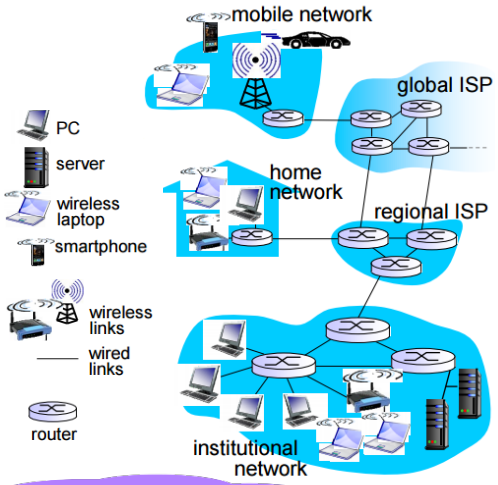
Didrik Kvanvik

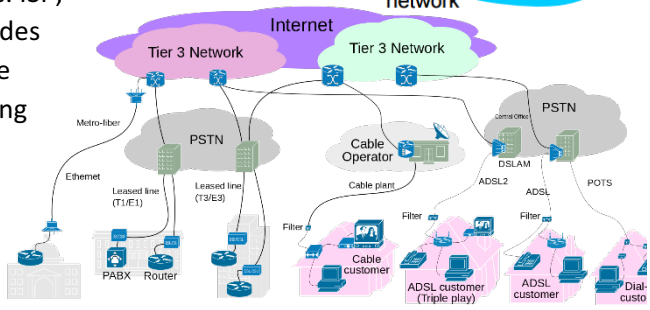
INF142 Notes for exam

Chapter 1

1.1 What's the Internet?

- Internet: millions of connected computing devises.
  - Hosts = end systems
  - Running network apps
- Communication links
  - Fiber, copper, radio, satellite
  - Transmission rate: bandwidth which is the bit-rate of available or consumed information capacity expressed in metric multiples of bits per second.
- Packet switches: forward packets
  - Routers and switches





Internet is a network of networks. That is interconnected ISPs. ISP, Interconnected service provider, is an organization that provides services for services for accessing, using or participating in the Internet. Protocols are created to control sending and receiving of messages such as TCP, IP, HTTP, Skype, 802.11 (WiFi). Internet standards are specification of a technology or methodology applicable to the Internet. They are created and published by IETF. The Internet is an infrastructure that provides services to applications such as Web, VoIP, email, games etc. It provides programming interface to apps, hooks that allow sending and receiving app programs to connect to Internet and provides service options, analogous to postal service.

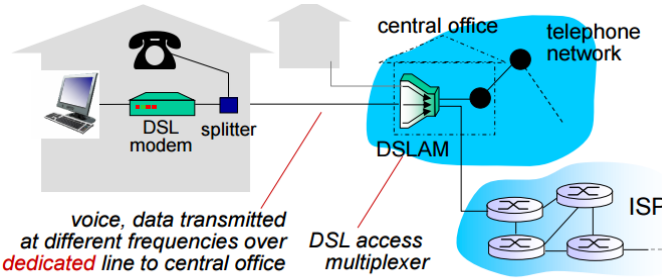
Protocols define *format, order* of *msgs sent and received* among network entities, and *actions taken* on msg transmission receipt. A normal human protocol is a conversation, while a computer network protocol is first TCP connection from pc to server, TCP response from server, retrieve msg and transmit msg to pc.

1.2 Network edge – end systems, access networks and links

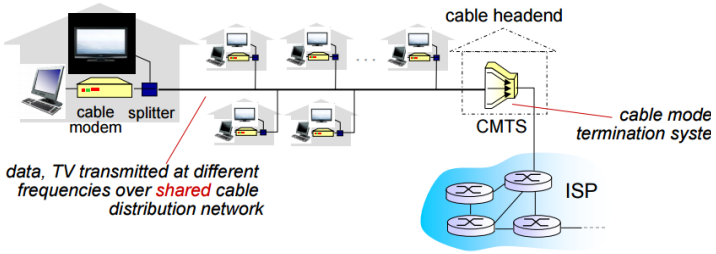
- Network edge:
  - Hosts: clients and servers
  - Servers often in data centers
- Access networks, physical media: is part of a telecommunications network which connects subscribers to their immediate service provider. Physical media are physical materials that are used to transmit information in data communications.
  - Wired, wireless communication links
- Network core: is the central part of a telecommunications network, that route's the network.
  - Interconnected routers
  - Network of networks

We connect systems to edge routers through residential access nets, institutional access networks such as company or schools, and through mobile access networks. Keep in mind the bandwidth of

the access network and if it's shared or dedicated. If it's shared you share the throughput available, but if it's dedicated you get what the server provides.

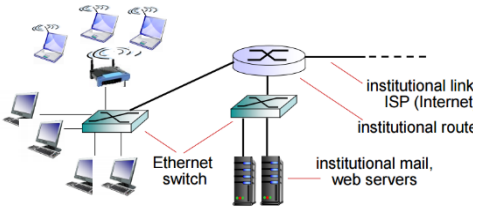


Access net: Digital subscriber line (DSL) use existing telephone line to central office DSLAM which is data over DSL phone line that goes to Internet. It has under 2.5 Mbps upstream transmission rate and under 24 Mbps downstream transmission rate. DSL is a technology for bringing high- bandwidth information to homes and businesses over ordinary copper telephone lines. *Frequency division multiplexing*is different channels transmitted in different frequency bands.



Hybrid Fiber Coax (HFC) is asymmetric with up to 30Mbps downstream transmission rate and a 2 Mbps upstream transmission rate.

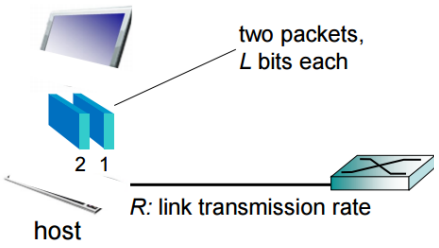
Ethernet is a family of computer networking technologies for local area networks (LANs) and metropolitan area networks (MANs). Typically used in companies and universities, see right picture.



Shared wireless access network connects end system to router via base station called access point. We have wireless LANs within buildings and wide-area wireless access provided by telco (cellular) operator that gives between 1 and 10Mbps in a area, either 3G, 4G or LTE.

Hosts sending packets of data: host sending function

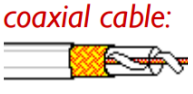
- Takes application message
- Breaks into smaller chunks, known as packets, of length *L* bits
- Transmits packet into access network at *transmission rate R*
    - Link transmission rate, aka link *capacity, aka link bandwidth*.



$$\text{packet transmission delay} = \text{time needed to transmit } L\text{-bit packet into link} = \frac{L \text{ (bits)}}{R \text{ (bits/sec)}}$$

Physical media:

- *Bit*: propagates between transmitter/receiver pairs
- *Physical link*: what lies between transmitter and receiver
- Signal carried in electromagnetic spectrum
- *Guided media*: signals propagate in solid media: copper, fiber and coax
- *Unguided media*: signals propagate freely, eg. Radio.



Twisted pair (TP) is two insulated copper wires, either category 5 with 100 Mbps, 1 Gbps Ethernet or category 6 with 10Gbps. *Coaxial cable* is two concentric copper conductors that are bidirectional and broadband with multiple channels on cable and HFC. *Fiber optic cable* is glass fiber carrying light pulses, each pulse a bit. Its high-speed operation, that's high-speed point-to-point transmission with example 10's-100's Gbps transmission rate. It has a low error rate where repeaters spaced far apart and immune to electromagnetic noise.

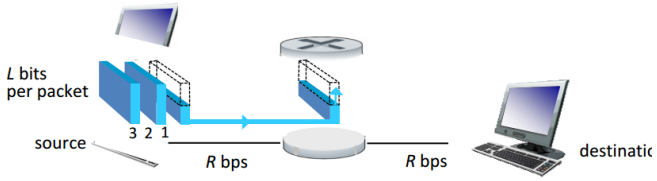
Unguided physical media: has no physical wire, its bidirectional, propagation environment effects are reflection, obstruction by objects and interference. Radio link types>

- Terrestrial microwave
  - Up to 45 Mbps channels
- LAN (e.g. WiFi)
  - 11Mbps, 54Mbps
- Wide-area (e.g. cellular)
  - 3G cellular: ~few Mbps
- Satellite
  - Kbps to 45Mbps channel(r multiple smaller channels)
  - 270 msec end-end delay
  - Geosynchronous versus low altitude

1.3 Network core – packet switching, circuit switching and network structure

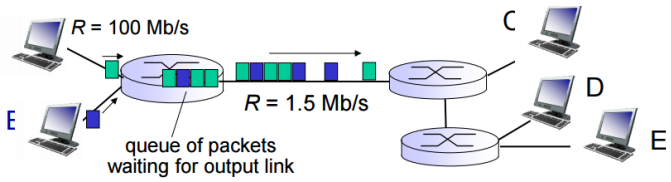
It's a mesh of interconnected routers. Packet-switching: hosts break application-layer messages into packets. That's forward packets from one router to the next, across links on path from source to destination. Each packet transmitted at full link capacity.

Store-and-forward: takes L/R seconds to transmit (push out) L-bit packet into link at R bps. Store and forward: entire packet must arrive at router before it can be transmitted on next link. End-end delay = 2L/R assuming zero propagation delay. Example; L = 7.5Mbps and R=1.5Mbps, then delay = 5 sec.

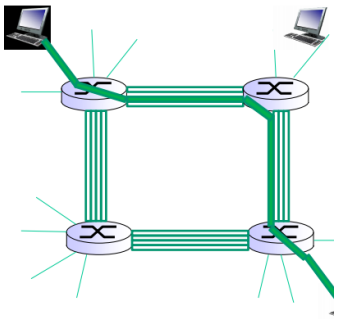


Queuing delay and loss: if arrival rate (in bits) to link exceeds transmission rate of link for a period of time:

- Packets will queue, wait to be transmitted on link
- Packets can be lost if memory (buffer) fills up



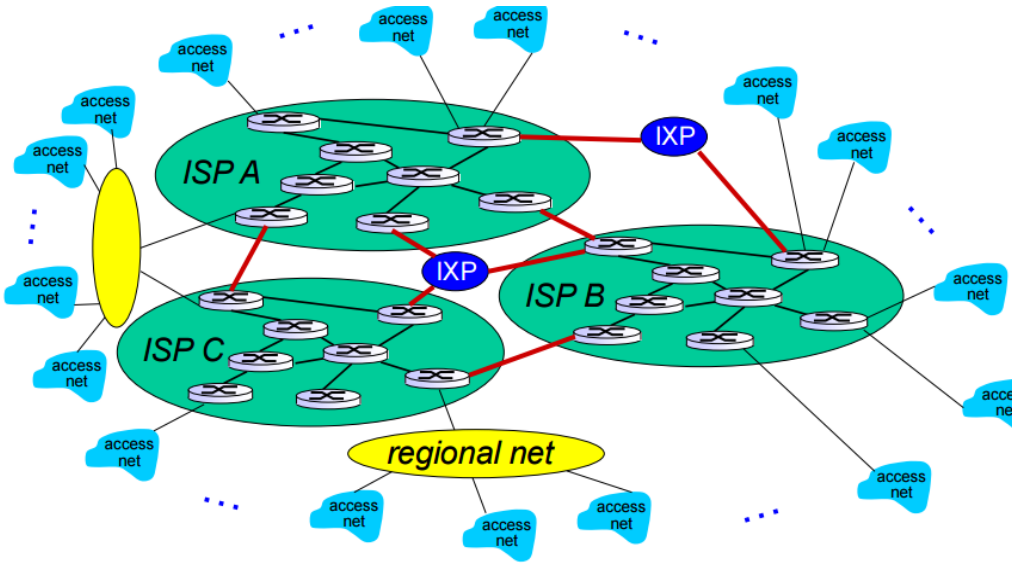
Alternative core: circuit switching: end-end resources allocated to, reserved for "call" between source and destination. In diagram, each link has four circuits. Call gets 3rd circuit in top link and 1st circuit in right link. Dedicated resources: no sharing where circuit-like (guaranteed) performance. Circuit segment idle if not used by call (no sharing). Commonly used in traditional networks.



Packet switching allows more users to use the network. Example: 1 Mbps with 100kb/s for each user that is active 10% of the time. Packet switching with 35 users has the probability that 10 active at the same time is less than 0.0004. Packet switching is great for bursty data, resource sharing, it's simpler than circuit and requires no call setup. It has excessive congestion possible: packet delay and loss, that's protocols needed for reliable data transfer, congestion control.

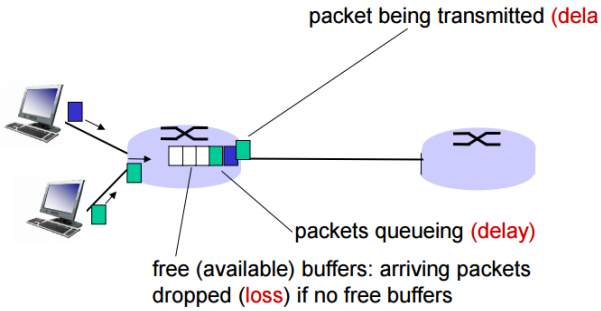
Internet structure: end systems connect to Internet via access ISPs. Access ISPs in turn must be interconnected so that any two hosts can send packets to each other. Resulting network of networks is very complex. Given millions of access ISPs, how do we connect them? Connect each access ISP to a global transit ISP. Costumer and provider ISPs have economic agreement. Still there can be several ISPs. Then the ISPs are connected through Internet exchange points (IXP). Regional network may

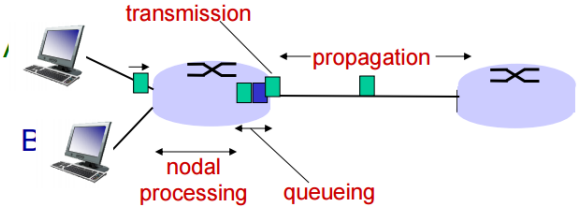
arise to connect access nets to ISPs. Content provider networks (Google, Microsoft) may run their own network, to bring services, content close to end users. This picture showed the end point with the content provider network over everything.



1.4 Delay, loss, throughput in networks

How do loss and delay occur? Packets *queue* in router buffers. Packet arrival rate to link (temporarily) exceeds output link capacity. Packets queue and wait for turn.





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

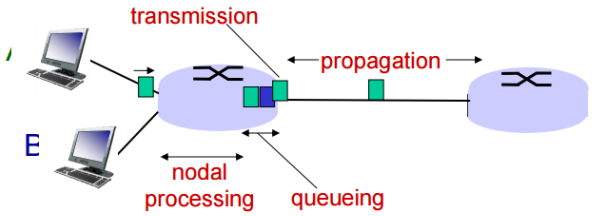
d_{proc}: nodal processing
- check bit errors
- determine output link
- typically < msec

d_{queue}: queueing delay
- time waiting at output link for transmission
- depends on congestion level of router

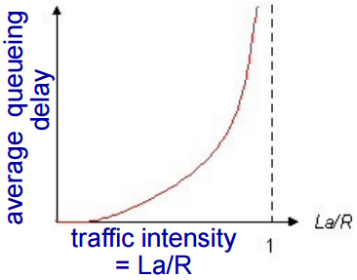
Queuing delay:

- **R:** Link bandwidth (bps)
- L: packet length (bits)
- A: average packet arrival time rate

La/R = 0, avg. queueing delay small
La/R → 1: avg. queuing delay large.
La/R > 1: avg. Queuing delay infinite!



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



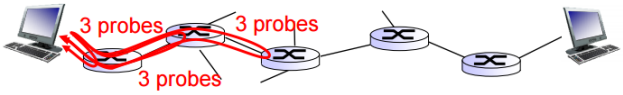
d_{trans}: transmission delay:
- *L*: packet length (bits)
- *R*: link *bandwidth (bps)*
- $d_{trans} = L/R$

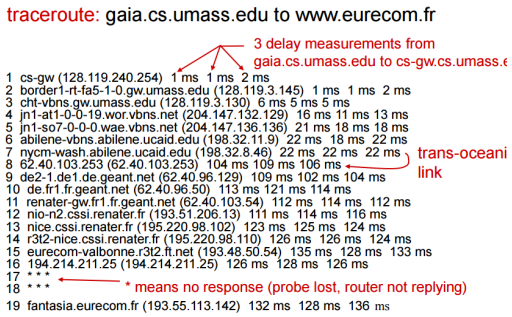
d_{prop}: propagation delay:
- *d*: length of physical link
- *s*: propagation speed in medium (~2×10^8 m/sec)
- $d_{prop} = d/s$

d_{trans} and d_{prop} *very* different

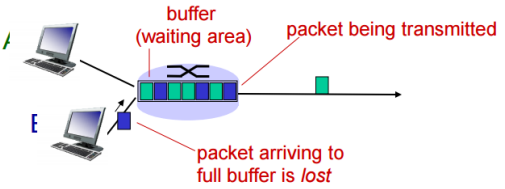
To find «real» Internet delay and loss we can use Traceroute (tracert) program that provides delay measurement from source to router along end-end Internet path towards destination. For all i:

- Sends three packets that will reach router i on path towards destination
- Router i will return packets to sender
- Sender times interval between transmission and reply.



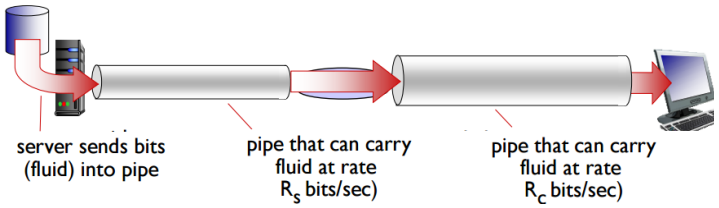


Packet Loss: queue (aka buffer) preceding link in buffer has finite capacity. Packet arriving to full queue dropped (aka lost). Lost packet may be retransmitted by previous node, by source end system, or not at all.



Throughput: rate (bits/time unit) at which bits transferred between sender/receiver. It's the rate of successful message delivery over a communication channel.

- *Instantaneous:* rate at given point in time
- *Average*: rate over longer period of time

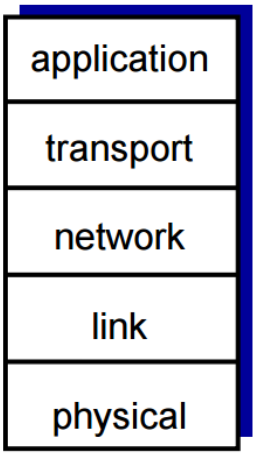


1.5 Protocol layers and service models

Networks are complex with many "pieces" such as hosts, routers, and links of various media, applications, protocols, hardware and software.

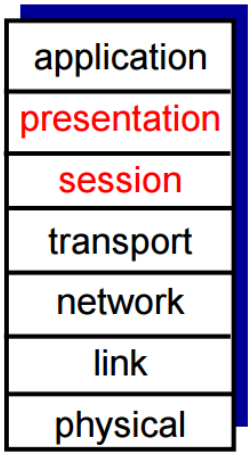
Internet Protocol Stack (IPS):

- Application: supporting network applications
  - FTP, SMTP, HTPP
- Transport: process-process data transfer
  - TCP, UDP
- Network: routing of datagrams from source to destination
  - IP, routing protocols
- Link: data transfer between neighboring network elements
  - Ethernet, 802.11 (WiFi), PPP
- Physical: bits "on the wire"

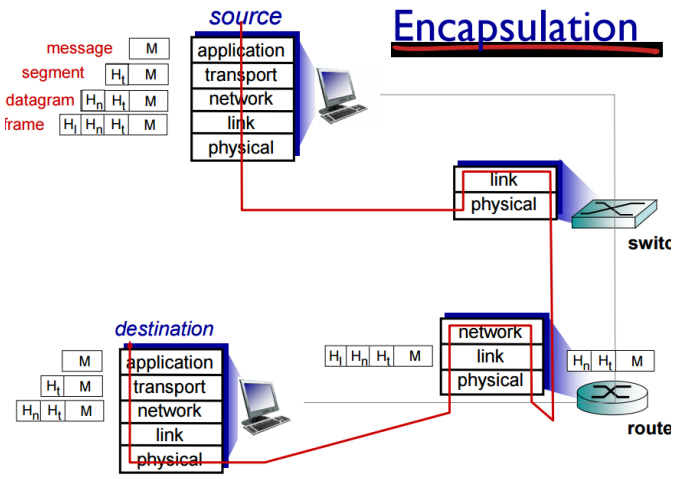


ISO/OSI reference model:

- Presentation: allows applications to interpret meaning of data, ex: encryption, compression, machine specific conventions.
- Session: synchronization, check pointing, recovery of data exchange.
- Internet stack "missing these layers"
  - These services, if needed, must be implemented in an application.



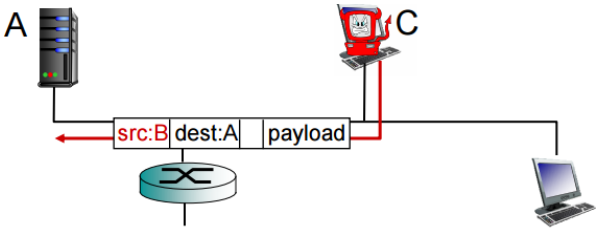
1.6 Networks under attack: security



Field on network security:

- How can bad guys attack computer networks
- How to defend networks against attacks
- How to design architectures that are immune to attacks

Internet not originally designed with security in mind. It was intended for a group of mutually trusting users attached to a transparent network. Malware can get in host from virus, which is a self-replicating infection by receiving/executing object or from worms, which are also self-replicating by passively receiving object that gets itself executed. Spyware malware can record keystrokes, web sites visited and upload the information to collection site. Infected host can be enrolled in botnet, used for spam and DDoS attacks. Denial of Service (DoS) is when attackers make resources such as servers and bandwidth unavailable to legitimate traffic by overwhelming resource with bogus traffic. It first selects the target, breaks into hosts around the network and sends packets to target from compromised hosts. Packet sniffing is promiscuous network interface reads or records all packets passing by. Programs such as Wireshark software are used for end-of-chapter labs in free packet sniffer. IP spoofing is to send packet with false source address.



Chapter 1 summary:

- Internet overview
- What's a protocol
- Network edge, core, access network
  - Packet-switching vs circuit switching
  - Internet structure
- Performance: loss, delay and throughput
- Layering, service models
- Security

Chapter 2 - Application layer

2.1 Principles of network applications

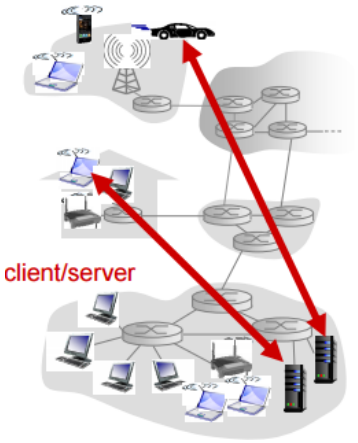
Goals: conceptual, implementation aspects of network application protocols such as the transport-layer service models, client-server paradigm and peer-to-peer paradigm. Learn about protocols by examining popular application-level protocols such as HTTP, FTP, SMTP/POP3/IMAP and DNS, and create network applications such as socket API.

Examples of network apps are e-mail, web, text messaging, P2P file sharing, and voice over IP, streaming stored video, social networking etc.

Creating network apps: write programs that run on (different) end systems, communicate over network and e.g. web server software communicates with browser software. There is no need to

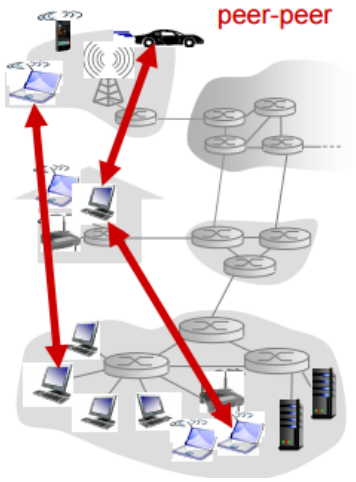
write application software for network-core devices, where network-core devices do not run user applications and applications on end systems allows for rapid app development and propagation.

Application architectures



Client-server:

- Server:
  - Always-on host
  - Permanent IP address
  - Data centers for scaling
- Clients:
  - Communicate with server
  - May be intermittently connected
  - May have dynamic IP addresses
  - Do no communicate directly with each other



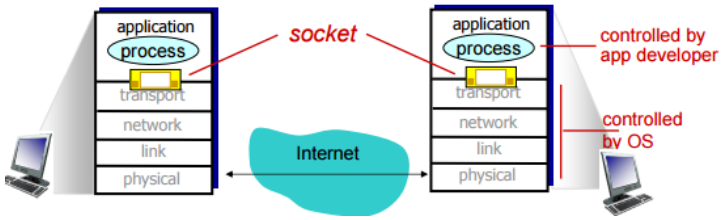
Peer-to-peer:

- *No* always-on server
- Arbitrary end systems directly communicate
- Peers request service from other peers, provide service in return to other peers
  - Self-scalability – new peers bring new service capacity, as well as new service demands
- Peers are intermittently connected and change IP addresses
  - Complex management

Process communicating: a process is a program running within a host. Within same host, two processes communicate using inter-process communication that is defined by OS. Processes in different hosts communicate by exchanging messages. Applications with P2P architectures have client processes and server processes. A client process is a process that initiates communication, while a server process is a process that waits to be contacted.

Sockets: process sends/receives messages to/from its *socket*. A network socket I s and endpoint of an inter-process communication across a computer network. Communication between computers are based on the Internet Protocol, therefore most network sockets are Internet sockets. Socket analogous to door

- Sending process shoves message out door
- Sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process.



Addressing processes: To receive messages, a process must have identifier. Host device has a unique 32-bit IP address. IP address of host does not suffice for identifying the process because many processes can be running on same host. *Identifier* includes both IP address and port numbers associated with process on host. Example of port numbers: HTTP server – 80 and mail server – 25.

Didrik Kvanvik

App-layer protocol defines:

- Types of messages exchanged
  - E.g. request, response
- Message syntax
  - What fields in messages and how fields are delineated
- Message semantics
  - Meaning of information in fields
- Rules – for when and how processes send and respond to messages
- Open protocols
  - Defined in RFCs, allows interoperability, e.g. HTTP, SMTP.
- Proprietary protocols
  - E.g. skype

What transport service an app needs:

application	data loss	throughput	time sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100' s msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few kbps up	yes, 100' s msec
text messaging	no loss	elastic	yes and no

- Data integrity
  - Some apps (e.g. file transfer, web transaction) require 100 % reliable data transfer
  - Apps that use audio can tolerate some loss
- Timing
  - Some apps (e.g. Internet telephony, interactive games) require low delay to be effective
- Throughput
  - Some apps (e.g. multimedia) require minimum amount of throughput to be effective
  - Elastic apps make use of whatever throughput they get
- Security
  - Encryption, data integrity

Internet transport protocols services:

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

TCP service:

- *Reliable transport* between sending and receiving process
- *Flow control*; sender won't overwhelm receiver
- Congestion control: throttle sender when network overloaded
- Does not provide: timing, minimum throughput guarantee and security
- Connection-oriented: setup required between client and server processes.

UDP service:

- Unreliable data transfer between sending and receiving process
- Does not provide: reliability, flow control, congestion control, timing, throughput guarantee, security or connection setup

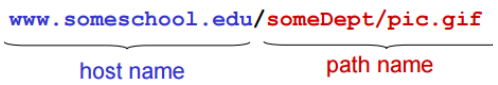
Securing TCP: TCP and UDP has no encryption and clear text passwords sent into socket traverse Internet in clear text. SSL provides encrypted TCP connection, it has data integrity and end-point authentication. SSL is an app layer, where apps use SSL libraries to talk to TCP. SSL socket API is used

to make sure that clear text passwords sent into socket traverse Internet is encrypted. Secure Sockets Layer (SSL) is a standard security technology for establishing an encrypted link between a server and a client. SSL allows sensitive information to be transmitted securely. SSL protocol determines variables of the encryption for both the link and the data being transmitted. Keys are used to set up the SSL connection (private and public), where anything encrypted with public key, can only be decrypted with private key and vice versa.

1. Browser connect to a web server secured with SSL (HTTPS), where the browser request that the server identify itself.
2. Server sends a copy of its SSL certificate, including the server's public key.
3. Browser checks the certificate root against a list trusted CAs and that the certificate is valid. If it's trusted, it is sends back a symmetric session key using the server's public key.
4. Server decrypts the symmetric session key using its private key and sends back an acknowledgement encrypted with the session key to start encrypted session.
5. Server and browser now encrypt all transmitted data with the session key.

2.2 Web and HTTP

A web page consist of objects, object can be HTML file, JPEG image, Java applet, audio file etc. Web page consists of base HTML- file which includes several referenced objects. Each object is addressable by a URL.



HTTP (Hypertext transfer protocol) is a web application layer protocol. A client/server model.

- Client: browser that requests, receives (using HTTP protocol) and displays web objects
- Server: Web server sends (using HTTP protocol) objects in response to requests

HTTP uses TCP: client initiates TCP connection (creates socket) to server using port 80. Server accepts TCP connection from client. HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server). TCP connection is then closed. HTTP is stateless, meaning that the server maintains no information about past client request. Protocols that maintain state are complex! We have two types of HTTP connections;

Non-persistent HTTP:

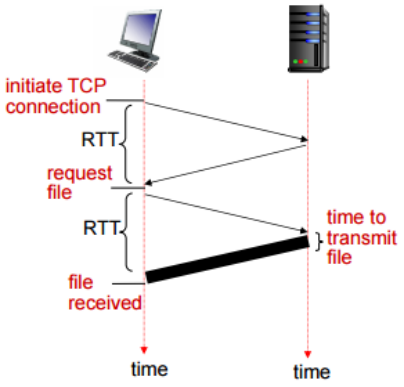
- At most one object sent over TCP connection, connection is then closed
- Downloading multiple objects required multiple connections
- Server leaves connection open after sending response
- Subsequent HTTP messages between same client/server sent over open connection
- Client sends requests as soon as it encounters a referenced object
- As little as one RTT for all the referenced objects.

Suppose user enters URL: `www.someSchool.edu/someDepartment/home.index`. First will HTTP client initiate TCP connection to HTTP server at www.someSchool.edu on port 80. Then HTTP server at host "a*ddress*" waiting for TCP connection at port 80, accepts connection, notifying client. HTTP client sends HTTP request message (containing URL) into TCP connection socket. Message indicates that client wants object "someDepartment/home.index". HTTP server receives request message, forms response message containing requested object, and sends message into its socket. HTTP server closes TCP connection. HTTP client receives response message containing HTML file, displays HTML. Then it's repeated for each jpeg object.

Didrik Kvanvik

RTT (definition): time for a small packet to travel from client to server and back. HTTP response time:

- One RTT to initiate TCP connection
- One RTT for HTTP request and fist few bytes of HTTP response to return
- File transmission time
- Non-persistent HTTP response time = 2RTT + file transmission time



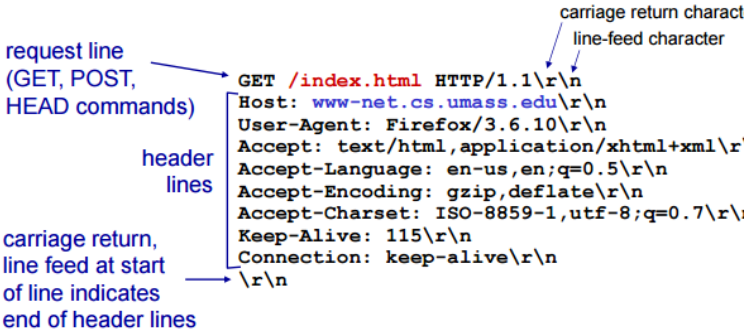
Persistent HTTP:

- Multiple objects can be sent over single TCP connection between client, server
- Issues:
  - Requires 2 RTTs per object
  - OS overhead for each TCP connection
  - Browsers often open parallel TCP connections to fetch referenced objects

There are two types of HTTP messages: response and request.

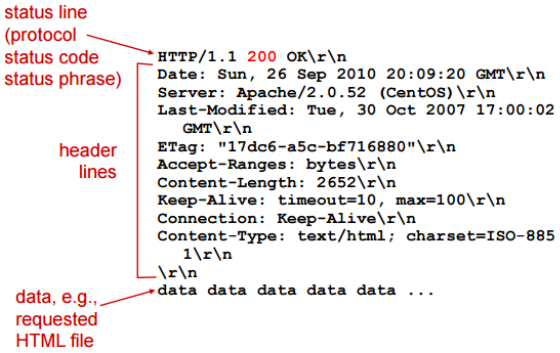
Request message is written in ASCII format.
Uploading form input:

- POST method:
  - Web page often includes form input
  - Input is uploaded to server in entity body
- URL method:
  - Uses GET method
  - Input is uploaded in URL field of request line



There are two method types:

- HTTP/1.0:
  - GET, POST and HEAD (HEAD: asks server to leave requested object out of response)
- HTTP/1.1:
  - GET, POST and HEAD
  - PUT – uploads file in entity body to path specified in URL field
  - DELETE – deletes file specified in the URL field

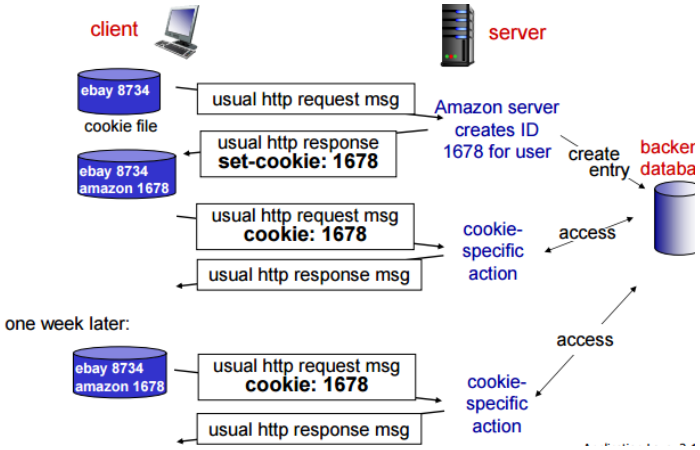


Response message: status cod appears in 1st line in server-to-client response message. Some examples:

- 200 – OK
  - Request succeeded, requested object later in this msg
- 301 – Moved permanently
  - Requested object moved, new location specified later in this msg
- 400 – Bad request
  - Request msg not understood by server

- 404 Not Found
    - Requested document not found on this server
- 505 – HTTP Version not supported

User-server state: cookies



Many Web sites use cookies, with four components:

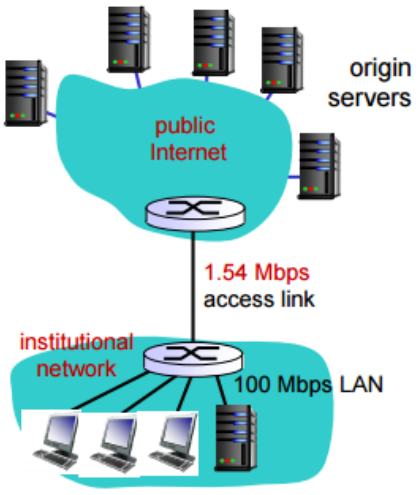
1. Cookie header line of HTTP *response* message
2. Cookie header line in next HTTP *request* message
3. Cookie file kept on user's host, managed by user's browser
4. Back-end database at web site.

Cookies can be used for authorization, shopping carts, recommendations, user session state (Web e-mail). Protocols endpoints maintain state at sender/receiver over multiple transactions. Cookies are http messages carry state. It's a small piece of data sent form a website and stored in the user's browser, and is used to remember stateful information or record the user's browsing activity or which pages were visited by the user.

Web cache: goal is to satisfy client request without involving origin server. User sets browser: web access via cache. Browser sends all HTTP requests to cache. Object in cache is returned, or the cache requests object from origin server and then returns object to client. Cache acts as both client and server and is typically installed by ISP. Web caching is used to reduce response time for client request and reduce traffic on an institution's access link. Internet dense with caches enables poor content providers to effectively deliver content. Cache example:

Assume:

- Avg object size: 100K bits
- Avg request rate from browsers to origin servers: 15/sek
- Avg data rate to browsers: 1.5 Mbps
- RTT: 2 sek
- Access link rate: 1.54 Mbps

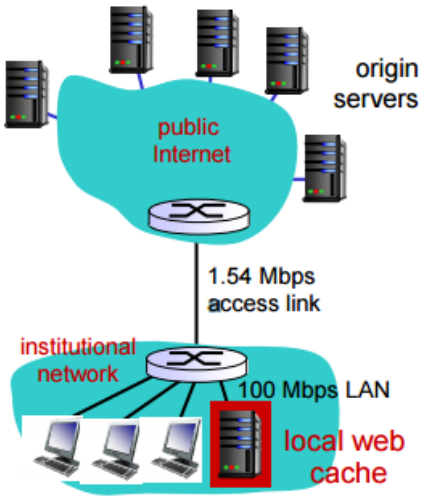


Consequences:

- LAN utilization: 1.5 %
- Access link utilization: 99 %
- Total delay = RTT + access delay + LAN delay
  = 2 sec + minutes + usecs

To increase access link speed we increase the access link rate to 15.4 Mbps, where the access link utilization is decreased to 9.9 % and the delay is msecs and not minutes.

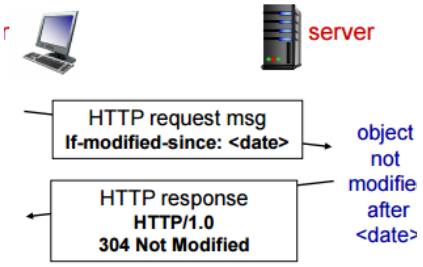
Install local cache: calculating access link utilization delay with cache:



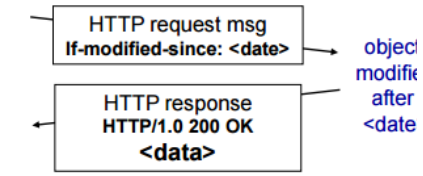
- suppose cache hit rate is 0.4
- access link utilization: 60 %
- data rate to browsers over access link
  = 0.6*1.5Mbps = 0.9 Mbps
  =utilization = 0.9/1.54 = 0.58
- Total average delay:
  0.6 * (delay from origin servers) + 0.4
  * (delay when satisfied at cache)
  = 0.6 (2.01) + 0.4 (~msecs)
  =~1.2 secs (less than with 15.4 Mbps link and cheaper)

Conditional GET:

The goal is to don't send object if cache has up-to-date cached version. That gives no object transmission delay and lower link utilization.



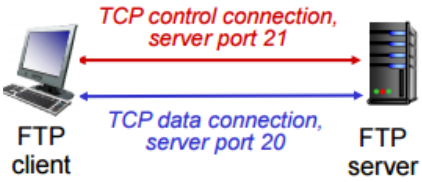
- Cache: specify date of cached copy in HTTP request.
- Server: response contains no object if cached copy is up-to-date



2.3 The File Transfer Protocol (FTP)

Transfer file to/from remote host. Client server model where the client side initiates transfer (either to/from remote) and server is remote host. Ftp: RFC 959 and ftp server: port 21.FTP is a standard network protocol used to transfer computer files form one host to another host over a TCP-based network. It's built on a client-server architecture and uses separate control and data connections between the client and the server.

Separate control and data connections:



- FTP client contacts FTP server at port 21, using TCP
- Client authorized over control connection
- Client browses remote directory, sends commands over control connection
- When server receivers file transfer command, server opens 2nd TCP data connection (for file) to client
- After transferring one file, sever closes data connection
- Server opens another TCP data connection to transfer another file
- Control connection: "out of band"
- FTP server maintains state: current directory, earlier authentication.

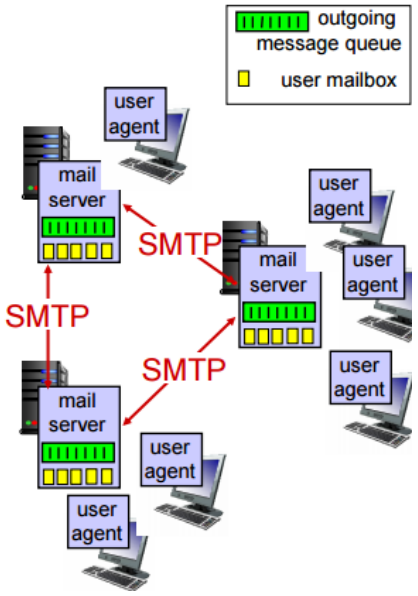
FTP commands and responses: USER *username*, PASS *password*, LIST return list of file in current directory, RETR *filename* retrieves file, STOR *filename* stores file onto remote host.

Most people prefer SSH over telnet and FTP for security reasons.

2.4 Electronic mail – SMTP, POP3 and IMAP

Three major components:

- User agents
  - "mail reader"
  - Composing, editing, reading mail messages
  - Outgoing, incoming messages stored on server
- Mail servers
  - Mailbox contains incoming messages for user
  - Message queue of outgoing mail messages
  - STMP protocol between mail servers to send email messages
    - Client: sending mail server
    - "Server": receiving mail server
- Simple mail transfer protocol (SMTP)
  - Uses TCP to reliably transfer email message from client to server, port 25 used
  - Direct transfer: sending server to receiving server
  - Three phases of transfer: greeting, transfer of message and closure
  - Command/response interaction (like HTTP and FTP)
    - Commands: ASCII text
    - Response: status code and phrase
  - Messages must be in 7-bit ASCI
  - Uses persistent connections
  - Uses CRLF.CRLF to determine end of message



comparison with HTTP:

- HTTP: pull
- SMTP: push
- both have ASCII command/response interaction, status codes
- HTTP: each object encapsulated in its own response msg
- SMTP: multiple objects sent in multipart msg

Mail message format: SMTP is protocol for exchanging email msgs while RFC 5322 is standard for text message format. Header lines include to, from and subject, while the body includes the message.

Mail access protocols: STMP is the delivery/storage to receiver's server. Mail access protocol: retrieval from server. Post office protocol (POP) with authorization and download. Internet Mail Access Protocol (IMAP) with more features, including manipulation of stored msgs on server. And the HTTP used with gmail, Hotmail, Yahoo etc.

POP3 protocol: is an application-layer Internet standard protocol used by local e-mail clients to retrieve e-mail from a remote server over a TCP/IP connection.

- Authorization phase:
  - Client command
    - User: declare username
    - Pass : password
  - Server response
    - +OK
    - -ERR
- Transaction phase, client:
  - List: list message numbers
  - Retr: retrieve message by number
  - Quite and Dele: delete

IMAP: keeps all messages in one place which is at server, allows user to organize messages in folders and keeps user state across sessions with names of folders and mappings between message IDs and folder name.

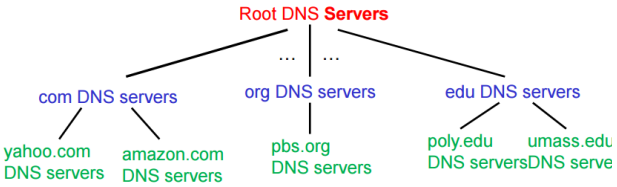
2.5 Domain Name System (DNS)

DNS is an Internet service that translates domain names into IP addresses (32/128 bit) used for addressing datagrams. DNS is distributed database implemented in hierarchy of many name servers. It's an application-layer protocol where name servers communicate to resolve names. DNS services:

- Hostname to IP address translation
- Host aliasing
- Mail server aliasing
- Load distribution
    - Replicated Web servers: many IP addresses correspond to one name
- Command line interface
    - Linux: host, dig
    - Windows: nslookup

The reason not to centralize DNS is because of single point of failure, traffic volume, distant centralized database and maintenance. Example:

Client wants IP for"www.amazon.com": client queries root server to find com DNS server. Client queries .com DNS server to get amazom.com DNS server. Client queries amazom.com DNS server to get IP address for www.amazon.com.



Root name servers: contacted by local name server that cannot resolve name. Root name server contacts authoritative name sever if name mapping not know, gets the mapping and returns the mapping to local name server.

Top-level domain (TLD): is one of the domains at the highest level in the hierarchical DNS of the Internet. TLD servers are responsible for:

- Com, org, net, edu, aero, jobs, museums, and all top-level country domains (no, uk, fra, ca)
- Versing maintains servers for .com TLD
- Educause for .edu TLD

Authoritative DNS servers:

- Organization's own DNS serves, providing authoritative hostname to IP mappings for organization's named hosts
- Can be maintained by organization or service provider

Local DNS name server: is the next layer that does not strictly belong to hierarchy. Each ISP has one, also called "default name server". When host makes DNS query, query is sent to its local DNS server, that has local cache of recent name-to-address translation pairs and acts as proxy, forwards query into hierarchy. DNS name resolution example; either Iterated query or recursive query.

Caching and update records: once (any) name server learns mapping, it caches mapping.

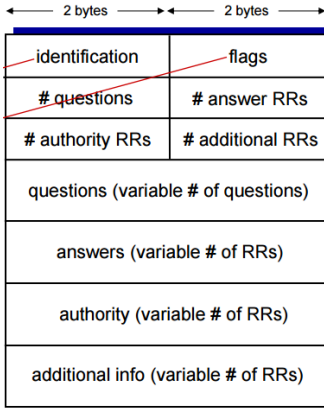
- Cache entries timeout (disappear) after some time (TTL)
- TLD servers typically cached in local name servers
    - Thus root name servers not often visited

- Cached entries may be out-of-date
  - If name host changes IP address, may not be known Internet-wide until all TLLs expire
- Update/notify mechanisms proposed IETF standard (TFC 2136)

Records: distributed db storing resource records (RR)

RR format: `(name, value, type, ttl)`

- Type A
  - Name is hostname and value is IP address
- Type NS
  - Name is domain and value is hostname of authoritative name server for this domain
- Type CNAME
  - Name is alias name for the real name and value is canonical (the real) name
- Type MX
  - Value is named of mail server associated with name

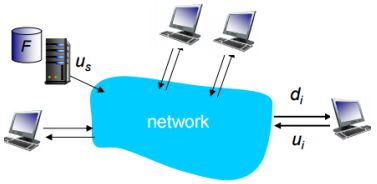


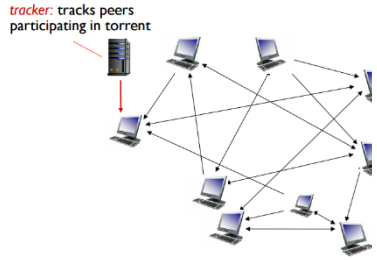
Messages: query and reply messages have the same message format. Message header with identification, that's a 16 bit # for query and reply uses the same. Flags are query or reply, recursion desired or available and reply is authoritative. Questions are name, type fields for a query. RR's in response to query are answers. Records for authoritative servers are authority and the final box is additional information.

Attacking DNS: DDoS: bombard root servers with traffic, it's not successful to date, traffic filtering, local DNS servers cache IPs of TLD servers that allows root server bypass. Bombard TLD servers which is potentially more dangerous. Redirect attacks are such as Man-in-middle which is to intercept queries. And DNS poisoning which is to send bogus replies to DNS server, which caches.

2.6 Peer-to-peer applications

- No always-on server
- Arbitrary end systems communicate directly
- Peers are intermittently connected and change IP addresses
- Examples: BitTorrent, Streaming and VoIP





File distribution time: Client server

- *Server transmission*: must sequentially send (upload) N files copies:
  - Time to send one copy: F/u_s
  - Time to send N copies: NF/u_s
- *Client*: each client must download file copy
  - D_{min} = min client download rate
  - Client download time is at least F/d_{min}

time to distribute F to N clients using client-server approach: $D_{c\text{-}s} \geq max\{NF/u_s, F/d_{min}\}$

increases linearly in N

File distribution time: P2P

- Server transmission: must upload at least one copy
  - Time to send one copy: F/u_s
- Client: each client must download file copy
  - Min client download time: F/d_{min}
- Clients: as aggregate must download NF bits
  - Max upload rate (limiting max download rate) is u_s + Sum of u_i

time to distribute F to N clients using P2P approach: $D_{P2P} \geq max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$

increases linearly in N …

… but so does this, as each peer brings service capacity

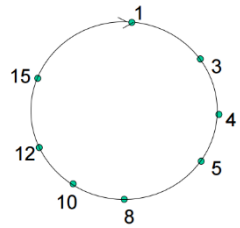
BitTorrent: is a method of downloading files using a distributed peer-to-peer file sharing system. It divides the file into 256Kb chunks where peers in torrent send/receive file chunks. Tracker tracks peers participating in torrent. Torrent is group of peers exchanging chunks of a file. Peer joining torrent has no chunks, but will accumulate them over time from other peers and registers with tracker to get list of peers, connects to subset of peers (neighbors). While downloading, peer uploads chunks to other peers. Peer may change peers with whom it exchanges chunks. Churn: peers may come and go. Once peer has entire file, it may leave or remain in torrent. Requesting chunks:

- At any given time, different peers have different subsets of file chunks
- Periodically, a peer asks each peer for list of chunks that they have, where the peer requests missing chunks from peers.

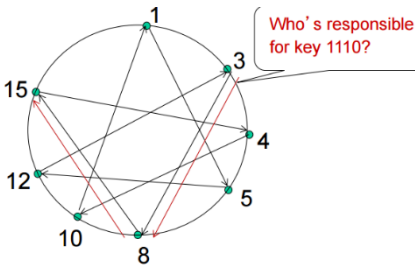
Distributed Hash Table (DHT):

Is a distributed P2P database. The database has (key, value) pairs. It distributes the pairs over millions of peers. A peer queries DHT with key and DHT returns values that match the key. Peers can also insert pairs. The central issue is assigning pairs to peers, where the basic idea is to convert each key to an integer, assign integer to each peer and put pair in the peer that is closest to the key.
Identifiers:

- Assign integer identifier to each peer in range [0,2n-1] for some n
  - Each identifier represented by n bits
- Require each key to be an integer in same range
- To get integer key, hash original key



Assign keys to peers: the rule is to assign key to the peer that has the closest ID. Convention in lecture is that the closest is the immediate successor of the key. Example where n = 4 and peers: 1, 3,4,5,8,10,12,14. Key is 13, then the successor peer is 14, if key is 15 then the successor peer is 1. This creates a circular DHT where each peer is only aware of immediate successor and predecessor which creates an overlay network. If peer 3 asks who is responsible for a key, and the peer responsible is peer 15, it has to ask through 6 peers to get an answer. By using shortcuts each peer keeps then track of successor, predecessor and short cuts, which reduces the message from 6 to 2. The design shortcuts goes from O(N) to O(log N) in message query.



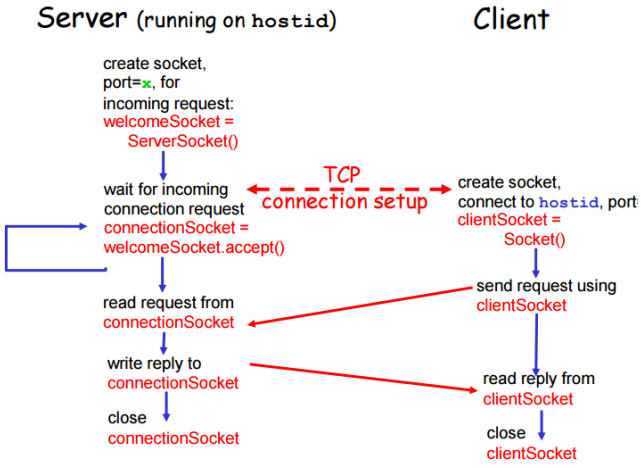
Peer churn: is when peers may come and go (churn), where each peer knows address of its two successors and each peer periodically pings its two successors to check aliveness. If immediate successor leaves, choose next successor as new immediate successor.

2.7 Socket programming with UDP and TCP

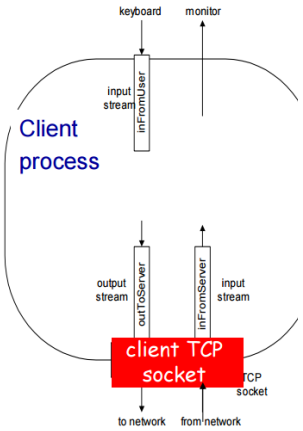
A socket is a host-local, application-created, OS-controlled interface into which application process can both send and receive messages to/from another application process. There are two types of

transport service via socket API: unreliable datagram and reliable byte stream-oriented. So a socket is a door between application process and end-end-transport protocol (UCP or TCP). TCP service is a reliable transfer of bytes from one process to another.

TCP



- Client must contact server
  - Server process must first be running
    - Server must have created socket that welcomes client's contact
- Client contacts server by:
  - Creating client-local TCP socket
  - When client creates socket: client TCP establishes connection to server TCP
  - When contacted by client, server TCP creates new socket for server process to communicate with client
    - Allows server to talk with multiple clients
    - Source port numbers used to distinguish clients



Stream is a sequence of characters that flow into or out of a process. Input stream is attached to some input source for the process (e.g. keyboard or socket). Output stream is attached to an output source (e.g. monitor or socket).

UDP (No connection between client and server)

Sender explicitly attaches UP address and port of destination to each packet, server must extract IP address, port of sender from received packet. Transmitted data may be received out or order, or lost.

Chapter 2 summary

- Application architectures
  - Client-server
  - P2P
- Application service requirements
  - Reliability, bandwidth and delay
- Internet transport service model
  - Connection-oriented, reliable: TCP
- Specific protocols
  - HTTP, FTP, SMTP, POP, IMAP, DNS, P2P: BitTorrent, DHT
- Socket programming with UDP and TCP sockets

Didrik Kvanvik

Chapter 3 – Transport Layer

Goals: understand principles behind transport layer services such as multiplexing, demultiplexing, reliable data transfer, flow control and congestion control. We are to learn about Internet transport layer protocols such as UDP (connectionless transport), TCP (connection-oriented reliable transport).

3.1 Transport-layer services

Transport services and protocols provide logical communication between app processes running on different hosts. Also provides services such as connection-oriented data steam support, reliability, flow control and multiplexing. They transport protocols run in end systems; send side to break app messages into segments and passes to network layer, rvc side reassembles segments into messages and passes to app layer. Its more than one transport protocol available to apps: TCP and UDP. Add control of communication session to make reliable transport session to make reliable transport service. TCP is reliable and in-order delivery that provides congestion control, flow control and connection setup. UDP on the other hand is unreliable and unordered delivery. Services that is not available is delay guarantees, bandwidth guarantees and security guarantees.

3.2 Multiplexing and demultiplexing

Transport vs network layer

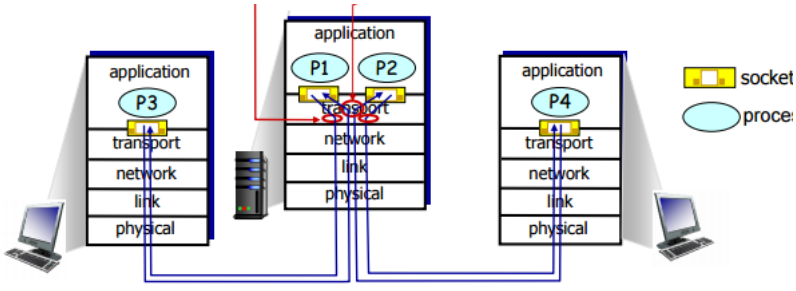
Transport layer:

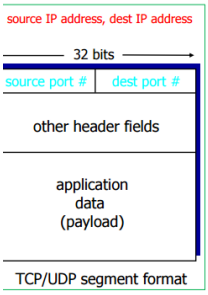
- Logical communication between processes that relies on network layer services.
- Responsible for checking that data available in session layer are error free
- Protocols used at this layer:
    - Transmission Control Protocol (TCP)
    - User Datagram Protocol (UDP)
    - SCTP (Stream Control Transmission Protocol)
- Ensures that the protocols operated at this layer provide reliable end-to-end flow and error control
- Is where the decision to use TCP/UDP is made.

Network layer:

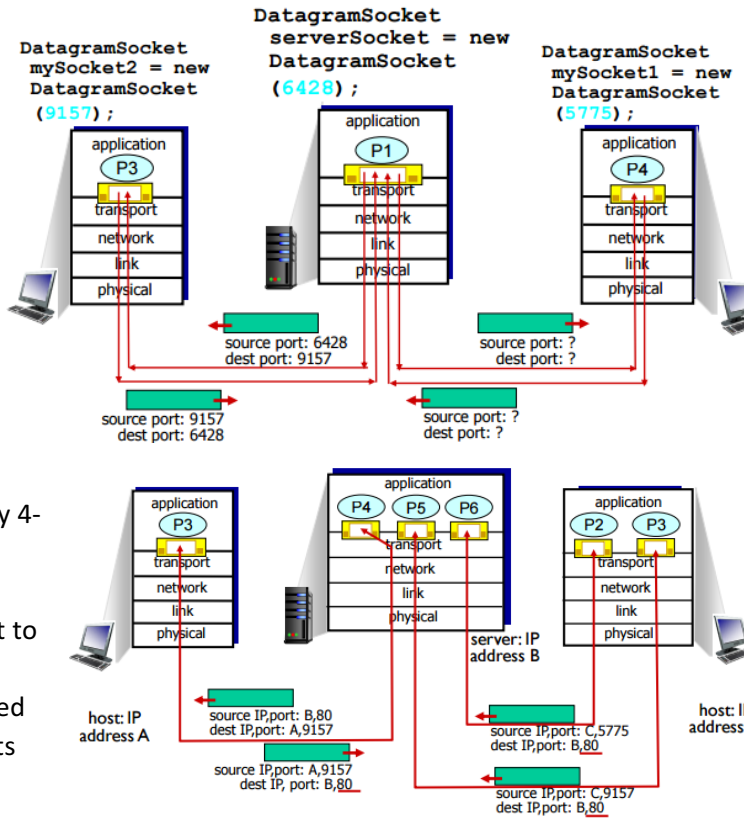
- Logical communication between hosts
- Responsible for logical addressing and translating logical addresses into physical addresses
- Protocols used at this layer:
    - Internet Protocol (IP)
    - ICMP, IGMP
- This layer controls routing of data from source to destination plus the building and dismantling data packets.

Multiplexing is sending multiple signals or streams of information on a carrier at the same time in the form of a single, complex signal and the recovering the separate signals at the receiving end. Multiplexing at sender to handle data from multiple sockets, add transport header to be later used for demultiplexing. Demultiplexing at receiver to use header into to deliver received segments to correct socket.





Demultiplexing: the host receives IP datagrams where each datagram has source IP address and destination IP address. Each of these datagrams carries one transport-layer segment and has source and destination port number. The host uses these port numbers and sometimes IP addresses to direct segment to appropriate socket.

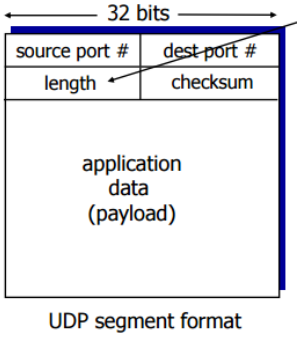


Connection*less* demultiplexing: recall that socket has host-local port # and when creating datagram to send into UDP socket you must specify destination IP address and port #. When host receives UDP segment it checks destination port # in segment and directs UDP segment to socket with that port #. IP datagrams with same destination port #, but different source IP addresses and/or source port numbers will be directed to same socket at destination.

Connection-oriented demux: TCP socket identified by 4-tuple: Source IP address, source port number, the destination IP address and destination port number. Demux receiver uses all four values to direct segment to appropriate socket. Server host may support many simultaneous TCP sockets where each socket identified by its own 4-tuple. Web servers have different sockets for each connecting client.

3.3 Connectionless transport – UDP

- "bare bones" Internet transport protocol
- "best effort" service, where segments may be:
  - Lost or delivered out-of-order to app
- *Connectionless:*
  - No handshaking between UDP and receiver
  - Each UDP segment handled independently of others
- UDP use:
  - Streaming multimedia apps (loss tolerant and rate sensitive
  - DNS
  - SNMP
- Reliable transfer over UDP:
  - Add reliability at application layer
  - Application-specific error recovery



We have a UDP because of the no connection establishment (which can add delay), no connection state at sender and receiver, small header size and no congestion control (UDP can blast away as fast as desired). UDP checksum goal is to detect "errors" (flipped bits) in transmitted segment:

example: add two 16-bit integers

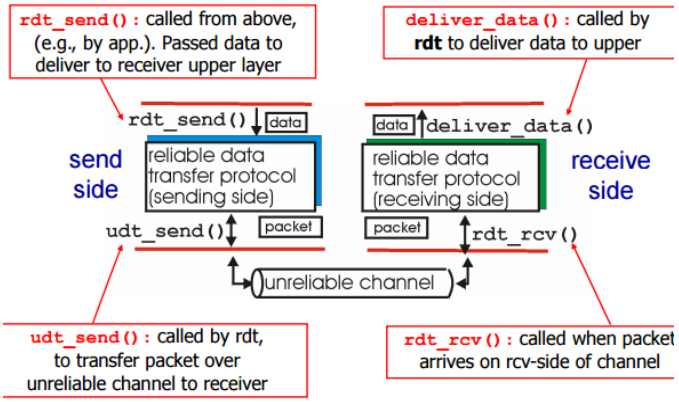
	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
wraparound	(1)	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
sum	1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

Note: when adding numbers, a carryout from the most significant bit needs to be added to the result

- Sender:
  - Treat segment contents, including header fields, as sequence of 16-bit integers
  - Checksum: addition of segment contents
  - Sender puts checksum value into UDP checksum field
- Receiver:
  - Compute checksum of received segment
  - Check if computed checksum equals checksum field value
    - NO: error detected
    - YES – no error detected

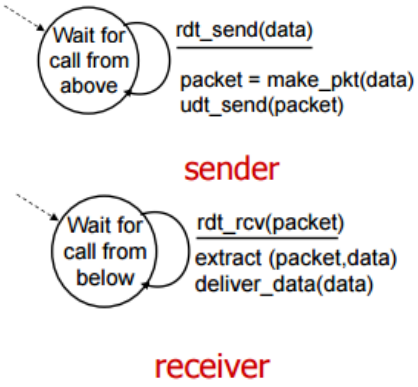
3.4 Principles of reliable data transfer

Important in application, transport and link layers. Characteristics of unreliable channel will determine complexity of reliable data transfer protocol (RDF).



How to:

- Incrementally develop sender, receiver sides of reliable data transfer protocol
- Consider only unidirectional data transfer
  - But control info will flow on both directions
- Use finite state machines (FSM) to specify sender and receiver



Rdt1.0 - Reliable transfer over a reliable channel:

- Underlying channel perfectly reliable
  - No bit errors
  - No loss of packets
- Separate FSMs for sender, 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
- The question: how to recover from errors:
  - Acknowledgements (ACKs): 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
- New mechanisms in rdt2.0
  - Error detection
  - Feedback: control msgs (ACK, NAK) from receiver to sender

If the ACK or NAK is corrupted the sender doesn't know what happened at receiver and if it retransmit it can transmit a duplicate and if now, possible loss. To handle duplicates, sender retransmits current pkt if ACK/NAK is corrupted. Sender adds sequence number to each pkt and receiver discards duplicate pkt. Stop and wait process: sender sends one packet, and then waits for receiver response.

Didrik Kvanvik

RDT2.1

Sender:

- Seq # added to pkt
- Two seg. #'s (0,1) will suffice
- Must check if received ACK(NAK is corrupted
- Twice as many states
    - State must remember whether expected pkt should have sequence # of 0 or 1

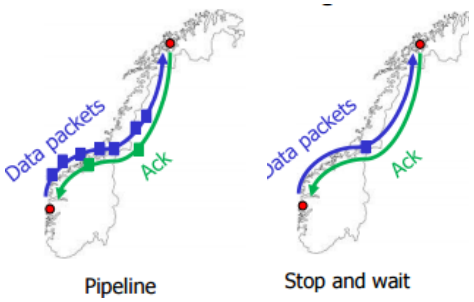
Receiver:

- Must check if received packet is duplicate
    - State indicates whether 0 or 1 is expected pkt sequence #.
- Receiver ca not know if its last ACK/NAK received OK at sender

RDT2.2 - NAK-free protocol: same functionality as rdt2.1, using ACKs only. Instead of NAK, receiver sends ACK for last pkt received OK where receiver must explicitly include sequence # of pkt being ACKed. Duplicate ACK at sender results in same action as NAK: retransmit current pkt.

RDT3.0 – channels with errors and loss: Underlying channel can also lose packets (data, ACKs). Checksum, seq. #, ACKs, retransmissions won't be enough. ACK loss result in stop and wait forever. The approach is that sender waits a reasonable amount of time for ACK. Retransmits if no ACK received in this time. If pkt or ACK just delayed the retransmission will be duplicate but seq. #'s already handles this. Receiver must specify seq. # of pkt being ACKed. Requires a countdown timer.

Pipelined protocols: pipelining means that sender allows multiple yet-to-be-acknowledged pkts where the range of sequence numbers must be increased and buffering at sender and/or receiver. Two generic forms of pipelined protocols: go-Back-N and selective repeat.



- Selective Repeat:
    - Sender can have up to N unacked packets in pipeline
    - Receiver sends individual ack for each packet.
    - Sender maintains timer for each unacked packet
        - When timer expires, retransmits only that unacked packet
- Go-Back-N:
    - Sender can have up to N unacked packets in pipeline
    - Receiver only sends cumulative ack
        - Doesn't ack packet if there's a gap
    - Sender has timer for oldest unacked packet
        - When timer expires, retransmits all unacked packets.

Go-back-N has k-bit seq. # in pkt header, with "window" of up to N, consecutive unacked pkts allowed. ACK(n): ACKs all pkts up to, including seq. # n. timer for oldest in-flight pkt. Timeout(n): retransmit packet n and all higher seq. # pkts in window. ACK-only: always send ACK for correctly-received pkt with highest in-order sequence #. It may generate duplicate ACKs and need only remember "expectedseqnum". Out-of-order pkt: discard (don't buffer) and re-ACK pkt with highest in-order sequence #.

Selective Repeat: receiver individually acknowledges all correctly received pkts. Buffers pkts, as needed, for eventual in-order delivery to upper layer. Sender only resends pkts for which ACK not received and sender time for each unacked pkt. Sender window with N consecutive sequence #'s and limits seq. #'s of send, unacked pkts.

- Sender:
  - Data from above:
    - If next available seq. # in window, send pkt
  - Timeout(n)
    - Resend pkt n, restart timer
  - ACK(n) in [sendbase, sendbase+N]
    - Mark pkt n as received
    - If n smallest unacked pkt, advance window base to next unacked seq. #
- Receiver:
  - Pkt n in [rcvbase, rcvbase+N-1]
    - Send ACK(n)
    - Out-of-order: buffer
    - In-order: deliver (also deliver buffered, in-order pkts), advance window to next not-yet-received pkt
  - Pkt n in [rcvbase-N, rcvbase-1]
    - ACK(n)
  - Otherwise ignore

Chapter 4 – Network Layer

Goals: understand principles behind network layer services: network layer service models, forwarding versus routing, how a router works, routing (path selection), broadcast, multicast and instantiation and implementation in the Internet.

4.1 Introduction

The network layer is layer 3 in the OSI model of computer networking. It's responsible for packet forwarding including routing through intermediate routers, whereas the data link layer is responsible for media access control, flow control and error checking.

- Transport segment from sending to receiving host
- On sending side encapsulates segments into datagrams
- On receiving side, delivers segments to transport layer
- Network layer protocols in *every* host and router
- Router examines header fields in all IP datagrams passing through it

Two key network-layer functions:

- Forwarding: process of getting through single interchange
  - move packets from router's input to appropriate router output
- Routing: process of planning trip from source to destination
  - determine route taken by packets from source to destination
    - routing algorithms

The routing algorithm determines end-to-end-path through network, while forwarding table determines local forwarding at this router.

4.2 Virtual circuit and datagram networks

A network service is an application running at the network application layer and above, that provides data storage, manipulation, presentation and communication or other capability which is often implemented using a client-server or peer-to-peer architecture based on application layer network protocols. Example services for individual datagrams: guaranteed delivery, guaranteed delivery with less than 40 msec delay or best-effort with no guarantees. Example services for a flow of datagrams: in-order datagram delivery, guaranteed minimum bandwidth to flow, limits on variation in inter-packet spacing or best-effort with no guarantees.

Network layer service models:

Network Architecture	Service Model	Guarantees ?				Congestic feedback
		Bandwidth	Loss	Order	Timing	

Connection, connection*less* service:

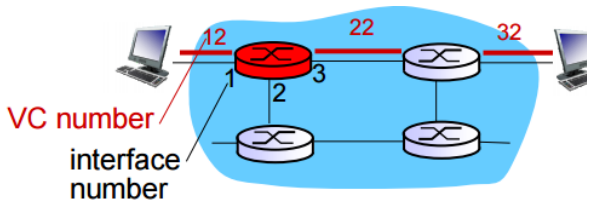
- datagram network provides network-layer connectionless service
- virtual-circuit network provides network-layer connection service
- analogous to TCP/UDP connection-oriented/connectionless transport-layer services, but in the network layer:
    - service: host-to-host
    - no choice: network provides one or the other
    - implementation: in network core

Connection setup: Important function in some network architectures is to need connection in order to provide service guarantees (ATM, frame relay, X.25). Before datagrams flow, two end hosts and intervening routers establish virtual network connections where routers get involved. Network layer connection service is between two hosts and may also involve intervening routers in case of VSs while the transport layer connection service is between two processes.

Virtual circuits (VCs): "source-to-destination path behaves much like telephone circuit", when seen performance-wise and network actions along source-to-destination path. Call setup for each call before data can flow, teardown after. Each packet carries VC identifier (not destination host address). Every router on source-destination path maintains "state" for each passing connection". Link, router resources (bandwidth, buffers) may be allocated to VC (dedicated resources = predictable service). Implementation: a VC consists of

1. *path* from source to destination
2. *VC numbers*, one number for each link along path
3. *Entries in forwarding tables* in routers along path

- Packet belonging to VC carries VC number (rather than dest address)
- VC number can be changed on each link
    - New VC number comes from forwarding table



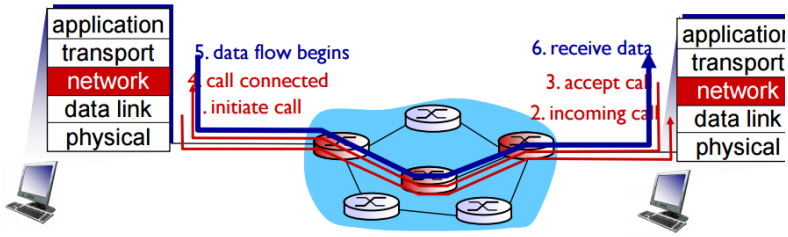
VC forwarding table:

Forwarding table in northwest router: →

VC routers maintain connection state information!

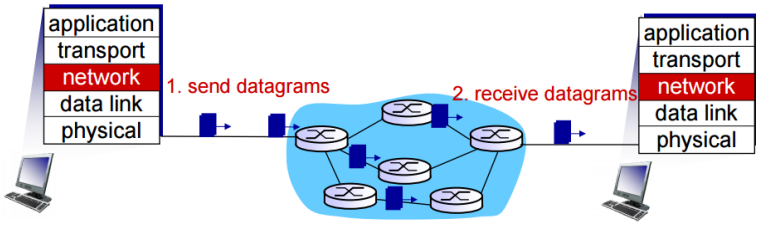
Incoming interface	Incoming VC #	Outgoing interface	Outgoing VC
1	12	3	22
2	63	1	18
3	12	2	22
1	87	3	87
…	…	…	…

VC signal protocols: are used to setup, maintain teardown VC, used in ATM, frame-relay, X.25, but not in today's Internet.



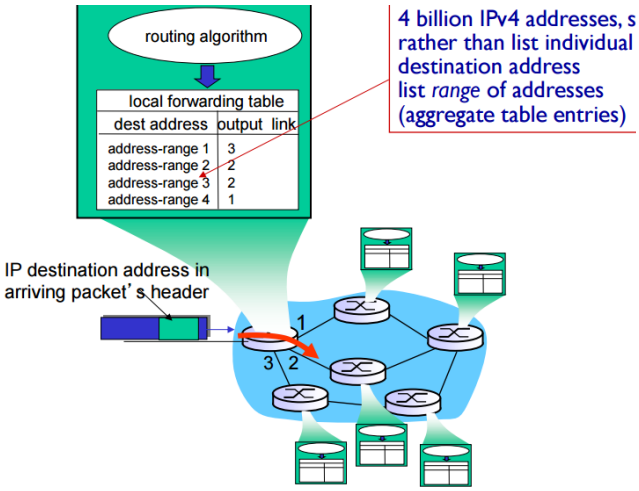
Datagram networks:

- No call setup at network layer
- Routers: no state about end-to-end connections
  - No network-level concept of "connection"
- Packets forwarded using destination host address



Longest prefix matching is when we are looking for forwarding table entry for given destination address, we 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



examples:

DA: 11001000 00010111 00010110 10100001 which interface?

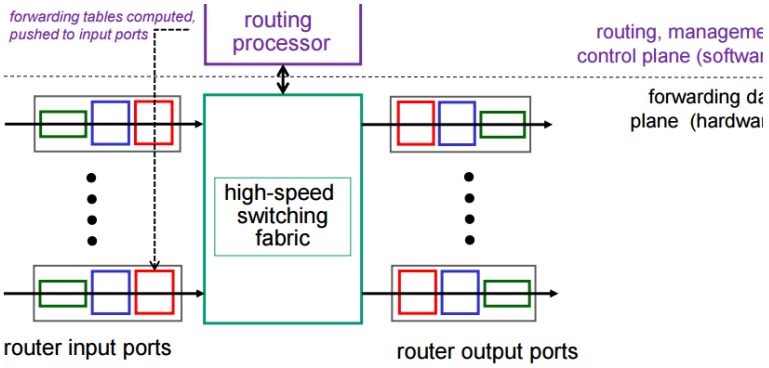
DA: 11001000 00010111 00011000 10101010 which interface?

- Internet (datagram)
  - Datagram exchange among computers
    - "elastic service, no strict timing required
  - Many link types
    - Different characteristics
    - Uniform service difficult
  - "smart" end systems
    - Can adapt, perform control, error recovery
    - Simplicity inside network, complexity at "edge"
- ATM (VC)
  - Evolved from telephony
  - Human conversation:
    - Strict timing, reliability requirements
    - Need for guaranteed service
  - "dumb" end systems
    - Telephones
    - Complexity inside network

4.3 What's inside a router

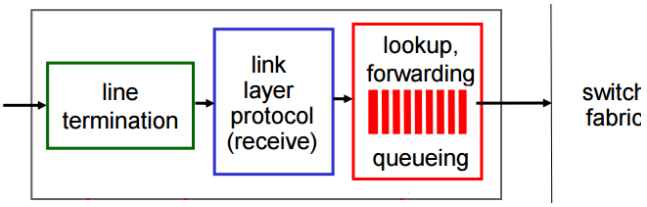
There are two key (Internet) router functions:

- Run routing algorithms/protocol (RIP, OSPF, BGP)
- Forwarding datagrams from incoming to outgoing link



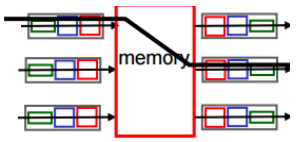
Input port functions:

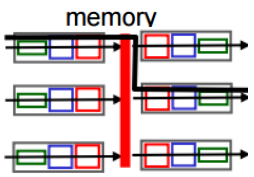
Line termination is the physical layer with bit-level representation. The link layer protocol is the data link layer. Decentralized switching happens in lookup, forwarding queuing:



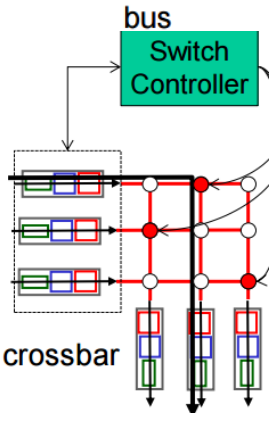
- Given datagram destination, lookup output port using forwarding table in input port memory
- Goal: complete input port processing at "line speed"
- Queuing: if datagrams arrive faster than forwarding rate into switch fabric

Switching fabrics: transfer packet from input buffer to appropriate output buffer. Switching rate is the rate at which packets can be transferred from inputs to outputs, often measured as multiple of input/output line rate, N inputs give switching rate N times line rate desirable. There are three types of switching fabrics:



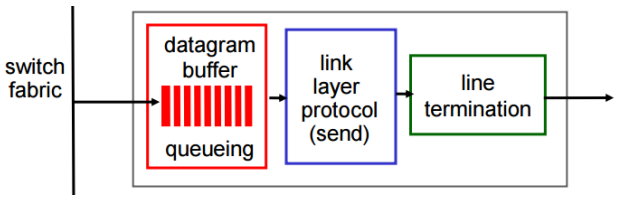


- Switching via memory:
  - Traditional computers with switching under direct control of CPU
  - Packet copied to system's memory
  - Speed limited by memory bandwidth (2 bus crossings per datagram).
- Switching via a bus:
  - Datagram from input port memory to output memory via a shared bus
  - Bus contention: switching speed limited by bus bandwidth
  - 32 Gbps, Cisco 5600: sufficient speed for access and enterprise routers
- Switching via interconnection network:
  - Overcome bus bandwidth limitations
  - Banyan networks, crossbar, other interconnection nets initially developed to connect processors in multiprocessor systems
  - Advances design: fragmenting datagram into fixed length cells, switch cells through the fabric.
  - Cisco 12000: switches 60 Gbps through the interconnection network.



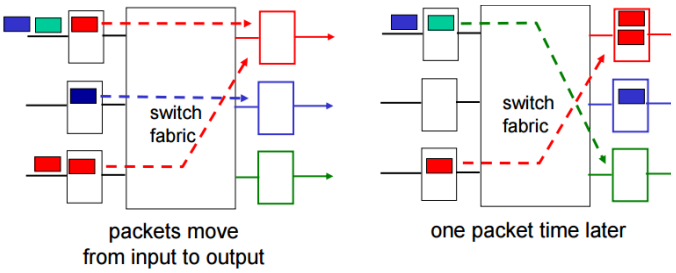
Output ports:

- *Buffering* required when datagrams arrive from fabric much faster than the transmission rate
- *Scheduling discipline* chooses among queued datagrams for transmission



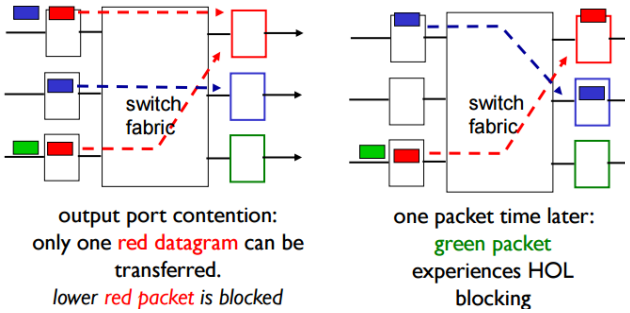
Output port queuing:

- *Buffering* when arrival rate via switch exceeds output line speed
- *Queuing (delay) and loss due to **output** port buffer overflow!*



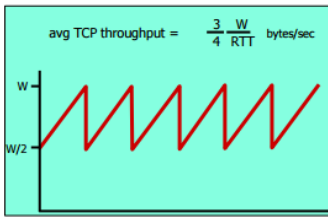
Input port queuing:

- If fabric slower than input ports combined, queuing may occur at input queues
  - *Queuing delay and loss due to **input** buffer overflow.*
- *Head-of-the-Line (HOL) blocking:* queued datagram at front of queue prevents others in queue from moving forward



How much buffering?

- RFC 3439 rule of thumb:
  - Buffering: "typical" RTT times link capacity C
  - E.g. = C = 10 Gbps link, "typical" 250 msec RTT: 2.5 Gbit buffer of RAM per output link
  - Based on experience with "few flows" through router:
  - Max TCP windows :
    W = RTT * max TCP throughput

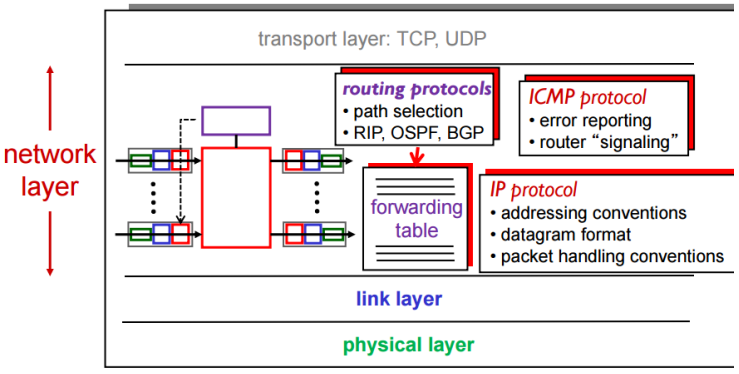


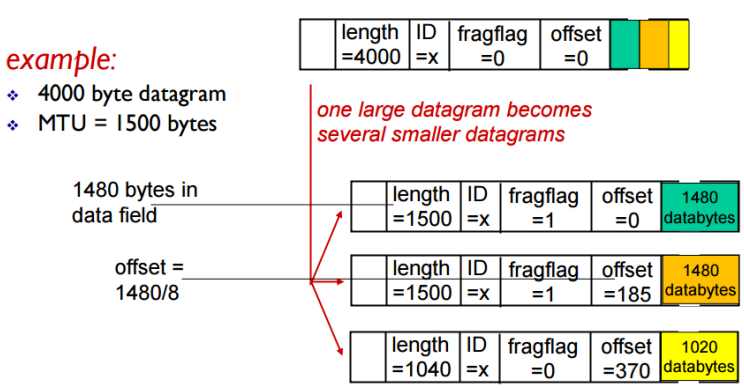
- Recent recommendation: with N (large) flows through link sufficient buffer space is per output link $\frac{RTT \cdot C}{\sqrt{N}}$

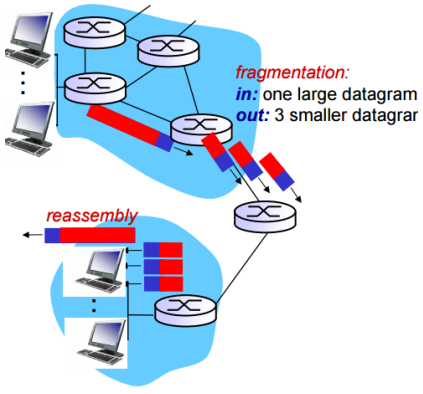
4.4 IP: Internet Protocol

IP fragmentation and reassembly

Network links have Max transfer unit (MTU) equal to the largest possible link-level frame. There exist different types of link types and MTUs. Large IP datagram is divided (fragmented) within net where one datagram becomes several datagrams. Reassembled only at final destination. The IP header bits are used to identify and order related fragments.

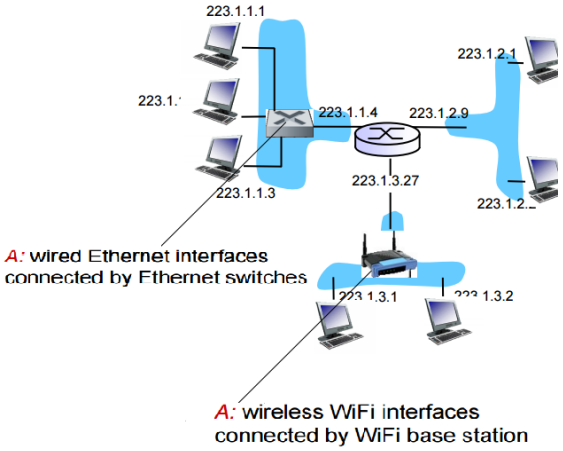






Didrik Kvanvik

IPv4 addressing:



- IP address: 32-bit identifier for host, router interface
- Interface: connection between host/router and physical link

  Host typically has one or two interfaces (e.g. wired Ethernet and wireless 802.11)
- IP addresses associated with each interface

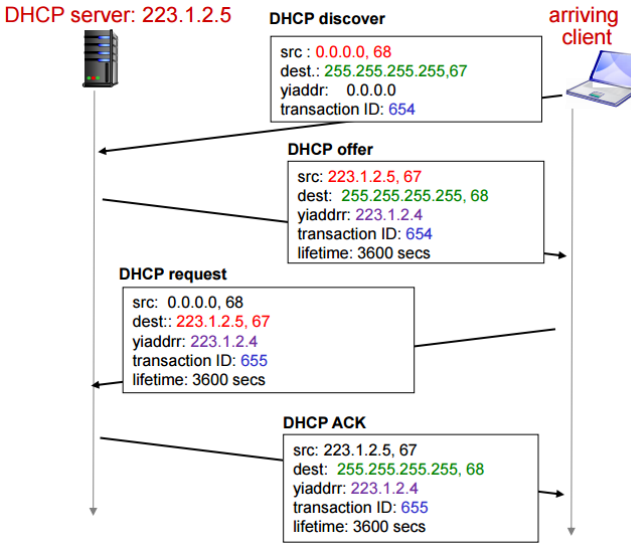
A **Subnet** is a device interface with the same subnet part of IP address that can physically reach each other without intervening router. We say within IP address the subnet part is the high order bits while the host part is the low order bits. To determine the subnets, detach each interface from its host or router, creating islands of isolated networks. Each isolated network is in fact called a subnet. A subnet is a logical, visible subdivision of an IP network. The routing prefix is expressed in **CIDR (Classless InterDomain Routing) notation:**

- Subnet portion of address of arbitrary length
- Address format: a.b.c.d/x, where x is # bits in subnet portion of address

Dynamic Host Configuration Protocol (DHCP):

Goal: allow host to *dynamically* obtain its IP address from network server when it joins network

- Can renew its lease on address in use
- Allows reuse of addresses (only hold address while connected/on)
- Support for mobile users who want to join network

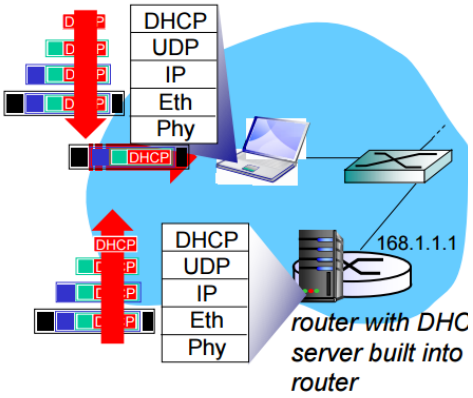


DHCP Overview for a scenario:

1. Host broadcasts "*DHCP discover*" msg [optional]
2. DHCP server responds with "*DHCP offer*" msg [optional]
3. Host requests IP address "*DHCP request*" msg
4. DHCP server sends address: "*DHCP ack*" msg

DHCP can return more than just allocated IP address on subnet. It can return address of first-hop router for client, name and IP address of DNS server and network mask (indicating network versus host portion of address)

Example of how DHCP works:

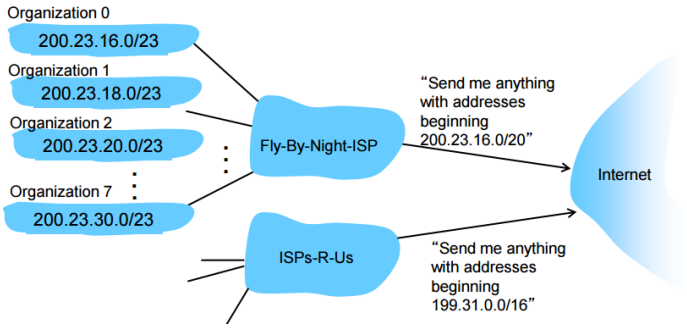


1. Connection laptop needs its IP address, addr of first-hop router, addr of DNS server: use DHCP. *DHCP DISCOVER* message encapsulated in UDP, IP and in Ethernet. Ethernet frame broadcast on LAN, received at router running DHCP server. Ethernet demuxed to IP demuxed to UDP demuxed to DHCP.
2. DHCP server formulates *DHCP OFFER* containing client's IP address, IP address of first-hop router for client, name and IP

address of DNS server. Encapsulation of DHCP server, frame forwarded to client, demuxing up to DHCP at client. Client now has a potential IP address, name and IP address of DNS server, IP address of its first-hoop router.

3. Laptop selects DHCP offer and prepares a DHCP request. *DHCP REQUEST* encapsulated in UDP, IP and in Ethernet. Ethernet frame sent by unicast on LAN, received at router running DHCP server. Ethernet demuxed to IP demuxed to UDP demuxed to DHCP.
4. DHCP server formulates *DHCP ACK* containing client's IP address, IP address of first-hop router for client, name and IP address of DNS server. Encapsulation of DHCP server, frame forwarded to client, demuxed up to DHCP at client. Client has the addressing information it needs.

How does network get subnet part of IP address? It gets allocated portion of its provider ISP's address space. Hierarchical addressing allows efficient advertisement of routing information. Router need a single entry in routing table, where anything destined to /20 goes on that link. Single entry that covers a huge chunk of IP address space and track an aggregated block of addresses instead of individual IP's.



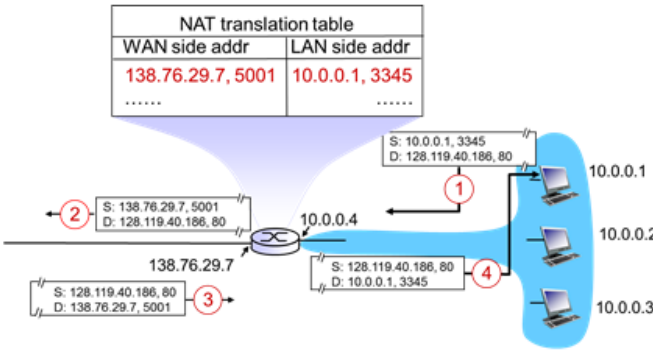
How does an ISP get block of addresses? ICANN allocate addresses, manages DNS and assigns domain names and resolve disputes.

Network addressing translation (NAT)

Local network uses just one IP address as far as outside world is concerned. Local network datagrams with source of destination in this network have 10.0.0/24 address for source and destination. Range of addresses not needed from ISP, just one IP address for all devices. Can change addresses of devices in local network without notifying outside world. Can change ISP without changing addresses of devices in local network. Devices inside local net not explicitly addressable, but visible by outside world.

Implementation: NAT router must:

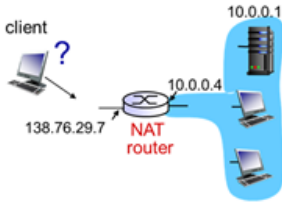
- Outgoing datagrams:
  - Replace (source #, port #) of every outgoing datagram to (NAT IP addr, new port #)
  - Remote clients/servers will respond using NAT IP addr
- Remember (in NAT translation table) every (source IP addr, port #) to (NAT IP addr, new port #) translation pair
- Incoming datagrams:
  - Replace (NAT IP #, new port #) in dest fields of every incoming datagram with corresponding (source IP addr, port number) stored in NAT table



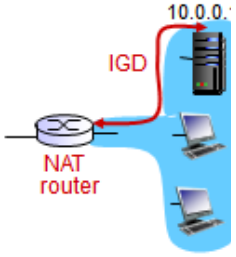
1. Host 10.0.0.1 sends datagram to 128.119.40.186, 80
2. NAT router changes datagram source addr from 10.0.0.1, 3345 to 128.76.29.7, 5001, updates table
3. Reply arrives dest. Addr: 138.76.29.7, 5001
4. NAT router changes datagram dest addr from 138.76.29.7, 5001 to 10.0.0.1, 3345

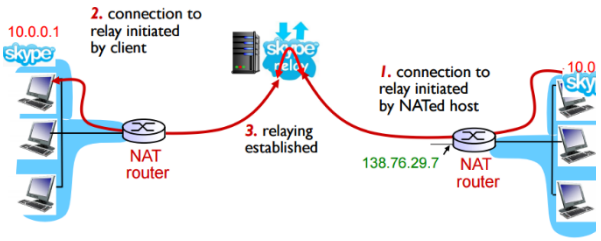
- 16-bit-port-number field with 60,000 simultaneous connections with a single LAN-side address
- NAT is controversial:
  - Routers should only process up to layer 3
  - Violates end-to-end argument
    - NAT possibly must be taken into account by app designers, e.g. P2P apps
  - Address shortage should instead be solved by IPv6

NAT traversal problem:



- Client wants to connect to server with address 10.0.0.1
  - Server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - Only one externally visible NATed address: 138.76.29.7
- *Solution 1:* statically configure NAT to forward incoming connection requests at given port server:
  - E.g. (123.76.29.7, 2500) always forwarded to (10.0.0.1, 25000)
- *Solution 2*: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATed host to:
  - Learn public IP address (138.76.29.7)
  - Add/remove port mappings (with lease times)
  - E.g. automate static NAT port configuration
- *Solution 3:* relaying (used in Skype)
  - NATed client establishes connection to relay
  - External client connects to relay
  - Relay bridges packets between to connections





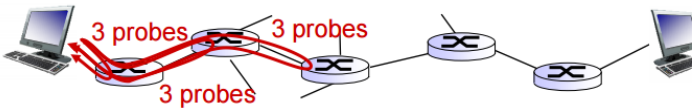
ICMP: Internet Control Message Protocol

Used by hosts and routers to communicate network-lever information such as error reporting: unreachable host, network, port, protocol or echo request/reply (used by ping). Network-layer "above" IP, where the ICMP messages carried in IP datagrams. ICMP message consist of type, code plus first 6 bytes of IP datagram causing error. Lives on NL which allows you to send error messages when things go bad. Most common is type 3 error.

Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachabl
3	1	dest host unreachable
3	2	dest protocol unreachabl
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congesti control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

Traceroute and ICMP

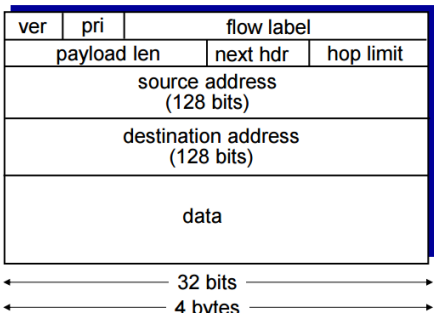
- Source sends series of ICMP messages to dest
  - First set has TTL = 1
  - Second set has TTL = 2, etc.
- When *n*th set of datagrams arrives to *n*th router:
  - Router discards datagrams and sends source ISMP messages (type II, code 0)
  - ICMP messages includes name of router and IP address.
- When ICMP messages arrives, source records RTTs
- Stopping criteria:
  - ICMP message eventually arrives at destination host
  - Destination return ICMP ping reply message
  - Source stops



Didrik Kvanvik

IPv6

Initial motivation: 32-bit address space soon to be completely allocated, header format helps speed processing/forwarding and header changes to facilitate QoS. IPv4 might run out of addresses, so IPv6 substantially increases address space. Instead of 32-bit addresses (1 billion), you can have 128-bit addresses. IPv6 datagram format: fixed-length 40 byte header and no fragmentation allowed.



- Priority: identify priority among datagram in flow
- Flow label: identify datagrams in same "flow"
- Next header: identify upper layer protocol for data

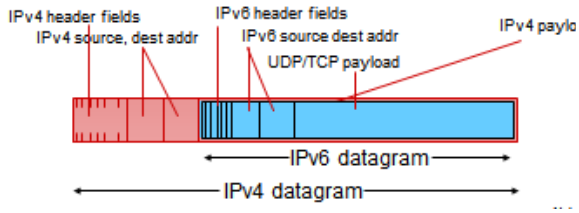
Changes from IPv4:

- Checksum: removed entirely to reduce processing time at each hop
- Options: allowed, but outside of header, indicated by "next header" field
- ICMPv6: new version of ICMP
    - Additional message types, e.g. "Packet too big"
    - Multicast group management functions

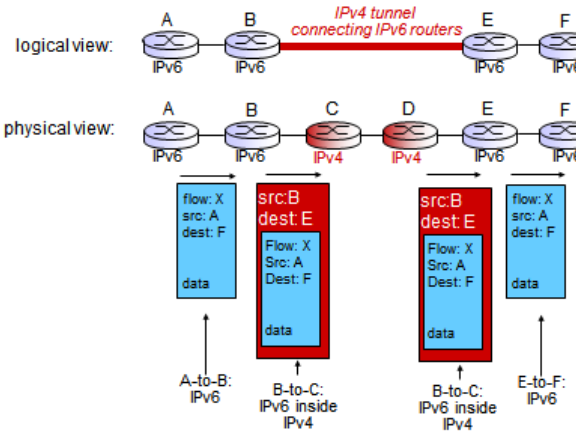
Transition from IPv4 to IPv6:

Not all routers can be upgraded simultaneously, which result in no "flag days".

Tunneling: IPv6 datagram carried as payload in IPv4 datagram among IPv4 routers. Looks in header of datagram to determine version.



A set of routers. In A, send packet to F. C, D speak IPv4. When B hits C, problem: solution is tunneling. Before B hands it over, it takes the IPv6 packet and sticks it inside an IPv4 packet and hands it to C. E opens ordinary IPv4 packet, and sees it as an IPv6 packet. Take one existing packet and sticking inside of another packet. Trick old router to deliver new packets called 'in IP encapsulation. IPv6 doesn't allow fragmentation in IPv4. IPv6 encapsulate in IPv4 (can be fragmented), then joins together in IPv4. IPv6 hardware and software upgrade required.



4.5 Routing algorithms

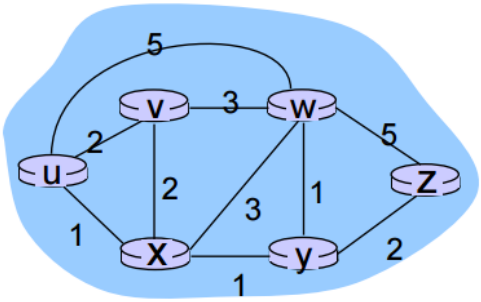
Assume routing table existed and somehow populated data, graph theory useful for routing algorithms. Routing algorithms determines end-to-end path through network, where forwarding table determines local forwarding at this router.

Graph abstraction

Write algorithms to parse and figure out where packets should go. Graph abstraction is useful in other network contexts, e.g. P2P, where N is set of peers and E is set of TCP connections. A routing algorithm is an algorithm that finds that least cost path through network.

Didrik Kvanvik

Graph: G = (V, E)



V: is the set of routers {u, v, w, x, y, z}

E: is set of links = {(u, v), (u, x), (v, x), (v, w), (x, w), (x, y), (w, y), (w, z), (y, z)}

- c(m, n) : cost of link (m, n). e.g. c(w, z)= 5
- cost >= 0
- Cost could always be 1, or inversely related to bandwidth, or inversely related to congestion.

cost of path (x_1 , x_2 , x_3 ,…, x_p) = c(x_1 ,x_2) + c(x_2 ,x_3) + … + c($_{xp-1}$,x_p)

The key question: what is the *least-cost* path between x and y? The points an algorithm focuses on:

- **Complexity of computation and communication**
- **Centralized or decentralized**
  - Centralized:
    - All routers have complete topology, link cost info.
    - “link state” algorithms
  - Decentralized:
    - Router knows physically-connected neighbors, link cost to neighbors
    - Iterative process of computation, exchange of info with neighbors
    - “Distance vector” algorithms
- **Static or dynamic**
  - Static:
    - Routes change slowly over time
  - Dynamic:
    - Routes change more quickly
      - Periodic update
      - In response to link cost changes
- **Least-cost in what sense?**
- **Single-path or multipath**
- **Unicast or multicast**

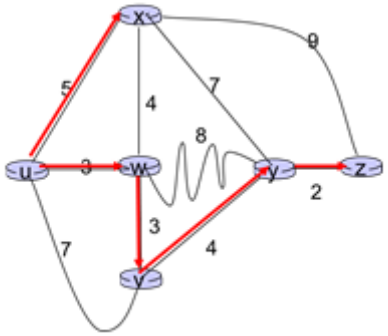
Link state:

The basic concept of link-state routing is that every node constructs a map of the connectivity to the network, in the form of a graph, showing which nodes are connected to which other nodes. Each node then independently calculates the next best logical path from it to every possible destination in the network. The collection of best paths will then form the node’s routing table. So each router gathers info about adjacent links, collects this info in link state packets, broadcasts Is packets to all routers. An algorithm used is the Djikstra’s algorithm.

Didrik Kvanvik

Djikstra's

Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					



Net topology, link costs known to all nodes. All nodes have same information. Computes least cost paths from one node (source) to all other nodes. Iterative: after k iterations, know least cost path to k destinations. First it designate state node. Then finds out what nodes are directly attached – put in candidate list. The main loop consist of adding one new node each time by searching for the most promising candidate and if that makes it better than before. It constructs shortest path tree by tracing predecessor nodes.

- Algorithm complexity: n nodes
  - Each iteration: need to check all nodes not in N
  - N(n+1)/2 comparisons: $O(n^2)$
  - More efficient implementation possible: O(n log n)
- Oscillations possible:
  - E.g., support link cost equals amount of carried traffic

Distance vector algorithm

$D_x(y)$ = estimate of least cost from x to y, then

$D_x(y) = \min \{c(x, v) + d_v(y)\}$

- min = taken over all neighbor x of x
- c(x, v) = cost to neighbor v
- $d_v(y)$ = cost from neighbor v to destination y
- x maintains distance vector $D_x = [D_x(y): y \in N]$

Node x knows cost to each neighbor v: c(x, v) and maintains its neighbor's distance vectors. For each neighbor v, x maintains $[D_v = D_v(y): y \in N]$. Key idea is that from time-to-time, each node sends its own distance vector estimate to neighbors, when x receives new DV estimate from neighbor, it updates its own DV using Bellman-Ford equation $D_x(y)$. It's an iterative and asynchronous algorithm where each local processing caused by local ink cost change and DV update message from neighbor. Distribution: each node notifies neighbors only if its DV changes or after a certain time has passed since the last update.

```
each node
    wait for (change in local link cost or message from neighbo
        recompute estimates
        IF DV to any destination has changed, notify neighbors
```

Link cost changes:

- Good news travel fast:
  - node detects local link cost change
  - updates routing info, recalculates distance vector
  - if DV changes, notify neighbor
    - t_0: y detects link-cost change, updates its DV, informs its neighbors
    - t_1: z receives update from y, updates its table, computes new least cost to x, sends its neighbors its DV
    - t_2: y receives z's update, updates its distance table. Y's least costs does not change, so y does not send a message to z.
- Bad news travel slow:
  - Node detects local link cost change
  - Bad news travels slow – "count to infinity" problem

- 44 iterations before algorithm stabilizes
- Poisoned reverse:
  - If > routes through Y and to get X:
    - Z tells Y it's (Z's) distance to X is infinite (so Y won't route to X via Z)
- Conflict – don't route through me because I'm routing through you, infinite loop because of different numbers
  - Isolation, bouncing message back and forth

Analysis:

- Iterative, asynchronous
- Link cost changes:
  - Good news travel fast:
    - Shortest path updates quickly
  - Bad news travels slow (count-to-infinite problem)
    - An increase change
    - Decision based off own table, don't see everyone else's changes

Comparison of LS and DV algorithms

- *Message complexity*
  - LS: with n nodes, E links, O(nE), msgs sent
  - DV: exchange between neighbors only, less data exchange
- *Speed of convergence*
  - LS: O(n2) algorithm requires O(nE) msgs, may have oscillations
  - DV: convergence times varies:
    - Loop networks: count-to-infinity problem
    - Still a problem with basic fixes, but can be solved
- *Robustness*
  - LS:
    - Node can advertise incorrect *link* cost
    - Each node computes only its own table
  - DV:
    - DV node can advertise incorrect path cost
    - Each node's table used by others
      - Error propagate thru network

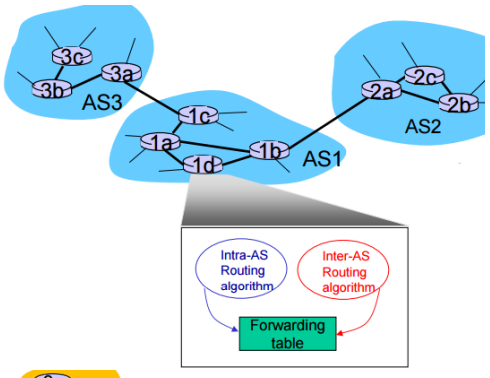
Hierarchical routing

- It makes internet scalable
  - Not of network prefixes/subnets, not on host IDs
  - Route aggregation: when router has a set of routing table entries that go to different prefixes, and then aggregate to a single entry
  - Hierarchical design: instead of having a single routing algorithm, break it up into inter and intra-AS routing of autonomous system
- Out routing study thus far – idealization
  - All routers identical, network "flat", not true in practice
- Scale: with 6000 million destinations
  - Can't store all destinations in routing tables, routing table exchange would swamp links

Didrik Kvanvik

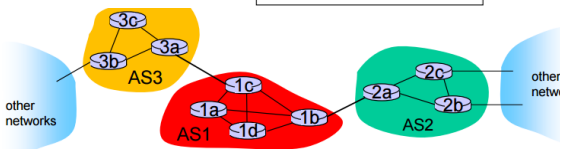
- Administrative autonomy
    - Internet = network of networks
    - Each network admin may want to control routing in its own network

Aggregate routers into regions, “autonomous systems” (AS). Routers in same AS run same routing protocol. That is “intra-AS” routing protocol where routers in different AS can run different intra-AS protocol. Gateway router: at “edge” of its routing protocol and has link to router in another AS.

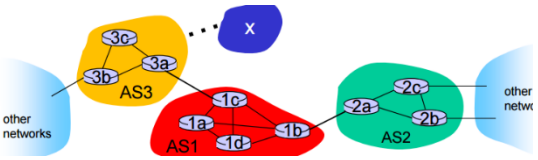


Interconnected ASes: forwarding table configured by both intra- and inter-AS routing algorithm. Intra-AS sets entries for internal destinations, inter-AS and intra-AS sets entries for external destinations.

Inter-AS tasks: suppose router in AS1 receives datagram destined outside of AS1: router should forward packet to gateway router, but to which one? AS1 must learn which destinations are reachable through AS2, which are through AS3. Propagate this reachability info to all routers in AS1.



<u>Example: setting forwarding table in router 1d:</u>



- Suppose AS1 learns (via inter-AS protocol) that subnet x reachable via AS3 (gateway 1c), but now via AS2 (gateway 1b).
    - Inter-AS protocol propagates reachability info to all internal router
- Router 1d determines from intra-AS routing info that its interface a is on the last cost path to 1c
    - Installs forwarding table entry (x, a)

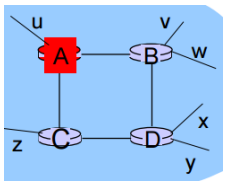
<u>Example: choosing among multiple ASes:</u>

- Now suppose AS1 learns from inter-AS protocol that subnet x is reachable from AS3 and from AS2.
- To configure forwarding table, router 1d must determine which gateway it should forward packets towards for dest x
    - This is also the job of inter-AS routing protocol
- *Hot-potato routing:* send packet towards closest of two routers
    1. Learn from inter-AS protocol that subnet x is reachable via multiple gateways
    2. Use routing info from intra-AS protocol to determine costs of least-cost paths to each of the gateways
    3. Hot-potato-routing: choose the gateway that has the smallest least cost
    4. Determine from forwarding table the interface I that leads to least-cost gateway. Enter (x, I) in forwarding table.

4.6 Routing in the Internet

Intra-AS routing is also known as *interior gateway protocols (IGP)*. Most common intra-AS routing protocols: Routing Information Protocol (RIP), Open Shortest Path First (OSPF), Interior Gateway Routing Protocol (IGRP).

Routing Information Protocol



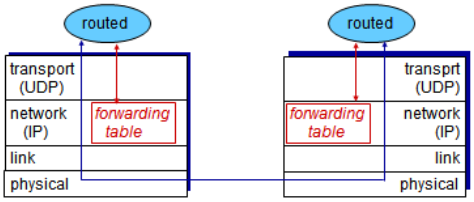
from router A to destination *subnet*

subnet	hops
u	1
v	2
w	2
x	3
y	3
z	2

- Includes in BSD-UNIX distribution in 1982.
- Distance vector algorithm
  - Distance metric: # hops (max = 15 hops), each link has cost I
  - DVs exchanged with neighbors every 30 sec in response message (aka advertisement)
  - Each advertisement: list up to 25 destination subnets (in IP addressing sense)

Link failure and recovery: if no advertisement is heard after 180 seconds, then neighbor/link declared dead. Routes via neighbor invalidated, new advertisements sent to neighbors, neighbors in turn send out new advertisements if tables changed, and link failure info quickly propagates to entire net. Poison reverse used to prevent ping-pong loops (infinite distance = 16 hops)

Table processing: RIP routing tables managed by *application-level* process called route-d (daemon). Advertisements sent in UDP packets, periodically repeated.



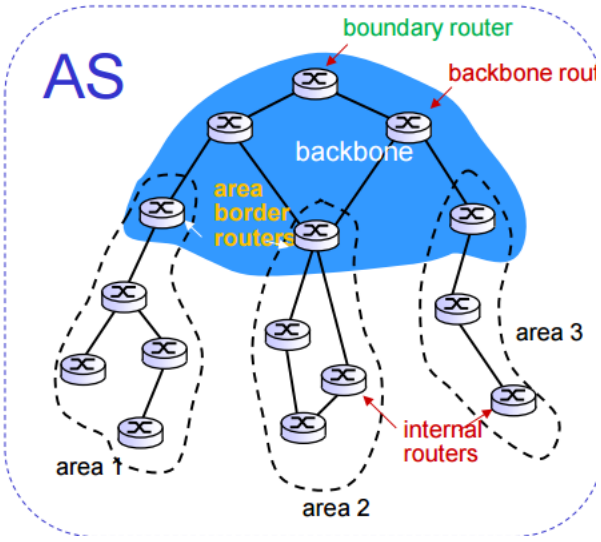
Open Shortest Path First

Is "open", that is publicly available. It uses link state algorithm with LS packet dissemination, topology map at each node and route computation using Djikstra's algorithm. OSPF advertisement carries one entry per neighbor. Advertisements flooded to entire AS, carried in OSPF messages directly over IP rather than TCP or UDP. OSPF features not in RIP:

- Security: all OSPF messages authenticated (to prevent malicious intrusion)
- Multiple same-cost paths allowed (only one path in RIP)
- For each link, multiple cost metrics for different ToS.
- Integrated uni- and multicast support
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- Hierarchical OSPF in large domains

Hierarchical OSPF:

- Two-level hierarchy: In AS: local area, backbone area
- Area border routers: "summarize" distances to nets in own area, advertise to other Area Border routers
- Backbone routers: run OOSPF routing limited to backbone
- Boundary routers: connect to other AS's.

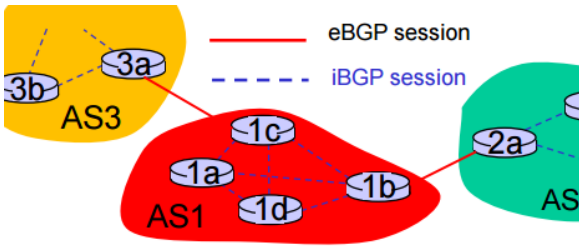


Didrik Kvanvik

Internet inter-AS routing: BGP

Border Gateway Protocol is an inter-domain routing protocol. It provides each AS a means to eBGP: obtain subnet reachability information from neighboring AS's, iBGP: propagate reachability information to all AS-internal routers and determines "good" routes to other networks based on reachability information and policy. BGP allows subnet to advertise its existence to the rest of the Internet.

- BGP session: two BGP routers ("peers") exchange BGP messages
  - Advertising paths to different destination network prefixes ("path vector" protocol)
  - Exchanged over semi-permanent TCP connections
- When AS3 advertises a prefix to ASI:
  - AS3 promises it will forward datagrams towards that prefix
  - AS3 can aggregate prefixes in its advertisement
- Distributing path information:
  - Using eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1
    - 1c can then use iBGP do distribute new prefix info to all routers in AS1
    - 1b can the re-advertise new reachability info to AS2 over 1b-to-2a eBGP session.
  - When router learns of new prefix it creates entry for prefix in its forwarding table



Path attributes and BGP routes:

- Advertised CIDR prefix includes BGP attributes
- Two important attributes
  - AS-PATH: contains Ass through which prefix advertisement has passed: e.g. AS 67, AS 17
  - NEXT-HOP: indicates specific internal-AS router to next-hop AS
- Gateway router receiving route advertisement uses import policy to accept/decline
  - E.g. never route through AS x
  - Policy-based routing

BGP route selection

Router may learn about more than one route to destination AS, selects route based on:

1. Local preference value attribute: policy decision
2. Shortest AS-PATH
3. Closes NEXT-HOP router: hot-potato routing
4. Additional criteria

BGP messages

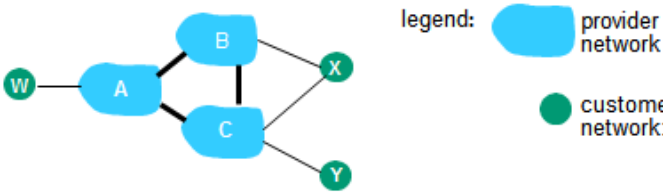
BGP messages exchanged between peers over TCP connection. It consists of:

- OPEN: opens TCP connection to peer and authenticates sender
- UPDATE: advertises new path (or withdraws old)
- KEEPALIVE: keeps TCP connection alive in absence of UPDATES; also ACKs OPEN request
- NOTIFICATION: reports errors in previous msg: also used to close connection.

Didrik Kvanvik

BGP routing policy

A, B and C are provider networks. X, W, Y are customer of provider networks. X is dual-homed: attached to two networks. X does not want to route from B via X to C, so X will not advertise to B a route to C. A advertises path AW to B. B advertises path BAW to X. Should then B advertise path BAW to C? No! B wants to force C to route to W via A and B wants to route only to/from its customers.

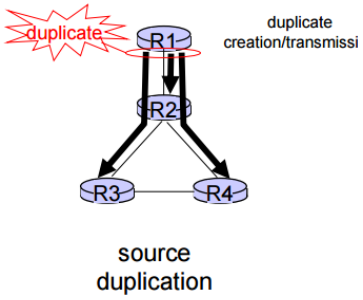


Different Intra- Inter-AS routing:

- Policy
  - Inter-AS: admin wants control over how its traffic routed, who routes through its net
  - Intra-AS: single admin, so no policy decisions needed
- Scale
  - Hierarchical routing saves table size, reduced update traffic
- Performance
  - Intra-AS: can focus on performance
  - Inter-AS: policy may dominate over performance

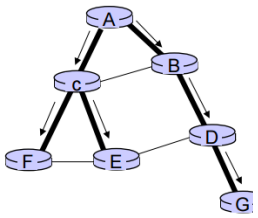
4.7 Broadcast routing

Deliver packets form source to all other nodes. Source duplication is inefficient. Source duplication how does source determine recipient addresses?



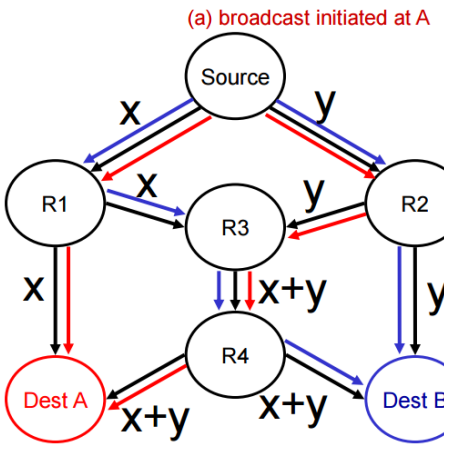
In-network duplication

- *Flooding:* when node receives broadcast packet, sends copy to all neighbors
  - Problem: cycles and broadcast storm
- *Controlled flooding:* node only broadcast pkt if it hasn't broadcast same packet before
  - Node keeps track of packet ids already broadcasted
- *Reverse path forwarding:* only forward packet if it arrived on shortest path between node and source
- *Spanning tree:* no redundant packets received by any node



First construct a spanning tree. Nodes then forward/make copies only along spanning tree.

Multicast routing: network coding: traditional multicast methods are inefficient compared to network coding. A and B destinations can both recover information this way.



- Routing
  - - Optimum routes hard to find
  - - Random routing inefficient
  - - Requires more packets sent
  - - Requires separate retransmission for each source/lost packet
  - + Allows variable packet size

- Network Coding
  - + Optimum routes easy to find
  - + Random coding efficient
  - + Fewer packets, more energy efficient, less delay
  - + Adapts easily to packet loss
  - - Requires fixed packet size

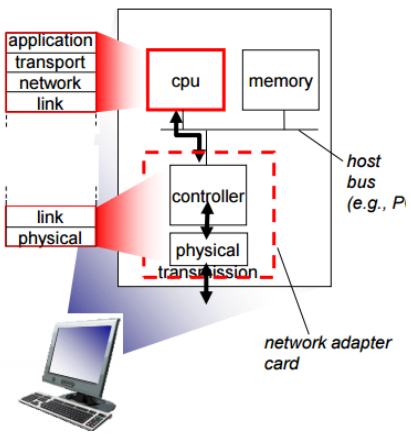
Chapter 5 – Link layer

Goals: understand principles behind link layer services such as error detection, correction, sharing a broadcast channel with multiple access, link layer addressing and local area networks such as Ethernet and VLANs, and implementation of various link layer technologies.

5.1 Introduction, services

Terminology: hosts and routers are referred to as nodes, communication channels that connect adjacent nodes along communication path are links (wired, wireless links and LANs). Layer-2 packet: frame, encapsulates datagram. *Data-link layer has responsibility of transferring datagram from one node to physically adjacent node over a link.* Datagram transferred by different link protocols over different links; 4G over first link, Ethernet on intermediate links, 802.11 on last link. Each link protocol provides different service.

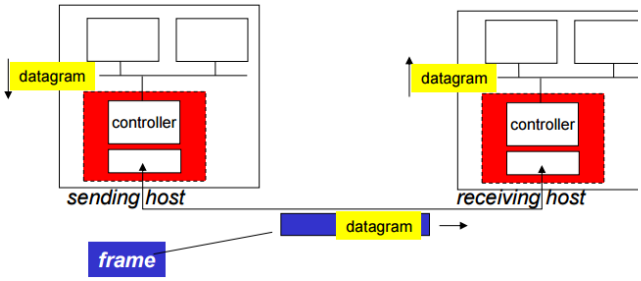
- Half-duplex and full duplex
  - With half duplex, nodes at both ends of link can transmit, but not at same time
- Reliable delivery between adjacent nodes
  - Much used in the link layer, but coding more advanced
  - Wireless links: high error rates
- Link access
  - Channel access if shared medium
  - "MAC" addresses used in frame headers to identify source and destination
    - Different from IP address
- Flow control
  - Pacing between adjacent sending and receiving nodes
- Error detection
  - Errors caused by signal attenuation, noise
  - Receiver detects presence of errors
    - Signals sender for retransmission or drops frame
- Error correction
  - Receiver identifies and *corrects* bit error(s) without resorting to retransmission
- Provide services by framing
  - Encapsulate datagram into frame, adding header, trailer



The link layer is implemented in each and every host. Link layer implemented in "adaptor" (aka network interface card NIC) or on a chip. Ethernet card, 802.11 card; Ethernet chipset. Attaches into host's system buses, is a combination of hardware, software and firmware.

Adaptors communication:

- Sending side:
  - Encapsulates datagram in frame
  - Adds error checking bits, rdt, flow control etc.
- Receiving side:
  - Looks for errors, rdt, flow control etc.
  - Extracts datagram, passes to upper layer at receiving side



5.2 Error detection and correction

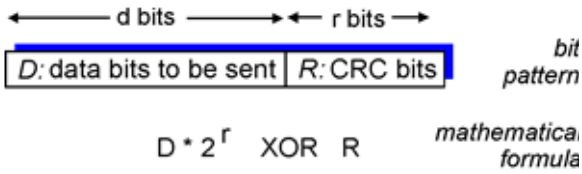
Error detection is noticing there is a problem, while error correction is noticing there is a problem and fixes it.

Error Detection

Error detection ads sufficient redundancy to a frame so that a receiver, with high probability, can detect the presence of an error and discard the frame. Send extra information across channel, control bits, to make it bigger and give you redundancy. If you detect a problem, throw away the frame and rely on something else to deal with it.

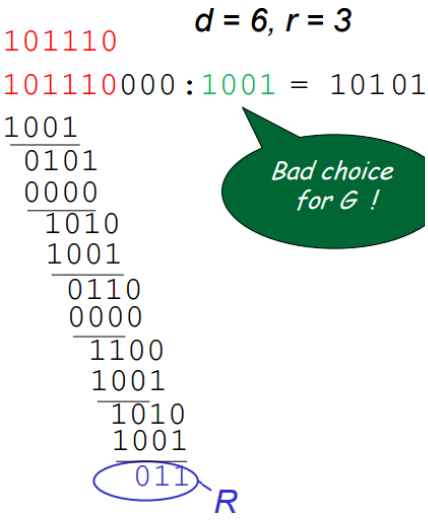
Cyclic redundancy code (CRC)

R bit checksum on frames. Is extremely powerful with 99.9 % detection rate and is widely used. It's based on polynomial arithmetic in base two, on large data frame on some size D, add r bits of checksum and that checksum can detect errors. Binary arithmetic. It's a more powerful error-detection coding. It views d data bits, D, as a binary number. Choose r+1 bit pattern (generator), G. The goal is to choose r CRC bits, R such that <D, R> exactly divisible by G (modulo 2), receiver knows G, divides <D, R> by G. If a non-zero remainder: error detected! Can detect all burst errors less than r+1 bits. It's widely used in practice with Ethernet, 802.11 WiFi and ATM.



Example of CRC:

- Want: $D * 2^r$ XOR $R = nG$
- Equivalently: $D * 2^r = nG$ XOR R
- Equivalently: if we divide $D * 2^r$ by G, want remainder R to satisfy:
  - $R = \text{remainder}[(D * 2^r) / G]$



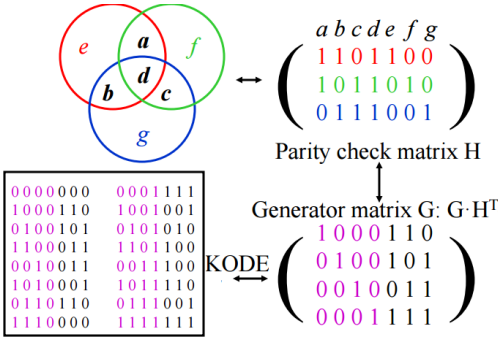
Error detection is simple, but requires return channel, which can result in extra delay and return channel may not exist.

Error correction

Adds sufficient redundancy to a message so that, with high probability, the receiver can detect the presence of an error, and fix it themselves without requiring retransmission.

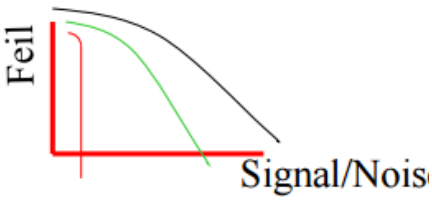
Example with parity check: Hamming codes

Data:	HP		
[101101]	[1]		
[000101]	[1]		
[000110]	[1]	→	[000010]?
[011011]	[1]	→	[011011]? VP is fine, but parity bit was changed
VP	[001010]		



It's less space consuming, detects and corrects single bit errors and run through data and parity bits and check for errors.

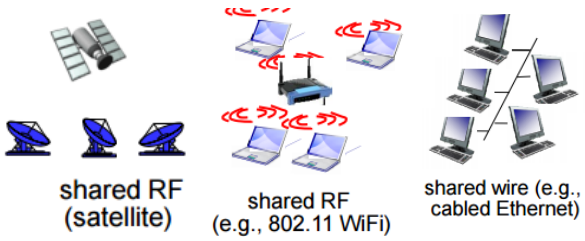
- Areas:
  - Data transmission
    - Modems, mobile phones, satellite connections, deep space communication, microwave links, power line communication, optical fibers, network coding
  - Data storage
    - CD's, DVD's, hard disks, flash memories, RAM, distributed storage in clouds
  - Others: birth number, ISBN, credit cards and bank account numbers
- Advantages:
  - Increased reliability
  - Less overhead for a given level of reliability
  - Increased transmission speed/Storage capacity



5.3 Multiple access protocols

There are two types of "links":

- Point-to-point
  - PPP for dial-up access
  - Point-to-point link between Ethernet switch, host
- Multiple access and broadcast (shared wire or medium)
  - Old-fashioned Ethernet
  - Upstream HFC
  - 802.11 wireless LAN



Multiple access protocols

Single shared broadcast channel, two or more simultaneous transmissions by nodes: interference, which result in collision if node receives two or more signals at the same time. Multiple access protocol: distributed algorithm that determines how nods share channel, i.e. determine when nodes can transmit, communication about channel sharing must use channel itself (no out-of-band channel for coordination). Ideal MAP: given a broadcast channel of rate R bps

1. When one node wants to transmit, it can send at rate R
2. When M nodes wants to transmit, each can send at average rate R/M
3. Fully decentralized:
   - No special node to coordinate transmissions
   - No synchronization of clocks, slots
4. Simple

Didrik Kvanvik

MAC protocols: Taxonomy

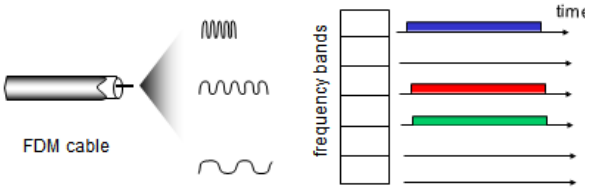
In the even-layer OSI model of computer networking, media access control (MAC) data communication protocol is a sublayer of the data link layer (layer 2), that provides addressing and channel access control mechanisms that make it possible for several terminals or network nodes to communicate within a multiple access network that incorporates a shared medium, e.g. a Ethernet network. There are four broad classes:

- Channel partitioning
  - Divide channel into smaller "pieces" (time slots, frequency, code)
  - Allocate pieces to node for exclusive use
  - Share channel efficiently and fairly at high load and inefficient at low lad (delay)
- Random access
  - Channel not divided, allow collisions
  - "recover" from collisions
  - Efficient at low load where single node can fully utilize channel
  - High load: collision overhead
- "taking turns"
  - Nodes take turns, but nodes with more to send can take longer turns
  - Uses best of both worlds
  - Polling from central site, token passing
  - Bluetooth, FFDDI, token ring
- Spread spectrum/CDMA

Channel partitioning MAC protocols:

TDMA: Time Division Multiple Access. Access to channel in "rounds". Each station gets a fixed length slot (length = pkt transfer time) in each round, where unused slots go idle. Example: 6-station LAN, 1, 3 and 4 have pkt, slots 2, 5 and 6 are idle.

FDMA: Frequency Division Multiple Access. Channel spectrum divided into frequency bands, where each station is assigned a fixed frequency band. Unused transmission time in frequency bands will go idle. Example: 6-station LAN, 1, 3 and 4 have pkt, frequency bands 2, 5 and 6 are idle.



CDMA: Code Division Multiple Access. Spread spectrum and are heavily used in wireless.

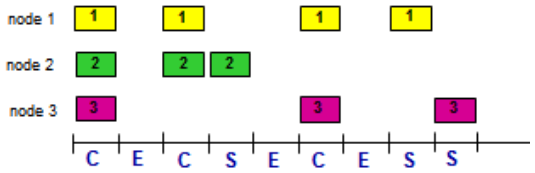
Random Access Protocols

When node has packet to send it transmit at full channel data rate R, and no priori coordination among nodes. If two or more nodes transmitting at once, will result in collision. Random Access MAC protocol specifies how to detect collisions and how to recover from collisions (e.g. via delayed retransmission). Example of Random Access MAC protocols: slotted ALOHA, ALOHA, CSMA, CSMA/CD and CSMA/CA.

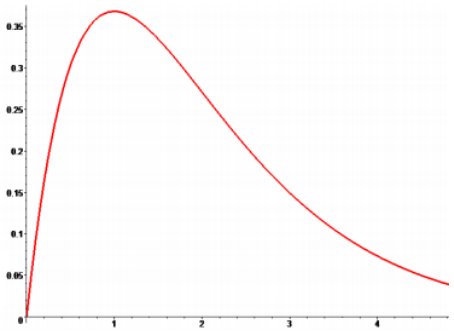
Didrik Kvanvik

Slotted ALOHA

```
When a station has a frame to sen
   wait for next slot
   send the frame
IF collision occurs
THEN wait a random time
```

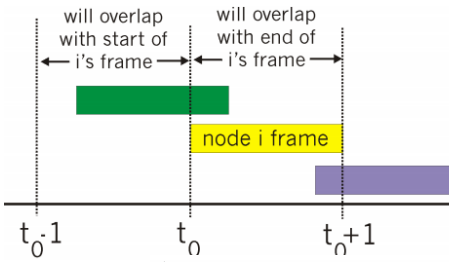


- Assumptions:
  - All frames same size
  - Time divided into equal size slots
  - Nodes start to transmit only slot beginning
  - Nodes are synchronized
  - If 2 or more nodes transmit in slot, all nodes detect collision
- Operation:
  - When node obtains frame, transmits in next slot
    - If no collision: node can send new frame in slot
    - If collision: node retransmits frame in each subsequent slot with prob. P until success.
- Pros:
  - Single active node can continuously transmit at full rate of channel
  - Highly decentralized: only slots in nodes need to be in sync
  - simple
- Cons:
  - Collisions and idle slots are wasted slots
  - Nodes may be able to detect collision in less than time to transmit packet
  - Clock synchronization
  - Rude



Efficiency: long-run fraction of successful slots (many nodes, all with many frames to send). Suppose N nodes with many frames to send, each transmit in slot with probability *p*. probability that given node has success in a slot = $p(1-p)^{N-1}$. Probability that any node has a success: $Np(1-p)^{N-1}$. Max efficiency: find p* that maximizes $Np(1-p)^{N-1}$. For many nodes, take limit of $Np(1-p)^{N-1}$ as N goes to infinity. At best: channel used for useful transmission 37 % of time.

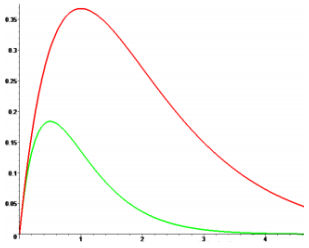
Pure (unslotted) ALOHA



Unslotted Aloha is simpler without the synchronization. When a frame first arrives it's transmitted immediately. Collision probability is increased: frame sent at t0 collides with other frames sent in $[t_0-1, t_0+1]$. Efficiency:

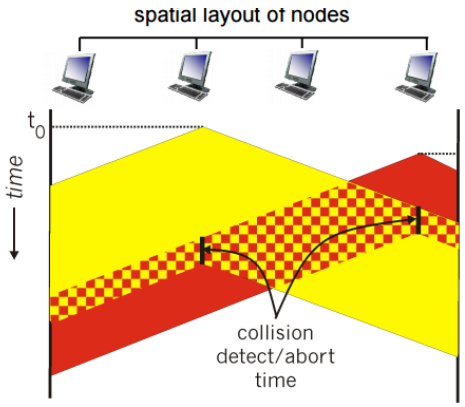
P (success by given node) = P (node transmits) * P (no other node transmits in $[t_0-1, t_0]$) * P (no other node transmits in $[t_0, t_0+1]$)

$= p * (1-p)^{N-1} * (1-p)^{N-1} = p * (1-p)^{2(n-1)}$. Choosing optimum p and letting N → infinity = 1 / (2e) = 0.18. Which is even worse than slotted aloha →



Didrik Kvanvik

Carrier Sense Multiple Access (CSMA): Listens before transmit if channel sensed idle: transmit entire frame, or if channel sensed is busy, defer transmission. Pretty much, just don't interrupt. Collisions can still occur: propagation delay means two nodes may not hear each other's transmission. Collision: entire packet transmission time is wasted (distance and propagation delay play role in determining collision probability).



CSMA/CD: is carrier sensing, deferral as in CSMA. Collisions are detected within short time. Colliding transmissions are aborted, which reduces channel wastage. Collision detection: easy in wired LANs: measure signal strengths, compare transmitted, received signals. It is difficult in wireless LAN's because received signal strength is overwhelmed by local transmissions strengths. The human analogy: the polite conversationalist.

D_{prop} = max prop delay between 2 nodes in LAN

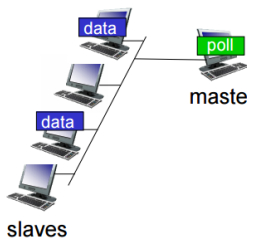
D_{trans}= time to transmit max-size frame

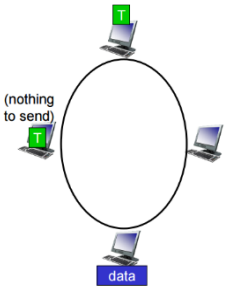
Efficiency: $\frac{1}{1+2e \cdot d_{prop}/d_{trans}} \approx \frac{1}{1+5.4d_{prop}/d_{trans}}$

Efficiency goes to 1 as d_{prop} goes to 0, and as d_{trans} goes to infinity. It has better performance than Aloha and is simple, cheap and decentralized.

"Taking turn" MAC protocols

- channel partitioning MAC protocols
  - share channel *efficiently* and *fairly* at high load
  - inefficient at low load: delay in channel access, 1/N bandwidth allocated even if only 1 active node
- random access MAC protocols
  - efficient at low load: single node can fully utilize channel
  - high load: collision overhead
- "taking turns" protocols look for best of both worlds
- *Polling:*
  - Master node "invites" slave nodes to transmit in turn
  - Typically used with "dumb" slave devices
  - Concerns:
    - Polling overhead
    - Latency
    - Single point of failure (master)
- *Token passing:*
  - Control token passed from one node to next sequentially
  - Token message
  - Concerns: same as with polling, but SPF is with token.





5.4 LANs

MAC addresses and ARP

- 32-bit IP address
- MAC (or LAN or physical or Ethernet) address:
  - Function: used "locally" to get frame from one interface to another physically-connected interface (same network, in IP-addressing sense)
  - 48-bit MAC address (for most LANs) burned in NIC ROM, also sometimes software settable
  - E.g. 1A-2F-BB-76-09-AD (Hexadecimal (base 16) notation, each # represents 4 bits)

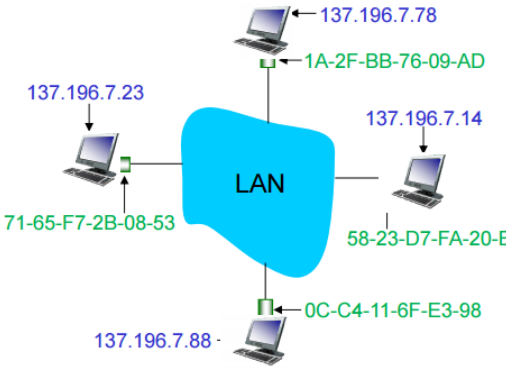
LAN addresses and ARP

- Each adapter on LAN has a unique LAN address, where MAC address allocation administered by IEEE.
- Manufacturer buys portion of MAC address space.
- Analogy:
  - MAC address: like Social Security Number
  - IP address: like postal address
- MAC flat address → portability
  - Can move LAN card from one LAN to another
- IP hierarchical address not portable
  - Address depends on IP subnet to which node is attached

Address Resolution Protocol (ARP)

Protocol for being able to map from MAC addresses to IP addresses.
The ARP table:

- Tracks IP-MAC address mapping.
- Each IP node (host, router) on LAN has table
- IP/MAC address mappings for some LAN nodes: <IP address; MAC address; TTL>
- TTL (Time to Live): time after which address mapping will be forgotten (~20 min)

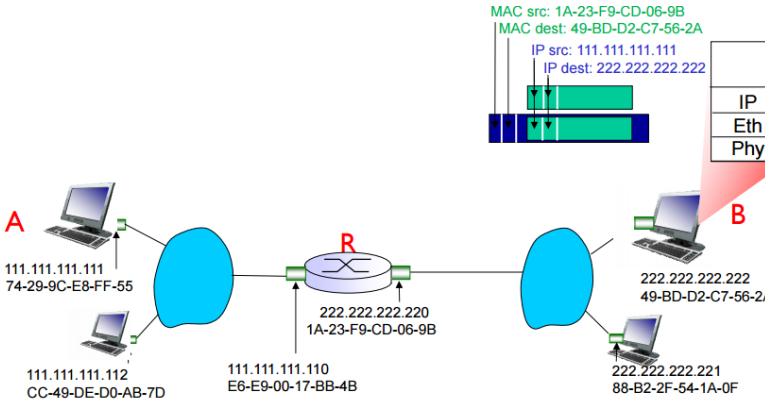


ARP protocol: same LAN

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

Didrik Kvanvik

Addressing: routing to another LAN

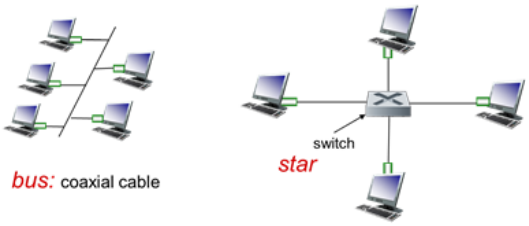


(1) Walkthrough: send datagram from A to B via R
    a. Focus on addressing – at IP (datagram) and MAC layer (frame)
    b. Assume A knows B's IP address
    c. Assume A knows IP address of first hop router, R
    d. Assume A knows R's MAC address
(2) A creates IP datagram with IP source A, destination B
(3) A creates link-layer frame with R's MAC address as (destination, frame) contains A-to-B IP datagram
(4) Frame sent to A to R
(5) Frame received at R, datagram, removed, passed up to IP
(6) R forwards datagram with IP source A, destination B
(7) R creates link-layer frame with B's MAC address as (destination, frame) contains A-to-B IP datagram

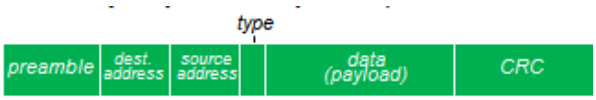
Ethernet

Most popular LAN technology, cheap NICs, first widely used LAN technology, simpler and cheaper than token LANs and ATM, kept up with speed race: 10 Mbps – 10 Gbps.

- Bus: popular through mid-90s
    - All nodes in same collision domain (can collide with each other)
- Star: prevails today
    - Active *switch* in center
    - Each «spoke» runs a (separate) Ethernet protocol (nodes do not collide with each other)



<u>Ethernet frame structure</u>: sending adapter encapsulates IP datagram (or other network layer protocol packet) in Ethernet frame. Preamble: 7 bytes with pattern 10101010 followed by one byte with pattern 10101011. Used to synchronize receiver, sender clock rates.



- Addresses: are 6 byte source, destination MAC addresses
    - If adapter receives frame with matching destination address, or with broadcast address, it passes data in frame to network layer protocol
    - Otherwise, adapter discards frame
- Type: indicates higher layer protocol (mostly IP but others possible)
- CRC: cyclic redundancy check at receiver
    - Error detected: frame is dropped

<u>Ethernet: unreliable and connectionless</u>

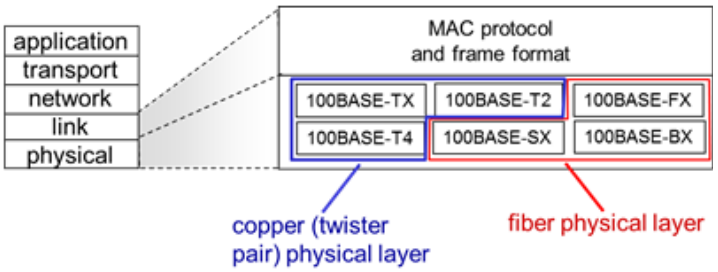
- Connectionless: no handshaking between sending and receiving ICs
- Unreliable: receiving NIC doesn't send acks or nacks to sending NIC

- Data in dropped frames recovered only if initial sender uses higher layer rdt (e.g. TCP), otherwise dropped data lost
- Ethernet's MAC protocol: unslotted CSMA/CD with binary backoff

802.3 Ethernet Standards: link and physical layers

Many different Ethernet standards:

- Common MAC protocol and frame format
- Different speeds: 2, 10 and 100 Mbps, 1, 10, 100 Gbps
- Different physical layer media: fiber, cable
- Improve Ethernet:
  - Restrict distance
  - Bigger frames
  - Fewer stations: switched Ethernet

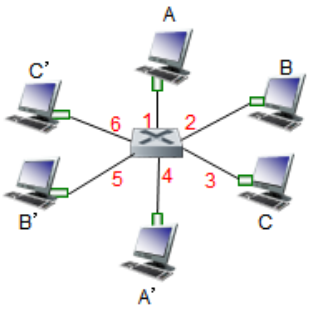


Ethernet CSMA/CD algorithm

(1) NIC receives datagram from network layer, creates frame
(2) If NIC senses channel idle, starts frame transmission. If NIC senses channel busy, waits until channel idle, then transmits
(3) IF NIC transmits entire frame without detecting another transmission, NIC is done with frame
(4) If NIC detects another transmission while transmitting, aborts and sends jam signal
(5) After aborting, NIC enters *exponential backoff*: after *m*th collision, NIC chooses K at random from {0, 1, 2… 2^m-1}. NIC waits K*512 bit times, return to step 2.

Ethernet switch

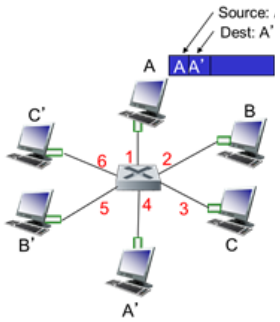
- Link-layer device: takes an active role
  - Store, forward Ethernet frames
  - Examine incoming frame's MAC address, selectively forward frame to one-or-more outgoing links when frame is to be forwarded on segment, uses CSMA/CD to access segment
- Transparent
  - Hosts are unaware of presence of switches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



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

Multiple simultaneous transmissions: hosts have dedicated, direct connection to switch, switches buffer packets. Ethernet protocol used on *each* incoming link, but no collisions; full duplex where each link is its own collision domain. Switching: A-to-A' and B-to-B' can transmit simultaneously, without collisions. Each switch has a switch table, with each entry has a MAC address of host, interface to reach host and time stamp.

Switch learns which hosts can be reached through which interfaces. When frame received, switch "learn" location of sender: incoming LAN segment. Records then sender/location pair in switch table.

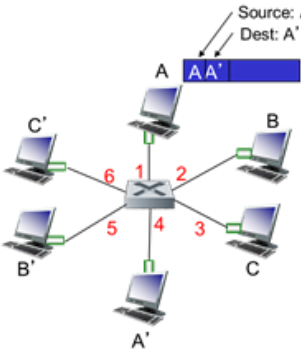


MAC addr	interface	TTL
A	*1*	*60*

Switch table (initially empty)

Frame filtering/forwarding: When frame received at switch:

(1) Record incoming link, MAC address of sending host
(2) Index switch table using MAC destination address
(3) If entry found for destination
    a. Then if destination on segment from which frame arrived
        i. Then drop frame
        ii. Else forward frame on interface indicated by entry
    b. Else flood: forward on all interfaces except arriving interface



Self-learning, forwarding example:

Frame destination, A', location unknown: *flood*

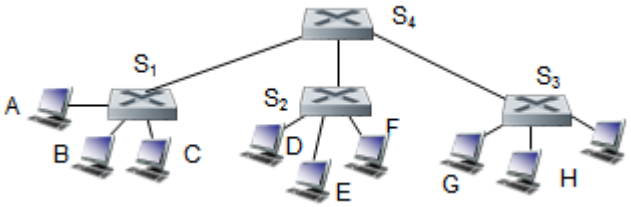
Destination A location known: selectively send on just one link →

MAC addr	interface	TTL
A	*1*	*60*

Switch table (initially empty)

Interconnecting switches:

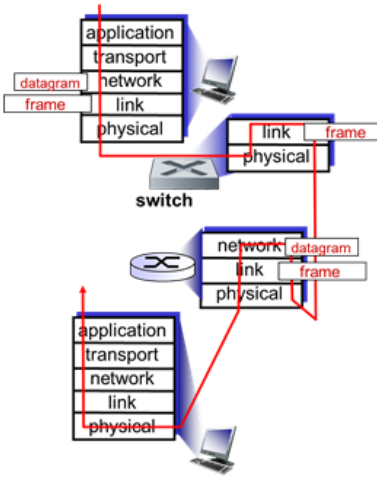
Switches can be connected together. Hook multiple switches together, connect in hierarchical fashion. S4 gives connectivity to all other switches.



Switches vs. routers

Both are store-and-forward:

- Routers: network-layer devices (examine network-layer headers)
- Switches: link-layer devices (examine link-layer headers)



Both have forwarding tables:

- Routers: compute tables using routing algorithms, IP addresses
- Switches: learn forwarding table using flooding, learning and MAC addresses

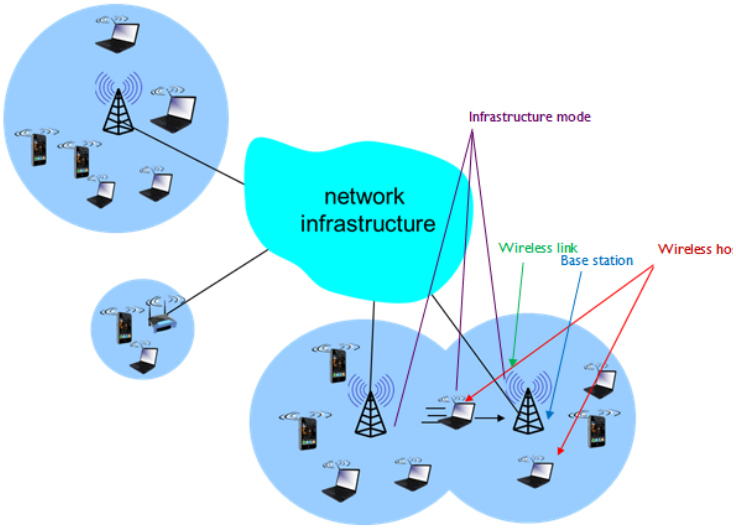
Chapter 6 – Wireless and Mobile Networks

Background:

- Number of wireless (mobile) phone subscribers and talk minutes now exceed number of wired phone subscribers 5-to-1.
- Number of wireless Internet-connected devices equals number of wireline Internet-connected devices
    - Laptops, Internet-enabled phones promise anytime untethered Internet access
- Two important but different challenges
    - Wireless: communication over wireless link
    - Mobility: handling the mobile user who changes point of attachment to network

Didrik Kvanvik

Elements of a wireless network



- Wireless hosts
  - Laptop, smartphone
  - Run applications
  - May be stationary or mobile
    - Wireless doesn't mean mobile
- Base station
  - Typically connected to wired network
  - Relay – responsible for sending packets between wired network and wireless host(s) in its 'area'
    - E.g. cell towers, 802.11 access points
- Wireless link
  - Typically used to connect mobile(s) to base station
  - Also used as backbone link
  - Multiple access protocol coordinates link access
  - Various data rates, transmission distance
- Infrastructure mode
  - Base station connects mobiles into wired network
  - Handoff: mobile changes base station providing connection into wired network
  - Activate device in ad-hoc mode and send frames back and forth.
- 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
  - Don't need internet, can send data directly to and from devices

<u>Wireless network taxonomy</u>

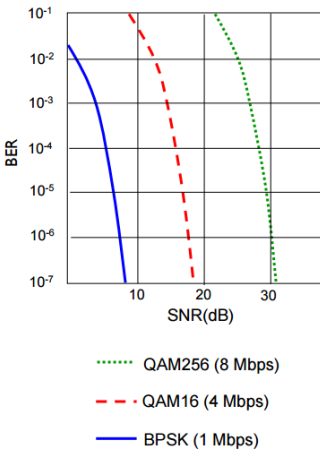
	Single hop	Multiple hops
Infrastructure	Host connects to base station which connects to larger internet (WiFi, WiMAX, cellular)	Host may have to relay through several wireless nods to connect to larger Internet mesh net
No infrastructure	No base station, no connection to larger Internet (Bluetooth, ad-hoc nets)	No base station, no connection to larger Internet. May have to relay to reach other a given wireless node MANET, VANET

6.2 Wireless links, characteristics

Link characteristics; important differences from wired link

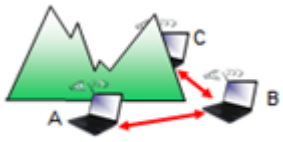
- *Decreased signal strength*: radio signal attenuates as it propagates (path loss) ~distance $^{-a}$, a = 2-4
- *Multipath propagation*: radio signal reflects off objects ground, arriving at destination at slightly different times... may vary with time
- *Interference from other sources*: standardized wireless network frequencies (e.g. 2.4 GHz) shared by other devices (e.g. phone); devices (motors) interfere also make
- *Noise*: white or colored, background or "own".

Make communication across (even a point to point) wireless link much more difficult



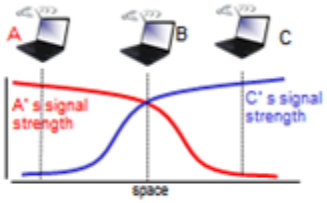
- SNR: signal-to-noise ratio
  - Larger SNR – easier to extract signal from noise
- SNR versus BER tradeoffs
  - Given physical layer: increase power → increase SNR → decrease BER
  - Given SNR: choose physical layer that meets BER req., giving highest throughput
    - SNR may change with mobility: dynamically adapt physical layer

Multiple wireless senders and receivers create additional problems, beyond multiple access:



Hidden terminal problem

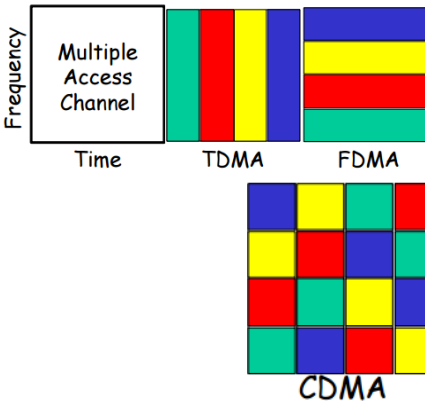
- *Hidden terminal problem*:
  - B, A hear each other
  - B, C hear each other
  - A, C cannot hear each other means A, C unaware of their interference at B
- *Signal attenuation*:
  - B, A hear each other
  - B, C hear each other
  - A, C cannot hear each other interfering at B



Signal attenuation

Code division Multiple Access (CDMA)

Unique "*code*" assigned to each user; i.e., code set partitioning (Direct sequence CDMA). All users share same frequency, but each user has own unique *"chipping" sequence* (code) to encode data. Allows multiple users to "coexist" and transmit simultaneously with minimal interference (if codes are "orthogonal"). Encoded signal = (original data) * (chipping sequence). Decoding: inner-product of encoded signal and chipping sequence.

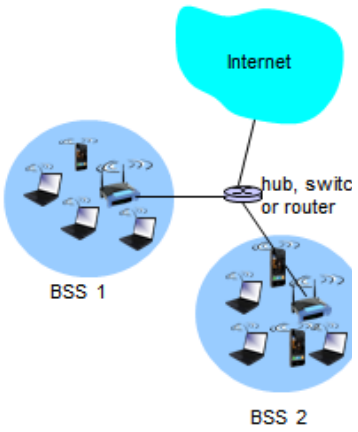


Didrik Kvanvik

6.3 IEEE 802.11 wireless LANs ("WiFi")

- 802.11a
  - 5-6 GHz range, up to 54 Mbps
- 802.11b
  - 2.4-5 GHz unlicensed spectrum, up to 11 Mbps
  - Direct sequence spread spectrums (DSSS) in physical layer
    - all hosts use same chipping code
- 802.11g
  - 2.4-5 GHz range, up to 54 Mbps
- 802.11n: multiple antennae
  - 2.4-5 GH< range, up to 200 Mbps

All use CSMA/CA for multiple access and have base-station and ad-hoc network versions.

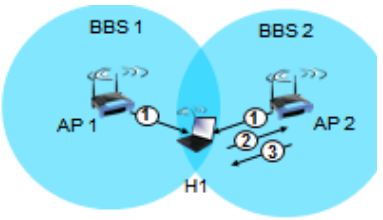


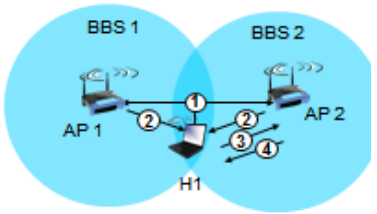
802.11 LAN architecture**:** wireless host communicates with base station which is the access point (AP). Basic service set (BSS) (aka 'cell') in infrastructure mode contains: wireless hosts and access point. Ad hoc mode: hosts only.

Channels and association

- 802.11b: 2.4 GHz-2.485 GHz spectrum divided into 11 channels at different frequencies
  - AP admin chooses frequency for AP
  - Interference possible: channel can be same as that chosen by neighboring AP!
- Host: must associate with an AP
  - Scans channels, listening for beacon frames containing AP's name (SSID) and MAC address
  - Selects AP to associate with
  - May perform authentication
  - Will typically run DHCP to get IP address in AP's subnet

Passive and active scanning



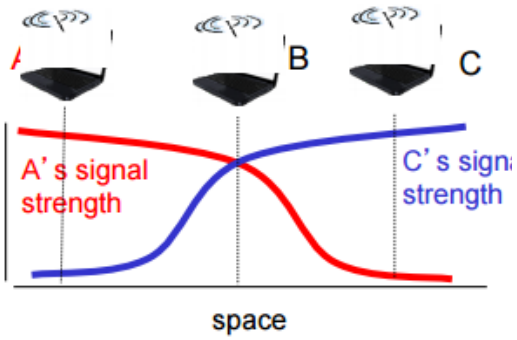


- Passive scanning:
  1. Beacon frames sent from Aps
  2. Association request frame sent: H1 to select AP
  3. Association Response frame sent from selected AP to H1
- Active scanning:
  1. Probe request frame broadcast from H1
  2. Probe response frames sent from Aps
  3. Association request frame sent: H1 to select AP
  4. Association response frame sent from selected AP to H1

Didrik Kvanvik

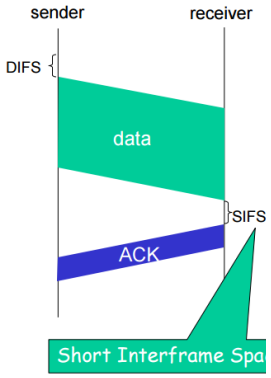
Multiple access

Avoid collision: 2+ nodes transmitting at same time.



- 802.11: CSMA – sense before transmitting
  - Don't collide with ongoing transmission by other node
- 802.11: no collision detection
  - Difficult to receive (sense collisions) when transmitting due to weak received signals (fading)
  - Goal: avoid collisions: CSMA/C(ollision)A(voidance)

IEEE 802.11 MAC protocol: CSMA/CA



802.11 sender

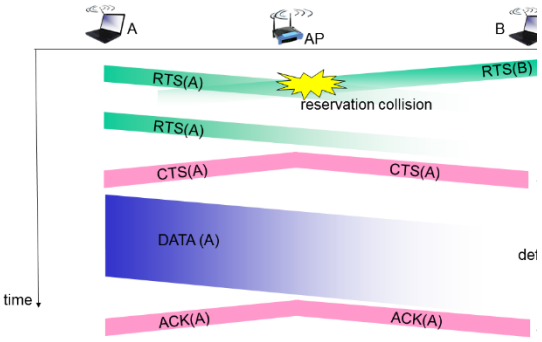
1. If sense cannel idle for Distributed Interframe Space (DIFS) then transmit entire frame
2. If sense channel busy then start random backoff time, timer counts down while channel idle, transmit when timer expires, if no ACK, increase random backoff interval and repeat 2.

802.11 receiver

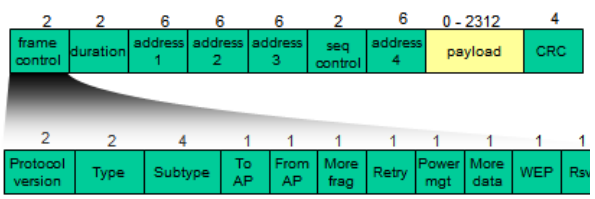
- If frame received OK, then return ACK after SIFS (ACK needed due to hidden terminal problem)

Avoiding collisions

Idea is to allow sender to "reserve" channel rather than random access of data frames: avoid collisions of long data frames. Sender first transmits *small* request-to-sent (RTS) packets to BS using CSMA, where RTSs may still collide with each other (but their short). BS broadcasts clear-to-send CTS in response to RTS. CTS heard by all nodes, sender transmits data frame and other stations defer transmissions. *Avoid data frame collisions completely by using small reservation packets!*



802.11 Frame Format: addressing

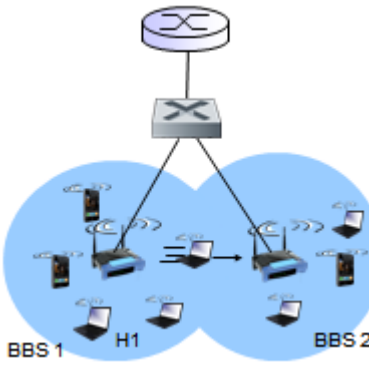


- Address 1: MAC address of wireless host or AP to receive this frame
- Address 2: MAC address of wireless host or AP transmitting this frame
- Address 3: MAC address of router interface to which AP is attached
- Address 4: used only in ad hoc mode

Variable sized frames support larger frames than Ethernet, duration of reserved transmission time (RTS/CTS).

Didrik Kvanvik

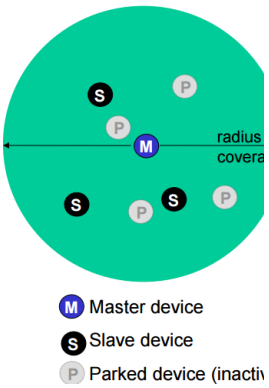
Mobility within same subnet



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

Advanced capabilities:

Rate adaptation: base station, mobile dynamically change transmission rate as mobile moves, SNR varies. Power management: node-to-AP: "I am going to sleep until next beacon frame", then 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, where node will stay awake if AP-to-mobile frames to be send; otherwise sleep again until next beacon frame.



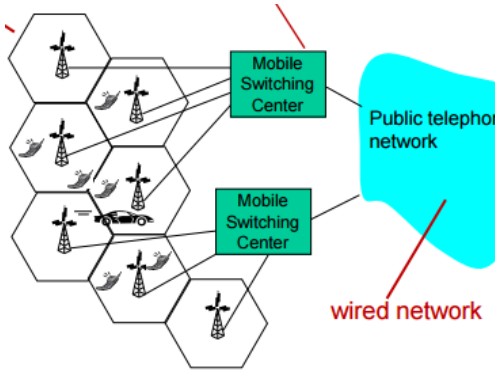
Personal area network

Less than 10 m diameter, replacement for cables, ad hoc has no infrastructure, master and slaves, 802.15 evolved from Bluetooth specification.

6.4 Cellular Internet Access

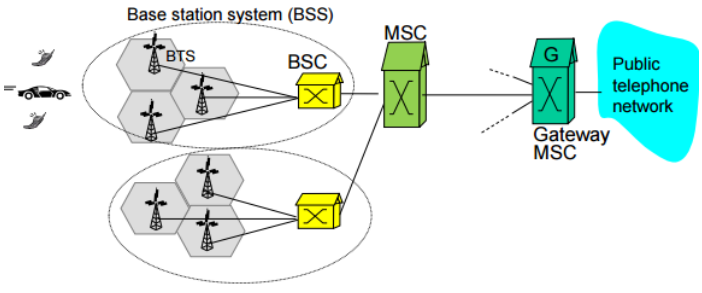
Components of cellular network architecture

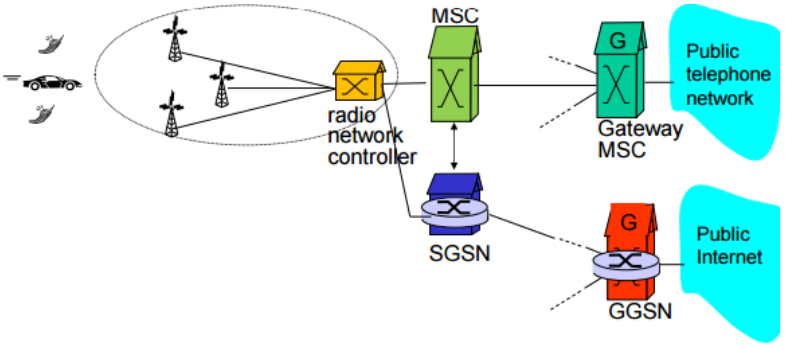
- Cell
  - Covers geographical region
  - Base Station analogous 802.11 AP
  - Mobile users attach to network through BS
  - Air-interface: physical and link layer protocol between mobile and BS
- MSC (mobile switching center)
  - Connects cells to wired tel. net.
  - Manages call setup
  - Handles mobility



Cellular networks: the first hop. There are two techniques for sharing mobile-to-BS radio spectrum. Combined FDMA/TDMA: divides spectrum in frequency channels, divide each channel into time slots. And CDMA (Code Division Multiple Access).

2G (voice, left) and 3G (voice + data, right). Key: insight: new cellular data network operates in parallel (except at edge) with existing cellular voice network: voice network unchanged in core and data network operates in parallel.





Didrik Kvanvik

Mobility

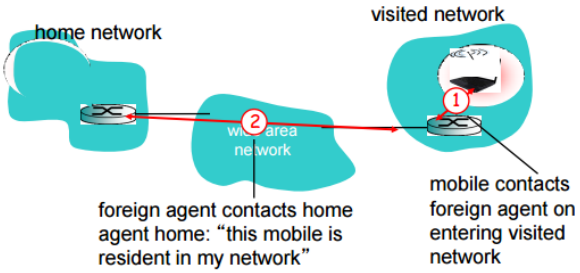
6.5 Principles: addressing and routing to mobile users

On the far left with no mobility are mobile wireless users, using same access point. In the middle is mobile user, connection/disconnecting from network using DHCP. On the far right is mobile user, passing through multiple access point while maintaining ongoing connections (like cell phone). *Home network*: permanent «home» of mobile computer. *Home agent:* entity that will perform mobility functions on behalf of mobile, when mobile is remove. *Permanent address:* address in home network, can always be used to reach home. *Permanent address:* remains constant. *Visited network*: network in which mobile currently resides. *Care-of-address:* address in visited network. *Correspondent*: wants to communicate with mobile. *Foreign agent:* entity in visited network that performs mobility functions on behalf of mobile.

Mobility approaches:

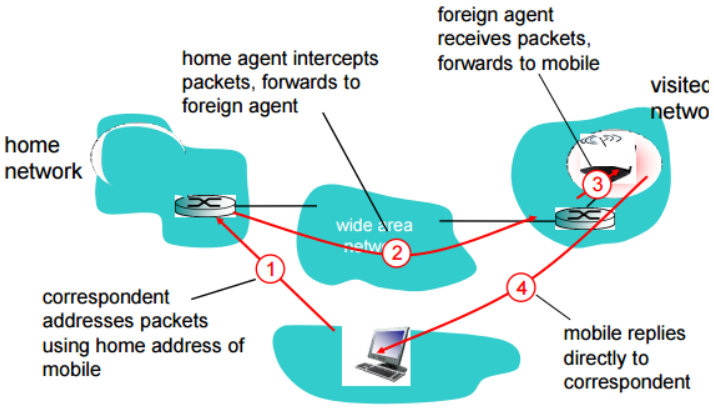
- *Let routing handle it:* routers advertise permanent address of mobile-nodes-in-residence via usual routing table exchange.
    - Routing tables indicate where each mobile located
    - No changes to end-systems
- *Let end-systems handle it:* (not scalable to millions of mobiles)
    - *Indirect routing:* communication from correspondent to mobile goes through home agent, then forwarded to remote
    - *Direct routing:* correspondent gets foreign address of mobile, sends directly to mobiles.

Registration: end result is that foreign agent knows about mobile, while home agent knows location of mobile

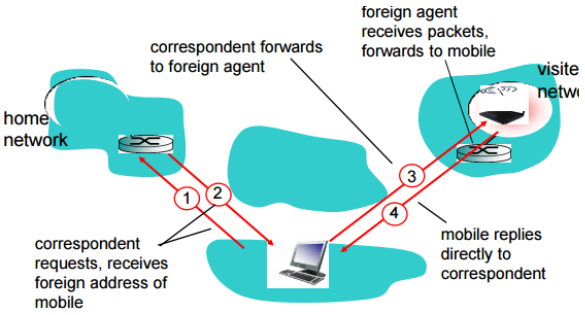


Routing via indirect routing

- Mobile uses two addresses:
    - *Permanent address:* used by correspondent (hence mobile location is *transparent* to correspondent).
    - *Care-of-address:* used by home agent to forward datagrams to mobile.
- Foreign agent functions may be done by mobile itself
- *Triangle routing:* correspondent-home-network-mobile
    - Inefficient when correspondent, mobile are in same network



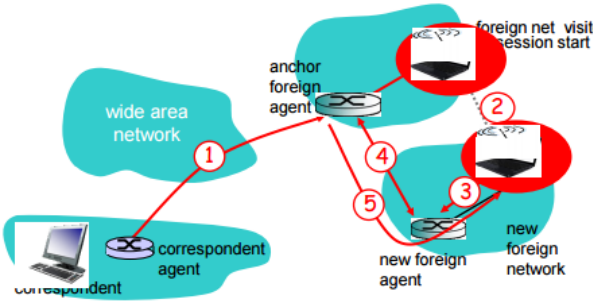
Moving between networks: suppose mobile user moves to another network; registers with new foreign agent, new foreign agent registers with home agent, home agent updates care-of-address for mobile and packets continue to be forwarded to mobile (but with new care-of-address). Mobility, changing foreign networks transparent: ongoing connections can be maintained.



Mobility via direct routing

Overcome triangle routing problem. Non-transparent to correspondent: correspondent must get care-of-address from home agent (what if mobile changes visited network?).

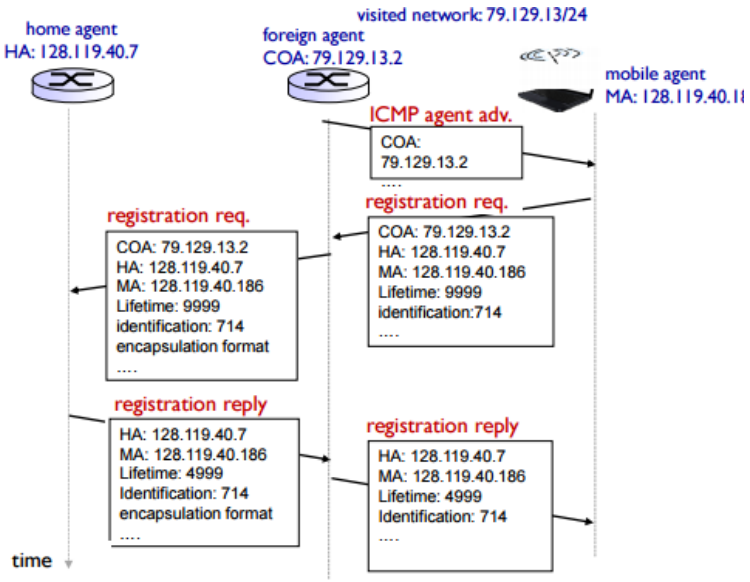
Accommodating mobility with direct routing: anchor foreign agent: FA in first visited network. Data always routed first to anchor FA. When mobile moves: new FA arranges to have data forwarded from old FA (chaining). →



6.6 Mobile IP

- RFC 3344
- Has many features we've seen:
  - Home agents, foreign agents, foreign-agent registration, care-of-addresses, encapsulation (packet-within-a-packet)
- Three components to standard:
  - Indirect routing of datagrams
  - Agent discovery
  - Registration with home agent

Indirect routing: packet sent by home agent to foreign agent: a packet within a packet. Agent discovery; agent advertisement: foreign/home agents advertise service by broadcasting ICMP messages. Registration example →



6.7 Handling mobility in cellular networks

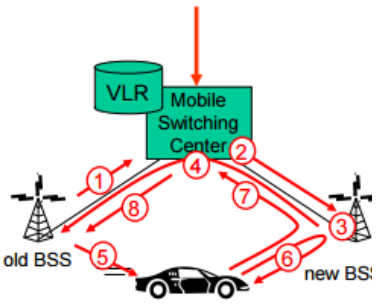
- *Home network:* network of cellular provider you subscribe to (e.g. Telenor, Netcom)
- *Visited network:* network in which mobile currently resides
  - *Visitor location register (VLR):* database with entry for each user currently in network
  - Could be home network

Global System for Mobile (GSM): handoff with common MSC. Handoff goal is to route call via new base station without interruption. Reasons for handoff are stronger signal to/from new BSS (continuing connectivity, less battery drain), load balance frees up channel in current BSS and GSM doesn't mandate why to perform handoff (policy, only how (mechanism). GSM is a standard developed by ETSI to describe protocols for 2G digital cellular networks used by mobile phones.

Didrik Kvanvik

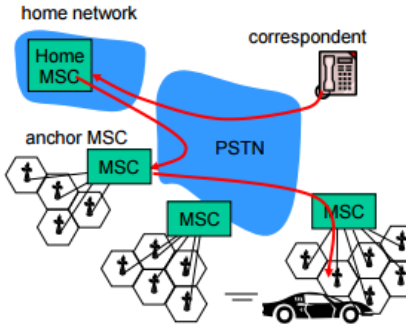
GSM: handoff with common MSC

1. Old BSS informs MSC of impending handoff, provides list of 1+ new BSSs
2. MSC sets up path (allocates resources) to new BSS
3. New BSS allocates radio channel for use by mobile)
4. New BSS signals MSC, old BSS: ready
5. Old BSS tells mobile: perform handoff to new BSS
6. Mobile, new BSS signal to activate new Channel
7. Mobile signals via new BSS to MSC: handoff complete. MSC reroutes call
8. MSC-old-BSS resources released



GSM: handoff between MSCs

- Anchor MSC: first MSC visited during call
- New MSCs add on to end of MSC chain as mobile moves to new MSC
- Optional path minimization step to shorten multi-MSC chain.



(b) after handoff

Wireless, mobility: impact on higher layer protocols

- Logically, impact should be minimal
  - Best effort service model remains unchanged
  - TCP and UDP can (and do) run over wireless, mobile
- But performance wise:
  - Packet loss/delay due to bit-errors (discarded packets, delays for link-layer retransmissions), and handoff
  - TCP interprets loss as congestion, will decrease congestion window un-necessarily
  - Delay impairments for real-time traffic
  - Limited bandwidth of wireless links

Chapter 6 summary

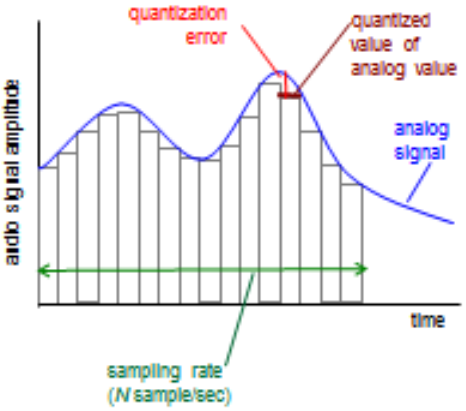
- *Wireless*
  - Wireless Links:
    - Capacity, distance
    - Channel impairments
    - CDMA
  - IEEE 802.11
    - CSMA/CA reflects wireless channel characteristics
  - Cellular access
    - Architecture
    - Standards (e.g. GSM, 3H, 4G LTE)
- *Mobility*
  - Principles: addressing, routing to mobile users
    - Home, visited networks
    - Direct, indirect routing
    - Care-of-addresses
  - Case studies
    - Mobile IP
    - Mobility in GSM
  - Impact on higher-layer protocols

Didrik Kvanvik

Chapter 7 – Multimedia Networking

7.1 multimedia networking applications

- Audio
  - Analog audio signal sampled at constant rate
    - Telephone: 8,000 samples/sec
    - CD music: 44,100 samples/sec
  - Each sample quantized, e.i. rounded
    - E.g. 2^8 = 256 possible quantized values
    - Each quantized value represented by bits, e.g. 8 bits for 256 values
  - Example: 8,000 samples/sec, 256 quantized values: 64,000 bps
  - Receiver converts bits back to analog signal:
    - Some quality reduction
  - Example rates:
    - CD: 1.411 Mbps
    - MP3: 96, 128, 160 kbps
    - Internet telephony: 5.3 kbps and up



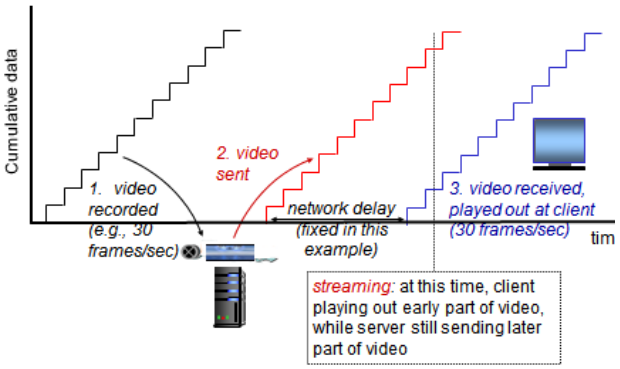
- Video
  - Digital image: array of pixels, where each pixel is represented by bits
  - Video: sequence of images displayed at constant rate, e.g. 24 images/sec
  - Coding: use redundancy within and between images to decrease number of bits used to encode image
    - Spatial (within image)
      - Instead of sending N values of same color, send only two values: color value and number of repeated values (N).
    - Temporal (from one image to next)
      - Instead of sending complete frame at i+1, send only difference from frame i.
  - Constant Bit Rate (CBR): video encoding rate fixed
  - Variable Bit Rate (VBR): constant quality to video, video encoding rate changes as amount of spatial, temporal coding changes.
  - Examples:
    - MPEG 1 (CD-ROM 1.5 Mbps
    - MPEG2 (DVD) 3-6 Mbps
    - MPEG4 (often used in Internet) < 1 Mbps

Application types:

- Streaming stored audio, video
  - Stored (at server): can transmit faster than audio/video will be rendered
  - Streaming: can begin playout before downloading entire file
  - E.g. YouTube, Netflix, Hulu
- Conversational voice/video over IP
  - Interactive nature of human-to-human conversation limits delay tolerance
  - E.g. Skype
- Streaming live audio, video
  - E.g. live sporting event

Didrik Kvanvik

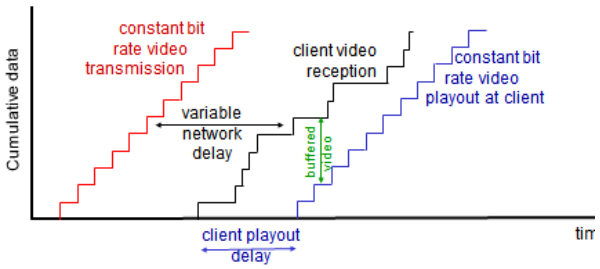
7.2 Streaming stored video



Challenges: continuous playout constraint, where once a client playout beings, playout must match original timing, but network delays are variable (jitter), so will need client-side buffer to match playout requirements. Other challenges are client interactivity such as pause, fast-forward, rewind, and jump through video and video packets may be lost and retransmitted.

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

(1) Initial fill of buffer until playout begins at t_p
(2) Playout begins at t_p
(3) Buffer fill level varies over time as fill rate x(t) varies and playout rate r is constant



Playout buffering: average fill rate (x hat), playout rate (r): x hat < r: buffer eventually empties (causing freezing of video playout until buffer again fills) x < r: buffer eventually empties (causing freezing of video playout until buffer again fills). x > r: buffer will not empty, provided initial playout delay is large enough to absorb variability in x(t). Initial playout delay tradeoff: buffer starvation less likely with larger delay, but larger delay until user begins watching. When there's a delay in the sending of frames for video over the network: Needs to be a client side buffering to have proper playback

UDP vs TCP

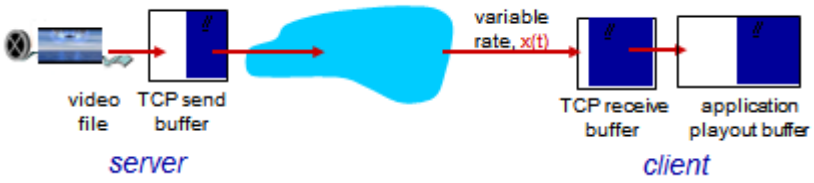
- At transport layer
- Early days: all video streaming was UDP (video quality poor)
- Recent trend is TCP based streaming
    - To get through firewalls
    - Firewall configurations tend to disable UDP
- Other protocols: RTP, RTSP, HTTP, DASH

UDP

- Server sends at rate appropriate for client
    - Often: send rate = encoding rate = constant rate
    - Transmission rate can be oblivious to congestion levels
- Short playout delay (2-5 seconds) to remove network jitter
- Error recovery: application-level, time permitting
- RTP [RFC 2326]: multimedia payload types
- UDP may not go through firewalls

Didrik Kvanvik

HTTP

- Multimedia file retrieved via HTTP GET
- Send at maximum possible rate under TCP
- 

- Fill rate fluctuates due to TCP congestion control, retransmissions (in-order delivery)
- Larger playout delay: smooth TCP delivery rate
- HTTP/TCP passes more easily through firewalls
- End-to-end control of TCP between sender and receiver buffer
- Media player – application buffer for playout

DASH

- Dynamic, Adaptive Streaming over HTTP
- Server:
    - Divides video file into multiple chunks
    - Each chunk stored, encoded at different rates
    - Manifest file: provides URLs for different chunks
- Client:
    - Periodically measures server-to-client bandwidth
    - Consulting manifest, requests one chunk at a time
    - Chooses maximum coding rate sustainable given current bandwidth
    - Can choose different coding rates at different points in time (depending on available bandwidth at time)
- Video encoded at multiple bit rates
- Dynamically choose between them based on available bandwidth
- Measure delivery rate over network periodically, dynamically choose the piece
- "Intelligence" at client: client determines:
    - When to request chunk (so that buffer starvation, or overflow does not occur)
    - What encoding rate to request (higher quality when more bandwidth available)
    - Where to request chunk (can request from URL server that is “close” to client or has high available bandwidth)

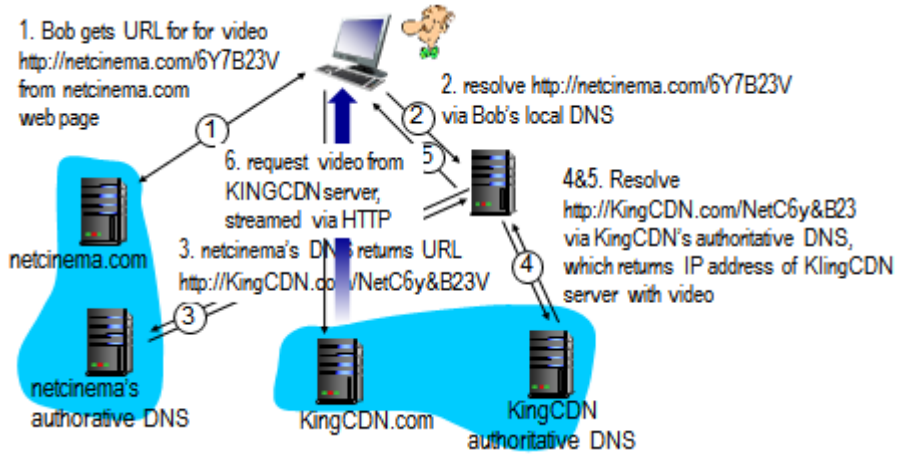
Content Distribution Networks

- Challenge: how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?
- Option 1: single, large "mega-server"
    - Single point of failure
    - Point of network congestion
    - Long path to distant clients
    - Multiple copies of video sent over outgoing link
- Quite simply: this solution doesn’t scale

- Option 2: store/serve multiple copies of videos at multiple geographically distributed sites (CDN)
    - Enter deep: push CDN servers deep into many access networks
        - Close to users
        - Used by Akamai, 1700 locations
    - Bring home: smaller number (10's) of larger clusters in POPs near (but not within) access networks
        - Used by Limelight
    - Used commercially, popular in cloud-based delivery
    - Move content closer to clients and get mirrored copies of it delivered to east/west coast/internal destinations, etc.

Example: 'Simple' Content Access Scenario

- Bob (client) requests video http://netcinema.com/6Y7B23V
    - Video stored in CDN at http://KingCDN.com/NetC6y&B23V



(3) Cinema redirects Bob

(4)Bob finds out where video is

(6) Movement of video done with KINGCDN server instead of netcinema.com

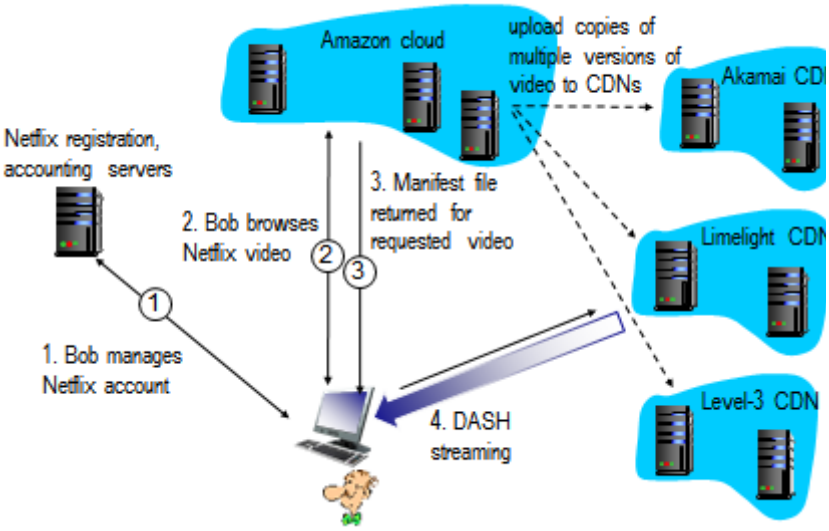
CDN Cluster Selection Strategy

- Challenge: how does CDN DNS select "good" CDN node to stream to client?
    - Pick CDN node geographically closest to client
    - Pick CDN node with shortest delay (or minimum number of hops) to client (CDN nodes periodically ping access ISPs, reporting results to CDN DNS)
    - IP anycast
- Alternative: let client decide - give client a list of several CDN servers
    - Client pings servers, picks 'best'
    - Netflix approach

Case Study: Netflix

- 30% downstream US traffic in 2011
- Owns very little infrastructure, uses 3rd party services:
    - Own registration, payment servers
    - Amazon (3rd party) cloud services:
        - Netflix uploads studio master to Amazon cloud
        - Create multiple version of movie (different endodings) in cloud

        - Upload versions from cloud to CDNs
        - Cloud hosts Netflix web pages for user browsing
    - Three 3rd party CDNs host/stream Netflix content: Akamai, Limelight, Level-3
- Infrastructure is minimal
- Accounting servers keeps money
- Everything else is farmed out
- Gets all different versions of Amazon cloud and those versions sent to CDNs and dash streaming



Se forelesnings notater for resten av kapittel 7, kapittel 8 og 9.

Didrik Kvanvik

KILDER:

Chapter 4:

http://wiki.ucalgary.ca/page/Courses/Computer_Science/CPSC_441.W2014/Chapter_4:_Network_Layer