CHAPTER 1

1.1 What is the internet

A **Network** consists of two or more computers that are linked in order to share resources, exchange files and allow electric communications. The computers on a network may be linked through cables, telephone lines, radio waves, satellites or infrared light beams.

Packets – chunks of data with a header (denotes where packet is coming from and its destination)

- small amount of computer data sent over a network. Each packet contains the address of its origin and destination and information that connects it to the related packets being sent (header)

Packet switching – the process of sending and receiving packets

Packets from many different locations can be sent on the same lines and be sorted and directed to different routes by various computers along the way.

Transmission rate – the rate in megabits per second (**Mbs**) that data can be transmitted.
Bandwidth – the amount of information/data that can be sent over a network in a given period of time

- Bandwidth also refers to the speed at which data can flow through computer and telecommunications networks without interference.

**The internet is a packet-switched network

Router – a device that determines the next network point to which a packet should be forwarded toward its destination
Switch – a device that channels (filters and forwards) incoming data from any of multiple input ports to the specific output port that will take data toward its intended destination

- Serves as a controller, enabling networked devices to talk to each other efficiently

**Switches create a network. Routers connect networks. A router links computers to the Internet, so users can share the connection. A router acts as a dispatcher, choosing the best path for information to travel so it's received quickly.

The **Internet** is a global network of billions of computers and other electronic devices.
ISP – internet service provider

Protocols – a standard (set of rules) used to define a method of exchanging data over a computer network (computers communicating with each other)

- Each protocol has its own method of how data is formatted when sent and what to do with it once received, how data is compressed or how to check for errors in data (controls the sending and receiving of packets)
- Protocols make sure that communicating devices 'speak' in same language
- Types of protocols: **HTTP** (used for accessing and receiving HTML files on internet), **SMTP** (used for transferring e-mail between devices), **FTP** (used for showing files to be copied between devices), **TCP** (ensures the delivery of information packets across networks), **IP** (responsible for logical addressing called IP address to route information between networks)

So **protocols** control the sending and receiving of information within the Internet.
Two most important protocols in the Internet:

- **Transmission Control Protocol (TCP)**
- **Internet Protocol (IP)** - specifies the format of the packets that are sent and received among routers and end systems.

It's important that everyone agree on what each and every protocol does, so that people can create systems and products that interoperate – that's why we need **internet standards**

An **Internet standard** (STD) is a specification that has been approved by the Internet Engineering Task Force (IETF). Such standard helps to promote a consistent and universal use of the Internet worldwide. A network **socket** is one endpoint in a communication flow between two programs running over a network. Sockets are created and used with a set of programming requests or "function calls" sometimes called the sockets application interface (**Sockets API)**

- API for the internet, program interface that makes the internet a black box (postal service analogy – provides with many reliable services)
- Provides server with commands such as listen, stream, etc.…

*** The IETF standards documents are called **requests for comments** (**RFCs**). RFCs started out as general requests for comments (hence the name) to resolve network and protocol design problems that faced the precursor to the Internet. RFCs tend to be quite technical and detailed. They define protocols such as TCP, IP, HTTP (for the Web), and SMTP (for e-mail).

Network infrastructure refers to the hardware and software resources of an entire network that enable network connectivity, communication, operations and management of an enterprise network. Network infrastructure provides the communication path and services between users, processes, applications, services and external networks/the Internet.

1.2 Network Edge

Network edge – applications and hosts (clients and servers) – device which provides an entry point into service provider core networks

- end devices connected to internet through access networks (provides access from one network to rest of internet)

Network core – central part of a telecommunications network that provides various services to customers who are connected by the access network

- interconnected routers, network of networks

An **Access network** is a user network that connects subscribers to a particular service provider and, through the carrier network (proprietary network infrastructure belonging to a telecommunications service provider), to other networks such as the internet.

- **DSL**, cable, home, Ethernet, **LAN**

Some types of access networks:

- **Ethernet** is the most commonly installed wired LAN (local area network) technology. Ethernet LAN typically uses coaxial cable or special grades of twisted pair wires.
- **Wireless LANs** allow mobile users to connect through a wireless (radio) connection.
- **Fibre optic** networks such as fiber to the home (FTTH) use optical fiber from a central point directly to individual buildings such as residences, apartment buildings and businesses.
- **ADSL** (Asymmetric Digital Subscriber Line) is a technology for transmitting digital information at a high bandwidth on existing phone lines to homes and businesses.

ACCESS NETWORKS

Digital Subscriber Line (DSL) – reuses telephone infrastructure

- Modem takes digital signal and transform to analog
- Voice and data sent on different signals

Local Area Network (LAN)

- Computers (end systems) connected to one another through a switch and then this switch connected to a router, which helps connect network to internet

Cable Network

- Uses of an existing cable television infrastructure
- Hybrid fiber coax – network of cable, fiber attaches to home ISP router

Hosts send packets of data that are of length **L bits and transmit packet into access network at transmission rate (capacity/link bandwidth) **R**

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

1.3 Network Core

1 - Packet Switching

The resources needed are not reserved, they are used on demand and therefore may have to wait (queue) to access a communication link.

Source ---------- Router 1 ---------- Destination
(link1) (link2)

When wanting to send file, the file is segmented and each segment has headers.
Transmit packet to link.
** So breaking data into packets and sending that data.

Transmission delay – amount of data represented in bits that we can transmit/push into the link in a fixed amount of time (bps)

Size of packet is L bits.
Transmission rate of R bits/second.
Time it takes for packet 1 to reach packet switch = L/R secs -> transmission delay
It takes 2L/R for packet 1 to reach destination.

Store-and-forward transmission -> we wait until we get every bit of packet before we start transmitting out. Only after the router has received all of the packet's bit can it begin transmitting onto the outbound link.
(Serial transmission)

Q: How long does it take for 3 packets to reach destination?

It would take 4L/R

Q: Assume we have N links (N-1 routers then), every link has a transmission rate of R-bits/second. How long will it take for one packets of size L bits to be transmitted end to end?

= NL/R

Q: How about for P packets?

= (N-1+P * L) / R

By PL/R, all packets have left the sending host (last one is in first link)

We have N-1 remaining links, so it will take (N-1)L/R to get that last packet to destination.

So PL/R + (N-1)L/R = L(N-1+P) / R

What if we have two sending hosts and they both start transmitting a packet at same time. Both will get to packet switch, but packet switch can only transmit one at a time.

For every outgoing link we have a **queue (output buffer / output queue)**. Packet 1 will be transmitted while packet 2 waits in the queue (queueing delay).

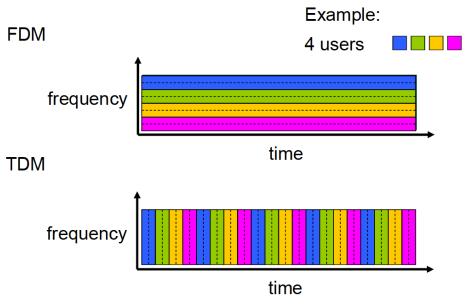
Queueing delay depends on traffic, **arrival rate** of packet and size of packets.

If the queue can only hold 4 packets and 5 packets arrive, it will drop one, which causes **packet loss**.

- **Routing:** determines source-destination route taken by packets
- **Forwarding:** moves packets from router's input to appropriate router output

2 – Circuit Switching

In circuit-switched networks, the resources needed along a path (buffers, link transmission rate) to provide for communication between the end systems are **reserved for the duration of the communication** session between the end systems.



1. **Frequency Division Multiplexing** (**FDM**)
   Given the links frequency spectrum, it will divide into different channels each given its own frequency.
   The link is divided into multiple channel, and each channel will take a fraction of the links capacity. Every channel is dedicated for a designated connection (can't be switched or changed).

2. **Timed Division Multiplexing** (**TDM**)
   The usage of a given link is shared – divides the access to link by time.
   Every link has frames and every frame has slots.

Say if we have 4 connections, we would get 4 slots in each frame. Every connection within that frame will have
If we assume every frame is 1 second, and every frame has 4 slots. One slot is ¼ of a second.
Transmission rate pf the link = 1mps = 1 sec -> 1 mb (mega bit)
¼ sec -> 250 kb

Every ¼ sec, connection can transmit 250kb.

Packet Switching	**Circuit Switching**
+ Resource sharing - No guarantee on end-to-end delay (b/c of queuing delay that depends on traffic) + Can handle bursty (intervals of activity followed by periods of inactivity) data	+ Guaranteed end-to-end delay - Does not scale well (limited to number of people – we always have to reserve those slots even if not active) - Wasteful in idle mode (when packets are not being transmitted but connection is established)

Comparison

The link = 1mbps / link
The probability of an active user (P(active)) is 10% of time and probability of idle is (P(idle)) 90% of time.
But when active, desired transmission rate = 100kps for every connection

Q: How many connections can we support in a circuit-switched networks?
1Mbps/ 100kbs = 10 users

Q:
Pr(N>= 11) = 1 – Pr(N<= 10) = 1 - 0.9996 = 0.0004
Probability of having more than 10 users at the same time.

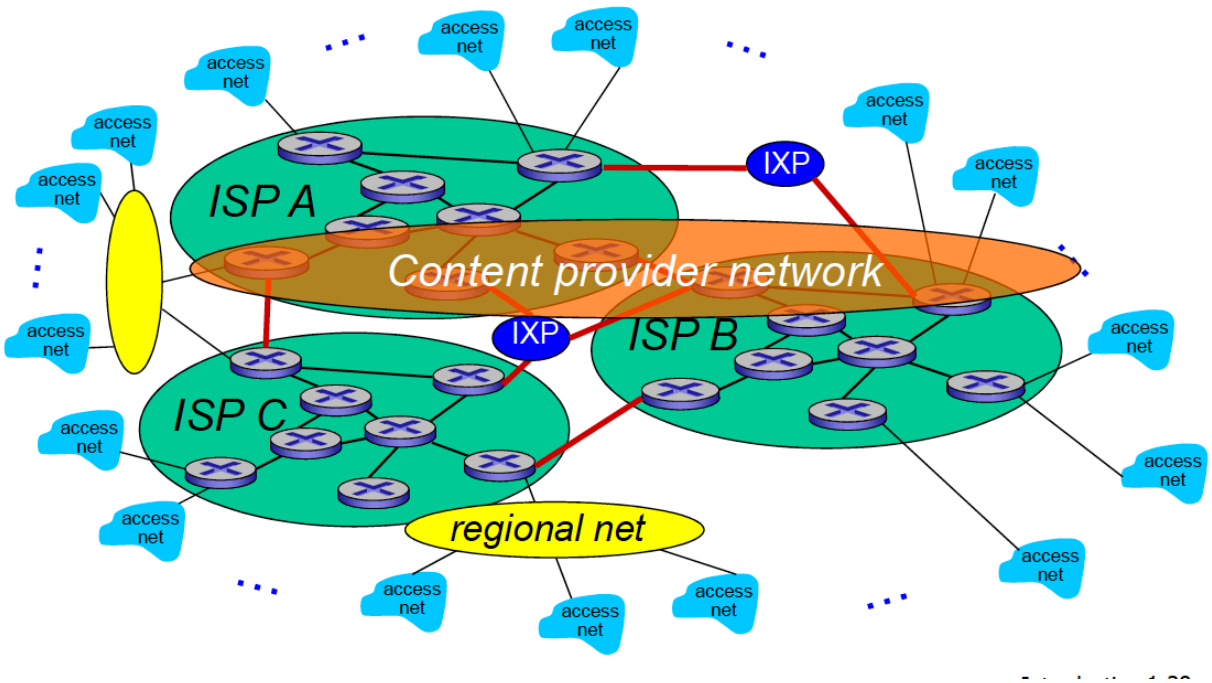
Q: If there are 10 users and 1 user wants to transmit 10^6 bits and the other 9 are idle, how long will it take to transmit this packet using circuit switching?
L/R = 10^6 / 100 x 10^3 = 10 sec

Q: How about packet switching?
10^6 / 10^6 = 1 sec

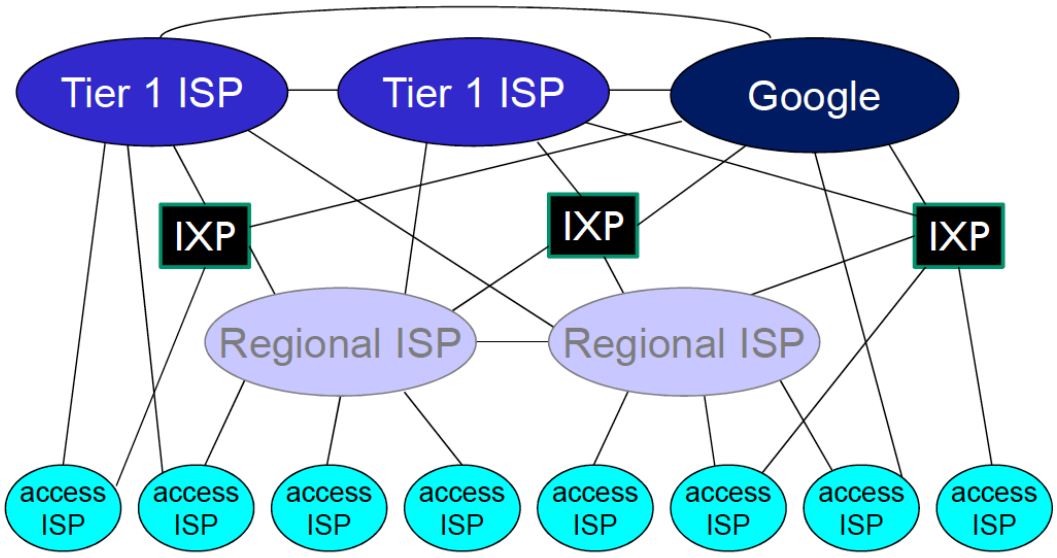
Internet Structure -> Networks of Networks
End systems connect to the Internet via an access ISP (which can provide wired or wireless connectivity).

- An ISP doesn't have to be a telco/cable company; it can simply be a university giving access to students

Access ISPs in turn must be interconnected so that any two hosts can send packets to each other.
peering link – one ISP allows another to send traffic settlement free (for no money)



Global ISP (green)
IXP (Internet exchange points) – hub for multiple ISP's
Peering link – red link between routers
There are regional networks to connect access networks to ISPs



1.4 Delay, loss, throughout in networks

DELAYS:
Total nodal delay = processing delay+ queuing delay + transmission delay + propagation delay

** A packet can be transmitted on a link only if there is no other packet currently being transmitted on the link and if there are no other packets preceding it in the queue

1. **Processing delay** – the time required to examine the packet's header and determine where to direct the packet (also time needed to check bit-level errors)
    - After this nodal processing, the router directs the packet to the queue that precedes the link to the next destination
2. **Queueing delay** – time the packet waits in the queue to be transmitted onto the ink
    - Unlike other delays, this can vary from packet to packet
3. **Transmission delay** – the amount of time required to push (transmit) all of the packet's bits into the link
    - Transmission delay = L (length of packet) / R (transmission rate of link from A to B)
4. **Propagation delay** – the time required to propagate from the beginning of the link to the end (from one router to the next – function of distance between routers)
    - The bit propagates at the propagation speed of the link, this speed depends on the physical medium of the link

Traffic intensity = La / R

- The average rate at which packets arrive at the queue
- If La/R > 1, the average rate at which bits arrive at the queue exceeds the rate at which the bits can be transmitted from the queue -> queue will tend to increase without bound and the queueing delay will approach infinity

Queueing capacity is finite.
So if a packet arrives to a full queue, the router will drop that packet (**packet loss**), because there is no place to store it.

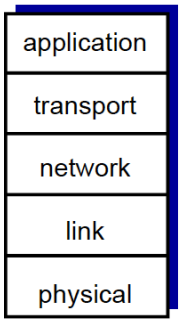
throughput – the amount of data per second that can be transferred

- Instantaneous throughput – at any instant of time is the rate (bits/sec) at which Host B is receiving the file
- Average throughput – F (number of bits in file) / T (time in seconds transfer takes)
    - Rate over longer period of time

1.5 Protocol Layers, Service Models

Internet Protocol Stack:

- *application:* supporting network applications
  - FTP, SMTP, HTTP
- *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, 802.111 (WiFi), PPP
- *physical:* bits "on the wire"



Protocols and the network hardware and software that implement them is organized in **layers.**

- Each protocol belongs to a layer

We are interested in the services that a layer offers to the layer above, called the **service model**.

- Each layer provides its service by (1) performing certain actions within the layer and by (2) using the services of the layer directly below it

When taken together, the protocols of the various layers are called the **protocol stack**.

- The Internet protocol stacks consists of 5 layers: physical, link, network, transport, application

1. **Application Layer**
   This is where the network applications and their protocols reside
   The Internet's application layer includes many protocols such as HTTP (provides web document request and transfer), SMTP (provides transfer for e-mails), FTP (provides transfer of files between end systems) and also the domain name system (DNS)
   The application in one end system uses the protocol to exchange packets of information with the application in another end system
   (packed -> **message**)

2. **Transport Layer**
   This layer transports application-layer messages between application endpoint.
   There are 2 transport protocols, TCP and UDP.
   (packets -> **segment**)

3. **Network Layer**
   This layer is responsible for moving network-layer packets from one host to another.
   The transport-layer protocol in a source host passes a segment and a destination address to the network layer (just like giving postal service a letter with destination address).
   The network layer provides the service of *delivering* the segment to the transport-layer protocol in the destination host.
   This layer includes the IP protocol (defines fields in datagram and how end systems and routers act on these fields).
   (packet -> **datagram**)

4. **Link Layer**
   The network layer routes a datagram through a series of routers between the source and destination. To move a packet from one node (host / router) to the next node in the route, the network layer relies on the services of the link layer.
   At each node, the network layer passes the datagram down to the link layer, which delivers the datagram to the next node along the route, and at this next node, the link layer passes the datagram up to the network layer.
   The services provided depend in the link-layer protocol (ex: reliable delivery).
   (packet -> **frames**)

5. **Physical Layer**
   While the job of the link layer is to move entire frames from one network element to an adjacent network element, the job of the physical layer is to move *individual bits* within the frame from one node to the next.
   (Ex: bits moved differently in twisted-pair copper wire than coaxial cable and fiber, etc…)

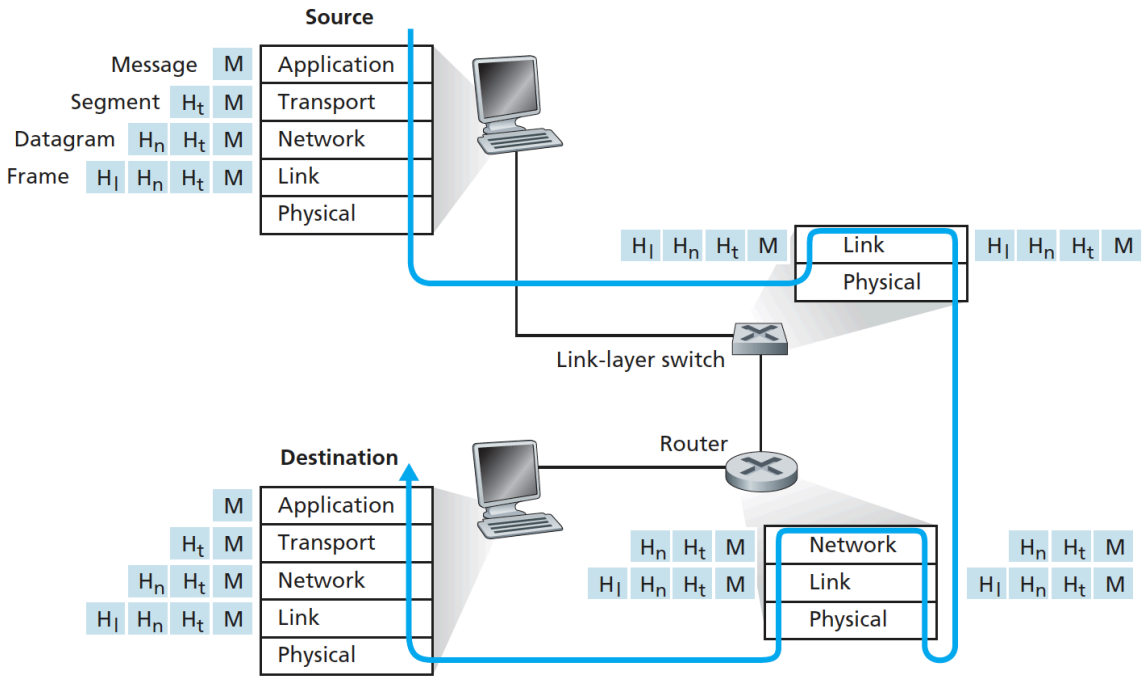


Figure 1.24 ♦ Hosts, routers, and link-layer switches; each contains a different set of layers, reflecting their differences in functionality

1.6 Security

CHAPTER 2 – APPLICATION LAYER

2.1 Principles of network applications

Network architecture: five-layer Internet Architecture (Application, Transport, Network, Link, Physical).

The **application architecture** is designed by the application developer and dictates how the application is structured over the various end systems.

*Programs that run on different end systems and communicate with each other

Two architecture paradigms:

1. **Client-server Architecture**
   - *There is an **always-on host**, called the **server**, which services requests from many other hosts, called **clients***
   - When a Web server receives a request for an object from a client host, it responds by sending the requested object to the client host.
   - Clients do not directly communicate with each other; for example, in the Web application, two browsers do not directly communicate.
   - The server has a fixed, well-known address, called IP address.
   - Because the server has a fixed, well-known address, the because it's always on, a client can always contact the server by sending a packet to the server's IP address.

   - Example: The Web application for which an always-on Web server services requests from browsers running on client hosts.

   - Sometimes a single-server host is capable of keeping up with all the requests from clients but other times it can be too overwhelming, so a **data center** – housing a large number of hosts – is used to create a powerful virtual server.

2. **Peer-to-Peer**
   - In a P2P there is minimal (or no) reliance on dedicated servers in data centers.
   - The application **exploits direct communication between pairs of intermittently connected hosts**, called peers (end systems directly communicate, ex: laptops and desktops).
   - The peers are not owned by the service provider

   - Example: Skype
   - 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

Process Communicating

Process – program running within a host

- Client process: process that initiates communication
- Server process: process that waits to be contacted

- Within same host, two processes communicate using **inter-process communication**
- Processes in different hosts communicate by exchanging **messages**

- A network application consists of pairs of processes that send messages to each other over a network.
    - In the Web, a browser process initializes contact with a Web server process; hence the browser process is the client and the Web server process is the server. In P2P file sharing, when Peer A asks Peer B to send a specific file, Peer A is the client and Peer B is the server in the context of this specific communication session.

- Processes (house) send/receive messages to/from their **socket** (sort of like a door)
    - The **socket** is the interface between the application layer and the transport layer within a host
        - The **transport-layer** protocol has the responsibility of getting the messages from the socket of sending process to the socket of the receiving process
        - It's also referred as the **Application Programming Interface** (**API**) between the application and network, since the socket is the programming interface with which network applications are built.
    - The application developer has control of everything on the application-layer side of the socket but has little control of the transport-layer side of the socket.
- Processes must have identifiers: **IP addresses** (32 bit) and **port numbers**

The services that a **Transport-layer protocol** could (not always though) guarantee:

- **Data integrity** (process-to-process reliable data transfer – meaning no data loss)
- **Timing** (low delay, taking no longer than x to arrive)
- **Throughput** (minimum amount of throughput – rate at which the sending process can deliver bits to receiving process, ex: x bits/sec)
    - Bandwidth-sensitive applications vs elastic applications
- **Security** (encryption, data integrity)

Two **Internet Transfer Protocols:**

TCP Service	**UDP 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 guarantees, security - **connection-oriented**: setup required between client and server process	**- unreliable data transfer** between sending and receiving process - **does not provide:** reliability, flow control, congestion control, timing, throughput guarantee, security or connection set up

**Here we notice neither provides encryption.
Secure Sockets Layer (SSL) is an enhancement for TCP that provides encryption, data integrity and end-point authentication.

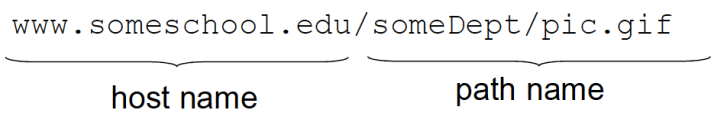
Application-Layer Protocols defines how an application's processes, running on different end systems, pass messages to each other.

- **Type of messages** exchanged (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

2.2 Web and HTTP

- web page consists of objects
- object can be HTML file, JPEG image, Java applet, audio file, …
- web page consists of base HTML-file which includes several referenced objects each object is addressable by a URL, e.g.,



HTTP: hypertext transfer protocol

- web's application layer protocol
- client-server model
  - **client**: browser that request, receives and displays Web objects
  - **server**: web server sends objects in response to requests
- uses TCP:
  1. client initiates TCP connection (creates socket) to server, port 80
  2. server accepts TCP connection from client
  3. HTTP messages exchanged between browser and web server
  4. TCP connection closed
- HTTP is "**stateless**" (not complex) – the server maintains no information about past clients
- Connections:
  - **Non-persistent HTTP**
    - At most one object sent over TCP connection (connection then closed)
    - Downloading multiple objects requires multiple connections
    - Response time = 1 RTT to initiate TCP connection + 1 RTT for HTTP request and first few bytes of HTTP response to return + file transmission time
      - 2RTTs per object
    - for each object:
      - initiation of connection, request message, receive and send response, close connection, receive response
  - **Persistent HTTP**
    - Multiple objects can be sent over single TCP connection between client & server
    - Server leaves connection open after sending response
    - Subsequent HTTP messages between same client/server sent over open connection
    - Response time: < 1 RTT
- Request messages:
  - **Request** (GET, POST, HEAD, PUT, DELETE) – request line (method, url), header line
  - **Response** – status code, header lines, entity body (data)

Round Trip Time (RTT): time for a small packet to travel from client to server and back

User-server state: **Cookies** - allow sites to keep track of users

- Four component:

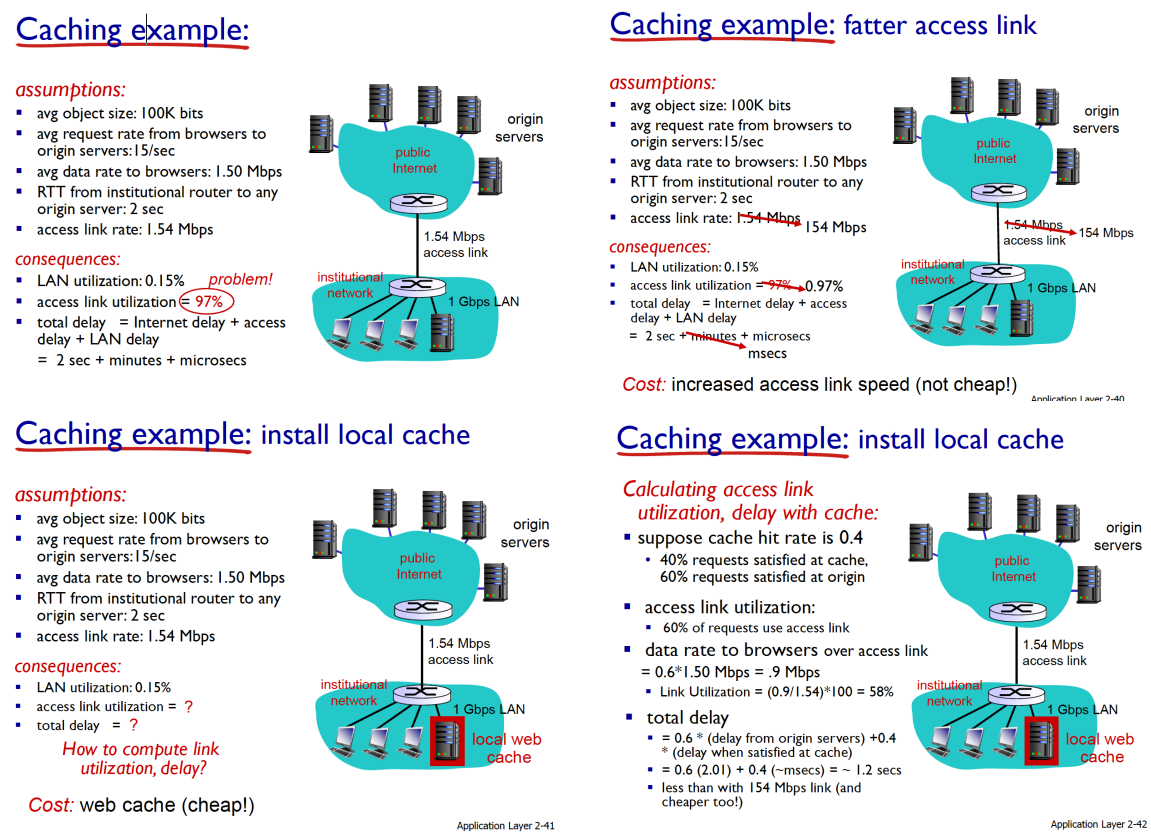
- Cookie header line of HTTP response message `(set-cookie: 1678)`
- Cookie header line in next HTTP request message `(cookie: 1678)`
- Cookie file kept on user's end system and managed by user's browser
- Back-end database at web site (using unique ID)

Cookies can therefore be used to create a user session layer on top of stateless HTTP

Web caching

A **web cache** (also called **proxy server**) is a network entity that satisfies HTTP requests based on behalf of an origin web server. The web cache, it is both a server and client, has its own disk storage and keeps copies of recently requested objects in this storage.

- Can reduce response time for client request
- Reduce traffic on an institution's access link to internet



conditional GET – cache checks that it has the correct version in cache before sending response to client

2.3 Electronic mail

Three major components of e-mail:

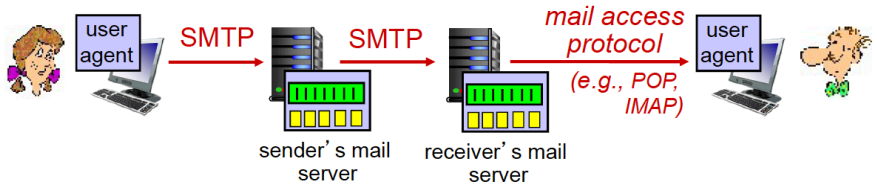
- **User agents** – "mail reader" – composing/editing/reading mail messages – Outlook/Apple Mail
- **Mail servers** – mailbox containing incoming messages, message queue of outgoing messages
- **Simple mail transfer protocol** (SMTP) – SMTP application-layer protocol between mail servers (sender's mail server and client's mail server) to send email messages
  - Uses TCP
  - SMTP server receives and SMTP client sends
  - Three phases of transfer: handshaking, transfer of messages, closure

- Command/response interaction
- Persistent connection

Scenario: Alice sends message to Bob

1) Alice uses UA to compose message "to" `bob@someschool.edu`
2) Alice's UA sends message to her mail server; message placed in message queue
3) client side of SMTP opens TCP connection with Bob's mail server
4) SMTP client sends Alice's message over the TCP connection
5) Bob's mail server places the message in Bob's mailbox
6) Bob invokes his user agent to read message

HTTP is a **pull protocol** – someone loads information on a web server and the users use HTTP to pull the information from the server at their convenience.
SMTP is a **push protocol** – the sending mail server pushes the file to the receiving mail serve.



In order to retrieve mail from (receiver's mail) server, we use **Mail Access Protocols**:

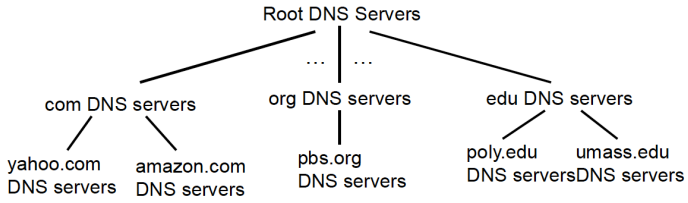
- **POP** (**Post Office Protocol**)
  - has 3 phases: authorization, transaction and update
  - "download and delete" vs "download and keep"
  - stateless across platforms
- **IMAP** (**Internet Mail Access Protocol**)
  - Keeps all messages in one place, server
  - Allows users to organize messages in folders
  - Keeps user state across sessions
- HTTP

2.4 DNS: Domain name system

Domain Name System
The main task is to translate hostnames to IP addresses.

- Distributed database – implemented in hierarchy of many name servers
- Application-layer protocol – hosts, name servers communicate to resolve names

DNS: a distributed, hierarchical database



client wants IP for www.amazon.com; 1st approximation:

- client queries root server to find com DNS server
- client queries .com DNS server to get amazon.com DNS server
- client queries amazon.com DNS server to get IP address for www.amazon.com

1. The same user machine runs the client side of the DNS application.
2. The browser extracts the hostname, www.someschool.edu, from the URL and passes the hostname to the client side of the DNS application.
3. The DNS client sends a query containing the hostname to a DNS server.
4. The DNS client eventually receives a reply, which includes the IP address for the hostname.
5. Once the browser receives the IP address from DNS, it can initiate a TCP connection to the HTTP server process located at port 80 at that IP address.

3 classes of DNS servers

1. **Root DNS servers**
   - Contacted by local name server that cannot resolve name
2. **Top-Level Domain (TLD) servers**
   - Responsible for com, org, edu, aero, jobs, museums and all top-level country domains, eg: uk, fr, ca, jp
   - Network solutions maintains severs from .com TLD
3. **Authoritative DNS Servers**
   - Organization's own DNS server(s), providing authoritative hostname to IP mappings for organization's named hosts
   - Can be maintained by organization or service provider

Local DNS Name Server

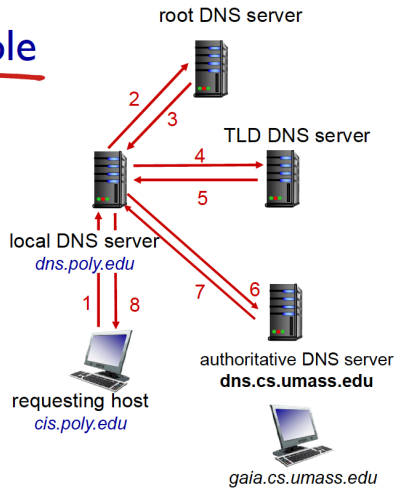
- Does not strictly belong to hierarchy
- Each ISP (university, company, residential ISP) has one
  - When a host connects to an ISP, the ISP provides the host with the IP address of one or more of its local DNS servers
- When host makes DNS query, query is sent to its local DNS server

DNS name resolution example

- host at cis.poly.edu wants IP address for gaia.cs.umass.edu

iterated query:

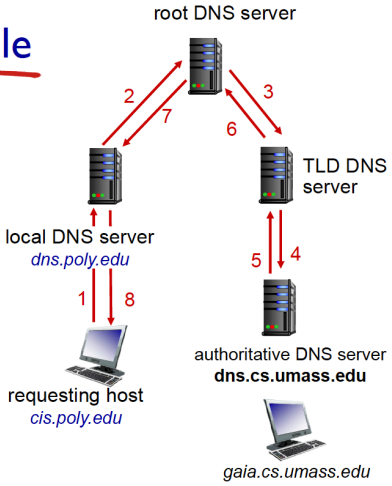
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



DNS name resolution example

recursive query:

- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy?



The DNS servers that together implement the DNS distributed database store **resource records (RRs)**, including RRs that provide hostname-to-IP address mapping.

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

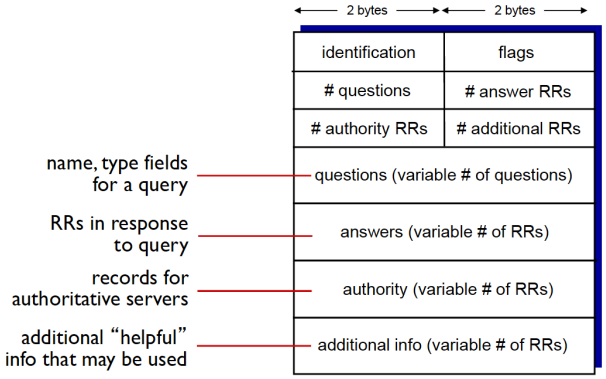
There are 4 RR types:

1. **Type=A** (**name** is hostname; **value** is IP address)
2. **Type=NS** (**name** is domain; **value** is hostname of authoritative name server for this domain)
3. **Type=CNAME** (**name** is alias name for canonical/real name; **value** is a canonical name)
4. **Type=MX** (**value** is name of mail server associated with name)

TTL is the time to live of the resource record; it determines when a resource should be moved from a cache.

There are only two kinds of DNS messages:

1. Reply messages
2. Query messages



The header contains:

- **Identification**: 16 bit # for query, reply to query uses same #
- **Flags**: query vs reply, recursion desired, recursion available, reply is authoritative

Inserting records into the DNS database (networkutopia.com for Network Utopia)

1. Register the domain name networkutopia.com at **registrar** (commercial entity that verifies the uniqueness of the domain name, enters the domain name into the DNS database and collects small fee from you for its services – GoDaddy)
2. When registering domain name networkutopia.com with some registrar, you need to provide the registrar with names and IP addresses of your primary and secondary authoritative DNS servers (dns1.networkutopia.com, dns2.networkutopia.com, 212.212.212.1 and 212.212.212.2)

DDoS bandwidth-flooding attack

- an attacker could attempt to send to each DNS root server a deluge of packets, so many that the majority of DNS queries never get answered
- DNS servers can be protected by packet filters

Man-in-the-Middle attack

- The attacker intercepts queries from hosts and returns bogus replies

DNS Poisoning attack

- The attacker sends bogus replies to a DNS server, tricking the server into accepting bogus records into its cache

The last 2 can be used to redirect an unsuspecting web user to the attacker's web site – hard to do

2.5 P2P applications

*distributing a large file from a single server to a large number of hosts

- no always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses

peers and server are connected to the internet via **access links**.

The **distribution time** is the time it takes to get a copy of the file to all N peers.

Distribution time for **client-server architecture**:

$$D_{cs} \geq \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$$

N peers, **F** is file size in bits, $\mathbf{u_s}$ is server's upload rate and $\mathbf{d_{min}}$ is the lowest peer download rate

Distribution for the **p2p architecture**:

$$D_{\text{P2P}} \geq \max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\}$$ (server to one peer, peer with lowest download rate, system must deliver F bits to N peers with total sum of upload rates)

F/u is the time it takes a peer to transmit a file.
➔ P2P is much faster than client-server!

How BitTorrent works:

- Files are divided into 256Kb chunks
- Peers in **torrent** (group of peers exchanging chunks of a file) send and receive file chunks (through TCP connections)
- A **tracker** tracks peers participating in torrent
- While peers download chunks, they also upload chunks
- Peer may change peers with whom it exchanges chunks
- **Churn**: peers may come and go
- Once a peer has entire file, it may selfishly leave or altruistically remain in torrent
- **Requesting chunks:**
    - At any given time, different peers have different subsets of file chunks
    - Periodically, a user asks each peer for a list of chunks that they have
    - The user requests missing chunks from peers, **rarest first**
- Sending chunks:
    - The user gives priority (to send) to neighbours that are currently supplying the user with data at the highest rate (the rate is periodically being calculated)
    - Peers that do not receive from the user are **choked** by that user
    - Every 30 seconds, the user selects one additional neighbour at random and sends it chunks – that random neighbour is said to be **optimistically unchoked**
    - **tit-for-tat:**
        - Tit-for-tat means that if user 1 optimistically unchokes user 2, and user 1 becomes one of user 2's top providers, user 2 will reciprocate and become one of 1's top providers

2.6 Video streaming and content distribution networks

Video traffic is the major consumer of internet bandwidth
Video: sequence of images displayed at a constant rate (ex: 24 images/secs)

- an uncompressed, digital encoded image consists of an array of pixels, with each pixel encoded into an array number of bits to represent luminance and colour
- CBR (constant bit rate): video encoding rate fixed
- VBR (variable bit rate): video encoding changes

Streaming multimedia: **DASH** (**Dynamic, Adaptive, Streaming over HTTP**)

- **Server**: divides video file into multiple chunks (each stored and encoded at different rates) – there is a **manifest file** is the file that provides URL for different chunks
- **Client**: periodically measures server-to-client bandwidth and while consulting manifest, requests one chunk at a time
    - Client determines <u>when</u> to request chunk (so that buffer starvation or overflow doesn't occur)
    - Client determines <u>what encoding rate</u> to request (higher quality when more bandwidth available)

- Client determines where to request chunk (can request from URL server that is "close" to client or has high available bandwidth)

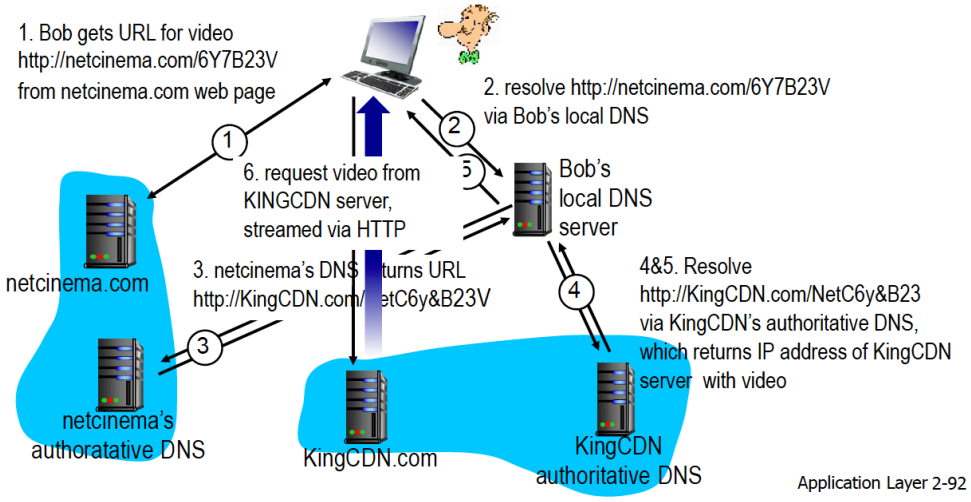
Content Distribution Networks

Challenge: how to stream content to hundreds of thousands of simultaneous users?

Solution: we can store multiple copies of videos at multiple geographically distributed sites (CDN)

- **Enter deep**: push CDN servers deep into many access networks
- **Bring home**: smaller number of larger clusters in POPs near access networks

CDN stores copies of content at CDN nodes (ex: Netflix stores copies of *Mad Men*)
Subscribers then request content from CDN



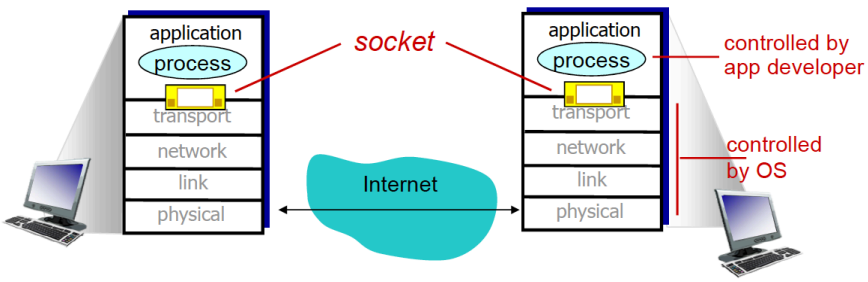
2.7 Socket programming with UDP and TCP

So a typical network application consists of a pair of programs – a client program and a server program – residing in two different end systems.
When these two programs are executed, a client process and a server process are created and these processes communicate with each other by reading from, and writing to, **sockets**.
When creating a network application, the developer's main task is therefore to write the code for both the client and server programs.

During the development phase, one of the first decisions the developer must make is whether the application is to run over TCP or over UDP.

A protocol is a particular set of rules for having a conversation between two computers to convey a specific set of information. A standard (and in the networking arena, many protocols are standards) is a document that specifies something that has the overwhelming support and agreement of the standards making body.

socket: door between application process and end-to-end-transport protocol



socket programming

Two socket types for two transport services:

- **UDP**: unreliable datagram
- **TCP**: reliable, byte stream-oriented

UDP: no "connection" between client and server	**TCP**:
- No handshaking before sending data - Sender explicitly attaches **IP destination address** and **port #** to each packet - Receiver extracts sender IP address and port # from received packet - UDP transmitted data may be lost or received out-of-order	- Client must contact server (server must first be running and have created a welcome socket) - Client contacts server by creating a TCP socket (establishing TCP connection), specifying IP address and port number of server process - When contacted by client, server TCP creates a TCP socket to communicate with client socket - Provides reliable, in-order byte-stream transfer ("pipe") between client and server

Socket programming with UDP

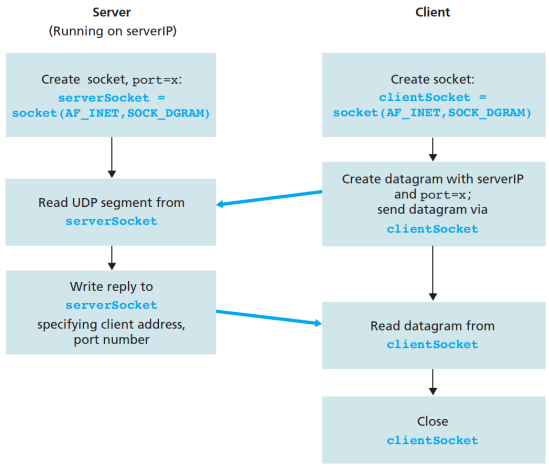


Figure 2.28 ♦ The client-server application using UDP

Example app: UDP client

Python UDPClient

include Python's socket library → `from socket import *`

`serverName = 'hostname'`

`serverPort = 12000`

create UDP socket for server → `clientSocket = socket(AF_INET, SOCK_DGRAM)`

get user keyboard input → `message = raw_input('Input lowercase sentence:')`

Attach server name, port to message; send into socket → `clientSocket.sendto(message.encode(), (serverName, serverPort))`

read reply characters from socket into string → `modifiedMessage, serverAddress = clientSocket.recvfrom(2048)`

print out received string and close socket → `print modifiedMessage.decode()`

`clientSocket.close()`

Example app: UDP server

Python UDPServer

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print ("The server is ready to receive")
while True:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.decode().upper()
    serverSocket.sendto(modifiedMessage.encode(),
                        clientAddress)
```

create UDP socket

bind socket to local port number 12000

loop forever

Read from UDP socket into message, getting client's address (client IP and port)

send upper case string back to this client

Socket programming with TCP

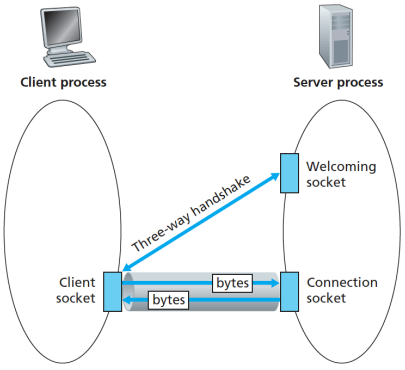


Figure 2.29 ♦ The `TCPServer` process has two sockets

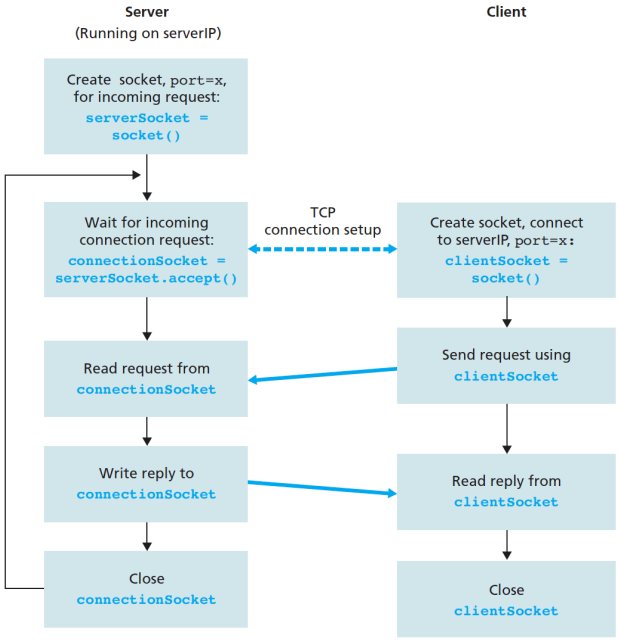


Figure 2.30 ♦ The client-server application using TCP

Example app: TCP client

Python TCPClient

```
from socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence.encode())
modifiedSentence = clientSocket.recv(1024)
print ('From Server:', modifiedSentence.decode())
clientSocket.close()
```

create TCP socket for server, remote port 12000

No need to attach server name, port

Example app: TCP server

Python TCPServer

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(1)
print 'The server is ready to receive'
while True:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence.
                                        encode())
    connectionSocket.close()
```

create TCP welcoming socket

server begins listening for incoming TCP requests

loop forever

server waits on accept() for incoming requests, new socket created on return

read bytes from socket (but not address as in UDP)

close connection to this client (but *not* welcoming socket)

CHAPTER 3 – TRANSPORT LAYER

3.1 Transport-Layer Services

- Provide **logical communication** between application processes running on different hosts
- Transport protocols run in end systems
    - **Send side**: breaks app messages into **segments**, passes to network slayer
    - **Receiving side:** reassembles segments into messages, passes to application layer
- **Network layer** (logical communication between hosts) vs **Transport Layer** (logical communication between processes)

Analogy:
12 kids in Ann's house sending letters to 12 kids in Bill's house:

- **hosts** = houses
- **processes** = kids
- **app messages** = letters in envelopes
- **transport protocol** = Ann who muxes letters from kids and Bill who demuxes to kids
- **network-layer protocol** = postal service

* the transport layer has the responsibility of delivering the data in the segments to the appropriate processes in the hosts

So we have TCP (Transmission Control Protocol) and UDP (User Datagram Protocol).

3.2 Multiplexing and Demultiplexing

Multiplexing at sender:

- Handle data from multiple sockets, add transport header
- Gather data chunks at the source host from different sockets, encapsulating each data chunk with header info to create segments

Demultiplexing at receiver:

- Use header info to deliver received segments to correct socket
- Host receives IP datagrams
    - Each datagram has source IP address, destination IP address and carries one transport layer segment (which contains a source destination port number)
    - The host uses IP address and port numbers to direct segment to appropriate socket

Connectionless demux	**Connection-oriented demux**
- UDP socket identified by2-tuple: destinationport number and IP address	- TCP socket identified by a 4-tuple: (source IP address, source port #, destination IP address, destination port #)

3.3 Connectionless Transport: UDP

UDP:

- "no frills", "bare bones" internet transfer protocol
- "best effort" service b/c segments may be lost and delivered out of order
- connectionless -> no handshaking, each segment handled independently of others
- uses: streaming multimedia apps, DNS, SNMP

Why UDP?

- **No delay** added by establishing a connection
- **Simple** because there is no connection state at sender or receiver
- It has a **small header size**
- No congestion control, so it can blast away as **fast** as desired

UDP checksum

- Used to **detect errors** (flipped bits) in transmitted segment
- sender:
    - treats segment contents as a sequence of 16-bit integers
    - checksum: of segment's content
    - sender puts checksum value into UDP checksum field
- receiver:
    - computes checksum of received segment
    - checks that the checksum computed equals checksum value:
        - NO – error detected
        - YES – no error detected

3.4 Principles of reliable data transfer (rdt)

- The layer below the reliable data transfer protocol may be unreliable

Sending side:

- rdt accepts data from upper layer via the rdt_send(data) event, creates a packet containing data and sends the packet into the channel

Receiving side:

- rdt receives a packet from the underlying channel via the rdt_rcv(packet) event, removes the data from the packet and passes the data up to the upper layer

ARQ (Automatic Repeat request) protocols:

The message-dictation protocol uses both **positive acknowledgements** and **negative acknowledgements** which allow the received to let the sender know what has been received correctly and what has been received in error and thus requires repeating.

ARQ requires:

- **error detection**: the receiver needs a mechanism to detect when bit errors have occurred
- **receiver feedback**: the receiver needs to provide feedback – positive ACK or negative ACK – to the sender
- **retransmission**: a packet that is received in error at the receiver will be transmitted by the sender

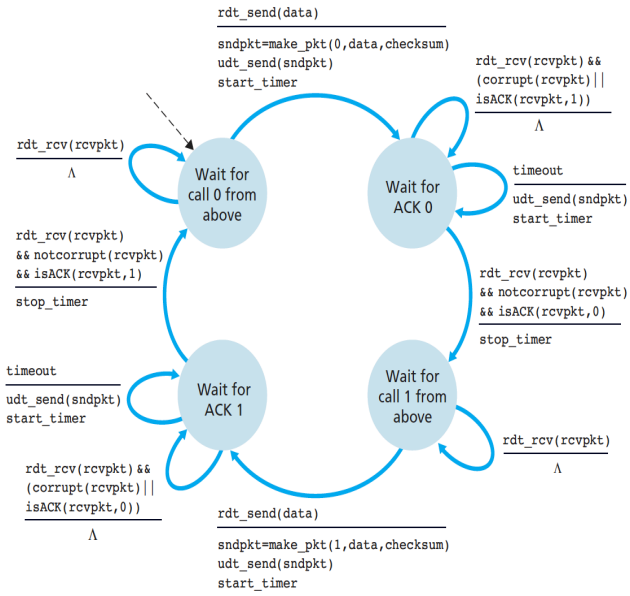
Data packets:

- contain a **sequence number** field, where the sender numbers the packets. The receiver uses this number to determine if a packet is a retransmission.

Stop and wait – sender sends one packet and waits for receiver response before sending next

rdt 3.0:

- sender transmits a data packet, which either that packet or the receiver's **ACK** of the packet gets lost
    - either case, no reply from the receiver is going to the sender, so the sender should re send the packet after waiting long enough for a round-trip delay between the sender and receiver
    - the sender therefore chooses a time value, **countdown timer**, such that a packet loss is likely to have happened
    - if the ACK is not received within this time, the packet is re transmitted
    - But, if the packet experiences a particular large delay, the sender may re transmit the packet even though it's not lost
        - **Duplicate data packets** are handled through **sequence numbers**
        - Data packets contain a **sequence number** field, where the sender numbers the packets. The receiver uses this number to determine if a packet is a retransmission.
        - If duplicate, deletes duplicate
    - The sender therefore needs to be able to start a timer every time a packet is sent, respond to a timer interrupt and stop the timer
    - The receiver specifies sequence number of packet being acknowledged
- Uses stop-and-wait (that's why only 0 and 1 sequence number)



Here only using sequence numbers 1 and 0.

Cases: no loss, packet loss, ACK loss, and premature timeout

- rdt 3.0 is correct but performance stinks

- e.g.: 1 Gbps link, 15 ms prop. delay, 8000 bit packet:

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

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

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

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

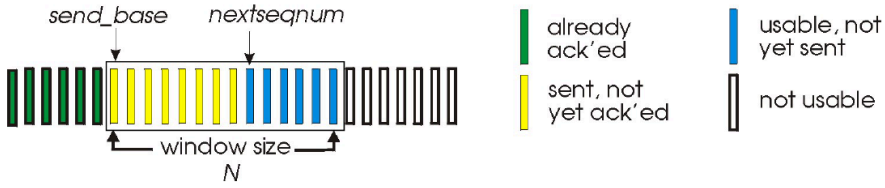
Pipelined protocols:

Pipelining: sender slows multiple, "in-flight", yet-to-be-acknowledged packets, therefore range in sequence numbers must be increased

1. **go-Back-N**
    - sender can have up to N unACK packets in pipeline
    - receiver only sends **cumulative ACK**
    - sender has timer for oldest unACKed packet

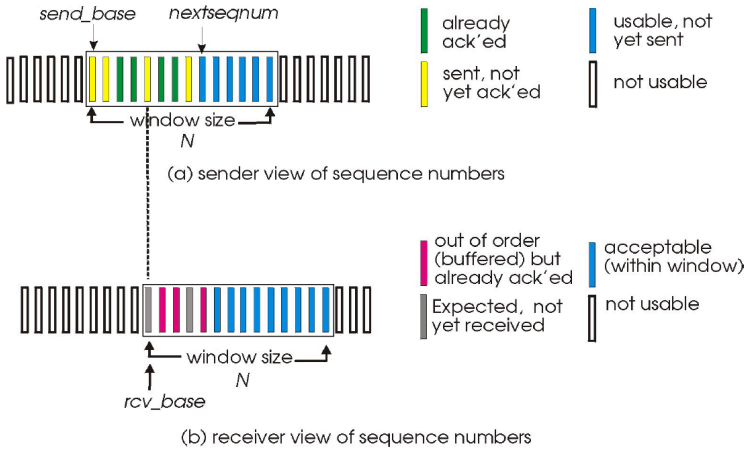
        - when timer expires, retransmit all unACKed packets
    - receiver only sends ACK for packet with highest in-order sequence
        - we assume that all previous packets have been ACKed since its cumulative
    - when receiver gets an out of order packet, it discards it and sends ACK with the last ACKed sequence number

- k-bit seq # in pkt header
- "window" of up to N, consecutive unack'ed pkts allowed



- ACK(n): ACKs all pkts up to, including seq # n - *"cumulative ACK"*
    - may receive duplicate ACKs (see receiver)
- timer for oldest in-flight pkt
- *timeout(n):* retransmit packet n and all higher seq # pkts in window

2. **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, only retransmit that unACKed packet
    - When receiver receives a packet out of order, it buffers it and sends ACK



**sequence number must be 2 * window size!!

3.5 Connection-oriented transport: TCP

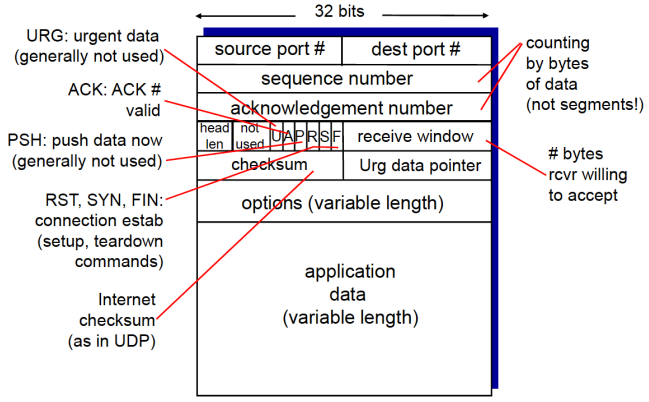
TCP

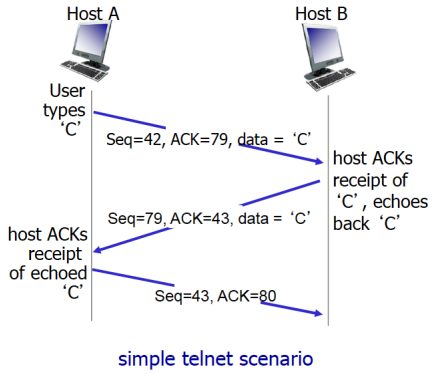
The internet's transport-layer protocol

- Point-to-point
- Reliable, in-order byte steam

- Pipelined
- Full duplex data (bi-directional data flow)
- Connection-oriented -> handshaking
- Flow controlled (sender not overwhelmed -> window)

TCP segment structure





simple telnet scenario

round-trip timeout must be longer than RTT

estimatedRTT = (1 – a) * estimateRTT + a * sampleRTT

devRTT = (1-B)*devRTT + B*|sampleRTT – estimatedRTT|

timeoutInterval = estimatedRTT + 4*devRtt

- usually a=0.125 and B=0.25
- the devRTT is the "safety margin"

Data Transmission:

- TCP creates rdt service on top of IP's unreliable service
- If seq=92 with 8 byte of data, ACK=100 (92+8)

**ACKs are cumulative and correctly received but our-of-order segments are not individually ACKed by receiver -> if out of order, receiver keeps sending same previously ACKed sequence

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

if there is segment lost, there will likely be many duplicate ACKS – GBN

Flow control:

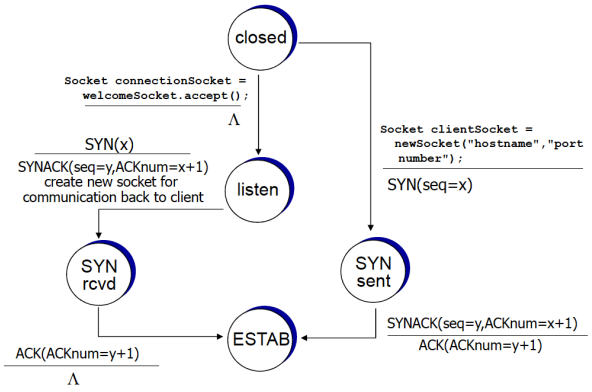
- Receiver advertises free buffer space by including **rwnd** value in TCP header so that the sender limits amount of unACKed data to receiver -> gurantees receiver will not overflow

Connection Management:

- Before exchanging data, sender and receiver "handshake" – agree on connection parameters

- 3-way handshake:

TCP 3-way handshake: FSM



3.6 Principles of congestion control

Congestion: too many resources sending too much data too fast for network to handle

- Manifestations: lost packets and long delays
- Causes: two senders, two receivers, one router

Host sending rate = λ
Router link capacity = R bytes/sec

1: 2 senders, 2 receivers, 1 router, infinite buffer, no error recovery:

- The closer the sending rate approaches maximum capacity (R/2), the larger the delay
- Infinite buffer means that the router stores an infinite amount of packets when packet-arrival rate exceeds outgoing link's capacity
- Throughput and delay as a function of host sending rate (λ_{in} bytes/sec)
- *Cost: large queuing delays are experiences as the packet-arrival rate nears the link capacity*

2: 2 senders, 2 receivers, 1 router, finite buffer, retransmission available (rdt):

- Finite buffer means that packets will be dropped when arriving to an already full buffer
- **Offered load** the rate at which the transport layer sends segments into the network
- Cost: the sender must perform retransmissions in order to compensate for dropped packets due to buffer overflow
- *Cost: unneeded retransmission by the sender in the face of large delays may cause a router to use its link bandwidth to forward unneeded copies of a packet*

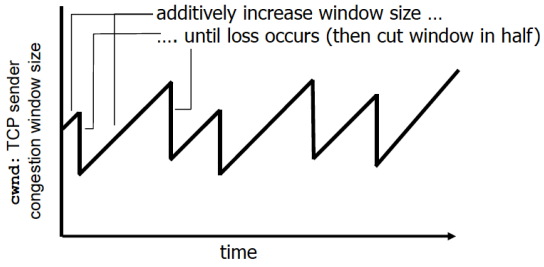
3: four senders, finite router, multihop paths, rdt:

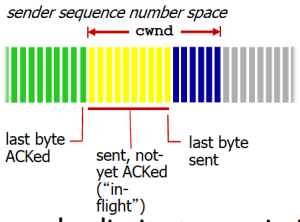
- For multihop paths, routers are shared
    - So for shared router A, when the sending rate of connection 1 increases, the all arriving packets from connection 2 are dropped and hence the work done to get connection 2 packets to router A was wasted
- *Cost: when a packet is dropped along a path, the transmission capacity that was used at each of the upstream links to forward that packet to the point at which it is dropped ends up being wasted*

3.7 TCP congestion control

Congestion control approach: sending increases transmission rate (window size), probing for usable bandwidth until loss occurs

1. **Additive increase**: increase **cwnd** (dynamic function of perceived network congestion) by 1 MSS every RTT
2. **Multiplicative decrese**: cut **cwnd** in half after loss





- sender limits transmission:

$$\texttt{LastByteSent-LastByteAcked} \leq \texttt{cwnd}$$

- **cwnd** is dynamic, function of perceived network congestion

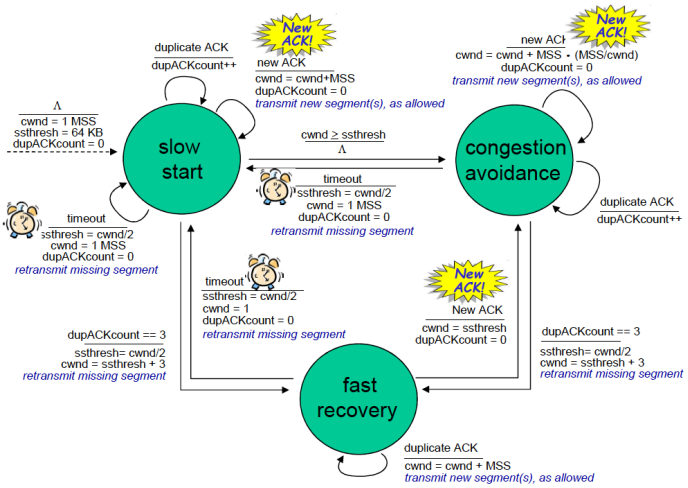
TCP sending rate:

- *roughly*: send cwnd bytes, wait RTT for ACKS, then send more bytes

$$\text{rate} \approx \frac{\text{cwnd}}{\text{RTT}} \text{ bytes/sec}$$

- When connection begins, increase rate exponentially until first loss event
  - Initially cwnd = 1 MSS (min seg size)
  - Double cwnd every RTT (inceement cwnd for every ACK received)
- Loss indicated by 3 duplucate ACKS
- Window then gros exponentially to threshold (1/2 of its value before timeout), then grows linearly

Summary: TCP Congestion Control



TCP throughput as a function of window size

- If window size = ¾ W, TCP throughput = ¾ W/RTT bytes/sec

TCP fairness: if K TCP sessions share same bottleneck link of bandwidth R, each should have average rate of R/K

**Some apps used UDP because they don't want a rate throttled by congestion control and will tolerate packet loss

CHAPTER 4 – NETWORK LAYER: THE DATA PLANE

4.1 Overview of Network layer

Network Layer

- Transport segment from sending host (encapsulates segments into datagrams) to receiving host (delivers segments to transport layer)
- The router examines header fields in all IP datagrams passing through it

Network-layer functions:

- **Forwarding:** moves packets from router's input [link] to appropriate router output [link]
- **Routing:** determine route taken by packets from source to destination (with algorithm)

Data plane - FORWARDING

- Local, per-router (forwarding) function
- Determines how datagram arriving on router input port is forwarded to router output port

Control plane - ROUTING

- Network-wide logic
- Determines how datagram is routed among routers along end-to-end path from source host to destination
- There are two control-plane approaches:
  - **Traditional routing algorithms:** implemented in routers
  - **Software-defined networking (SDN):** implemented in (remote) servers

Per-router control plane

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

Logically centralized control plane

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

Network Service Model

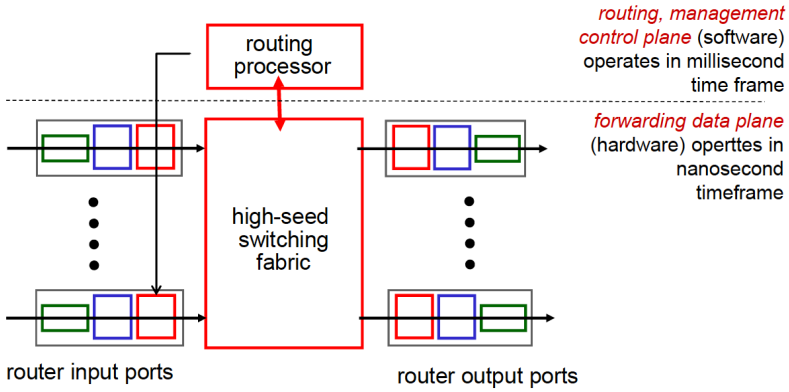
Defines the characteristics of end-to-end transport of packets between sending and receiving end systems

- Services for individual datagrams:
  - Guaranteed delivery with a bounded delay (ex: within 40 msec)
- Service for a flow of datagrams:
  - In-order datagram, guaranteed minimum bandwidth to flow (specified bit rate), restrictions on changes in inter-packet spacing (the amount of time between the transmission of two successive packets at the sender is equal to the amount of time between their receipt at destination or that this spacing changes by no more than some specified value), security services (encryption)

Datagram – a network-layer packet

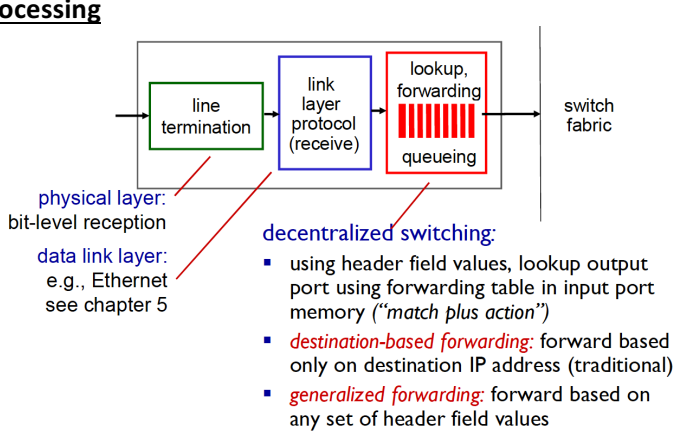
4.2 What's inside a router

A router's input ports, output ports and switching fabric together implement the **forwarding function**. The forwarding functions are sometimes collectively referred as the **router forwarding plane**.
The **router control plane** functions are executing the routing protocols, responding to attached links that go up or down and performing management functions.



- **Input ports**: performs the physical layer function of terminating an incoming physical link at a router. Also performs link-layer function needed to interoperate with the link layer at the other side of the incoming link
- **Switching fabric**: connects the router's input ports to its output ports
- **Output ports**: stores packets received and transmits these packets to the outgoing link
- **Routing processor**: executes the routing protocols, maintains routing tables and attached link state information and computes the forwarding table for the router

Input Processing



Lookup: In a datagram network, each time an end system wants to send a packet, it stamps the packet with the address of the destination end system and then pops the packet into the network.

- Each router uses the packets destination address to forward the packet
- Each router has a **forwarding table** that maps destination addresses to link interfaces
    - When a packet arrives at the router, the router uses the packet's destination address to look up the appropriate output link interface in the forwarding table
- One way to use this forwarding table is by **longest prefix matching**
    - When looking for a table entry for a given destination address, use longest address prefix that matches destination address
    - This Is often performed using **ternary content addressable memories (TCAMs)**

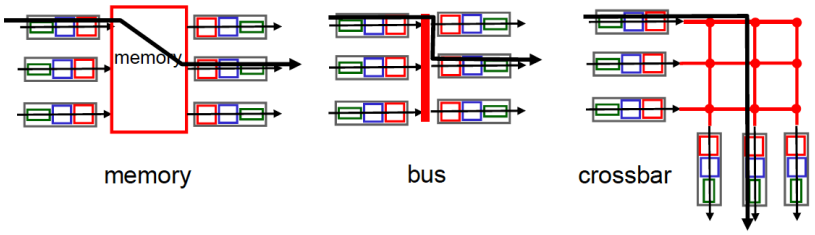
- **Content addressable:** present address to TCAM, retrieve address in one clock cycle

Switching Fabrics

Transfer packet from input buffer to appropriate output buffer

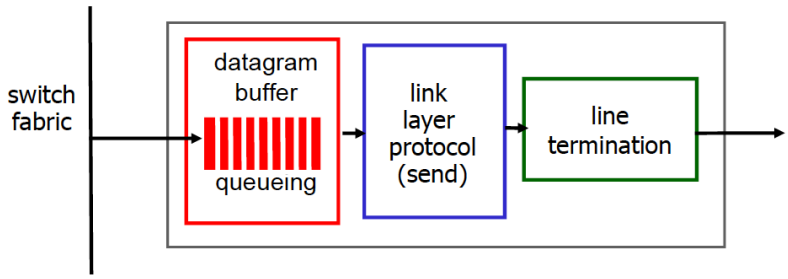
- **Switching rate**: rate at which packets can be transferred from inputs to output

There are 3 types:



1. **Memory**
   - These are the first generation routers
   - Traditional computers with switching under direct control of CPU
   - Packet copied to system's memory
   - Speed limited by memory bandwidth
2. **Bus**
   - An input port transfers a packet directly to the output port over a shared bus without any intervention by the routing processor
     - The packet is received by all output ports but only the port that matches label (on header, put there by input port, removed at output port) will keep the packet
   - **Bus convention**: switching speed limited by bus bandwidth
3. **Interconnection Network**
   - Crossbar switch, interconnection network consisting of 2N buses that connect N input ports to N output ports
     - Each vertical bus intersects each horizontal bus at a crosspoint, which can be opened or closed at any time by the **switch fabric controller**
     - When a packet arrives from port A and needs to be forwarded to port Y, the switch controller closes the crosspoint at the intersection of A and Y, and port A then sends the packet onto its bus, which is picked up only by bus Y.
     - Crossbar networks are therefore capable of forwarding multiple packets in parallel as long as they have different input and output ports.
   - Thus overcomes bus bandwidth limitation

Output Processing



Output port processing takes packets that have been stored in the output port's memory and transmits them over the output link. This includes selecting and de-queueing packets for transmission and performing the needed link-layer and physical-layer transmission functions.

- **Buffering** required from fabric faster rate
    - datagram (packets) can be lost due to congestion, lack of buffers
- **Scheduling** datagrams
    - Priority scheduling -> who gets best performance, network neutrality

Queuing

It is here, at these queues within a router that packets are actually dropped and lost.

Input port queuing

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

Output port queueing

- Buffering when arrival rate via switch exceeds output line speed
- Queueing delay and loss due to output port buffer overflow
- The number of queued packets could grow large enough to exhaust available memory at output port, in which case packets are dropped.
    - The output port can transmit only a single packet in a unit of time.

How much buffering?

Buffering = $\frac{RTT \cdot C}{\sqrt{N}}$

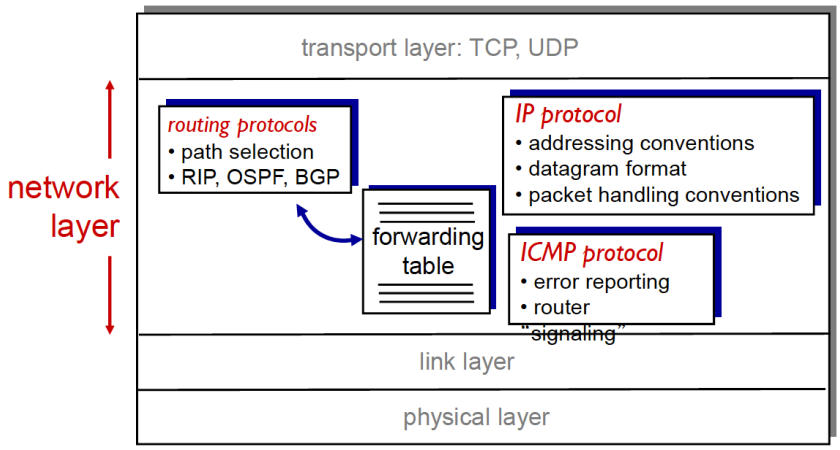
Where C = link capacity and N is number of flows

Scheduling Mechanisms

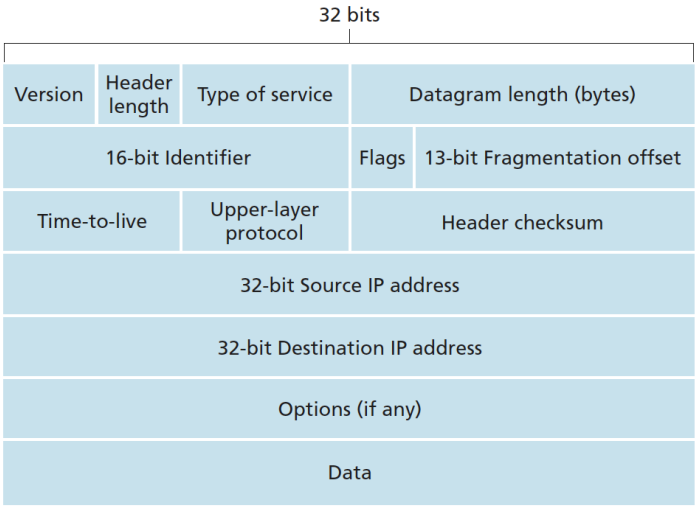
Choosing which packet to send next on link.

- **FIFO Scheduling -** Send in order of arrival to queue
- **Priority Scheduling –** send highest priority queued packet – multiple classes with different priorities
- **Round Robin Scheduling –** cyclically scan class queues, sending one complete packet from each class
- **Weighted Fair Queuing (WFQ) Scheduling –** generalized round robin, each class gets weighted amount of service in each cycle

4.3 IP: Internet Protocol



IP Datagram Format

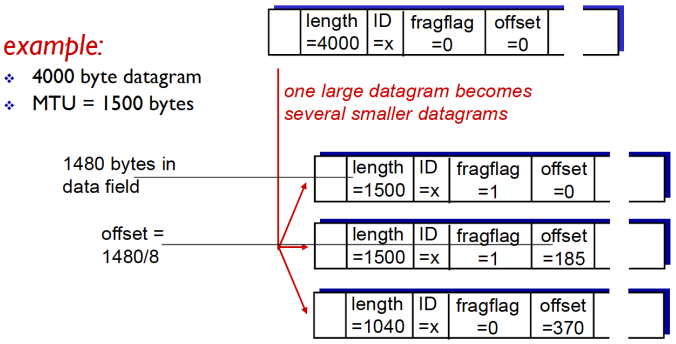


- **Version number** – 4 bits specify the IP protocol version of datagram, tells router how to interpret the remainder of the datagram
- **Header length** – 4 bits needed to determine where in the IP datagram the data begins
- Type of service – TOS bits included to allow different types of IP datagrams (types of data) to be distinguished from each other
- **Datagram length** – 16 bits for total length of datagram (header + data) measured in bytes
- **Identifier, flags, fragmentation offset** – fields that have to do with IP fragmentation/reassembly
- **Time-to-live** – TTL field if included to ensure that datagrams do not circulate forever (from loop) in the network – the field is decremented and if it reaches 0, datagram is dropped
- **Protocol** – field used when a datagram reaches its final destination – it indicates the specific transport-layer protocol to which the data portion of this datagram should be passed (TCP/UDP)
    - Binds the transport and network layers together
- **Header checksum** – helps the router in detecting bit errors, a router computes the header checksum for each received datagram and detects an error condition if the checksum carried in the datagram header does not equal the computed value.
    - This value must be recomputed and stored again at each router (TTL field changes)
- **Source and destination addresses** – stores source and destination IP addresses
    - The source host often determines destination address via DNS lookup

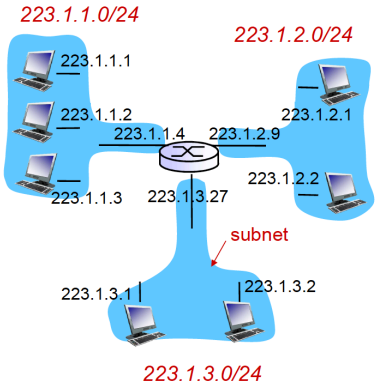
- **Options** – examples such as timestamp, record route taken, specify list of routers to visit, etc…
- **Data** – Contains data, sometimes contains transport-layer segment to be delivered to destination

IP Datagram Fragmentation

- Not all link-layer protocols can carry network-layer packets of the same size
- The maximum amount of data that a link-layer frame can carry is the **maximum transmission unit (MTU)**
    - Because each datagram is encapsulated within the link-layer frame for transport from one router to the next router, the MTU of the link-layer protocol places a hard limit on the length of a datagram
    - The problem is that each link along the route between sender and receiver could use different link-layer protocols that can have different MTUs
    - Large IP datagrams are divided ("fragmented") into **fragments** within net
        - One datagram becomes several datagrams
        - "reassembled" only at final destination
        - IP header bits are used to identify and order related fragments



IP Addressing



IP Address – 32-but identifier for host, router interface
- Subnet part – high order bits
- Host part – low order bits

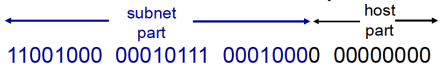
Interface – connection between host/router and physical link
- Routers typically have multiple interfaces
- Hosts typically have 1 or 2 interfaces

Recipe – to determine the subnets, detach each interface from its host/router, creating island of isolated networks
- Each isolated network is called a **subnet**

Subnet – device interfaces with same subnet part of IP address
- Can physically reach each other without intervening router

CIDR: Classless Inter Domain Routing

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



200.23.16.0/23

How does a host get an IP address?

- Hard-coded by system admin in a file
  Control Panel -> network -> Configuration -> TCP/IP -> Properties
- **DHCP: Dynamic Host Configuration Protocol** – dynamically get address from a server
  - "plug-and-play"
  - 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 whole connected / "on")
    - Support for mobile users who want to join network
  - Overview:
    - Hosts broadcasts "DHCP discover" message (optional)
    - DHCP server responds with "DHCP offer" message (optional)
    - Host requests IP address: "DHCP request" message
    - DHCP server sends address: "DHCP ack" message
  - DHCP can return more than just allocated IP address on subnet
    - Address of first-hop router for client
    - Name and IP address of DNS server
    - Network mask (indicating network versus host portion of address)

How does an ISP get a block of addresses?

- In order to obtain a block of addresses for use within an organization's subnet
  - A network administrator first contacts its ISP, which would provide addresses from a larger block of addresses that had already been allocated to the ISP
- **ICANN: Internet Corporation for Assigned Name and Numbers**
  - Allocates addresses
  - Managers DNS
  - Assigns domain names, resolves disputes

Network Address Translation (NAT)

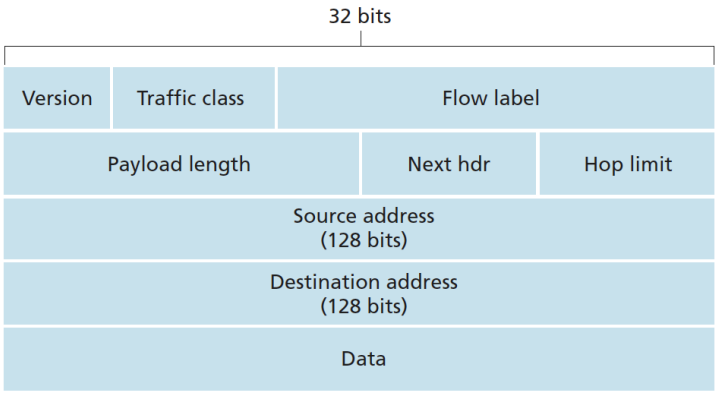
- A realm with private addresses refers to a network whose addresses only have meaning to devices within a network
- The NAT router behaves to the outside world as a single device with a single IP address
  - Hides the details of the home network from the outside world
- The **NAT translation table** allows the router to know the internal host to which it should forward a given datagram

- Local networks use just one IP address as far as outside world is connected
  - Range of addresses is not needed from ISP: just one IP address without notifying outside world
  - Can change ISP without changing addresses of devices inside local network
  - Devices inside local network not explicitly addressable, visible by outside world (good for security)
- NET router must:
  - Replace (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
  - Remember (in NAT translation table) every (source IP address, port #) to (NAT IP address, new port #) translation pair

- Replace (NAT IP address, new port #) in destination fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

IPv6

- 32-but address space soon to be completely allocated
- **datagram format:**
  - fixed-length 40-byte header
  - no fragmentation allowed



- priority – identify priority among datagrams in flow
- flow Label – identify datagrams in same "flow"
- next header – identify upper layer protocol for data
- 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 messages types and multicast group functions

CHAPTER 5 – NETWORK LAYER: THE CONTROL PLANE

5.1 Introduction

Network-layer functions:

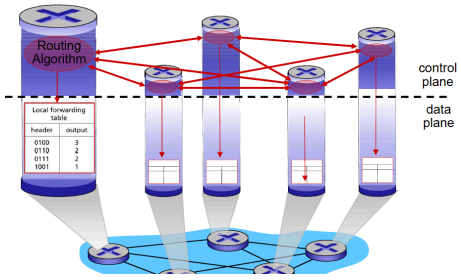
- **Forwarding:** moves packets from router's input [link] to appropriate router output [link]
- **Routing:** determine route taken by packets from source to destination (with algorithm)

Data plane - FORWARDING

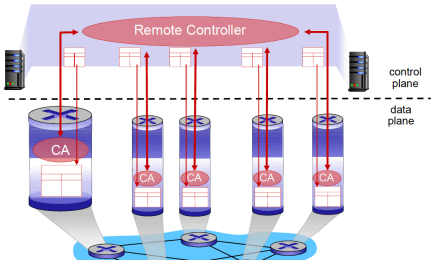
- Local, per-router (forwarding) function
- Determines how datagram arriving on router input port is forwarded to router output port

Control plane - ROUTING

- Network-wide logic
- Determines how datagram is routed among routers along end-to-end path from source host to destination
- There are two control-plane approaches:
  - **Traditional:** implemented in routers (per router control)
  - **Software-defined networking (SDN):** implemented in remote servers (logically defined networking)



Traditional (per-router) Control Plane
Individual routing algorithm components in each and very router interact with each other in control plane to compute forwarding tables.



Logically centralized Control Plane
A distinct (typically remote) controller interacts with local control agents (CAs) in routers to compute forwarding tables

5.2 Routing protocols

Routing protocol goal: determine "good" paths from sending hosts to receiving hosts, through network of routers.

- **Path**: sequence of routers packets will traverse in going from given initial source host to given final destination host
- **"good"**: least cost, fastest, least congested

We use graphs to denote networks, where we have G = (N,E) where N is the set of routers and E is the set of links.

- c(x,y) = cost of link when going from x to y

Routing algorithm classification

Global vs Decentralized

- **global**:
  - all routers have complete topology, link cost info
  - "link state" algorithms
  - computes the least-cost path between a source and destination using complete, global knowledge about the network (needs all information about connectivity and link costs)
- **decentralized**:
  - router knows physically-connected neighbors, link costs to neighbors
  - iterative process of computation, exchange of info with neighbours
  - "distance vector" algorithms
  - each node begins with only the knowledge of the costs of its own directly attached links, then through an iterative process of calculation and exchange of information, a node gradually calculates the least-cost path

Static vs Dynamic

- **static**:
  - routes change slowly over time
- **dynamic**:
  - routes change more quickly (periodic update, in response to link cost changes)

Link-State Routing Algorithm -> Dijkstra's Algorithm

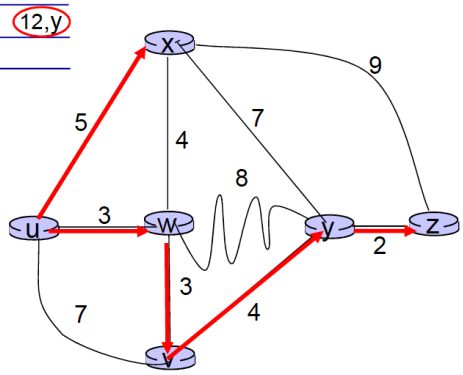
- network topology, link costs known to all nodes (all nodes have same information)
- computes least cost paths from one node (source) to all other nodes
  - gives forwarding table for that node
- Notation:
  - c(x,y) – link cost from node x to node y
  - D(v) – current value of cost of path from source to destination v
  - p(v) – predecessor node along path from source to v
  - N' – set of nodes whose least cost path is definitively known

Dijkstra's algorithm: example

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					

notes:

- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)



- Complexity: O(nlogn)

Distance Vector Algorithm -> Bellman-Ford Equation

let

$d_x(y)$:= cost of least-cost path from x to y

then

$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

$d_v(y)$: cost from neighbor v to destination y

$c(x,v)$: cost to neighbor v

$\min_v$: *min* taken over all neighbors v of x

- use previously computed shortest paths
- $d_x(y)$ starts as an estimate
- 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 B-F min equation
- **Iterative, asynchronous:**
    - each local iteration caused by:
        - local link cost change
        - DV update message from neighbor
- **Distributed:**
    - Each node notifies neighbors only when its DV changes
        - Neighbors then notify their neighbors if necessary
- Link costs changes:
    - When a link costs decreases -> fast to stabilize
    - When link costs increases -> slow to stabilize -> **count to infinity problem**

Comparison of LS and DV algorithms

message complexity

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

speed of convergence

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

robustness: what happens if router malfunctions?

LS:

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

DV:

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

5.3 Intra-AS routing in the Internet: OSPF

Scale: with billions of destinations

- Can't store all of them in routing tables
- Routing table exchange would swamp links

Administrative autonomy:

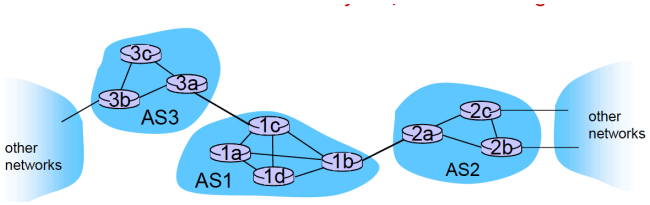
- Internet is a network of networks
- Each network admin may want to control routing in its own network

The internet approach to scalable routing

- Aggregate routers into regions known as "**autonomous systems**" (**AS**) -> a.k.a "domains"
    - With each AS consisting of a group of routers that are typically under the same administrative control

Inter-AS routing

- Routing among AS's
- Gateway routers perform inter-domain routing (as well as intra-domain routing)
- **Tasks:**
    - Suppose a router in AS1 receives a datagram destined outside of AS1, router should forward packet to gateway router, but which one?
        - AS1 must learn which destinations are reachable through different AS_'s
        - And AS1 must also propagate this reachability information to all of its routers



Intra-AS routing

- Routing among hosts, routers in same AS ("network")
- All routers in AS must run same intra-domain protocol
- Gateway router: at "edge" of its own AS, has link(s) to router(s) in other AS's
- Also known as **interior gateway protocols (IGP)**
- Most common routing protocols:
    - **RIP**: Routing information protocol
    - **OSPF**: Open Shortest Path First
    - **IGRP**: Interior Gateway Routing Protocol

OSPF (Open Shortest Path First)

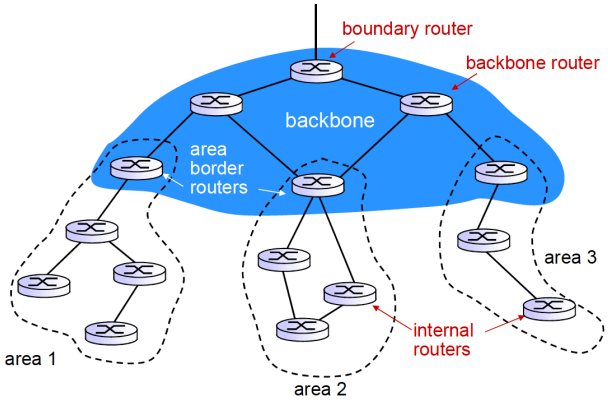
- "open": publicly available
- uses link-state algorithm (Dijkstra's algorithm)
- router **floods** OSPF link-state advertisements to all other routers in entire AS
    - carried in OSPF messages directly over IP (rather than TCP or UDP)
    - link state: for each attached link
- IS-IS routing protocol is nearly identical to OSPF

- A router constructs a complete topological map (graph) of the entire autonomous system
- The router then runs Dijkstra's algorithm to determine a shortest-path tree to all subnets, with itself as the root note
- Individual link costs are configured by the network administrator
- OSPF doesn't mandate a policy for how link weights are set, but instead provides the protocol for determining least-cost path routing for the given set of link weights.
- The router broadcasts routing information to all other routers in the autonomous system, not just its neighbours
    - A router broadcasts a link state's information whenever there is a change in a link's state and also periodically

        - OSPF advertisements are contained in OSPF messages that are carried directly by IP
            - OSPF protocol must implement reliable message transfer and link state broadcast
            - The OSPF also checks that links are operational (HELLO message)
    - Advanced features:
        - Security (authentication)
        - Multiple same-cost paths allowed
        - **Hierarchical OSPF** in large domains

Hierarchical OSPF

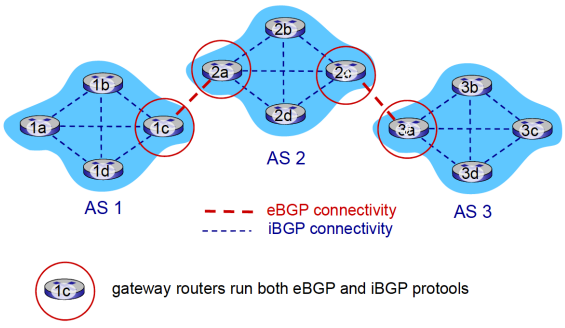
- Two-level hierarchy: local area, backbone
    - Link-state advertisements only in area
    - Each node has detailed area topology, only know direction (shortest path) to nets in other areas
- **Area border routers:** "summarize" distances to networks in own area, advertise to other area border routers
- **Backbone routers**: run OSPF routing limited to backbone
- **Boundary routers:** connect to other AS's



5.4 Inter-AS Routing [among the ISPs]: BGP

BGP (Border Gateway Protocol): "glue that holds the internet together"

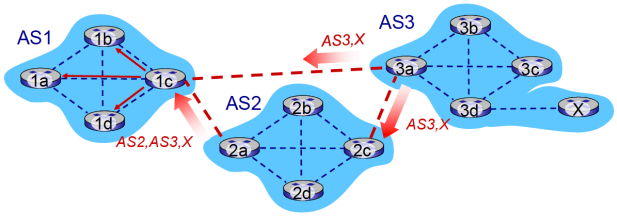
- BGP (path vector protocol) provides each AS as means to:
    - **eBGP**: (external) obtain subnet reachability information from neighboring ASes
    - **iBGP**: (internal) propagate reachability information to all AS-internal routers
    - determine "good" routes to other networks based on reachability information and policy
- allows subnet to advertise its existence to rest of Internet



- **BSP session**: two BGP routers exchange BGP messages over semi-permanent TCP connection:
    - Advertising paths to different destination network prefixes
- When a gateway router in any AS received eBGP-learned **prefixes** (138.16.64/22), the gateway router uses its iBGP sessions to distribute the prefixes to the other routers in the AS.
- In BGP, an AS is identified by its globally unique **autonomous systems number (ASN)**
    - When a router advertises a prefix across a BGP session, it includes with the prefix a number of **BGP attributes**

- "**route**" = prefix + attributes
    - two important attributes:
        - AS-PATH: list of ASes through which prefix advertisement has passed
        - NEXT-HOP: indicates specific internal-AS router to next-hop AS
    - In BGP destinations are not hosts but instead are CIDRized **prefixes,** with each prefix representing a subnet or a collection of subnets
- **Policy-based routing:**
    - Gateway receiving route advertisement uses **import policy** to accept/decline path
    - AS policy also determines whether to advertise path to other neighboring ASes

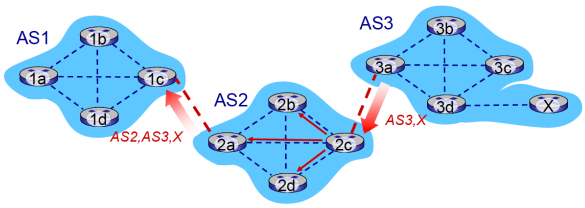
BGP path advertisement



gateway router may learn about multiple paths to destination:

- AS1 gateway router 1c learns path *AS2,AS3,X* from 2a
- AS1 gateway router 1c learns path *AS3,X* from 3a
- Based on policy, AS1 gateway router 1c chooses path *AS3,X, and advertises path within AS1 via iBGP*

BGP path advertisement



- AS2 router 2c receives path advertisement AS3,X (via eBGP) from AS3 router 3a
- Based on AS2 policy, AS2 router 2c accepts path AS3,X, propagates (via iBGP) to all AS2 routers
- Based on AS2 policy, AS2 router 2a advertises (via eBGP) path AS2, AS3, X to AS1 router 1c

BGP messages

BGP messages exchanged between peers over TCP connection

- **OPEN**: opens TCP conn to remote BGP peer and authenticates sending BGP peer
- **UPDATE**: advertises new path (or withdraws old)
- **KEEPALIVE**: keeps conn alive in absence of UPDATES, also ACKs open request
- **NOTIFICATION**: reports errors in previous message, closes connection

How does router set forwarding table entry to distant prefix? ** slide 49

BGP Route Selection

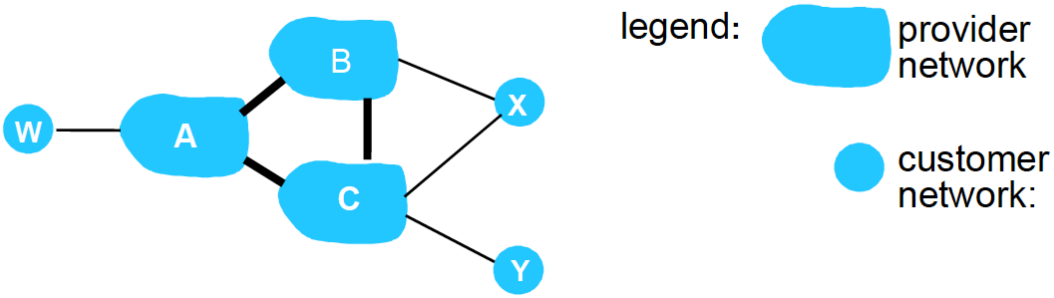
A router may learn about more than one route to destination AS, selects route based on:

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

Hot Potato Routing

Choose local gateway that has least intra-domain cost (even if it has more AS hops to destination)

- Don't worry about inter-domain cost!



- B can **choose not to advertise** a path A might have advertised to it
- A, B, C are **service providers**
- X, W, Y are **customers**
- X is **dual-homed**: attached to two networks

Why different Intra-, Inter- 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

5.6 ICMP: The Internet Control Message Protocol

- Used by hosts and routers to communicate network-level information
- Network-layer "above" IP: ICMP messages carried in IP datagrams
- **ICMP message:** type + code + 8 first bytes of IP datagram causing error

- Source sends series of UDP segments to destination
- When datagram in nth set arrives to nth router:
    - Router discards datagram and sends source ICMP message
    - ICMP message includes name of router and IP address
- When ICMP message arrives, source records RTTs
- *Stopping criteria:*
    - UDP segment eventually arrives at destination host
    - Destination returns ICMP "port unreachable" message
    - Source stops

5.7 Network management and SNMP

Autonomous systems (aka networks): thousands of interacting hardware/software components

Network management includes the deployment, integration and coordination of the hardware, software and human elements to monitor, etc….., the network and element resources to meet the real-time, operational performance and quality of service requirements at a reasonable cost

*** MORE ABOUT MANAGEMENT

CHAPTER 6 – LINK LAYER

6.1 introduction, service

6.2 Error detection, correction

6.3 Multiple Access protocols

6.4 LANs

6.7 A day in the life of a web request