Chapter 1:

Internet

General

- Billions of connected computing devices:
  - Hosts=end systems
  - Running network apps at Internets "edge"
- Packet switches: forward data as packages (chunks of data)
- Communication links
  - Fiber, copper, radio, satellite
  - Transmission rate = bandwidth
- Networks: Collection of devices, rooters and link (managed by organisation)
- Internet = Network of Networks – Interconnected ISPs
- Protocols: control sending, receiving
  - HTTP (S)
  - TCP
  - IP,…
- *Internet standards:*
  - *RFC: Request of Comments*
  - *IETF: Internet Engineering Task Force*

Services

- Infrastructure that provides services to applications – SOCKETS(=Interfaces)
- Provides programming interface to distributed applications
  - Hooks allowing sending/receiving
  - API = Basis für Programmierung

Protocols

- Definition: define the format, order of messages sent and received among network entities, and actions taken on message transmission, receipt

Network edge

- -> hosts (clients and server)
- Network core:
  - Interconnected routers
  - Network of networks
- Access networks:
  - Cable-based access:
    - Frequency division multiplexing (FDM): different channels transmitted in different frequency bands
    - HFC: hybrid fiber coax
    - Network of cable – fiber attaches homes to ISP router
    - DSL: digital subscriber line
      - Use existing telephone line to central office DSLAM
  - Host sends packets of data
    - Takes application message
    - Breaks into smaller chunks known as packets of length L (bits)
    - Transmits packet into access network at transmission rate R
    - Packet transmission delay (time)=L/R
  - Links: physical media
    - Bit: propagates between transmitter/receiver pairs

- Physical link
- Guided media (copper, fiber, coax)
- Unguided media (radio)
- Wired Types
  - Twisted Pair (TP): Two insulated Copper wires
  - Coaxial Cable
    - Two concentric copper conductors
    - Bidirectional
  - Fiber optic cable:
    - Glass fiber carrying light pulses
    - High speed operation
- Wireless Radio
  - WLAN
  - WAN
  - Bluetooth
  - Terrestrial microwave
  - Satellite

Network core:

- Mesh of interconnected routers
- Packet-switching: host break application-layer messages into packets
- Network forwards packets from one router to next until at destination
- Two key network core functions:
  - Forwarding – local: move arriving packets from router input link to appropriate router output link
  - Routing – global: determine source-destination paths taken by packets

Packet switching

- Store-and-forward:
  - Packet transmission delay: L/R seconds to transmit L-bit packet into link at R bps
  - Store and forward: entire packet must arrive at router before it can be transmitted on next link
- Queueing
  - Occurs when work arrives faster than it can be serviced
  - If arrival rate to link exceeds transmission rate of link for some period of time:
    - Packets will queue, waiting to be transmitted on output link
    - Packets can be dropped if memory (buffer) is full
- Alternative to packet switching – circuit switching:
  - End-end resources allocated to, reserved for “call” between source and destination
  - Commonly used in traditional telephone networks
  - Types:
    - Frequency Division Multiplexing (FDM)
      - Division into frequency bands
      - Each call allocated own band
    - Time Division Multiplexing (TDM)
      - Time divided into slots
- Packet switching vs circuit switching:
  - Circuit switching less than packets switching maximum users
  - Packet switching great for bursty data
  - Excessive congestion possible: packet delay and loss

Internet structure

- Hosts connect to Internet via ISPs

NET – Summary

- ISPs must be interconnected
  - ANY host should be able to contact ANY other host
- Connect all ISPs with each other? NO
- Connect local ISPs to global ISPs that connect to each other (last one with internet exchange points IXP)

Performance

- Packets queue in router buffers waiting for their turn
- Packet loss occurs when memory to hold queued packets fill up
- Packet delay: $d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$
  - D_proc: nodal processing
    - Check bit errors
    - Determine output link
  - D_queue: queueing delay
    - Time waiting at output link for transmission
    - Depends on congestion level of router
  - D_trans: transmission delay
    - L/R
  - D_prop: propagation delay
    - D… length of physical link
    - S… propagation speed
    - D/S
- Packet queueing delay
  - α: average packet arrival rate
  - $\frac{L*\alpha}{R} = \frac{\text{arrival rate of bits}}{\text{service rate of bits}}$ …. Traffic intensity
  - ~0… average queueing delay small
  - ->1… average queueing delay large
  - >1 … more work arriving than can be serviced – average delay infinite!
  - Traceroute = program – provides delay measurement from source to router along end-end internet paths to destination
- Packet loss
  - Lost packet may be retransmitted by previous node, by source end system or not at all
- Throughput: rate at which bits are being sent from sender to receiver
  - Instantaneous: rate at given point in time
  - Average: rate over longer period of time

Security

- Internet not originally designed with security in mind
- Packet interception:
  - Packet sniffing: Network interface reads/records all packets
- Fake identity:
  - IP spoofing: injection of packet which false source address
- Denial of Service (DoS):
  - Attackers make resources unavailable to legitimate traffic by overwhelming resource with bogus traffic
- Lines of defence:
  - Authentication
  - Confidentiality: encryption
  - Integrity checks: signatures
  - Access restrictions: password-protected VPNs
  - Firewalls

Protocol layers, service models

- Networks consist of many pieces:
  - Hosts
  - Routers
  - Links of various media
  - Applications
  - Protocols
  - Hardware, software
- Why layering?
  - Explicit structure allows identification, relationship of system's pieces
  - Modularization eases maintenance
- Layered Internet Protocol Stack
  - Application:
    - Supporting network applications
    - HTTP, IMAP, SMTP, DNS
  - Transport:
    - Process data transfer
    - TCP, UDP
  - Network:
    - Routing of datagrams from source to destination
    - IP, routing protocols
  - Link:
    - Data transfer between neighbouring network elements
    - Ethernet
  - Physical:
    - Bits "on the wire"
- Explanation:
  - Application exchanges messages to implement some application service using services of transport layer
  - Transport layer protocol transfers message from one process to another using services of network layer
  - Network layer protocol transfers transport layer segment from one host to another using link layer services
  - Link layer protocol transfers datagram from host to neighbouring host using network layer services
- Network specific information
  - Source and destination access all layers
  - Routers access physical, link and network
  - Switches only physical and link

Chapter 2 – Application Layer:

Principles of network applications

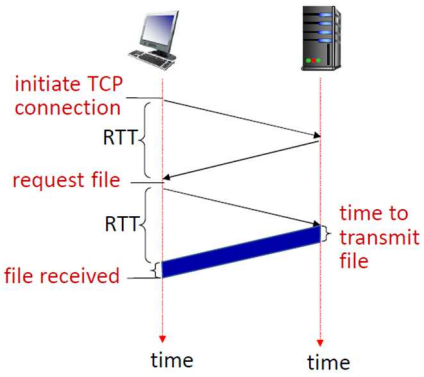
- Create a network app:
  - Write programs that:
    - Run on different end systems
    - Communicate over the network
  - No need to write software for network-core devices
    - Network-core devices do not run user applications
    - Application on end systems allows for rapid app development
- Architectures:
  - Client-Server Paradigm

        - Server
            - Always on host
            - Permanent IP-address
        - Clients
            - Contact and communicate with server
            - Do not communicate directly with each other
        - Examples: HTTP, IMAP, FTP
    - Peer-peer architecture
        - No always-on server
        - Arbitrary end systems directly communicate
        - Complex management because IP can change
- Process communicating
    - Within same host two processes communicate using inter-process communication
    - Different hosts: messages
- Sockets:
    - Process sends/receives messages to/from its socket
- Addressing:
    - To receive messages, process must have identifier
    - Host device has unique 32 bit IP address
    - Identifier includes both IP address and port number associated with the process on host
- Application layer protocol defines:
    - Types of messages exchanged
    - Message syntax
    - Message semantics
    - Rules for when and how processes send and respond to messages
- Types:
    - Open protocols:
        - Defined in RFC, everyone has access to protocol definition
        - Allows for interoperability
        - E.g. HTTP, SMTP
    - Proprietary protocols:
        - E.g. Skype, Zoom
- What transport service does an app need?
    - Data integrity – reliable data transfer
    - Timing – low delay
    - Throughput
    - Security: encryption, data integrity, …
- Internet transport protocols services:
    - TCP service:
        - Reliable transport
        - Flow control – no overwhelming the receiver
        - Congestion control – throttle sender when network overloaded
        - Connection-oriented – setup required between client and server process
        - Does not provide:
            - Timing
            - Minimum throughput guarantee
            - Security
    - UDP service:
        - Unreliable data transfer
        - Does not provide

- Reliability
- Flow control
- Congestion control
- Timing
- Throughput guarantee
- Security
- Connection setup

Web and HTTP

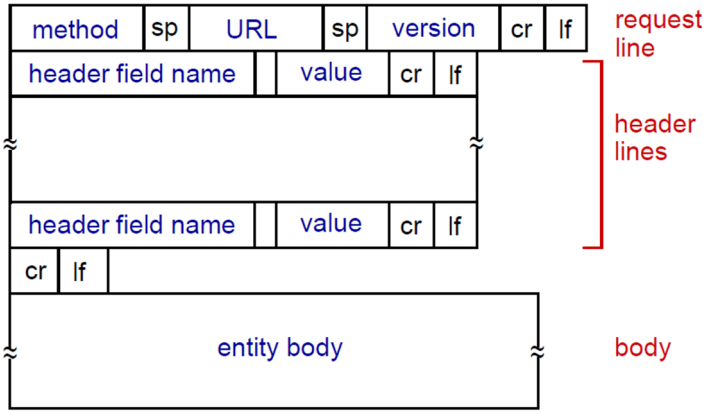
- HTTP overview:
  - Name: hypertext transfer protocol
  - Client-server model:
    - Client: browser that requests and receives and displays Web object
    - Server: Web server sends object in response to request
  - HTTP uses TCP:
    - Client initiates TCP connection
    - Server accepts TCP connection
    - HTTP messages exchanged
    - TCP connection closed
  - HTTP is stateless
    - Server maintains no information about past client requests
- HTTP connection types:
  - Non-persistent HTTP
    - Schedule:
      - TCP connection opened
      - At most one object sent over TCP connection
      - TCP connection closed
    - Downloading multiple objects required multiple connections
    - RTT (definition): time for a small packet to travel from client to server and back
    - HTTP response time (per object):
      - One RTT to initiate TCP connection
      - One RTT for HTTP request and first few bytes of HTTP response to return
      - Object/file transmission time
    - Issues:
      - Requires 2 RTT per object
      - OS overhead for each TCP connection



  - Persistent HTTP
    - Schedule:
      - TCP connection opened to a server
      - Multiple objects can be sent over a single TCP connection
      - TCP connection closed
    - Server leaves connection open after sending response
    - Subsequent HTTP messages between same client/server sent over open comm.
    - Client sends request as soon as it encounters a referenced object
    - Only one RTT for all referenced object
- HTTP request message
  - 2 Types: request/response

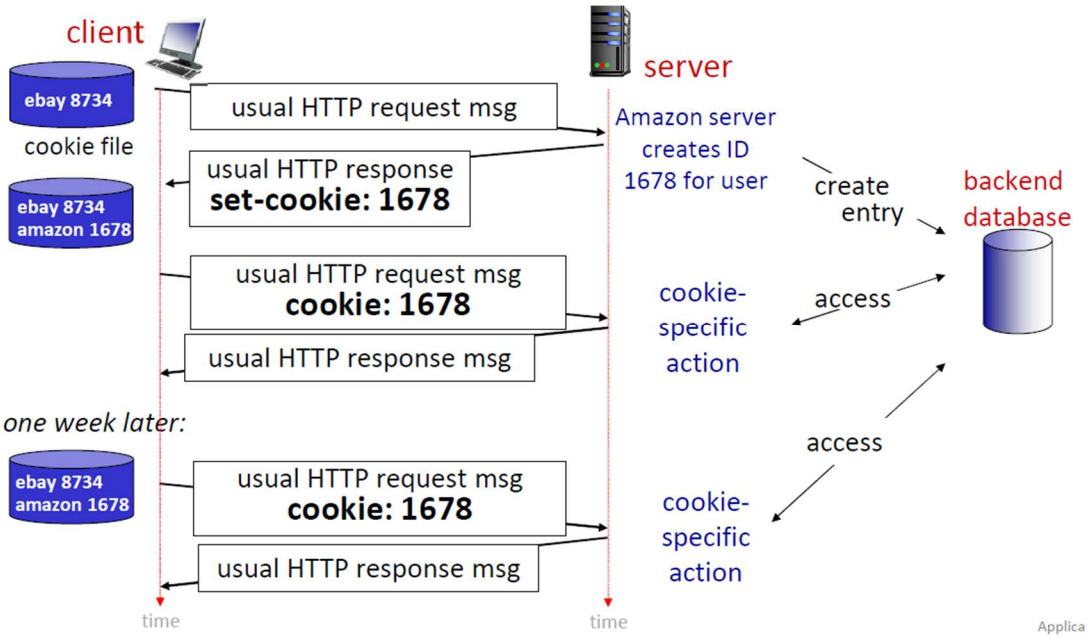
NET – Summary

- Request



  - Other request messages:
    - POST
    - GET
    - HEAD (only fetch request headers)
    - PUT (upload new file or replace an existing)
  - Status codes:
    - 200 – OK
    - 301 – Moved permanently
    - 400 – bad request
    - 404 – not found
    - 505 – HTTP version not supported

- Cookies
  - Maintain server/user state
  - HTTP is statelesds
  - Four components:
    - Cookie header line of HTTP response message
    - Cookie header line in next HTTP request message
    - Cookie file kept on user's host manages by the browser
    - Back-end database at web site



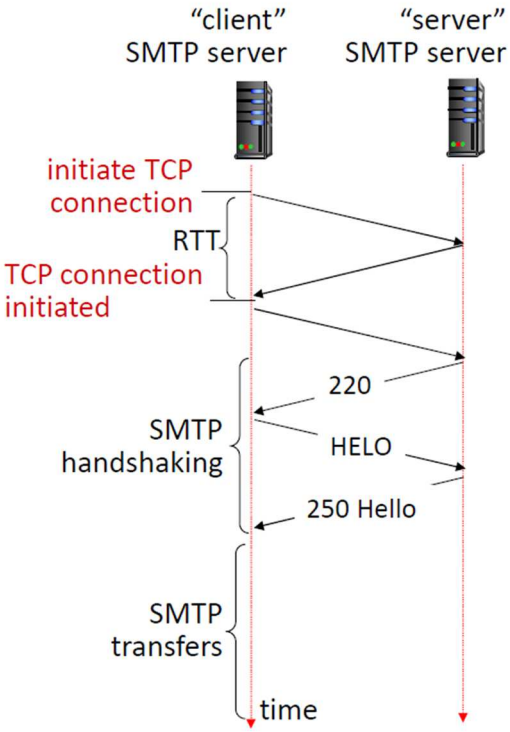
- Cookie usage:
  - Authorization

- Shopping carts
- Recommendations
- User session state

- Web Caches aka Proxy server:
  - Goal: satisfy client requests without involving origin server
  - User configurates browser to point to a web cache
  - Browser sends all HTTP requests to cache
  - Web cache both server and client
  - Why Caching:
    - Reduce response time
    - Reduce traffic for server
- Conditional GET
  - Goal: don't send object if cache has up-to-date cached version
    - No object transmission delay
  - Client: specify date of cached copy in HTTP request
  - Server: responds with 304 Not Modified or new state
- HTTP/2
  - Key goal: decreased delay in multi-object HTTP requests
  - HTTP1.1 introduced multiple, pipelined GETs over single TCP connection
    - Server responds in-order
    - FCFS->small objects may have to wait (head of line blocking)
    - Loss recovery stalls object transmission
  - HTTP/2: increased flexibility at server in sending objects to client:
    - Methods, status codes, most header fields unchanged
    - Transmission order of requested objects based on client-specific object priority
    - Push unrequested objects to client
    - Divide objects into frames, schedule frames to mitigate head of line blocking
      - Round robin can fix head of line blocking as data with fewer frames are transmitted fully first
  - HTTP/2 to 3
    - HTTP2 over single TCP connection means:
      - Recovery from packet loss still stalls all object transmissions
      - No security over vanilla TCP connection
    - HTTP3 adds security per object error and congestion control over UDP

E-Mail, SMTP, IMAP

E-Mail



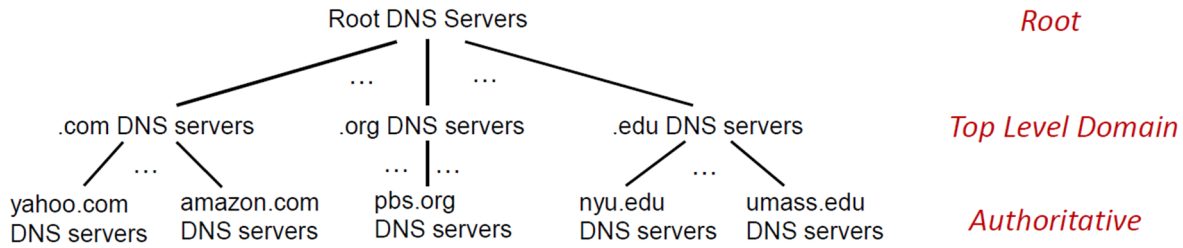
- Three major components:
  - User agents
  - Mail servers
  - Simple mail transfer protocol : SMTP
- User agents
  - Aka mail reader
  - Composing, editing, reading mail messages
  - Outgoing, incoming messages stored on server
- Mail servers
  - Mailbox – contains incoming messages
  - Message queue – outgoing messages
- SMTP protocol between mail servers to send email messages
- SMTP RFC
  - Uses TCP to reliably transfer message, port 25
  - Three phases of transfer
    - SMTP handshaking
    - SMTP transfer of messages
    - SMTP closure
  - Command/response interaction (like HTTP)
    - Commands: ASCII text
    - Response: status code and phrase
- SMTP observations – comparison with HTTP:
  - HTTP: client pull
  - SMTP: client push
  - Both have ASCII command/response interaction, status codes
  - HTTP: each object encapsulated in separate response message
  - SMTP: multiple objects sent in multipart message
  - SMTP uses persistent connections
  - SMTP requires message to be in 7 bit ASCII
  - SMTP server uses CRLF.CRLF to determine end of message
- Retrieving email – mail access protocols
  - SMTP delivery/storage of e-mail messages to receivers server
  - IMAP: mail access/deletion protocol
  - HTTP: web based interface on top of SMTP, IMAP or POP to retrieve messages

Domain Name System – DNS

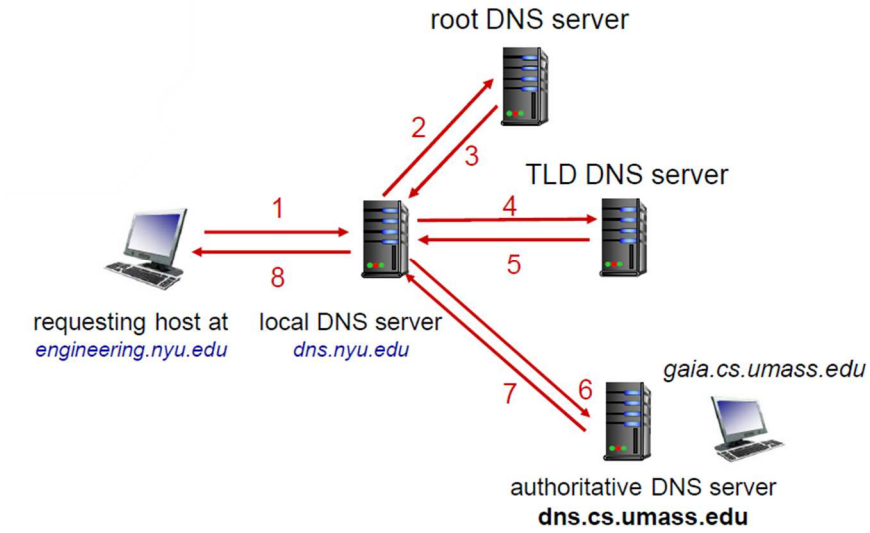
- Mapping between address link and IP address
- Key points:
  - Distributed database implemented in hierarchy of many name servers
  - Application-layer protocol
    - Hosts, DNS server communicate to resolve addresses
    - Core Internet function
    - Complexity at network's edge
- DNS services
  - Hostname to IP translation
  - Host aliasing
  - Mail server aliasing
  - Load distribution (replicated Web services, many IP addresses for one name)

NET – Summary

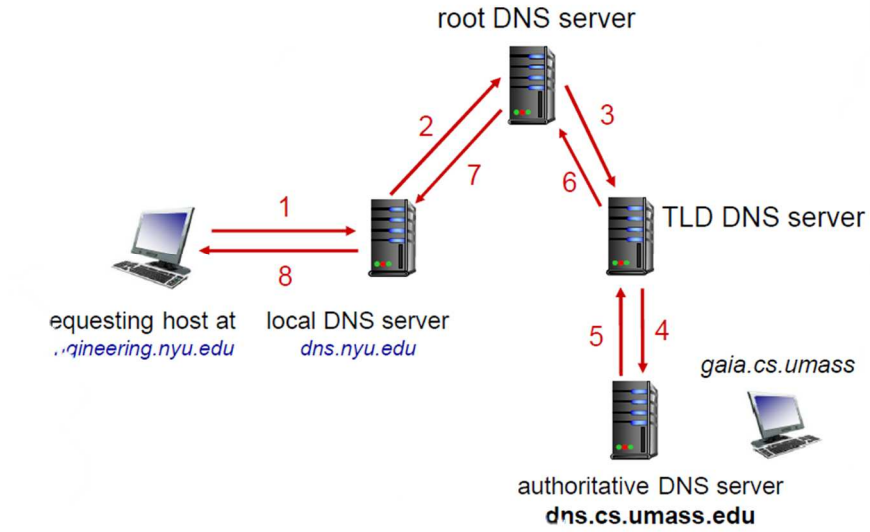
- Properties
  - Humongous distributed database – billions records
  - Handles many trillions of queries each day
    - More reads than writes
    - Performance matters – each transaction interacts with DNS
  - Organizationally, physically decentralized
  - Bulletproof – reliable and secure
- DNS – a distributed and hierarchical database



  - Queries from top to bottom
- DNS – root name servers
  - Official, contact of last resort by name servers that cannot resolve name
  - Incredibly important Internet function
    - Internet could not work without it
    - DNSSEC – provides security
  - ICANN manages root DNS domain
- Top-Level Domain and authoritative servers
  - Top-level Domain(TLD) servers:
    - Responsible for .com, etc… and all top-level country domains
    - Network solutions: authoritative registry for .com, .net
  - Authoritative DNS servers:
    - Organizations own DNS server, providing authoritative hostname to IP mapping for organization's named hosts
    - Can be maintained by organization or service provider
- Local DNS name servers
  - Host makes DNS query -> sent to local DNS server
    - Local DNS server reply:
      - From local cache or recent name to address translation pairs
      - Forward request to DNS hierarchy for resolution
    - Each ISP has local DNS name server
  - Local DNS server doesn't strictly belong to hierarchy
- DNS name resolution
  - Iterated query:
    - Contacted server replies with name of server to contact



- Recursive query:
  - Puts burden of name resolution on contacted name server
  - Heavy load at upper levels of hierarchy



- Caching DNS information
  - Once name server learns mapping it caches mapping and immediately returns a cached mapping in response
    - Caching improved response time
    - Cache entries timeout after some time (TTL)
    - TLD servers typically cached in local name servers
  - Cached entries may be out-of-date
    - Best effort name to address translation
- DNS records
  - DNS – distribution database storing resource records (RR)
  - RR format: (name, value, type, ttl)
  - Type=A
    - Name is hostname
    - Value is IP address
  - Type=NS
    - Name is domain
    - Value is hostname of authoritative name server

- Type=CNAME
  - Name is alias for some canonical name
  - Value is canonical name
- Type=MX
  - Value is name of SMTP mail server associated with name

- DNS protocol messages
  - DNS query and reply messages format
  - Header
    - Identification: 16 bit number for query
    - Flags:
      - Query or reply
      - Recursion desired
      - Recursion available
      - Replay is authoritative
    - Questions
    - Answers RR
    - Authority RR
    - Additional RR
  - Body
    - Questions and RRs
- DNS security
  - DDoS attacks
    - Bombard root servers with traffic
      - Not successful
      - Traffic filtering
      - Local DNS servers cache Ips of TLD servers allowing root server bypass
    - Bombard TLD servers
      - Potentially more dangerous
  - Spoofing attacks
    - Intercept DNS queries, return bogus replies
    - DNS cache poisoning

P2P – applications

- P2P architectures :
  - No always-on server
  - Arbitrary end systems directly communicate
  - Peers request service from other peers provide service in return
    - Self-scalability – new peers bring new service capacity and new service demands
  - Peers are intermittently connected and change IP addresses
    - Complex management
  - Examples: BitTorrent, Kankan, VoIP
- File distribution:
  - Client-server
    - Server transmissions
      - Must sequentially send N file copies
      - Time to send one copy=t_s
      - Time to send n copies = $n*t_s$
    - Client – each client must download file copy
      - D_{min} = min client download rate
      - Min client download time = F/d_{min}
    - Increases nearly linearly in N

- P2P
  - Server transmissions: must upload at least one copy
    - Time to send one copy
  - Client – each client must download file copy
    - Min client download time
  - Clients – as aggregate must download NF bits
  - Faster as server is not blocked and data can be fetched from client instead
  - While downloading peer upload chunks to other peers
  - Once peer has entire file it may
    - Leave
    - Stay in torrent and resume providing
- BitTorrent – requesting and sending of file chunks
  - Requesting chunks
    - At a given time different peers have different subsets of file chunks
    - Periodically each peer is asked for available chunks
    - Request missing chunks from peers, rarest first
  - Sending chunks: tit-for-tat
    - Sending chunks to those four peers currently sending chunks to "me" at highest rate
    - Every 30 seconds: randomly select another peer and start sending chunks

Video streaming and content distribution networks

- Context:
  - Streaming video traffic: major consumer of Internet bandwidth
  - Challenge: scaling
  - Challenge: heterogeneity (different user different capability)
  - Solution: distributed application layer infrastructure
- Multimedia: Video
  - Coding: use redundancy within and between images
    - Spatial (within image)
    - Temporal (from one image to next)
  - CBR = constant bit rate … video encoding rate fixed
  - VBR = variable bit rate … video encoding rate changes as amount of spatial and temporal coding changes
- Streaming:
  - Challenge – continuous playout constraint
    - But client buffering needed
  - Other challenges: client interactions, packet loss
  - DASH – dynamic adaptive streaming over http
    - Server:
      - Divides video file into chunks
      - Each chunk encoded at different rates
      - Stored in different files
      - Files replicated over different nodes
      - Manifest file provides url for chunks
    - Client
      - Periodically estimates server to client bandwidth
      - Consulting manifest, requests one chunk at a time
        - Chooses maximum coding rate sustainable given current bandwidth
        - Can choose different rates and different servers at different points of time
      - Intelligence:
        - When to request

        - What encoding rate
        - Where to request from
    - Streaming Video = encoding + DASH + playout buffering
- Content distribution networks:
    - Challenge: how to stream content to many simultaneous users
    - Store and serve multiple copies of videos at multiple geographically distributed cites
    - Subscriber requests content, service provider returns manifest
    - OTT challenges – copying with congestion
        - What content to place where
        - From which to retrieve

Socket programming with UDP and TCP

- Goal: learn how to build client/server applications that communicate using sockets
- Socket: door between application process and end-end-transport protocol
- Two socket types for two transport services:
    - UDP: unreliable datagram
    - TCP: reliable byte stream oriented
- Socket programming with UDP
    - No connection between client and server
        - No handshaking
        - Sender explicitly attaches IP destination address and port
        - Receiver extracts sender IP address and port
    - Transmitted data may be lost or received out of order
- Socket programming with TCP
    - Client must contact server
    - Client contacts server by
        - Creating TCP socket
        - Specifying IP address
        - Port number of server process
    - when client creates socket: client TCP establishes connection to server TCP
    - when contacted by client the server TCP creates a new socket for server process to communicate
        - allows server to talk to multiple clients
        - source port numbers used to distinguish clients

Chapter 3 – Transport Layer

Transport layer services

- provide logical communication between application processes running on different hosts
- transport protocols actions in end systems:
  - sender: breaks application messages into segments then passed to network layer
  - receiver: reassembles segments into messages then passes to application layer
- two transport protocols available to Internet applications:
  - TCP
  - UDP
- Transport vs network layer services and protocols
  - Network layer: logical communication between hosts
  - Transport layer: logical communication between processes
- Actions:
  - **Sender** is passed an application layer message
  - Determines segment header fields values
  - Creates segment
  - Passes segment to IP
  - **Receiver** receives segment from IP
  - Checks header values
  - Extracts message
  - Demultiplexes message up to application via socket
- Two principal Internet transport protocols
  - TCP – Transmission Control Protocoll
    - Reliable, in order delivery
    - Congestion control
    - Flow control
    - Connection setup
  - UDP – User Datagram Protocol
    - Unreliable, unordered delivery
    - No-frills extension of best effort IP
  - Not available
    - Delay guarantees
    - Bandwidth guarantees

Multiplexing and demultiplexing

- Multiplexing at sender
  - Handle data from multiple sockets, add transport header
- Demultiplexing at receiver
  - Use header information to deliver received segments to correct socket
- Demultiplexing:
  - Host receives IP datagrams. Each datagram…
    - Has source IP address and destination IP address
    - Carries one transport-layer segment
    - Has source and destination port number
- Connectionless demultiplexing UDP
  - Host uses IP address and port to direct segment to appropriate socket
    - Source is not relevant for socket choice
- Connection oriented demultiplexing TCP
  - TCP socket identified by 4-tuple
    - Source IP & port

        - Destination IP & port
    - Demux: receiver uses all four values to direct segment to appropriate socket
    - Server may support many simultaneous TCP sockets
        - Each socket identified by its own tuple
        - Each socket associated with a different connecting client

Connectionless transport UDP

- No frills
- Bare bones
- Best effort service but segments may be lost or delivered out of order
- Connectionless:
    - No handshaking between UDP sender and receiver
    - Each UDP segment handled independently of others
- Small header size
- No congestion control
- Can function when network service is compormised
- UDP use:
    - Streaming multimedia apps
    - DNS
    - SNMP
    - HTTP/3
- If reliable transfer needed over UDP:
    - Add needed reliability at application level
    - Add congestion control at application level

UDP structure

- Segment header
    - Source port
    - Destination port
    - Length of UDP segment in bytes including header
    - Checksum
    - Application data
- Checksum:
    - Goal: detect errors in transmitted segment
    - Sender:
        - Treat content of UDP segment as sequence of 16 bit integers
        - Checksum = addition of segment content
        - Checksum value put into field
    - Receiver:
        - Compute checksum of received segment
        - Check if match:
            - No – error detected
            - Yes – no error detected

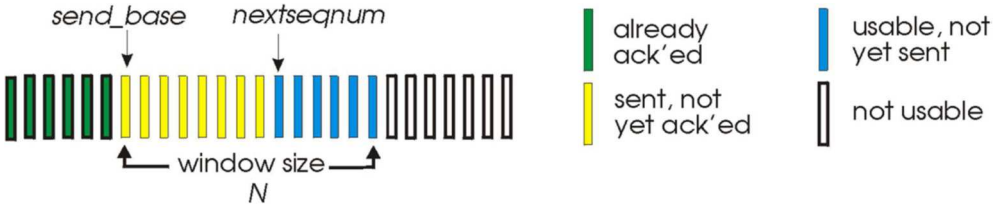
Principles of reliable data transfer

- Overview:
    - Needs implementation to realize reliability
    - Each participant of communication does not know the state of the other one
    - (use finite state machines to specify sender and receiver)
- Rdt1.0 – reliable transfer over reliable channel
    - Underlying channel perfectly reliable

        - No bit errors
        - No loss of packets
    - Separate FSMs for sender and receiver
        - Sender sends data into underlying channel
        - Receiver reads data from underlying channel
- Rdt2.0 – channel with bit errors
    - Underlying channel may flip bits in packet
        - Checksum to detect bit errors
    - How to recover from errors?
        - Acknowledgements (ACK): receiver explicitly tells sender that pkt received OK
        - Negative acknowledgements (NAKs): receiver explicitly tells sender that pkt had errors
        - Sender retransmits pkt on receipt of NAK
    - →STOP AND WAIT
    - What if ACK/NAK is corrupted?
        - Sender does not know what happened at receiver
        - Cannot just retransmit -> possible duplicates
    - Handling duplicates
        - Sender retransmits current pkt if ACK/NAK corrupted
        - Sender adds sequence number to each pkt
        - Receiver discards duplicate pkt
    - Discussion:
        - Sender:
            - Sequence number added
            - 2 sequence numbers
            - Must check if ACK/NAK corrupted
            - Twice as many states
        - Receiver:
            - Must check if received packet is duplicate
    - Version 2.2:
        - Same functionality including ACKs
        - Instead of NAK, receiver sends ACK for last pkt received
            - Receiver must explicitly include sequence number of pkt being ACKed
        - Duplicate ACK at sender results in same action as NAK
- Rdt3.0 – channels with errors and loss
    - New channel assumption: underlying channel can also lose packets (data, ACKs)
    - Approach: sender waits a reasonable amount of time for ACK
        - Retransmits if no ACK received in this time
        - If pkt just delayed:
            - Retransmission will be duplicate, but handled by sequence number
            - Receiver must specify sequence number in ACK
        - Use countdown timer to interrupt after reasonable amount of time
    - Performance
        - U_{sender}: utilization – fraction of time sender busy sending
        - $U_{\text{sender}} = \frac{\frac{L}{R}}{RRT+\frac{L}{R}}$
        - Performance bad
        - Protocol limits possible performance of underlying infrastructure (channel)
    - Pipelining:
        - Sender allows multiple yet to be acknowledged packets
            - Range of sequence numbers must be increased

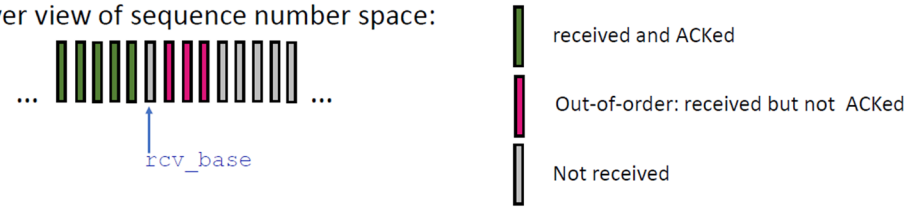
- Buffering at sender and or receiver
- Increased utilization

- Go-Back-N: sender
  - Sender "window" of up to N consecutive but unACKed pkts



    k-bit sequence number in pkt header
  - Cumulative ACK: ACK(n): ACKs all packets up to (including) sequence number n
    - On receiving ACK(n) move window forward to begin at n+1
  - Timer for oldest in-flight packet
  - Timeout(n): retransmit packet n and higher sequence number packets in window
- Go-Back-N: receiver
  - ACK-only: always send ACK for correctly-received packet so far with highest in-order sequence number
    - May generate duplicate ACKs
    - Need only remember rcv_base
  - On receipt of out-of-order packet:
    - Can discard or buffer – implementation choice
    - Re-ACK pkt with highest in-order sequence number

Receiver view of sequence number space:



- Selective repeat
  - Receiver individually acknowledges all correctly received packets
    - Buffers packets as needed for eventual in-order delivery to upper layer
  - Sender times-out/retransmits individually for unACKed packets
    - Sender maintains timer for each unACKed packet
  - Sender window
    - N consecutive sequence numbers
    - Limits sequence numbers of sent, unACKed packets
  - Sender
    - Data from above:
      - If next available sequence number in window, send packet
    - Timeout(n)
      - Resend packet n, restart timer
    - ACK(n) in [sendbase, sendbase+N]
      - Mark packet n as received
      - If n smallest unACKed packet advance window base to next unACKed sequence number
  - Receiver
    - Packet n in [rcvbase, rcvbase+N-1]

- Send ACK(n)
- Out-of-order: buffer
- In-order: advance window to next not yet received packet

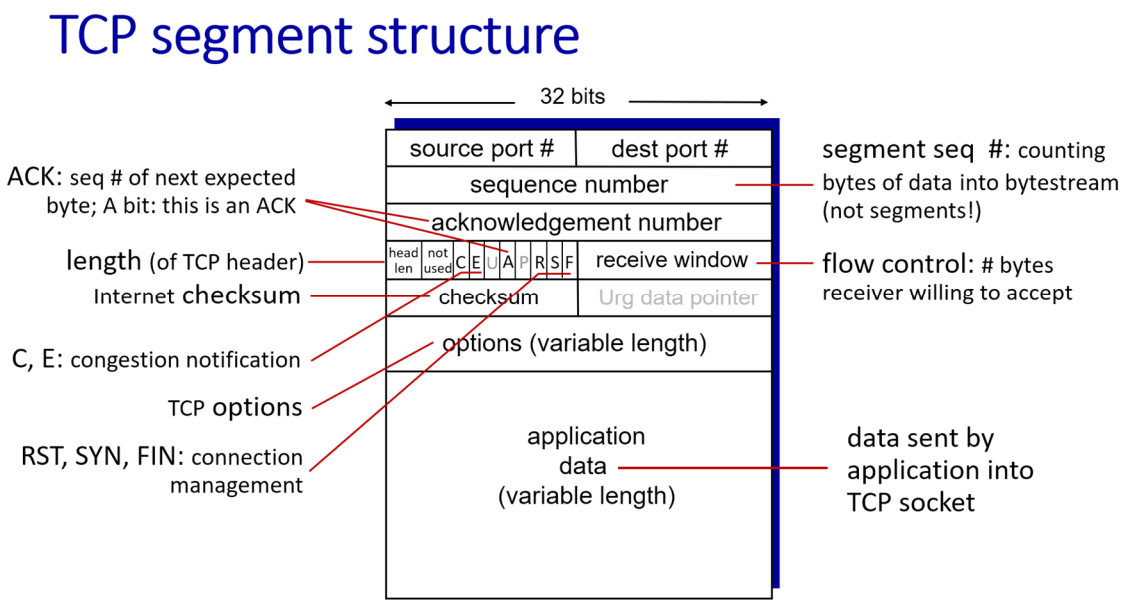
- Packet n in [rcvbase-N, rcvbase-1]
  - ACK(n)
- Otherwise: ignore

- Dilemma if sequence number only between 0 and 4 and window size 3:
  - Problem if ACKs are lost old packages could be accepted as next batches packets
  - What relation is needed between sequence number size and window size to avoid problem? Sequence number size >> window size (at least twice as much)

Connection-oriented transport: TCP

TCP overview

- Point to point
  - One sender, one receiver
- Reliable, in order byte stream
  - No message boundaries
- Full duplex data
  - Bi-directional data flow in same connection
  - MSS: maximum segment size
- Cumulative ACKs
- Pipelining:
  - TCP congestion and flow control set window size
- Connection-oriented
  - Handshaking initiates sender and receiver state before data exchange
- Flow controlled
  - Sender will not overwhelm receiver



- Sequence numbers:
  - Byte stream number of first byte in segments data
- Acknowledgements:
  - Sequence number of next byte expected from other side
  - Cumulative ACK
- TCP round trip time – how to estimate
  - SampleRTT: measured time from segment transmission until ACK receipt

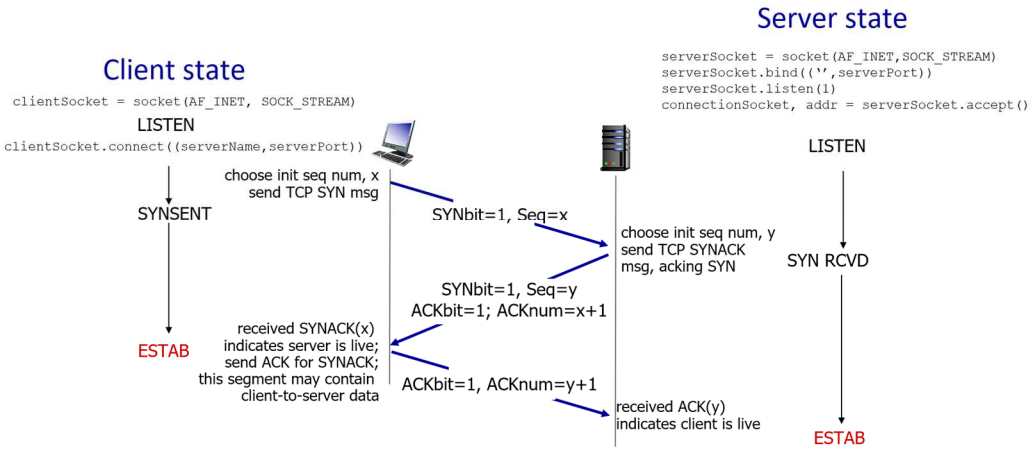
        - Ignore retransmission
    - SampleRTT will vary; want estimated RTT to be smoother
        - Average several recent measurements not just current
    - EstimatedRTT=$(1-\alpha) * \text{EstimatedRTT} + \alpha * \text{SampleRTT}$
        - Exponential weighted moving average (EWMA)
        - Influence of past sample decreases exponentially fast
        - Typical value: $\alpha = 0{,}125$
- TCP timeout
    - Longer than RTT but RTT varies
    - Too short: premature timeout, unnecessary retransmissions
    - Too long: slow reaction to segment loss
    - Timeout interval: EstimatedRTT plus "safety margin"
        - Large variation in EstimatedRTT – want a larger safety margin
        - $\text{TimeoutInterval} = \text{EstimatedRTT} + 4 * \text{DevRTT}$
    - DevRTT: EWMA of SampleRTT deviation from EstimatedRTT:
        - $\text{DevRTT} = (1-\beta) * \text{DevRTT} + \beta * |\text{SampleRTT} - \text{EstimatedRTT}|$
        - Typically $\beta = 0{,}25$
- TCP Sender
    - Event : data received from application
        - Create segment with segment number
        - Sequence number is byte-stream number of first data byte in segment
        - Start timer if not already running
            - Think of timer as for oldest unACKed segment
            - Expiration interval: TimeOutInterval
    - Event: timeout
        - Retransmit segment that caused timeout
        - Restart timer
    - Event: ACK received
        - If ACK acknowledges previous unACKed segments
            - Update what is known to be ACKed
            - Start timer if there are still unACKed segments
- TCP Receiver

Event at receiver	*TCP receiver action*
arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK
arrival of in-order segment with expected seq #. One other segment has ACK pending	immediately send single cumulative ACK, ACKing both in-order segments
arrival of out-of-order segment higher-than-expect seq. # . Gap detected	immediately send *duplicate ACK*, indicating seq. # of next expected byte
arrival of segment that partially or completely fills gap	immediate send ACK, provided that segment starts at lower end of gap

- TCP fast retransmit
    - If sender receives 3 additional ACKs for same data->resend unACKed segment with smallest sequence number
    - Don't wait for timeout

Flow control and connection management

- Flow control – concepts
  - What if network layer delivers data faster than application layer
  - Buffers full
  - ->flow control in answer
    - Field receive window: flow control: number of bytes receiver is willing to accept
  - Receiver controls sender so sender won't overflow receivers buffer
- TCP flow control
  - TCP receiver advertises free buffer space in rwnd field in TCP header
    - RcvBuffer size set via socket options
    - many operating systems autoadjust RcvBuffer
  - sender limits amount of unACKed data to received rwnd
  - guarantees receive buffer will not overflow
- TCP connection management
  - Before exchanging data sender/receiver performs handshake
    - Agree to establish connection
    - Agree on connection parameters
  - 2-way handshake may not be enough if delays occur or messages get lost
  - 3-way handshake with sequence and acknumbers that depend on what the previous message brought



  - Closing a tcp connection
    - Client and server each close their side of connection
      - Send TCP segment with FIN bit=1
    - Respond to received FIN with ACK
      - On receiving FIN, ACK can be combined with own FIN
    - Simultaneous FIN exchanges can be handled

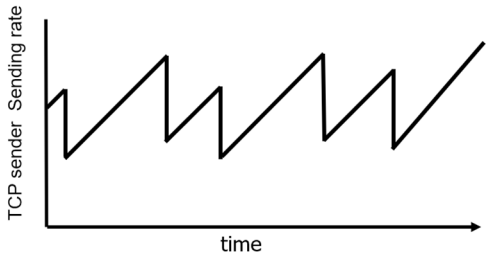
Principles of congestion control

- Congestion:
  - Informally: too many sources sending too much data too fast for network to handle
  - Manifestations:
    - Long delays (queueing in router buffers)
    - Packet loss (buffer overflow)
  - Different from flow control (many senders sending too fast vs one sender to fast for one receiver)
  - A top-10 problem
  - Cost of congestion
    - More work for given receiver throughput
    - Unneeded retransmissions: link carries multiple copies of a packet
      - Decreasing maximum achievable throughput

        - When packet dropped any upstream transmission capacity and buffering used for that packet was wasted
    - Causes/Costs of congestion: insights:
        - Throughput can never exceed capacity
        - Delay increases as capacity approached
        - Loss/retransmission decreases effective throughput
        - Un-needed duplicates further increases effective throughput
        - Upstream transmission capacity/buffering wasted for packets lost downstream
- Approaches towards congestion control
    - End-end congestion control
        - No explicit feedback from network
        - Congestion inferred from observed loss, delay
        - Approach taken by TCP
    - Network-assisted congestion control
        - Routers provide direct feedback to hosts with flow passing through congested router
        - May indicate congestion level or explicitly set sending rate
        - TCP ECN, ATM, DECbit protocols

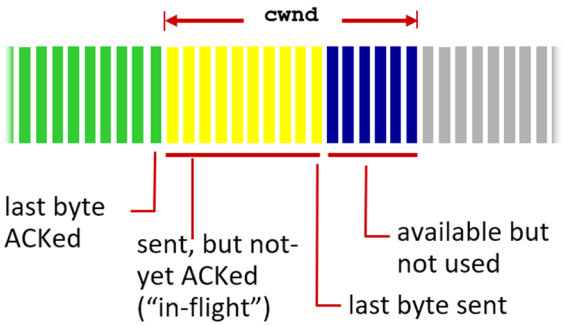
TCP congestion control

- AIMD
    - Approach: senders can increase sending rate until packet loss occurs, then decrease sending rate on loss event
    - Additive increase:
        - Increase sending rate by 1 maximum segment size every RTT until loss detected
    - Multiplicative decrease
        - Cut sending rate in half at each loss event
    - AIMD sawtooth behaviour: probing for bandwidth



- TCP AIMD: more
    - Multiplicative decrease detail: sending rate is
        - Cut in half on loss detected by triple duplicate ACK
        - Cut to 1 MSS (maximum segment size) when loss detected by timeout
    - Why AIMD?
        - AIMD – a distributed, asynchronous algorithm has be shown to:
            - Optimize congested flow rates network wide
            - Have desirable stability properties

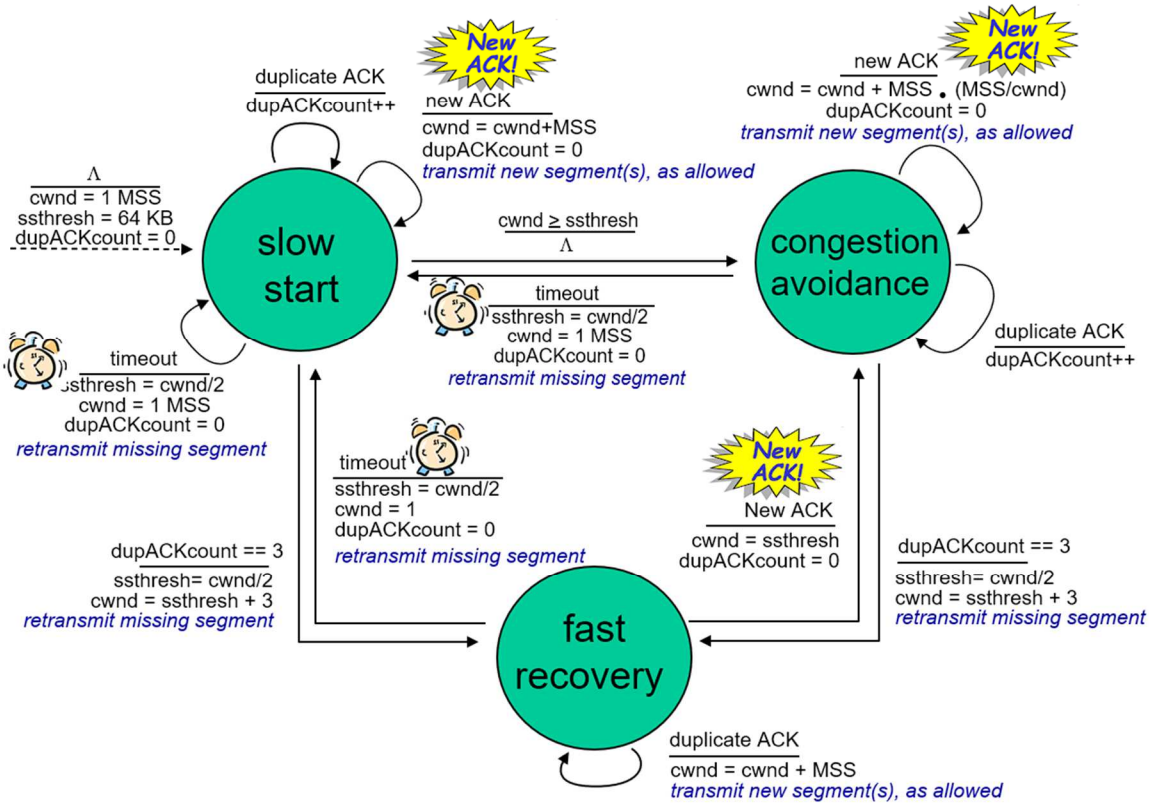
- TCP congestion control: details
  sender sequence number space



  - TCP sender limits transmission: $\text{LastByteSent} - \text{LastByteAcked} \leq cwnd$
  - Cwnd is dynamically adjusted in response to observed network congestion
  - TCP sending behaviour:
    - Roughly: send cwnd bytes, wait RTT for ACKs, then send more bytes
      TCP rate $\cong \frac{cwnd}{RTT}$ bytes/sec

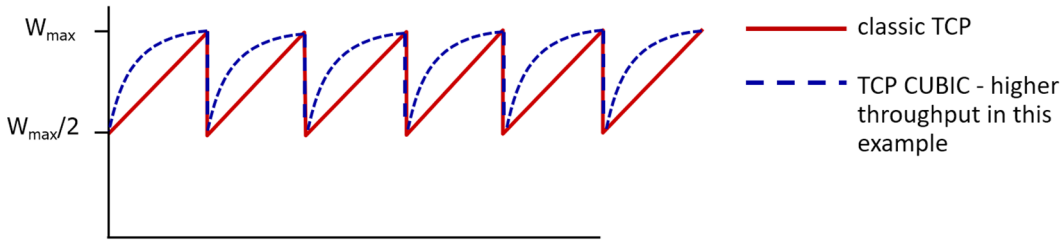
- TCP slow start:
  - When connection begins -> increase rate exponentially until first loss event:
    - Initially cwnd=1MSS
    - Double cwnd every RTT
    - Done by incrementing cwnd for every ACK received
  - Summary: initial rate is slow but ramps up exponentially fast

Summary: TCP congestion control



- TCP CUBIC
  - Is there a better way than AIMD to probe for bandwidth?
  - Insight/intuition

- $W_{\max}$ sending rate at which congestion loss was detected
- Congestion state of bottleneck link probably hasn't changed much
- After cutting rate/window in half in loss, initially ramp to $W_{\max}$ faster but then approach $W_{\max}$ more slowly



  - K: point in time when TCP window size will reach $W_{\max}$
    - K itself is tunable
  - Increase W as function of the cube of the distance between current time and K
    - Larger increase when further away from K
    - Smaller increase when nearer K
  - TCP CUBIC default in Linux, most popular TCP for popular Web servers
- TCP and the congested "bottleneck link"
  - TCP increase sending rate until packet loss occurs at some router: bottleneck link
  - Understanding congestion: useful to focus on congested bottleneck link
  - Goal: keep the end-end pipe just full but not fuller
  - Delay-based TCP congestion control
    - Keeping sender-to-receiver pipe just full enough but no fuller:
      → keep bottleneck link busy transmitting but avoid high delays/buffering
    - $RTT_{\min}$ – minimum observed RTT (uncongested path)
    - Uncongested throughput with congestion window cwnd is cwnd/RTT_min
    - Congestion control without inducing/forcing loss
    - Maximizing throughput while keeping delay low
    - A number of deployed TCPs take a delay-based approach
  - Explicit congestion notification (ECN)
    TCP deployments often implement network-assisted congestion control
    - Two bits in IP header (ToS field) marked by network router to indicate congestion
      - Policy to determine marking chosen by network operator
    - Congestion indication carried to destination
    - Destination sets ECE bit on ACK segment to notify sender of congestion
    - Involves both IP and TCP
  - TCP fairness
    - Fairness goal: if K TCP session share same bottleneck link of bandwidth R, each should have average rate of R/K

Fairness in network apps

- Fairness and UDP
  - Multimedia apps often do not use TCP
    - Do not want to be throttled by congestion control
  - Instead use UDP
    - Send at constant rate and tolerate loss
  - There is no Internet police enforcing congestion control
- Fairness and parallel TCP connections
  - Application can open multiple parallel connections between 2 hosts

Evolution of transport-layer functionality

- TCP, UDP principal transport protocols for 40 years

- Different versions for specific scenarios
- Moving transport-layer functions to application layer on top of UDP
    - HTTP/3: QUIC
- QUIC: Quick UDP Internet Connections
    - Application layer protocol
        - Increase performance of http
    - Adopts approaches of connection establishment, error control and congestion control
        - Error and congestion control: similar to TCP
        - Connection establishment: reliability, congestion control, authentication, encryption, state established in one RTT
    - Multiple application-level streams multiplexed over single QUIC connection
        - Separate reliable data transfer
        - Common congestion control

Chapter 4: Network Layer – Data Plane

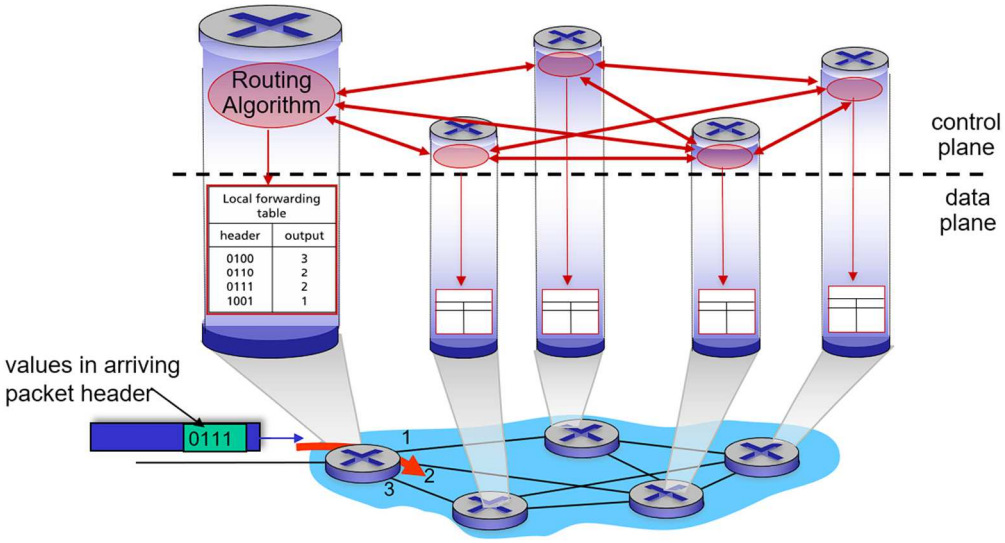
- Goals:
    - Understand principles behind network layer services focusing on data plane:
        - Network layer service models
        - Forwarding vs routing
        - How a router works
        - Addressing
        - Generalized forwarding
        - Internet architecture
    - Instantiation and implementation in the Internet
        - IP protocol
        - NAT, middleboxes

Network layer – overview

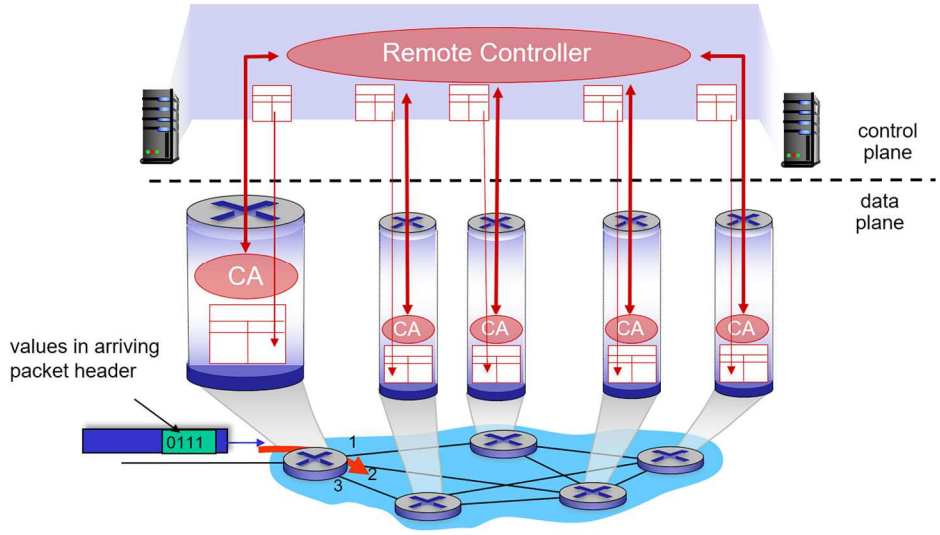
- Network layer services and protocols
    - Transport segment from sending to receiving host
        - Sender: encapsulate segments into datagrams and pass to link layer
        - Receiver: delivers segments to transport layer protocol
    - Network layer protocols in every internet device: hosts, routers, ect.
    - Routers:
        - Examines header fields in all IP datagrams passing though
        - Moves datagram from input port to output port to transfer datagrams along end-end path
- Two key network layer functions
    - Network layer functions:
        - Forwarding:
            - Move packet from router input link to appropriate router output link
        - Routing:
            - Determine route taken by packet from source to destination
            - Routing algorithms
    - Analogy: taking a trip
        - Forwarding: process of getting through single interchange
        - Routing: process of planning trip from source to destination
- Data plane
    - Local, per-router function
    - Determines how datagram arriving on router input port is forwarded to router output port
- Control plane:

- Network-wide logic
- Determines how datagram is routed through Network
- Two control-plane approaches:
  - Traditional routing algorithms:
    - Implemented in routers
  - Software-defined networking (SDN):
    - Implemented in servers

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



- Software-defined networking (SDN) control plane
  - Remote controller computes and installs forwarding tables in routers



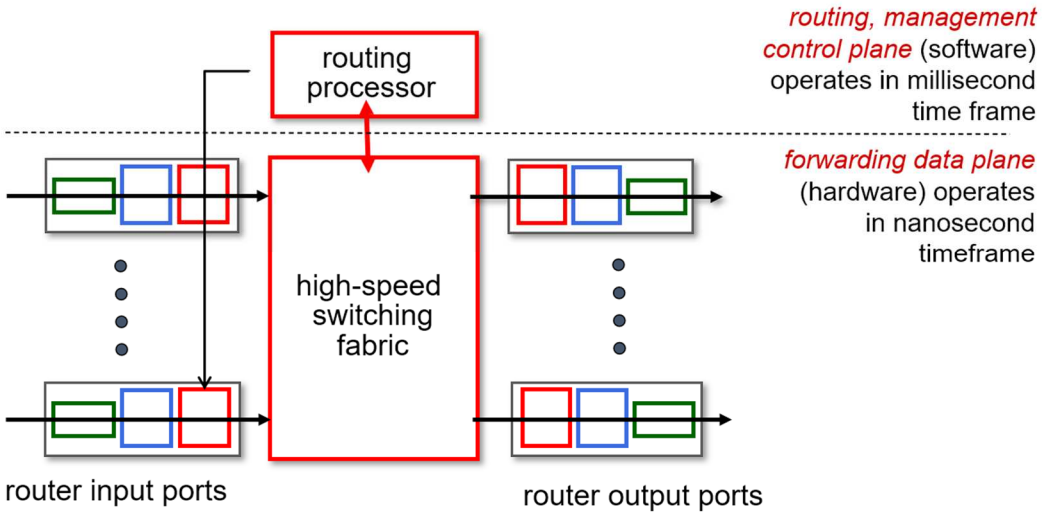
- Service models
  - Best effort service model
    - No guarantees on:
      - Successful datagram delivery to destination
      - Timing or order of delivery
      - Bandwidth available to end-end flow

Network Architecture	Service Model	Quality of Service (QoS) Guarantees ?			
		Bandwidth	Loss	Order	Timing
Internet	best effort	none	no	no	no
ATM	Constant Bit Rate	Constant rate	yes	yes	yes
ATM	Available Bit Rate	Guaranteed min	no	yes	no
Internet	Intserv Guaranteed (RFC 1633)	yes	yes	yes	yes
Internet	Diffserv (RFC 2475)	possible	possibly	possibly	no

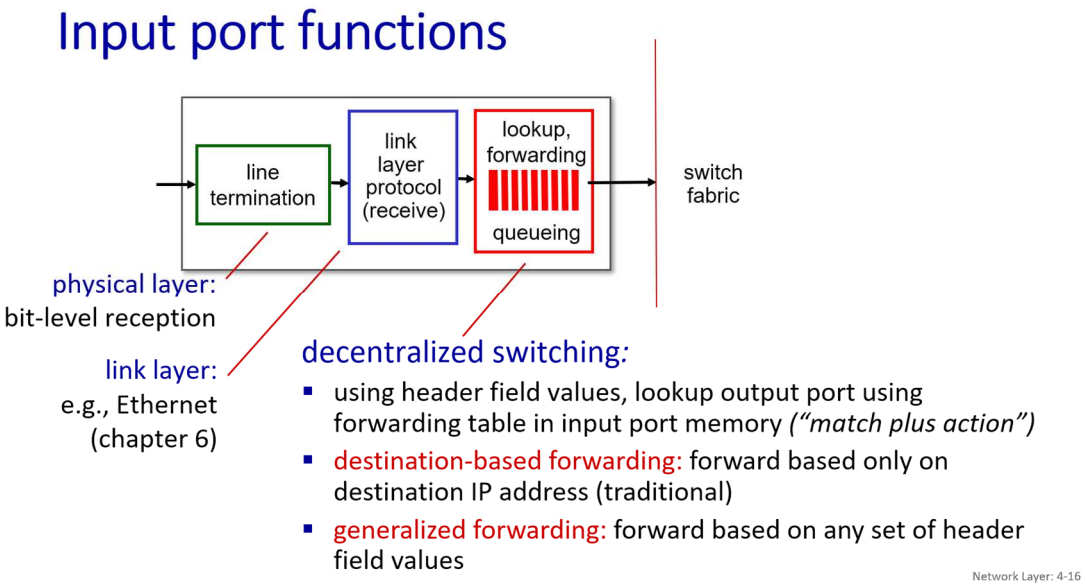
- Reflections on best-effort service:
  - Simplicity of mechanism
    - Allows Internet to be widely deployed
  - Sufficient provisioning of bandwidth
    - Allows performance of real-time applications
  - Replicated application layer distributed services
    - Allow services to be provided from multiple locations
  - Congestion control of elastic services helps

What's inside a router

- Router architecture overview:
  - High level view of generic router architecture:



- Input port functions:



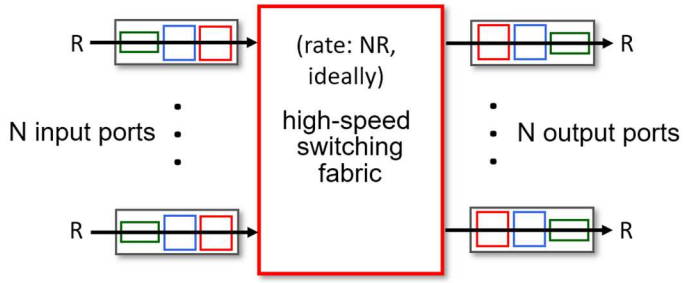
- Destination based forwarding:
  - Uses forwarding table
- Longest prefix matching:
  - Def: when looking for forwarding table entry for given destination address, use longest address prefix that matches destination address

Destination Address Range	Link interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

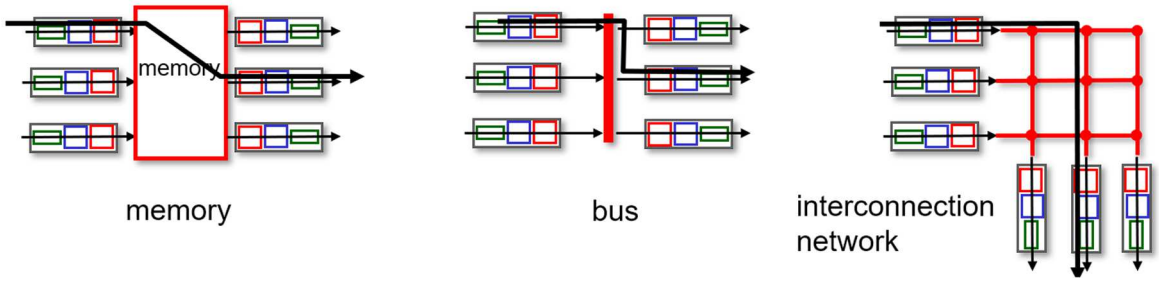
  - Often performed using ternary content addressable memories (TCAMs)
    - Content addressable: present address to TCAM: retrieve address in one clock cycle regardless of table size
    - Cisco Catalyst: ~1M routing table entries in TCAM

NET – Summary

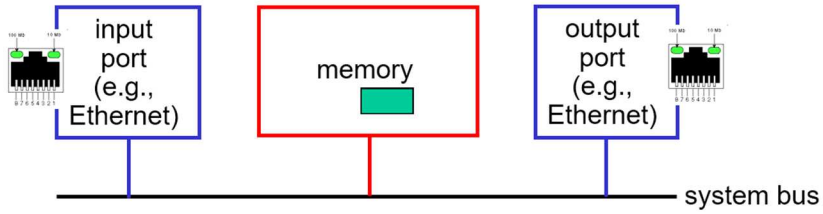
- Switching fabrics
  - Transfer packet from input link to appropriate output link
  - Switching rate: rate at which packets can be transferred from input to outputs
    - Often measured as multiple of input/output line rate
    - N inputs: switching N times line rate desirable



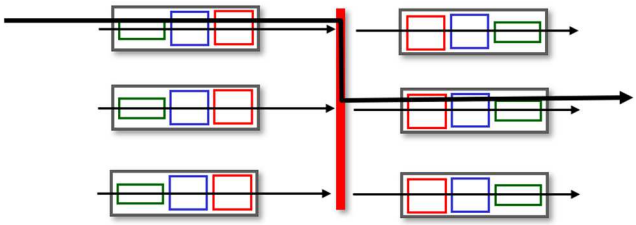
- Three major types of switching fabrics:



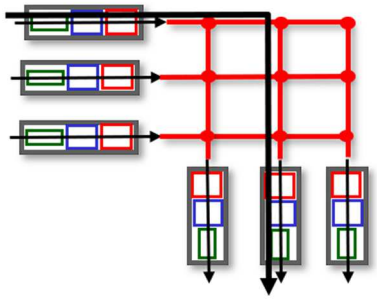
- Switching via memory:
  - First generation routers:
    - Traditional computers with switching under direct control of CPU
    - Packet copied to system's memory
    - Speed limited by memory bandwidth (2 bus crossing per datagram)



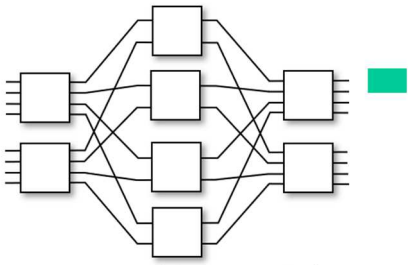
- Switching via a bus
  - Datagram from input port memory to output port memory via a shared bus
  - Bus connection: switching speed limited by bus bandwidth
  - 32 Gbps bus, Cisco 5600: sufficient speed for access routers



- Switching via interconnection network
  - Crossbar, Clos networks, other interconnection nets initially developed to connect processors in multiprocessors
  - Multistage switch: nxn switch from multiple stages of smaller switches
  - Exploiting parallelism:
    - Fragment datagram into fixed length cells on entry
    - Switch cells through the fabric, reassemble datagram at exit



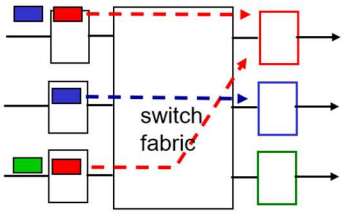
3x3 crossbar



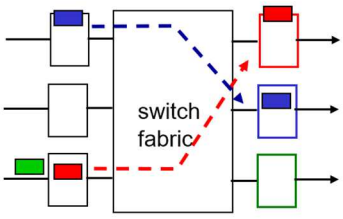
8x8 multistage switch
built from smaller-sized switches

- Scaling using multiple switching planes in parallel:
  - Speedup and scaleup via parallelism
- Cisco CRS router:
  - Basic unit: 8 switching planes
  - Each plane: 3-stage interconnection network
  - Up to 100's Tbps switching capacity

- Input port queueing:
  - If switch fabric slower than input ports combined -> queueing may occur at input queues
    - Queueing delay and loss due to input buffer overflow
  - Head-of-line-blocking: queue datagram at front of queue prevents others in queue from moving forward



output port contention: only one red datagram can be transferred. lower red packet is *blocked*



one packet time later: green packet experiences HOL blocking

- Output port queueing:
  - Buffering required when datagrams arrive from fabric faster than link transmission rate.
    - Drop policy: which datagrams to drop if no free buffers?
  - Scheduling discipline chooses among queued datagrams for transmission -> priority scheduling: best performance
  - Buffering when arrival rate via switch exceeds output line speed
  - Queueing (delay) and loss due to output port buffer overflow
- How much buffering?
  - Rule of thumb: average buffering equal to typical RTT times link capacity C
  - More recent recommendation: with N flows, buffering equal to $\frac{RTT*C}{\sqrt{N}}$
  - But too much buffering can increase delays
    - Long RTTs: poor performance for realtime apps, sluggish TCP response
    - Recall delay-based congestion control: "keep bottleneck link just full enough but no fuller"
- Buffer Management:
  - Drop: which packet to add or drop when buffers are full
    - Tail drop: drop arriving packet
    - Priority: drop/remove on priority basis
  - Marking: which packets to mark to signal congestion (ECN, RED)
- Packet Scheduling: FCFS – deciding which packet to send next on link
  - FCFS
  - Priority
    - Arriving traffic classified and queued by class

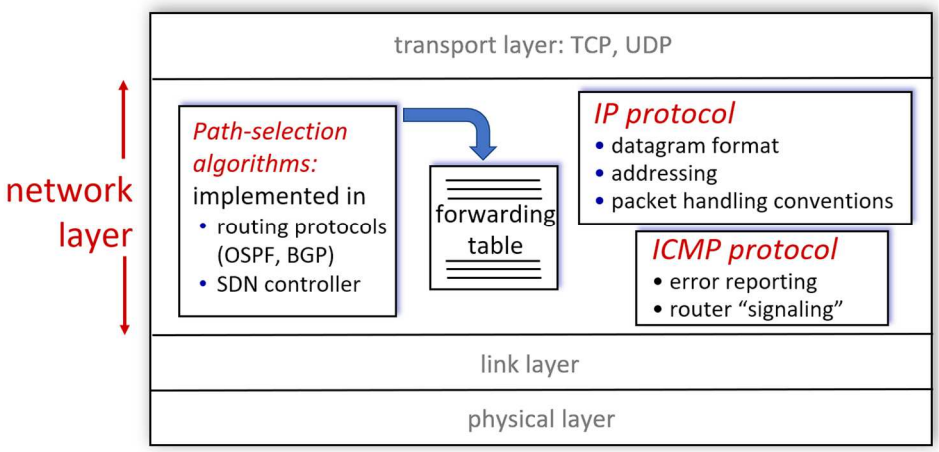
        - Any header fields can be used for classification
        - Send packet from highest priority queue that has buffered packets FCFS
    - Round robin
        - Arriving traffic classified, queued by class
        - Server cyclically repeatedly scans class queues and sends one complete packet from each class in turn
    - Weighted fair queueing
        - Generalized Round Robin
        - Each class i has weight w_i and gets weighted amount of service in each cycle: $\frac{w_i}{\sum_j w_j}$
        - Minimum bandwidth guarantee

Network Neutrality

- Definition:
    - Technical: how an ISP should share/allocate its resources
        - Packet scheduling and buffer management are the mechanisms
    - Social, economic principles
        - Protecting free speech
        - Encouraging innovation, competition
    - Enforced legal rules and policies
- US rules:
    - No blocking (of lawful and non-harmful content)
    - No throttling
    - No paid prioritization

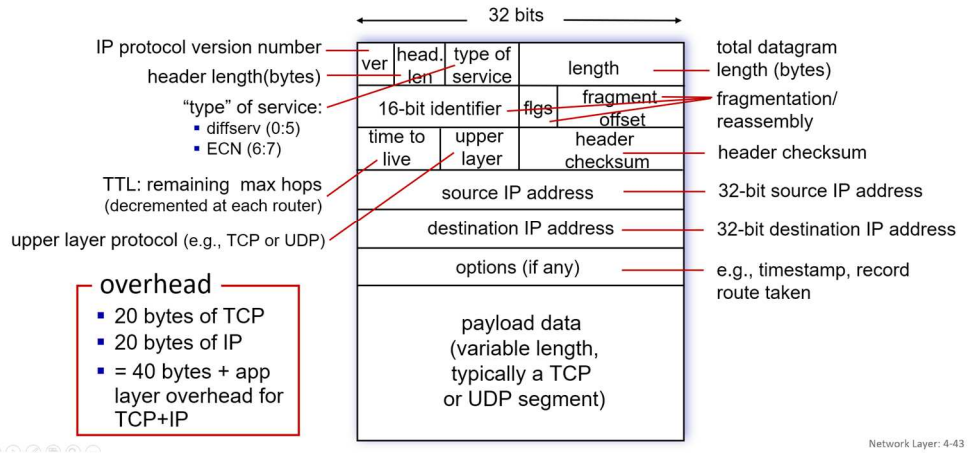
IP – the Internet Protocol

- Network layer – Internet
    - Host, router network layer functions:



NET – Summary

- IP Datagram format



- IP addressing – introduction
  - IP address: 32 bit identifier associated with each host or router interface
  - Interface: connection between host/router and physical link
    - Routers typically have multiple interfaces
    - Host typically has one or two interfaces (WLAN, LAN)
- Subnets:
  - Explanation:
    - Device interfaces that can physically reach each other **without passing through an intervening router**
  - IP addresses have a structure:
    - Subnet part: devices in same subnet have common high order bits
    - Host part: remaining low order bits
  - Recipe for defining subnets:
    - Detach each interface from its host or router -> creates islands of isolated networks
    - Each isolated network is called subnet

IP addressing

- CIDR – Classless InterDomain Routing
  - Subnet portion of address of arbitrary length
  - Address format: a.b.c.d/x where x is number of bits in subnet portion of address
- How to get IP – Question split into:
  - How does a host get an IP address within network
  - How does a network get IP address
- How does host get IP
  - Hard-coded by sysadmin in config file
  - DHCP: Dynamic Host Configuration Protocol
- DHCP:
  - Key concepts:
    - Goal: host dynamically obtains IP address from network server when it joins network
    - Can renew its lease on address in use
    - Allows reuse of address
    - Support for mobile users who join/leave network
  - Overview:
    - Host broadcasts DHCP discover msg
    - DHCP server responds with DHCP offer msg
    - Host requests IP address: DHCP request msg
    - DHCP server sends address: DHCP ack msg
  - More than IP addresses

        - DHCP can return more than just allocated IP addresses on subnet:
            - Address of first-hop router for client
            - Name and IP address of DNS server
            - Network mask
- how does network get IP address?
    - Gets allocated portion of providers ISP's address space
- Hierarchical addressing: route aggregation
    - Allows efficient advertisement of routing information
- How does ISP get block of addresses?
    - ICANN – Internet Corporation for Assigned Names and Numbers
        - Through 5 regional registries (RRs)
        - Manages DNS root zone including delegation or individual management

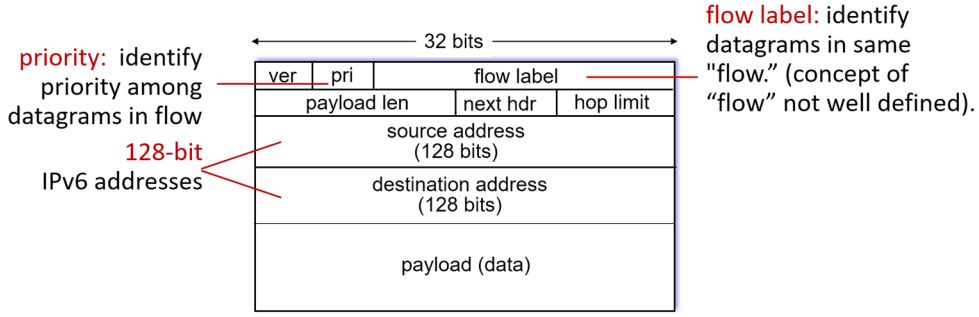
IP – the Internet Protocol

NAT – network address translation

- All devices in local network share just ONE IPv4 address as far as outside world is concerned
  - All datagrams leaving local network have same source NAT IP address but different source port numbers
  - Datagrams with source or destination in this network have 10.0.0/24 addresses for source and destination
- All devices in local network have 32 bit addresses in private IP address space that can only be used in local network
- Advantages:
  - Just one IP address needed from provider ISP for all devices
  - Can change addresses of host in local network without notifying outside world
  - Can change ISP without changing addresses of devices in local network
  - Security: devices inside local net not directly addressable and visible by outside world
- Implementation: NAT router must:
  - Outgoing datagrams: replace (source IP, port) of every outgoing datagram to (NAT IP, new port)
  - Remember (in NAT translation table) every translation pair
  - Incoming datagrams: replace (NAT IP, new port) with corresponding (source IP address, port)
- NAT has been controversial:
  - Routers should only process up to layer 3
  - Addresses shortage should be solved by IPv6
  - Violates end-to-end argument
  - NAT traversal: what if client wants to connect so server behind NAT
- But NAT is here to stay:
  - Extensively used in home and institutional nets and cellular nets

IPv6

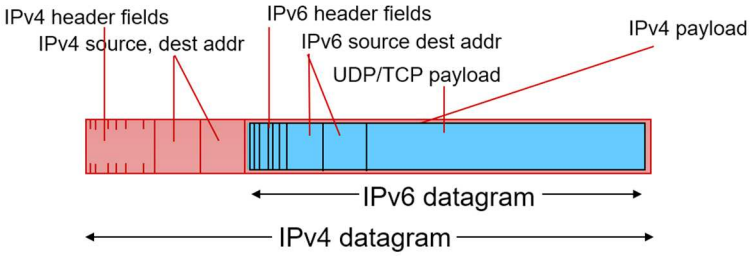
- Motivation:
  - Initial motivation: 32 bit IPv4 address space would be completely allocated
  - Additional motivation:
    - Speed processing/forwarding: 40 byte fixed length header
    - Enable different network layer treatment of flows
- Datagram format:



What's missing (compared with IPv4):
- no checksum (to speed processing at routers)
- no fragmentation/reassembly
- no options (available as upper-layer, next-header protocol at router)

- Transition from IPv4 to IPv6
  - Not all routers can be upgraded simultaneously
    - No flag days
    - How will network operate with mixed IPv4 and IPv6 routers?
  - Tunnelling: IPv6 datagram carried as payload in IPv4 datagram among IPv4 routers (packet within packet)

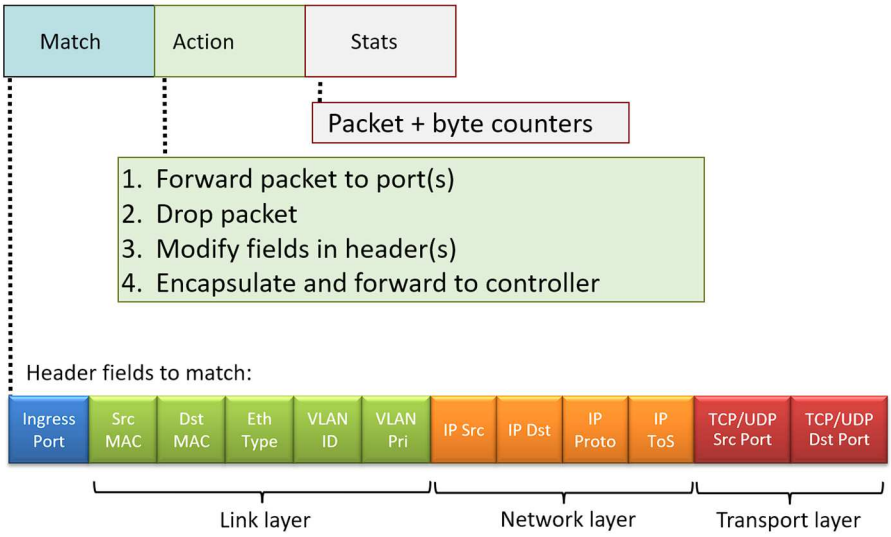
- Tunnelling used extensively in other context (4G/5G)



- IPv6 adoption
  - Google: ~30% of client access services via IPv6
  - NIST: 1/3 of all US government domains are IPv6 capable
  - Long time for deployment, use:
    - 25 years
    - Think of application level changes in last 25 years

Generalized Forwarding, SDN

- Review: each router contains forwarding table
  - Match plus action abstraction: match bits in arriving packet, take action
    - Destination based forwarding: forward based on destination IP address
    - Generalized forwarding:
      - Many header fields can determine action
      - Many actions possible: drop/copy/modify/log packet
- Flow table abstraction:
  - Flow: defined by header field values
  - Generalized forwarding: simple packet-handling rules:
    - Match: patter 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: number of bytes and number of packets
- OpenFlow – flow table entries:



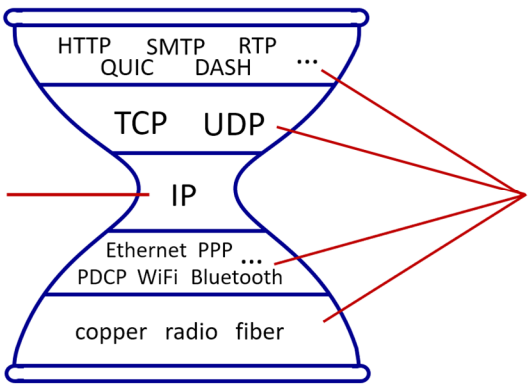
  - Usage for firefall for example
- OpenFlow abstraction:
  - Match+action: abstraction unifies different kinds of devices:
    - Router
      - Match: longest destination IP prefix
      - Action: forward out a link

        - Switch
            - Match: destination MAC address
            - Action: forward of flood
        - Firewall
            - Match: IP addresses and TCP/UDP port numbers
            - Action: permit or deny
        - NAT
            - Match: IP address and port
            - Action: rewrite address and port
- Generalized forwarding: summary
    - Match plus action abstraction: match bits in arriving packet header in any layer and take action
        - Matching over many fields
        - Local actions: drop, forward, modify or send matched packet to controller
        - Program network-wide behaviours
    - Simple for of network programmability
        - Programmable, per packet processing
        - Historical roots: active networking
        - Today: more generalized programming: P4

Middleboxes

- Definition: any intermediary box performing functions apart from normal, standard functions of an IP router on the data path between a source host and destination host
- Examples:
    - NAT: home, cellular, institutional
    - Application-specific: service providers, institutional, CDN
    - Firewall, IDS: corporate, institutional, service providers, ISPs
    - Load balancers: corporate, service providers, data center, mobile nets
    - Caches: service provider, mobile, CDNs
- Initially: proprietary hardware solutions
- Move towards whitebox hardware implementing open API
    - Move away from proprietary hardware solutions
    - Programmable local actions via match+action
    - Move towards innovation/differentiation in software
- SND: centralized control and configuration management often in private/public cloud
- Network functions virtualization (NFV): programmable services over white box networking, computation and storage
- IP hourglass:

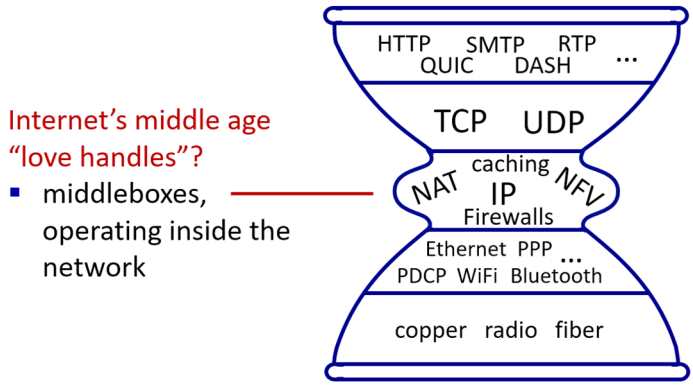
Internet's "thin waist":
- *one* network layer protocol: IP
- *must* be implemented by every (billions) of Internet-connected devices



many protocols in physical, link, transport, and application layers

NET – Summary

- IP hourglass – middle age



Architectural Principles of the Internet

- Three cornerstone beliefs:
  - Simple connectivity
  - IP protocol: that narrow waist
  - Intelligence and complexity at network edge
- The end-end argument
  - Some network functionality can be implemented in network or at network edge

Intelligence and where to find it

- 20th century phone net:
  - Intelligence/computing at network switches
- Internet (pre-2005)
  - Intelligence and computing at edge
- Internet (post-2005)
  - Programmable network devices
  - Intelligence, computing and massive application-level infrastructure at edge

Chapter 5: Network layer – control plane

Introduction

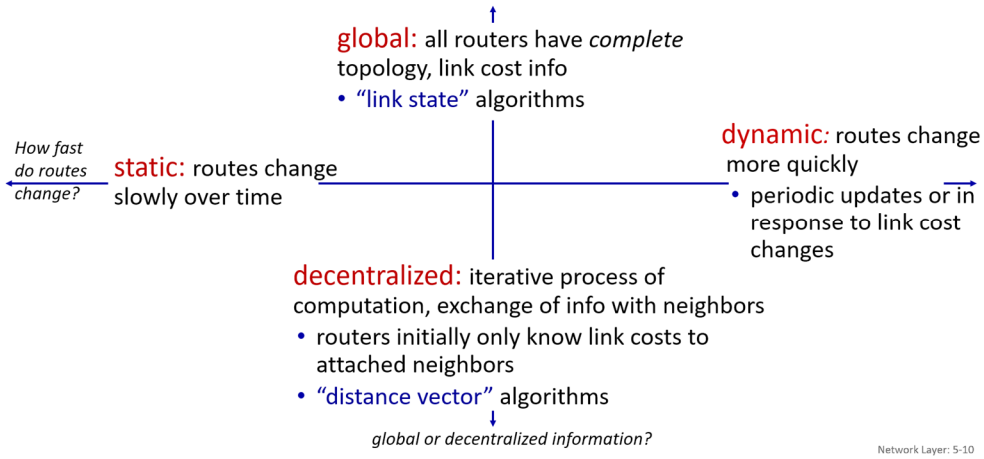
- Network-layer functions
  - Forwarding: move packets from routers input to appropriate router output – data plane
  - Routing: determine route taken by packets from source to destination – control plane
- Two approaches to structuring network control plane:
  - Per-router control (traditional)
  - Logically centralized control (software defined networking)

Routing protocols

- Routing protocol goals: determine good paths from sending host to receiving host
  - Path: sequence of routers
  - "good": least cost, fastest, least congested
- Graph abstraction: link costs
  - $c_{a,b}$: cost of direct link connecting a and b
    cost defined by network operator (always 1, related to bandwidth or related to congestion)
  - Graph: $G = (N, E)$
    - N: set of routers $= \{u, v, w, x, y, z\}$
    - E: set of links $= \{(u, v), (u, x), \dots\}$

NET – Summary

- Routing algorithm classification



Link state

notation

- $c_{x,y}$: direct link cost from node *x* to *y*; = ∞ if not direct neighbors
- *D(v)*: *current* estimate of cost of least-cost-path from source to destination *v*
- *p(v)*: predecessor node along path from source to *v*
- *N'*: set of nodes whose least-cost-path *definitively* known

- Dijkstra's link-state routing algorithm
  - Centralized: network topology, link costs know to all nodes
    - Accomplished via link state broadcast
    - All nodes have same info
  - Computes least cost paths from one node to all other nodes
  - Iterative: after k iterations, know least cost paths to k destinations

```
1  Initialization:
2   N' = {u}                    /* compute least cost path from u to all other nodes */
3   for all nodes v
4    if v adjacent to u         /* u initially knows direct-path-cost only to direct neighbors */
5       then D(v) = c_{u,v}     /* but may not be minimum cost! */
6    else D(v) = ∞
7
8   Loop
9    find w not in N' such that D(w) is a minimum
10   add w to N'
11   update D(v) for all v adjacent to w and not in N' :
12      D(v) = min ( D(v), D(w) + c_{w,v} )
13   /* new least-path-cost to v is either old least-cost-path to v or known
14   least-cost-path to w plus direct-cost from w to v */
15  until all nodes in N'
```

  - Algorithm complexity: n nodes
    - Each of n iteration: need to check all nodes, w not in N
    - $\frac{n*(n+1)}{2}$ comparisons: $O(n^2)$ complexity
    - More efficient implementations possible : $O(n * \log n)$
  - Message complexity :
    - Each router must broadcase its link state information to other n routers
    - Efficient broadcast algorithms: O(n) link crossings to disseminate a broadcast message from one source
    - Each router's message crosses O(n) link: overall complexity: $O(n^2)$
  - Oscillations possible
    - When link costs depend on traffic volume – route oscillations

Distance vector

- Distance vector algorithm
  - Based on Bellman-Ford (BF) equation (dynamic programming):
  - Bellman-Ford equation:
    Let $D_x(y)$ be the cost of least-cost path from x to y.
    Then: $D_x(y) = \min_v\{c_{x,v} + D_v(y)\}$
    - Min taken over all neighbours v of x

        - $D_v(y)$… v's estimated least cost path to y
    - Key idea:
        - From time to time each node sends its own distance vector estimate to neighbours
        - When x receives new DV estimate from any neighbour it updates its own DV using B-F equation
        - Under minor, natural condition the estimate $D_x(y)$ converges to the actual least cost $d_x(y)$
    - Each node:
        - Wait
        - Recompute DV
        - If DV has changed notify neighbours
        - Repeat
    - Iterative, asynchronous: each local iteration caused by:
        - Local link cost change
        - DV update message from neighbour
    - Distributed, self-stopping: neighbour only notified if DV changes
    - State information diffusion
      Iterative communication, spreads with each additional iteration further through the network
- Link cost changes
    - Node detects local link cost change
    - Updates routing info, recalculates local DV
    - If DV changes, notify neighbours
    - Good news travel fast
    - Bad news travels slow – count to infinity problem

Comparison

- Message complexity
    - LS: n routers $O(n^2)$ messages sent
    - DV : exchange between neighbours, convergence time varies
- Speed of convergence
    - LS $O(n^2)$ algorithm, $O(n^2)$ messages – may have oscillations
    - DV: convergence time varies
        - May have routing loops
        - Count to infinity problem
- Robustness: what happens if router malfunctions or is compromised
    - LS:
        - Router can advertise incorrect link cost
        - Each router computes only its own table
    - DV:
        - Router can advertise incorrect path cost: black-holing
        - Each routers table used by others: error propagate through network

Intra-ISP routing – OSPF

- Making routing scalable
    - Our routing study thus far – idealized
        - All routers identical
        - Network flat
        - Not true in practice
    - Scale: billions of destinations:
        - Cannot store all destinations in routing tables
        - Routing table exchange would swamp links

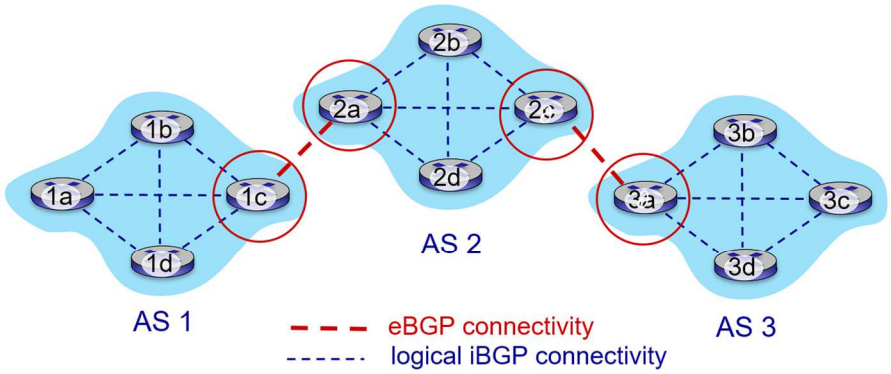
- Administrative authority:
  - Internet: a network of network
  - Each network admin may want to control routing in own network

- Internet approach to scalable routing
  - Aggregate routers into regions known as “autonomous systems” (AS) (aka domains)
  - Intra-AS – routing among within same AS
    - All routers in AS must run same intra-domain protocol
    - Routers in different AS can run different intra-domain routing-protocols
    - Gateway router: router at edge of network – has links to “foreign” routers
  - Inter-AS – among different AS’es
    - Gateways perform inter-domain routing
- Interconnected ASes
  - Forwarding table configured by intra-and inter-AS routing algorithms
    - Intra-AS routing determine entries for destinations within AS
    - Inter-AS &intra-AS determine entries for external destinations
  - Inter-AS routing
    - role in intra-domain forwarding
      - Inter-domain routing must:
        - Learn which destinations reachable through connected ASes
        - Propagate this reachability info to all routers in own AS
    - Routing within an AS
      - Most common intra-AS routing protocols:
        - RIP: Routing Information Protocol
          - Classic DV – exchanged every 30 seconds
          - No longer widely used
        - EIGRP: Enhanced Interior Gateway Routing Protocol
          - DV based
          - Formerly Cisco-proprietary for decades
        - OSPF: Open Shortest Path First
          - Link-state routing
          - IS-IS protocol essentially same as OSPF
- OSPF
  - “open”: publicly available
  - Classic link-state
    - Each router floods OSPF link-state advertisements to all other routers in entire AS
    - Multiple link cost metrics possible: bandwidth, delay
    - Each router has full topology, uses Dijkstra’s algorithm to compute forwarding table
  - Security: all OSPF messages authenticated
  - Hierarchical OSPF:
    - Two-level hierarchy: local area, backbone
      - Link-state advertisements flooded only in area, or backbone
      - Each node has detailed area topology; only knows direction to reach other destinations

Routing among ISPs: BGP

- BGP (Border Gateway Protocol): the de facto inter-domain routing protocol
- Allows subnet to advertise its existence and the destinations it can reach to rest of Internet
- BGP provides each AS a means to:
  - eBGP: obtain subnet reachability information from neighbouring ASes
  - iBGP: propagate reachability information to all AS-internal routers
  - determine good routes to other networks based on reachability information and policy

- eBGP, iBGP connections



gateway routers run both eBGP and iBGP protocols

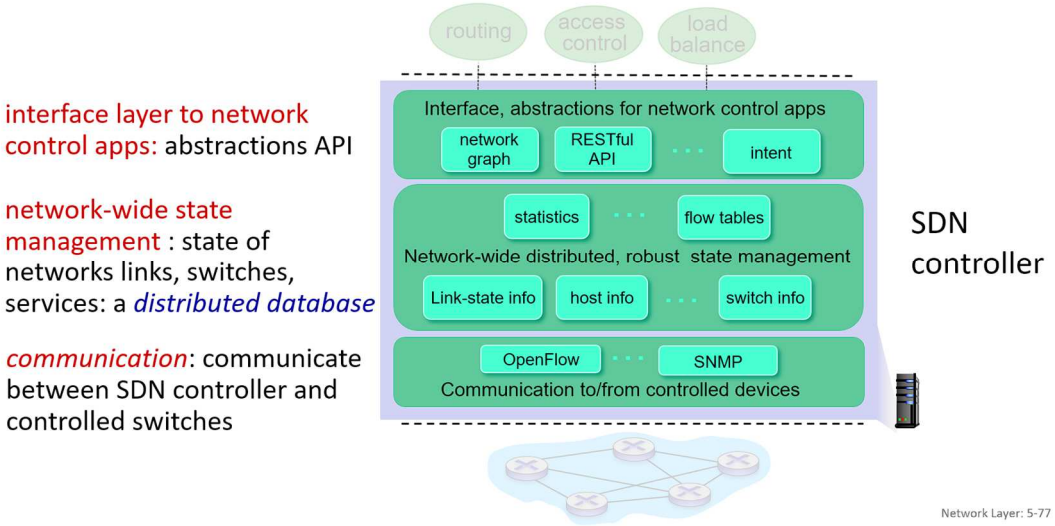
- BGP basics:
  - Session: two BGP routers exchange BGP messages over semi-permanent TCP connection:
    - Advertising paths to different destination prefixes (BGP is a path vector protocol)
  - When AS gateway router advertises path AS,X to other AS gateway router:
    - AS promises to other AS it will forward datagrams towards X
- Path attributes and BGP routes
  - BGP advertised route: prefix+attributes
    - Prefix: destination being advertised
    - Two important attributes:
      - AS-PATH: list of ASes through which prefix advertisement has passed
      - NEST-HOP: indicates specific internal-AS router to next-hop AS
  - Policy-based routing:
    - Gateway receiving route advertisement uses import policy to accept/decline path
    - AS policy also determines whether to advertise path to other neighbouring ASes
    - Gateway may learn about multiple paths to destination:
      - Chooses one and advertises path within AS
- BGP messages:
  - Exchanged between peers over TCP connection
  - Messages:
    - OPEN: opens TCP connection to remote BGP peer and authenticates sending BGP peer
    - UPDATE: advertises new path
    - KEEPALIVE: keeps connection alive in absence of UPDATES; also ACKs OPEN request
    - NOTIFICATION: reports errors in previous msg; also uses to close connection
- Why different Intra-, Inter-AS routing?
  - Policy:
    - Inter-AS: admin wants control over how its traffic routed and who routes through network
    - Intra-AS: single admin, so policy less of an issue
  - Scale:
    - Hierarchical routing saves table size, reduced update traffic
  - Performance:
    - Intra-AS can focus on performance
    - Inter-AS policy dominates over performance
- Hot-potato routing
  - Choose local gateway that has least intra-domain cost
- Achieving policy via advertisements
  - ISP only want to route traffic to/from its customer networks

- BGP route selection
    - Router may learn about more than one route to destination AS, selects route based on:
        - Local preference value attribute: policy decision
        - Shortest AS-PATH
        - Closes NEXT-HOP router: hot potato routing
        - Additional criteria

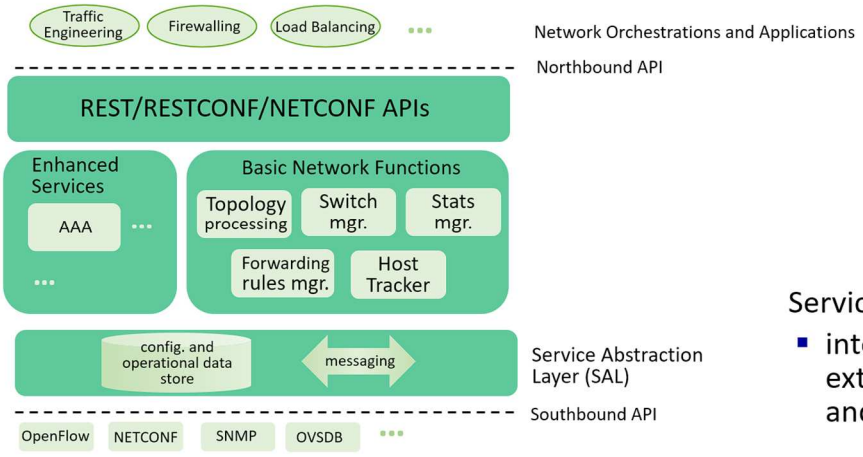
SDN control plane

- Internet network layer: historically implemented via distributed, per-router control approach:
    - Monolithic router contains switching hardware
    - Different middleboxes for different network layer functions:
        - Firewalls, load balancers, NAT boxes,…
- 2005: renewed interest in rethinking network control plane
- Per-router control plane:
    - Individual routing algorithm components in each and every router interact in the control plane to compute forwarding tables
- Software-Defined Networking (SDN) control plane
    - Remote controller computes, installs forwarding tables in routers
- SDN:
    - Why?
        - Easier network management: avoid router misconfigurations, greater flexibility of traffic flows
        - Table-based forwarding allows programming routers
            - Centralized programming easier: compute tables centrally and distribute
            - Distributed programming more difficult:
                - Compute tables as result of distributed algorithm implemented in each router
        - Open implementation of control plane
- Traffic engineering: difficult with traditional routing
    - Specific route requires redefinition of link weights
        - Link weights are only control knobs; not much control
    - Split traffic not possible
    - Different traffic different routing not possible
- SDN
    - data plane switches:
        - Fast, simple, commodity switches implementing generalized data-plane forwarding in hardware
        - Flow table computed and installed under controller supervision
        - API for table-based switch control
        - Protocol for communicating with controller
    - SDN controller:
        - Maintain network state information
        - Interacts with network control applications above via northbound API
        - Interacts with network switches below via southbound API
        - Implemented as distributed system for performance, scalability, fault tolerance and robustness
    - Network-control apps:
        - Brains of control: implement control functions using lower-level services, API provided by SDN controller
        - Unbundled: can be provided by 3rd party: distinct from routing vendor or SDN controller

- Components of SND controller



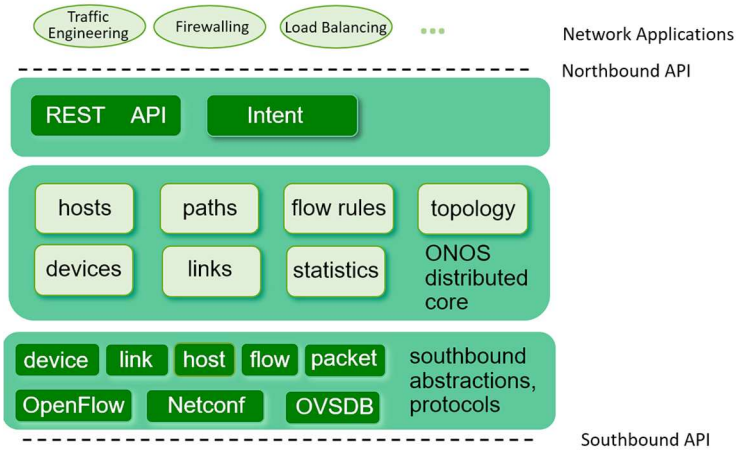
- OpenFlow protocol
  - Operates between controllers, switch
  - TCP used to exchange messages
    - Optional encryption
  - Three classes of OpenFlow messages:
    - Controller-to-switch
    - Asynchronous (switch to controller)
    - Symmetric
  - Distinct from OpenFlow API
    - API used to specify generalized forwarding actions
- OpenFlow controller to switch messages
  - Key controller-to-switch messages:
    - Features: controller queries switch features, switch replies
    - Configure: controller queries/sets switch configuration parameters
    - Modify-state: add, delete, modify flow entries in the OpenFlow tables
    - Packet-out: controller can send this packet out of specific switch port
    - Packet-in: transfer packet to controller. See packet-out message from controller
    - Flow-removed: flow table entry deleted at switch
    - Port status: inform controller of a change on a port
  - OpenDaylight (ODL) controller



Service Abstraction Layer:

- interconnects internal, external applications and services

- ONOS controller



- control apps separate from controller
- intent framework: high-level specification of service: what rather than how
- considerable emphasis on distributed core: service reliability, replication performance scaling

- SDN: selected challenges
  - Hardening the control plane: dependable, reliable, performance scalable, secure distributed system
    - Robustness to failure: leverage strong theory of reliable distributed system for control plane
  - Networks, protocols meeting mission-specific requirements
  - Internet-scaling: beyond a single AS
  - SDN critical in 5G cellular networks
- SDN and the future of traditional network protocols
  - SDN computed versus router computed forwarding tables
  - One could imagine SDN computed congestion control

Internet Control Message Protocol

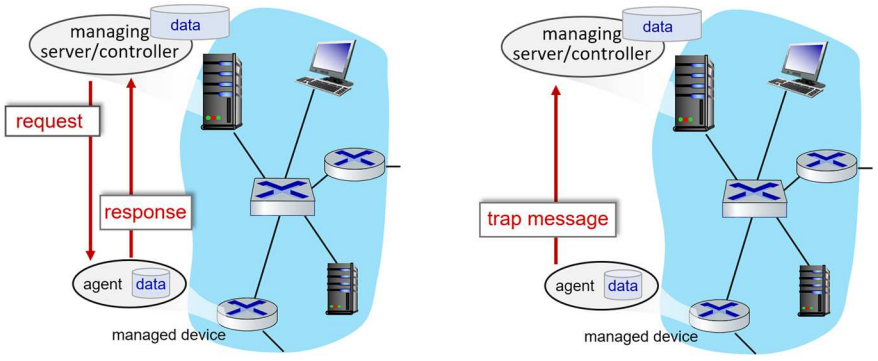
- Used by hosts and routers to communicate network-level information
  - Error reporting: unreliable host, network, port, protocol
  - Echo request/reply
- Network-layer “above” IP:
  - ICMP messages carried in IP datagrams
- ICMP message: type, code plus first 8 bytes of IP datagram causing error
- Traceroute and ICMP
  - Source sends sets of UDP segments to destination
  - Datagram in n-th set arrives to n-th router
    - Router discards datagram and sends source ICMP message
    - ICMP message possibly includes name of router & IP address
  - When ICMP message arrives at source: record RTTs
  - Stopping criteria:
    - UDP segment eventually arrives at destination host
    - Destination returns ICMP “port unreachable” message
    - Source stops

Network management, configuration

- What is network management ?
  - Autonomous systems : 1000s of interacting hardware/software components
  - Other complex systems requiring monitoring, configuration, control

> "Network management includes the deployment, integration and coordination of the hardware, software, and human elements to monitor, test, poll, configure, analyze, evaluate, and control the network and element resources to meet the real-time, operational performance, and Quality of Service requirements at a reasonable cost."

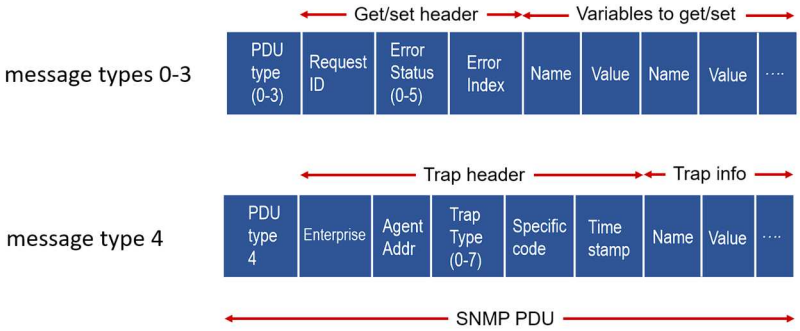
- Components of network management
  - Managing server:
    application, typically with network managers in the loop
  - Network management protocol:
    - used by managing server to query, configure, manage device
    - used by devices to inform managing server of data and events
  - Managed device:
    equipment with manageable configurable hardware, software components
  - Data:
    device state configuration data, operational data, device statistics
- Network operator approaches to management
  - CLI
    - Operator issues direct to individual devices
  - SNMP/MIB
    - Operator queries/sets devices data using Simple Network Management Protocol
  - NETCONF/YANG
    - More abstract, network-wide, holistic
    - Emphasis on multi-device configuration management
    - YANG: data modelling language
    - NETCONF: communicate YANG-compatible actions/data to/from/among remote devices
- SNMP protocol
  Two ways to convey MIB info, commands:



  - Message types:

Message type	Function
GetRequest GetNextRequest GetBulkRequest	manager-to-agent: "get me data" (data instance, next data in list, block of data).
SetRequest	manager-to-agent: set MIB value
Response	Agent-to-manager: value, response to Request
Trap	Agent-to-manager: inform manager of exceptional event

- Message format

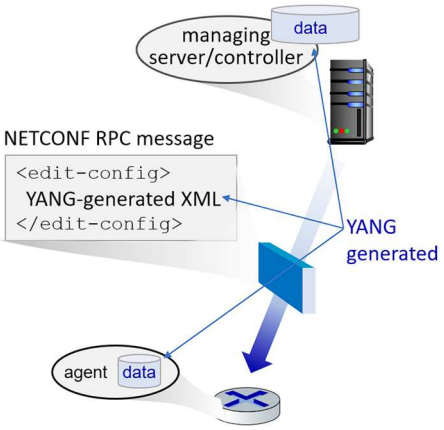


  - Management Information Base (MIB)
    - Manages device's operational data
    - Gathered into device MIB module
    - Structure of Management Information (SMI): data definition language

- NETCONF overview:
  - Goal: actively manage/configure devices network-wide
  - Operates between managing server and managed network devices
    - Actions: retrieve, set, modify, activate configurations
    - Atomic-commit actions over multiple devices
    - Query operational data and statistics
    - Subscribe to notifications from devices
  - Remote procedure call (RPC) paradigm
    - NETCONF protocol messages encoded in XML
    - Exchanged over secure, reliable transport protocol
  - Selected NETCONF operations

NETCONF	Operation Description
<get-config>	Retrieve all or part of a given configuration. A device may have multiple configurations.
<get>	Retrieve all or part of both configuration state and operational state data.
<edit-config>	Change specified (possibly running) configuration at managed device. Managed device <rpc-reply> contains <ok> or <rpcerror> with rollback.
<lock>, <unlock>	Lock (unlock) configuration datastore at managed device (to lock out NETCONF, SNMP, or CLIs commands from other sources).
<create-subscription>, <notification>	Enable event notification subscription from managed device

- YANG
  - Data modelling language used to specify structure, syntax, semantics of NETCONF network management data
  - XML document describing device, capabilities can be generated from YANG description
  - Can express constraints among data that must be satisfied by a valid NETCONF configuration



  -

Chapter 6 – The Link Layer and LANs

Introduction

- Layer-2 packet: frame, encapsulates datagram
- Link layer has responsibility of transferring datagram from one node to physical adjacent node over a link

Context

- Different link protocols over different links
- Each link protocol provides different services

Services

- Framing, link access:
  - Encapsulate datagram into frame, adding header, trailer
  - Channel access if shared medium
  - MAC address in frame headers identify source, destination (DIFFERENT FROM IP)
- Reliable delivery between adjacent nodes
  - We already know how to do this
  - Seldom used on low bit-error links
  - Wireless links: high error links
- Flow control:
  - Pacing between adjacent nodes
- Error detection
  - Errors caused by signal attenuation, noise
  - Receiver detects errrors, signals retransmission, or drop frames
- Error correction
  - Receiver identifies and corrects bit errors without retransmission
- Half-duplex and full-duplex
  - With h-d nodes at both ends of link can transmit, but not at the same time
  - F-d: same time transmitting

Where is the link layer implemented:

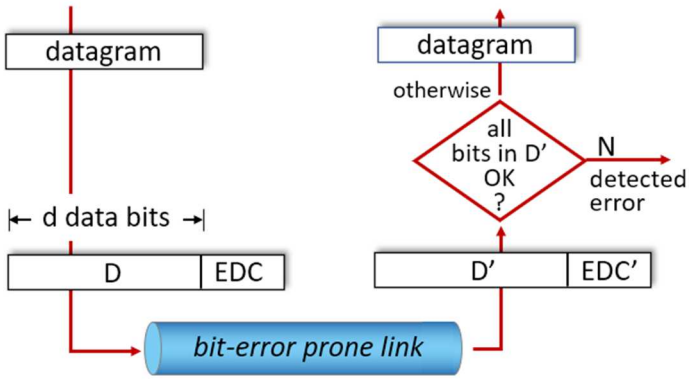
- In each-and-every host
- Link layer implemented in network interface card (NIC) or on a chip
- Attaches into hosts system buses
- Combination of hardware, software and firmware

Interface communicating

- Sending side:
  - Encapsulate datagram in frame
  - Add error checking bits, reliable data transfer, flow control, etc.
- Receiving side:
  - Looks for errors, reliable data transfer, flow control, etc.
  - Extracts datagram, passes to upper layer at receiving side

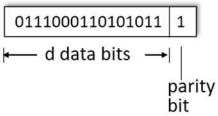
Error detection, correction

- Error detection
  - EDC: error detection and correction bits
  - D: data protected by error checking, may include header fields



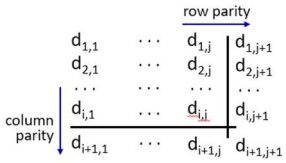
Parity checking

single bit parity:
- detect single bit errors



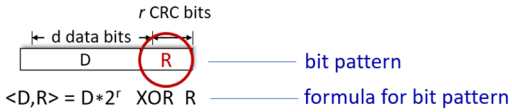
Even parity: set parity bit so there is an even number of 1's

two-dimensional bit parity:
- detect *and correct* single bit errors



Check out the online interactive exercises for more

- Internet checksum
  - Goal: detect errors in transmitted segment
  - Sender:
    - Trad contents of UDP segment as sequence of 16 bit integers
    - Checksum: addition of segment content
    - Checksum value put into UDP checksum field
  - Receiver:
    - Compute checksum of received segment
    - Check if computed checksum equals checksum field value
      - Equal – error
      - Not equal: no error detected
- Cyclic Redundancy Check (CRC)
  - more powerful error-detection coding
  - D: data bits (given, think of these as a binary number)
  - G: bit pattern (generator), of *r+1* bits (given)



goal: choose *r* CRC bits, R, such that <D,R> exactly divisible by G (mod 2)
- receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
- can detect all burst errors less than r+1 bits
- widely used in practice (Ethernet, 802.11 WiFi)

- Example (Generally division of D with adding 0s at end and division better)

We want:

$D{\cdot}2^r \ XOR \ R = nG$

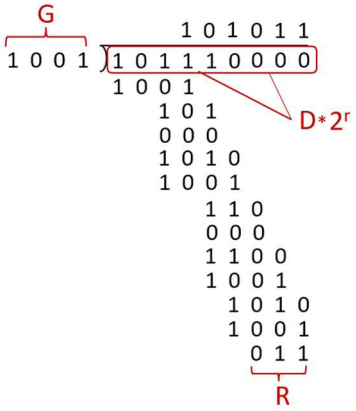
or equivalently:

$D{\cdot}2^r = nG \ XOR \ R$

or equivalently:

if we divide $D{\cdot}2^r$ by G, want remainder R to satisfy:

$$R = remainder\left[\frac{D{\cdot}2^r}{G}\right]$$



Multiple access protocols

- Two types of links:
  - Point to point
    - Between ethernet switch, host
    - PPP for dial-up access
  - Broadcast (shared wire or medium)
    - Old-fashioned ehternet
- Single shared broadcast channel
- Two or more simultaneous transmissions by nodes: interference
  - Collision if node receives two or more signals at the same time
- Multiple access protocol:
  - distributed algorithm that determines how nodes share channel and when node can transmit
  - communication about channel sharing must use the channel itself

MAC protocols: taxonomy

- three broadcasting classes:
  - channel partitioning
    - divide channel into smaller pieces
    - allocate piece to node for exclusive use
  - random access
    - channel not divided, allow collisions
    - recover from collisions
  - taking turns
    - nodes take turns, but nodes with more to send can take longer turns
- Channel partitioning:
  - TDMA – time division multiple access
    - Access channel in rounds
    - Each station gets fixed length slot
    - Unused slots go idle
  - FDMA – frequency division multiple access
    - Channel spectrum divided into frequency bands
    - Each station assigned fixed frequency
    - Unused transmission time in frequency go idle
- Random access protocols:
  - When node has packet to send transmit at full channel data rate R
  - Two or more transmitting nodes : collision
  - Random access MAC protocol specifies:
    - How to detect collisions
    - Hot to recover from them

- Examples:
  - Slotted ALOHA
    - Assumptions:
      - All frames same size
      - Time divided equally
    - Operation:
      - Node obtains fresh frame
        - If no collision: node can send next frame in next slot
        - If collision: node retransmits frame in each subsequent lot with probability p until success
    - Pros:
      - Single active node can transmit at full rate
      - Highly decentralized: only slots in nodes need to be in sync
    - Cons:
      - Collisions, wasting slots
      - Nodes may be able to detect collisions in less time than to transmit packet
      - Clock synchronization
    - Efficiency at most 37%
  - Pure ALOHA:
    - Unslotted Aloha: simpler, no synchronization – transmit immediately
    - Collision probability increases with no synchronisation
    - Efficiency lower than slottet, only 18%
  - CSMA (carrier sense multiple access)
    - Simple CSMA: listen before transmit
      - Send when channel idle
    - CSMA/CD: CSMA with collision detection:
      - Collision detected within short time
      - Colliding transmissions aborted, reducing channel wastage
      - Collision detection easy in wired, difficult with wireless
    - Collisions:
      - Can still occur with carrier sensing
      - Collision: entire packet transmission time wasted
    - Ethernet CSMA/CD algorithm
      - NIC receives datagram from network layer, creates frame
      - IF NIC senses channel:
        - If idle: start frame transmission
        - If busy: wait until channel idle, then transmit
      - If NIC transmits entire frame without collision, NIC done with frame
      - If NIC detects another transmission while sending: abort, send jam signal
      - After aborting, NIC enters binary (exponential) backoff

- Taking turns protocol
  - Look for best of both channel partitioning and random access
  - Types:
    - Polling:
      - Master node invites other nodes to transmit in turn
      - Typically used with dumb devices
      - Concerns:
        - Polling overhead
        - Latency
        - Single point of failure

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

Cable access network: FDM, TDM and random access

- Multiple downstream FDM channels
    - Single CMTS transmits into channels
- Multiple upstream channels
    - Multiple access: all users contend for certain upstream channel time slots, others assigned TDM
- DOCSIS: data over cable service interface specification
    - FDM over upstream, downstream frequency channels
    - TDM upstream: some slot assigned, some have contention
        - Downstream MAP frame: assign upstream slots
        - Request for upstream slots transmitted random access in selected slots

Summary

- Channel partitioning by time, frequency or code
    - Time Division, Frequency Division
- Random access (dynamic)
    - Carrier sensing easy in wire, hard in wireless
    - CSMA/CD for Ethernet
- Taking turns:
    - Polling from central side, token passing
    - Bluetooth, token ring, etc.

LANs

MAC addresses

- 32 bit IP address:
    - Network-layer address for the interface
    - Used for layer 3 forwarding
- MAC address:
    - Function: used locally to get frame from one interface to another physically connected interface
    - 48 bit MAC address burned in NIC ROM
- Each interface on LAN
    - Has unique 48 bit MAC address
    - Has a locally unique 32 bit IP address
- Mac address allocation administered by IEEE
- Manufacturer buys portion of MAC address space
- MAC flat address: portability
    - Can move interface from one LAN to another
    - Recall IP address not portable: depends on IP subnet to which node is attached
- ARP: address resolution protocol
    - How to determine interface MAC address knowing the IP?
    - →ARP table: each IP node on LAN has table
        - IP/MAC address mappings for some LAN nodes
        - TTL time after which mapping will be forgotten
        - Find MAC by broadcase ARP query with FF-FF-FF-FF-FF-FF mac address port

NET – Summary

- Routing to another subnet: addressing
    - Walkthrough: sending a datagram from A to B via R
        - Focus on addressing – at IP and MAC layer leves
        - Assume that:
            - A knows B's IP address
            - A knows IP address of first hop router, R
            - A knows R's MAC address

Ethernet

- Dominant wired LAN technology:
- Physical topology
    - Bus: all nodes in same collision domain
    - Switched: prevails today
        - Active link-layer 2 switch in center
        - Each spoke runs a ethernet protocol
- Frame structure
    - Sending interface encapsulates IP datagram in Ethernet frame

    - Preamble:
        - Used to synchronize receiver, sender clock rates
    - Addresses: 6 byte structure, destination MAC address
    - Type: indicates higher layer protocol
        - Mostly IP but others possible
        - Used to demultiplex up at receiver
    - CRC: cyclic redundancy check at receiver
- Unreliable and connectionless:
    - Connectionless: no handshaking between sending and receiving NICs
    - Unreliable: receiving NIC doesn't send ACKs or NAKs to sending NIC
    - Ethernet MAC protocol: unslotted CSMA/CD with binary backoff
- 802.3 ethernet standards: link & physical layers
    - Many different ethernet standards
        - Common MAC protocol and frame format

Switches

- Switch is a link-layer device: takes an active role
    - Store, forward ethernet frames
    - Examine incoming frame's MAC address, selective forwarding
- Transparent: host unaware of presence of switches
- Plug-and-play, self-learning
- Multiple simultaneous transmissions:
    - Hosts have dedicated, direct connection to switch
    - Switches buffer packets
    - Ethernet protocol used on each incoming link
        - No collision, full duplex
        - Each link is its own collision domain
    - Switching: A-to-A' and B-to-B' can transmit simultaneously without collisions
- Switch forwarding table
- Self-learning
    - Switch learns which hosts can be reached through which interfaces

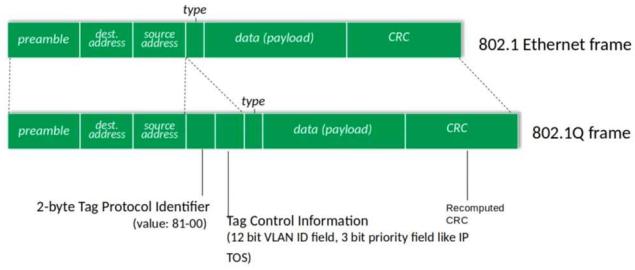
NET – Summary

- Frame filtering/forwarding
  - Forward to specific interface if entry in table, else flood to all interfaces except sender
  - Record incoming link and MAC address
- Switches vs. Routers
  - Both are store-and-forward
    - Routers: network-layer devices
    - Switches: link-layer devices
  - Both have forwarding tables
    - Routers: compute tables using routing algorithms, IP addresses
    - Switches: learn forwarding using flooding, learning MAC addresses

VLANs

- Motivation:
  - Single broadcast domain
    - Scaling
    - Efficiency, security, privacy
  - Administrative issues:
    - Different physical attachment that logical attachment
- VLAN – Virtual Local Area Network
  - Switch supporting VLAN capabilities can be configured to define multiple virtual LANs over single physical LAN infrastructure
- Port-based VLAN: switch ports grouped so that single physical switch operates multiple virtual switches
  - Traffic isolation: frames to/from ports can only reach ports
  - Dynamic membership: ports can be dynamically assigned among VLANs
  - Forwarding between VLANs: done via routing
- VLANs spanning multiple switches:
  - Trunc port: carries frames between VLANs defined over multiple physical switches

802.1Q VLAN frame format



Link virtualization – MPLS

- MPLS – Multiprotocol label switching
  - Goal: high-speed IP forwarding among network of MPLS capable routers, using fixed length label
    - Faster lookup using fixed length identifier
    - Borrowing ideas from Virtual Circuit approach
    - But IP datagram still keeps IP address
- MPLS capable routers
  - Aka label-switched router
  - Forward packet to outgoing interface based only on label value
  - Flexibility: MPSL forwarding decisions can differ from those of IP
- MPSL versus IP paths
  - IP routing: path to destination determined by destination address alone
  - MPSL routing: path to destination can be based on source and destination address
    - Flavour of generalized forwarding
    - Fast reroute: precompute backup routes in case of failure

- MPLS signaling
  - Modify OSPF, IS-IS link-state flooding protocols to carry info used by MPLS routing
  - Entry MPSL router uses RSVP-TE signaling protocol to set up MPLS forwarding at downstream routers
- MPLS forwarding tables

Data center networking

- 10's to 100's of thousands of hosts, often closely coupled in close proximity
- Challenges:
  - Multiple applications, each serving massive number of clients
  - Reliability
  - Managing/balancing load
- Network elements:
  - Border routers
  - Tier 1 switches
  - Tier 2 switches
  - Top of Rack (TOR) switch
  - Server racks
- Multipath:
  - Rich interconnection among switches, racks:
    - Increased throughput between racks
    - Increased reliability via redundancy
- Application-layer routing
  - Load balancer:
    - Receives external client requests
    - Directs workload within data center
    - Returns results to external client
- Protocol innovations
  - Link layer:
    - RoCE: remote DMA over Converged Ethernet
  - Transport layer:
    - ECN used in transport-layer congestion control
    - Experimentation with hop-by-hop congestion control
  - Routing, management:
    - SDN widely used within/among organizations datacenters
    - Place related services, data as close as possible to minimize tier-2, tier-1 communication

Chapter 7 – Wireless and Mobile Networks

Introduction

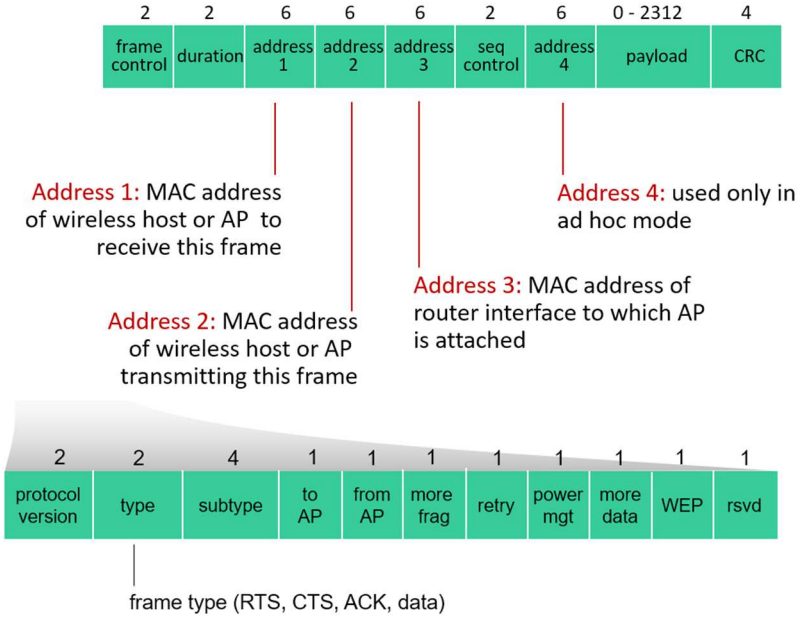
- Wireless and Mobile Networks: context:
  - More wireless phone subscibers than fixed phone subscribers
  - More mobile than fixed connected devices
  - Two important challenges:
    - Wireless
    - Mobility
- Elements of a wireless network
  - Wireless hosts
  - Base station:
    - Typically connected to wired network
    - Relay functionality
  - Wireless link
    - Typically used to connect mobiles to base station, also used as backbone link
    - Multiple access protocol coordinates link access
    - Various transmission rates and distances, frequency bands
  - Infrastructure mode
    - Base station connects mobiles into wired network
    - Handoff: mobile changes base station providing connection into wired network
  - Ad hoc mode:
    - No base stations
    - Nodes can only transmit to other nodes within link coverage
    - Nodes organize themselves into a network: route among themselves
- Wireless network taxonomy:
  - Single hop:
    - Infra: Hosts connect to base station which connects to larger internet
    - No infra: No base station, no connection to larger internet
  - Multiple hop:
    - Infra: host may have to relay through several wireless nodes to connect to larger Internet
    - No infra: no base station, no connection to larger Internet. May have to relay.

Wireless

- Wireless link characteristics
  - Important differences from wired link:
    - Decreased signal strength: radio signal attenuates as it propagates through matter
    - Interference from other sources: wireless network frequencies shared by many devices
    - Multipath propagation: radio signal reflects off objects ground, arriving at destination at slightly different times
  - SNR: signal-to-noise-ratio
    - Larger SNR: easier to extract signal from noise
  - SNR versus BER tradeoff
    - Given physical layer: increase power->increase SNR->decrease BER
    - Given SNR: choose physical layer that meets BER requirement, giving highest throughput
  - Multiple wireless senders, receivers create additional problems
    - Hidden terminal problem
    - Signal attenuation
- Code Division Multiple Access (CDMA)
  - Unique code assigned to each user
    - All users share same frequency, but each user has own chipping sequence to encode data

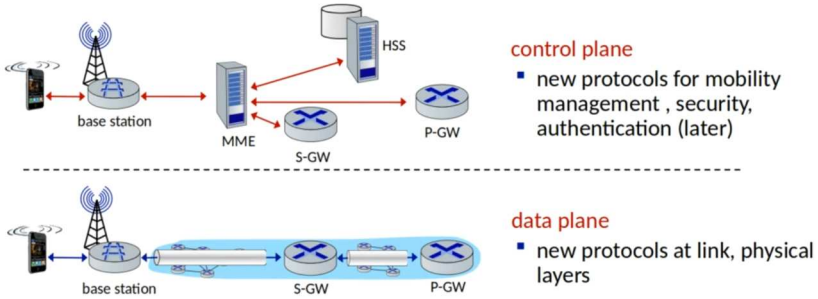
        - Allows multiple users to coexist and transmit simultaneously with minimal interference
    - Encoding: inner product (original data)x(chipping sequence)
    - Decoding: summed inner-product: (encoded data)x(chipping sequence)
- Wireless LAN
    - All use CSMAUCA for multiple access and have base-station and ad-hoc network versions
    - 802.11 LAN architecture
        - Wireless host communicates with base station (=access point)
        - Basis Service Set(BSS) in infrastructure mode constrains:
            - Wireless host
            - Access point: base station
            - Ad hoc mode: hosts only
    - 802.11: channels and association
        - Spectrum divided into channels at different frequencies
        - Arriving host: must associate with an AP
    - Passive/active scanning:
        - Passive scanning:
            - Beacon frames sent from Aps
            - Association request frame sent: H1 to selected AP
            - Association response frame sent from selected AP to H1
        - Active scanning:
            - Probe request frame broadcast from H1
            - Probe response frames sent from Aps
            - Association request frame sent: H1 to selected AP
            - Association response frame sent from AP to H1
    - Multiple access:
        - Avoid collisions: 2^+ nodes transmitting at the same time
        - CSMA – sense before transmitting
        - No collision detection
    - CSMA/CA
        - Sender
            - If sense channel idle for DIFS then transmit frame
            - If sense channel busy then backoff
        - Receiver
            - If frame received return ACK after SIFS
    - Avoiding collisions:
        - Idea: sender reserves channel use for data frames using small reservation packets
            - Sender first transmits small request to send (RTS) packet to BS using CSMA
            - BS broadcast clear-to-send CTS in response to RTS
            - CTS heard by all nodes

- Frame – addressing:



  - Advanced capabilities
    - Rate adaptation
      - Base station, mobile dynamically change transmission rate as mobile moves, SNR varies
    - Power management
      - Node-to-AP
        - AP knows not to transmit frames to this node
        - Node wakes up before next beacon frame
      - Beacon frame: contains list of mobiles with AP-to-mobile frames waiting to be sent
- Personal area network: Bluetooth
  - Less than 10m diameter
  - Replacement for cables
  - Ad hoc: no infrastructure
  - TDM
  - FDM
  - Parked mode: clients can “go to sleep” and later wake up
  - Bootstrapping: nodes self-assemble into piconet
- Cellular Networks
  - 4G/5G
    - The solution for wide-area mobile Internet
    - Widespread deployment/use
    - Up to 100’s Mbps
  - Similarities to wired Internet
    - Edge/core distinction
    - Global cellular network – network of networks
    - Widespread use of protocols
    - Interconnected to wired Internet
  - Differences
    - Different wireless link layer
    - Mobility as a 1st class service
    - User identity(SIM card)
    - Business model
  - Elements of 4G LTE architecture

        - Mobile device
        - Radio access network -|- all-IP Enhanced Packet Code (EPC)
        - Base station
        - Home Subscriber Service
        - Serving Gateway (S-GW), PDN Gateway (P-GW)
            - Lie on data path from mobile to/from internet
        - Mobility Management Entity
            - Device authentication
    - LTE: data plane control plane separation



    - LTE data plane protocol stack: first hop
        - LTE link layer protocols:
            - Packet Data Convergence: header compression, encryption
            - Radio Link Control (RLC) Protocol: fragmentation/reassembly, reliable data transfer
            - Medium Access: requesting, use of radio transmission slots
        - LTE radio access network
            - Downstream channel: FDM, TDM within frequency channel
                - Orthogonal: minimal interference between channels
            - Upstream: FDM, TDM similar to OFDM
    - LTE data plane control stack : packet core:
        - Tunnelling
    - …: associating with a BS
        - BS broadcast for synch every 5ms
        - Mobile finds primary synch signal, then 2nd
        - Mobile selects which BS to associate with
    - Global cellular network: a network of IP networks
        - All IP:
            - Carriers interconnect with each other and public internet at exchange points
        - Home network HSS
    - 5G
        - Goal: 10x increase in peak bitrate, 10x decrease in latency, 100x increase in traffic capacity
        - Millimeter wave frequencies: much higher data rates, but shorter distances

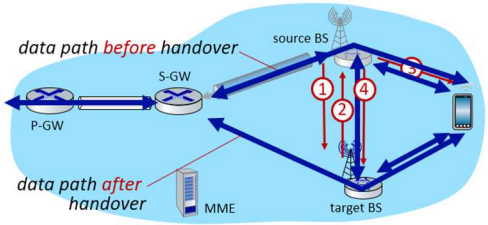
Mobility:

- Mobility approaches:
    - Let network (routers) handle it:
        - Routers advertise well-known name, address or number of visiting mobile node via usual routing table exchange
        - Internet routing could do this already with no changes! Routing tables indicate where each mobile located via longest prefix match
          →Not scalable

- Let end-systems handle it: functionality at the edge
  - Indirect routing: communication from correspondent to mobile goes through home network, then forward to remote mobile
  - Direct routing: correspondent gets foreign address of mobile, send direct to mobile.
- Mobility with indirect routing: comments
  - Triangle routing:
    - Inefficient when correspondent and mobile are in the same network
  - Mobile moves among visited networks: transparent to correspondent
    - Register in new visited network
    - New visited network registers with home HSS
    - Datagrams continue to be forwarded from home network to mobile in new network
    - On-going connections can be maintained!
- Mobility with direct routing: comments
  - Overcomes triangle routing inefficiencies
  - Non-transparent to correspondent: correspondent must get care of address from home agent

Mobility in 4G

- Major mobility tasks
  - Base station association
    - Covered earlier
    - Mobile provides IMSI – identifying itself, home network
  - Control plane configuration
    - MME; home HSS establish control-plane state – mobile is in visited network
  - Data-plane configuration:
    - MME configures forwarding tunnels for mobile
    - Visited, home network establish tunnels from home P-GW to mobile
  - Mobile handover:
    - Mobile device changes its point of attachment to visited network
- Configuring LTE control-plane elements:
  - Mobile communicate with local MME via BS control-plane channel
  - MME uses mobile's IMSI info to contact mobile's home HSS
    - Retrieve authentication, encryption, network service information
    - Home HHS knows mobile now resident in visited network
  - BS, mobile select parameters for BS-mobile data-plane radio channel
- Configuring data-plane tunnels for mobile
  - S-GW to BS tunnel: when mobile changes base stations, simply change endpoint IP address of tunnel
  - S-GW to home P-GW tunnel: implementation of indirect routing
  - Tunnelling via GTP: mobile datagram to streaming server encapsulated using GTP inside UDP, inside datagram
- Handover between BSs in same cellular network

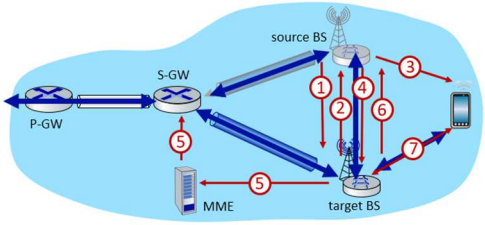


(1) current (source) BS selects target BS, sends *Handover Request message* to target BS

(2) target BS pre-allocates radio time slots, responds with HR ACK with info for mobile

(3) source BS informs mobile of new BS
- mobile can now send via new BS - handover *looks* complete to mobile

(4) source BS stops sending datagrams to mobile, instead forwards to new BS (who forwards to mobile over radio channel)



⑤ target BS informs MME that it is new BS for mobile
- MME instructs S-GW to change tunnel endpoint to be (new) target BS

⑥ target BS ACKs back to source BS: handover complete, source BS can release resources

⑦ mobile's datagrams now flow through new tunnel from target BS to S-GW

- Mobile IP
  - Mobile IP architecture standardized ~20 years ago
  - Mobile IP architecture:
    - Indirect routing to node using tunnels
    - Mobile IP home agent: combined roles of 4G HSS and home P-GW
    - Mobile IP foreign agent: combines roles of 4G MME and S-GW
    - Protocols for agent discovery in visited network, registration of visited location in home network via ICMP extensions
- Wireless, mobility: impact on higher layer protocols
  - Logically impact should be minimal
    - Best effort service model remains unchanged
    - TCP and UDP can run over wireless
  - But performance-wise:
    - Packet loss/decay and handover loss
    - TCP interprets loss as congestion, will decrease congestion window unnecessarily
    - Delay impairments for real-time traffic
    - Bandwidth a scare resource for wireless links