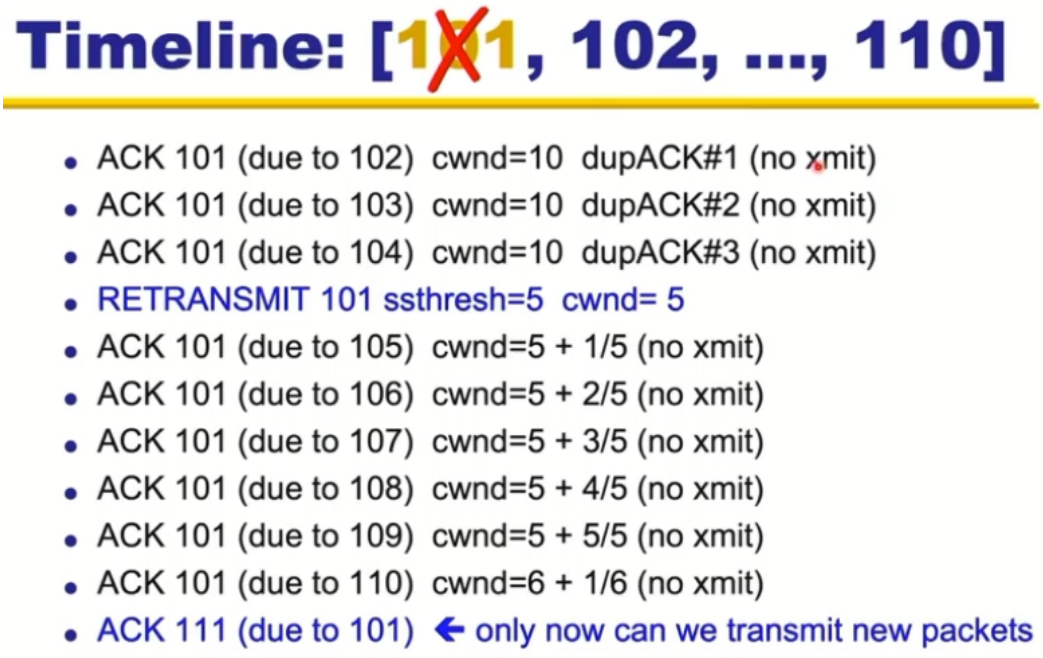


What is the ssthresh at the 1st round, 18th round, 24th round?

1st: 32
18th: 42 / 2 = 21
24th: 29 / 2 = 14

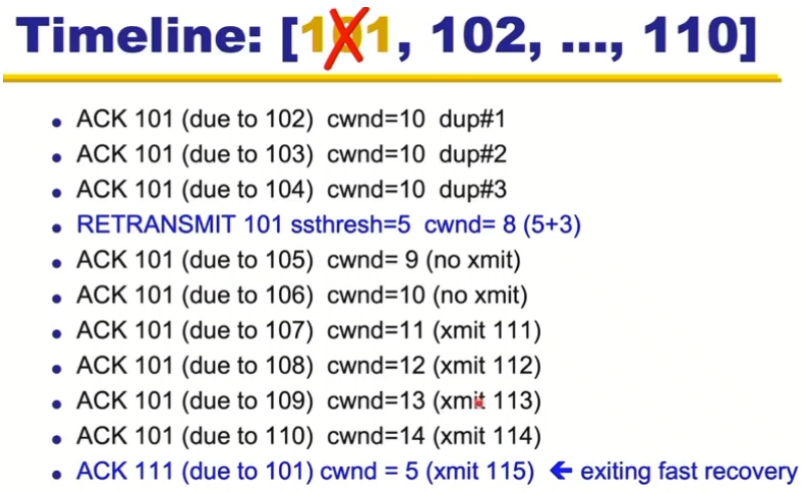


- Problem: too slow incrementing the window



- Plus no packets in flight so ACK "clocking" stalls for another RTT

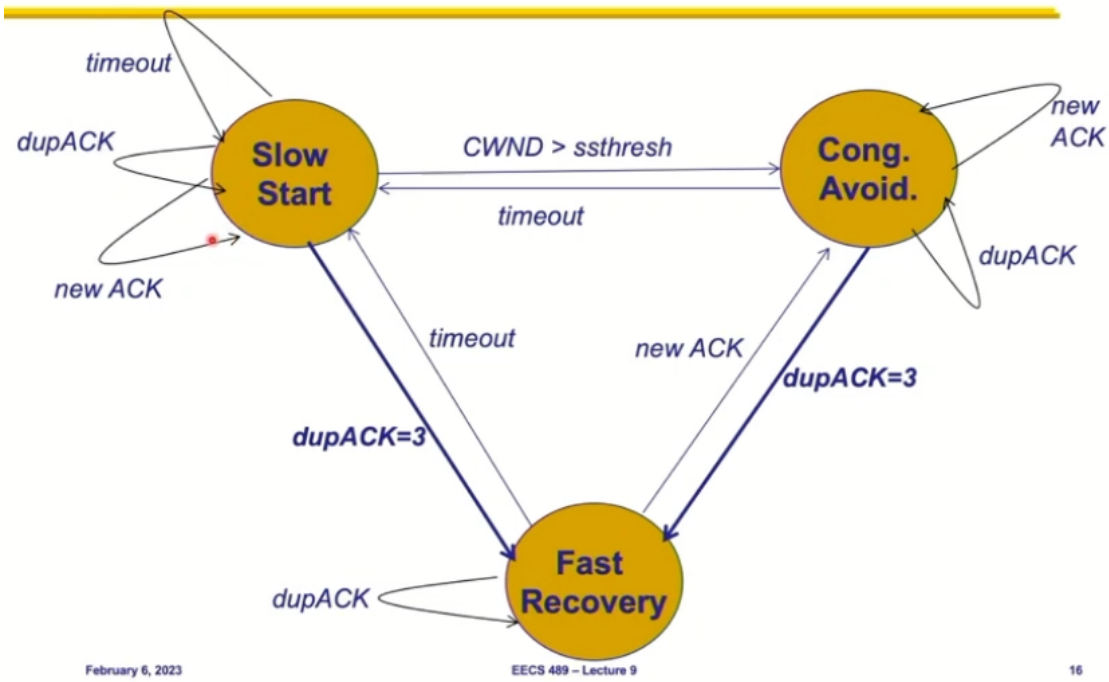
- Solution -- fast recovery
  - Grant the sender temporary "credit" for each dupACK so as to keep packets in flignt
  - If dupACKcount = 3
    - ssthresh = CWND / 2
    - CWND = ssthresh + 3
  - While in fast recovery
    - CWND = CWND + 1 for each additional dupACK
  - Exit fast recovery after receiving new ACK
    - Set SWND = ssthresh



We are sending more data to network, retransmission working

- Packet 111-114 already in flight

TCP state machine

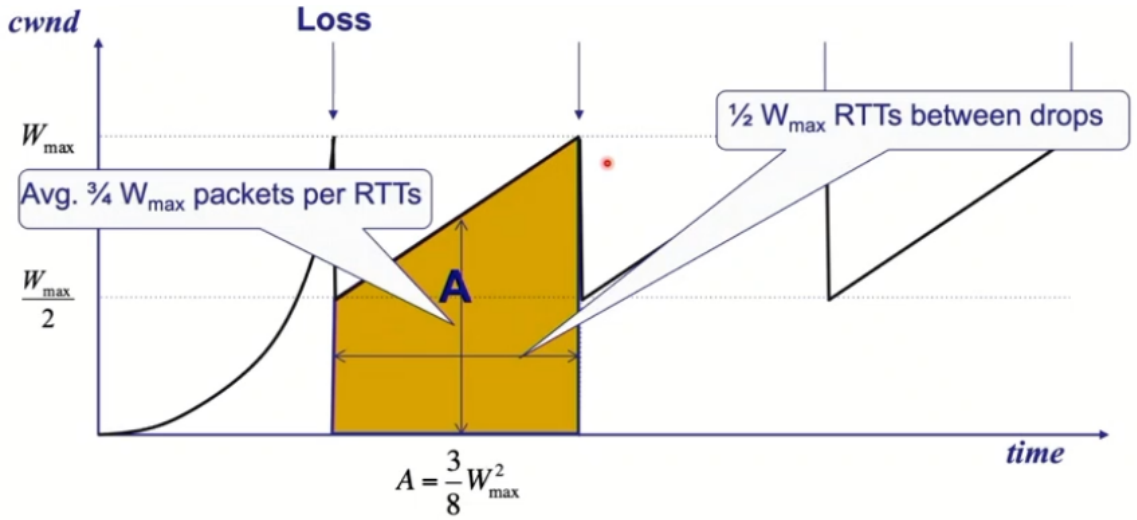


TCP flavors

- TCP-Tahor(outdated)
  - CWND = 1 on 3 dupACKs
- TCP-Reno
  - CWND = 1 on timeout
  - CWND = CWND/2 on 3 dupACKs
- TCP-newReno (out default)
  - TCP-Reno + improved fast recovery
- TCP-SACK
  - Incorporate selective acknowledgement

They can coexist -- all follow the same principle (increase on good news and vise versa)

TCP Throughput Equation



$$A = \frac{3}{8} W_{max}^2$$

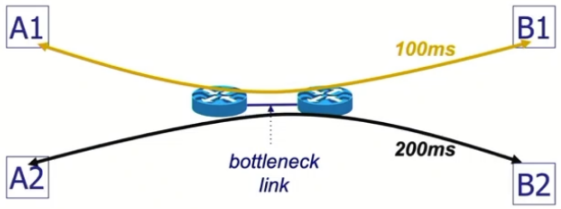
Packet drop rate, $p = 1/A$

$$\text{Throughput, } B = \frac{A}{\left(\frac{W_{max}}{2}\right) RTT} = \sqrt{\frac{3}{2}} \frac{1}{RTT\sqrt{p}}$$

Implications (1): Different RTTs

$$\text{Throughput} = \sqrt{\frac{3}{2}} \frac{1}{RTT\sqrt{p}}$$

- Flows get throughput inversely proportional to RTT
- TCP unfair in the face of heterogeneous RTTs!



Implications (2): High-speed TCP

$$\text{Throughput} = \sqrt{\frac{3}{2}} \frac{1}{RTT\sqrt{p}}$$

- Assume RTT = 100ms, MSS=1500bytes, BW=100Gbps
- What value of p is required to reach 100Gbps throughput?
  - ~ 2×10^{-12}
- How long between drops?
  - ~ 16.6 hours
- How much data has been sent in this time?
  - ~ 6 petabits

Adapting TCP to high speed

- Once past a threshold speed, increase CWND faster
  - A proposed standard [Floyd'03]: once speed is past some threshold, change equation to $p^{-.8}$ rather than $p^{-.5}$
  - Let the additive constant in AIMD depend on CWND
- Other approaches?
  - Multiple simultaneous connections (hack but works today)
  - Router-assisted approaches

Implications (3): Rate-based CC

$$\text{Throughput} = \sqrt{\frac{3}{2}}\frac{1}{RTT\sqrt{p}}$$

- TCP throughput is swings between W/2 to W
- Apps may prefer steady rates (e.g., streaming)
- "Equation-Based Congestion Control"
  - Ignore TCP's increase/decrease rules and just follow the equation
  - Measure drop percentage p, and set rate accordingly
- Following the TCP equation ensures "TCP friendliness"
  - i.e., use no more than TCP does in similar setting

Implications (4): Loss not due to congestion?

- TCP will confuse corruption with congestion
- Flow will cut its rate
  - Throughput ~ 1/sqrt(p) where p is loss prob.
  - Applies even for non-congestion losses!

Implications (5): Short flows cannot ramp up

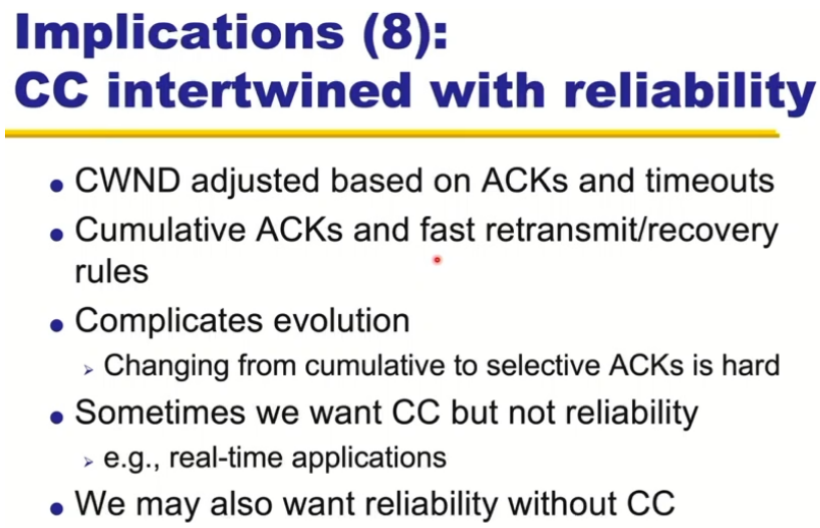
- 50% of flows have < 1500B to send; 80% < 100KB
- Implications
  - Short flows never leave slow start!
    - »They never attain their fair share
  - Too few packets to trigger dupACKs
    - »Isolated loss may lead to timeouts
    - »At typical timeout values of ~500ms, might severely impact flow completion time

Implications (6): Short flows share long delays

- A flow deliberately overshoots capacity, until it experiences a drop
- Means that delays are large, and are large for everyone
  - Consider a flow transferring a 10GB file sharing a bottleneck link with 10 flows transferring 100B
  - Larger flows dominate smaller ones

Implications (7): Cheating

- Three easy ways to cheat
  - Increasing CWND faster than +1 MSS per RTT
  - Using large initial CWND
    - »Common practice by many companies
  - Opening many connections



Router-Assisted Congestion Control

TCP Problems

- TCP treats congestion & non-congestion loss all to congestion loss
  - Non-congestion loss: channel error, package reordering. package corruption
  - Router can tell endpoints if they are congested
- Fills up queues leading to high delays
  - Router can tell endpoints if they are congested, congest sooner before loss
- Short flows complete before discovering available capacity
  - Routers tell endpoints what rate to send at, since router sees the traffic
- AIMD impractical for high-speed links
  - Routers tell endpoints what rate to send at, since router sees the traffic
- Saw tooth discovery to choppy for some apps (up down, up down behavior)
  - Routers tell endpoints what rate to send at, since router sees the traffic
- Unfair under heterogeneous RTTS (different flows have different RTTs) -- sharing same bandwidth
  - Routers tell endpoints what rate to send at, since router sees the traffic
- Tight coupling with reliability mechanisms
  - Routers enforce fair sharing
- End hosts can cheat --not easy to detect
  - Routers enforce fair sharing

Most of them can be fixed with help from routers

Router assisted congestion Control

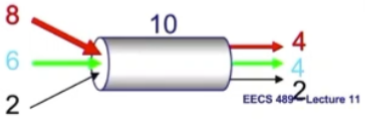
Three tasks for congestion control

- Isolation/fairness
- Adjustment
- Detecting congestion

Fairness

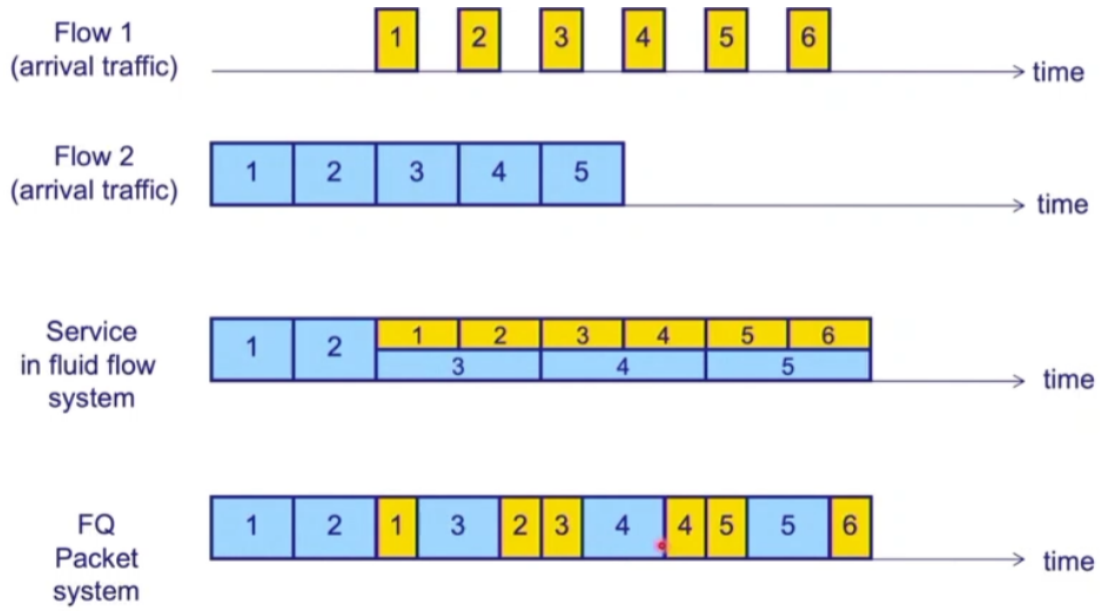
- General approach
  - Routers classify packets into "flows"
    - assume flows a TCP connection
  - Each flow has its own FIFO queue in router
  - Router services flows in a fair fashion
    - When line becomes free, take packet from next flow in a fair order
- **MAX-MIN fairness**: given set of bandwidth demands r_i and total bandwidth C, max-min bandwidth allocations are 

  - $a_i = min(f, r_i)$
  - where f is the unique value such that $Sum(a_i) = C$
  - **If you do not get full demand, then no one else can get more than you get**
  - C = 10; r_1 = 8, r_2 = 6, r_3 = 2; N = 3
  - C/3 = 3.33 →
    - r_3 needs only 2
      - »Can service all of r_3
    - Remove r_3 from the accounting: C = C – r_3 = 8; N = 2
  - C/2 = 4 →
    - Can't service all of r_1 or r_2
    - So hold them to the remaining fair share: f = 4

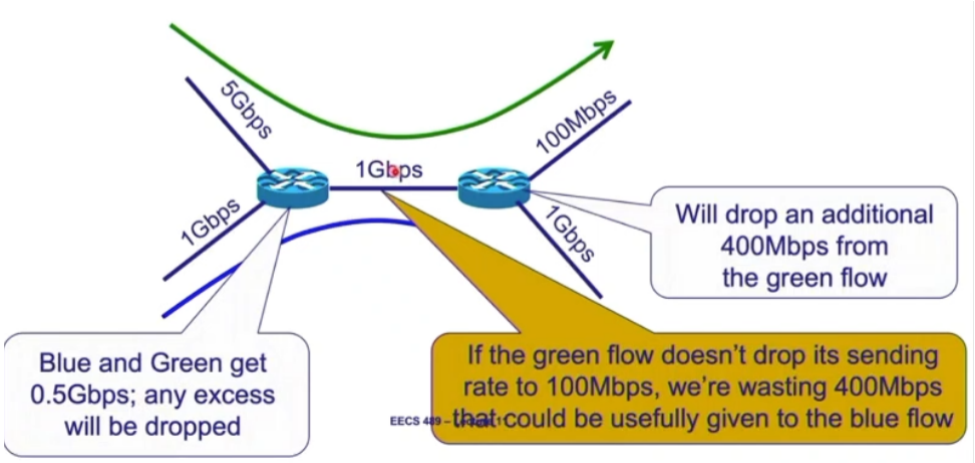


f = 4:
min(8, 4) = 4
min(6, 4) = 4
min(2, 4) = 2

  - Deal with packets of different sizes:
    - Bit-by-bit round robin -- not practical since cannot preempt(stop in middle) packets

        - Try to approximate: Fair Queuing
            - For each packet, compute the time at which the last bit of a packet would have left the router if flows are served bit-by-bit
            - Then serve packets in the increasing order of their deadlines
        - 

        - Implementation of round-robin to the case where not all packets are equal sized
        - Weighted fair queuing: assign different flows different shares
- FQ vs FIFO
    - FQ:
        - Advantage:
            - Isolation: cheating flows don't benefit
            - Bandwidth share does not depend on RTT
            - Flows can pick any rate adjustment scheme they want
            - Robust to cheating, variation in RTT, details of delay, reordering, retransmission
        - Disadvantage:
            - More complex than FIFO
                - Overhead: per flow queue, additional per-packet book-keeping
- FQ does not eliminate congestion -- it manage the congestion
    - What would the end-to-end arguments say w.r.t congestion control
        - The congestion control still needs to be implemented at the end host (not full solution)

- 

- Fairness is a controversial goal

Adjustment

Why not let routers tell what rate end hosts should use?

Packets carry 'rate field', router insert 'fair share' field in packet header

End-hosts set sending rate to field

RCP

- Flows react faster

Detecting congestion

ECN: Explicit congestion notification

- Single bit in packet header, set by congested router
  - If data packet has bit set, then ACK has ECN bit set
- Many options for when routers set the bit: tradeoff between link utilization and packet delay
- Congestion semantics can be exactly like that of drop
- Advantage
  - Don't confuse corruption/congestion, recovery faster
  - Serve as an early indicator of congestion to avoid delays
  - Easy to incrementally deploy -- Common in datacenter

IP Layers

- Autonomous System, or Domain: Region of a network under a single administrative entity
- Border router: at the edge of network, communicate between each other
- Interior router only need to connection inner network

Forwarding

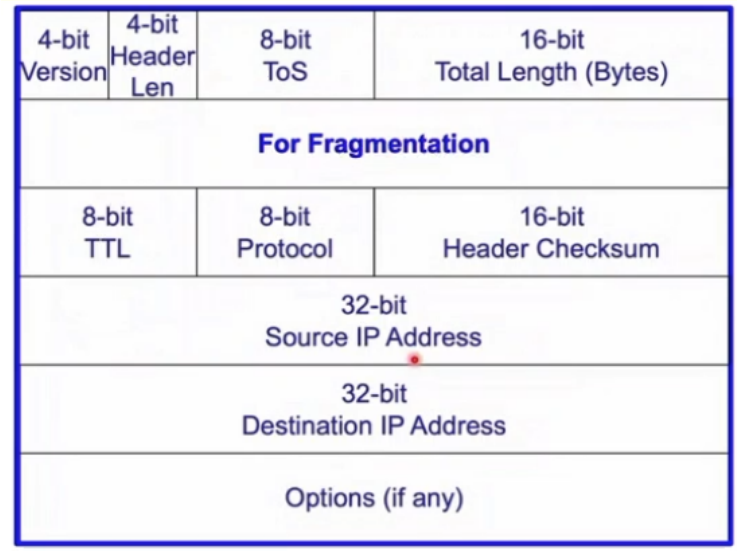
- Look at destination header IP and match the entry in the forwarding table, if no match then use default
- Directing a packet to the correct interface so that it progress to its destination
    - Local operation : read address and search
- "Data plane": use local routing state

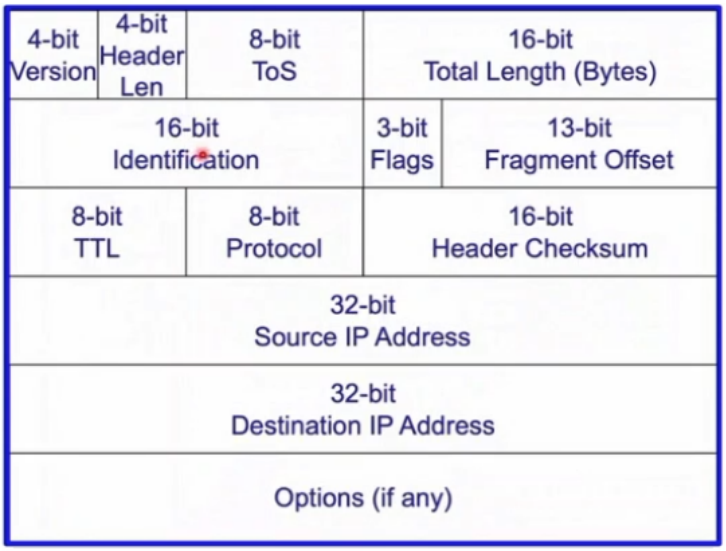
Routing

- Setting up network-wide forwarding table to enable end-to-end communication
    - Global operation: use different routing protocols
- "Control plane": need to compute using distributed algorithm and update table

Routing and Forward has very different timescale -- local is fast

IP Header



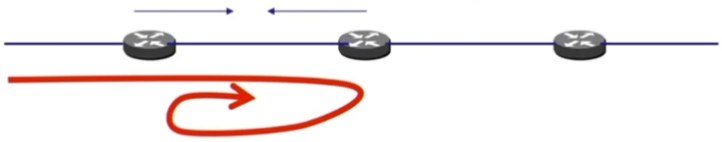


- Task for IP
  - Parse packet
    - **IP version number (4 bits), packet length (16 bits)**
  - Carry packet to the destination
    - **Destination's IP address (32 bits)**
  - Deal with problems along the way
    - Loops: **TTL (8 bits)**
    - Corruption: **checksum (16 bits)**
    - Packet too large: **fragmentation fields (32 bits)**
  - Accommodate evolution: **version number (4 bits)**
  - Special handling: **ToS (8 bits), Options (variable length)**
  - Tell end-host how to handle packet: **Transport Layer Protocol (8 bits)**
    - Identifies the higher level protocol

        - Important for de-multiplexing at receiving host
        - 6 for TCP, 17 for UDP
    - Get response to the packet back to the source
        - **Source IP address (32 bits)**
    - Total 20 - 60 Byte

Preventing loops (TTL)

- Forwarding loops cause packets to cycle for a long time
  - Left unchecked would accumulate to consume all capacity



- Time-to-Live (TTL) Field (8 bits)
  - Decremented at each hop; packet discarded if 0
    - »"Time exceeded" message is sent to the source

Header corruption (Checksum)

- Checksum (16 bits)
  - Particular form of checksum over packet header
- If not correct, router discards packets
  - So it doesn't act on bogus information
- Checksum recalculated at every router

(header is modified)

Fragmentation

- Every link has a “Maximum Transmission Unit” (MTU)
  - Largest number of bits it can carry as one unit
- A router can split a packet into multiple “fragments” if the packet size exceeds the link’s MTU
- Must reassemble to recover original packet

Special handling

- “Type of Service” (8 bits)
  - Allow packets to be treated differently based on needs
    - »e.g., indicate priority, congestion notification
  - Has been redefined several times
  - Now called “Differentiated Services Code Point (DSCP)”

- Special handling to give priority

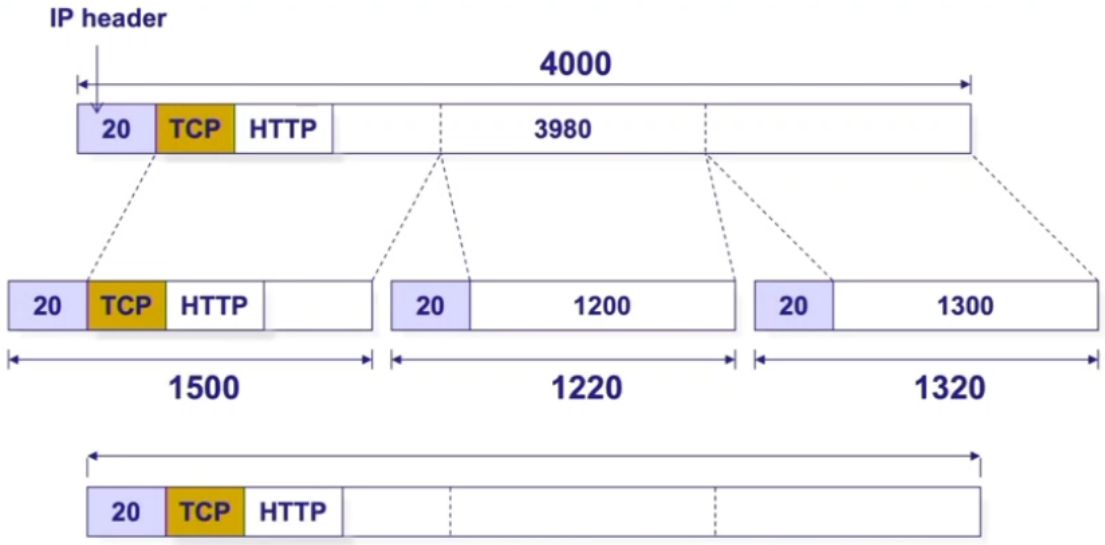
Options

- Optional directives to the network
  - Not used very often
  - 16 bits of metadata + option-specific data
- Examples of options
  - Record Route
  - Strict Source Route
  - Loose Source Route
  - Timestamp

Parse packet

- Header length (4 bits)
  - Number of 32-bit words in the header
  - Typically “5” (for a 20-byte IPv4 header)
  - Can be more when IP options are used

Fragmentation



- Every link has a maximum transmission unit -- MTU
    - Largest number of bits it can carry as one unit
- A router can split a packet into multiple "fragments" if the packet size exceeds the link's MTU
- Must reassemble to recover original packet
    - Or there are package does not have TCP header information
- Where we should do reassemble
    - Classic case of E2E principle
        - At next-hop router imposes burden on network
            - Complicated reassembly algorithm
            - Must hold onto fragments/state
        - Any other router may not work
            - Fragments may take different paths
        - Little benefit, large cost for network reassembly
        - **Done at the destination**
- How to reassemble
    - Need a way to identify fragments -- fill the holes
        - **IPv4's fragmentation fields**
            - **Identifier**: which fragments belong together
            - **Flags**:
                - Reserved: ignore

        - DF: don't fragment -> might trigger error
        - MF: more fragments coming
    - Offset: where this fragment at related to original payload
- Why not use numbering each fragment -- allow further fragmentation of fragments

Datagram split into 3 pieces. Possible first piece:

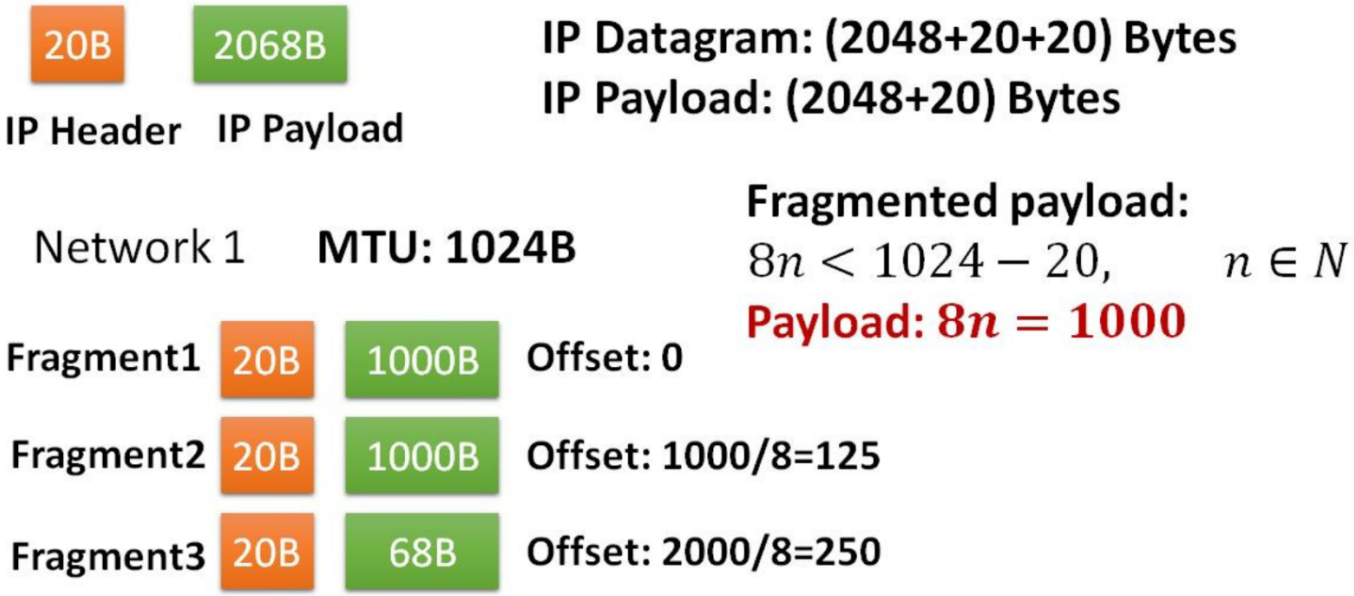
Version 4	Header Len 5	ToS 0	Total Length (Bytes) 1500
Identification 56273		R/D/M 0/0/1	Fragment Offset 0
TTL 127	Protocol 6		Header Checksum xxx
Source IP Address 1.2.3.4			
Destination IP Address 5.6.7.8			

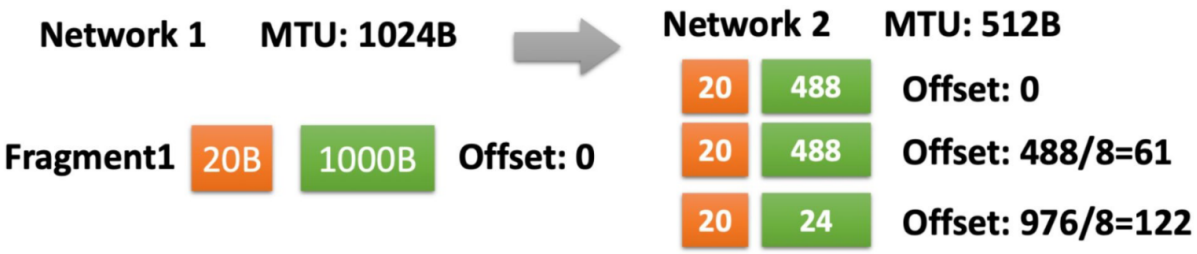
Possible second piece: Frag#1 covered 1480bytes

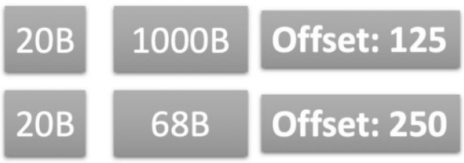
Version 4	Header Len 5	ToS 0	Total Length (Bytes) 1220
Identification 56273		R/D/M 0/0/1	Fragment Offset 185 (185 * 8 = 1480)
TTL 127	Protocol 6		Header Checksum yyy
Source IP Address 1.2.3.4			
Destination IP Address 5.6.7.8			

Possible third piece: 1480+1200 = 2680

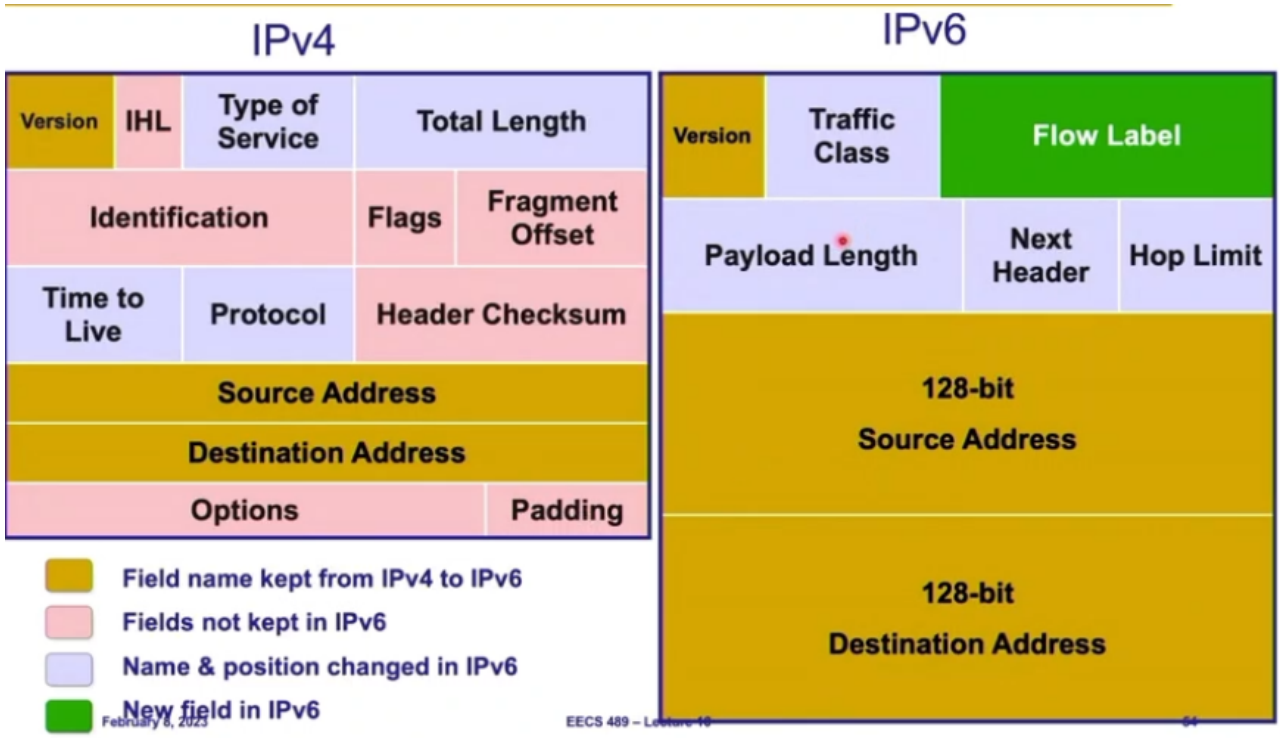
Version 4	Header Len 5	ToS 0	Total Length (Bytes) 1320
Identification 56273		R/D/M 0/0/0	Fragment Offset 335 (335 * 8 = 2680)
TTL 127	Protocol 6		Header Checksum zzz
Source IP Address 1.2.3.4			
Destination IP Address 5.6.7.8			







IPv6

- 128 bit 4 times of v4
- 

- Changes
  - Eliminate fragmentation: end-host will be in charge of doing this -- it can discover the MTU for entire path
  - Eliminate checksum: check in layer three (UDP)

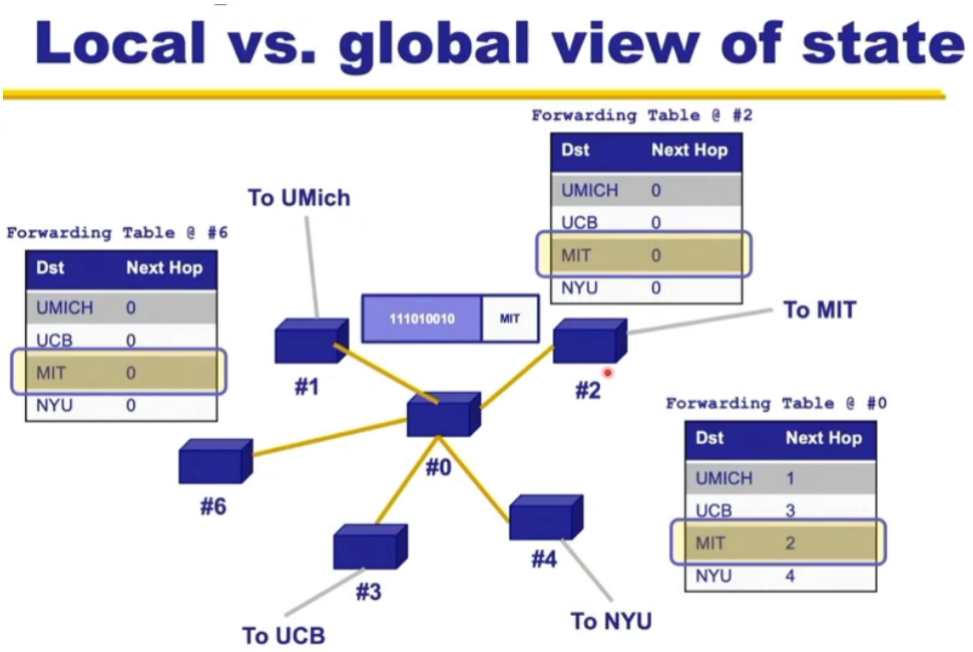
    - New options mechanism: make protocol extensive, header is simple, we don't want router to do much work
    - Eliminate header length: fixed length -- 40bytes
    - Expended address
    - Add Flow label: for flow balancing -- to choose path and to recover from error
- Philosophy of change -- leave to ends!
    - Why we still need TTL: network is best way to handle loops
    - Simplify handling
    - Provide general flow label for packet
        - Not tied to semantics
        - Flexibility

Routing Fundamentals

Goal of routing

- 1. find a path to a destination
- **how do we know state contained in a forwarding table meets our goal:**
    - "validity" of routing table
    - back-hole
    - routing loop
- 2. Find a **least-cost path** to a given destination

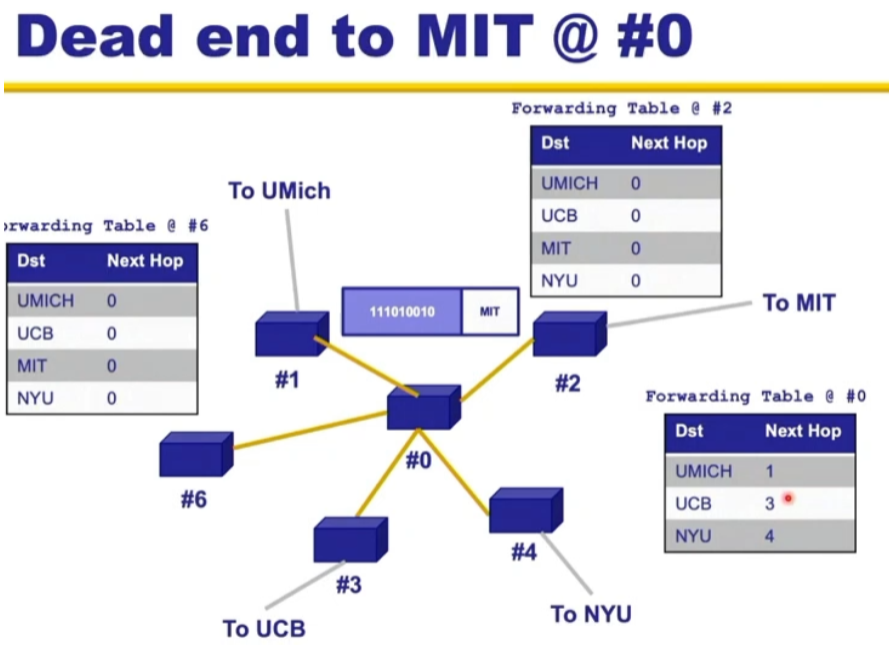
Local vs global view of state



- Local routing state is the forwarding table in a single router
  - Cannot be evaluated
  - Must be evaluated in term of global context: you cannot discover @2 has routing loop from itself
- Global state is the collection of forwarding tables in each of routers
  - Determines which path packets to take

Valid routing state

- Global state is valid if it produce forwarding decisions that always deliver packets to their destinations
- Goal of routing protocol: compute valid state
- **Need necessary and sufficient condition**
  - Global routing state is valid iff
    - **There is no dead ends other than dest**
    - **No loops**
  - A dead end is there is no outgoing link (next hop)
    -



Checking the validity of routing state

- Focus only on a single dstination
  - Ignore all other
- Make outgoing link with arrow
  - There is only one at each node
- Eliminate all links with no arrows
- Look at what's left
- If result in spanning tree -> valid
- If not -> invalid

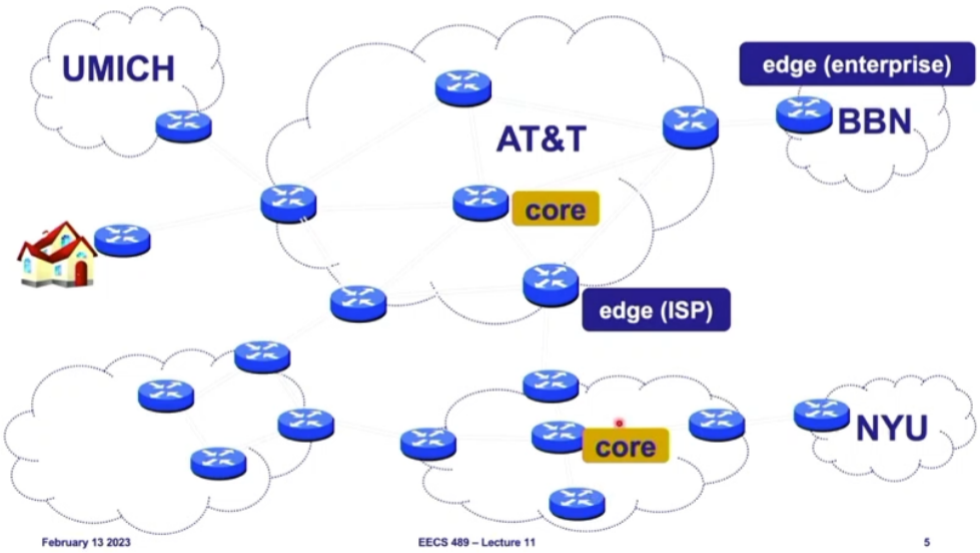
Least-cost path routing

- Propagation delay
- Load
- Cost

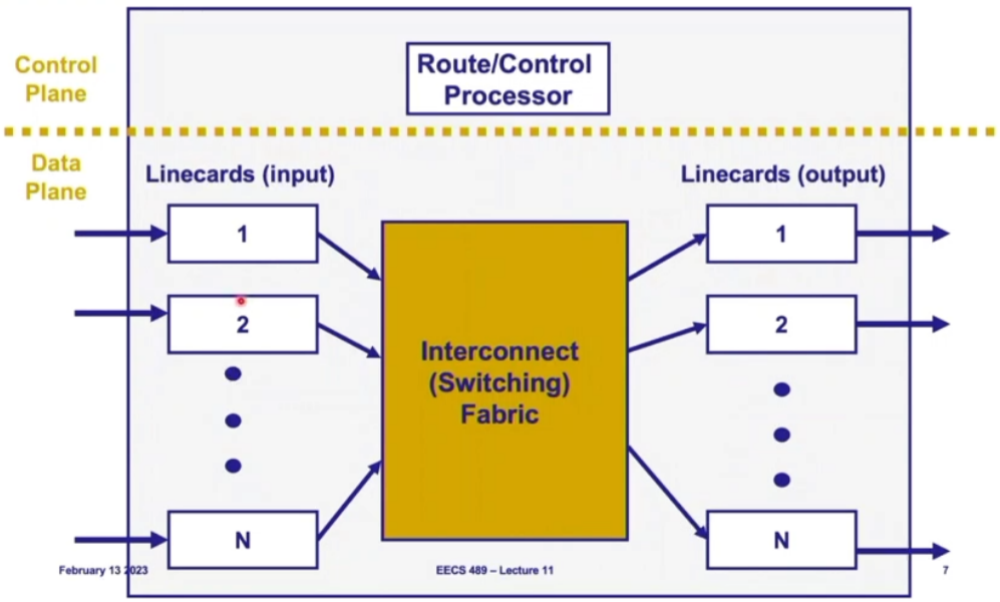
Given: router graphs & costs

- **Least-cost routes provides an easy way to avoid loops**
- Least-cost paths from a spanning tree for each destination rooted at that destination
- Dijkstra

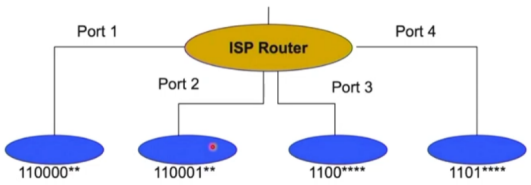
IP Routers

- Router Capacity: N * R
  - N is the number of external router "ports"
  - R is the speed of a port
- Routers
  - Core/Edge, edge does not offer service for others(bottom of 等级，consumer)
    -
      - Core
        - R = 10/40/100 Gbps
        - NR = O(100) Tbps (Aggregated)
      - Edge
        - R = 1/10/40
        - NR = O(100) Gbps
      - Small business
        - R = 10/100/1000 Mbps
        - NR < 10 Gbps
    -
    - 

- Router Structure
  -



- Processor: computing the route, decide destination
  - Not frequently replaced, expensive and and fabric needs also get upgrade with processor
- Linecards:
  - Input and output for the same port are on the same physical linecard
    - **Input**: Receive incoming packets and update IP header
      - TTL, Checksum, Options, Fragments to decide if to send
    - Look up output port for IP
      - **Aggregated** IP, assign address range mapping among ports
        - Longest prefix matching



        - Tree structure (decision tree), record port associated with latest match, and only override when it matches another prefix during walk down tree
        -

Range for interface 2 cannot be described with a single prefix! Need to split.

Destination Address Range	Interface
11100000 00000000 00000000 00000000 11100000 00111111 11111111 11111111	0
11100000 01000000 00000000 00000000 11100000 01000000 11111111 11111111	1
11100000 01000001 00000000 00000000 11100001 01111111 11111111 11111111	2
otherwise	3

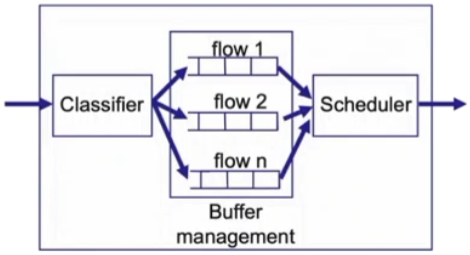
11100000 01000001 00000000 00000000
11100000 11111111 11111111 11111111

11100001 00000000 00000000 00000000
11100001 01111111 11111111 11111111

-

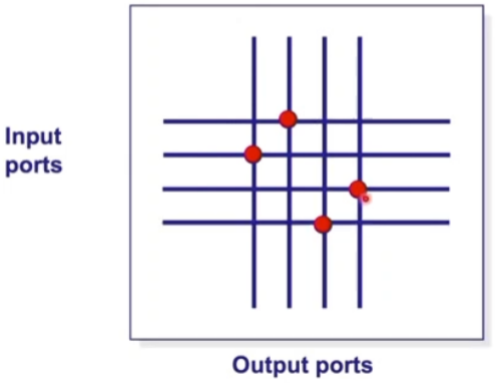
Destination Address Range	Interface
11100000 00(/10)	0
11100000 01000000(/16)	1
11100000 (/8)	2
11100001 0(/9)	2
otherwise	3

- Queue
- **Output**:



  - Packet classification: map packets to flows (Buffer management)
    - Classify IP packet based on a number of fields in the packet header
      - Src/Des IP address (32 bit)
      - Src/Des TCP port number (16 bit)
      - Type of service, TOS byte(8 bits)
      - Type of protocol (8 bits), priority TCP since UDP are a lot
    - fields are specified by range: classification require a multi-dimensional range search
  - Buffer management: decide when and which packet to drop
  - Scheduler: decide when and which packet to transmit
    - FIFO: when buffer is full drop the incoming packet

            - Problem for large data -- unfairness
        - Priority scheduler: in highest priority serve first -- starvation
            - **Round-robin scheduler**: packets are served from each queue in turn: if serve a large packet from a queue, do not serve that queue next round
                - Fair queuing: round-robin for packets of different size
                - Weighted fair queuing: serve proportional to wright
                    - gives equal weight to each flow
        - One queue per flow
        - Goals: fast, depends on the policy being implemented
    - **Connection input and output**
        - Mini-network, three ways to switch
            - **Shard memory**
            - **Bus**
            - **Inter-connection network eg cross-bar**
                - 2N buses intersecting with each other
                - Non-blocking



                - better approach: provide isolation, not affecting each other
    - Speed matters: 100B @ 40Gbps -> new package every 20 nano secs
        - Implemented with specialized ASICs (network processors) -> expensive
- Fabric
    - Transfer packets from input to output ports

- Intra-Domain Routing
  - OSPF
    - Link State
  - RIP
    - Distance Vector

  IGP = Interior Gateway Protocol
- Inter-Domain Routing
  - BGP = Border Gateway Protocol
    - eBGP = external BGP
    - iBGP = internal BGP

Intra-AS Routing Protocols(inside a domain)

From routing algorithm to protocol

- Dijkstra is a local computation
  - Computed by a node given complete network graph
- We can
  1. Do a separate machine runs the algorithm
  2. **Every router runs the algorithm** -- currently used

Link-state routing

- Every router knows its local "link state"
- Each router **floods** its local link state to all other routers in the network
  - Does it periodically or when its link state changes
- Each router learns the enter network graph
  - Each runs Dijkstra locally to compute forward table
- Scalability
  - O(NE) messages
  - $O(N^2)$ computation time
  - O(Network diameter) convergence delay
  - O(N) entries in forwarding table

Flooding link state

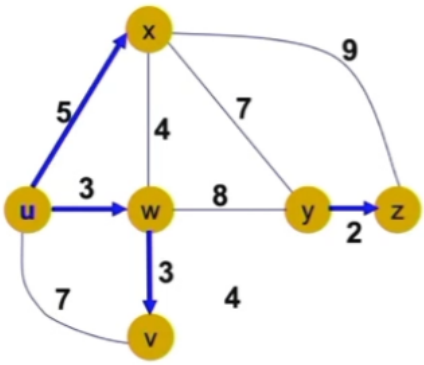
- **Flooding**
  - A node sends its link-state info our on all of its links
  - The next node forwards the info on all of its links except the one the information arrived at
- **Initiate flooding**
  - Topology change -- link/node failure/recovery
  - Configuration change -- link cost change
  - Periodically
    - 30 mins typically
    - To refresh link-state information

Convergence (quicker than 30 min refresh)

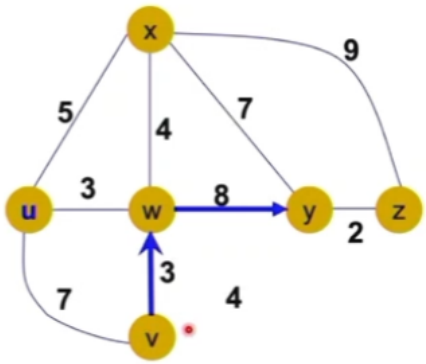
- Why flood
  - To get all the nodes in the network to converge to the new topology
- Upon convergence, all nodes will have **consistent routing information**(no loop) and can **compute consistent forwarding**:
  - All nodes have the same link-state database
  - All nodes forward packets on shortest paths
  - The next router on the path forwards to the expected next hop

Convergence delay

- Time to achieve convergence
- **Sources of convergence delay**
  - **Time to detect failure**
  - **Time to flood link-state information**
  - **Time to re-compute forwarding tables**
- What happen if too long to converge -- inconsistent state



u and w think that the path to y goes through v



v thinks that the path to y goes through w

Performance during convergence period

- **Looping packets**
  - 因为有的走的比较慢，之后的还在用old state
- **Lost packets** due to black holes
- **Out-of-order packets** arriving destination

Link-state routing protocols

OSPF: Open Shortest Path Frist(Public available)

- Uses link-state algorithm
  - Link-state packet dissemination
  - Topology map at each node
  - Route computation using Dijkstra's algorithm
- Router floods OSPF link-state advertisements to all other routers in entire AS
  - **Carried in OSPF message directly over IP (not UDP/TCP)**
    - **Requires reliable transmission**

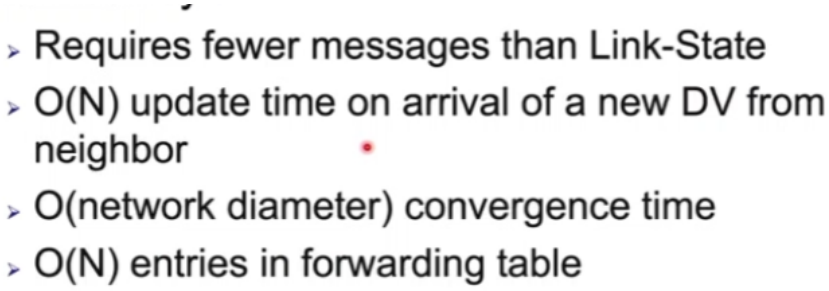
IS-IS: Intermediate System to Intermediate System, similar to OSPF

Distance-vector protocol

Property

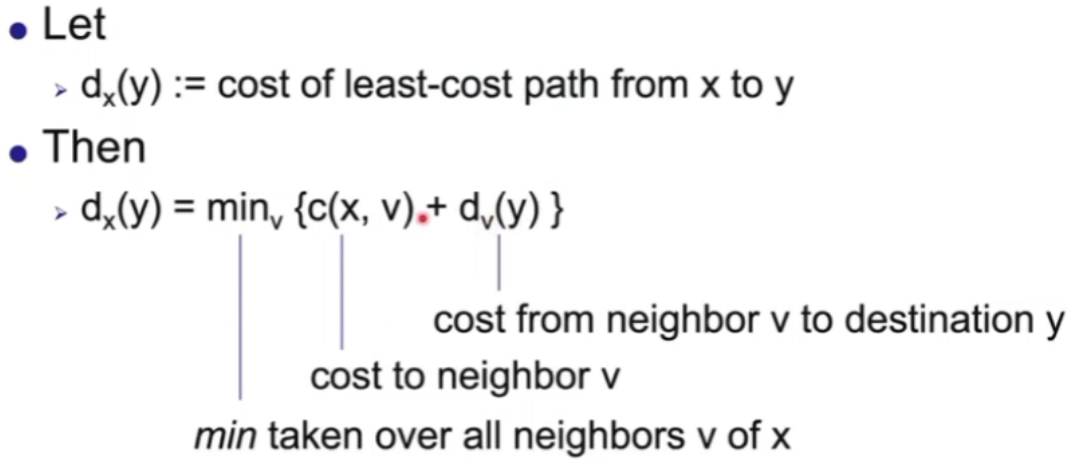
- **DV loop occurs from removing a link**
- **Count-to-infinity problem may occur when the cost of a link increase**
- Link-state routing protocol

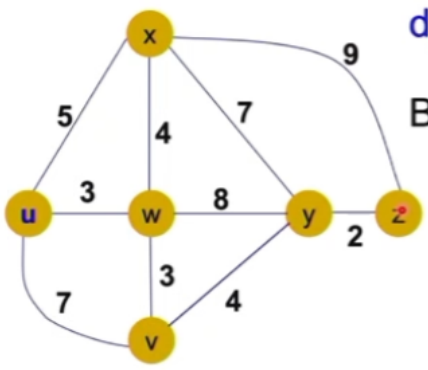
    - Each node broadcasts its local information
- Distance-vector routing protocol
    - Each nodes tells its neighbor about its global view (its view of how to reach dest)



    - RIP is a protocols

Bellman-Ford equation





$d_x(z) = 9, d_w(z) = 9, d_v(z) = 6$

B-F equation says:

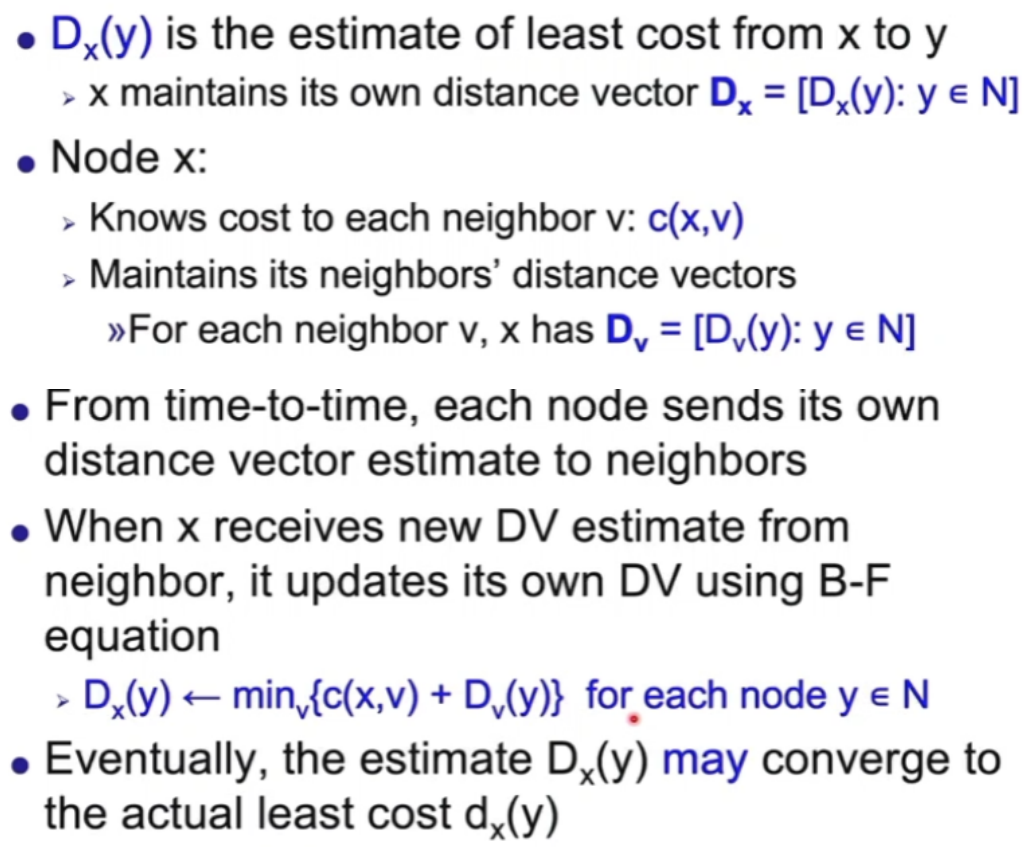
$$d_u(z) = \min \{c(u,x) + d_x(z), c(u,w) + d_w(z), c(u,v) + d_v(z)\}$$
$$= \min \{5 + 9, 3 + 9, 7 + 6\} = 12$$

Neighbor achieving the minimum (w) is next hop in shortest path, used in forwarding table

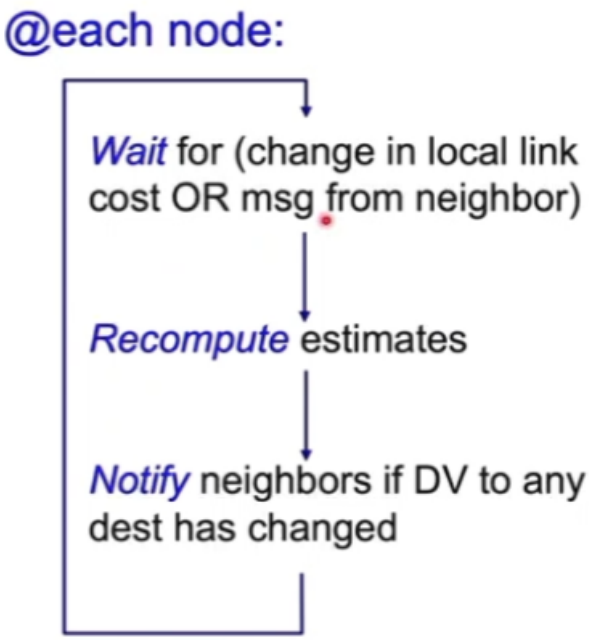
- Might ends up with a loop!

Distance vector algorithm

- Dx(y) is the estimate of least cost from x to y
  - x maintains its own distance vector Dx = [Dx(y):y in N]
- Node x:
  - Knows cost to each neighbor v:c(x, v)
  - Maintain its neighbors distance vectors
  - for each neighbor v, x has Dv = [Dv(y):y in N]



- Iterative asynchronous
  - Local iterations caused by
    - Local link cost change
    - DV update messages from neighbor
- Distributed
  - Each node notifies neighbors only when its DV chanegs
    - Neighbors then notify their neighbors if necessary



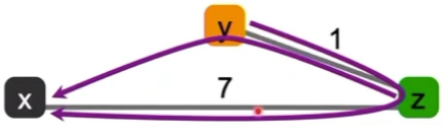
- Scalability?
  - Requires fewer messages than Link-State
  - O(N) update time on arrival of a new DV from neighbor
  - O(network diameter) convergence time
  - O(N) entries in forwarding table

- RIP is a protocol that implements DV (IETF RFC 2080)

Problem with Bellman-Ford

- Routing Loops that takes a long time to resolve
  - Count-to-infinity scenario
- -> Poisoned reverse: One heuristic to avoid count-to-infinity
  - z advertises to y what its cost to x is infinite
  - y never decides to route to x through z
  - **Not guaranteed**, not working for 3+-hop loop
  - Loop-free routing examples
    - Path vector - BGP

- Source tracing



Messaging complexity

- LS: with N nodes, E links, O(NE) messages sent
- DV: exchange between neighbors only

Speed of convergence

- LS: relatively fast
- DV: convergence time varies
  - Count-to-infinity problem

Robustness: what happens if router malfunctions?

- LS:
  - Node can advertise incorrect link cost
  - Each node computes its *own* table
- DV:
  - Node can advertise incorrect path cost
  - Each node's table used by others (error propagates)

Similarities between LS and DV routing

- Both are shortest-path based routing
  - Minimizing cost metric (link weights) a common optimization goal
    - »Routers share a common view as to what makes a path "good" and how to measure the "goodness" of a path
- Due to shared goal, commonly used inside an organization
  - RIP and OSPF are mostly used for intra-domain routing

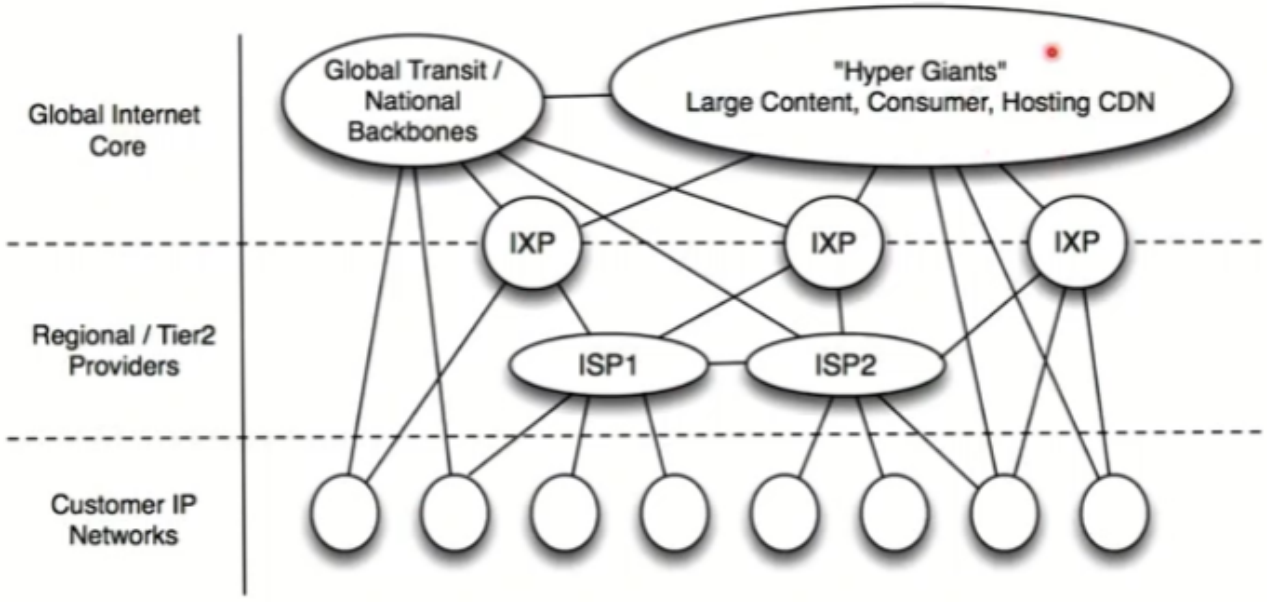
Intro-domain routing(between different domain)

Domain/Autonomous System: region of a network under a single administrative entity.

Border routers: at the edge of domain, connect to other network

Interior routers: inside of domain

AS-Level internet



Internet Inter-Domain Traffic, SIGCOMM, 2010

IXP: internet exchange point

ISP: Internet service provider

Autonomous systems, AS

- An AS is a network under a single administrative control
  - Currently over 74000 ASes
- ASes are sometimes called domains
- Each AS is assigned a unique identifier (ASN)

"Intra-Domain" routing: within an AS

- Link-State(OSPF) and Distance-Vector(RIP)
- Focus:
  - Fast convergence
  - Find least cost path

"Inter-domain" routing between ASes

- 2 key challenges
    - Scaling
    - Administrative structure
        - Issues of autonomy, policy, pricavy
- Component
    - Addressing (scalability)
    - BGP

Scaling

- A router must be able to reach any destination
    - Given packet's destination address, lookup next hop
- Naive: have an entry for each dest -- 10^8 entries, slow and large (and routing updates per destination)

Fix: longest-prefix matching

Careful address assignment -> can aggregate multiple addresses into one range -> scalability

Akin to reducing the number of destinations

- Better: have an entry for a range of address
    - Can't do this if addresses are assigned randomly
- **Host addressing is key to scaling**

Administrative structure shapes inter-domain routing

- AS want **freedom in picking routes**
- AS want **autonomy** -- choose own internal routing protocol and policy

Choice of routing algorithm

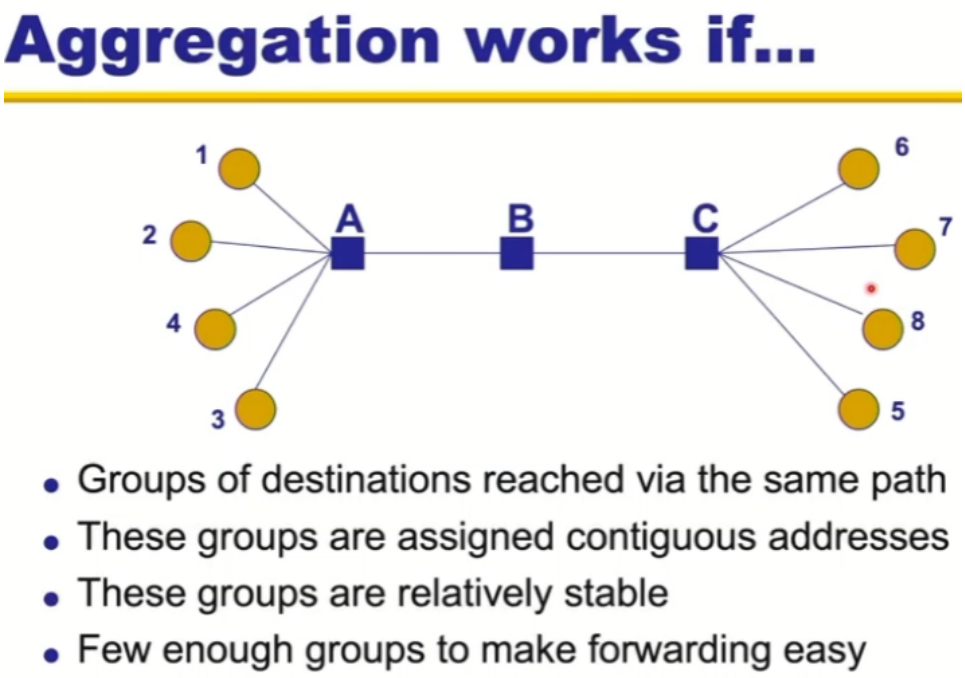
- Link-state:
    - No privacy, limited autonomy
- Distance-vector
    - Give some control
    - But wasn't designed to implement policy

    - vulnerable to loops
- **Border Gateway Protocol extends distance-vector ideas to accommodate policy**

IP Addressing

Goal of addressing: Scalable routing

- State: Small forwarding tables at routers
    - Much less than the number of hosts
- Churn: limited rate of change in routing tables
- **Ability to aggregate** addresses is crucial for both

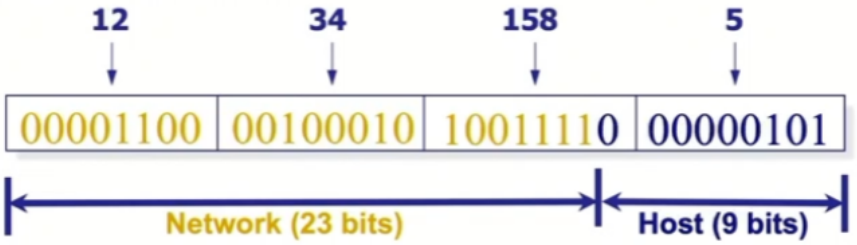


IP addressing is hierarchical

- Structure
- Allocation
- Addresses and routing scalability

IP address (IP v4)

- Unique 32-bit number associated with a host
- Represented with dotted-decimal noration
- Group them into four groups, each 8 bits
- 32 bits are partitioned into a prefix and suffix components
- Prefix is the **network** component, suffix is the **host** component



- Inter-domain routing operates on the **network** prefix - in routing table we only care about prefix part

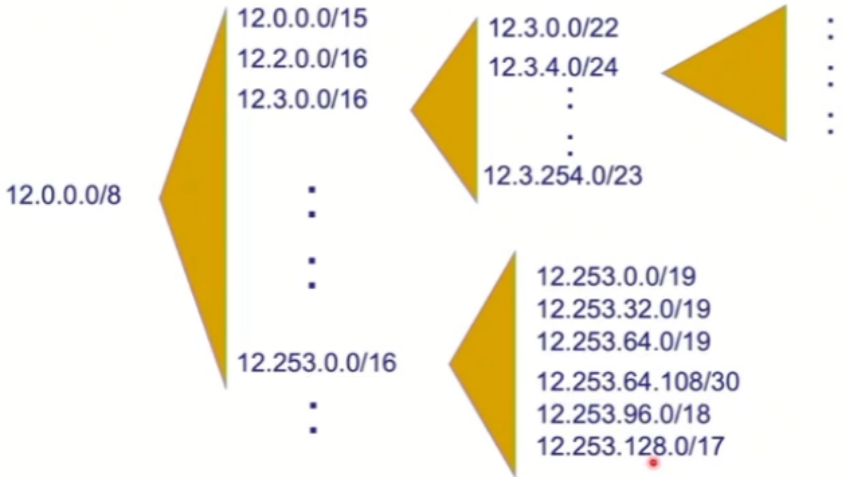
Before CIDR: Classful addressing

- Three classes
  - 8-bit network prefix (Class A),
  - 16-bit network prefix (Class B), or
  - 24-bit network prefix (Class C)
- Example: an organization needs 500 addresses.
  - A single class C address is not enough (<500 hosts)
  - Instead, a class B address is allocated (~65K hosts)
    - » Huge waste!

CIDR: Classless inter-domain routing

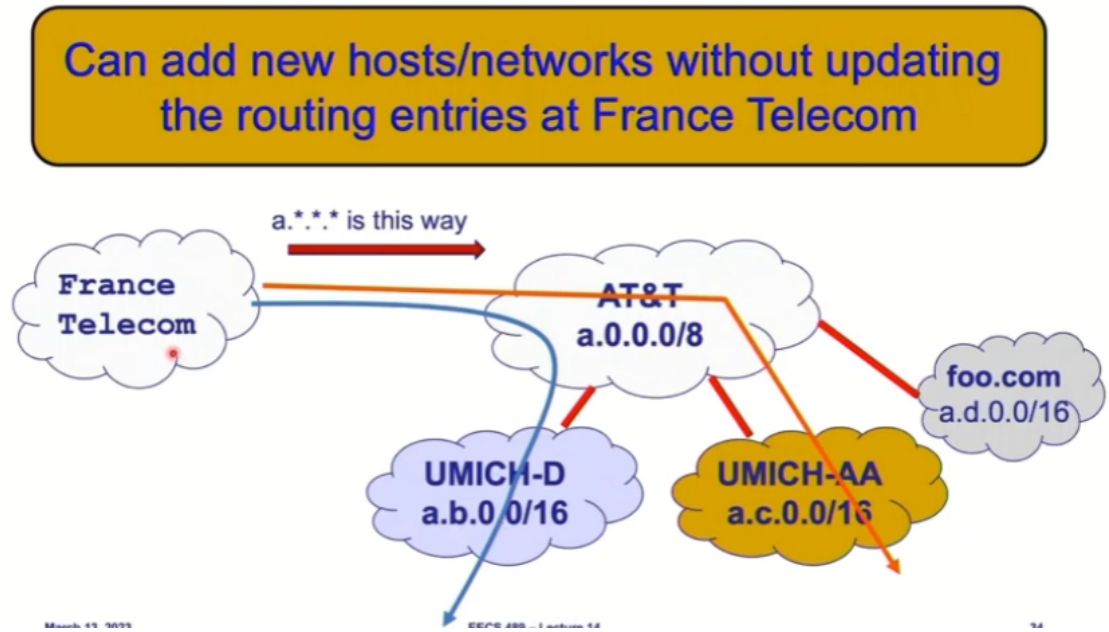
- Flexible division between network and host address
- Offer a better tradeoff between size of routing table and efficient use of IP address space
- Suppose a network has 50 computers
  - Allocate 6 bits for host address 2^6
  - Remaining 32-6=26 bits as network prefix
- Flexible boundary means the boundary must be explicitly specified with the network address
  - 128.23.9/26
  - Formally, prefix represented with a 32-bit mask: 255,255,255,192: all network is 1 and host suffix is 0 --**subnet mask**: group of machines with same prefix
- Recursively break down chunks as get closer to host

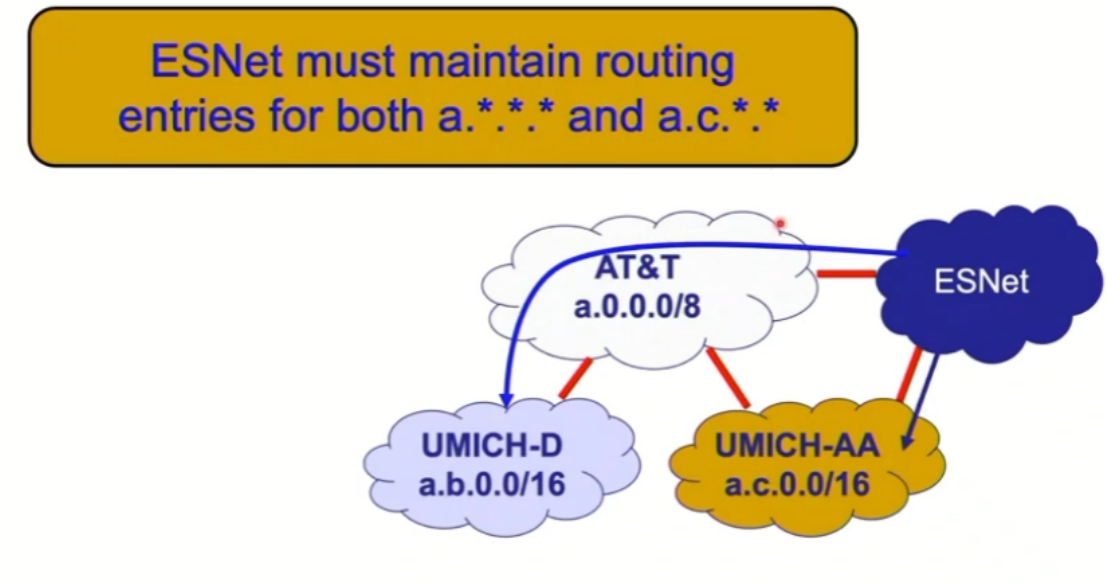
- Recursively break down chunks as get closer to host



- ICANN gave ARIN several /8s
- ARIN gave AT&T one /8, **12.0/8**
  - Network Prefix: 00001100
- AT&T gave UMICH a /16, **12.34/16**
  - Network Prefix: 0000110000100010
- UMICH gave EECS a /24, **12.34.56/24**
  - Network Prefix: 000011000010001000111000
- EECS gave me specific address **12.34.56.78**
  - Address: 00001100001000100011100001001110

IP addressing -> Scalable routing





- Hierarchical address allocation only helps routing scalability if allocation matches topological hierarchy
- **May not be able to aggregate addresses for multi-homes networks**
    - A multi-homes network is connected to more than one AS for fault-tolerance, load balancing, etc

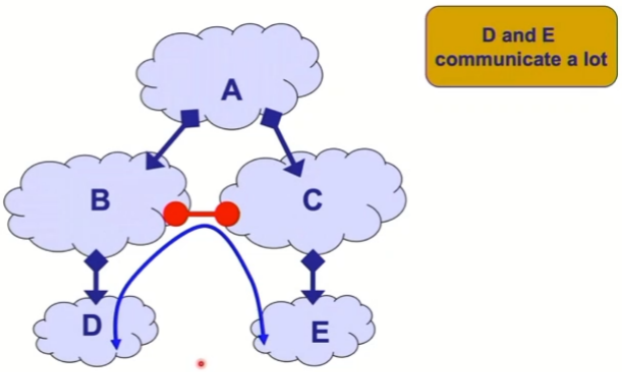
Inter-domain routing: Setup

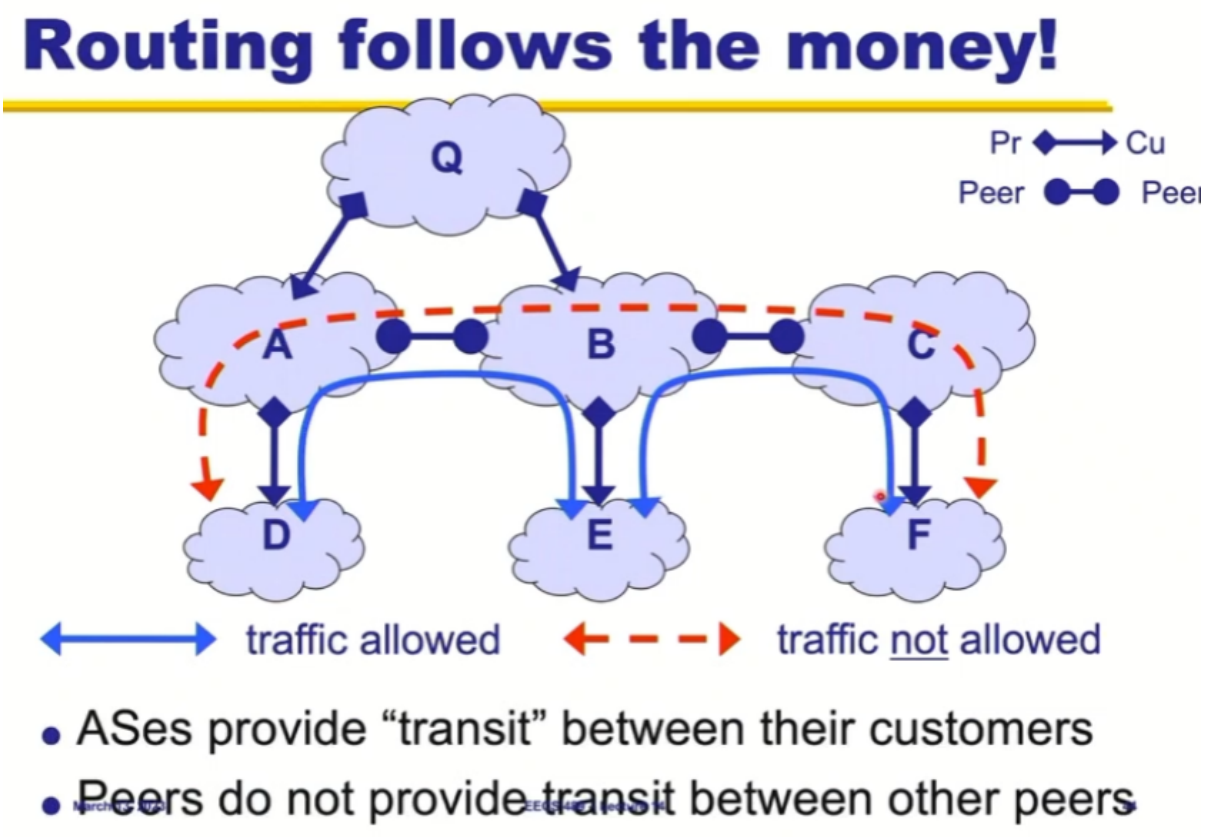
- Destinations are IP prefixes (12.0.0.0/8)
- Nodes are Autonomous Systems (ASes)
    - Internals of each AS are hidden
- Links represent both physical links and business relationships
- BGP (Border Gateway Protocol) is the Inter-domain routing protocol
    - Implemented by AS border routers

BGP: Border Gateway Protocol

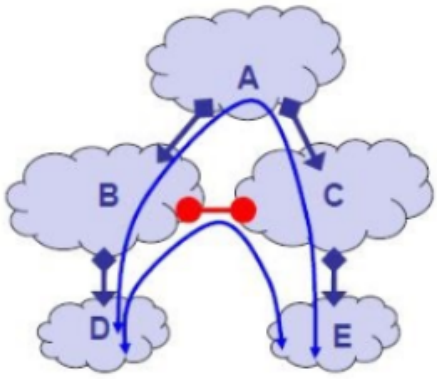
Disallowing some path being used

Topology & policy shared by inter-AS business relationship

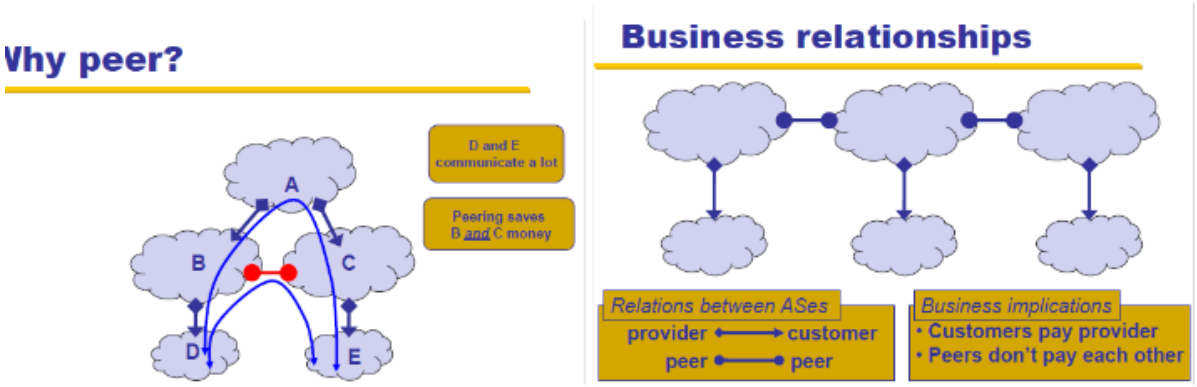
- Customer pay provider
- Provider
- Peer: don't charge each other -- equal traffic exchange to be fair
    - 

    - 

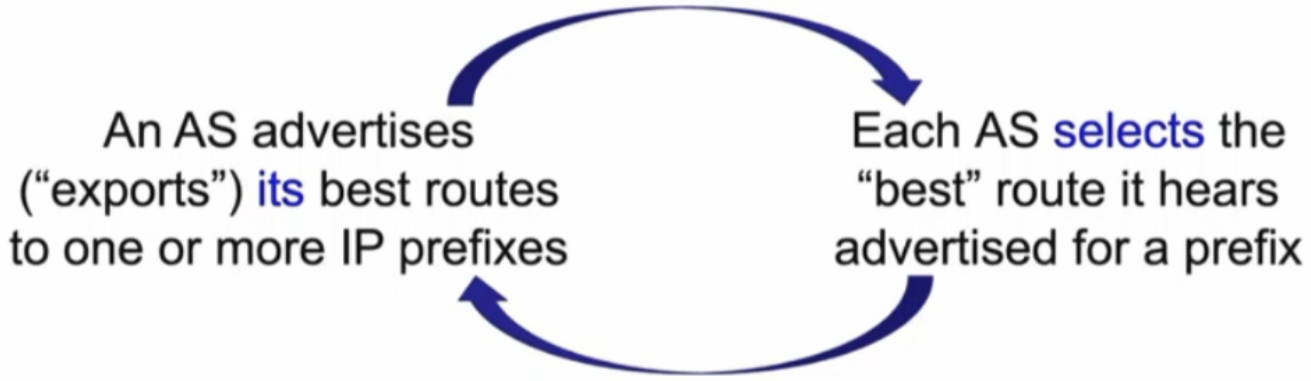
True or False. In the following network, B and C do not need to pay money to communicate if they communicate via A. 1 points



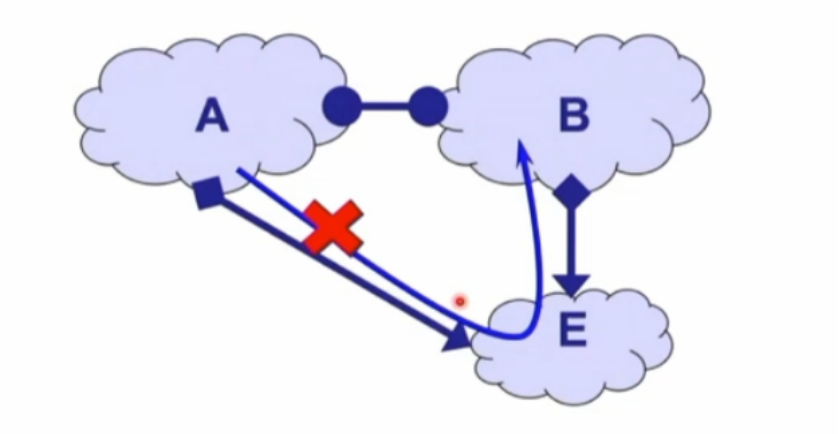
- ○ True
- ◉ False ✓



Basic idea:



valley free:



BGP inspired by Distance -Vector

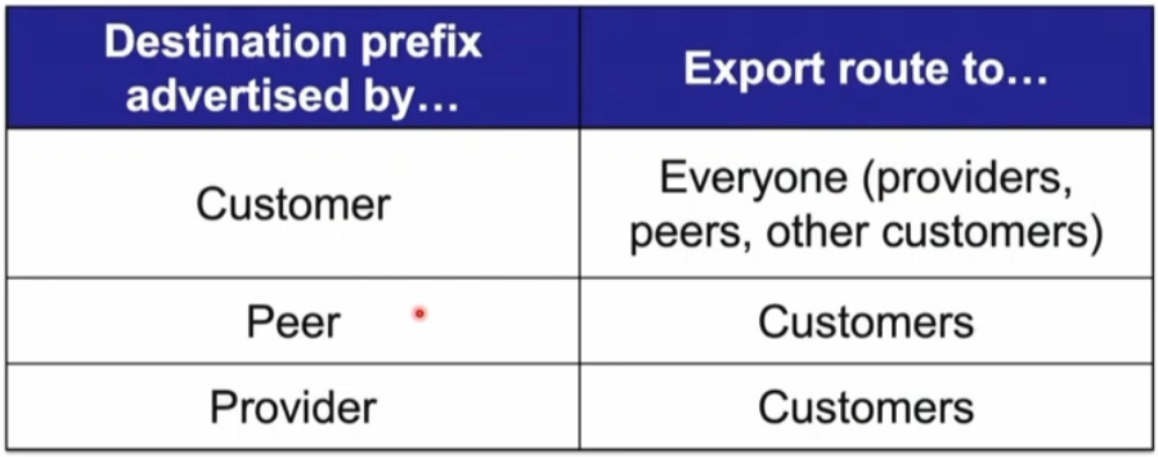
- Per-destination route advertisements
- No global sharing of network topology information - we dont know if some path went down
- Iterative and distributed convergence on path

Differences

1. not picking shortest path route -- BGP is policy driven
2. **path vector routing**
    1. send the **entire path** instead of distance metric
    2. avoid loop and flexible
3. selective route advertisement
    1. for policy reason
    2. **reachability is not guaranteed**
4. BGP may aggregate routes
    1. aggregate routes for different prefixes

Policy

- Policy dictates how routes are "selected" and "exported"
    - **Selection: which path to use (leaves the network)**
    - **Export: which path to advertise (enters the network)**
- Typical selection policies
    - In decreasing order of priority
        - Make/save money: **customer > peer > provider**
        - Maximize performance (path length)

        - Minimize use of my network bandwidth , hot potato - hand off as fast as you can
    - 

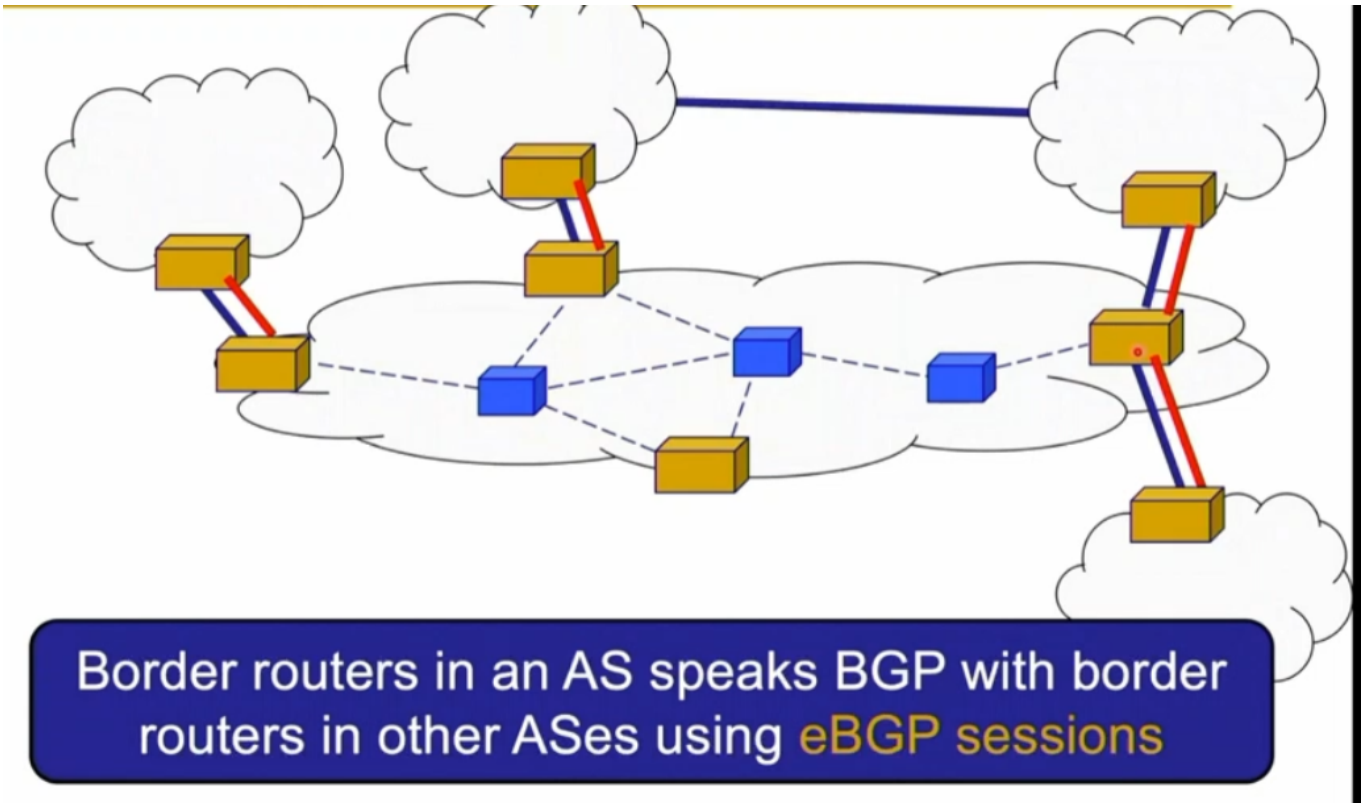
Destination prefix advertised by…	Export route to…
Customer	Everyone (providers, peers, other customers)
Peer	Customers
Provider	Customers

        - Gao-Rexford: the AS policy graph is a directed acyclic graph and routes are "valley free"

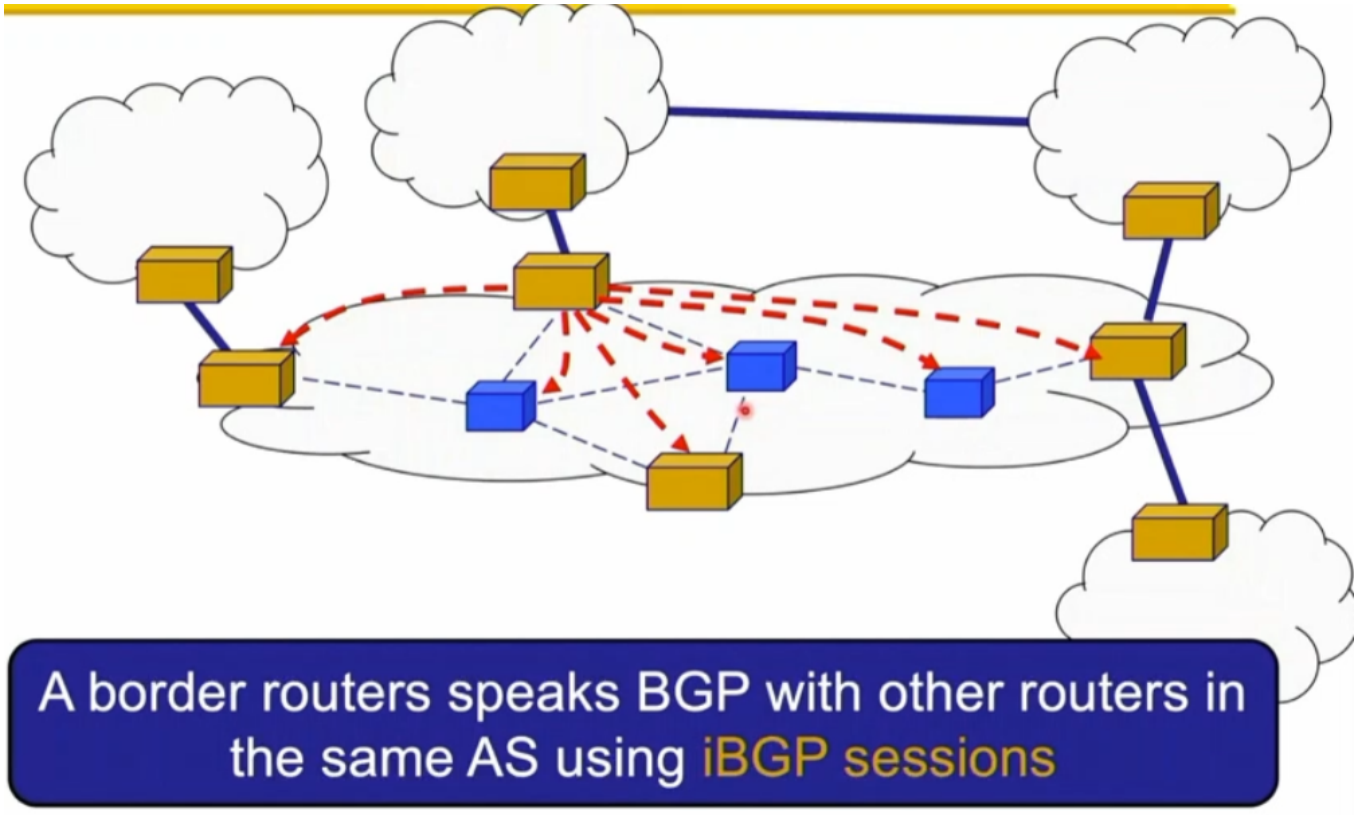
BGP protocol detail

- Border routers in an Autonomous System speaks BGP,
    - implement the BGP protocol
    - Specifies the message to exchange with other BGP speakers
    - How to process these messages
        - Follow BGP state machine and policy decisions

BGP sessions: External/Internal/IGP



- Between border routers in different ASes
    - Learn routes to external destinations



- Between border routers and other routers within the same AS
    - Distribute externally learned routes internally

- IGP: "Interior Gateway Protocol" = Intra-domain routing protocol
    - Provide internal reachability
    - OSPF, RIP
    - within same network, same domain

Working together

- Learn routes to external destinations using eBGP
- Distribute externally learned routes internally using iBGP
- Travel shortest path to egress using IGP

Basic message in BGP

- Open
    - Establish BGP session (TCP)
- Notification
    - Report unusual conditions
- Update
    - Inform neighbor of new routes
    - Inform neighbor of old routes that become inactive
- Keep-alive
    - Inform neighbor that connection is still viable

Route updates

- Format: <IP prefix: route attributes>
    - attributes describe properties of the route
- Two kinds of updates
    - **Announcements**: new routes or changes to existing routes
    - **Withdrawal**: remove routes that no longer exist

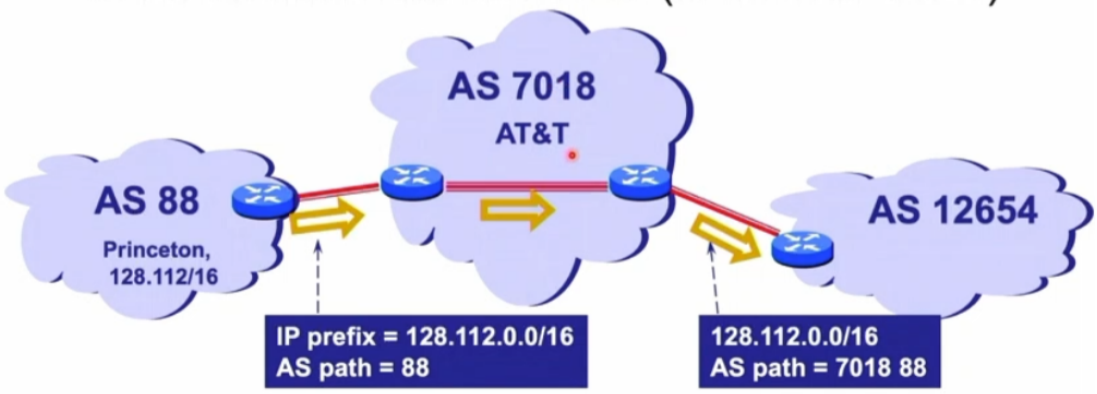
Route attributes

Priority	Rule	Remarks
1	LOCAL PREF	Pick highest LOCAL PREF
2	ASPATH	Pick shortest ASPATH length
3	MED	Lowest MED preferred
4	eBGP > iBGP	Did AS learn route via eBGP (preferred) or iBGP?
5	iBGP path	Lowest IGP cost to next hop (egress router)
6	Router ID	Smallest next-hop router's IP address as tie-breaker

- Routes are described using attributes
  - Used in route selection/export decisions
- Some attributes are local
  - private within an AS, not included in announcements
- Some attributes are propagated with eBGP route announcements
- There are many standardized attributes in BGP
  - ASPATH:
    - Carried in route announcement
    - Vector that lists all the ASes a route advertisement has traversed (in reverse order)
    - AS path length can be misleading -> An AS may have many router-level hops. The attribute may not represent the actual number of router level hops.
    -

Attributes: (1) ASPATH

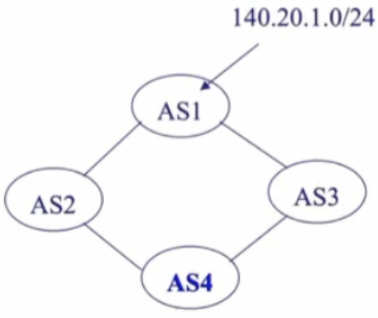
- Carried in route announcements
- Vector that lists all the ASes a route advertisement has traversed (in reverse order)



- LOCAL PREF
  - Local preference in choosing between different AS paths
    - Local to an AS, carried only by iBGP
  - The higher the value the more preferred
  -

Attributes: (2) LOCAL PREF

- Local preference in choosing between different AS paths
  - Local to an AS; carried only in iBGP messages
- The higher the value the more preferred



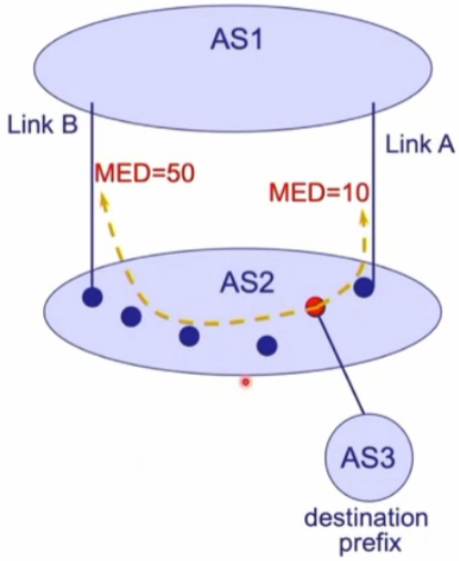
BGP table at AS4:

Destination	AS Path	Local Pref
140.20.1.0/24	AS3 AS1	300
140.20.1.0/24	AS2 AS1	100

- MED
  -

Attributes: (3) MED

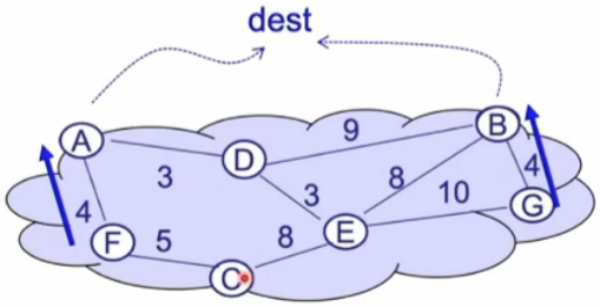
- Multi-exit discriminator is used when ASes are interconnected via 2 or more links; it specifies how close a prefix is to the link it is announced on
- Lower is better
- AS that announces a prefix sets MED
- AS receiving the prefix (optionally!) uses MED to select link



- IGP cost
  -

Attributes: (4) IGP cost

- Used for hot-potato routing
  - Each router selects the closest egress point based on the path cost in intra-domain protocol

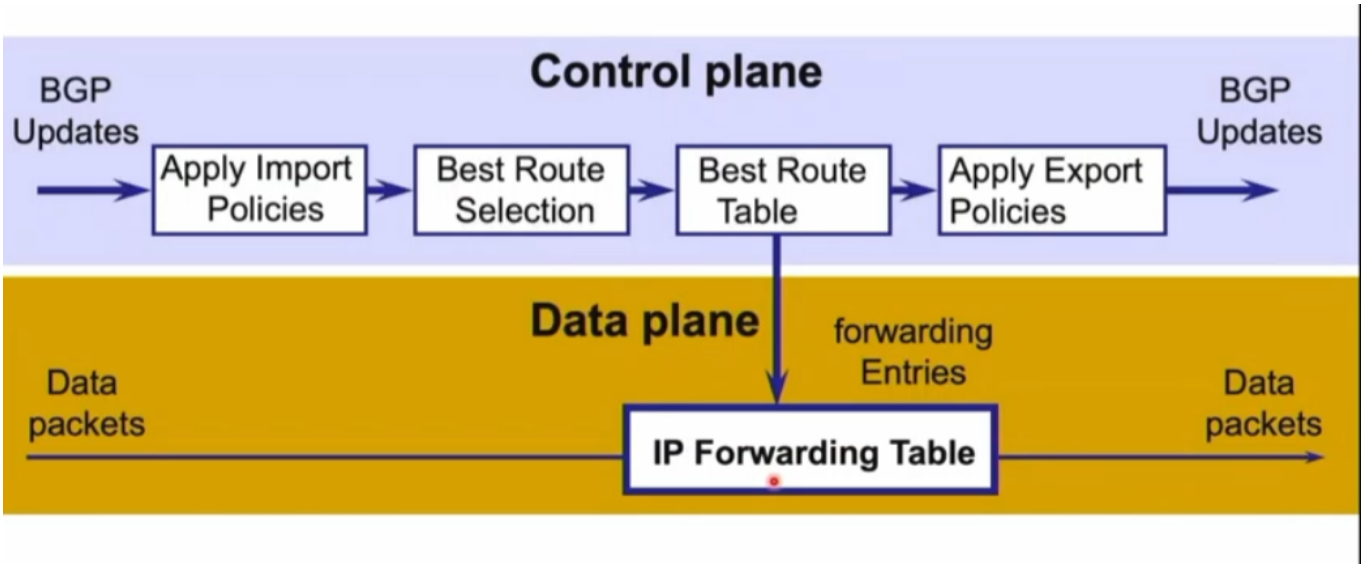


Using attributes

- Rules for route selection in priority order

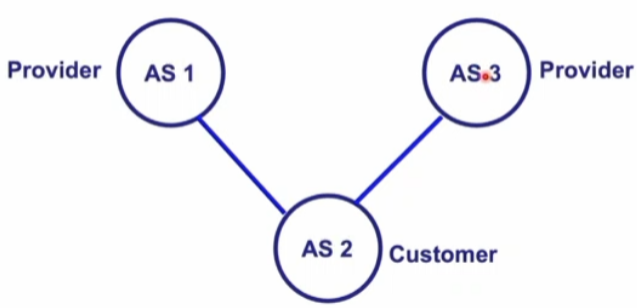
Priority	Rule	Remarks
1	LOCAL PREF	Pick highest LOCAL PREF
2	ASPATH	Pick shortest ASPATH length
3	MED	Lowest MED preferred
4	eBGP > iBGP	Did AS learn route via eBGP (preferred) or iBGP?
5	iBGP path	Lowest IGP cost to next hop (egress router)
6	Router ID	Smallest next-hop router's IP address as tie-breaker

BGP UPDATE processing

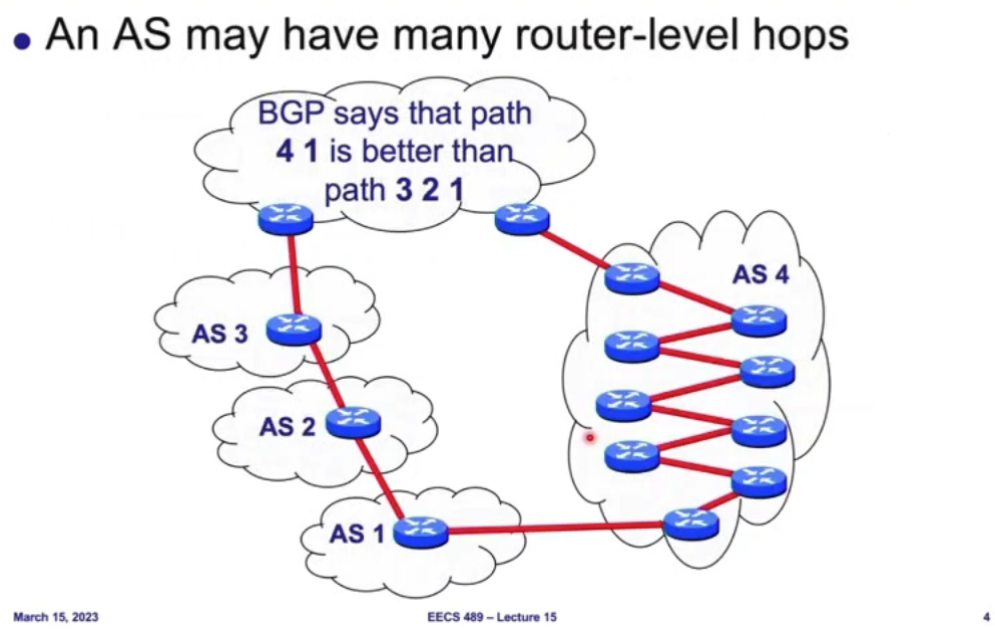


BGP issues in Practice

- Reachability
  - In normal routing, if graph is connected then reachability is assured but policy make it not always hold
  -



- Security
  - An AS can claim to serve a prefix that they do not have a route to (blackholing)
    - Fixable: make them prove they have a path
    - Important since AS autonomu
    - Problem not specific to policy or path vector
  - AS can forward packets different from what is advertised
    - Tell customers fictitious short path
    - harder to fix
- Convergence
  - **Not all AS policies follow "Gao-Rexford" rules, else guaranteed convergence**
- Performance nonissues
  - **Performance nonissues**
    - Internal routing
      - Domains typically use “hot potato” routing
      - Not always optimal, but economically expedient
    - Policy is not always about performance
      - Policy-driven paths aren’t the shortest
    - AS path length can be misleading
      - 20% of paths inflated by at least 5 router hops
    -



- Real performance issue: **slow convergence**
  - BGP outages are biggest source of internet problems
  - Most popular paths are very stable
- BGP misconfiguration
  -
    - BGP protocol is bloated yet underspecified
      - Lots of attributes
      - Lots of leeway in how to set and interpret attributes
      - Necessary to allow autonomy, diverse policies
        - »But also gives operators plenty of rope
    - Configuration is mostly manual and ad hoc
      - Disjoint per-router configuration to effect AS-wide policy

Summary

- Network layer deals with data plane(forwarding) and control plane (routing)
- Control plane deals with intra-domain routing (LS and DV) and inter-domain routing (BGP)

SDN

Software-defined networking

- Networks are much more primitive and less understood than other computer systems

A tale of two planes

- Data plane: forwarding packets
  - Based on local forwarding state
- Control plane: computing that forwarding state
  - Involves coordination with rest of system

Control plane

Original goals for the control plane

- **Basic connectivity**: route packets to destination
  - Local state computed by routing protocols
  - Globally distributed algorithms
- **Inter-domain policy**: find policy-compliant paths
  - Done by globally distributed BGP
- What other goals are there in running a network?

Extended role of control plane

- network management tasks
  - Where to route, how much, what rate, should we route

Bottom line

- Many different control plane mechanisms
- Each designed from scratch for their intended goal
- Encompassing a wide variety of implementations
  - Distributed, manual, centralized,…
- None of them particularly well designed
- Network control plane is a complicated mess!

The power of abstraction

- "Modularity based on abstraction is the way things get done"
  - Barbara Liskov
- Abstractions ➔ Interfaces ➔ Modularity

Control plane mechanisms

- Variety of goals, no modularity
  - Routing: distributed routing algorithms
  - Isolation: ACLs, Firewalls,…
  - Traffic engineering: adjusting weights,…
- Control Plane: mechanism without abstraction
  - Too many mechanisms, not enough functionality

Task: Compute forwarding state -- Abstraction

- Consistent with low-level hardware/software
  - Which might depend on vendor
- Based on entire network topology
  - Because many control decisions depend on topology
- For all routers/switches in network
  - Every router/switch needs forwarding state

Separate concerns with abstractions

- Be compatible with low-level hardware/software
  - Forwarding abstraction
- Make decisions based on entire network
  - Network state abstraction
- Compute configuration of each physical device
  - Specification abstraction

1. Forwarding abstraction

- Express intent independent of implementation
  - Don't want to deal with proprietary HW and SW
- Design details concern exact nature of
  - Header matching
  - Allowed actions

2. Network state abstraction

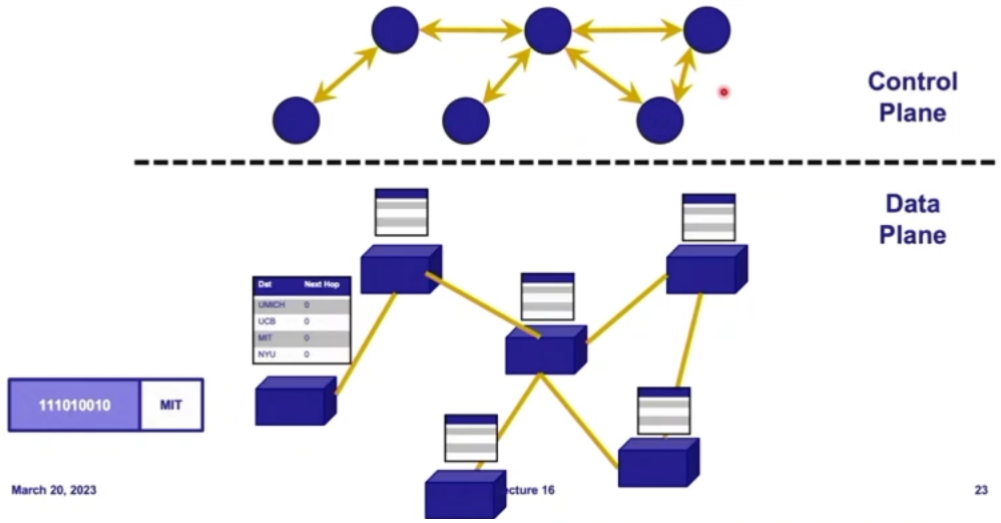
- Abstraction: global network view
  - Annotated network graph provided through an API
- Creates a logically centralized view of the network
  - Runs on replicated servers in network ("controllers")
- Information flows both ways
  - Information from routers/switches to form "view"
  - Configuration to routers/switches to control forwarding

3. Specification abstraction

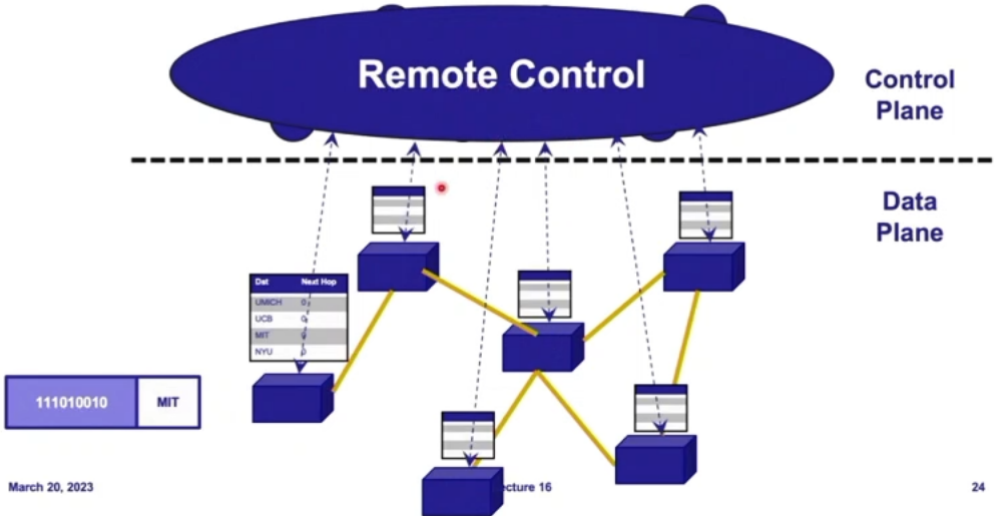
- Control mechanism expresses desired behavior
    - Whether it be isolation, access control, or QoS
- It should not be responsible for implementing that behavior on physical network infrastructure
    - Requires configuring the forwarding tables in each switch
- Abstract view of network
    - Models only enough detail to specify goals
    - Will depend on task semantics

Traditional fully decentralized control plane

- Individual routing algorithm components in every router interact in the control plane



- A distinct (typically remote) controller interacts with local control agents (CAs)



- Each goal is an app via specification abstraction

Logically centralized control plane(physically that might be replicated)

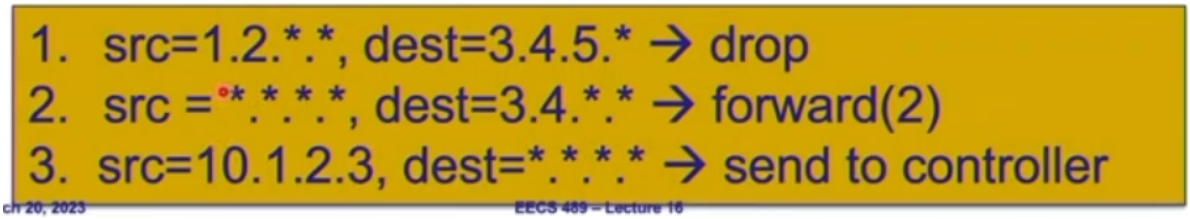
- A distinct controller interacts with local control agents (remote)
- Each router contains a flow table
- Each entry of the flow table defines a match-action rule
- Entries of the flow table is computed and distributed by the logically centralized controller

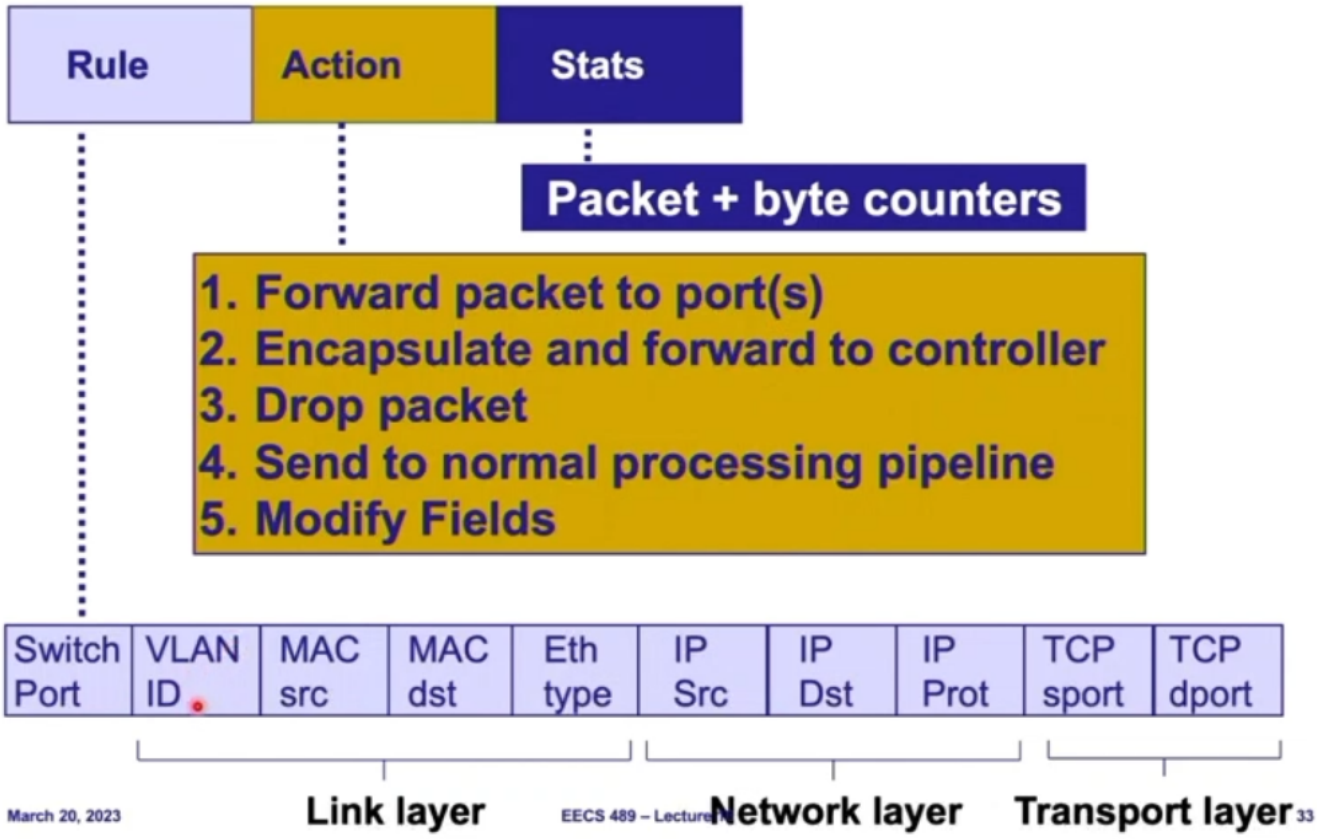
SDN: Many challenges remain

- Hardening the control plane: dependable, reliable, performance-scalable, secure distributed system
  - Robustness to failures: leverage strong theory of reliable distributed system for control plane
  - Dependability, security: “baked in” from day one?
- Networks, protocols meeting mission-specific requirements
  - E.g., real-time, ultra-reliable, ultra-secure
- Internet-scaling

Data plane

OpenFlow data plane abstraction

- Flow is defined by header fields
- Generalized forwarding: simple packet-handling rules
  - **Pattern**: match values in packet header fields
  - **Actions**: For matched packet: drop, forward, modify, matched packet or send matched packet to controller
  - **Priority**: disambiguate overlapping patterns
  - **Counters**: #bytes and #packets
  - 



- Key switch-to-controller messages
  - Packet-in/out: transfer packet and its control to controller
  - Flow-removed: flow table entry deleted at switch
  - Port status: inform controller of a change on a port
- Network operator do not "program" switches by creating&sending OpenFlow messages directly

Forwarding abstraction

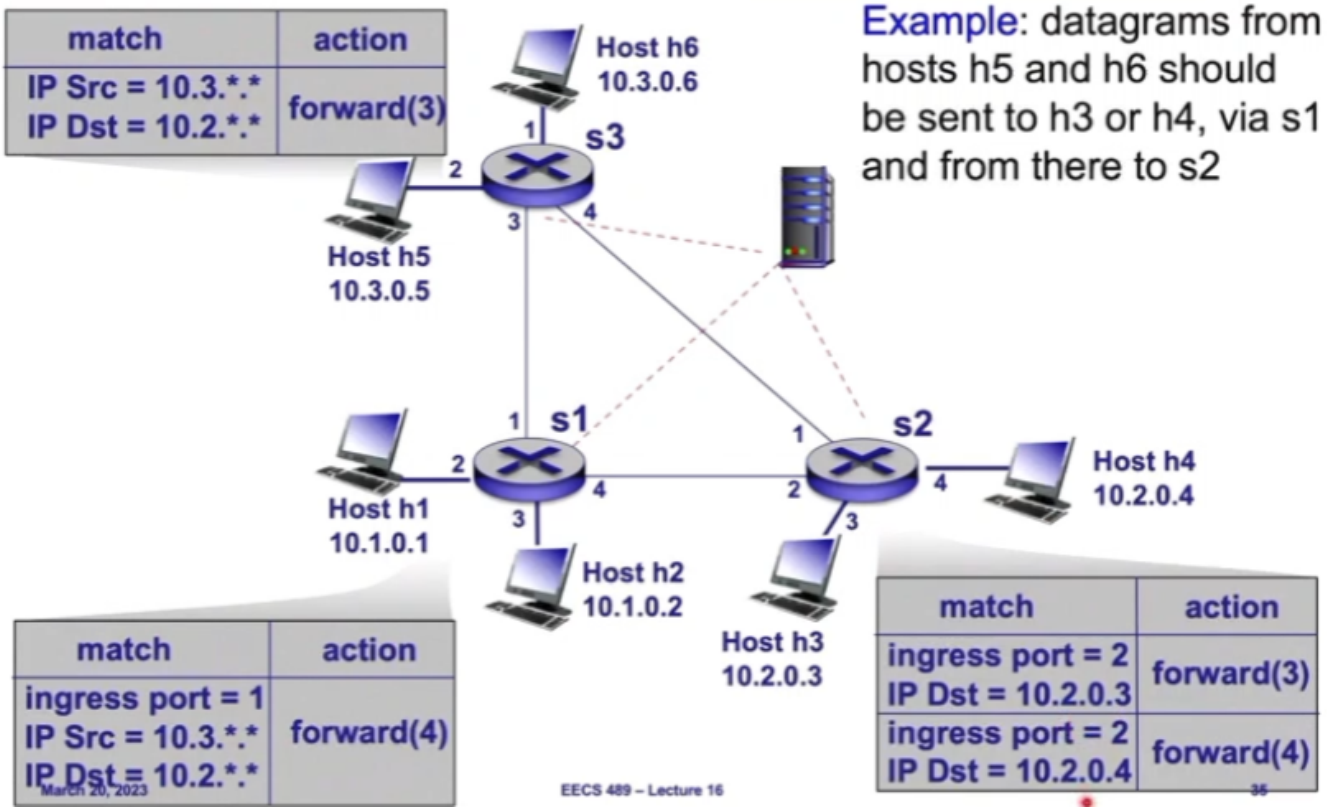
Match+Action: unifies different kinds of devices

- Router
  - Match: longest destination IP prefix
  - Action: forward out a link
- Switch
  - Match: destination MAC address
  - Action: forward or flood
- Firewall
  - Match: IP addresses and TCP/UDP port numbers
  - Actions: permit or deny

- NAT
  - Match: IP address and port
  - Action: rewrite address and port

OpenFlow protocol

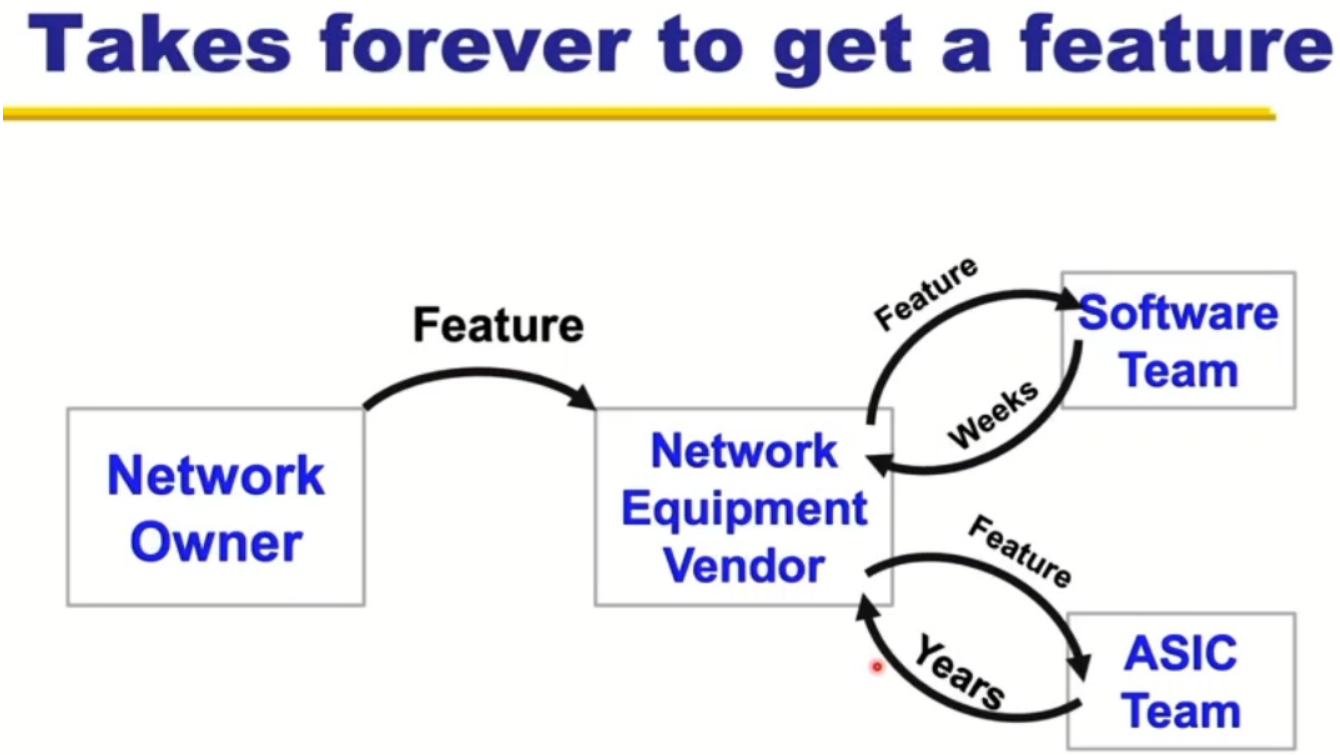
- Operates between controller, switch
- TCP used to exchange messages
  - Optional encryption
- Three classes of OpenFlow messages:
  - Controller-to-switch
  - Asynchronous (switch to controller)
  - Symmetric (misc.)



Programmable networks

Fixed-function data plane

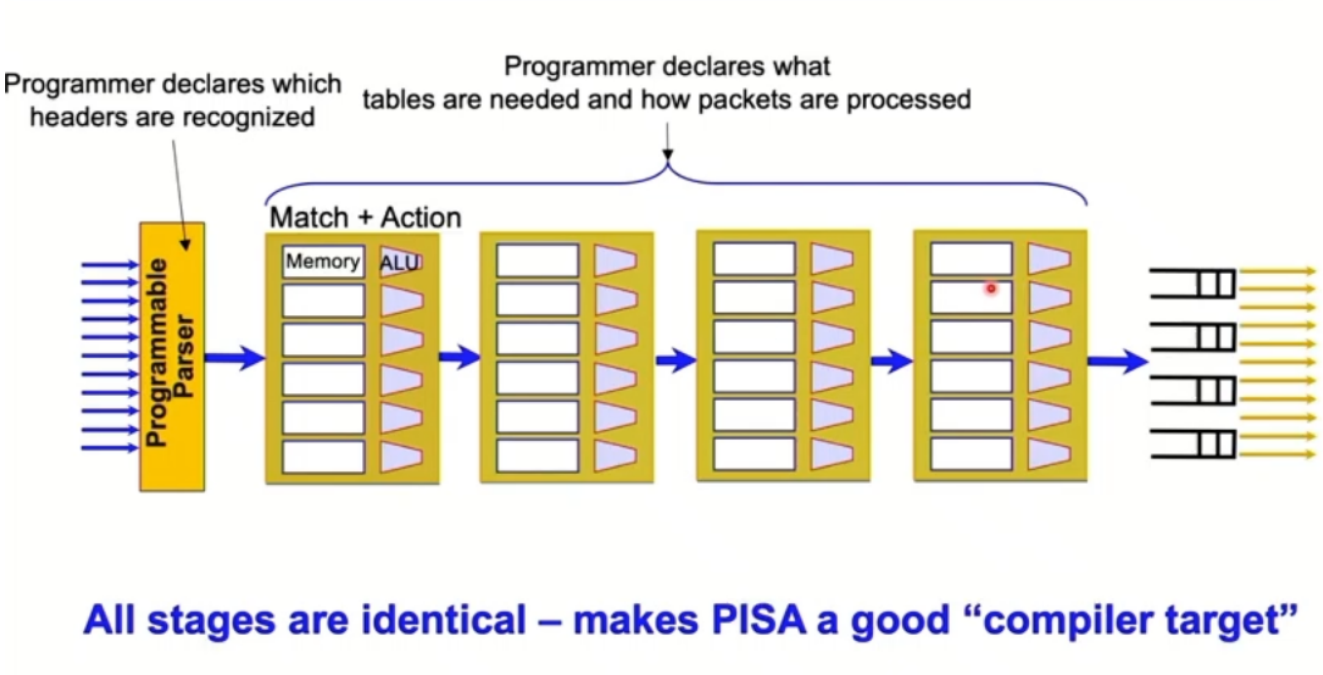
- Traditional switches are fixed-function
  - They can do whatever they can do at birth, but they cannot change!
  - Bottom-up design
- Even OpenFlow was designed to be a fixed protocol
  - With a fixed table format
  - Capable of doing limited things



Top-down approach

- Precisely specify what you want to do and how you want a packet to be processed
- Compile it down to be something runnable on a programmable switch
  - Similar to other high-level languages we use to run code on hardware
  - P4 for programming switches
    - Programming Protocol-Independent Packet Processors

PISA: Protocol Independent Switch Architecture(not ISP networks)

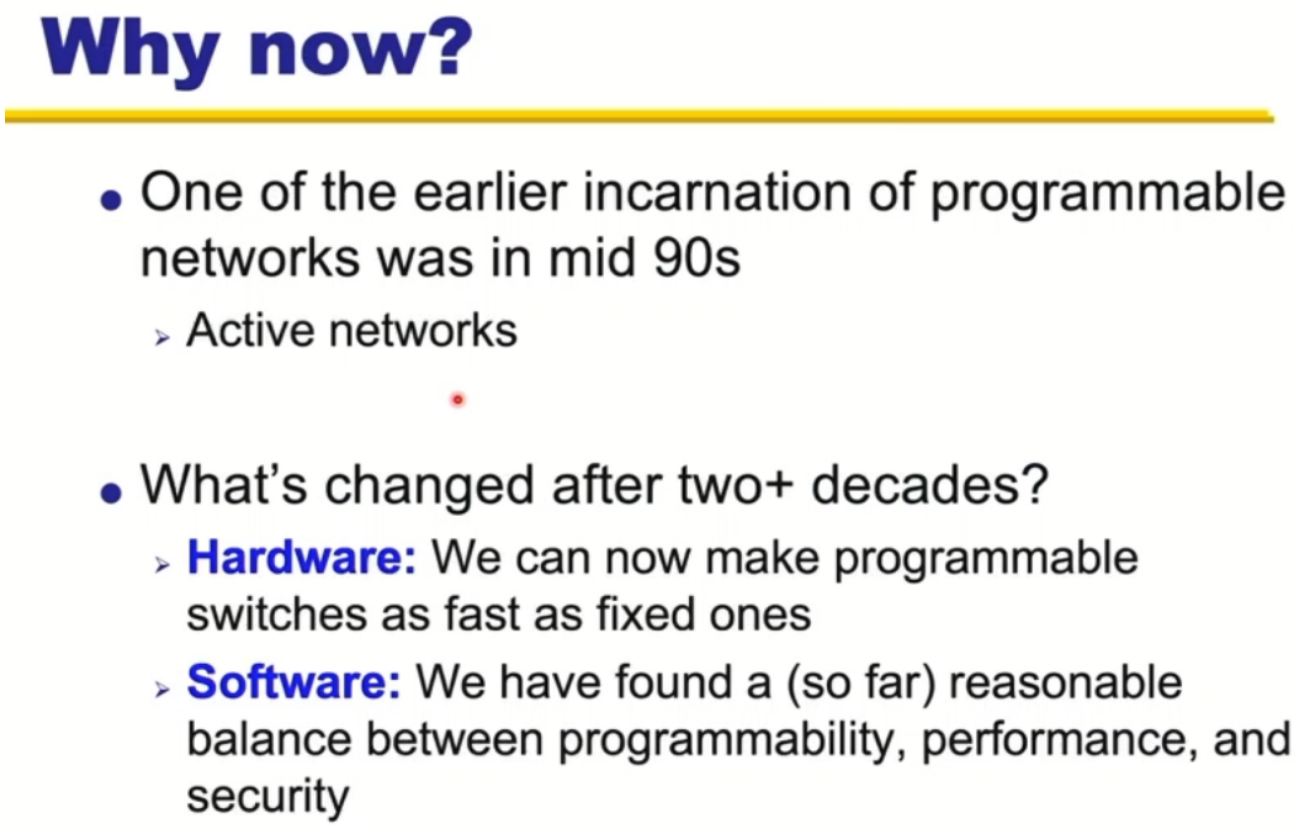


How is programmability used today

- Remove features to reduce complexity
- Add proprietary features
- Silicon independence or avoid vendor lock-in
- Telemetry and measurements

Example: In-band network telemetry (INT)

- Which path did my packet take
- Which rules did my packet follow
- How long did it queue at each switch
- Who did it share the queues with

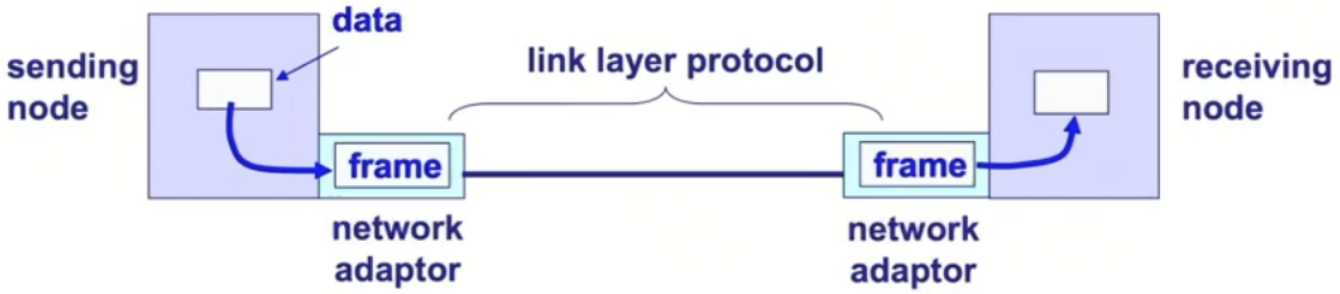


Layer 2

Data link layer

- Provides four primary services
  - **Framing**
    - Encapsulate network layer data
  - **Link access**
    - Medium access control, MAC, protocol defines when to transmit frames (avoid collision)
  - **Reliable delivery**
    - Primarily for mediums with high error rates (wireless) not end-to-end guarantee
  - **Error detection and correction**

"Packets" are "frames"



- Frames encapsulate network layer packets
- Link layer protocols are implemented in h/w
- Frame formats can change based on link layer protocol

point to point vs broadcast medium

- ptp: dedicated pairwise communication
    - Long distance fiber link
    - Point to point link b/n Ethenet switch and host
- Broadcast
    - share wire or medium
    - traditional ethernet
    - 802.11 wireless LAN

Multiple access algorithm

- Context: a shared broadcast channel
    - need to avoid multiple nodes speaking at once
    - need distributed algorithm to determine which node can transmit

Class of techniques

- Channel partitioning: divide channel into pieces
- Taking turns: scheme for deciding who transmits
- Random access: allow collisions then recover (modern way)
    - Carrier sense
    - Collision detection
    - Randomness
        - wait for random time

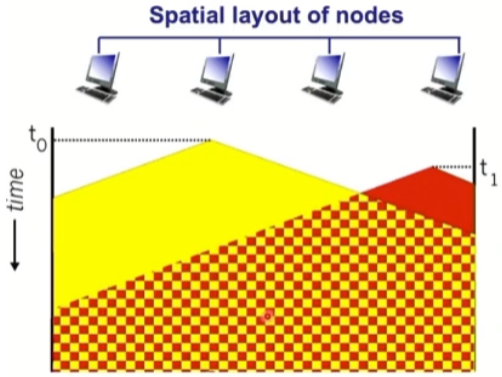
Random access

- When node has packet to send
  - Transmit at full channel data rate without coordination
- Two or more nodes -> collision, data lost
- Random access MAC protocol specifies
  - How to detect and recover from collisions
- Example
  - ALOHA Slotted ALOHA
  - CSMA, CSMA/CD(actively detect, more efficient when idle), CSMA/CA(avoidance, more efficient when many users)

CSMA (Carrier Sense Multiple Access)

- Listen before transmit
  - If idle, transmit entire frame
  - If busy, defer transmission
- Does not eliminate all collisions

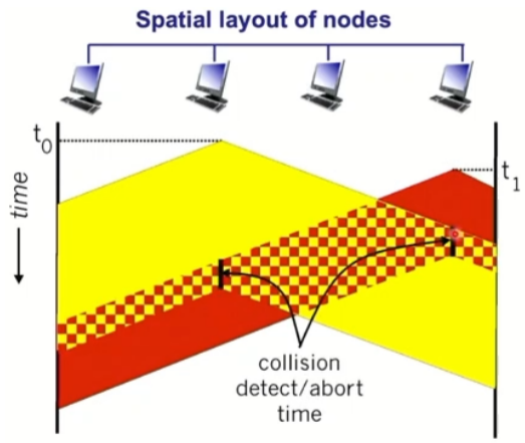
CSMA collisions

- Propagation delay: two nodes may not hear each other before sending
  - 

  - CSMA can reduce but cannot eliminate collisions
  - collision entire packet transmission time wasted

CSMA/ CS Collision Detection

- Carrier sensing, deferral as in CSMA
  - Collision detected within short time
  - Colliding transmission aborted, reduce waste

- Detection easy in wired/broadcast LANs
    - Compare transmitted, received signal
- Collision detection is difficult in wireless -- cannot listen and send as the same time
- 

- Need restrictions on minimum frame size and maximum distance
    - Cannot detect collision
    - Latency depends on a physical length of link
        - Time to propagate a frame from one end to other
    - Suppose A sends a frame at time t
        - B see idle line at a time just before t+latency d
    - B detects a collision, and sends jamming signal
        - But A cannot see collision until t +2d
    - A needs to wait for time 2d to detect collision
- Imposes restrictions; e.g., for 10 Mbps Ethernet
    - Maximum length of the wire: 2,500 meters
    - Minimum length of a frame: 512 bits (64 bytes)

Random access-binary exponential back-off

- After collision, wait a random time before retrying
- After m^th collision, choose K randomly from {0, ..., 2^m-1}
    - Wait for K*512 bit time before try again
    - if transmission occur when ready to send, wait until end of transmission

Efficiency of CSMA/CS

- Efficiency is defined as the long-run fraction of time during which frames are being transmitted without collision
- d_{prop} = max propagation time between two adapters
- d_{trans} = time to transmit a max-sized frame

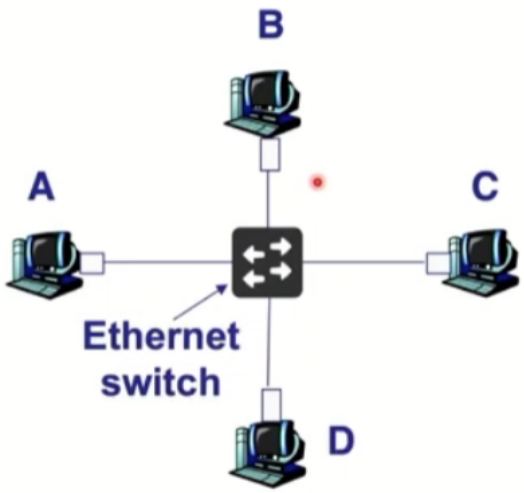
$$\text{Efficiency} \approx \frac{1}{1 + 5\, d_{prop} / d_{trans}}$$

- $d_{prop} \rightarrow 0$
  - Efficiency approaches 1
  - Colliding nodes abort immediately
- $d_{trans} \rightarrow \infty$
  - Efficiency approaches 1
  - Each frames uses the channel for a long time

$$\text{Efficiency} \approx \frac{d_{trans}}{d_{trans} + 5\, d_{prop}}$$

Why we have switched ethernet?

- Enables concurrent communication
  - Host A can talk to C, while B talks to D
  - No collisions and no need for CSMA/CD
  - No constraints on link lengths, etc.

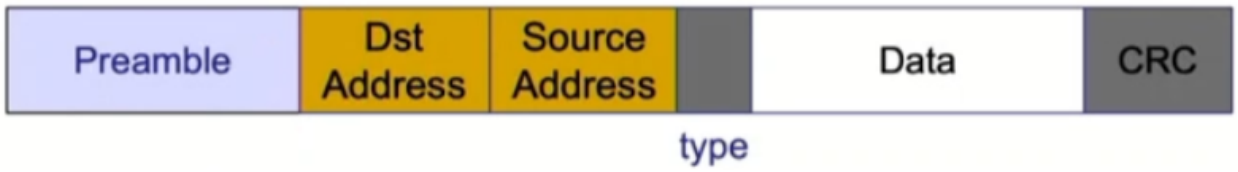


- Changed almost everything except the frame format
  - From the shared media coax cables to dedicated links
  - From 3 Mbit/s to 100 Gbit/s
  - From electrical signaling to optical
- Lesson: the right interface can accommodate many changes
  - Evolve the implementation while maintaining the interface (backward compatibility)

Ethernet Topic

- Invented as a broadcast technology
  - Hosts share channel
  - Each packet received by all attached hosts
  - CSMA/CD for media access control
- Modern Ethernets are "switched" (later)
  - Point to point links between switches and between a host and switch
  - No sharing -> no CSMA/CS
    - Use self learning and spanning tree for routing

Frames and framing

- Encapsulates IP datagram
- 

- Preamble: 7 bytes for clock synchronization and 1 byte to indicate start of frame
- Address: 6 bytes
- Type: 2 bytes, higher-layer protocol
- Data payload: max 1500 bytes, min 46 bytes

Framing frames

- Physical layer puts bits on a link
- but two hosts connected on same physical medium need to be able to exchange frames
    - Service provided by link layer
    - Implemented by network adapter
- Framing problem: how does the link layer determine where each frame begins and ends

Approachs:

Count bytes

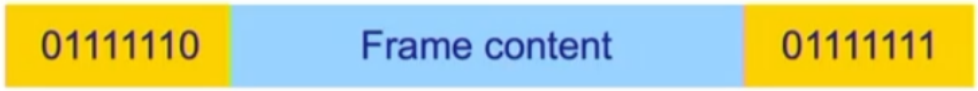
- Sender includes number of bytes in header
    - Receiver extracts this number of bytes of body
- What if the count field is corrupted? -GG, framing error
- 

    - CRC tells you to discard this frame, but what about next frame

Desynchronization & Resynchronize

- Framing with sentinel bits
    - Delineate frames with patter
    - 01111110 => start, 01111111=> end
- Bit stuffing

- Sender always inserts a 0 after five 1s in frame contents
- Receiver always remove a 0 appearing after five 1s



- If next bit 0, remove it; begin counting again
  - Because this must be a stuffed bit; we can't be at beginning/end of frame (those had six or seven 1s)
- If next bit 1 (i.e., we've seen six 1s) then:
  - If following bit is 0, this is start of frame
    - »Because the receiver has seen 01111110
  - If following bit is 1, this is end of frame
    - »Because the receiver has seen 01111111

Addressing

MAC address

- Numerical address associated with a network adapter
- Flat name space of 48 bits
- Unique, hard-coded in the adapter when it is built
- Hierarchical Allocation
  - Blocks: assigned to vendors by IEEE
    - First 24 bits
  - Adapter: assigned by the vendors from its block
    - Last 24 bits

MAC address vs. IP address

MAC Addresses

- Hard-coded when adapter is built
- Flat name space of 48 bits (e.g., 00-0E-9B-6E-49-76)
- Like a social security number
- Portable, and can stay the same as the host moves
- Used to get packet between interfaces on same network

IP Addresses

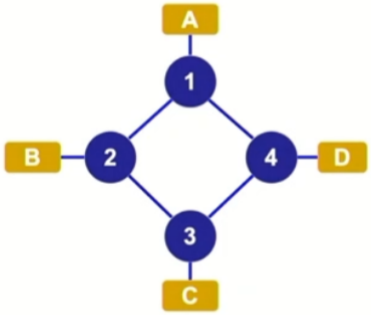
- Configured, or learned dynamically
- Hierarchical name space of 32 bits (e.g., 12.178.66.9)
- Like a postal mailing address
- Not portable, and depends on where the host is attached
- Used to get a packet to destination IP subnet

Routing

- Why ethernet don't use LS/DV -- scalability
  - MAC is flat and cannot be aggregate like IP
  - Legacy: maintain ethernet "plug-n-play"
    - Congestion -- too many packets
      - Solved by spanning tree!(Perlman's idea, eliminate loops in the topology)
- Routing with broadcast Ethernet
  - Sender transmits frame onto broadcast link
  - Each receiver's link layer passes the frame to the network layer
    - If destination address matches the receiver's MAC address OR if the destination address is the broadcast MAC address (ff:ff:ff:ff:ff:ff)
  - Ethernet is "plug-n-play"
  - A new host plug into ethernet without configuration by users or network operators, broadcast as a means of bootstrapping communication

Flooding still leads to loops

- Example: A wants to broadcast a message
  - A sends packet to 1
  - 1 Floods to 2 and 4
  - 2 Floods to B and 3
  - 4 Floods to D and 3
  - 3 Floods packet from 2 to C and 4
  - 3 Floods packet from 4 to C and 2
  - 4 Floods packet from 3 to D and 1
  - 2 Floods packet from 3 to B and 1
  - 1 Floods packet from 2 to A and 4
  - 1 Floods packet from 4 to B and 2
  - ....



- Broadcast storm still happens in a switched network if it contains a cycle of switches

Spanning tree approach (protocol by which bridges construct a spanning tree)

- Take arbitrary topology
- Zero configuration, self healing
- Pick subset of links that form a spanning tree
- From extended LAN to switched Ethernet

Switched Ethernet

- Constraints for backward compatibility
  - no changes to end hosts
  - maintain plug-n-play aspect
- Earlier Ethernet achieved plug-n-play by leveraging a broadcast medium

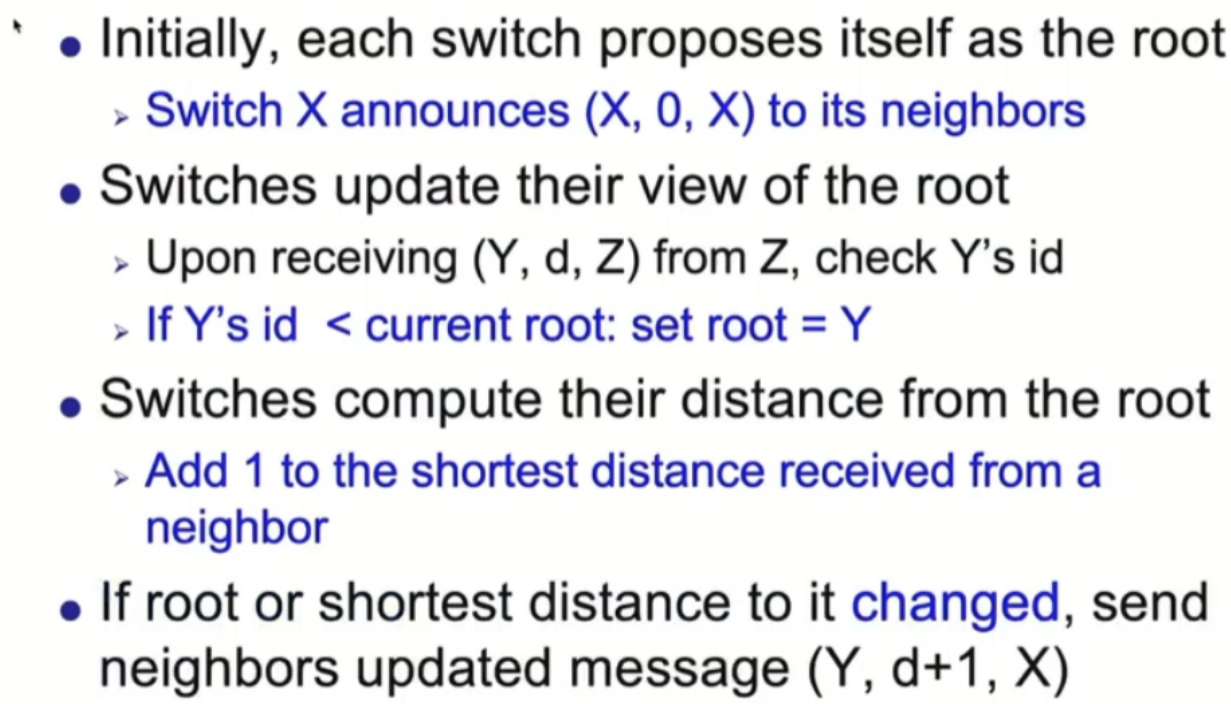
2 aspects

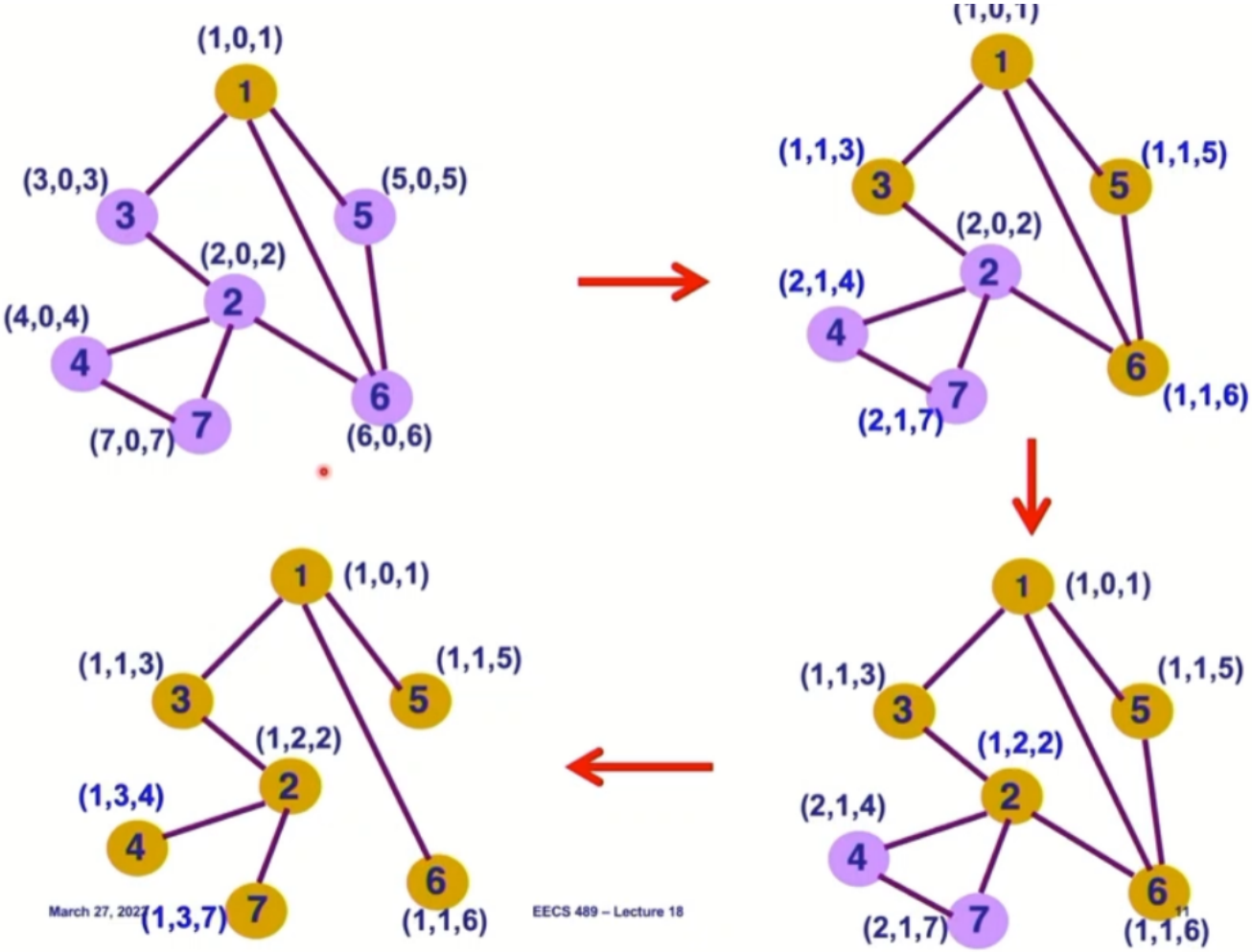
- Pick a root
  - Destination to which shortest paths go
  - Pick the one with smallest identifier (MAC addr)
- Compute shortest paths to the root
  - No shortest path can have a cycle
  - Only keep links on shortest-paths
  - Break ties in some way
    - When there are multiple shortest paths to the root, choose the path that uses the neighbor switch with the lower ID

- Ethernet's spanning tree construction does both with a single algorithm

Constructing a spanning tree

- Messages(Y, d, x)
  - From node X
  - Proposing Y as the root
  - Advertising a distance d to Y
- Switches elect the node with smallest identifier (MAC address) as root
- Each switch determines if a link is on its shortest path to the root; excludes it from the tree if not





Robust spanning tree algorithm

- Algorithm must react to failures
  - Failure of the root node
  - Failure of other switches and links
- Root switch sends periodic root announcement messages
  - Other switches continue forwarding messages
- Detecting failures through timeout
  - If no word from root, time out and claim to be the root

Forwarding

Flooding on a spanning tree

- Switches flood using the following rule:
  - (Ignore all ports not on spanning tree)
  - Originating switch sends packet out all ports

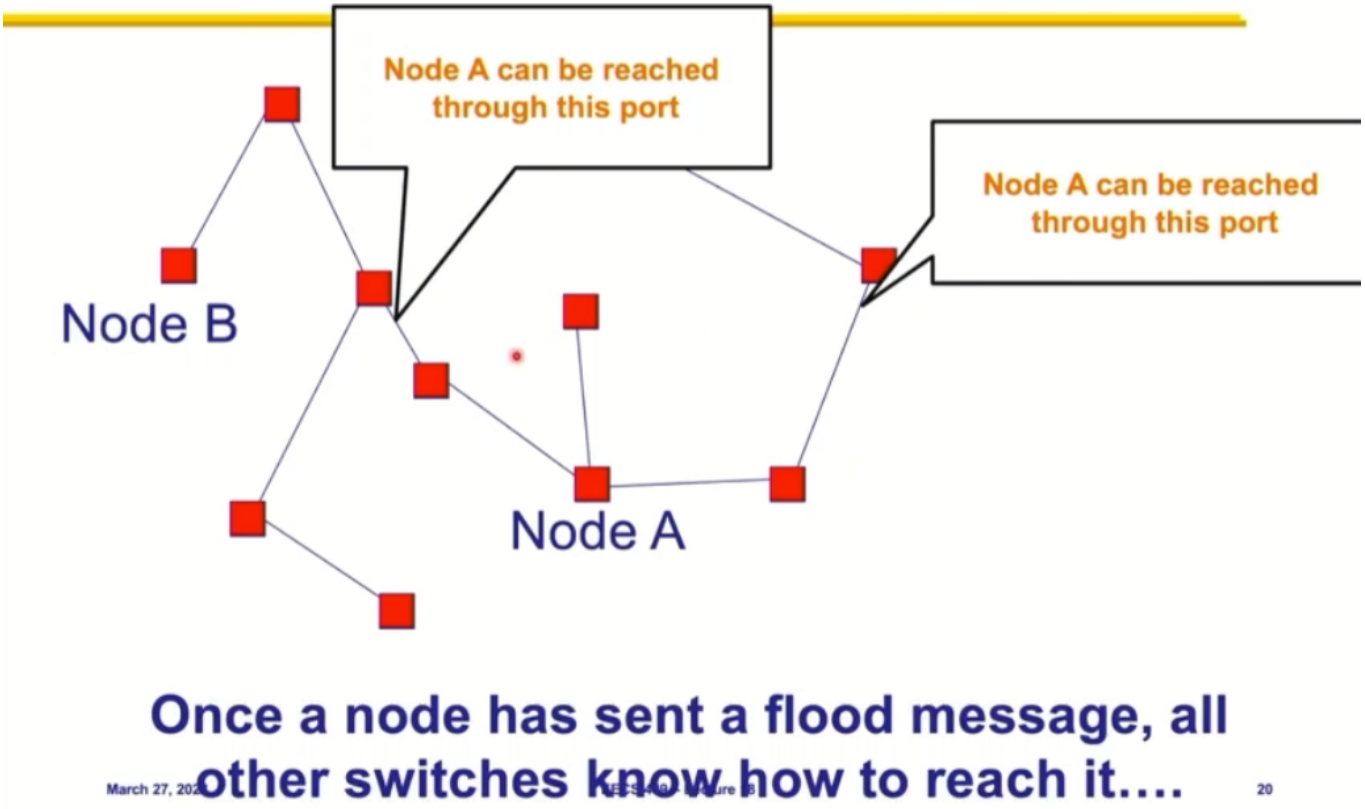
- When a packet arrives on one incoming port, send it out all ports other than the incoming port

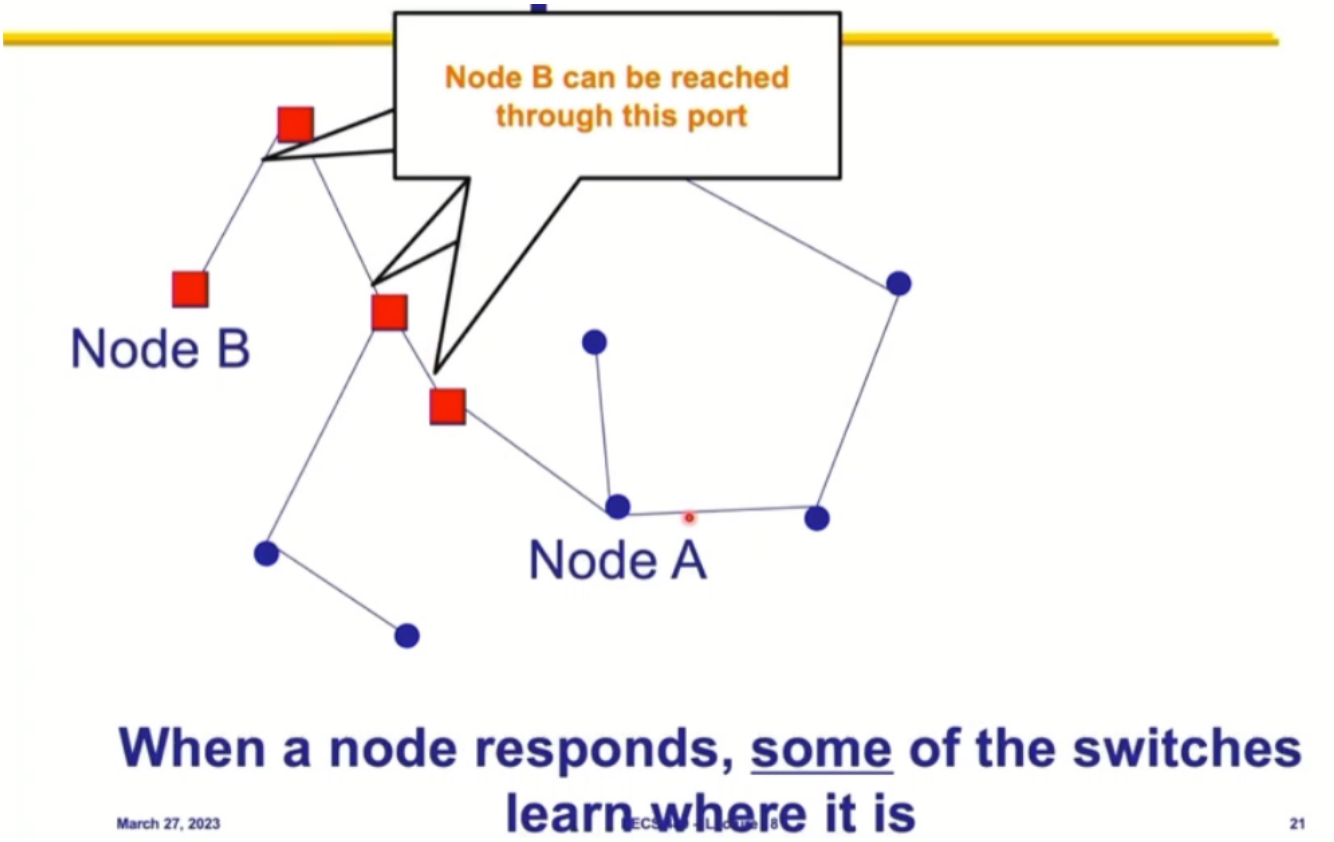
Flooding wasteful

- But we can use it to bootstrap more efficient forwarding
- **Idea: watch the packets going by, and learn from them**
  - **If node A sees a packet from node B come in on a particular port, it knows what port to use to reach B**
  - Works because there is only one path to B
- Nodes can "learn routes"
  - Switch learn how to reach nodes by remembering where flooding packets came from
    - If flood packet from Node A entered switch on port 4, then switch uses port 4 to send to Node A

General approach

1. Flood first packet to node you are trying to reach
2. All switches learn where you are
3. When destination responds, some switches learn where it is
    1. Only some switches, since packets follow direct path





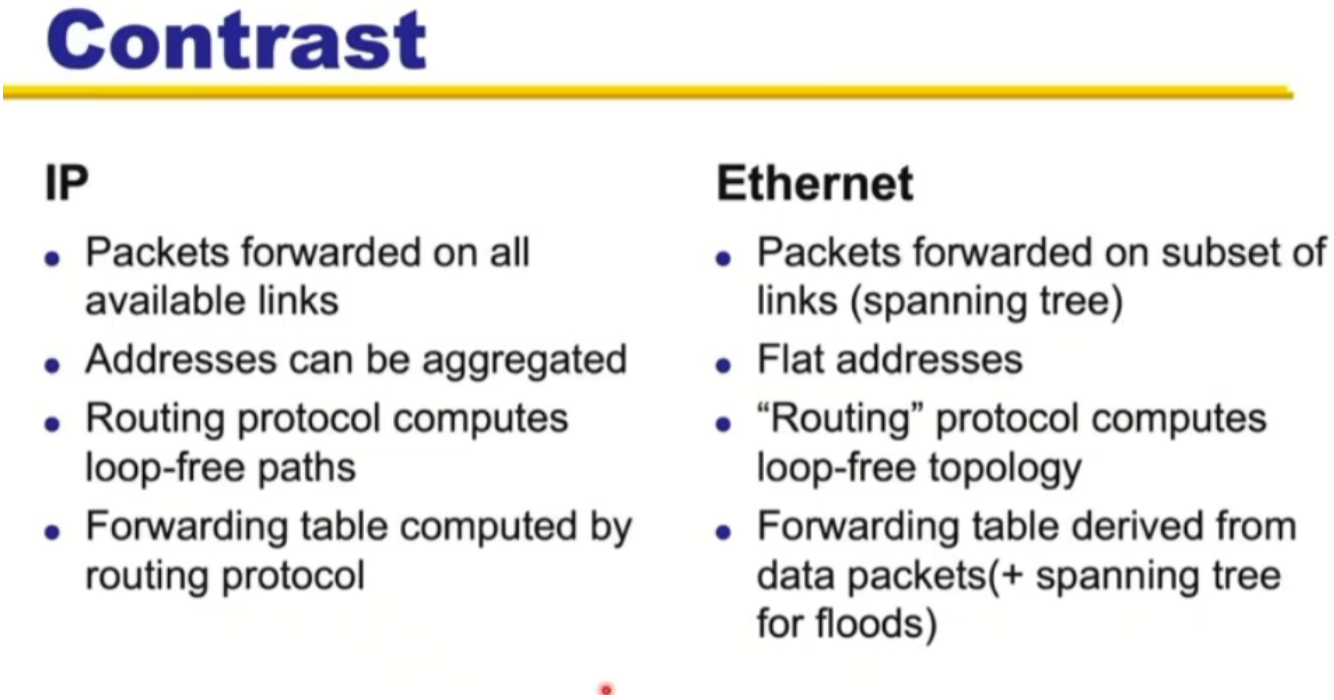
Ethernet switches are "self learning"

- When a packet arrives:
  - Inspect source MAC address, associate with incoming port
  - Store mapping in the switch table
  - Use **time-to-live** field to eventually forget mapping
- Handling misses:
  - When packet arrives with unfamiliar destination
  - Forward packet out all other ports
  - response may teach switch about the destination

Summary of learning approach

- Avoid loop by restricting to spanning tree
  - But flooding possible
- Flooding allows packets to reach destination and in the process switch learn how to reach source of flood
- No route computation
- Forwarding entries a consequence of traffic pattern

IP vs Enthernet



- Ethernet is smaller topology -- allow flooding
- Ethernet Pro and Cons
    - Pros
        - Plug-n-Play: zero-configuration
        - Simple
        - Cheap
    - Cons
        - Much of the network bandwidth goes unused (spanning tree)
        - Delay in reestablishing spanning tree (network down until spanning tree rebuilt)
            - Rebuilt might be different
        - Slow to reach to host movement
            - Entries must time out
        - Poor predictability
            - Location of root and traffic pattern determines forwarding efficiency

Discovery

- A host is "born" knowing only its MAC address
- Must discover lots of information before it can communicate with a remote host B
    - Its IP address, host's IP address
    - Host's MAC address if local
    - what is its first hop router's address if not local

ARP and DHCP

- Link layer discovery protocols
    - ARP -> Address Resolution Protocol
    - DHCP -> Dynamic Host Configuration Protocol
    - Confined to a single local-area network (LAN)
    - Rely on broadcast capability
- Serve two functions
    - Discovery of local end-hosts
        - For communication between hosts on the same LAN
    - Bootstrap communication with remote hosts

DHCP for IP

- Dynamic Host Configuration Protocol
    - Defined in RFC 2131
- A host uses DHCP to discover
    - Its own IP address
    - Its netmask
    - IP addresses for its local DNS name servers
    - IP addresses for its first-hop default routers

Operation

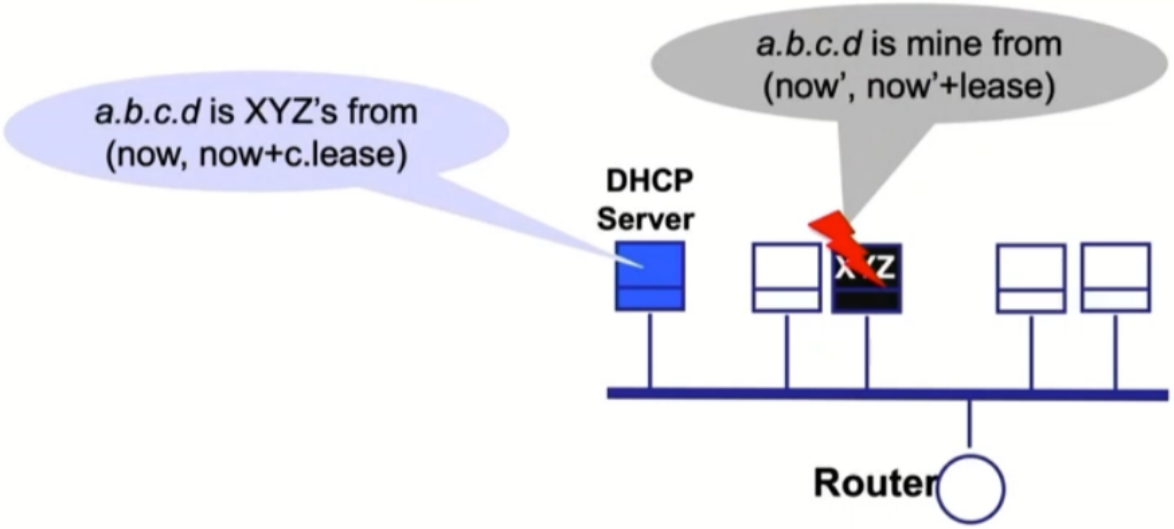
- One or more local DHCP servers maintain required information
    - IP address pool, netmask, DNS servers, etc
    - UDP poty 67 (reliable in local network)
- Client broadcast a DHCP discovery message
    - L2 broadcast: to the MAC address FF:FF:FF:FF:FF:FF

- One or more DHCP servers respond with a DHCP request message
- Client broadcasts a DHCP request message
    - Specifies with offer it wants
    - Echoes accepted parameters
    - Other DHCP servers learn they were not chosen
- Selected DHCP server responds with an ACK
- DHCP "relay agents" used when the DHCP server is not on the same broadcast domain

DHCP use "soft state"

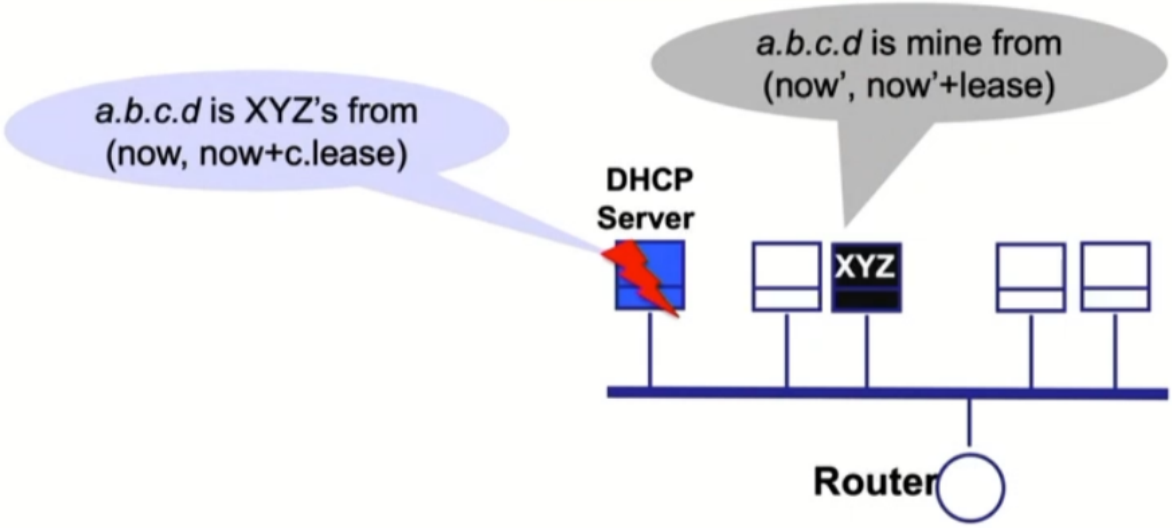
- Soft state: if not refreshed, state is forgotten
    - Hard state: allocation/revocation is deliberate
- Implementation:
    - Address allocation have a lease period
    - Server sets a timer for each allocation
    - Client must request a refresh before lease expires
    - Server resets timer when a refresh arrives and ACKs
        - Or reclaim allocated address when timer expires
- Simple, but robust under failure

Soft state under failure



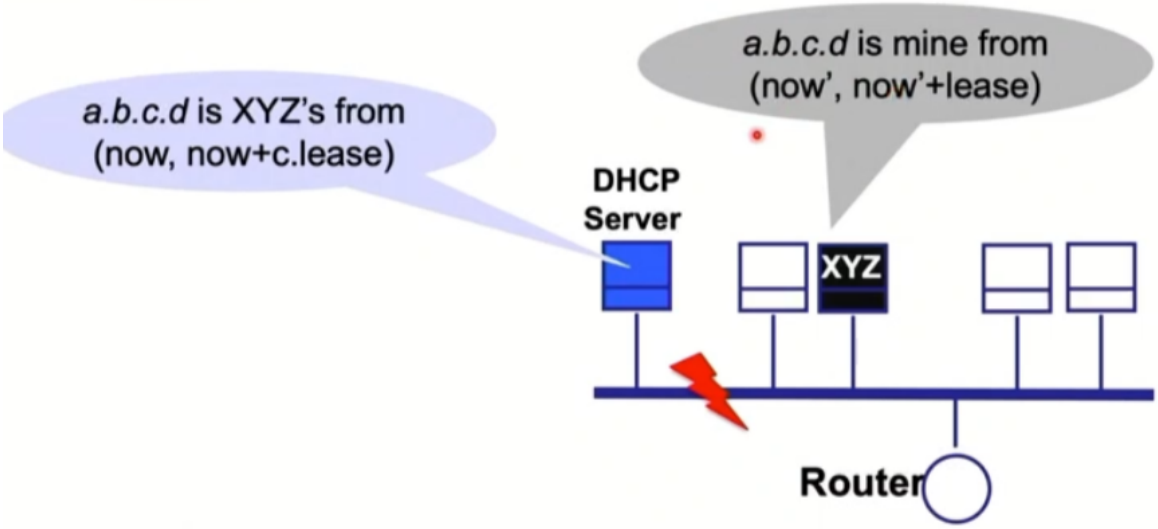
- What happens when host XYZ fails?
  - Refreshes from XYZ stop
  - Server reclaims a.b.c.d after O(lease period)

Soft state under failure



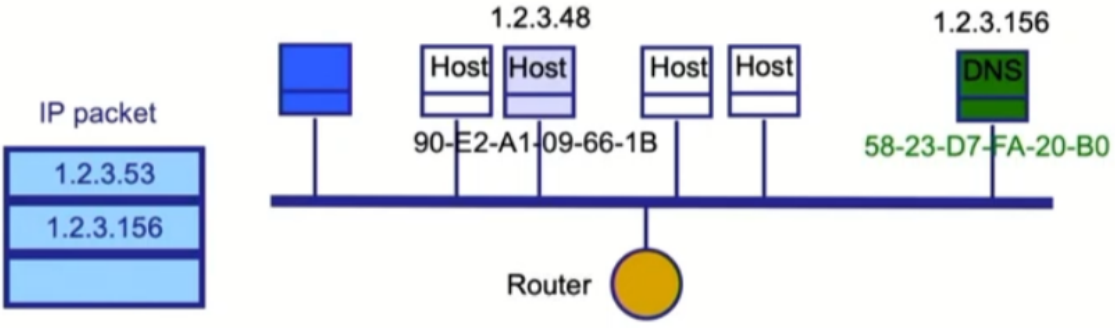
- What happens when server fails?
  - ACKs from server stop
  - XYZ releases address after O(lease period); send new request
  - A new DHCP server can come up from a `cold start' and we are back on track in ~lease time

Soft state under failure



- What happens if the network fails?
  - Refreshes and ACKs don't get through
  - XYZ release address; DHCP server reclaims it

Sending packets over link Layer



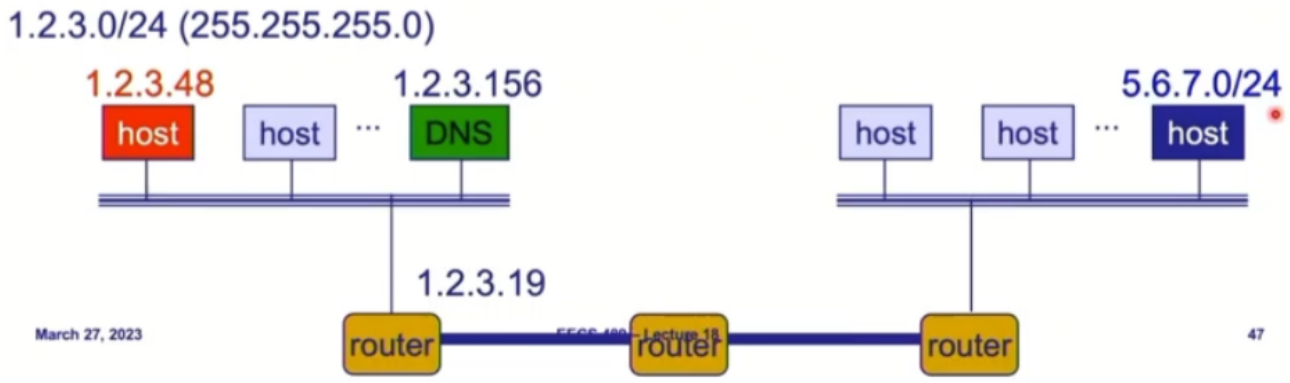
- Link layer only understands MAC addresses
  - Translate the destination IP address to MAC address
  - Encapsulate the IP packet in a link-level (Ethernet) frame

ARP: Address Resolution Protocol

- Every host maintain an ARP table
  - List of (IP address -> MAC address) pairs
- Consult the table when sending a packet
  - Map dest. IP address to dest. MAC address
  - Encapsulated IP data packet with MAC header, retransmit
- What if IP address is not in the table
  - Sender broadcasts: Who has IP address 1.2.3.156
  - Receiver replies: MAC address xxxxxx
  - Sender caches result in table

What if the destination is remote?

- Look up the MAC address of the first hop router
  - 1.2.3.48 uses ARP to find MAC address for first-hop router **1.2.3.19** rather than ultimate destination IP address
- How does the red host know the destination is not local?
  - Uses netmask (discovered via DHCP)
- How does the red host know about 1.2.3.19?
  - Also DHCP



Key ideas in both ARP and DHCP

- Broadcasting: Can use broadcast to make contact
  - Scalable because of limited size
- Caching: remember the past for a while
  - Store the information you learn to reduce overhead
- Soft state: eventually forget the past

ID resolution in the networking stack

Layer	Examples	Structure	Configuration	Resolution Service
App. Layer	cse.umich.edu	Organizational hierarchy	~ manual	DNS
Network Layer	123.45.6.78	topological hierarchy	DHCP	ARP
Link layer	45-CC-4E-12-F0-97	vendor (flat)	hard-coded	

Discovery mechanisms

- We have seen two approaches
  - Broadcast (ARP, DHCP)
    - »Flooding does not scale
    - »No centralized point of failure
    - »Zero configuration
  - Directory service (DNS)
    - »No flooding = scalable
    - »Root of the directory is vulnerable (caching is key)
    - »Needs configuration to bootstrap (local, root servers, etc.)

Wireless Network

Recap: Random access MAC protocols

- When node has packet to send
  - Transmit at full channel data rate without coordination
- Two or more transmitting nodes => collision
  - Data lost
- Random access MAC protocol specifies
  - **How to detect and recover from collisions**
  - **Detection is hard so we try to avoid collisions**
- Examples
  - ALOHA and Slotted ALOHA
  - CSMA, CSMA/CS, **CSMA/CA**(wireless)

Elements of a wireless network

Wireless hosts

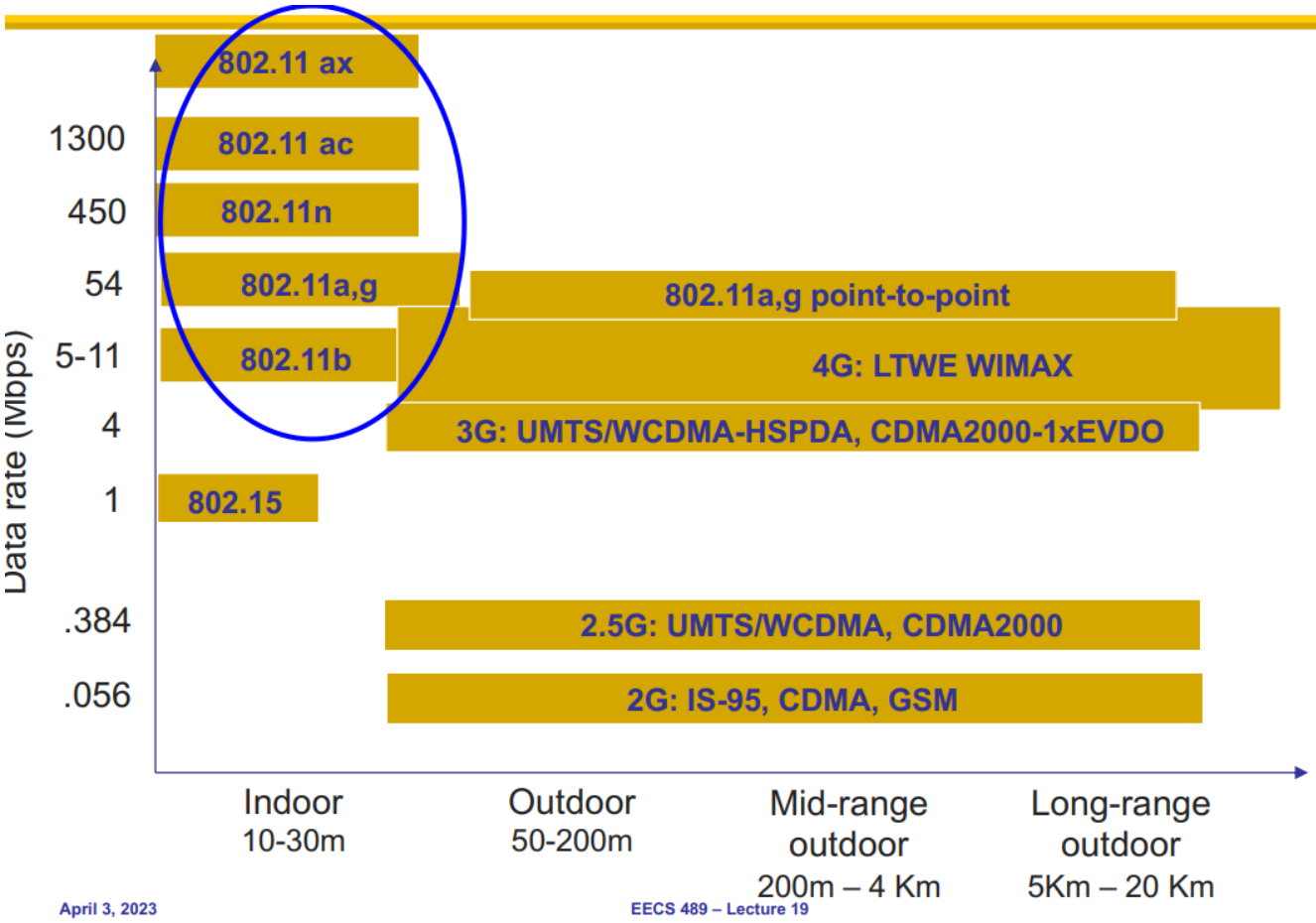
- Laptop, smartphone
- Run applications
- May be stationary or mobile
    - Wireless does not always mean mobility

Base station

- Typically connected to wired network
- Relay: responsible for sending packets between wired network and wireless host(s) in its "area"
    - **Access points (AP)**

Wireless link

- Typically used to connect mobiles to base station
- Also used as backbone link
- Multiple access protocol coordinates link access
- Various data rates, transmission distance
- **Frequency ↑ cover range ↓ speed ↑**

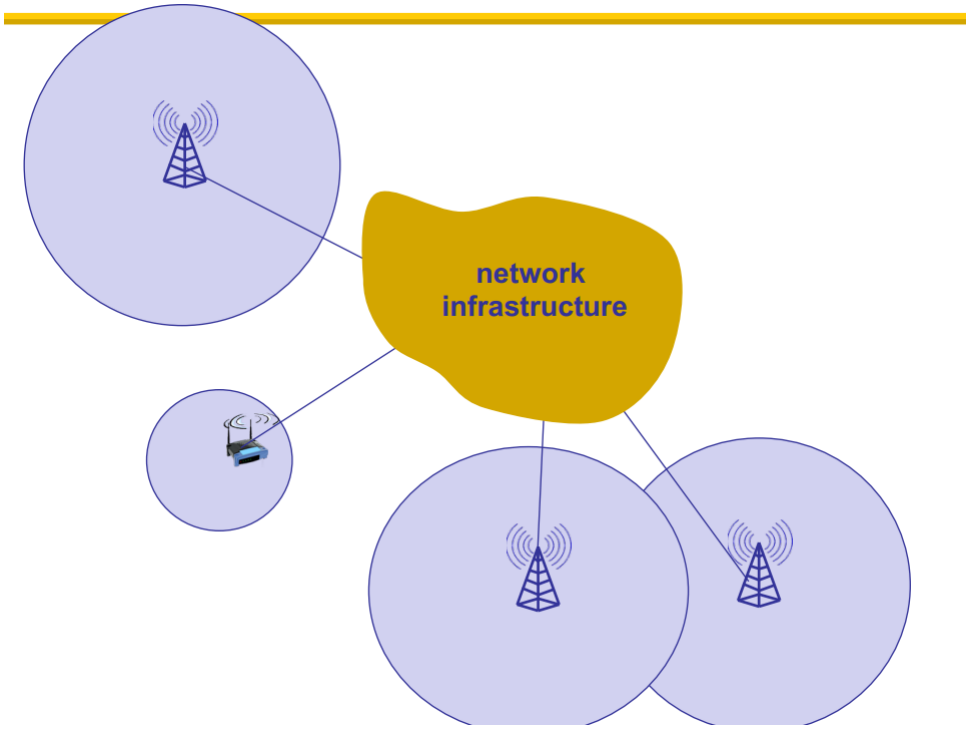


Wireless vs. mobile

- Wireless network deal with communication over wireless links
- Mobility deals with handling mobile users that change point of attachment to network
    - Non-wireless networks may also have to deal with mobility issues
    - **Handoff**: Mobile changes base station providing connection into wired network
        - vertical: inside same network
        - horizontal: between different network technology(5G to LTE)
        - This usually interrupt your connection

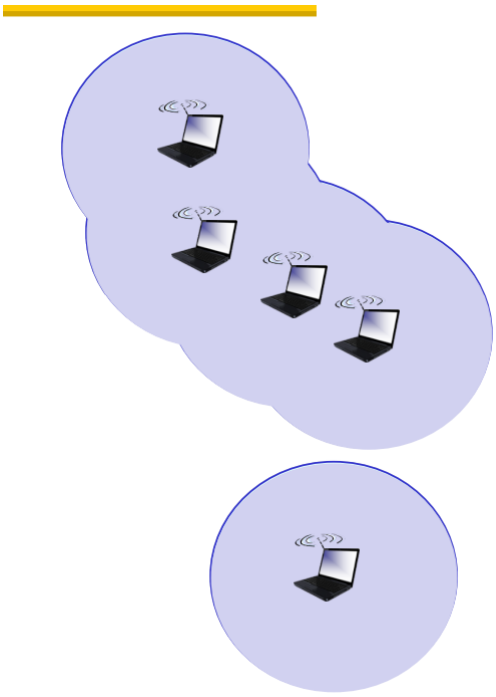
Two modes of operation

Infrastructure mode: Base stations connect mobiles to wired network



Ad-hoc mode(pair to pair): Wireless hosts organize themselves to communicate

- No base station
- Nodes can only transmit to others within link coverage
- Nodes organize themselves into a network: route among themselves



Wireless network taxonomy

	Single hop	Multiple hops
Infrastructure (APs)	Host connects to base station(Wifi, WiMAX, cellular), which connects to larger internet	Host may have to relay through several wireless nodes to connect to larger Internet: mesh net. That is, have infrastructure set up, and host can connect to out of range host in ad-hoc mode for communication
No infrastructure	No base station, no connection to larger Internet(Bluetooth, ad hoc nets)	No base station, no connection to larger Internet. May have to relay to reach other a given wireless node MANET, VANET -- only within the domain group range

Wireless link characteristics

- Three important difference from wired link ..

Decreased signal strength: radio signal attenuates as it propagates through matter(path loss)

Path loss/path attenuation

- Free Space Path Loss (FSPL):

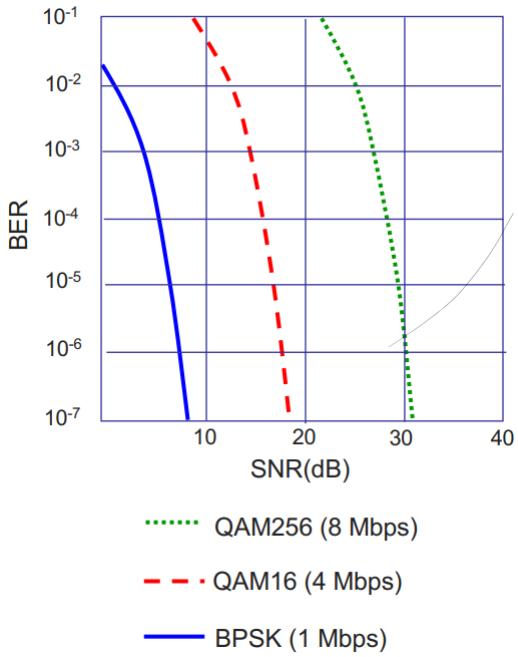
$$FSPL = \left(\frac{4\pi df}{c}\right)^2$$

  - d = distance
  - λ = wave length (c/f)
  - f = frequency
  - c = speed of light
- Due to
  - Reflection, diffraction, absorption, terrain contours (urban, rural, vegetation), humidity

SNR and BER

- **SNR**: Signal-to-noise ratio
  - Smaller SNR means cannot communicate in high ratio
  - **Larger SNR makes it easier to extract signal from noise(good)**

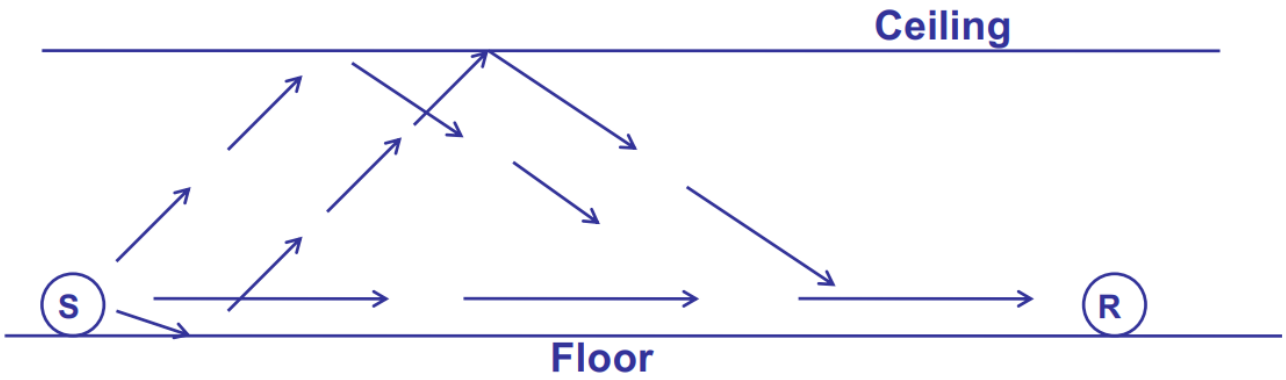
- **BER**: Bit error rate, how frequent bit corrupted
- SNR vs BER tradeoffs
    - **Given physical layer**: Increase power -> Increase SNR -> decrease BER
    - **Given SNR**: Choose physical layer that meets BER requirement, giving highest throughput
    - **SNR may change with mobility**: Dynamically adapt physical layer
        - SNR goes up when you close to base station



Dealing with bit errors

- Wired vs. wireless links: most loss due to congestion vs. higher, time-varying BER
- Dealing with high wireless bit-error rates
    - **Sender could increase transmission power**
        - Needs high energy(bad for batter-powered hosts)
        - Creates more interference with other senders
    - **Stronger error detection and recovery**
        - More powerful error detection/correction codes
        - Link-layer retransmission of corrupted frames
    - **Many TCP alternatives/extensions for wireless**
        - TCP Westwood uses Explicit Loss Notification (ELN)

Multipath propagation: radio signal reflects off objects ground, arriving at destination at slightly different times -- go to different path

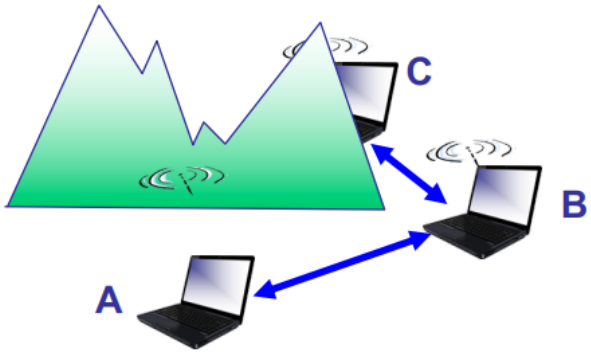


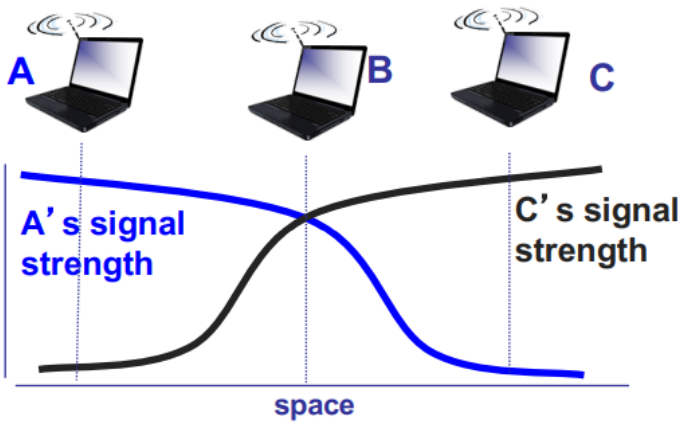
- Signal bounce off surface and interfere with one another
- Self-interference

Interference from other sources: Standardized wireless network frequencies (e.g. 2.4 GHz) shared by other devices (e.g. phone); devices(motors) interfere as well

Wireless network characteristics

- Broadcast medium
  - Anybody in proximity can hear and interfere
- Cannot receive while transmitting
  - Our own transmission is **deafening** our receiver => **Half-duplex**
  - Recent work has shown that full duplex is possible
- Signals sent by sender don't always end up at receiver intact
- Multiple wireless senders and receivers create many problems
  - Multiple access issues
  - Hidden terminal problem -- 2 sending sending at the same time
    - A, C cannot hear each other so they are not aware of their interference at B
    - B cannot interpret signal since both A and C arriving same time
    -





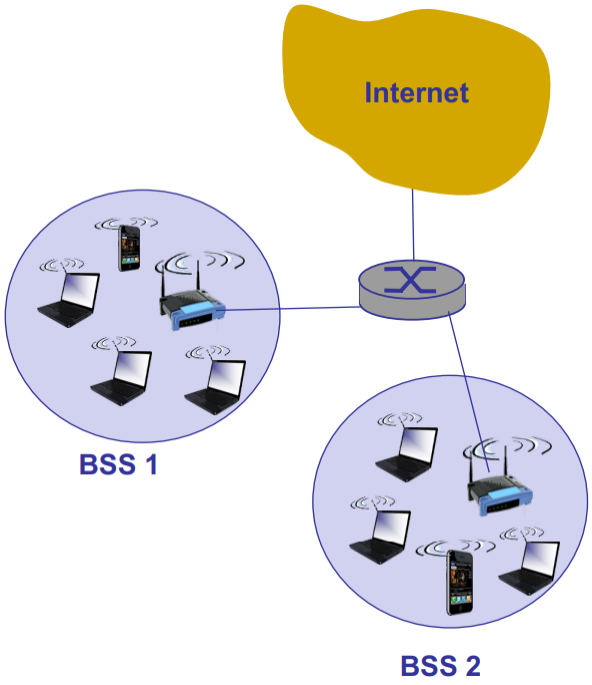
802.11 wireless LAN (WiFi)

- Many variations
  - 802.11b, 802.11a, 802.11g, 802.11n, 802.11a
- All use CSMA/CA for multiple access
- All have infrastructure and ad-hoc modes
  - 5G: higher rate communication, smaller range, more base station

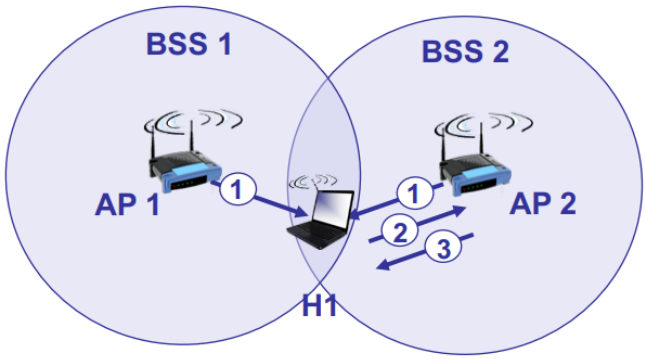
802.11 LAN architecture

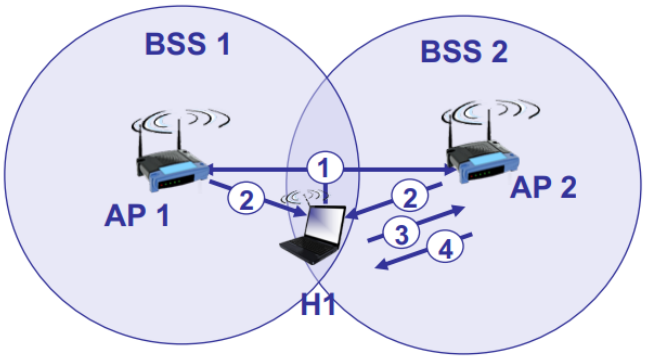
- Wireless host communicates with base station
  - Base station = AP
- Basic Service Set(BSS), or "cell", in infrastructure mode contains
  - Wireless hosts
  - AP: base station
- Ad-hoc mode: hosts only
- Designed for limited area
- AP is set to specific channel
  - Broadcast beacon messages with SSID(Service Set Identifier) and MAC Address periodically

- Hosts scan all the channels to discover the AP's
    - Host associates with AP



802.11: Passive/active scanning





Passive scanning

1. Beacon frames sent from APs
2. Association Request frame sent: H1 to selected AP
3. Association Response frame sent from selected AP to H1

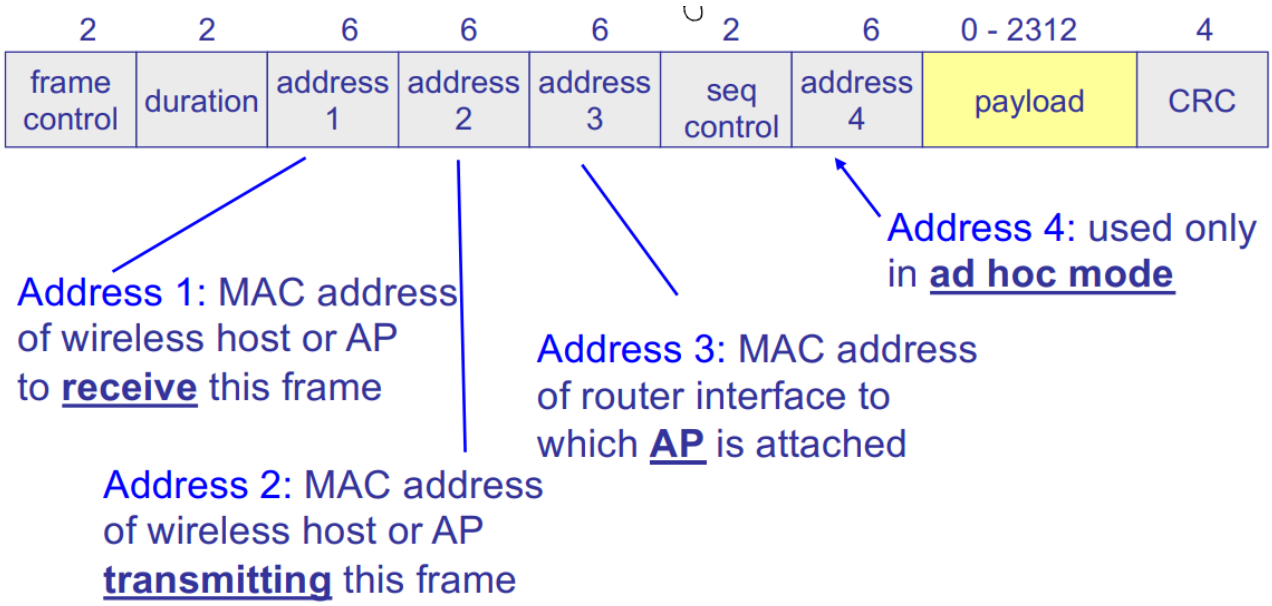
Active scanning

1. Probe Request frame broadcast from H1
2. Probe Response frames sent from APs
3. Association Request frame sent from H1 to selected AP
4. Association Response frame sent from selected AP to H1

802.11 multiple access

- 802.11 CSMA: sense before transmitting
  - Don't collide with ongoing transmissions by others
- 802.11 has no collection detection
  - Difficult to receive (sense collisions) when transmitting due to weak received signals (fading)
  - Can't sense all collisions in any cases: hidden terminal, fading
- Avoid collisions: CSMA/ CA
  - CA: Collision Avoidance

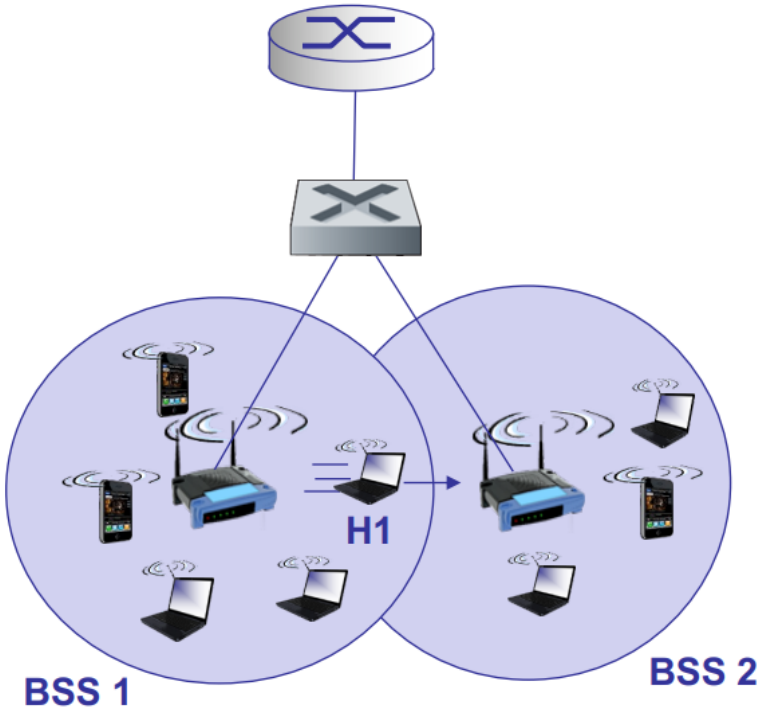
802.11 frame: addressing



- Address 3 is used when sending respond back to the router, it will need AP's MAC addr to fill in the 802.3 frame

802.11: Mobility within same subnet

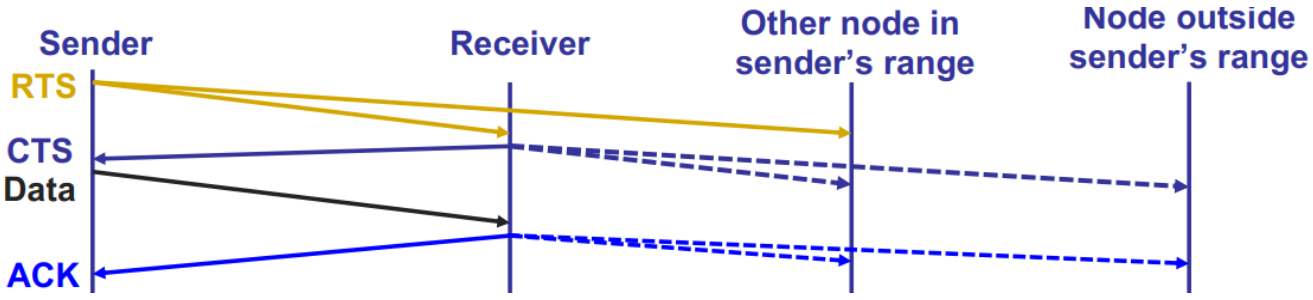
- H1 remains in same IP subnet: IP address can remain same
- Switch: which AP is associated with H1?
  - Self-learning: Switch will see frames from H1 and remember which switch port can be used to reach H1



Basic collision avoidance

- Carrier sense:
  - When medium busy, choose random interval
  - Wait that many idle timeslots to pass before sending
- When a collision is interred, retransmit with binary exponential backoff (like Ethernet)
  - Use ACK from receiver to infer "no collision"
  - Use exponential backoff to adapt contention window

CSMA/CA



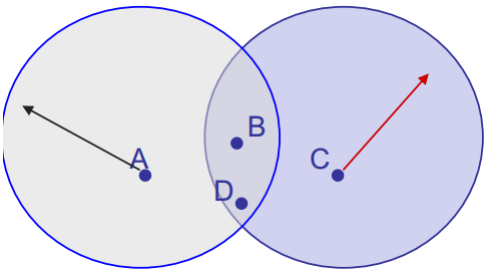
- Before every data transmission
  - Sender sends a Request to Send(RTS, do you wanted to talk, if so, talk for how long) frame with the length of transmission and the destination

    - Receiver respond with a Clear to Send(CTS) frame
    - Sender sends data
    - Receiver sends an ACK
- If sender doesn't get a CTS back, it assumes collision and wait to give some time

RTS/ CTS

- Works by reserving the channel using short frames before transferring much longer DATA frame
    - **Explicitly reserving the channel** enables avoidance
- Required to avoid hidden terminals
    - Hidden terminals will hear CTS from the receiver

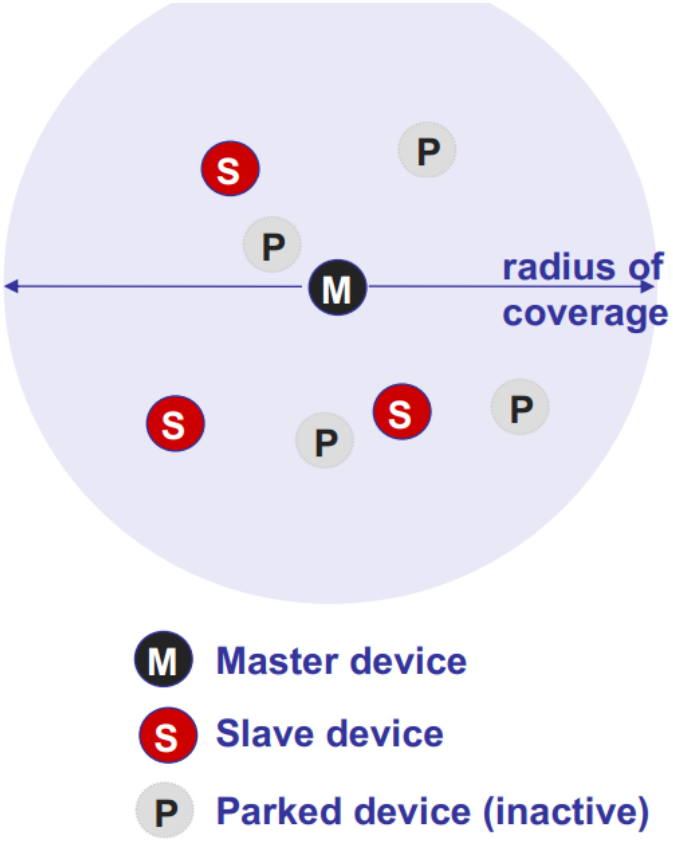
Preventing collisions altogether

- Frequency Spectrum partitioned into several channels
    - Nodes within interference range can use separate channels
    - E.g. A and C agree to use separate channel communicating B and D
    - 

    - Aggregate Network throughput doubles

802.15: Personal area network

- Evolved from Bluetooth specification
- Less than 10 m diameter
- Replacement for cables (mouse, keyboard, headphones)
- Ad-hoc: no infrastructure
- Master/slaves:
    - Slave request permission to send to master
    - Master grants requests



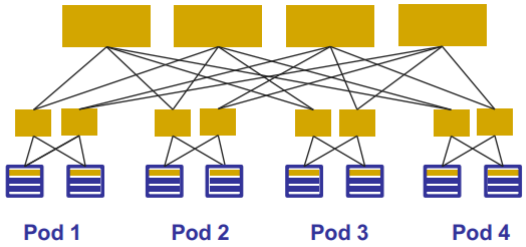
Summary

- Wireless networking introduces more challenge than wired networks
  - Interference, attenuation, multipath, hidden terminals
- CSMA/CS doesn't work because collision detection is difficult
  - Instead, CSMA/CA is used that avoid collisions by reserving the channel a priori

Datacenter network

- Datacenter network requirements
  - High "bisection bandwidth" -- max bandwidth of child link
  - Low latency, even in the worst-case
  - Large scale => power is the limit but not room
  - Low cost => resilience to failure
- Clos topology -- full bisection bandwidth
  - Multi-stage network
  - k pods, where each pod has two layers of k/x switches

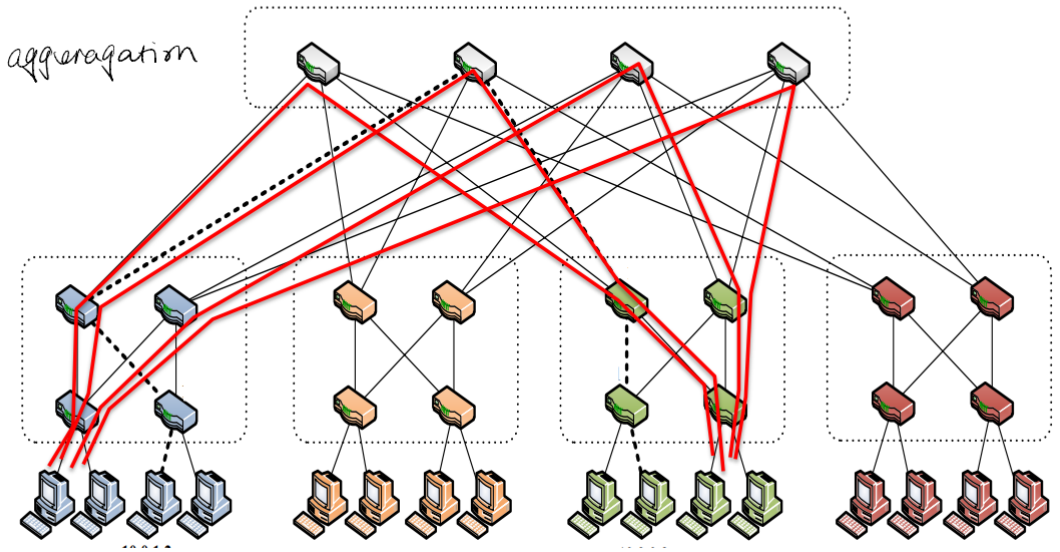
        - k/2 ports up and k/2 down
    - All links have the same b/w
    - At most (k^3)/4 machines
    -



Networking in modern datacenters

- L2/L3 design
    - Addressing/routing/forwarding in the Fat-Tree
- L4 design
    - Transport protocol design (with Fat-Tree)
- L7 design
    - Exploiting application-level information(With Fat-Three)

Using multiple paths well



L2/L3 design

Goals

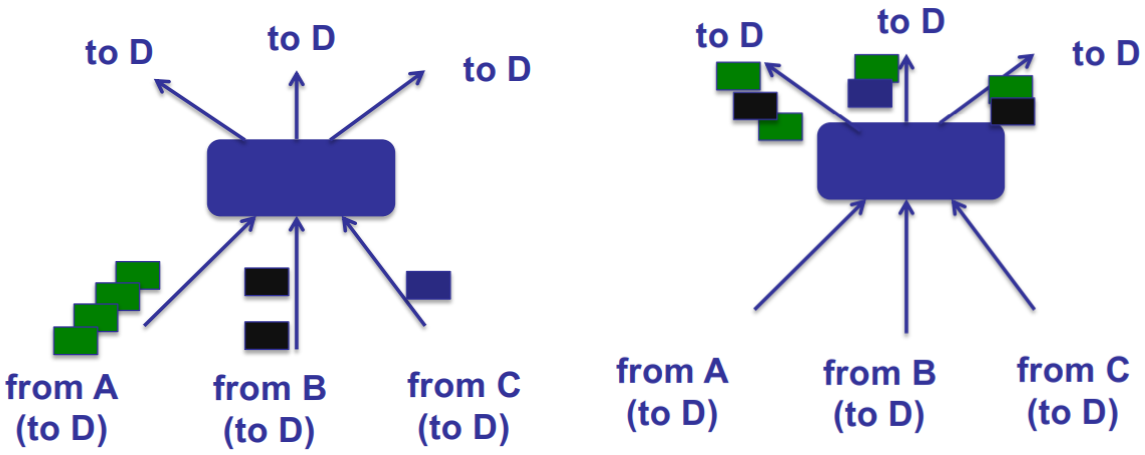
- Routing protocol must expose all available paths => control plane, logic
- Forwarding must spread traffic evenly over all paths => data plane, use multiple path

Extended DV/LS ?

- Routing
  - DV: remember all next-hops that advertise equal cost to a destination
  - Link-State: extend dijkstra's to compute all equal cost shortest paths to each destination
- Forwarding: how to spread traffic across next hops => if same cost, which paths we should choose

Forwarding

Per-packet load balancing

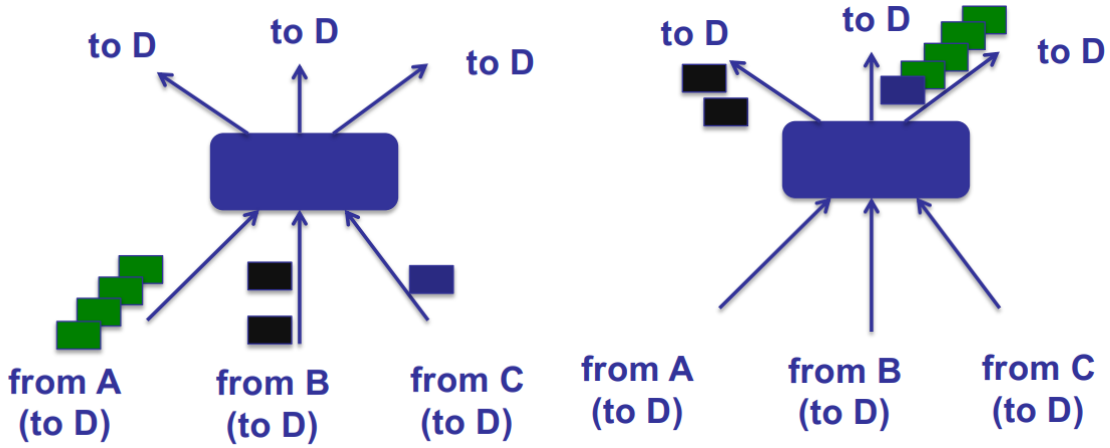


- Traffic well spread even with elephant(big) flows
- But TCP interact poorly => TCP doesn't like out of order traffic, that will cause packet loss and retransmit
-

TCP w/ per-packet load balancing

- Consider
  - Sender sends seq#: 1,2,3,4,5
  - Receiver receives: 5,4,3,2,1
  - Sender will enter fast retransmit, reduce CWND, retransmit #1, …
  - Repeatedly!
- Information sharing between multiple paths affects TCP
  - One RTT and timeout estimator for multiple paths
  - CWND halved when a packet is dropped on any path

Per-flow load balancing (Equal cost multi path)

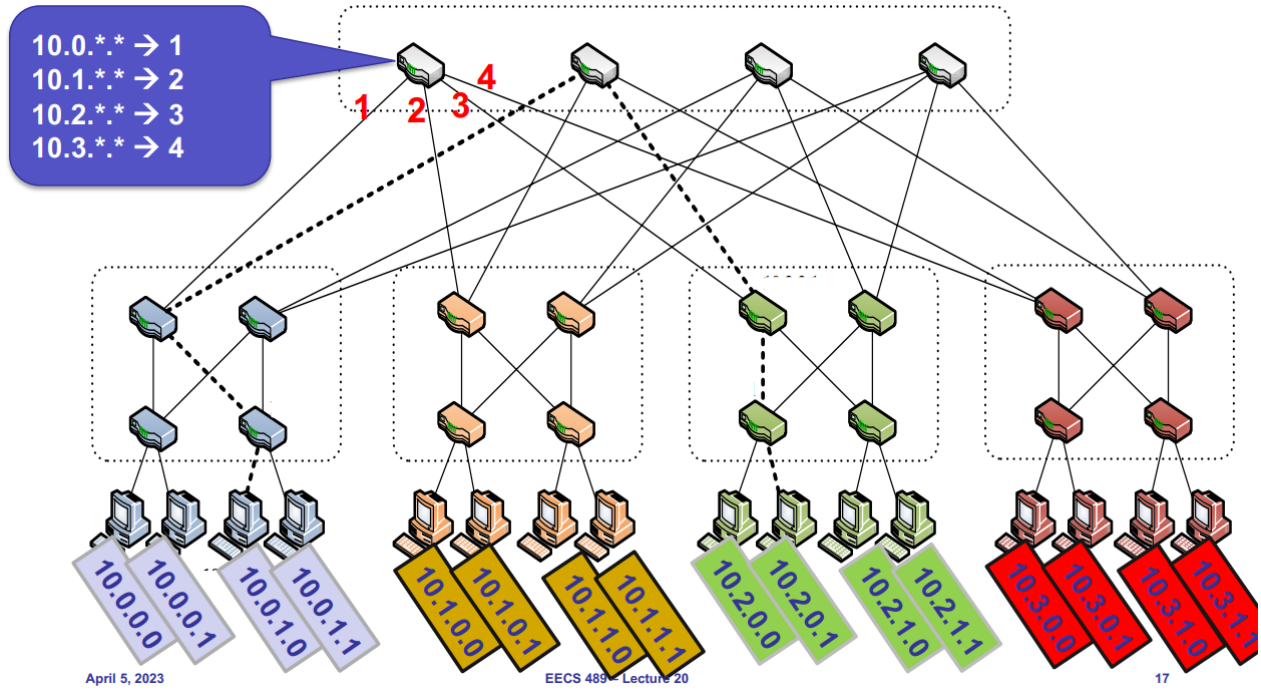


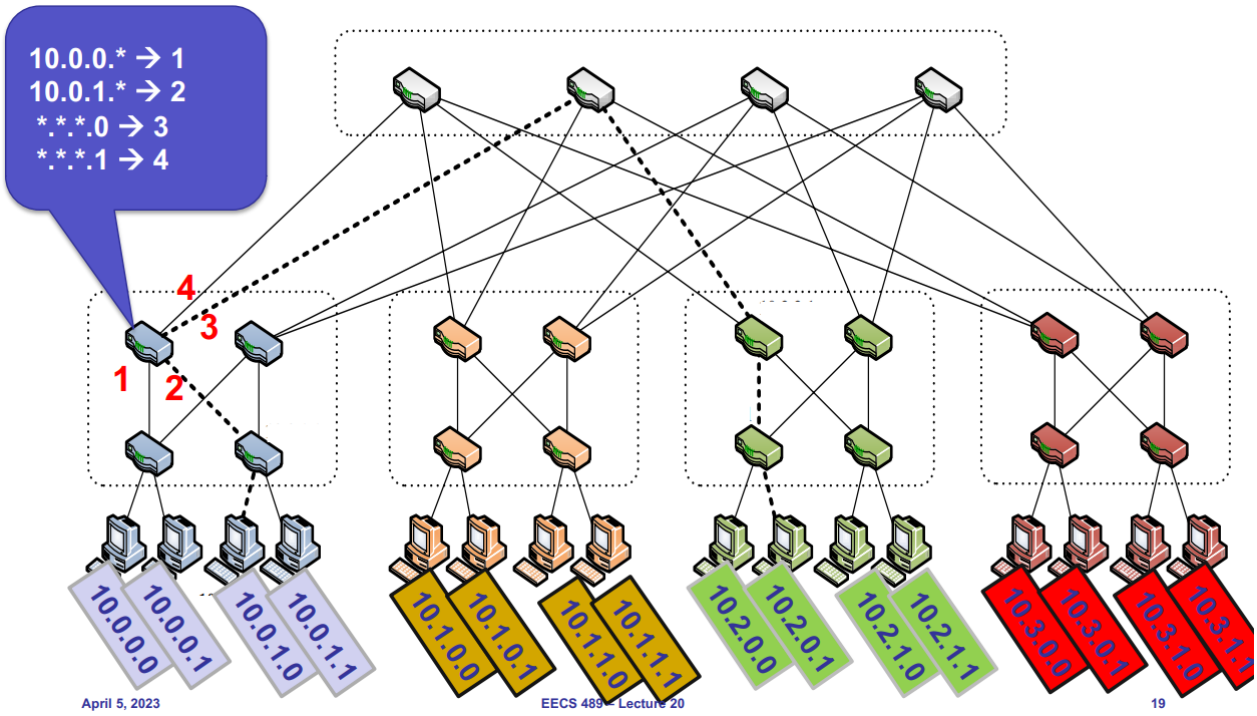
- A flow follows a single path (do well with TCP)
- hash header and get output to determine which out port to use => deterministic hash give deterministic result
- suboptimal load-balancing: elephant flow is problem

Extended DV/LS ?

- How:
  - Simple extensions to DV/LS
  - ECMP for load balancing
- Benefits:
  - Simple
  - Reuse of existing solution
- Problem: poor scaling
  - N dest, O(N) routing entries and messages
  - N is in millions
  - **solution => aggregation**

Solution 1: Topology-aware addressing





- Addresses embed location in regular topology
- Maximum #entries/switch: k (=4 in example)
  - Constant, independent of #destinations
- No route computation / message / protocols
  - Topology is hard-coded, but still need localized link failure detection
- Problems
  - **VM migration: ideally, VM keeps its IP address when it moves**
  - **Vulnerable to topology and addresses misconfiguration**

Solution 2: Centralize + source routes (controller design)

- Centralized "controller" server knows topology and compute routes
- Controller hands servers all paths to each destination
    - O(#destinations) state per server, but server memory cheap (e.g., 1M routes x 100B/route = 100 MB)
- Server inserts entire path vector into packet header
    - E.g., header = [dst=D | index =0 | path={s5, s1, s3,s7}]
- Switch forwards based on packet header
    - index++, next-hop = path[index]
- No #entries per switch
- #routing messages akin to a broadcast from controller to all servers
- Pro:
    - switches very simple and scalable
    - flexibility: end-points control route selection
- Cons:
    - Scalability / robustness of controller since only one controller and it cannot fail
    - Clean-state design of everything

L4 design

Workloads

- Partition-Aggregate traffic from user-facing queries
    - Numerous short flows with small traffic footprint
    - **Latency-sensitive**
- Map-Reduce traffic from data analytics
    - Comparatively fewer large flows with massive traffic footprint
    - **Throughput-sensitive**

Tension between requirements

- High throughput
    - Deep queues at switches
        - Queueing delays increase latency

- Low latency
  - Shallow queues at switches
    - Bad for bursts and throughput
- Objective
  - **Low queue occupancy and high throughput**

Data Center TCP(DCTCP)

- Proposal from Microsoft Reasearch
  - Incremental fixes to TCP for DC environments
- Leverages Explicit Congestion Notification(ECN)
- Mark the packet so that receiver can react
- React early, quickly, and with certainty using ECN
- React in proportion to the extent of congestion, not its presence
-

ECN Marks	TCP	DCTCP
1 0 1 1 1 1 0 1 1 1	Cut window by 50%	Cut window by 40%
0 0 0 0 0 0 0 0 0 1	Cut window by 50%	Cut window by 5%

- 6/10 bits are set. α = 0.6. Hence the window is cut by α/2 = 0.6/2 = 0.3 → 30%
-

Actions due to DCTCP

- At the switch
  - If instantaneous queue length > k
    - »Set ECN bit in the packet
- At the receiver
  - If ECN bit is set in a packet, set ECN bit for its ACK
- At the sender
  - Maintain an EWMA of the fraction of packets marked (α)
  - Adapt window based on α: W ← (1 – α/2) W
  - α = 1 implies high congestion: W ← W/2 (like TCP)
-

DCTCP: Why it works

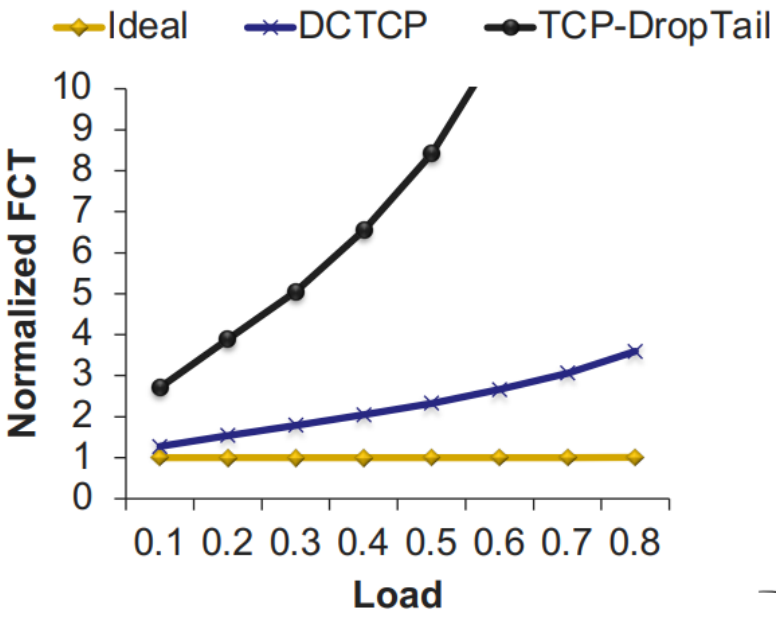
- React early and quickly: use ECN
  - Avoid large buildup in queues → lower latency
- React in proportion to the extent of congestion, not its presence
  - Maintain high throughput by not over-reacting to congestion
  - Reduces variance in sending rates, lowering queue buildups
- Still far from ideal

What's ideal for a transport protocol?

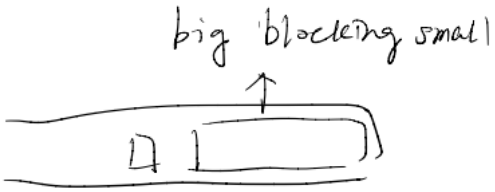
- When the flow is completely transferred?
- Latency of each packet in the flow?
- Number of packet drops?
- Link utilization?
- Average queue length at switches?

Flow Completion Time (FCT)

- Time from when flow started at the sender, to when all packets in the flow were received at the receiver



  - Drop tail is drop if the queue is full
- Queues are still shared => head-of-line blocking
  -



Solution: use priorities

- Packets carry a single priority number
  - Priority = remaining flow size
- Switches
  - very small queues (e.g., 10 packets)
  - send highest-priority/ drop lowest-priority packet
- Servers
  - Transmit/retransmit aggressively (at full link rate)
  - Drop transmission rate only under extreme loss (timeouts)
- Provides FCT close to the ideal

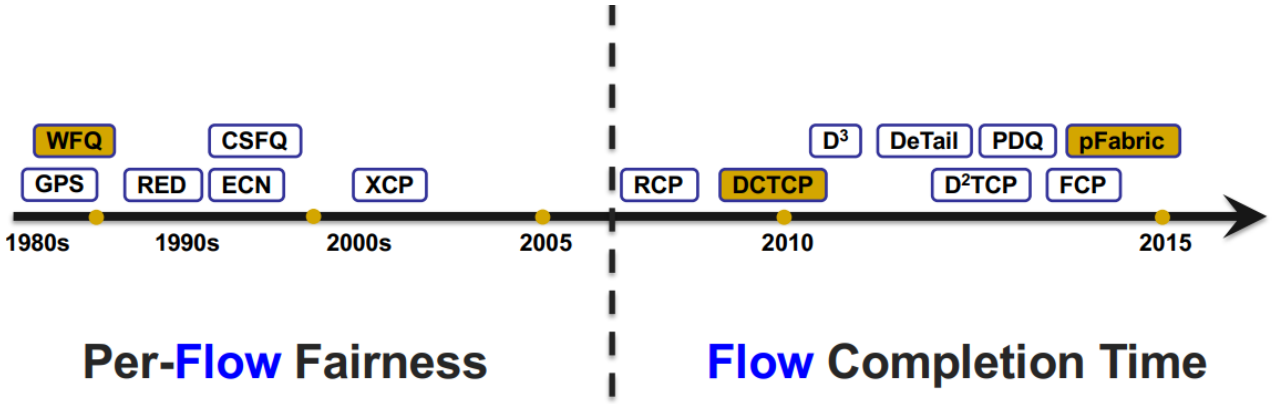
Multipath TCP

- Multipath TCP (MPTCP) is an ongoing effort to extend TCP to coexist with multipath routing
  - Value beyond datacenters (wifi and 4G co-work)
  - Boost performance and use more power

L7 design

- Map-reduce: a communication stage cannot complete until all its flows have completed
    - If there is a really slow job, prioritize that

Flow-based solutions

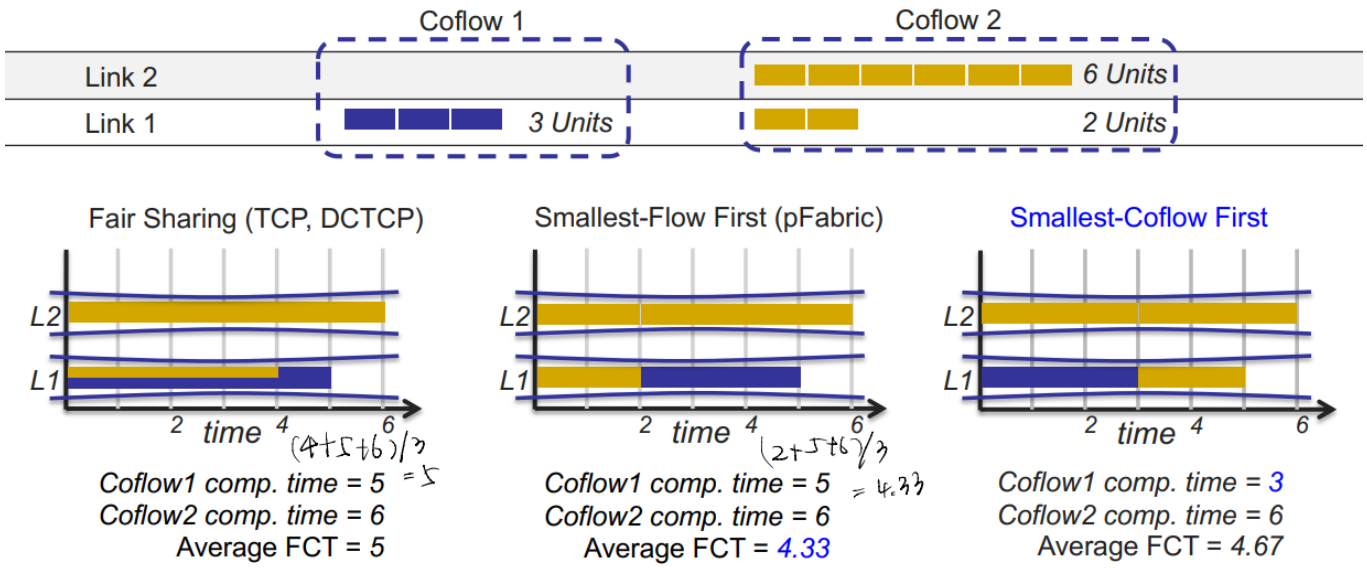


- We need to know which flow is slow
- Independent flows cannot capture collective communication patterns that are common in data parallel applications

The coflow abstraction

- Coflow is a communication abstraction for data-parallel applications to express their performance goals
    - Minimize completion times (most common way)
    - Meet deadlines or Perform fair allocation
- Not for individual flows; for entire stages

Benefits of inter-coflow scheduling



Coflow completion time(CCT) is a better predictor of job-level performance than FCT(local level metric)

How to implement coflows?

- Modify applications to annotate coflows
  - Possible to infer them as well [SIGCOMM'16]
- Managed communication
  - Applications do not communicate; instead, a central entity does the communication on their behalf
- Centralized scheduling