Table of Contents

COMPUTER NETWORKS NOTES

Computer Networks and the Internet

1.1 What is the Internet?

The **Internet is a computer network that interconnects** hundreds of millions of **hosts/end systems**. End systems are connected together by a network of **communication links** and **packet switches**. There are many types of communication links such as coaxial cable, copper wire, optical fibre and radio spectrum. Depending on the type of link, it can transmit data at different rates, with the **transmission rate** of a link measure in **bits/second**. When one end system sends data to another end system, the data is segmented and header bytes are added to each segments. These packages of information are referred to as **packets**, which are then reassembled at the target location into the original data.

A **packet switch** takes a packet arriving on one of its incoming communication links and forwards that packet on one of its outgoing communication links. There are many different types of packet switches but the most prominent types currently are **routers** and **link-layer switches**. **Routers** are used in the **network core** whilst **link-layer switches** are typically used in **access networks**. The sequence of the packet travelling from one end user to another is known as a **route** or **path** through the network.

End systems access the Internet through **Internet Service Providers (ISPs)**, which include:

- residential ISPs such as local cable, or telephone companies
- corporate ISPs
- university ISPs
- ISPs that provide Wi-Fi access in airports, hotels and other public places.

Each **ISP is a network of packet switches and communication links**, which provide a variety of types of network access:

- residential broadband access such as cable modem or DSL
- high-speed local area network access
- wireless access
- 56 kbps dial-up modem access

End systems, packet switches and other pieces of the Internet run protocols that control the sending and receiving of information with the Internet. The **Transmission Control Protocol (TCP)** and the **Internet Protocol (IP)** are two of the most important protocols in the Internet. The **IP protocol specifies the format of the packets that are sent and received** among routers and end systems. The Internet's principal protocols are collectively known as **TCP/IP**. **Internet standards are developed by the Internet Engineering Task Force (IETF)**. The IETF standards **documents are called request for comments (RFCs)**. RFCs started out as general requests for comments, to resolve network and protocol design problems. They define protocols such as TCP, IP, HTTP and SMTP.

The **Internet** can also be referred to as an **infrastructure that provides services to applications**. These applications include electronic mail, web surfing, social networks, IM, Voice-over-IP (VoIP), peer-to-peer (P2P) file sharing, remote login etc. These are referred to as **distributed applications** since they involve multiple end systems. End systems attached to the Internet provide an **Application Programming Interface (API)** that **specifies how a program running on one end system** asks the Internet infrastructure to **deliver data to a specific destination program running** on **another end system**. The Internet has an **API** that the program sending data must follow to have the Internet deliver the data to the program that will receive the data. It can also provide multiple services to its applications.

What is a protocol?

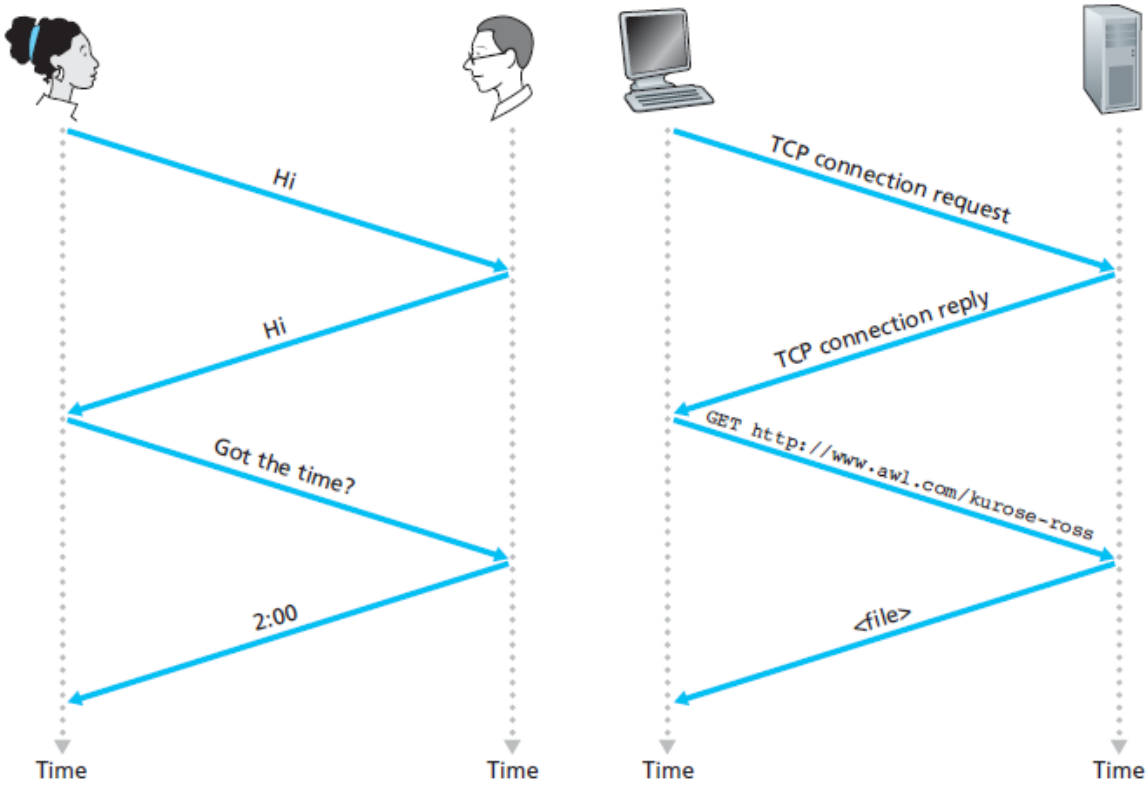


Figure 1.2 ♦ A human protocol and a computer network protocol

All activity in the Internet involving two or more communication remote entities is governed by a protocol. For example:

- hardware-implemented protocols in two physically connected computers control the flow of bits on the "wire" between the two network interface cards
- congestion-control protocols in end systems control the rate at which packets are transmitted between sender and receiver
- router protocols determine a packet's a path from source to destination

A **protocol defines the format and the order of messages exchanged between two or more communicating entities**, as well as the **actions t**aken on the transmission and/or **receipt of a message** or other event.

1.2 The Network Edge

End systems include desktop computers, servers and mobile computers and can also be **referred to as hosts** because they host application programs such as a web browser program, web server program, email programs etc. (**host = end system**). Hosts, however, sometimes are further divided into two categories: **clients and servers**. Clients are typically desktop and mobile PCs whereas servers are more powerful machines that store and distribute web pages, stream video and so on. These servers usually reside in large **data centres** where we receive search results, email etc.

Access Networks

The **Access Network is the network that physically connects an end system to the first router** (known as the "edge router") on a path from the end system to any other distant end system.

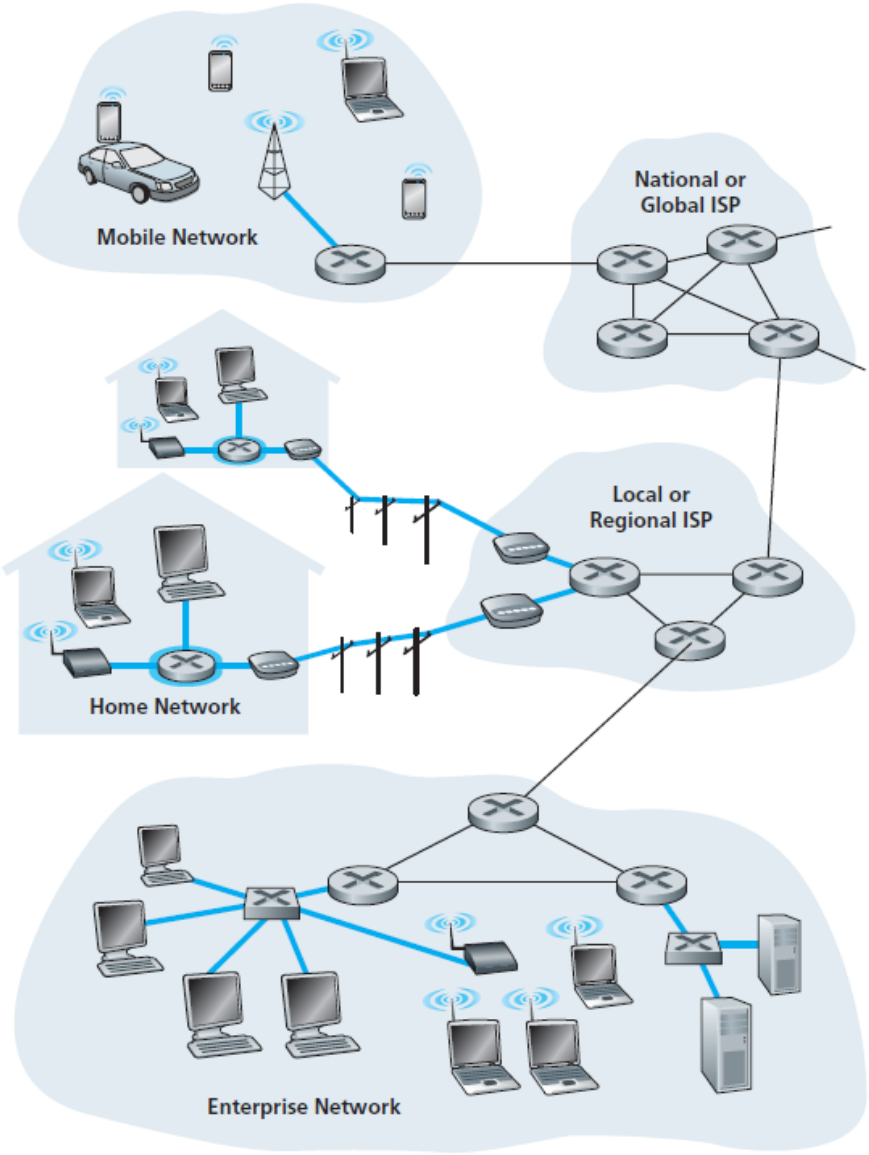


Figure 1.4 ♦ Access networks

The most prevalent types of broadband residential access are digital subscriber line (DSL) and cable.

Digital Subscriber Line (DSL)

A resident typically obtains DSL from the same local telephone company (Telco) that provides its wired local phone access. When DSL is used a customer's Telco is also its ISP.

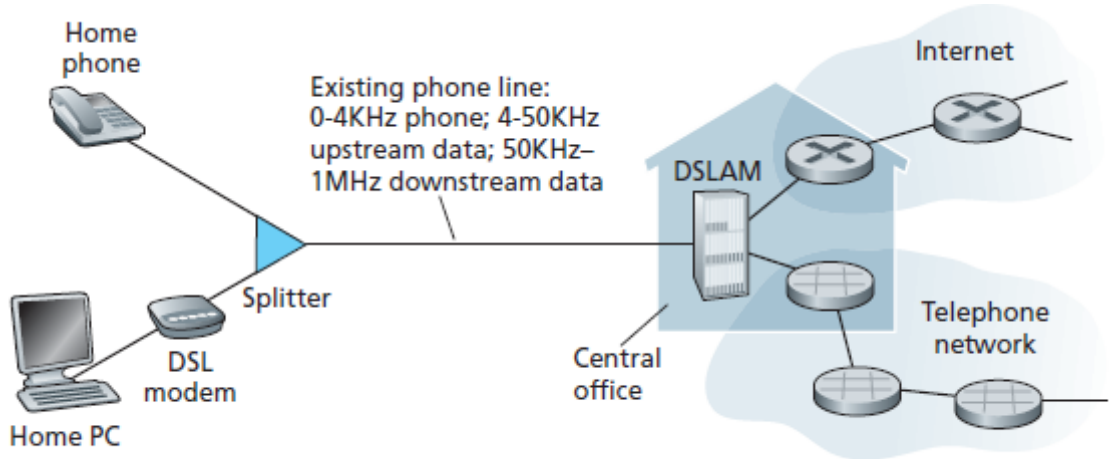


Figure 1.5 ♦ DSL Internet access

Each customer's DSL modem uses the existing telephone line (twisted-pair copper wire) to exchange data with a digital subscriber line access multiplexer (DSLAM) located in the Telco's local central office (CO). The home's DSL modem takes digital data and translates it to high-frequency tones for transmission over telephone wires to the CO, which are translated back into digital format at the DSLAM. The residential telephone line carries both data and traditional telephone signals simultaneously, which are encoded at different frequencies:

- A high-speed **downstream channel**, in the **50 kHz to 1 MHz band**
- A medium-speed **upstream channel**, in the **4 kHz to 50 kHz band**
- An ordinary two-way **telephone channel**, in the **0 to 4 kHz band**

A splitter separates the data and telephone signals arriving to the home and forwards the data signal to the DSL modem. In the CO, the DSLAM separates the data and phone signals and sends the data into the Internet. Both the downstream and upstream rates are different, so the **access is asymmetric**.

Cable

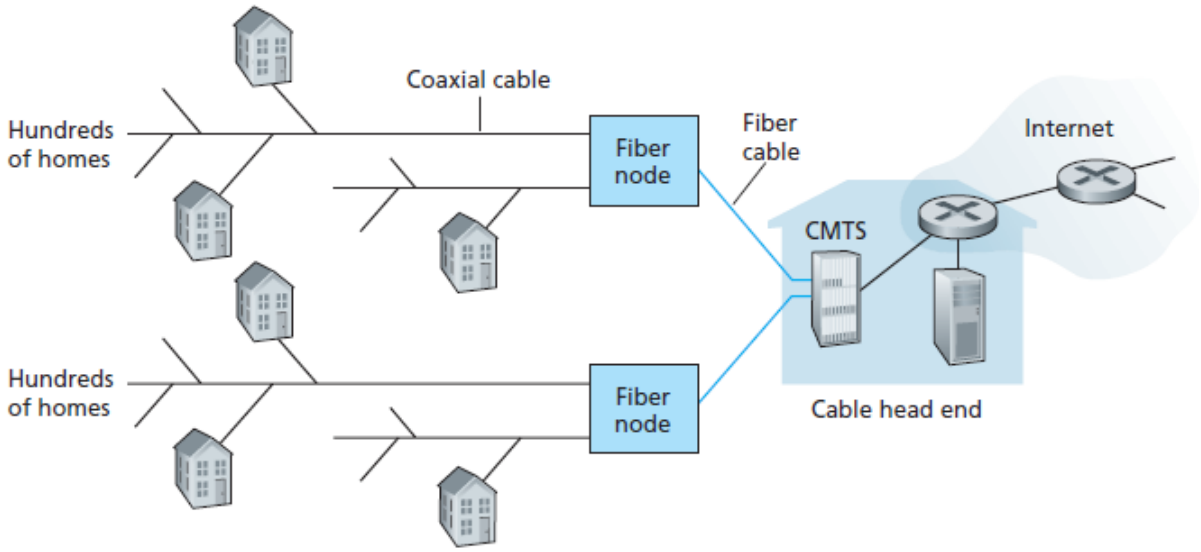


Figure 1.6 ♦ A hybrid fiber-coaxial access network

Cable Internet access makes use of the cable television company's existing cable television infrastructure. Fiber optics connect the cable head end to the neighbourhood-level junctions, where coaxial cable to then used to reach individual houses and apartments. Both fiber and coaxial cable are employed in this system, which is why it is often referred to as **hybrid fiber coax (HFC).** Cable internet requires special modems, called cable modems, and it connects to the home PC through a Ethernet port. At the cable head end, the **cable modem termination system (CMTS)** serves a similar function to the DSL networks' DSLAM, turning the analogue signal sent from the cable modems back to digital format. **Cable modems divide the HFC network into two channels, downstream and upstream**. Like DSL, access is also **asymmetric** with downstream allocated a higher transmission rate than upstream. An important characteristic is that every packet sent by the head end travels downstream on every link to every home and every packet set by the head end travels downstream on every link to every home and every packet sent by a home travels on the upstream channel to the head end. Therefore if **more users are simultaneously downloading a file**, the **actual rate** will be **lower** however if there are **less active users**, then **downstream rate** will be much **faster**.

Fiber to the home (FTTH)

FTTH provides an **optical fiber path from the CO directly the home**. The simplest optical distribution network is called direct fiber, with one fiber leaving the CO for each home. Most commonly though, each **fiber** leaving the CO is actually **shared** by many homes and when the **fiber gets relatively close** to the homes, then it is **split into individual customer-specific fibres**. There are two competing optical-distribution network architectures that perform this splitting: **active optical networks (AONs)** and **passive optical networks (PONs)**.

Local Area Network (LAN)

There are many types of LAN technologies however Ethernet is the most prevalent access technology in corporate, university and home networks. Ethernet users use twisted-pair copper wire to connect to an Ethernet switch which is then connected into the larger Internet.

However people are also using wireless LAN settings to transmit/receive packages to/from an access point that is connected into the enterprise's network, which in turn is connected to the wire Internet.

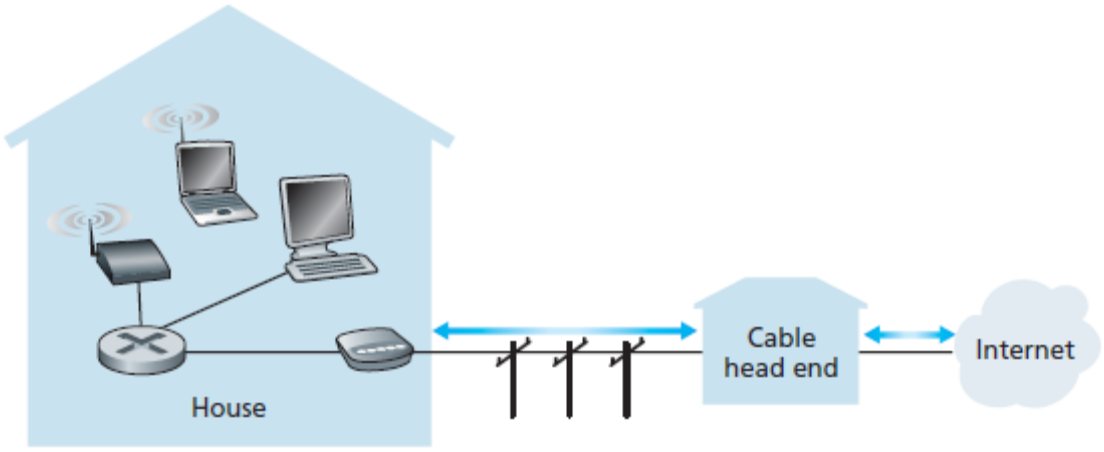


Figure 1.9 ♦ A typical home network

Wide-Area Wireless Access: 3G and LTE

Unlike Wi-Fi, users may be much further away from the base station. Telecommunications companies have made enormous investments in **third-generation (3G) wireless**, which provides packet-switched wide-area wireless at speeds in **excess of 1Mbps** and an even higher speed would be fourth-general (4G) wireless. **Long-Term Evolution (LTE)** has its roots in 3G however have even higher rates in **excess of 10 Mbps**.

1.3 The Network Core

Packet Switching

Packets are transmitted over each communication link at a **rate equal to the full transmission rate of the link**. So if the source end system, or a packet switch is sending a packet of **L bits** over a link with a transmission rate **R bits/sec**, then the **time to transmit the packet is L/R second**s.

Most packet switches use **store-and-forward transmission** at the inputs tot the links. This means that the packet switch must **receive the entire packet before it can begin to transmit the first bit** of the packet onto the outbound link.

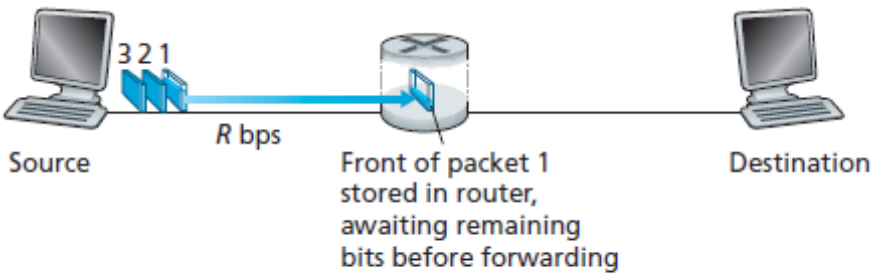


Figure 1.11 ♦ Store-and-forward packet switching

Only after the router has received *all* the packet's bits can it begin to transmit the packet onto the outbound link. **End to end delay = 2L/R** assuming zero propagation delay.

Queuing Delays and Packet Loss

Each **packet switch has multiple links** attached to it, and for **each attached link**, the packet switch has an **output queue/buffer**. This stores the packets that the router is about to send into that link. Therefore in addition to the store-and-forward delays, the packets suffer output buffer **queuing delays**. The amount of buffer space is finite and if it is completely full, **packet loss** with occur, either

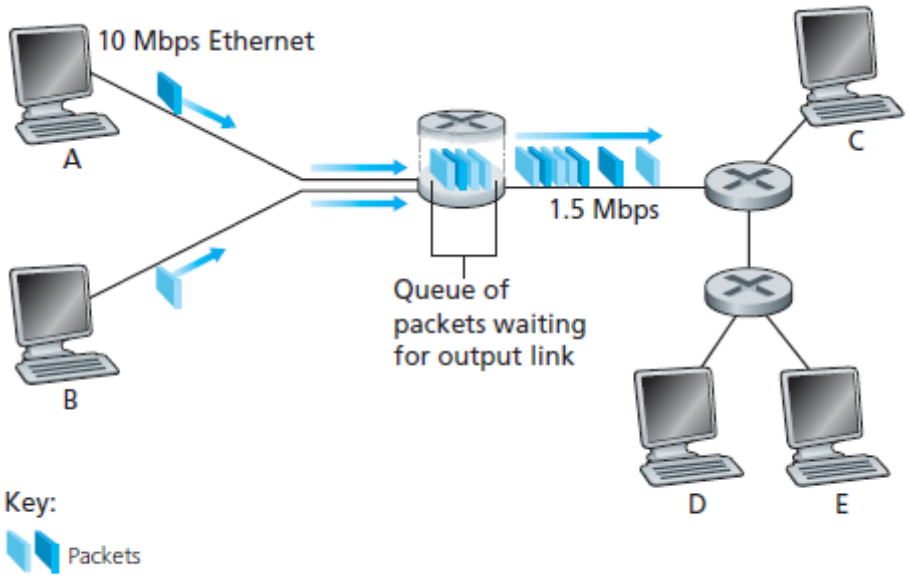


Figure 1.12 ♦ Packet switching

the arriving packet or the one of the already-queued packets will be dropped.

Forwarding Tables and Routing Protocols

Each router has a **forwarding table** that **maps destination addresses to that router's outbound links**. When a packet arrives at a router, the router examines the address and searches its forward table, using this destination address (or portions of the destination addresses), to find the appropriate outbound link. Routing protocols are used to automatically set the forwarding tables, it may be used to determine the shortest path from each router to each destination and use the shortest path results to configure the forwarding tables.

Circuit Switching

There are two approaches to moving data through a network of links and switches, **circuit switching** and **packet 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. In packet-switched networks, these resources are not reserved. When a *bona fide* connection occurs, the sender and receiver maintain connection state for that connection. This connection is called a **circuit** and a constant transmission rate in the network's links are reserved, meaning data is transferred at the guaranteed constant rate.

Frequency-division Multiplexing (FDM) & Time-division Multiplexing (TDM)

With **FDM**, the **frequency spectrum of a link is divided up among the connections** established across the link. The link dedicates a frequency band to each connection for the duration of the connection.

For a **TDM link**, **time is divided into frames of fixed duration**, and each **frame** is **divided** into a **fixed number of time slots**. When the network establishes a connection across a link, the network dedicates one time slot in every frame to this connection.

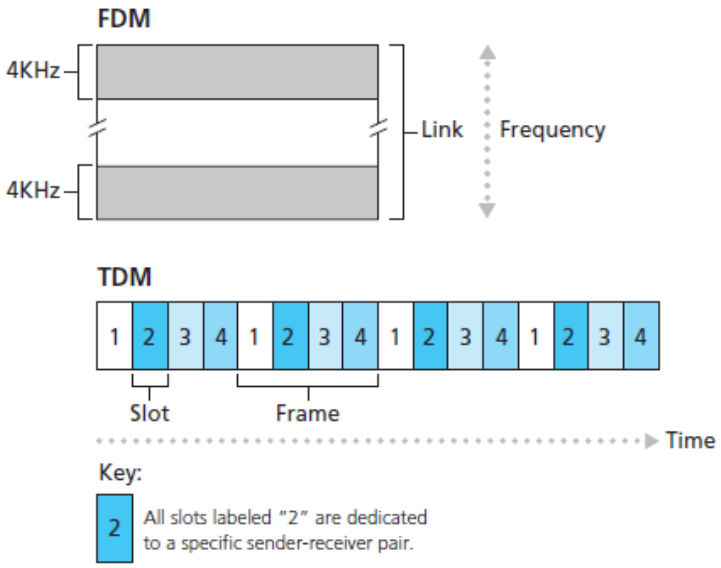


Figure 1.14 ♦ With FDM, each circuit continuously gets a fraction of the bandwidth. With TDM, each circuit gets all of the bandwidth periodically during brief intervals of time (that is, during slots)

Packet switching vs. Circuit Switching

Example:

- 1 Mb/s link
- Each user:
  - » 100kb/s when "active"
  - » Active 10% of time

Circuit Switching

- 10 users

Packet Switching

- 35 users, probability of 10 active at same time is less than 0.0004%

Probability that the event A will occur X times

$$A = \binom{n}{x} p^x (1-p)^{n-x}$$

$$P = 0.1, X = 10, P(X > 10), N = 35$$

Solve,

$$A = 1 - P(X \leq 10)$$

Network of Networks

A naive approach is to have each access ISP directly connect with every other access ISP, however this is much too costly for the access ISPs, as it would require each access ISP to have a separate communication link to each of the other access ISPs all over the world.

Our first network structure, Network Structure 1, interconnects all of the access ISPs with a single global transit ISP. The global transit ISP is a network of routers and communication links that not only spans the globe but at least one touter near each access ISP. The access ISP pays the global transit ISP, the **access ISP** is said to be a **customer** and **global transit ISP** is said to be a **provider**.

The second network structure, Network Structure 2, consists of hundreds of thousands of access ISPs and **multiple** global transit ISPs. In any given region, there may be a **regional ISP** to which the access ISPs in the region connect. To build a network that more closely resembles today's Internet, we must add **points of presence (PoP), multi-homing, peering** and the **internet exchange points (IXPs)**.

A **PoP**, is a group of one or more routers (at the same location) in the provider's network where customer ISPs can connect into the provider ISP. Any ISP may choose to **multi-home**, which is to connect to two or more provider ISPs. This way, ISP can continue to send and receive packets into the Internet even if one of its providers has a failure. To **reduce the costs of traffic exchange** with the provider, **ISPs at the same level of the hierarchy can peer**, which is **directly connecting** their **networks** together so that all the **traffic** between them **passes over** the **direct connection** rather than through upstream intermediaries. Companies can also create an **IXP** which is a meeting point where multiple ISPs can peer together.

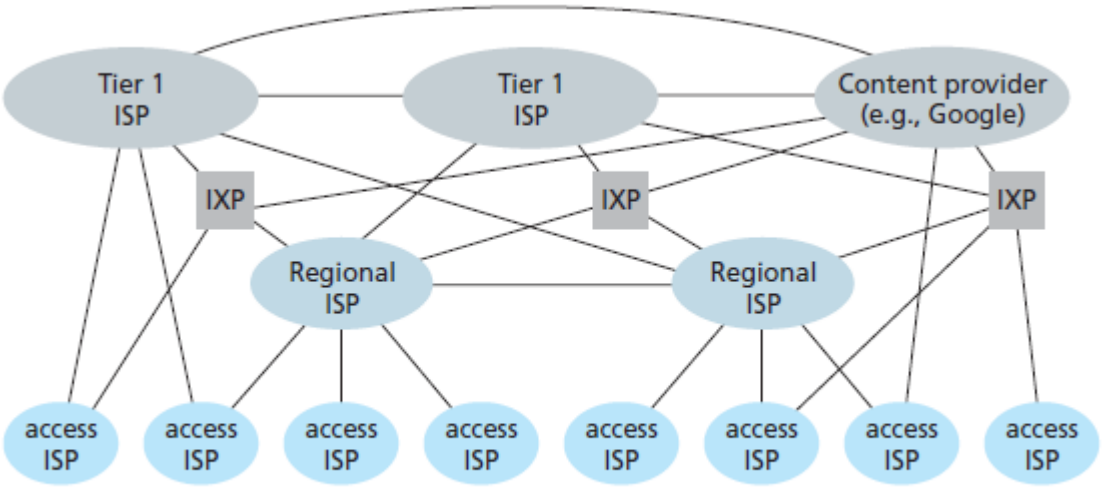


Figure 1.15 ♦ Interconnection of ISPs

1.4 Delay, Loss, and Throughput

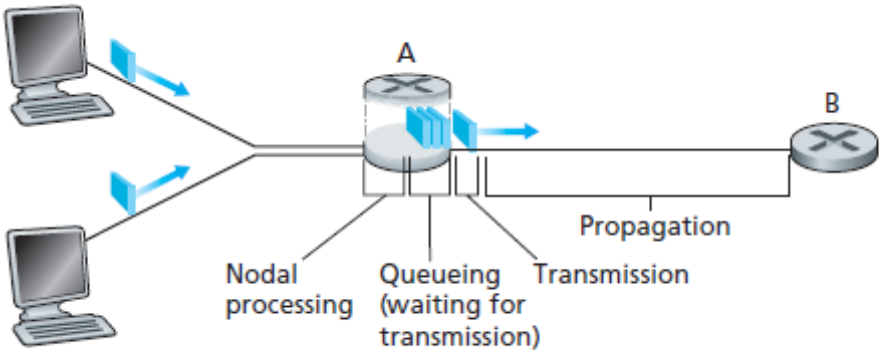


Figure 1.16 ♦ The nodal delay at router A

Nodal Processing Delay

This is the time required to examine the packet's header and determine where to direct the packet. This may also include other factors, such as time needed to check for bit-level errors. This delay is usually microseconds or less.

Queuing Delay

This delay occurs as it waits to be transmitted onto the link. This is dependent on the number on earlier arriving packets that are queued and waiting for transmission. If the queue is empty, then the delay is zero.

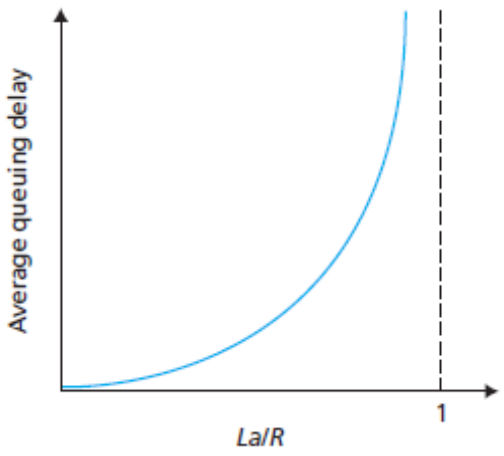
Transmission Delay

Packets can only be transmitted only after all the packets that have arrived before it has been transmitted.

L = Packet Length (bits)
R = Link Bandwidth (bps)
a = average packet arrival rate

La/R ~ 0 → average queuing delay small
La/R ≤ 1 → average queuing delay large
La/R >1 → average delay infinite

Delay = L/R



Propagation Delay

This delay is the time required to propagate from the beginning of the link to router B.

D = length of psychical link

S = propagation speed in medium

Delay = D/S

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

Packet Loss

The queue capacity is finite, so packet delays do not really approach infinity as the diagram shows. When a packet arrives to a full queue, with no place to store the packet, a ro**uter will drop that packet**, and the packet will be lost. From the end system viewpoint a packet loss will look like a packet having been transmitted into the network core but never emerging from the network at the destination.

Throughput

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

- **Instantaneous throughout** is the rate (in bits/sec) at which Host B is receiving the file at any given point in time
- **Average throughput** is the rate at which Host B takes to receive all the file.

1.5 Protocol Layers

A protocol layer can be implemented in software, in hardware or in a combination of the two. Application-layer protocols such as HTTP and SMTP are almost always implemented in software in the end systems, so are transport-layer protocols.

The advantages are:

- Provides structured way to discuss system components
- Modularity makes it easier to update system components

The disadvantages are:

- One layer may duplicate lower-layer functionality
- Functionality at one layer may need information that is present only in another layer

When taken together, the protocols of the various layers are called the protocol stack which are comprised of:

- **Application Layer**: supporting network applications (HTTP, SMTP,FTP)
- **Transport Layer**: host-host data transfer (TCP, UDP)
- **Network Layer**: routing of datagram's from source-destination (IP, Routing Protocols)
- **Link Layer**: data transfer between neighbouring network elements (PPP, Ethernet)
- **Physical Layer**: bits "on the wire" (twisted-pair copper wire, single mode fiber optics)

Application Layer

2.1 Principles of Network Applications

When developing a new application, software needs to written in a way that it will run on multiple end systems. The application architecture is designed by the application developer and dictates how the application is structured over the various end systems.

Client-server architecture

Server

- Always-on host
- Permanent IP address
- Server farms for scaling

Clients

- Communicate with server
- May be intermittently connected
- May have dynamic IP addresses
- Do not communicate directly with each other

Peer-to-peer P2P architecture

- Minimal (or no) reliance on a server
- Arbitrary end systems directly communicate
- Peers are intermittently connected and change IP addresses
- Highly scalable but difficult to manage

Hybrid of client-server and P2P

Skype

- VoIP P2P application
- Servers are used to track the IP addresses of users
- User-to-user messages are sent directly between user hosts

Process Communicating

A **process is a program that is running within an end system**. When **processes** are **running** on the **same end system**, they can **communicate** with each other with **interproccess communication**, using **rules** that are governed by the end system's **operating system**. A sending process creates and sends messages into the network; a receiving process receives these messages and possibly responds by sending messages back.

The process that **initiates the communication** is labelled as the **client**, and **the process that waits** to be contacted **to begin** the **session** is the **server**.

Socket

Any message sent from one process to another must go through the underlying network through a software called a socket. A socket is the interface between the application layer and the transport layer within a host. It is referred to as the **Application Programming Interface (API)** between the

application and the network. The only control the application developer has on the transport-layer side is:

1. The choice of transport protocol
2. Perhaps the ability to fix a few transport-layer parameters (max buffer, max segment sizes)

Addressing Processes

To identify the receiving process, two pieces of information are needed:

1. The address of the host
2. An identifier that specifies the receiving process in the destination host

In the Internet, the host is indentified by its **IP address** which uniquely identifies the host. An identifier is needed because in general a host could be running many network applications at one time which is usually referred to as a **port number** i.e. HTTP is identified by port number 80 & SMTP is identified by port number 25.

Transport service for Applications

Data loss

- Can be acceptable for loss-tolerant applications (conversational audio/video)
- Other apps require 100% reliable data transfer (file transfer, telnet, electronic mail)

Timing

- Some apps require low delay to be "effective" (internet telephony, interactive games)

Bandwidth(throughput)

- Applications that have throughout requirements are said to be bandwidth-sensitive (internet telegraphony application encodes voice at 32 kbps)
- Elastic applications make use of as much, or as little throughput happens to be available (file transfer, electronic mail)

Security

- Encrypt all data, data integrity

Application	Data Loss	Throughput	Time-Sensitive
File transfer/download	No loss	Elastic	No
E-mail	No loss	Elastic	No
Web documents	No loss	Elastic (few kbps)	No
Internet telephony/ Video conferencing	Loss-tolerant	Audio: few kbps–1 Mbps Video: 10 kbps–5 Mbps	Yes: 100s of msec
Streaming stored audio/video	Loss-tolerant	Same as above	Yes: few seconds
Interactive games	Loss-tolerant	Few kbps–10 kbps	Yes: 100s of msec
Instant messaging	No loss	Elastic	Yes and no

Figure 2.4 ♦ Requirements of selected network applications

Internet transport protocol services

Transmission Control Protocol Service

- **Connection-oriented service**: setup required between client and server processes
- **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 throughout guarantees, security

User Datagram Protocol Service

- unreliable transfer between sending and receiving process
- does not provide: connection setup, reliability, flow control, congestion control, timing, throughout guarantee or security

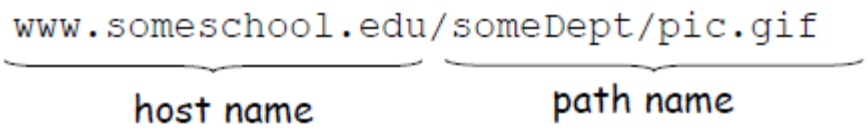
A benefit of UDP over TCP is that since it does not need to retransmit lost packets nor does it do any connection setup, sending data incurs less delay. This lower delay makes UDP an appealing choice for delay-sensitive applications like audio and video. UDP is generally used when fast connection matters over reliability.

Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP [RFC 5321]	TCP
Remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
File transfer	FTP [RFC 959]	TCP
Streaming multimedia	HTTP (e.g., YouTube)	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary (e.g., Skype)	UDP or TCP

Figure 2.5 ♦ Popular Internet applications, their application-layer protocols, and their underlying transport protocols

2.2 Web and HTTP

The **HyperText Transfer Protocol (HTTP)**, the web's application-layer protocol, is implemented in two programs: a client program and a server program. A web page consists of objects which is simple a file such as an HTML file, a JPEG image etc, that is addressable by a single URL.



- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068

HTTP uses TCP as its underlying transport protocol:

- Client initiates TCP connection (creates socket) to server, port 80
- Server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and web server (HTTP server)
- TCP connection closed

The **HTTP server maintains no information about** the **clients** therefore it is said to be a **stateless protocol**.

HTTP connections

Non-persistent HTTP

- At most one object is sent over a TCP connection
- HTTP/1.0 uses non-persistent HTTP

Persistent HTTP

- Multiple objects can be sent over single TCP connection between client and server
- HTTP/1.1 uses persistent connections in default mode

Response time modelling

Round-trip time (RTT) is the **time** to **send a small packet to travel from client to server and back**. The RTT includes packet-propagation delays, packet-queuing delays and packet-processing delays.

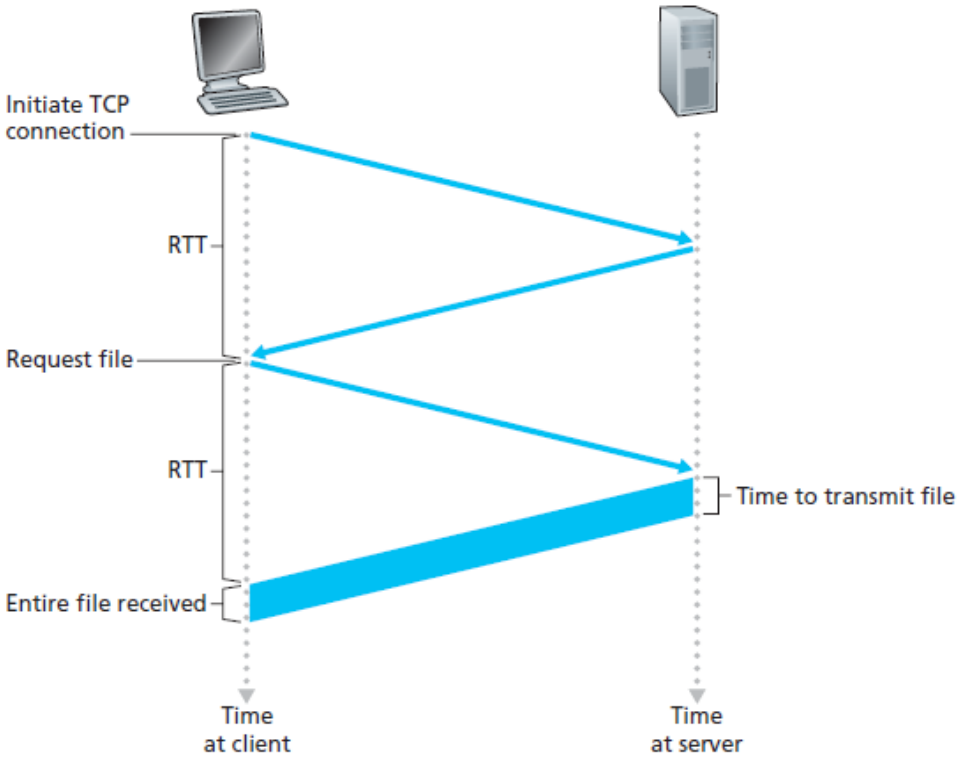
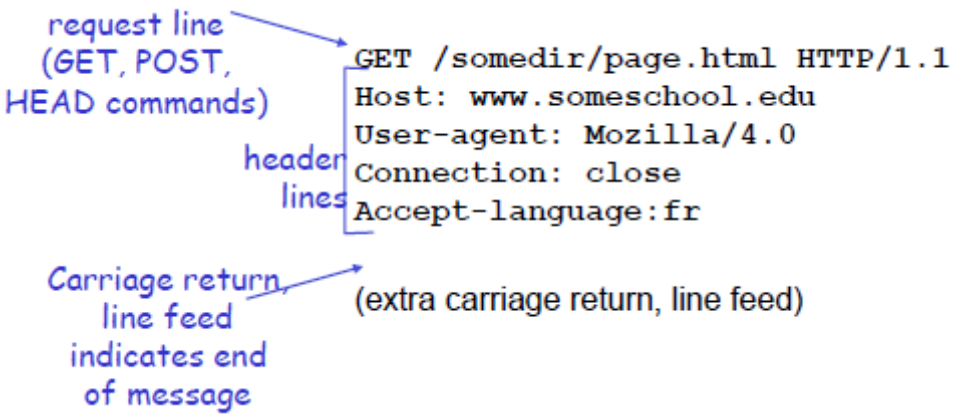


Figure 2.7 ♦ Back-of-the-envelope calculation for the time needed to request and receive an HTML file

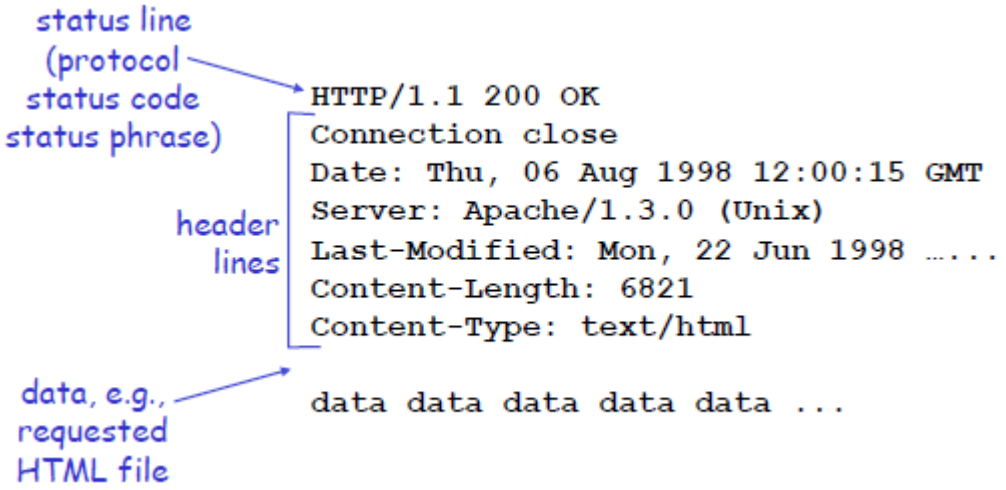
File transmission time = 2 x RTT + transmission time at the server of the HTTML file

HTTP message

Request



Response



Status Codes

- **200 OK**
  - » Request succeeded, request object later in this message
- **301 Moved Permanently**
  - » Requested object moved, new location specified later in this message (Location:)
- **400 Bad Request**
  - » Request message not understood by server
- **404 Not Found**
  - » Request document not found on this server
- **505 HTTP Version No Supported**

HTTP also uses cookies to keep track of users. When a request comes onto a web server, the server creates a unique identification number and creates an entry in its back-end database that is indexed by the identification number.

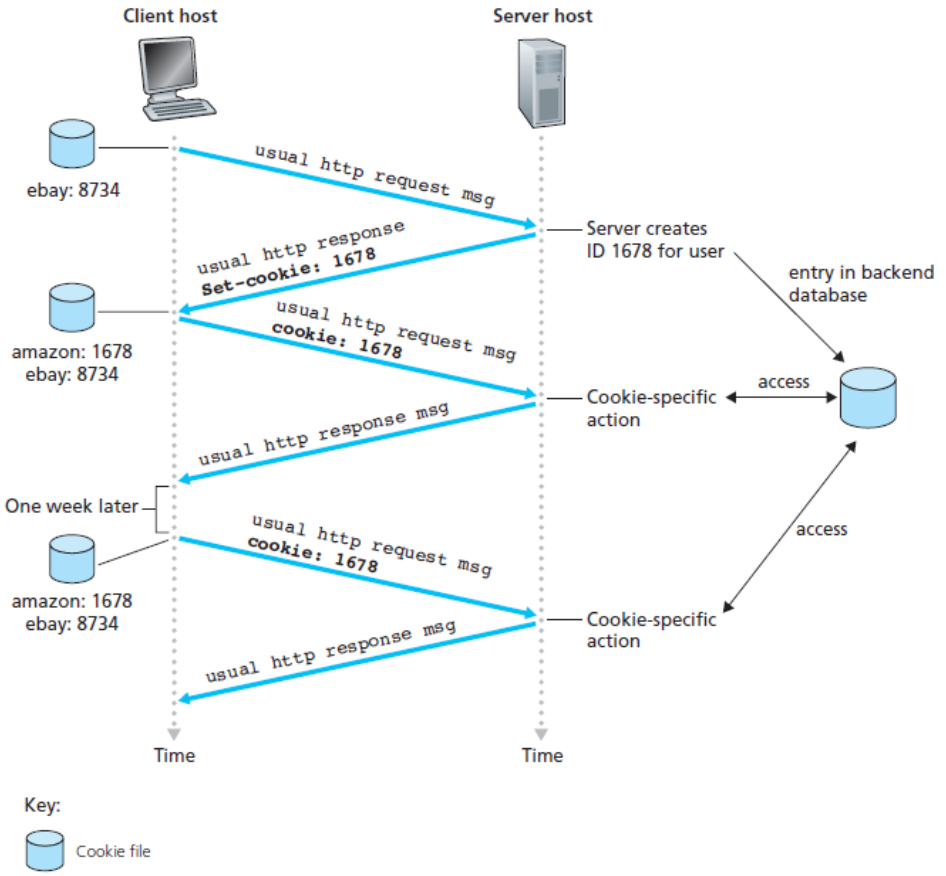


Figure 2.10 ♦ Keeping user state with cookies

Web Caching

A **web cache**, also known as a **proxy server is a network entity that satisfies client request without involving origin server**.

1. User sets browser: web accesses via cache
2. Browser sends all HTTP requests to cache
    a. If object in cache, cache returns object
    b. Else cache requests object from origin server, then returns object to client

The cache acts as both client and server and typically it is purchased and **installed by the ISPs**. There are many advantages of web caching:

- Reduce response time for client request
- Reduce traffic on an institution's access link
- Reduce web traffic in the internet as a whole enabling "poor" content providers to effectively deliver content (but so does P2P file sharing)

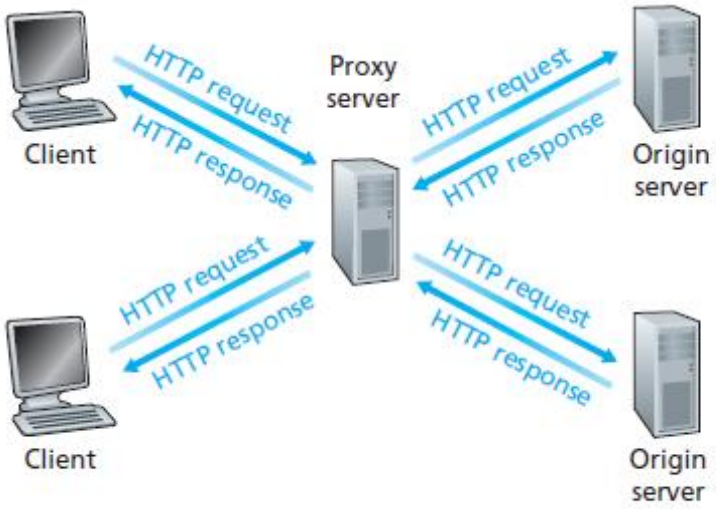


Figure 2.11 ♦ Clients requesting objects through a Web cache

Conditional GET

The goal of this is to **not send object if cache has up-to-date cached version**. The cache can **specify date** of cached copy in HTTP request (If-modified-since: <date>). The server response will contain no **object if cached copy is up-to-date** (HTTP/1.0 304 Not Modified).

Content Distribution Networks (CDN)

CDN deploys caches all over the world which are **connected via the internet** or **dedicated networks**. If a website employs the service of a CDN, the **CDN stores the popular content of the website** in its **caches**. A **request** is then **directed** to the **closest CDN caches**. Delay can be high for far away customers, so by use of CDN they can be closer to the clients that request the data, rather than clients having to get it from the origin server (can be in a distant continent).

2.4 Electronic Mail

The three major components are:

- **User agents**
  - » a.k.a "mail reader"
  - » composing, editing, reading mail messages
  - » e.g. Outlook, Thunderbird, Mail,
  - » outgoing, incoming messages stored on server
- **Mail servers**
  - » Mailbox contains incoming messages for user
  - » Message queue of outgoing (to be sent) mail messages
  - » SMTP between mail servers to send email messages
    - Client: sending mail server
    - "Server": receiving mail server
- **Simple mail transfer protocol SMTP**
  - » Uses TCP to reliably transfer email message from client to server (port 25)
  - » Direct transfer: sending server to receiving server
  - » Three phases of transfer
    - Handshaking (greeting)

        - Transfer of messages
        - Closure
    - Command/response interaction
        - Commands: ASCII text
        - Response: status code and phrase
    - Messages must be in 7-bit ASCII

2.5 Domain Name System (DNS)

This is a **directory server that translate hostnames to IP addresses**. It is a **distributed database implemented in hierarchy** or **many name servers** and it is also an **application-layer protocol** that **allows hosts** to **query** the **distributed database**. It is employed by other application-layer protocols, including HTTP, SMTP and FTP, to translate user-supplied hostnames to IP addresses. Overall the DNS services include:

- **Hostname to IP address translation**
- **Host aliasing**
    - Canonical and alias names
    - CN: relay1.sydney.wifi.com.au
    - Alias: wifi.com.au
- **Mail server Aliasing**
    - MS: relay1.sydney.hotmail.com
    - Alias: hotmail.com
- **Load Distribution**
    - Replicated web servers are set of IP addresses for one canonical name
        - Rotation policy (can be used for email so that multiple mail servers can have one same alias name)
    - CDN use IP address of requesting host to find best suitable server
        - Closest, least-loaded, etc

DNS Example

A simple design for DNS would be a centralised system however there are many problems:

- **Single point of failure**, if the DNS server crashes, so does the Internet
- **Traffic volume**, single DNS server would have to handle all DNS queries from hundreds of millions of hosts
- **Distant centralised database**, a single DNS server cannot be close to all querying clients
- **Maintenance**, not only would the database be huge, it would be need to be updated frequently to account for every new host

Distributed, Hierarchical Database

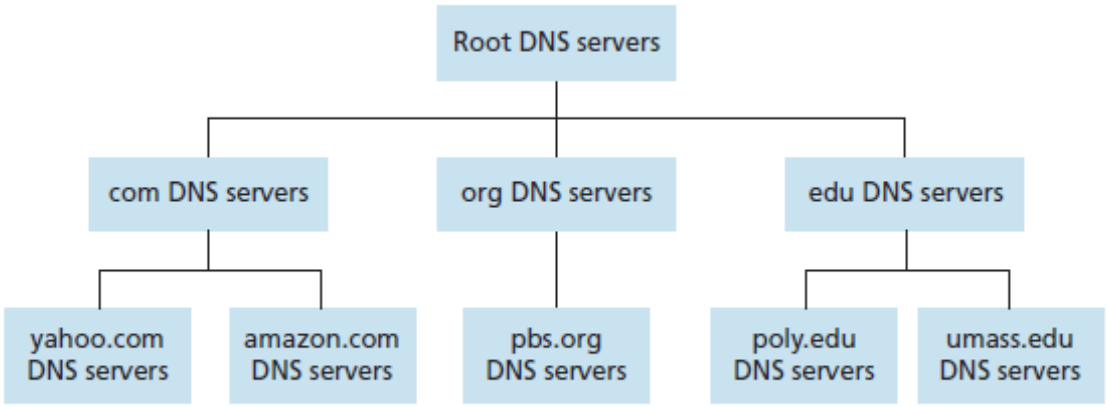
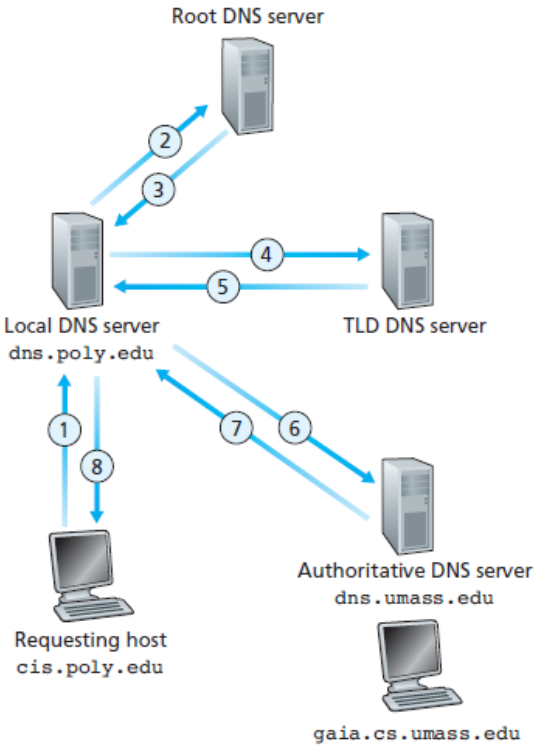


Figure 2.19 ♦ Portion of the hierarchy of DNS servers

The **three classes of DNS servers**:

- **Root DNS servers**
  - » Contacted by local name server that cannot resolve name
    1. Contacts authoritative name server if name mapping is not know
    2. Get mapping
    3. Returns mapping to local name server
- **Top-level domain (TLD) servers**
  - » These are responsible for com, org, net, edu, etc and all top-level country domains au, uk, fr, ca, jp.
  - » The company Educause maintains the TLD servers for the edu top-level domain
- **Authoritative DNS servers (Local Name Server)**
  - » Does not strictly belong to hierarchy
  - » Each ISP has one (also called **default name server**)
  - » When host makes a DNS query, query is sent to its local DNS server
    - Acts as a proxy, forwards query into hierarchy

Iterated Query



Recursive Query

- Puts burden of name resolution on contacted name server
- Heavy load
- Typically recursive to local DNS and then iterative

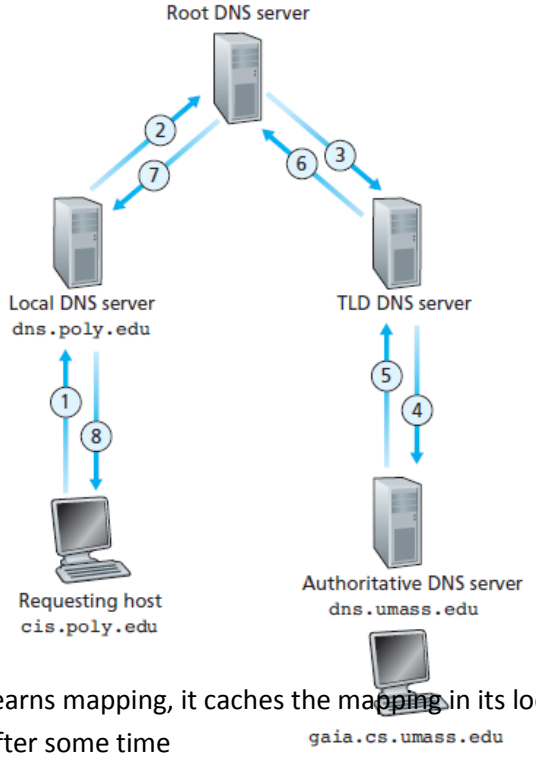


Figure 2.22 ♦ Recursive queries in DNS

DNS Caching

- Once any name server learns mapping, it caches the mapping in its local memory
- Cache entries timeout after some time
- TLD servers typically cached in local name servers, thereby allowing the local DNS server to bypass the root DNS servers
- TTL field in DNS record indicates how long to cache
- When passing DNS record to another name server, TTL is updated = (original TTL – time record cached for), which is often set to 2 days

DNS Records and Messages

The DNS servers that implement the DNS distribute database store **resource records (RRs)** which provide hostname-to-IP address mappings.

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

TTL is the time to live of the resource record (lifetime of the cache). The meaning of Name and Value depend on the Type:

- **Type=A**
  - » **Name**: hostname
  - » **Value**: IP address
- **Type = NS**
  - » **Name**: domain (e.g. foo.com)
  - » **Value**: hostname of authoritative name server for this domain (e.g. dns.foo.com)
- **Type= CNAME**
  - » **Name**: alias name for some "canonical" (the real) name

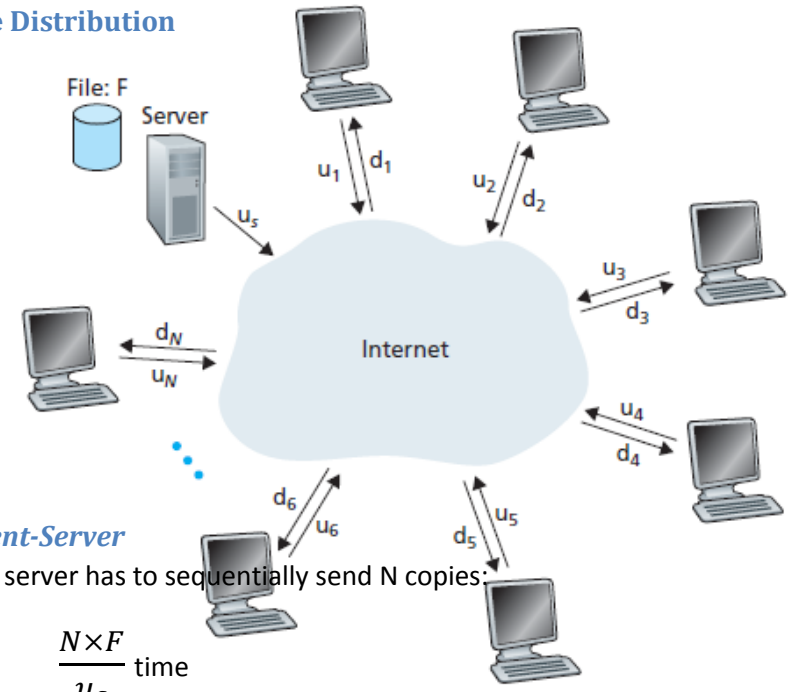
» **Value**: canonical name

- **Type = MX**
  - » **Name**: alias name of mail server
  - » **Value**: canonical name of mail server\

2.6 P2P file sharing

There is **minimal** (or no) **reliance** on **always-on infrastructure servers**. Pairs of intermittently connected hosts, called peers, communicate directly with each other. These peers are usually desktops and laptops controlled by users.

File Distribution



F = file size (bits)

u_s = server upload bandwidth

u_i = peer i upload bandwidth

d_i = peer i download bandwidth

Client-Server

The server has to sequentially send N copies:

$\frac{N \times F}{u_s}$ time

Figure 2.24 ♦ An illustrative file distribution problem

Client i takes $\frac{F}{d_i}$ time to download

Time to distribute F to N clients: $d_{cs} \geq \max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$ increases linearly in N

P2P

The server has to sequentially send N copies:

$\frac{F}{u_s}$ time

Client i takes $\frac{F}{d_i}$ time to download

Fastest possible upload rate: $d_{cs} \geq \max\left\{\frac{F}{u_s}, \frac{F}{\mathrm{d_{min}}}, \frac{NF}{(u_s + \sum_{i=1}^{N} u_i)}\right\}$

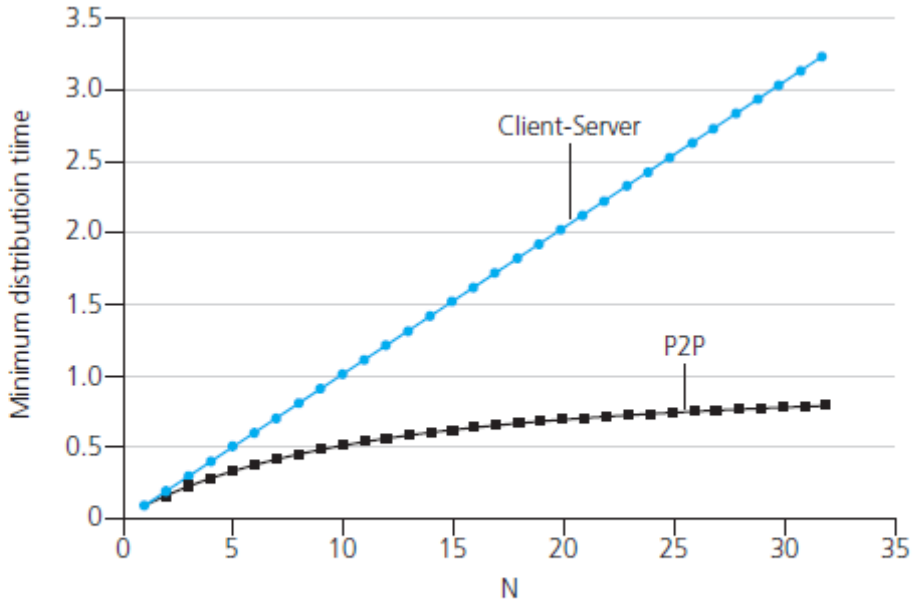


Figure 2.25 ♦ Distribution time for P2P and client-server architectures

BitTorrent

A **torrent** is a **group of peers exchanging chunks of a file**. There are trackers which tracks peers participating in the torrent. The file is divided into 256KB chunks. When a peer joins a torrent, they have no chunks, but will accumulate them over time. While downloading, peer uploads chunks to other peers. At any given time, different peers have different subsets of file chunks, periodically a peer asks each neighbour for list of chinks that they have. A peer sends requests for their missing chunks (rarest first). **Peers send chunks to four neighbours sending it data** to them at the **highest rate**. Every 10 seconds, the rates are revaluated. These **four peers are referred to be unchoked.** Additionally every 30 seconds, one additional neighbour is picked at random, and chunks are sent, in which they are referred to be **optimistically unchoked**. Then that peer will end up being the peers top four pick anyways, because more chunks are being exchanged between them.

Transport Layer

3.1 Transport-Layer Services

A **transport-layer protocol provides for logical communication between application process running** on **different hosts**. Transport protocols run in end systems but not in network routers:

- **Sending side**
  - » Transport layer converts the application-layer messages it receives from a sending application process into transport-layer packets (**transport-layer segments**) to network layer
- **Receiving side**
  - » Network layer extracts the transport-layer segment from the datagram up to the transport layer

- The transport layer then processes the segments into messages and then passes to application layer

There may be **more than one transport-layer protocol** available to network applications, such as **TCP** and **UDP**.

Transport Layer (the kids who distribute letters)
- Logical communication between **processes** (children)

Network Layer (the courier service)
- Logical communications between **hosts** (houses)

3.2 Multiplexing and Demultiplexing

Extending host-host delivery service provided by the network layer **to process-to-process delivery service** for applications running on the hosts is called transport-layer **multiplexing** and **demultiplexing**.

The job of **delivering the data** in a transport layer segment to **the correct socket** is called **demultiplexing**. The job of **gathering data chunks** at the source host **from different sockets**, encapsulating each data chunk with header information to create segments, and **passing to the network layer** is calling **multiplexing**.

Demultiplexing

- **Host receives IP datagram's**
  - Each datagram has a source IP address, destination IP address
  - Each datagram carries on transport-layer segment
  - Each segment has source, destination port number
- Host **uses IP addresses** & **port numbers** to **direct segment** to **appropriate socket**

Connectionless demultiplexing

- **Recall**: created socket has host-local port #
  - DatagramSocket mySocket1 = new DatagramSocket(12534)
- **Recall**: when creating datagram to send into UDP socket, the destination IP address and destination port # must be specified
- When host receives UDP segment
  - Checks destination port # in segment
  - Directs UDP segment to socket with that port #
- IP datagram's with **same destination port #,** but different source IP addresses and/or source port numbers will be directed to the **same socket at destination.**

Connection-oriented demultiplexing

- TCP socket identified by 4-tuple
  - Source IP address
  - Source port #
  - Destination IP address
  - Destination port #
- Uses all four values to direct segment to appropriate socket
- Server host may support many simultaneous TCP sockets
  - Each socket identified by its own 4-tuple

- Web servers have different sockets for each connecting client
    - » Non-persistent HTTP will have different socket for each request

3.3 Connectionless transport: UDP

UDP is a "no-frills", "bare-bones" transport protocol and the segments may be lost or delivered out-of-order to application. UDP is said to be **connectionless** because there is **no handshaking** between UDP sending and the receiver. Each UDP segment is **handled independently** of others as well. The advantages of UDP over TCP are:

- **Finer application-level control over what data is sent, and when**
    - » Package the data inside UDP segment and immediately pass to network layer, has no congestion-control mechanism (UDP much faster)
    - » Real-time applications often require a minimum sending rate, do not want overly delay segment transmission (loss tolerant, rate sensitive)
- **No connection establishment**
    - » DNS runs on UDP because there is no delay, however HTTP uses TCP rather than UDP because it requires reliability
- **No connection state**
    - » Does not track buffers, congestion-control etc
    - » Can support many more active clients
- **Small packet header overhead**
    - » TCP has 20 bytes vs. UDP has 8 bytes

UDP checksum

The UDP **checksum provides for error detection**. It is used to determine whether bits within the UDP segment have been altered as it moved from source to destination.

- **Sender**
    - » Treat segment contents, including header fields, as sequence of 16-bit integers
    - » Checksum: addition (one's complement sum) of segment contents
    - » Sender puts checksum value into UDP checksum field
- **Receiver**
    - » Computer checksum of received segment
    - » Check if computed checksum equals checksum field value:
        - NO – error detected
        - YES – no error detected (but maybe errors nonetheless)
    - » No action on error detection by UDP

3.4 Principles of Reliable Data Transfer

- Characteristics of unreliable channel will determine complexity of **reliable data transfer protocol (RDT)**

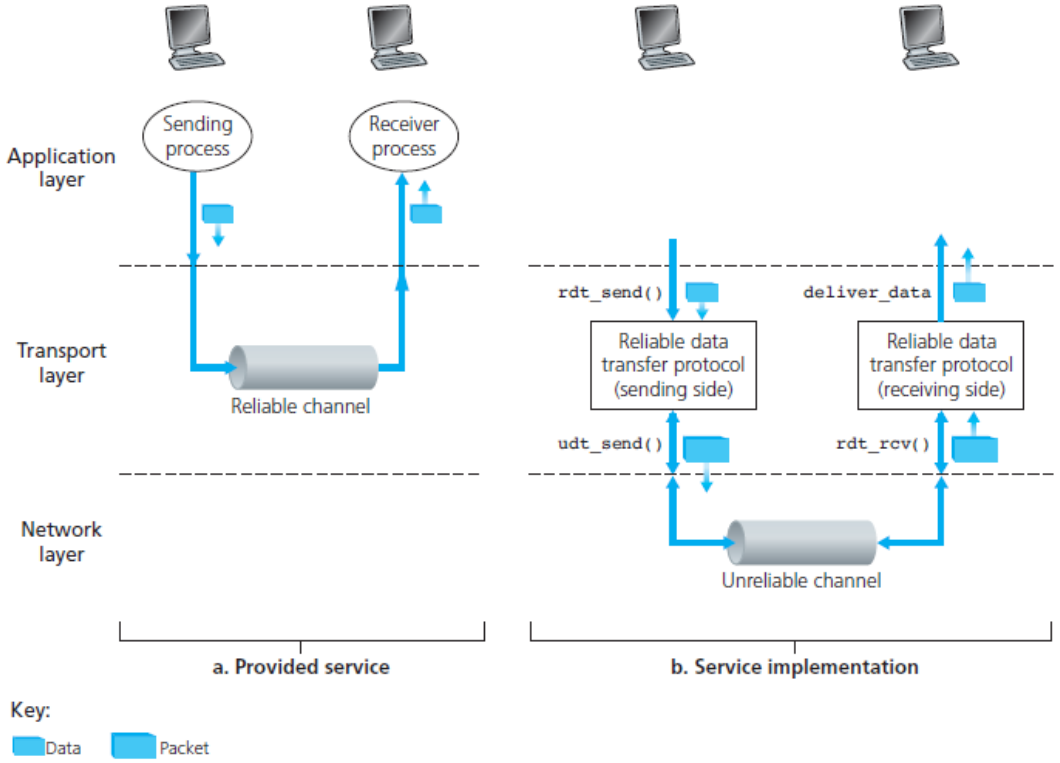


Figure 3.8 ♦ Reliable data transfer: Service model and service implementation

rdt1.0: Reliable Data Transfer over a Perfectly Reliable Channel

- Underlying **channel perfectly reliable**
  - » No bit errors
  - » No loss of packets
- Separate **finite-state machine (FSMs)** for sender & receiver
  - » **Sender** sends data into underlying channel
  - » **Receiver** reads data from underlying channel

rdt2.0: Reliable Data Transfer over a Channel with Bit Errors

- Underlying **channel may flip bits in packet**
  - » Checksum to detect bit errors
- Recovering from errors
  - » **Acknowledgements (ACKs)**, receiver explicitly tells sender that **packet received is OK**
  - » **Negative acknowledgements (NAKs)**, receiver explicitly tells sender that **packet had errors**
  - » Sender **retransmits packet** on receipt of **NAK**
- New mechanisms in rdt2.0
  - » Error detection
  - » Feedback, control messages (ACK,NAK) from receiver to sender

» a.k.a "**Automatic Repeat reQuest" (ARQ) protocols**

- known as "stop-and-wait" protocols
- **Flaws**
  - » ACK/NAK may be corrupted
    - Sender doesn't know what happened at receiver
    - Cannot just retransmit, possible duplication
      - Sender retransmit current packet if ACK/NAK corrupted
      - Sender adds **sequence number** to each packet
      - Receiver discards duplicate packet

rdt2.1: Reliable Data Transfer over a Channel with Bit Errors Revised

- **Sender**
  - » Sequence # is added to the packet
  - » Two sequence #'s (0,1) will suffice because it is either ACK or NAK
  - » Must check if received ACK/NAK is corrupted
  - » Twice as many states
    - States must "remember" whether "expected" packet should have sequence # of 0 or 1
- **Receiver**
  - » Must check if received packet is duplicate
    - State indicates whether 0 or 1 is expected packet sequence #
    - Note: receive cannot know if its last ACK/NAK received OK at sender

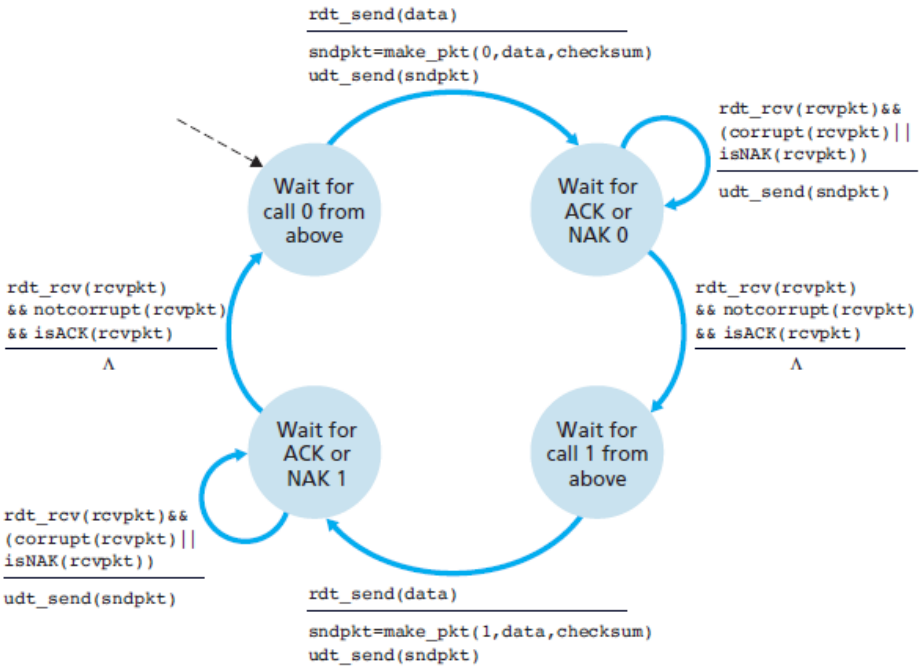


Figure 3.11 ♦ rdt2.1 sender

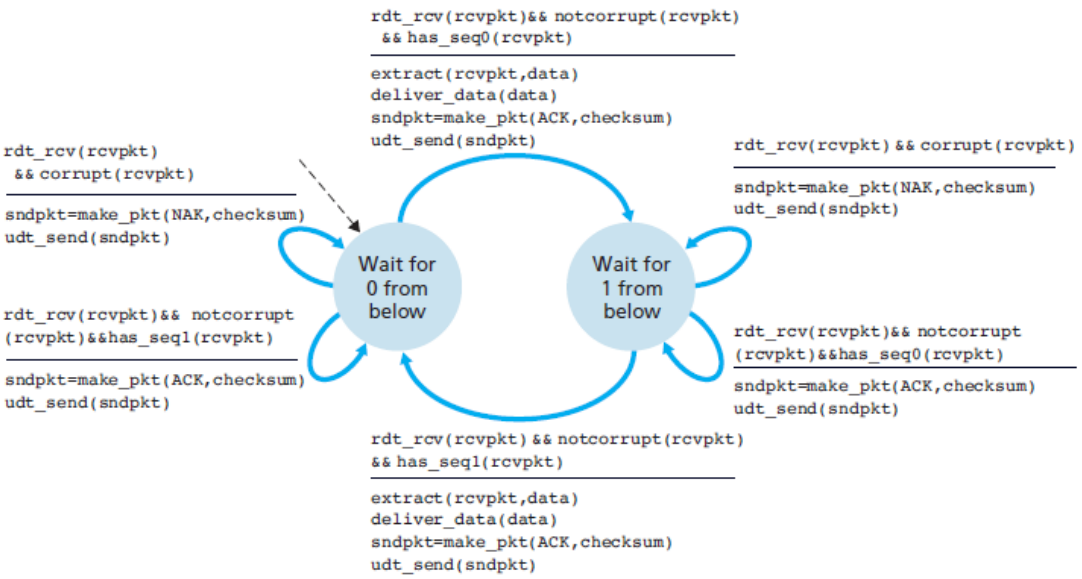


Figure 3.12 ♦ rdt2.1 receiver

rdt2.2: Reliable Data Transfer over a Channel with Bit Errors Revised

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

rdt3.0: Reliable Data Transfer over a Lossy Channel with Bit Errors

Underlying channels can also lose packets (data, ACKs) and although checksum, sequence #, ACKs, retransmissions can help, it will not solve all the issues. A way to solve this issue is to wait a "reasonable" amount of time for ACK:

- Retransmits if no ACK received in this time
- If packet (or ACK) just delayed (not lost)
  - » Retransmission will be duplicate, but sequence #'s already handles this
  - » Receiver must specify sequence # of packet being ACKed
- Requires countdown timer

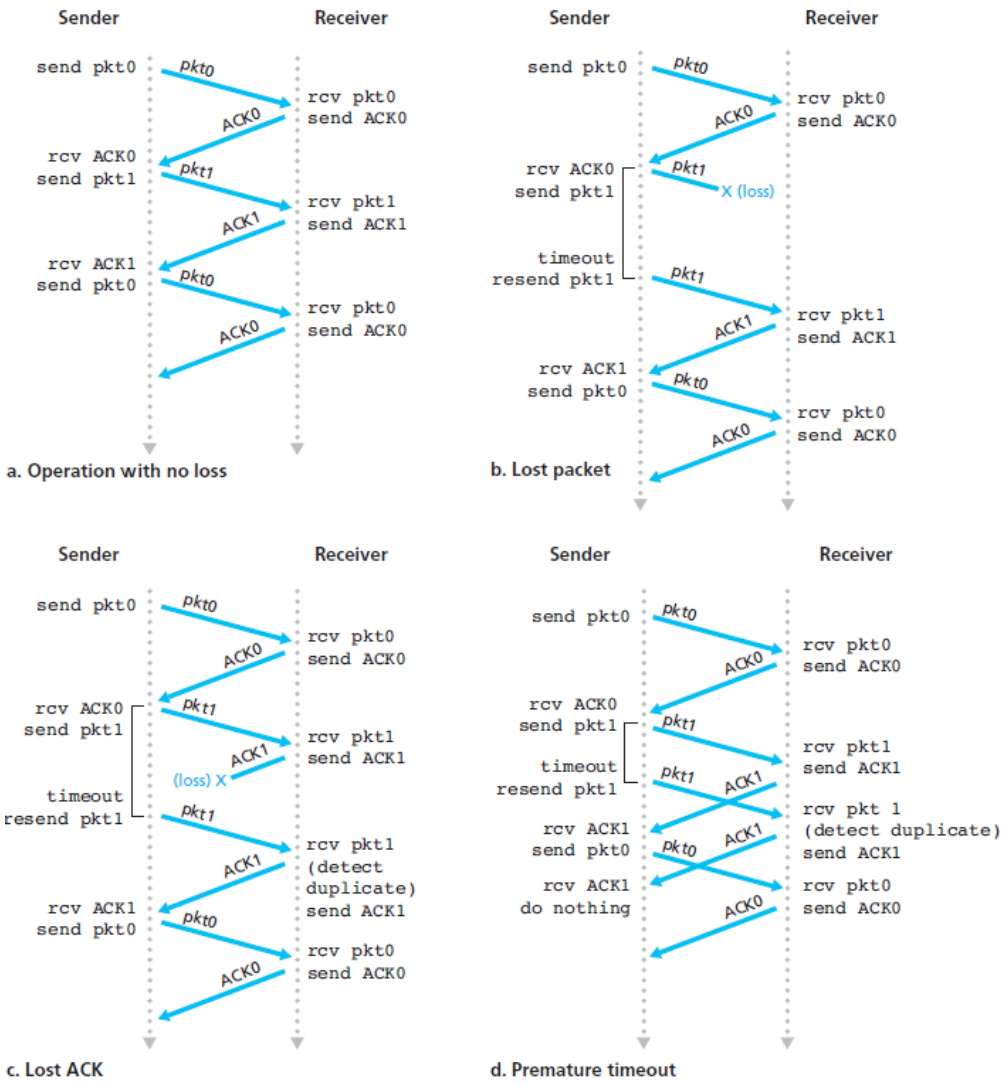


Figure 3.16 ♦ Operation of rdt3.0, the alternating-bit protocol

Pipelined protocols

Pipelining is when the sender allows multiple, "in-flight", yet-to-be-acknowledge packets

- Range of sequence numbers must be increased
- Buffering at sender and/or receiver

Two generic forms of pipelined protocols

- **Go-Back-N**
  - » Sender can have up to N unACKed packets in pipeline
  - » **Receiver** only sends **cumulative ACK**
    - Doesn't ACK packet if there's a gap
  - » Sender has **timer for oldest unACKed packet**
    - When timer expires, **retransmit all unACKed packets**

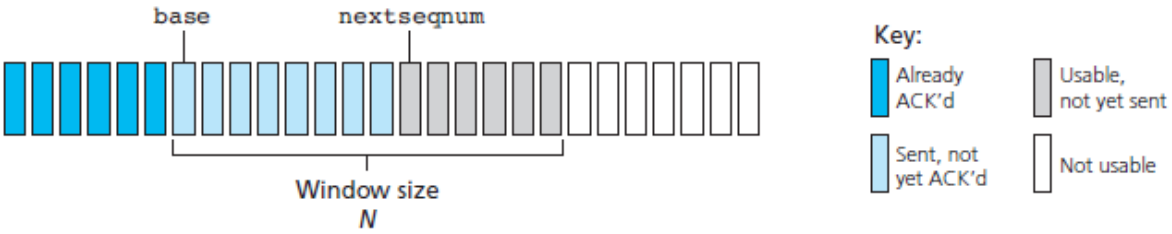


Figure 3.19 ♦ Sender's view of sequence numbers in Go-Back-N

- **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, **retransmit only that unACKed packet**

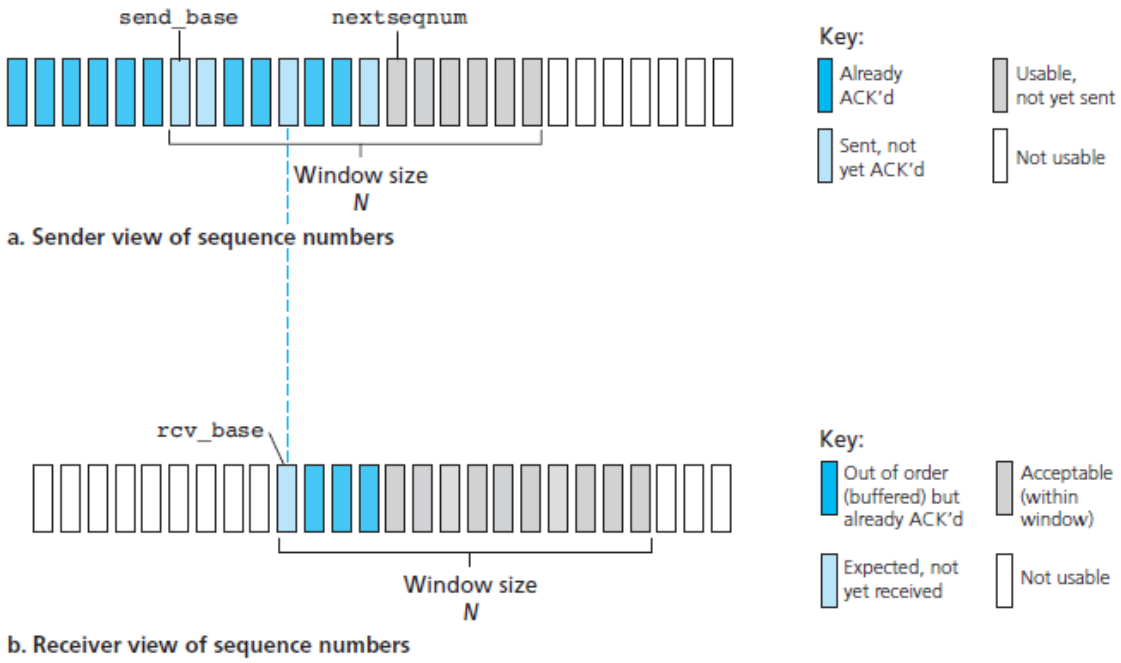


Figure 3.23 ♦ Selective-repeat (SR) sender and receiver views of sequence-number space

3.5 Connection-Oriented Transport: TCP

Segment Structure

- **Point-to-point**
  - » One sender, one receiver
- **Reliable, in-order byte steam**
  - » No "message boundaries"
- **Pipelined**
  - » TCP congestion and flow control set window size
- **Full duplex service**
  - » Bi-directional data flow in same connection (A↔B)
  - » MSS, maximum segment size
- **Connection-oriented**
  - » Handshaking (exchange of control messages) initiates sender & receiver state before data exchange
  - » Send some preliminary segments to each other to establish the parameters
- **Flow controlled**
  - » Sender will not overwhelm receiver

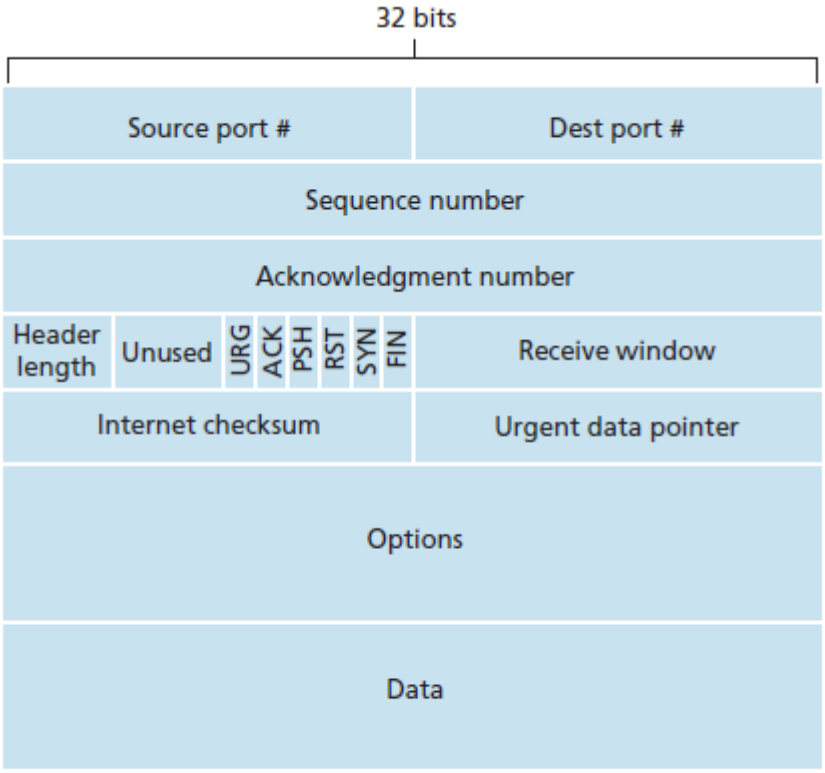


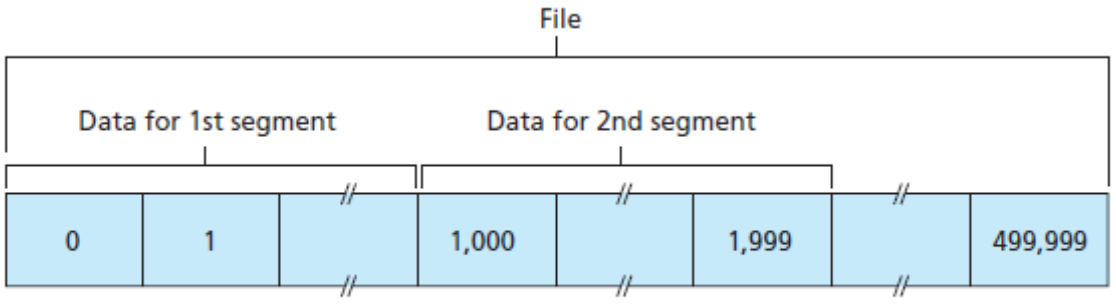
Figure 3.29 ♦ TCP segment structure

As with UDP, the **header** includes the **source port #** & **destination port #** which are used for **multiplexing/demultiplexing** data from/to upper-layer applications. The header also includes a **checksum** field. The **sequence number field** and **acknowledgement number field** are used by the TCP sender and receiver in implementing a **reliable data transfer service**. The **receive window** field is used for flow control. The **header length field** specifies the length of the TCP header in 32 bit words. The optional and variable length **options field** is used when a sender and receiver negotiate the **maximum segment size (MSS)** or as a window scaling factor for use in high-speed networks. The **flag field** contains 6 bits:

- **URG:** indicates whether data is urgent
- **ACK**: indicates the value carried is valid (acknowledgement for segment being successfully received)
- **PSH:** indicates that the receiver should pass the data to the upper layer immediately (push data now)
- **RST, SYN, FIN**: used for connection setup and teardown

Sequence Numbers & Acknowledgements

The **sequence number** for a segment is the **byte-stream number** of the **first byte in the segment**.



Th
by

Figure 3.30 ♦ Dividing file data into TCP segments

the stream, meaning it provides **cumulative acknowledgments**. There are two ways of handling out-of-order segments (TCP RFCs do not impose any rules):

1. Discard out-of-order segments
2. Wait for missing bytes to fill in gaps (most commonly used as it is more efficient on bandwidth)

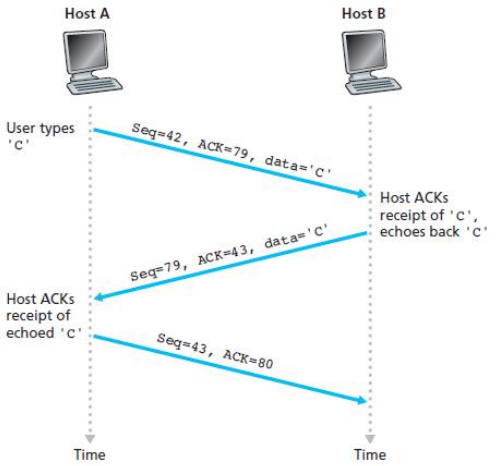


Figure 3.31 ♦ Sequence and acknowledgment numbers for a simple Telnet application over TCP

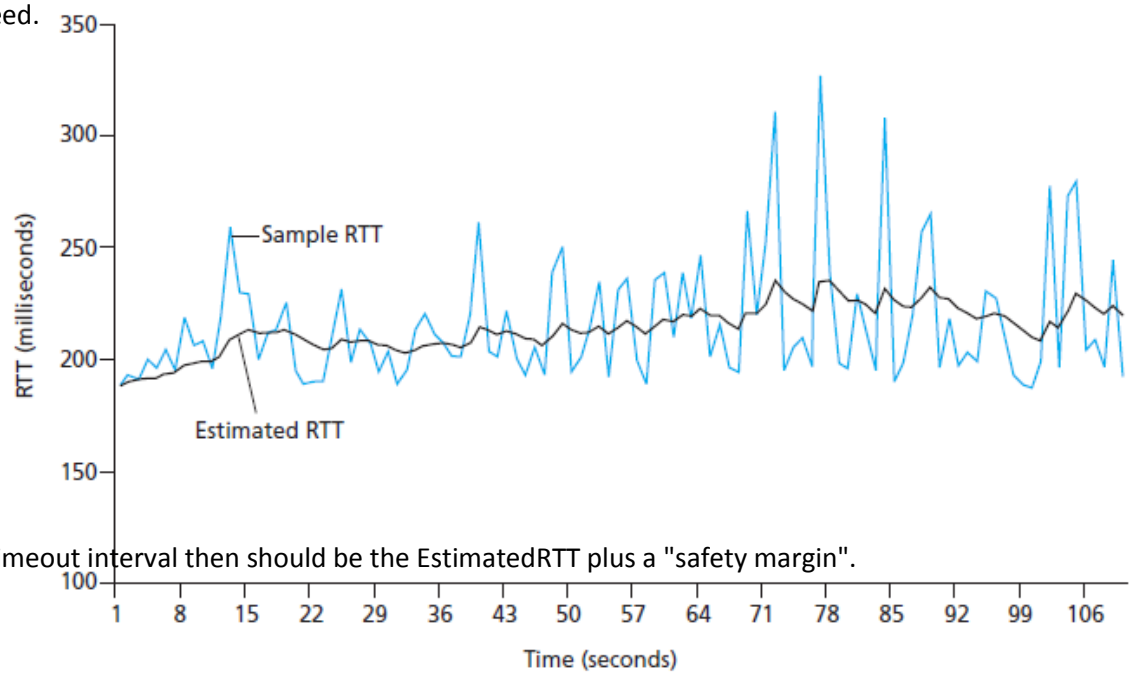
Round-Trip Time (RTT) Estimation and Timeout

Like rdt, TCP uses a timeout/retransmit mechanism to recover the lost segments. Length of the timeout intervals:

- Longer than the connection's RTT
  - » Too short, premature timeout and unnecessary retransmissions
  - » Too long, slow reaction to segment loss

$$\boldsymbol{EstimatedRTT} = (1-\propto) \times EstimatedRTT + \propto \times SampleRTT$$

The n**ew EstimatedRTT** is a **weighted combination** of the **previous value of EstimatedRTT** and the **new value of SampleRTT**. The **weighted average** puts **more weight** on **recent samples** rather **than old samples**. A t**ypical value : α = 0.125**. Also known as the **exponential weight moving average** (EWMA) because the weight of the given SampleRTT decays exponentially fast as the updates proceed.



The timeout interval then should be the EstimatedRTT plus a "safety margin".

Figure 3.32 ♦ RTT samples and RTT estimates

$$\textbf{\textit{Timeout Interval}} = EstimatedRTT + 4 \times DevRTT$$

Reliable data transfer

- TCP creates reliable data transfer service on top of IP's unreliable service. This is through:
  - » Pipelined segments
  - » Cumulative ACKs
  - » Single retransmission timer
- Retransmissions triggered by
  - » Timeout events
  - » Duplicate ACKs

TCP sender events

Data received from application

- Create segment with sequence #
- Sequence # is byte-stream number of first data byte in segment
- Start timer if not already running
  - » Timer is for oldest unACKed segment
  - » Expiration interval: TimeOutInterval

Timeout

- Retransmit segment that caused timeout
- Restart timer

Acknowledgement received

- If ACK acknowledges previous unACKed segments
  - » Update what is known to be ACKed
  - » Start timer if there are still unACKed segments

Flow control

TCP provides **flow control service** to **eliminate the possibility** of the sender **overflowing** the **receiver's buffer**. Flow control is a speed-matching service, matching the rate at which the sender is sending against the rate at which the receiving application is reading. TCP is provided by:

- Maintaining a variable called the **receive window**
  - » Give the sender an idea of how much free buffer space there is
- Sender limits amount of unACKed data to receiver's receive window value
- Guarantees receive buffer will not overflow

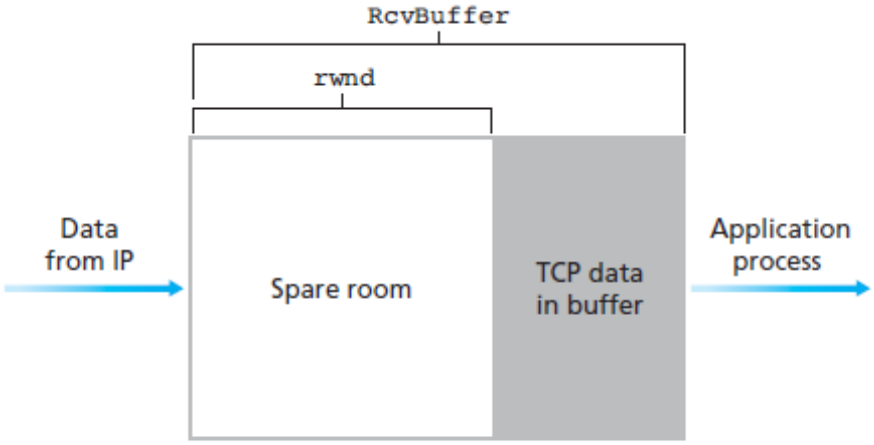


Figure 3.38 ♦ The receive window (`rwnd`) and the receive buffer (`RcvBuffer`)

Connection management

Before the exchange of data, the sender & receiver "handshake" to agree to establish a connection, and to agree on connection parameters.

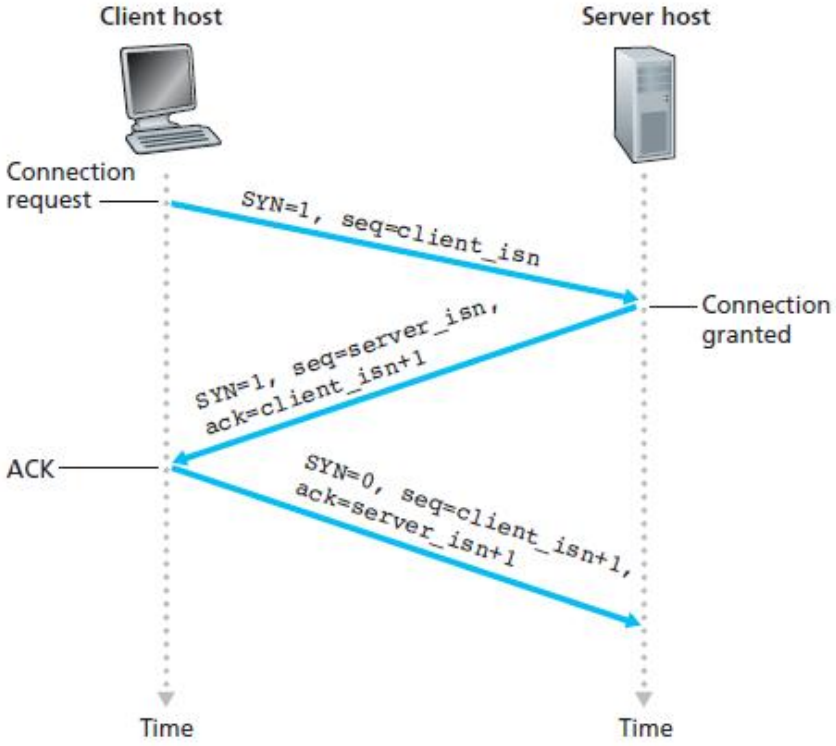


Figure 3.39 ♦ TCP three-way handshake: segment exchange

Closing a connection

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

3.6 Principles of Congestion Control

Congestion is when **too many sources are sending too much data too fast** for network **to handle** (different from flow control).

Scenario 1: Two Senders, a Router with Infinite Buffers

- There are two senders and two receivers
- One router, with infinite buffers
- Output link capacity R
- No retransmission

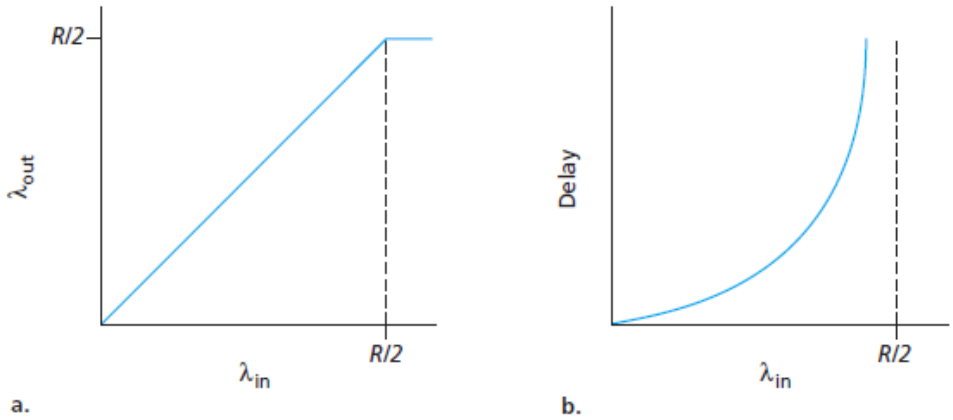


Figure 3.44 ♦ Congestion scenario 1: Throughput and delay as a function of host sending rate

Scenario 2: Two Senders, a Router with Finite Buffers

- Sender retransmission of timed-out packet
- **Idealisation**
  - » Packets can be lost, dropped at router due to full buffers
  - » Sender only resends if packet known to be lost
- **Realistic**
  - » Packets can be lost, dropped at router due to full buffers
  - » Sender times out prematurely, sending two copies, both of which are delivered
- More retransmission for given "goodput"
- Unneeded retransmissions: link carries multiple copies of packet
  - » Decreasing goodput

Scenario 3: Four Senders, Routers with Finite Buffers, and Multihop Paths

- As λ_{in} and λ_{in}' increases, all arriving packets at upper queue are dropped.
- When packet is dropped, any "upstream" transmission capacity used for λ_{out} packet is wasted, throughput → 0

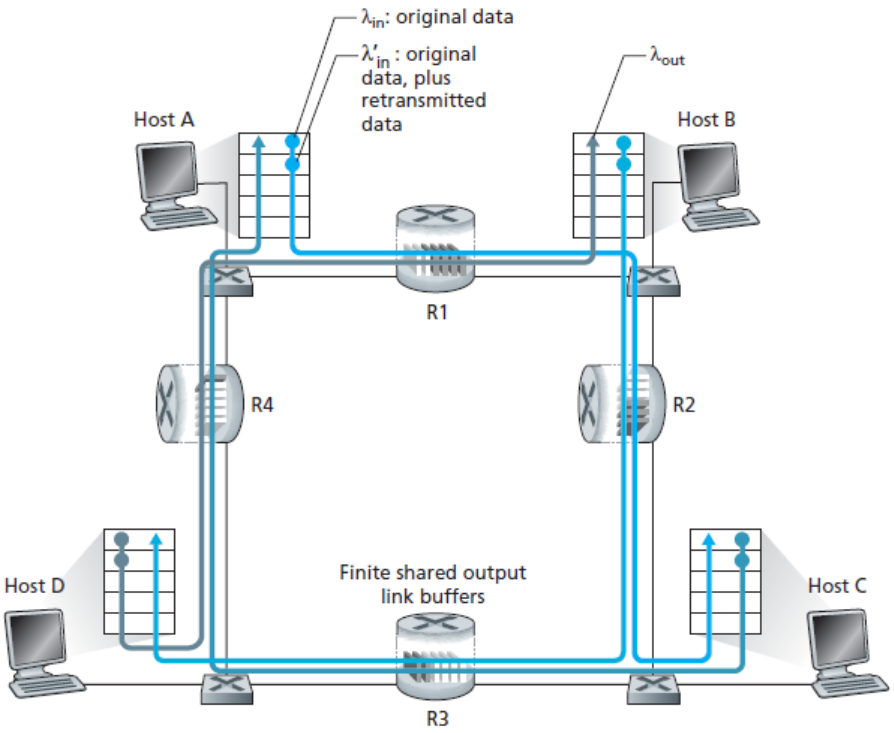


Figure 3.47 ♦ Four senders, routers with finite buffers, and multihop paths

3.7 TCP Congestion Control

Two approaches towards congestion control

- **End-end congestion control**
  - » No explicit feedback from network
  - » Congestion inferred from end-system observed loss, delay
  - » Approach taken by TCP
- **Network-assisted congestion control**
  - » Routers provide feedback to end systems
    - Single bit indicating congestion (SNA, DECbit, TCP/IP ECN, ATM)
    - Explicit rate for sender to send at
    - *Not covered in this subject*

Congestion Control

The sender increases transmission rate (window size), probing for usable bandwidth, until loss occurs:

- **Additive increase**: increase congestion window by 1 MSS every RTT until loss detected
- **Multiplicative decrease**: cut congestion window in half after loss

TCP sending rate:

- Roughly: send congestion window bytes, wait for RTT for ACKs, then send more bytes

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

- Sender limits transmission

$$LastByteSent - LastByteAcked \leq \min\{cwnd, rwnd\}$$

- Congestion window is dynamic, function of perceived network congestion
- For simplicity, assume receiver window is large enough

Slow Start

- When connection begins, **increase rate exponentially until first loss event**
  - » Initially cwnd = 1 MSS
  - » Double cwnd every RTT
  - » Done by incrementing cwnd for every ACK received
- Initial rate is slow but ramps up exponentially fast
- Loss indicated by timeout
  - » Cwnd set to 1 MSS
  - » Window then grows exponentially to threshold, then grows linearly
- Loss indicated by 3 duplicate ACKs: TCP RENO
  - » Dup ACKs indicate network capable of delivering some segments
  - » cwnd is cut in half window then grows linearly
- TCP Tahoe always set cwnd to 1 (timeout or 3 duplicate ACKs

Congestion Avoidance

- cwnd gets to 1/2 of its value before timeout, then switch to linear increase
- On entry to congestion-avoidance state, the value of cwnd is half its value since congestion was encountered

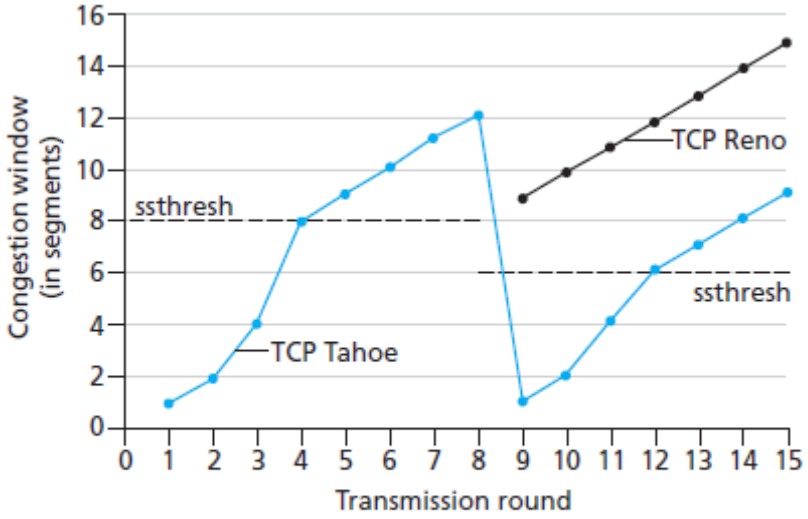


Figure 3.53 ♦ Evolution of TCP's congestion window (Tahoe and Reno)

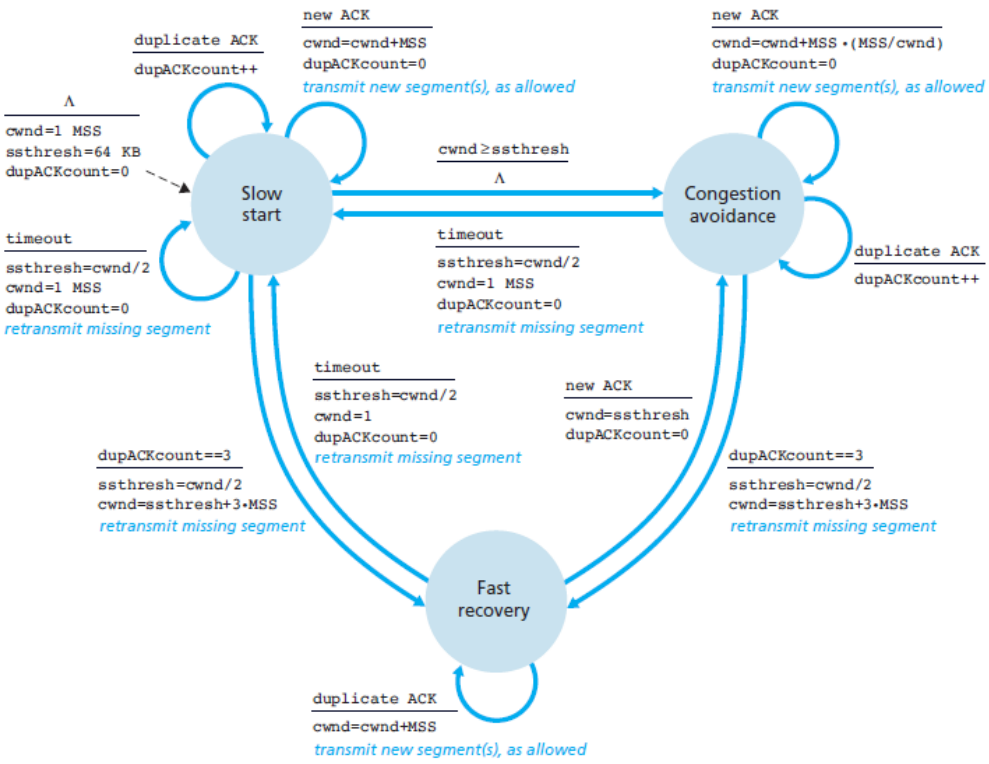


Figure 3.52 ♦ FSM description of TCP congestion control

TCP throughput

- W = window size (bytes) where loss occurs
  - » Average window size (# in-flight bytes) is 3/4 W ($\frac{1}{2}(\frac{w}{2}+w)$)
  - » Average through put is 3/4 W per RTT

$$average\ TCP\ throughput = \frac{\frac{3}{4}\frac{W}{RTT}bytes}{sec}$$

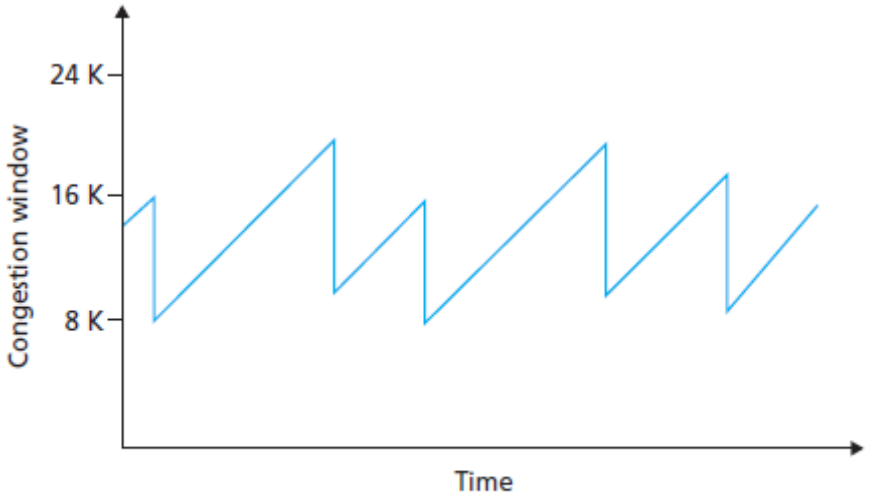


Figure 3.54 ♦ Additive-increase, multiplicative-decrease congestion control

Fairness

TCP has a **fairness goal** in which if K TCP sessions share the same **bottleneck link** of bandwidth R, each should have a **average rate or R/K**

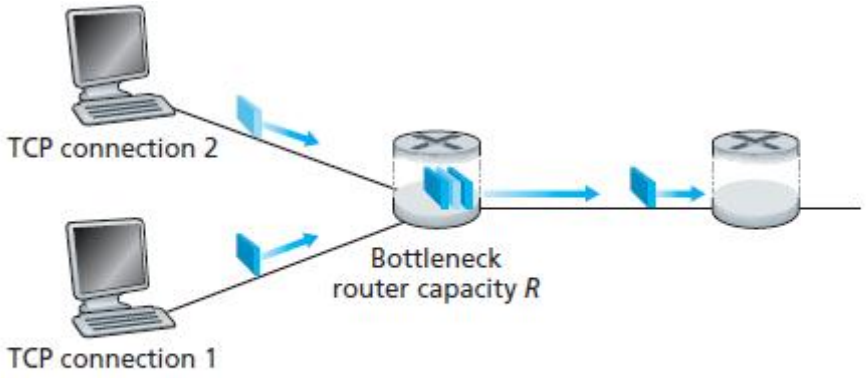


Figure 3.55 ♦ Two TCP connections sharing a single bottleneck link

Fairness and UDP

- Multimedia apps often do not use TCP
  - » Do not want rate throttled by congestion control
- Instead use UDP
  - » Send audio/video at constant rate, tolerate packet loss

Fairness, parallel TCP connections

- Applications can open multiple parallel connections between two hosts
- Web browsers do this
- E.g. link of rate R with 9 existing connections
  - » New app asks for 1 TCP, gets rate R/10
  - » New app asks for 11 TCPs, gets R/2

Network Layer

4.1 Introduction

The role of the network layer is to move packets from sending host to receiving host.

Forwarding

- When packet arrives at router's input link, router moves the packet to the appropriate output link
- Refers to router-local action of transferring a packet from an input link interface to the appropriate output link interface

Routing

- Network layer must determine the route or path taken by packets from a sender to a receiver. The algorithms that calculate these paths are referred to as routing algorithms
- Refers to the network-wide process that determines the end-to-end paths that packets take from source to destination

Every router has a **forward table**. A router **forwards a packet** by **examining** the **value** of a field in the arriving **packet's header**, and then uses this value to **index** into the router's forwarding **table**. The value stored in the table for that header **indicates** the router's **outgoing link interface** to which that packet is to forwarded.

The routing algorithm may be **centralised** (e.g. with an algorithm **executing on a central site** and downloading routing information to each of the routers) or **decentralised** (i.e. with a piece of the **distributed routing algorithm** running in **each router**).

Packet switch is a **device** that **transfers a packet from input link** interface **to output link** interface, according to the **value in a field** in the **header** of the packet. Some of these are called **link-layer switches**, which base their forward decision on **values in the fields of the link-layer frame**. Other packet switches are called **routers**, which base their forwarding decision on the value in the **network-layer field** (thus called network-layer devices).

Connection setup refers to the **process** where **routers** along the chosen path from source to destination **handshake**, to **set up a state before transferring** network-layer data packets.

Service Model

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

Services for **individual datagrams**

- Guaranteed delivery
  - » Guarantees the packet will eventually arrive at its destination
- Guaranteed delivery with bounded delay
  - » Guarantees delivery and also within a specified host-to-host delay bound (for example, within 100msec)

Services for a **flow of datagrams**

- In-order packet delivery
  - » Guarantees packets arrive at the destination in the order that they were sent
- Guaranteed minimal bandwidth
  - » Emulates the behaviour of a transmission link of a specified bit rate (e.g. 1 Mbps) between sending and receiving hosts
  - » As long as the sending hosts transmits bits at a rate below the specified bit rate, then no packet is lost and each packet arrives within a pre-specified host-to-host delay (e.g. within 40 msec)
- Guaranteed maximum jitter
  - » Guarantees 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 the destination (their spacing changes by no more than some specified value)
- Security services
  - » Uses secret session key known only by a source and destination host, encrypts the payloads of all datagrams being sent to the destination host

» The network layer in the destination host would be responsible for decrypting the payloads
» Provides confidentially, data integrity and source authentication

Network Architecture	Service Model	Bandwidth Guarantee	No-Loss Guarantee	Ordering	Timing	Congestion Indication
Internet	Best Effort	None	None	Any order possible	Not maintained	None
ATM	CBR	Guaranteed constant rate	Yes	In order	Maintained	Congestion will not occur
ATM	ABR	Guaranteed minimum	None	In order	Not maintained	Congestion indication provided

4.2 Virtual circuits and datagram networks

A network layer can provide connectionless or connection-oriented service, like a transport system can choose between UDP and TCP. A **connection service** at the network layer are called **virtual-circuit (VC) networks**; whilst a **connectionless** service is called **datagram networks**.

Virtual-Circuit Networks

A VC consists of:

1. A path (series of links and routers) between source and destination hosts
2. VC numbers, one number for each link along path
3. Entries in the forwarding table in routers along path

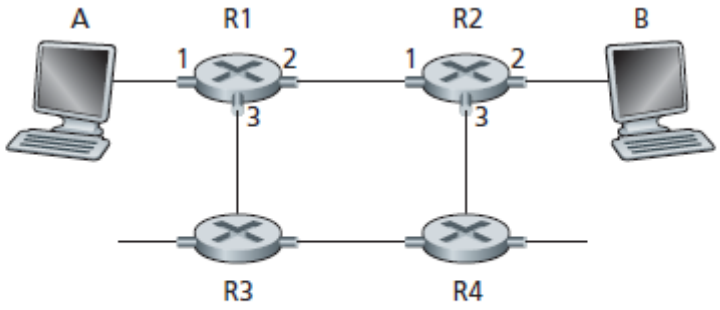


Figure 4.3 ♦ A simple virtual circuit network

A packet belonging to a VC will carry a VC number in its header. A VC may have a different VC number on each link, therefore each intervening router must replace the VC number of each traversing packet with a new VC number. The new number is obtained from the forwarding table.

Setting up a virtual circuit:

- VC Setup, source hosts sends a setup message
  - » Setup message contains destination address
- All intermediate routers

    - » Chooses an unused VC number (usually lowest available) as the incoming VC number
    - » Determine the outgoing interface from the routing table (**note**, still need routing tables for the setup phase)
    - » Create an entry in the VC table with incoming interface, incoming VC number, outgoing interface but an empty outgoing VC number
    - » Forward the setup message to the next hop
- Setup message reaches the destination
    - » The destination chooses an available VC number as the incoming VC number
        - The chosen incoming VC number will become the outgoing VC number of the penultimate router in this VC
        - The chosen VC number is inserted in an acknowledgement of connection setup which is sent along the reverse path
- The intermediate routers
    - » Completes the VC table entry
    - » Sends acknowledgement containing the outgoing VC number that the upstream router required

Datagram Networks

- No call setup at network layer
- Routers: no state about end-to-end connections
    - » No network-level concept of "connection"
- Packets forwarded using destination host address
    - » Packets between same source-destination pair may take different paths

Datagram (internet)	**VC Network (ATM)**
• data exchange among computers » "elastic" service, no strict • timing requirements » "smart" end systems (computers) » can adapt, perform control, • error recovery » simple inside network, • complexity at "edge" » many link types » different characteristics • • uniform service difficult	• evolved from telephony » human conversation: » strict timing, reliability • requirements » need for guaranteed service » "dumb" end systems » telephones • • complexity inside network

4.3 What's inside a router

There are four router components:

- Input ports
    - » Performs the physical layer function of terminating an incoming physical link at a router

    - » Performs link-layer functions needed to interoperate with the link layer at the other side of the incoming link
    - » Lookup function
- Switching fabric
    - » Connects the router's input ports to its output ports
- Output ports
    - » Stores packets received from the switching fabric and transmits these packets on the outgoing link
    - » If link is bidirectional, the output port will typically be paired with the input port for that link on the same line card
- Routing processor
    - » Executes the routing protocols and performs network management functions

These four components implement the **forwarding function** and are **implemented in hardware**. These are collectively known as the router forwarding plane which operates at the nanosecond time scale. A **router's control functions** – executing the routing protocols, responding to the attached links that go up or down, and performing management functions – operate at the millisecond or second timescale. These router control plane functions are usually **implemented in software** and execute on the routing processor (traditional CPU).

Input ports

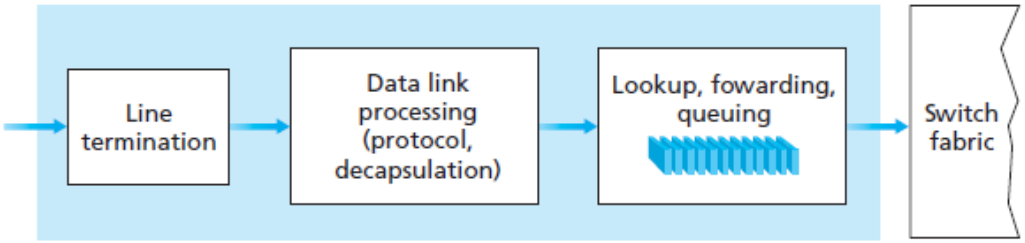


Figure 4.7 ♦ Input port processing

- Router uses the forwarding table to look up the output port
- The forwarding table is computed and updated by the routing processor, with a shadow copy typically stored at each input port
- Forwarding table is copied from the routing processor to the line cards over a separate bus (e.g. PCI bus)
- Once output port has been determined via lookup, packet can be sent into the switching fabric

Switching

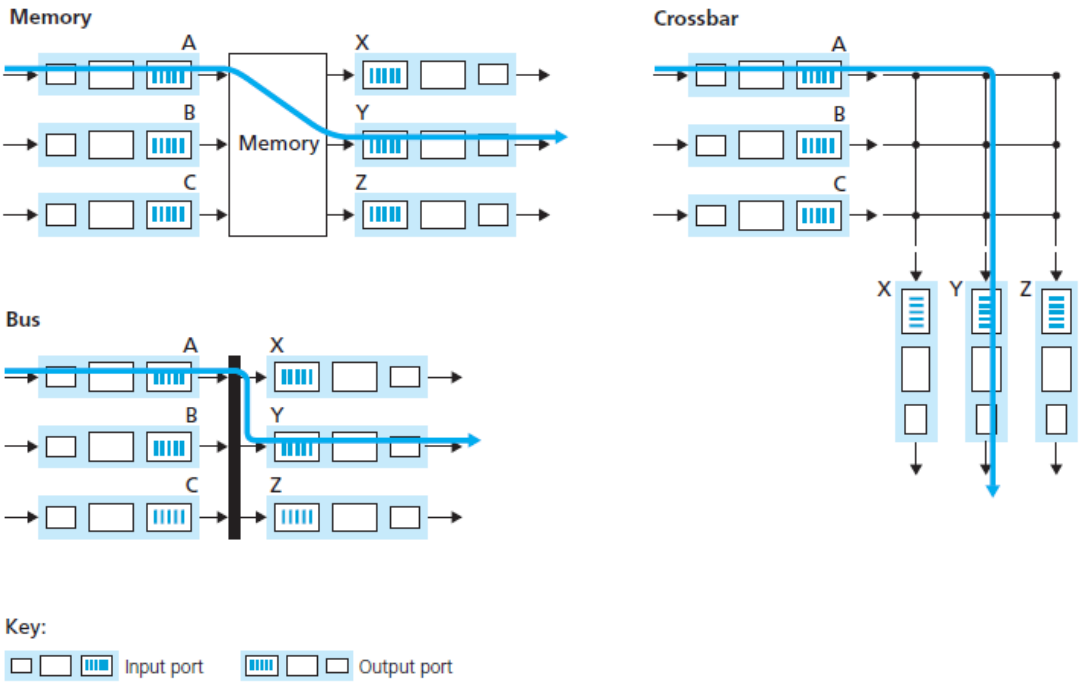


Figure 4.8 ♦ Three switching techniques

Switching via Memory

- Switching via the CPU
- Input and output ports functioned as traditional I/O devices
- Input port with an arriving packet first signalled the routing processor via an interrupt
- Packet was then copied from input into processor memory
- Extracted destination address from the header, looking up appropriate output port in forwarding table, and copied this packet to the output port's buffers
- Memory bandwidth be **B packets/second** (can be written or read), **overall forwarding throughput is less then B/2**
- **Note**: two packets cannot be forwarded at the same time

Switching via Bus

- Input port transfers a packet directly to the output port over a shared bus
- Input port pre-pend a switch-internal label (header) to the packet indicating the local output port to which this packet is being transferred and transmitting the packet onto a bus
- Packet is received by all output ports, but only the port that matches the label will keep the packet
- Label is then removed at output port (label only used to cross the bus)
- If multiple packets arrive to the router at the same time, each at a different input port, all but one has to wait since only one packet can cross the bus at a time
- Switching speed is limited to the bus speed
- Sufficient for routers that operate in small local area or enterprise networks

Switching via an Interconnection Network

- Cross bar switching is an interconnection network consisting of 2N buses that connect N input ports to N output ports
- 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 busses A and Y, and port A then sends the packet onto its bus, which is picked up (only) by bus Y (refer to Figure 4.8)
- **Note**: a packet from port B can be forwarded to port X at the same time, since the A-to-Y and B-to-X packets use different input and output busses.
- Crossbar networks are capable of forwarding multiple packets in parallel only if it is not to the same output port

Output

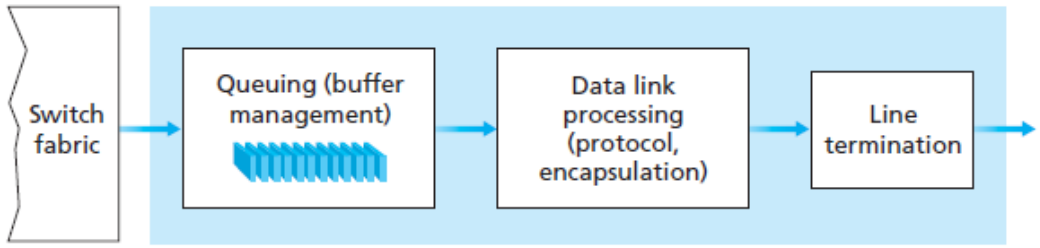


Figure 4.9 ♦ Output port processing

- Takes packets that have been stored in the output port's memory and transmits them over the output link (selecting and de-queuing packets for transmission)

Queuing

Router's memory is finite, and it will eventually be exhausted, causes packet loss when no memory is available.

4.4 IP: Internet Protocol

The internet is a network of **heterogeneous** networks:

- Uses different link layer technologies
  - » Ethernet, PPP, Wi-Fi, 3G...
- Belonging to different administrative authorities
- Arranged in a loosely hierarchical manner

The **goal of IP** is to **interconnect** all these diverse networks so that they can **send data end-to-end** without any knowledge of the intermediate networks. **Routers** are machines that forward packets between heterogeneous networks.

Datagram Format

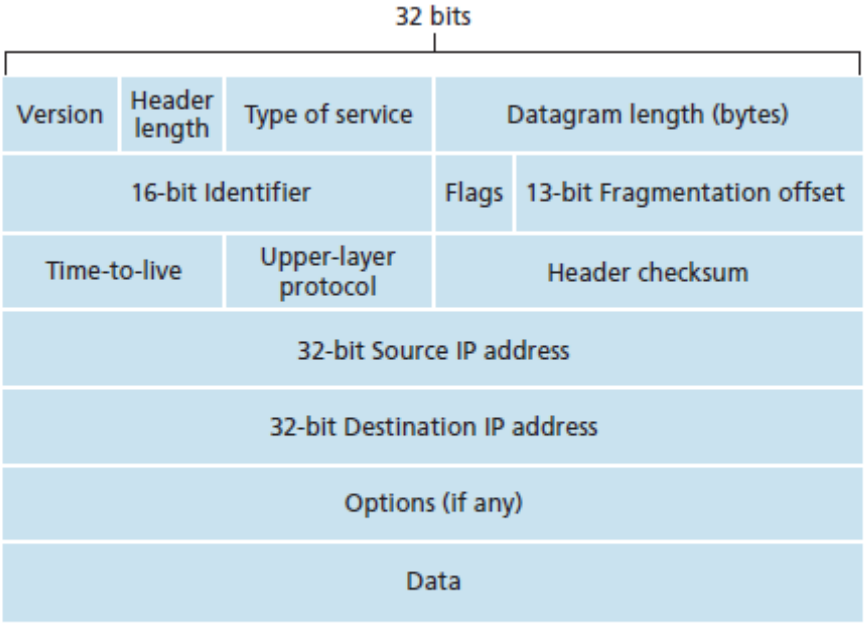


Figure 4.13 ♦ IPv4 datagram format

Version number: 4bits specify IP protocol version. Different versions of IP use different datagram formats.

Header length: 4 bits to determine where in the IP datagram the data actually begins (typically **20 byte header**).

Type of service: TOS bits to allow different types of IP datagrams (e.g. datagrams particularly requiring low delay, high throughput, or reliability) to be distinguished from each other.

Datagram length: (max **16 bytes, 65,535 bytes**) total length of the IP datagram (**header plus data**) measured in bytes. Datagrams are rarely larger than 1,500 bytes.

Identifier, flags, fragmentation offset: three fields have to do with IP fragmentation

Time-to-live: ensures that datagrams do not circulate forever (if the TTL reaches 0, datagram is dropped)

Protocol: used when an IP datagram reaches its final destination. Indicates the specific transport-layer protocol to which the data portion of this IP datagram should be passed (e.g. 6 = TCP, 17 = UDP).

Header checksum: aids a router in detecting bit errors in a received IP datagram

Source and destination IP addresses: source inserts its IP address into the source IP address field and inserts the address of the ultimate destination into the destination IP address field which is often determined via DNS lookup.

Options: allow IP header to be extended, but used rarely and thus was dropped in the IPv6 header.

Data (payload): contains the transport-layer segment (TCP or UDP) to be delivered to the destination, but can carry other types of data such as ICMP messages.

TCP Overhead: 20 bytes of TCP + 20 bytes of IP (no options) = **40 bytes + app layer overhead**

IP Datagram Fragmentation

The maximum amount of data that a link-layer frame can carry is called the maximum transmission unit (MTU) (e.g. Ethernet = 1500 bytes, wide area links = 576 bytes). If the datagram is larger than the MTU then the solution is to **fragment** the IP datagram.

To allow the destination host to perform the reassembly tasks, the designers of IPv4 put **identification**, **flag** and **fragmentation offset** fields in the IP datagram header.

IPv4 Addressing

IP address: 32 bits (4 bytes) identifier for host, router interface (total of 2^{32} possible IP addresses)

Interface: connection between host/router and physical link

- router's typically have multiple interfaces
- host may have multiple interfaces
- IP addresses associated with each interface

The addresses are typically written in dotted-decimal notation, where each byte of the address is written in its decimal form and is separated by a period from other bytes in the address (separated into 4 x 8 bits of address).

Subnets

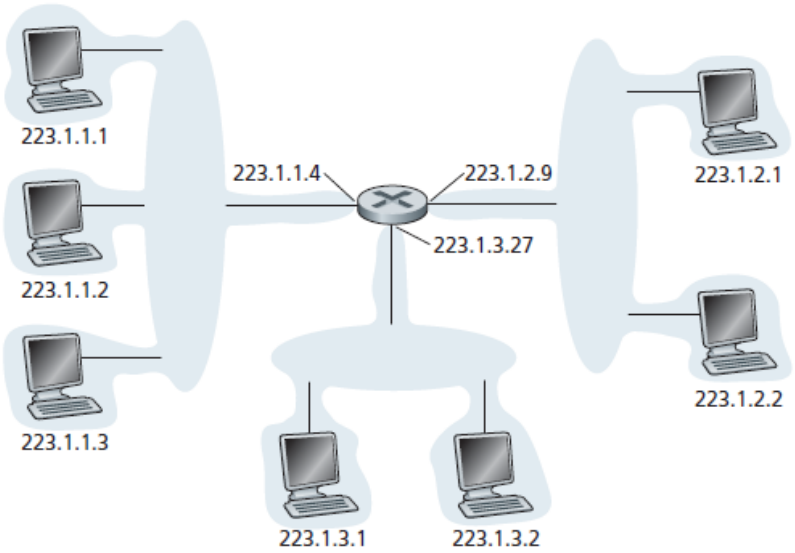


Figure 4.15 ♦ Interface addresses and subnets

This network interconnecting three host interfaces and one router interface forms a **subnet**, which is also called an IP network. IP addressing assigns an address to this subnet: 223.1.1.0/24, where the /24 notation (subnet mask) indicates that the leftmost 24 bits define the subnet address.

Classless InterDomain Routing (CIDR)

CIDR is the Internet's address assignment strategy which generalises the notion of subnet addressing. It also has the dotted-decimal form *a.b.c.d/x*, where *x* indicates the most significant bits of an address (constitutes the network portion of the IP address), and is often referred to as the **prefix.** The remaining bits of an address is typically used to **distinguish devices within the organisation** (as they have the same network prefix).

Before CIDR was adopted, an addressing scheme called classful addressing was used where 8-, 16-, 24- bit subnet addresses were called A, B and C. This was poorly implemented as class A did not support enough hosts (only 256) whilst class B supported way too many hosts which meant the address space was poorly utilized.

IP broadcast address 255.255.255.255 is used when a host sends a message which is delivered to all hosts on the same subnet.

How to get IP addresses?

- Block of addresses is obtained through its ISP
- Hard-coded by system admin in a file
  - » Wintel: control-panel →network → configuration → TCP/IP → properties
  - » UNIX: /etc/rc.config
- Dynamic Host Configuration Protocol (DHCP): dynamically gets address from a server
  - » "plug-and-play" as it automates network-related aspects of connecting a host into a network
  - » May receive the same IP address each time, or may be assigned a temporary IP address (which will be different each time)
  - » Allows a host to learn additional information
    - Subnet mask
    - Address of first-hop router (default gateway)
    - Address of its local DNS server

DHCP (client-server protocol)

- Useful in residential Internet access networks, and in wireless LANs
  - » E.g. student in university
    - Carries a laptop from dormitory room to a library to a classroom
    - Will be connecting to new subnet each time, therefore new IP addresses at each location
    - Many users coming and going and addresses only needed for a limited amount of time
  - » E.g. residential ISP (2,000 customers)
    - No more than 400 customers ever online at the same time
    - Instead of 2,048 addresses, only need 512 addresses
    - Each time a host joins the DHCP server allocates an arbitrary address from its current pool of available addresses; when a hosts leaves its address is returned to the pool

Process of newly arriving host

- **DHCP server discovery**
  - host has to find a DHCP server to interact with, using a **DHCP discover message**, which a client sends within a UDP packet to port 67
  - DHCP client creates an IP datagram containing its DHCP discover message along with the broadcast destination IP address of 255.255.255.255 and a "this host" source IP address of 0.0.0.0
- **DHCP server offer(s)**
  - DHCP server receiving a DHCP discover message responds to the client with a **DHCP offer message** that is broadcast to all nodes on the subnet (IP broadcast address)
  - Several DHCP servers can be present on the subnet so the client may be able to choose from among several offers
  - Server offer message contains the transaction ID or the received discover message, the proposed IP address, the network mask and an IP address lease time (time IP address is valid, common to be several hours or days)
- **DHCP request**
  - Arriving client will choose and respond to its selected offer with a **DHCP request message**, echoing back the configuration parameters
- **DHCP ACK**
  - Server responds with a **DHCP ACK message**, confirming the requested parameters

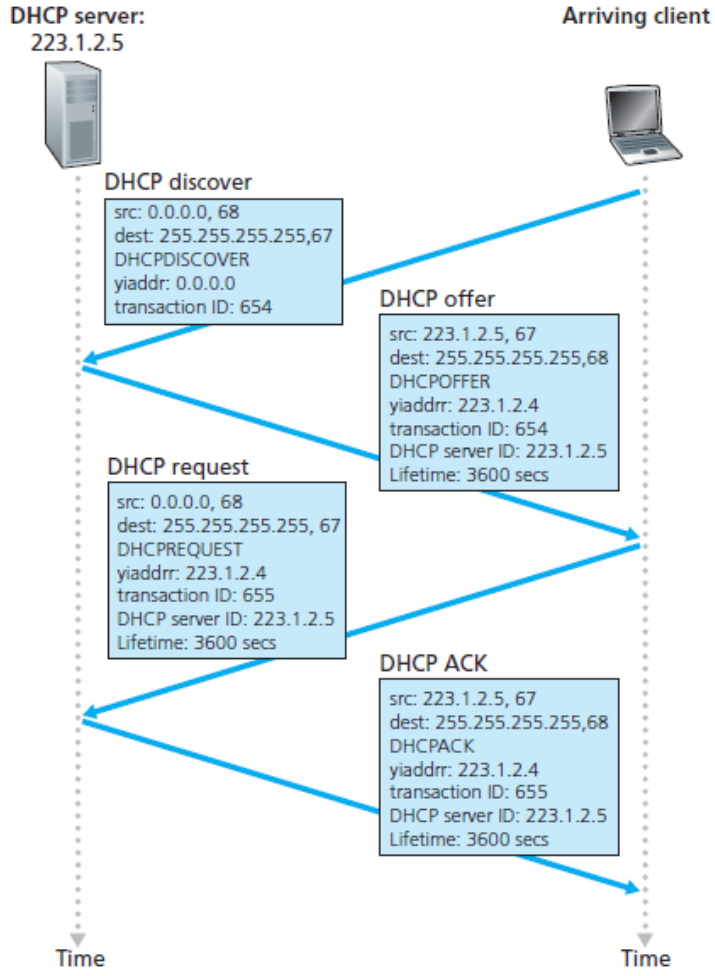


Figure 4.21 ♦ DHCP client-server interaction

DHCP also provides a mechanism that allows a client to renew its lease on an IP address. Further details include:

- DHCP uses UDP and port numbers 67 (server side) and 68 (client side)
- Usually the MAC address is used to identify clients
  - » DHCP server can be configured with a "registered list" of acceptable MAC addresses
- DHCP offer message includes ip address, length of lease, subnet mask, DNS servers, default gateway
- DHCP security holes
  - » DoS attack by exhausting pool of IP addresses
  - » Masquerading as a DHCP server
  - » Authentication for DHCP - RFC 3118

Network Address Translation (NAT)

A local network uses just one public IP address (on the outside) but each device on the local network is assigned a private IP address. The address space 10.0.0.0/8 is one of the three portions of the IP address space that is reserved in [RFC 1918] for a private network or a realm with private address, such as a home network.

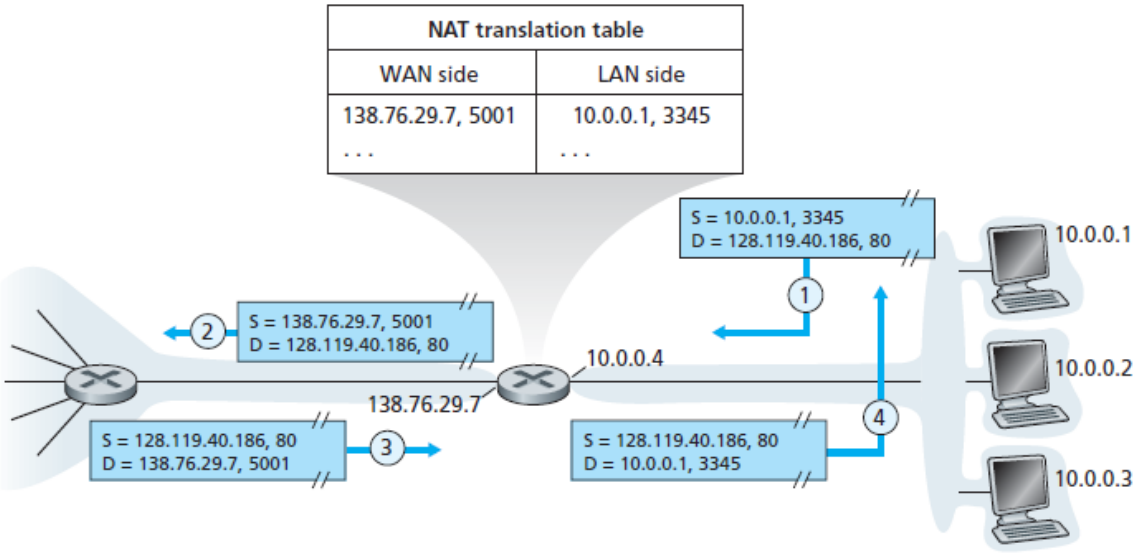


Figure 4.22 ♦ Network address translation

All datagrams leaving local network have same single source NAT IP address: 138.76.29.7, difference source port numbers. However Datagrams with source or destination in this network have 10.0.0/24 address for source, destination.

Why do we need NAT?

- IP addresses are 32 bit long and globally unique
  - » # of possible IP addresses are: 232
- That means we are running out of IP addresses
  - » # of devices being connected to the Internet is growing rapidly
- Abundance of SOHO (small office, home office) subnets
- Fix the problem
  - » IPv6 - not yet deployed universally

- Network Address Translation (NAT)
- Uses private addresses

Benefits

- local network uses just one IP address as far as outside word is concerned
  - no need to be allocated range of addresses from ISP: - just one IP address is used for all devices
  - can change addresses of devices in local network without notifying outside world
  - can change ISP without changing addresses of devices in local network
  - devices inside local net not explicitly addressable, visible by outside world (a security plus)
- Target Market
  - Small businesses, Public Internet access networks (hotspots), residential networks

Implementation

NAT router must:

- outgoing datagrams: replace (source IP address, source port #) of every outgoing datagram to (NAT IP address, new port #)

  . . . remote clients/servers will respond using (NAT IP address, new port #) as destination address
- remember (in NAT translation table) every (source IP address, source port #) to (NAT IP address, new port #) translation pair
- incoming datagrams: replace (NAT IP address, new port #) in destination fields of every incoming datagram with corresponding (source IP address, source port #) stored in NAT table
- But what if there are really public addresses with the same IP addresses as the private addresses?
  - RFC 1918 - 3 address ranges reserved for private addresses - not routable
    - 10.0.0.0/8, 172.16.0.0/12, 192.68.0.0/16

Issues

- 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!
- NAT is controversial:
  - port numbers are used for addressing processes, not hosts
  - routers should only process up to layer 3
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications
  - address shortage should instead be solved by IPv6
    - Increases size of IP addresses from 32 bits to 128 bits

Practical issues

- NAT modifies port # and IP address
  - Requires recalculation of TCP and IP checksum

- Some applications embed IP address or port numbers in their message payloads
  - » DNS, FTP (PORT command), SIP, H.323
  - » For legacy protocols, NAT must look into these packets and translate the embedded IP addresses/port numbers
  - » What if these fields are encrypted ? (SSL/TLS, IPSEC, etc)
- If applications change port numbers periodically, the NAT must be aware of this
- NAT Traversal Problems
  - » E.g. How to setup a server behind a NAT router?
  - » How to talk to a Skype user behind a NAT router?

4.5 Routing algorithms

Routing algorithm classification

Global vs. Decentralised

Global

- Computers the least-cost path between a source and destination using complete, global knowledge
- Algorithm takes the connectivity between all nodes and all link costs as input
- ***"link state" algorithms***

Decentralized

- router knows physically connected neighbours, link costs to neighbours
- iterative process of computation, exchange of info with neighbours
- ***"distance vector" algorithms***

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

- net topology, link costs known to all nodes
  - » accomplished via "link state broadcast"
  - » all nodes have same info
- computes least cost paths from one node ('source") to all other nodes
  - » gives forwarding table for that node
- iterative: after k iterations, know least cost path to k destinations

Distance Vector Algorithm

The least costs paths are equated by the **Bellman-Ford equation** (dynamic programming):

$$D_x(y) := \text{cost of least-cost path from x to y, then } d_x(y) = \min\{c(x,v) + d_v(y)\}$$

- Each node maintains a routing table indexed by, and containing one entry (two parts) for, each router in subnet
  - » Preferred outgoing line, and estimate of distance or time for destination (other metrics possible)
  - » Nodes assumed to know the distance to its neighbours (one hop for metrics = hop)
  - » Example uses delay metrics, each node sends to neighbour a list of its estimated delay to each destination
- Node receives a list with an entry from a neighbour X saying my estimate to node i is Xi. This node knows that it takes m msec to reach X. It concludes that it can reach i via X in Xi+m msec.

Link State vs. Distance Vector

	Link State	**Distance Vector**
Message Complexity	• With n nodes, E links, O(nE) messages sent	• Exchange between neighbours only • Convergence times varies
Speed of Convergence	• $O(n^2)$ algorithm requires O(nE) messages • May have oscillations	• Convergence times varies • May be routing loops • Count-to-infinity problem
Robustness (what happens if router malfunctions)	• Node can advertise incorrect link cost • Each node computes only its own table	• DV node can advertise incorrect path cost • Each nodes table used by others » Error propagate through network

Hierarchical Routing

In practice, there are 200 million destinations meaning you **cannot store all destinations in routing tables**, otherwise routing table exchange would swamp links. Since the internet is a network of networks, each **network admin** may want to **control routing in its own network**. Instead, **aggregate routers into regions, "autonomous systems" (AS)**. Routers in the same AS run the same routing protocol but routers in **different AS can run different intra-AS routing protocols**. A **gateway router** directly links to router in another AS.

Inter-Autonomous Systems tasks

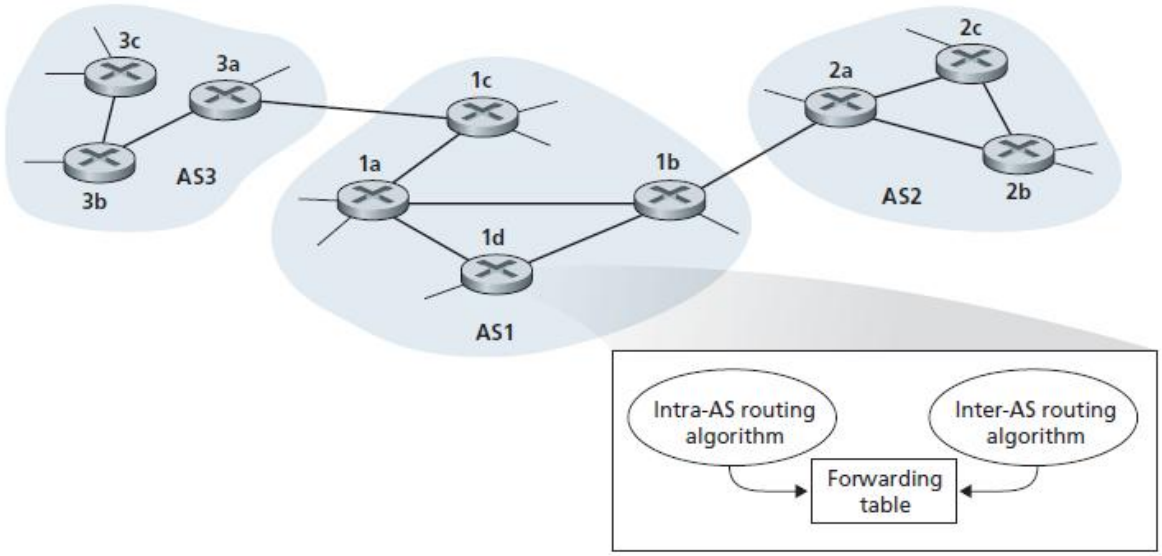


Figure 4.32 ♦ An example of interconnected autonomous systems

- Suppose router in AS 1 receives datagram for which destination is outside of AS1
- AS1 needs:
  - » To learn which destinations are reachable through AS2 and which through AS3
  - » To propagate this reachability information to all routers in AS1 (job of inter-AS routing)

Choosing among multiple ASes

- Suppose there is an inter-AS protocol that subnet x is reachable from both AS3 and AS2
- To configure forwarding table, router 1d must determine towards which gateway it should forward packets for destination x.

Hot potato routing involves sending packets towards the **closer** one of the two **routers**.

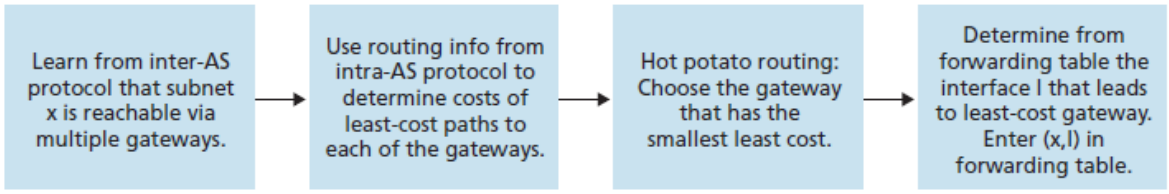


Figure 4.33 ♦ Steps in adding an outside-AS destination in a router's forwarding table

Data Link Layer

5.1 Introduction to the Link Layer

The data-link layer has the responsibility of transferring datagram from one node to physically adjacent node over a link.

- **Node**: any device that runs a link-layer protocol (includes hosts, routers, switches and Wi-Fi access points)
- **Links**: communication channels that connect adjacent nodes along the communication path (wire links, wireless links, LANs)

The datagram is transferred by different link protocols over different links. Each link protocol provides different services (may or may not provide rdt over link).

On the sending side:

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

On the receiving side:

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

Services provided by the Link Layer

- **Framing**
  - » Encapsulate datagram into frame, adding header, trailer
- **Link access**
  - » Channel access if shared medium
  - » Medium Access Control (MAC) protocol addresses used in frame headers to identify source, destination
    - Different from IP address
- **Reliable delivery**
  - » Guarantees to move each network-layer datagram across the link without error
  - » Often used for links that are prone to high error rates, such as wireless link
- Error detection and correction
  - » Provides a mechanism to detect bit errors
    - Done by having the transmitting node include error-detection bits in the frame and having the receiving node perform an error check
    - Can determine exactly where in the frame the error occurs

Where the link layer implemented?

- in each and every host
- link layer implemented in “adaptor” (aka ***network interface card*** NIC) or on a chip
  - » Ethernet card, 802.11 card; Ethernet chipset
  - » implements link, physical layer
- attaches into host’s system buses
- combination of hardware, software, firmware

5.2 Error Detection & Correction Techniques

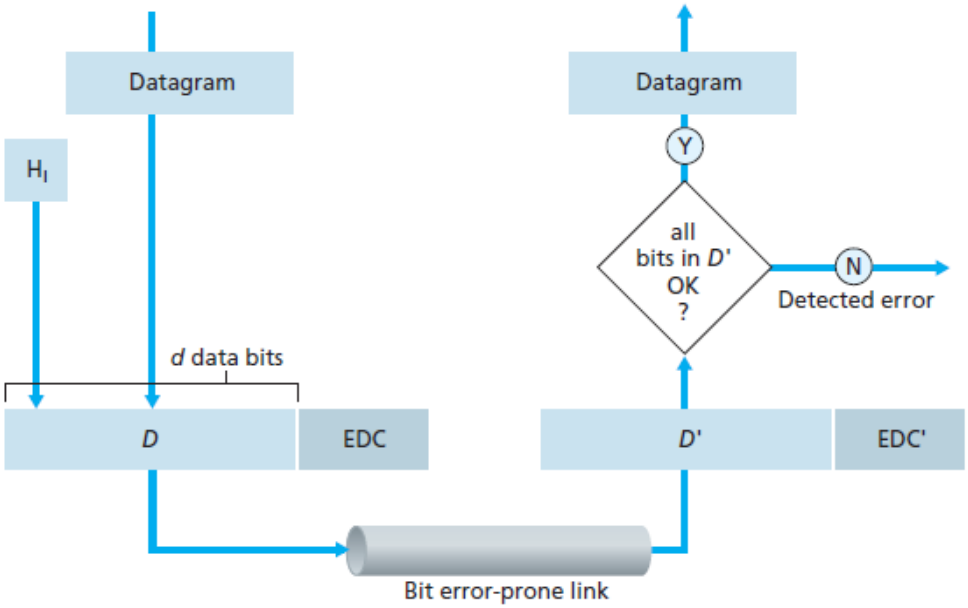


Figure 5.3 ♦ Error-detection and -correction scenario

- **D** = Data protected by error checking, may include header files
- **EDC** = Error Detection and Correction bits (redundancy)
- Error detection not 100% reliable
  - » Protocol may miss some errors
  - » Larger EDC field yields better detection and correction

Parity Checks

Single parity bit: detect single bit errors

Two-dimensional bit parity: detect and correct single bit errors

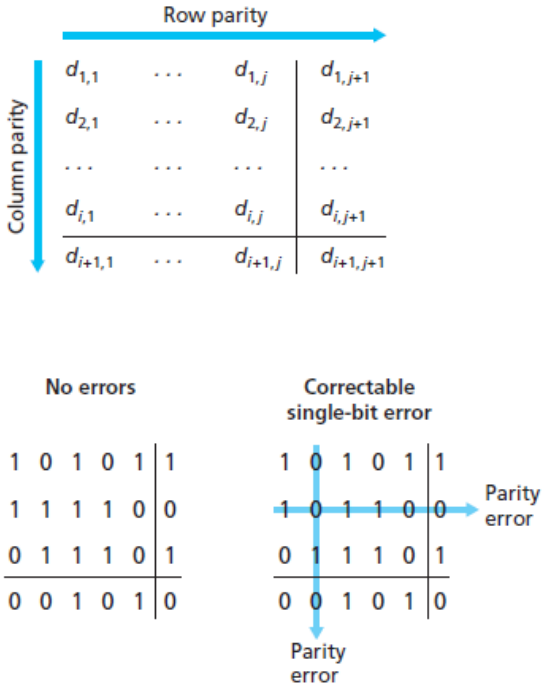
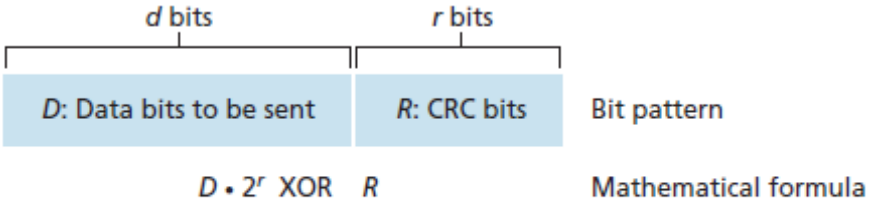


Figure 5.5 ♦ Two-dimensional even parity

For example, a single bit error occurs in the original *d* bits of information. With this **two-dimensional parity** scheme, the parity of both the column and the row containing the flipped bit will be in error. The receiver can thus not only *detect* the fact that a single bit error has occurred, but can use the column and row indices of the column and row with parity errors to actually identify the bit that was corrupted and *correct* that error

Cyclic Redundancy Checks

- more powerful error-detection coding
- view data bits, **D**, as a binary number
- choose r+1 bit pattern (generator), **G**
- goal: choose r CRC bits, **R**, such that
  - » <D,R> exactly divisible by G (modulo 2)
  - » receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
  - » can detect all burst errors less than r+1 bits
- widely used in practice (Ethernet, 802.11 WiFi, ATM)



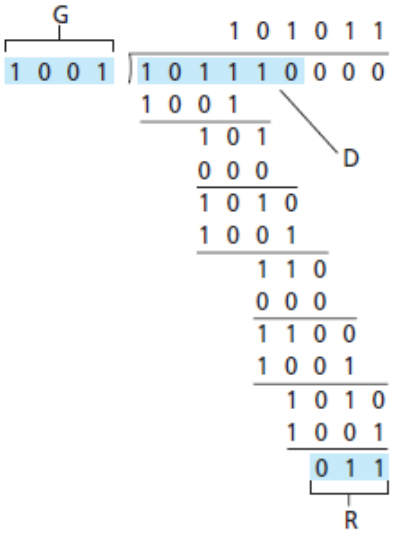


Figure 5.7 ♦ A sample CRC calculation

5.3 Multiple Access protocols

There are two types of "links":

- **Point-to-point**
  - » Single sender at one end of the link and a single receiver at the other end of the link
  - » PPP for dial-up access
  - » Point-to-point link between Ethernet switch, host
- **Broadcast** (shared wire or medium)

- » Multiple sending and receiving nodes all connected to the same, single, sharead broadcast channel
- » Old-fashioned Ethernet
- » Upstream HFC
- » 802.11 wireless LAN

Computer networks have Multiple Access Protocols by which nodes regulate their transmission into the shared broadcast channel.

- single shared broadcast channel
- two or more simultaneous transmissions by nodes: interference
  - » collision if node receives two or more signals at the same time
- distributed algorithm that determines how nodes share channel, i.e., determine when node can transmit
- communication about channel sharing must use channel itself
  - » no out-of-band channel for coordination

Ideally a multiple access protocol for a broadcast channel of rate R bits per second should have the following desirable characteristics:

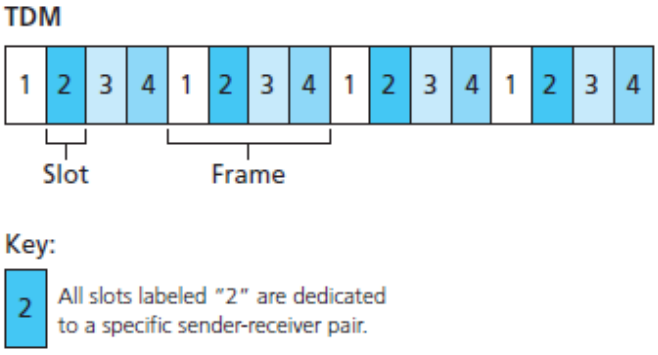
- when one node has data to send, that node has a throughput of R bps
- When M nodes have data send, each node has a throughput of R/M bps (on average)
- The protocol is decentralised, there is no master node that represents a single point of failure
- The protocol is simple, inexpensive to implement

Channel Partitioning protocols (time division multiple access TDMA)

- Divide channel into smaller "pieces" (time slots, frequency, code)
- Allocate piece to node for exclusive use

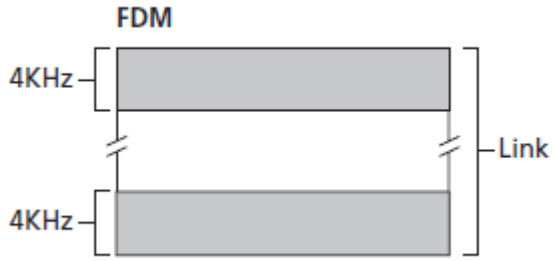
Time Division Multiple Access (TDMA)

- Access to channel in "rounds"
- Each station gets fixed length slot (length = packet transmission time) in each round
- Unused slots go to idle



Frequency Division Multiple Access (FDMA)

- Channel spectrum divided into frequency bands
- Each station assigned fixed frequency band
- Unused transmission time in frequency bands go idle



Random Access protocols

- Waits a random delay before retransmitting the frame
- When node has packet to send
  - » Transmit at full channel data rate R.
  - » no *a priori* coordination among nodes
- Two or more transmitting nodes ➜ "collision"
- Random access MAC protocol specifies:
  - » How to detect collisions
  - » How to recover from collisions (e.g., via delayed retransmissions)
- E.g. CSMA, CSMA/CD, CSMA/CA

Carrier Sense Multiple Access (CSMA)

- Listens before transmit
  - » If channel sensed idle → transmit entire frame
  - » If channel sensed busy → defer transmission for a random amount of time
- Collisions can still occur, propagation delay means two nodes may not hear each other's transmission
- When collision occurs, entire packet transmission time is wasted
  - » Distance & propagation delay play role in determining collision probability

Carrier Sense Multiple Access Collision Detection(CSMA/CD)

- collisions detected within short time
- colliding transmissions aborted, reducing channel wastage
- collision detection:
  - » easy in wired LANs: measure signal strengths, compare transmitted, received signals
  - » difficult in wireless LANs: received signal strength overwhelmed by local transmission strength

$$Efficiency = \frac{1}{1 + 5d_{prop}/d_{trans}}$$

D_{prop} = the maximum time it takes signal energy to propagate between any two adapters

D_{trans} = the time to transmit a maximum-size frame (approx. 1.2msecs for a 10 Mbps Ethernet)

Summary of operation

1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel idle, starts frame transmission. If NIC senses channel busy, waits until channel idle, then transmits.
3. If NIC transmits entire frame without detecting another transmission, NIC is done with frame

4. If NIC detects another transmission while transmitting, aborts and sends jam signal
5. After aborting, NIC enters binary (exponential)back off (the more collisions experienced by a frame, the larger the interval from which K is chosen)
    - after mth collision, NIC chooses K at random from {0,1,2, …, 2m-1}. NIC waits K·512 bit times, returns to Step 2
    - longer back off interval with more collisions

Taking-Turns protocols

- Nodes take turns, but nodes with more to send can take longer turns
- E.g. Bluetooth, FDDI, token ring

Polling protocol

- Master node "invites" slave nodes to transmit in turn
- Typically used with "dumb" slave devices
- Concerns:
    - » Polling overhead
    - » Latency
    - » Single point of failure (master)

Token-passing protocol

- Control token passed from one node to next sequentially
- Token message
- Concerns:
    - » Token overhead
    - » Latency
    - » Single point of failure (token)

Case Study: Cable Access Network

- Connects several thousand residential cable modems to a cable modem termination system (CMTS) at the cable network headend
- Data-Over-Cable Service Interface Specifications (DOCSIS) specifies the cable data network architecture and its protocols
    - » Uses FDM to divide the downstream and upstream network segments into multiple frequency channels
        - Downstream channel is 6MHz wide with max throughput as 40 Mbps
        - Upstream is 6.4 MHz wide with max throughput as 30Mbps
    - » Each upstream and downstream channel is a broadcast channel
    - » Each upstream channel is divided into intervals of time, containing a sequence of mini-slots
        - Mini-slot-request frames are transmitted in a random access manner, may collide
        - if there is little traffic, cable modem may actually transmit data frames during slots nominally assigned for mini-slot-request frames

5.4 Switched Local Area Networks (LANs)

Link-Layer Addressing & ARP

MAC Addresses

- 32-bit IP address:
  - » Network-layer address for interface
  - » Used for layer 3 (network layer) forwarding
- MAC (or LAN or physical or Ethernet) address:
  - » Function: used 'locally" to get frame from one interface to another physically-connected interface (same network, in IP addressing sense)
  - » 48 bit MAC address (for most LANs) burned in NIC ROM, also sometimes software settable
  - » E.g. : 1A-2F-BB-76-09-AD
- Each adapter on LAN has a unique LAN address
- MAC address allocation administered by IEEE
  - » Manufacturer buys portion of MAC address space (to assure uniqueness)
- Analogy:
  - » MAC address: like Social Security Number
  - » IP address: like postal address
- MAC flat address ➜ portability
  - » Can move LAN card from one LAN to another
- IP hierarchical address not portable
  - » Address depends on IP subnet to which node is attached

Address Resolution protocol (ARP)

- Translates the network-layer and link-layer addresses
- ARP table: each IP node (host, router) on LAN has table
  - » IP/MAC address mappings for some LAN nodes: <IP address; MAC address; TTL>
  - » Time To Live (TTL): time after which address mapping will be forgotten (typically 20 minutes)

Case Study: ARP (same LAN)

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

Case Study: ARP (different LAN)

Sending datagram from A (left) to B (right) via R (middle router)

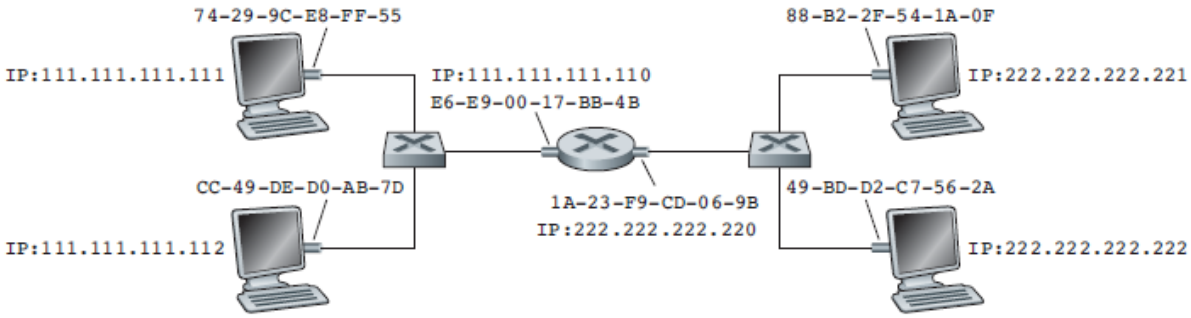
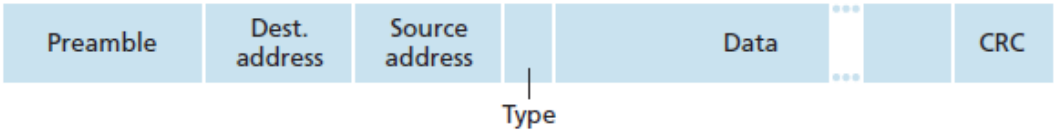


Figure 5.19 ♦ Two subnets interconnected by a router

- A creates IP datagram with IP source A, destination B
- A creates link-layer frame with R's MAC address as destination, frame contains A-to-B IP datagram
- Frame sent from A to R
- Frame received at R, datagram removed, passed up to IP
- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as destination, frame contains A-to-B IP datagram

Ethernet

Frame Structure



- **Addresses**: 6 byte source, destination MAC addresses
  - » If adapter receives frame with matching destination address, or with broadcast address (e.g. ARP packet), it passes data in frame to network layer protocol
  - » Otherwise, adapter discards frame
- **Type**: indicates higher layer protocol (mostly IP but others possible, e.g., Novell IPX, AppleTalk)
- **CRC**: cyclic redundancy check at receiver
  - » Error detected: frame is dropped
- **Connectionless**: no handshaking between sending and receiving NICs
- **Unreliable**: receiving NIC doesn't send acks or nacks to sending NIC
  - » Data in dropped frames recovered only if initial sender uses higher layer rdt (e.g., TCP), otherwise dropped data lost
- **Ethernet's MAC protocol**: unslotted CSMA/CD with binary backoff
- many different Ethernet standards
  - » common MAC protocol and frame format
  - » different speeds: 2 Mbps, 10 Mbps, 100 Mbps, 1Gbps, 10G bps
  - » different physical layer media: fibre, cable

Switches

The switch itself is transparent to the hosts and routers in the subnet (hosts are unaware of the presence of switches).

Forwarding and Filtering

- **Filtering**: function that determines whether a frame should be **forwarded** or **dropped**
- **Forwarding:** function that determines the interfaces to which a frame should be **directed**
- Switch filtering and forwarding are done with a **switch table** which contains
    - » MAC address
    - » Switch interface that leads toward that MAC address
    - » The time when the entry was placed in the table

Self Learning (plug-and-play)

Switches do not need to be configured, the **table** is **built automatically**, dynamically and autonomously.

- For each incoming frame received on an interface, the switch stores in its switch table
- The switch deletes an address in the table if no frames are received with that address as the source address after some period of time (aging time)

Properties

- **Elimination of collisions**
    - » Switches buffer frames and never transmit more than one frame on a segment at any one time
- **Heterogeneous links**
    - » Can operate at different speeds and can run over different media
    - » Ideal for mixing legacy equipment with new equipment
- **Management**
    - » Enhanced security and eases network management
    - » Can detect malfunctions and internally disconnect the problem
    - » Gathers statistics on bandwidth usage, collision rates and traffic types

Switches vs. Routers

Store-and-Forward

- **Routers**: network-layer devices
- **Switches**: link-layer devices

Forwarding Tables

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

Virtual Local Area Networks (VLANS)

Three drawbacks of LANs:

- **Lack of traffic isolation**
    - » Single broadcast domain

» All layer-2 broadcast traffic(ARP, DHCP, unknown location of destination MAC addresses) must cross entire LAN

- **Inefficient use of switches**
- **Managing users**
  - » Physical cabling must be changed to connect the employee to different switch
- **Security/Privacy issues**

These issues can be handled by a **switch** that **supports multiple virtual LANs**, allowing multiple VLANs to be defined over a **single physical LAN** infrastructure.

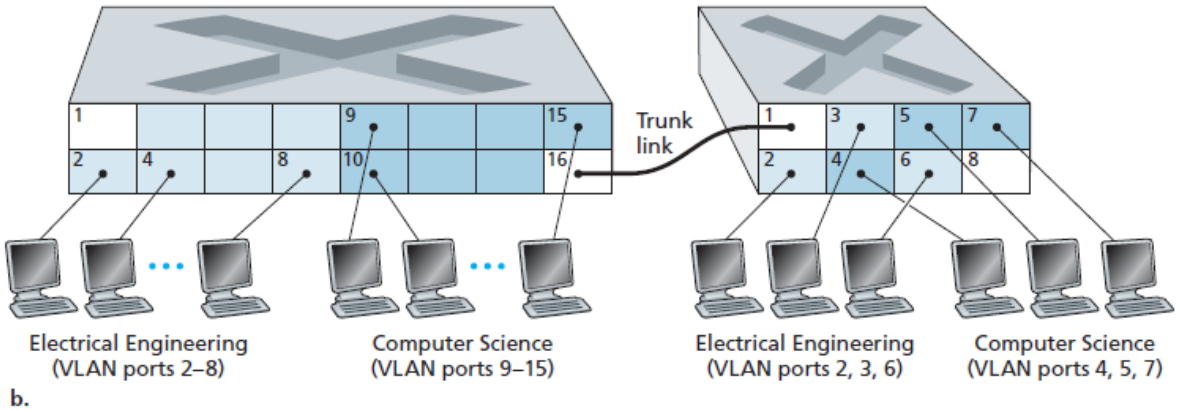
Port-based VLAN

The switch's ports (interfaces) are divided into groups by the network manager. Each group constitutes a VLAN, with the ports in each VLAN forming a broadcast domain.

- **Traffic isolation**: frames to/from ports 1-8 can only reach ports 1-8
  - » Can also define VLAN based on MAC addresses of endpoints, rather than switch port
- **Dynamic membership**: ports can be dynamically assigned among VLANs
- **Forwarding between VLANS**: done via routing (just as with separate switches)
  - » In practice vendors sell combined switches plus routers

Trunk porting carries frames between CLANS defined over multiple physical switches.

- Frames forwarded within VLAN between switches can't be vanilla 802.1 frames (must carry VLAN ID info)
- 802.1q protocol adds/removes additional header fields for frames forwarded between trunk ports
- Port 16 (left switch) connects to port 1 (right switch) configured as trunk ports (belong to all VLANs)



b.

5.6 Data Centre Networking

- 10's to 100's of thousands of hosts, often closely coupled, in close proximity:
  - » E-business (e.g. Amazon)
  - » Content-servers (e.g., YouTube, Akamai, Apple, Microsoft)
  - » Search engines, data mining (e.g., Google)
- Challenges:
  - » Multiple applications, each serving massive numbers of clients
  - » Managing/balancing load, networking, data bottlenecks

- Load Balancer: application-layer routing
  - » Receives external client requests
  - » Direct workload within data centre
  - » Returns results to external client (hiding data centre internals from client)
- Rich interconnection among switches, racks:
  - » Increase throughout between racks (multiple routing paths possible)
  - » Increased reliability via redundancy

5.7 A Day in the Life of a Web Request

Page 495

Wireless/Mobile Networks and Security

6.1 Introduction

Elements in a wireless network:

- **Wireless hosts**
  - » Laptop, PDA, IP phone
  - » Run applications
  - » May be stationary (non-mobile) or mobile
    - Wireless does not always mean mobility
- **Wireless links**
  - » Typically used to connect mobile(s) to base station
  - » Also used as backbone link
  - » Multiple access protocol coordinates link access
  - » Various data rates, transmission distance
- **Base station**
  - » Typically connected to wire network
  - » Relay – responsible for sending packets between wire network and wireless host(s) in its "area"
    - E.g cell towers 802.11 access points

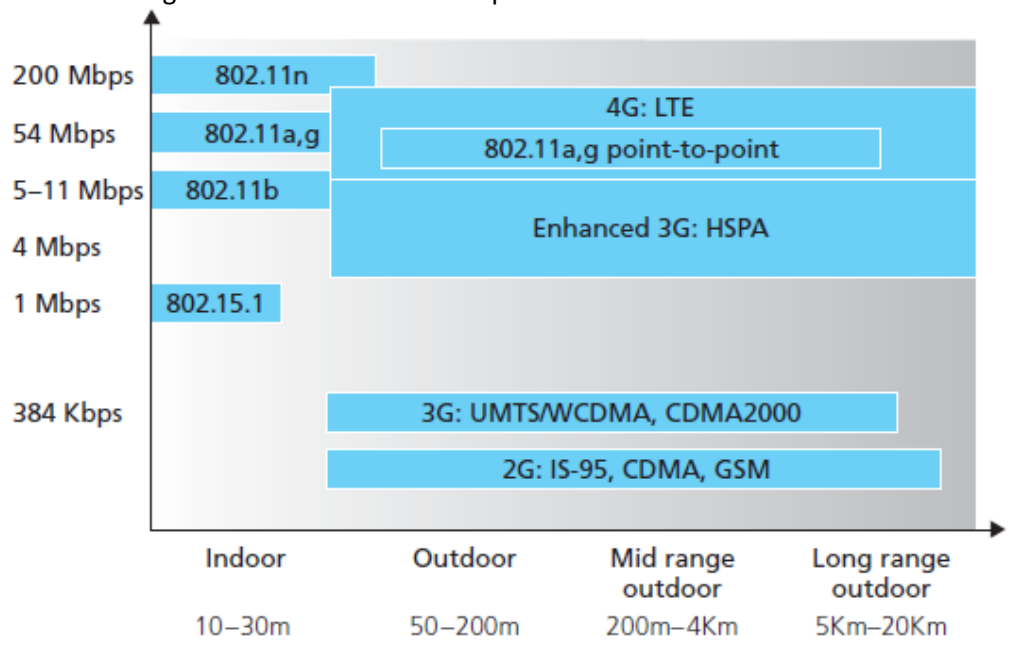


Figure 6.2 ♦ Link characteristics of selected wireless network standards

- Infrastructure mode
  - » Base station connects mobiles into wired network
  - » Handoff: mobile changes base station providing connection into wired network
- Ad Hoc mode
  - » No base stations
  - » Nodes can only transmit to other nodes within link coverage
  - » Nodes organise themselves into a network: route among themselves

	Single Hop	Multiple Hops
Infrastructure (e.g. APs)	Host connects to base station (Wi-Fi, WiMAX, cellular) which connects to larger Internet	Host may have to relay through several wireless nodes to connect to larger Internet: mesh networks
No infrastructure	No base station, no connection to larger Internet (Bluetooth, ad hoc nets)	No base station, no connection to larger Internet, may have to relay to reach a destination, may be mobile, a class of networks called MANET, VANET

6.2 Wireless Links & Network Characteristics

Differences between a wired link and wireless link which make wireless link much more "difficult":

- **Decreasing signal strength**
  - » Radio signal attenuates as it propagates through matter (path loss)
- **Interference from other sources**
  - » Standardised wireless network frequencies (e.g. 2.4 GHz) shared by other devices (e.g. phone); devices (motors) interfere as well
- **Multipath propagation**
  - » Radio signal reflects off objects and the ground, arriving at the destination at slightly different times

The **signal-to-noise ratio (SNR)** is a relative measure of the **strength** of the **received signal** and this **noise** (typically measured in units of decibels dB). The **larger the SNR**, the easier to extract signal from noise (a "good thing").

SNR vs. BER tradeoffs

- **Given physical layer**: increase power -> increase SNR->decrease BER (bit error rate)
- **Given SNR**: choose physical layer that meets BER requirement, giving highest throughput
  - » SNR may change with mobility: dynamically adapt physical layer (modulation technique, rate)

Code Division Multiple Access (CDMA)

- Used in several wireless broadcast channels (cellular, satellite, etc) standards
- Unique “code” assigned to each user; i.e., code set partitioning
- All users share same frequency, but each user has own “chipping” sequence (i.e., code) to encode data
- **Encoded signal** = (original data) x (chipping sequence)

- **Decoding**: inner-product of encoded signal and chipping sequence
- Allows multiple users to "coexist" and transmit simultaneously with minimal interference (if codes are "orthogonal")
- Assume original data are represented by 1 and -1
- **Encoded signal** = (original data) modulated by (chipping sequence)
  - » assume c_m = 1 1 1 -1 1 -1 -1 -1
  - » if data is 1, send 1 1 1 -1 1 -1 -1 -1
  - » if data is -1 send -1 -1 -1 1 -1 1 1 1
- **Decoding**: inner-product (summation of bit-by-bit product) of encoded signal and chipping sequence
  - » if inner-product > threshold, the data is 1; else -1
- CDMA codes are orthogonal
- E.g. (1,1,1,-1,1,-1,-1,-1) and (1,-1,1,1,1,-1,1,1)
- Inner product of the codes should be zero

C_1:	1	1	1	-1	1	-1	-1	-1
C_2:	1	-1	1	1	1	-1	1	1
$C_1 . C_2$ =	1 +(-1) + 1 + (-1)			+1 +	1+ (-1)+(-1)=0			

- If there are multiple CDMA codes all of the codes have to be orthogonal to each other.
  - » E.g. 3 codes: A, B and C. Then A x B = 0, B x C = 0 and A x C = 0

6.3 IEEE 802.11 wireless LANs ("Wi-Fi")

- **802.11b**
  - » 2.4-5 GHz unlicensed spectrum
  - » Up to 11 Mbps
  - » Direct sequence spread spectrum (DSSS) in physical layer
    - all hosts use same chipping code
- **802.11a**
  - » 5-6 GHz range
  - » Up to 54 Mbps
- **802.11g**
  - » 2.4-5 GHz range
  - » Up to 54 Mbps
- **802.11n**: multiple antennae
  - » 2.4-5 GHz range
  - » Up to 200 Mbps
  - » Standard ratified in Sept 2009

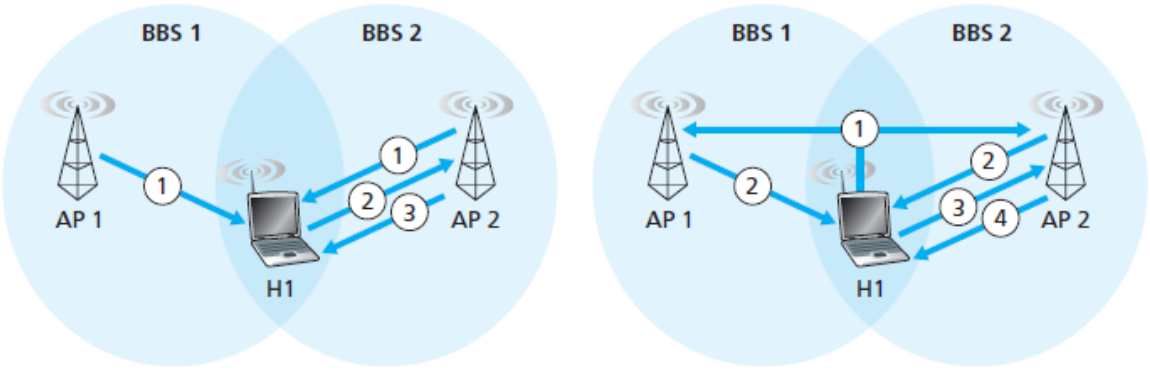
LAN Architecture

- Wireless host communicates with base station
  - » **Base station = access point (AP)**
- **Basic Service Set (BSS)** (aka "cell") in infrastructure mode contains:
  - » Wireless hosts
  - » Access Point (AP): base station
  - » Ad Hoc mode: hosts only

Channels and Association

- **802.11b:** 2.4GHz-2.485GHz spectrum divided into 11 channels at different frequencies
  - » AP admin chooses frequency for AP
  - » Interference possible: channel can be same as that chosen by neighbouring AP
  - » 3 non-overlapping channels in 802.11b: 1, 6 and 11
- **Host**: must associate with an AP
  - » Scans channels, listening for beacon frames containing AP's name (SSID) and MAC address
  - » Selects AP to associate with
  - » May perform authentication
  - » Will typically run DHCP to get IP address in AP's subnet

Passive & Active Scanning



Passive Scanning

1. beacon frames sent from APs (host scans for beacons on 11 channels)
2. association Request frame sent: H1 to selected AP
3. association Response frame sent: Selected AP to H1

Active Scanning

1. Probe Request frame broadcast from H1
2. Probes response frame sent from APs
3. Association Request frame sent: H1 to selected AP
4. Association Response frame sent: Selected AP to H1

Multiple Access

- Avoid collisions: 2^+ nodes transmitting at same time
- 802.11: CSMA - sense before transmitting
  - » Don't collide with ongoing transmission by other node
- 802.11: no collision detection! Why?
  - » Difficult to receive (sense collisions) when transmitting due to weak received signals (fading)
  - » Can't sense all collisions in any case: hidden terminal, fading
  - » Goal: avoid collisions: CSMA/C(ollision)A(voidance)

Network Security

8.1 What is Network Security?

- **Confidentiality**: only sender, intended receiver should "understand" message contents
  - » sender encrypts message
  - » receiver decrypts message
- **Authentication**: sender, receiver want to confirm identity of each other
- **Message Integrity**: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection
- **Access and Availability**: services must be accessible and available to users

8.2 Principles of Cryptography

Types of Cryptography

- **Public Key**
  - » Involves the use of two keys
- **Symmetric key**
  - » Involves the use of one key
  - » **Block cipher**
    - Break plaintext message in equal-size blocks
    - Encrypt each block as a unit
    - Used in many Internet protocols (PGP-secure email, SSL (secure TCP), IPSec (secure net-transport layer)
  - » **Stream ciphers**
    - Encrypt one bit at time
    - Used in secure WLAN
- **Hash Functions**
  - » Involves the use of no keys

8.3 Message Integrity and End-Point Authentication Sections

Message Integrity

- Allows communicating parties to verify that received messages are authentic
  - » Content of message has not been altered
  - » Source of message is who/what you think it is
  - » Message has not been replayed
  - » Sequence of messages is maintained

Message Digests

- Function H( ) that takes as input an arbitrary length message and outputs a fixed-length string: "message signature"
- Note that H( ) is a many-to-1 function
- H( ) is often called a "hash function"
- Desirable properties:
  - » Easy to calculate
  - » Irreversibility: Can't determine m from H(m)

- » Collision resistance: Computationally difficult to produce m and m' such that H(m) = H(m')
- » Seemingly random output

8.7 Wire Equivalent Privacy (WEP)

- Authentication as in protocol ap4.0
  - » Host requests authentication from access point
  - » Access point sends 128 bit nonce
  - » Host encrypts nonce using shared symmetric key
  - » Access point decrypts nonce, authenticates host
- No key distribution mechanism
- Authentication: knowing the shared key is enough

Data Encryption

- host/AP share 40 bit symmetric key (semi-permanent)
- host appends 24-bit initialization vector (IV) to create 64-bit key
- 64 bit key used to generate stream of keys, k_i^{IV}
- k_i^{IV} used to encrypt ith byte, d_i, in frame:

  $c_i = d_i$ XOR k_i^{IV}
- IV and encrypted bytes, c_i sent in frame
- New IV for each frame
- RC4 Algorithm : Input: 64 bit Key (Ks, IV), Ouput : a stream of keys