COMPUTER NETWORKS AND SECURITY

Module-1 **10 hours**

Application Layer:

Principles of Network Applications: Network Application Architectures, Processes Communicating, Transport Services Available to Applications, Transport Services Provided by the Internet, Application-Layer Protocols. The Web and HTTP: Overview of HTTP, Non-persistent and Persistent Connections, HTTP Message Format, User-Server Interaction: Cookies, Web Caching, The Conditional GET, File Transfer: FTP Commands & Replies, Electronic Mail in the Internet: SMTP, Comparison with HTTP, Mail Message Format, Mail Access Protocols, DNS; The Internet's Directory Service: Services Provided by DNS, Overview of How DNS Works, DNS Records and Messages, Peer-to-Peer Applications: P2P File Distribution, Distributed Hash Tables, Socket Programming: creating Network Applications: Socket Programming with UDP, Socket Programming with TCP.

Module-2 **10 hours**

Transport Layer :

Introduction and Transport-Layer Services: Relationship Between Transport and Network Layers, Overview of the Transport Layer in the Internet, Multiplexing and Demultiplexing: Connectionless Transport: UDP,UDP Segment Structure, UDP Checksum, Principles of Reliable Data Transfer: Building a Reliable Data Transfer Protocol, Pipelined Reliable Data Transfer Protocols, Go-Back-N, Selective repeat, Connection-Oriented Transport TCP: The TCP Connection, TCP Segment Structure, RoundTrip Time Estimation and Timeout, Reliable Data Transfer, Flow Control, TCP Connection Management, Principles of Congestion Control: The Causes and the Costs of Congestion, Approaches to Congestion Control, Network-assisted congestion-control example, ATM ABR Congestion control, TCP Congestion Control: Fairness.

Module-3 **10 hours**

The Network layer:

What's Inside a Router?: Input Processing, Switching, Output Processing, Where Does Queuing Occur? Routing control plane, IPv6,A Brief foray into IP Security, Routing Algorithms: The Link-State (LS) Routing Algorithm, The Distance-Vector (DV) Routing Algorithm, Hierarchical Routing, Routing in the Internet, Intra-AS Routing in the Internet: RIP, Intra-AS Routing in the Internet: OSPF, Inter/AS Routing: BGP, Broadcast Routing Algorithms and Multicast.

Module-4 **10 hours**

Network Security:

Overview of Network Security:Elements of Network Security , Classification of Network Attacks ,Security Methods ,Symmetric-Key Cryptography :Data Encryption Standard (DES),Advanced Encryption Standard (AES) , Public-Key Cryptography :RSA Algorithm ,Diffie-Hellman Key-Exchange Protocol , Authentication :Hash Function , Secure Hash Algorithm (SHA) , Digital Signatures , Firewalls and Packet Filtering ,Packet Filtering , Proxy Server .

Module-5 **10 hours**

Multimedia Networking:

Properties of video, properties of Audio, Types of multimedia Network Applications, Streaming stored video: UDP Streaming, HTTP Streaming, Adaptive streaming and DASH, content distribution Networks Voice-over-IP :Limitations of the Best-Effort IP Service ,Removing Jitter at the Receiver for Audio ,Recovering from Packet Loss Protocols for Real-Time Conversational Applications , RTP , SIP

Textbooks:

1. James F Kurose and Keith W Ross, Computer Networking, A Top-Down Approach, Sixth edition, Pearson,2017 .

2. Nader F Mir, Computer and Communication Networks, 2nd Edition, Pearson, 2014.

Reference Books:

1. Behrouz A Forouzan, Data and Communications and Networking, Fifth Edition, McGraw Hill, Indian Edition

2. Larry L Peterson and Brusce S Davie, Computer Networks, fifth edition, ELSEVIER

3. Andrew S Tanenbaum, Computer Networks, fifth edition, Pearson

4. Mayank Dave, Computer Networks, Second edition, Cengage Learning

MODULE 1: APPLICATION LAYER

1.1 Principles of Network Applications

- Network-applications are the driving forces for the explosive development of the internet.
- Examples of network-applications:
  1) Web
  2) File transfers
  3) E-mail
  4) P2P file sharing
  5) Social networking (Facebook, Twitter)
  6) Video distribution (YouTube)
  7) Real-time video conferencing (Skype)
  8) On-line games (World of War craft)
- In network-applications, program usually needs to
  → run on the different end-systems and
  → communicate with one another over the network.
- For ex: In the Web application, there are 2 different programs:
  1) The browser program running in the user's host (Laptop or Smartphone).
  2) The Web-server program running in the Web-server host.

1.1.1 Network Application Architectures

- Two approaches for developing an application:
  1) Client-Server architecture
  2) P2P (Peer to Peer) architecture

1.1.1.1 Client-Server Architecture

- In this architecture, there is a server and many clients distributed over the network (Figure 1.1a).
- The server is always-on while a client can be randomly run.
- The server is listening on the network and a client initializes the communication.
- Upon the requests from a client, the server provides certain services to the client.
- Usually, there is no communication between two clients.
- The server has a fixed IP address.
- A client contacts the server by sending a packet to the server's IP address.
- A server is able to communicate with many clients.
- The applications such as FTP, telnet, Web, e-mail etc use the client-server architecture.

1.1.1.1.1 Data Center

- Earlier, client-server architecture had a single-server host.
- But now, a single-server host is unable to keep up with all the requests from large no. of clients.
- For this reason, data-center a is used.
- A data-center contains a large number of hosts.
- A data-center is used to create a powerful virtual server.
- In date center, hundreds of servers must be powered and maintained.
- For example:
  - Google has around 50 data-centers distributed around the world.
  - These 50 data-centers handle search, YouTube, Gmail, and other services.

1.1.1.2 P2P Architecture

- There is no dedicated server (Figure 1.1b).
- Pairs of hosts are called peers.
- The peers communicate directly with each other.
- The peers are not owned by the service-provider. Rather, the peers are laptops controlled by users.
- Many of today's most popular and traffic-intensive applications are based on P2P architecture.
- Examples include file sharing (Bit Torrent), Internet telephone (Skype) etc.

- Main feature of P2P architectures: self-scalability.
- For ex: In a P2P file-sharing system,
  - Each peer generates workload by requesting files.
  - Each peer also adds service-capacity to the system by distributing files to other peers.
- Advantage: Cost effective '.' Normally, server-infrastructure & server bandwidth are not required.

- Three challenges of the P2P applications:

1) ISP Friendly

- Most residential ISPs have been designed for asymmetrical bandwidth usage.
- Asymmetrical bandwidth means there is more downstream-traffic than upstream-traffic.
- But P2P applications shift upstream-traffic from servers to residential ISPs, which stress onthe ISPs.

2) Security

- Since the highly distribution and openness, P2P applications can be a challenge to security.

3) Incentive

- Success of P2P depends on convincing users to volunteer bandwidth & resources to theapplications.

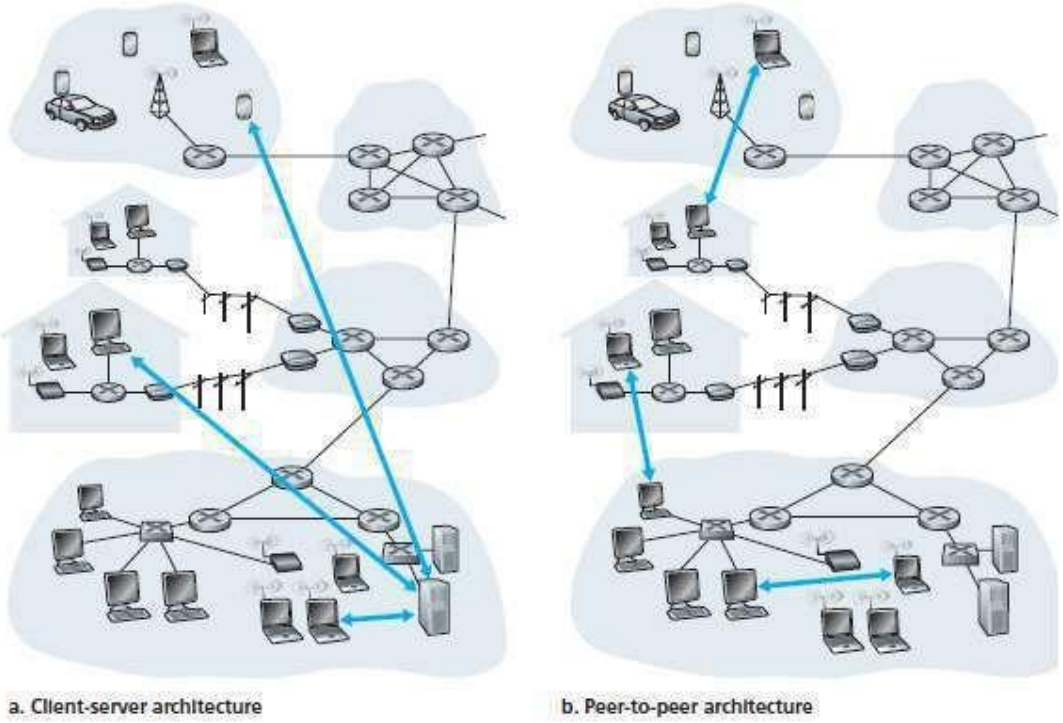


Figure 1.1: (a) Client-server architecture; (b) P2P architecture

1.1.2 Processes Communicating

1.1.2.1 Process

- A process is an instance of a program running in a computer.(IPC : inter-process communication).
- The processes may run on the 1) same system or 2) different systems.

1) The processes running on the same end-system can communicate with each other using IPC.

2) The processes running on the different end-systems can communicate by exchanging messages.
   i) A sending-process creates and sends messages into the network.
   ii) A receiving-process receives the messages and responds by sending messages back.

1.1.2.1.1 Client & Server Processes

- A network-application consists of pairs of processes:
  1) The process that initiates the communication is labeled as the client.
  2) The process that waits to be contacted to begin the session is labeled as the server.
- For example:
  1) In Web application, a client-browser process communicates with a Web-server-process.
  2) In P2P file system, a file is transferred from a process in one peer to a process in another peer.

1.1.2.1.2 Interface between the Process and the Computer Network Socket

- Any message sent from one process to another must go through the underlying-network.
- A process sends/receives message through a software-interface of underlying-network called socket.
- Socket is an API between the application-layer and the transport layer within a host (Figure 1.2).
- The application-developer has complete control at the application-layer side of the socket.
- But, the application-developer has little control of the transport-layer side of the socket.For ex: The application-developer can control:
  1) The choice of transport-protocol: TCP or UDP. (API : Application Programming Interface)
  2) The ability to fix parameters such as maximum-buffer & maximum-segment-sizes.

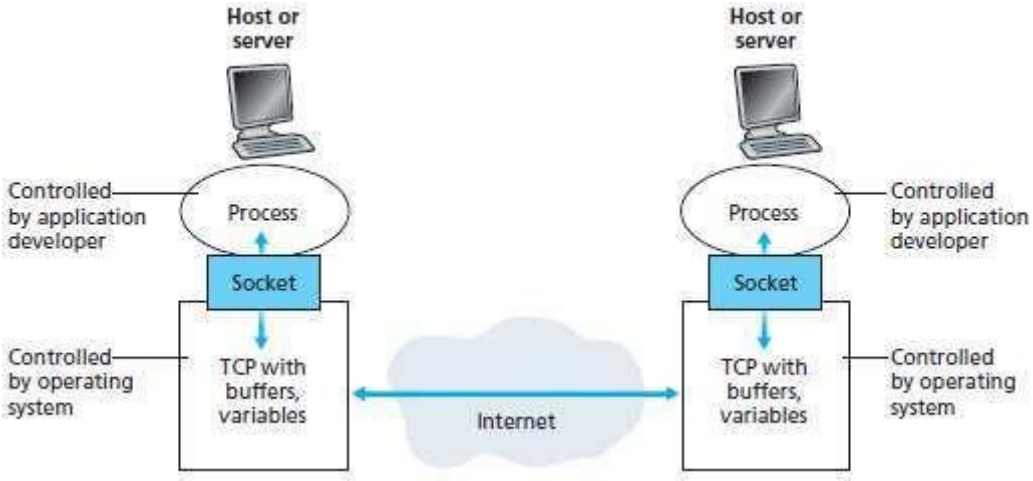


Figure 1.2: Application processes, sockets, and transport-protocol

1.1.2.1.3 Addressing Processes

- To identify the receiving-process, two pieces of information need to be specified:
  1) IP address of the destination-host.
  2) Port-number that specifies the receiving-process in the destination-host.
- In the Internet, the host is identified by IP address.
- An IP address is a 32-bit that uniquely identify the host.
- Sending-process needs to identify receiving-process '.' a host may run several network-

applications.
- For this purpose, a destination port-number is used.
- For example,
  A Web-server is identified by port-number 80.A mail-server is identified by port-number 2

1.1.3 Transport Services Available to Applications
- Networks usually provide more than one transport-layer protocols for different applications.
- An application-developer should choose certain protocol according to the type of applications.
- Different protocols may provide different services.

1.1.3.1 Reliable Data Transfer
- Reliable means guaranteeing the data from the sender to the receiver is delivered correctly.For ex: TCP provides reliable service to an application.
- Unreliable means the data from the sender to the receiver may never arrive.For ex: UDP provides unreliable service to an application.
- Unreliability may be acceptable for loss-tolerant applications, such as multimedia applications.
- In multimedia applications, the lost data might result in a small glitch in the audio/video.

1.1.3.2 Throughput
- Throughput is the rate at which the sending-process can deliver bits to the receiving-process.
- Since other hosts are using the network, the throughput can fluctuate with time.
- Two types of applications:

1) Bandwidth Sensitive Applications
- These applications need a guaranteed throughput.For ex: Multimedia applications
- Some transport-protocol provides guaranteed throughput at some specified rate (r bits/sec).

2) Elastic Applications
- These applications may not need a guaranteed throughput.For ex: Electronic mail, File transfer & Web transfers.

1.1.3.3 Timing
- A transport-layer protocol can provide timing-guarantees.
- For ex: guaranteeing every bit arrives at the receiver in less than 100 msec.
- Timing constraints are useful for real-time applications such as
  → Internet telephony
  → Virtual environments
  → Teleconferencing and
  → Multiplayer games

1.1.3.4 Security
- A transport-protocol can provide one or more security services.

- For example,
    1) In the sending host, a transport-protocol can encrypt all the transmitted-data.
    2) In the receiving host, the transport-protocol can decrypt the received-data.

1.1.4 Transport Services Provided by the Internet

- The Internet makes two transport-protocols available to applications, UDP and TCP.
- An application-developer who creates a new network-application must use either: UDP or TCP.
- Both UDP & TCP offers a different set of services to the invoking applications.
- Table 1.1 shows the service requirements for some selected applications.

Table 1.1: Requirements of selected network-applications

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 MbpsVideo: 10 kbps–5 Mbps	Yes: 100s of ms
Streaming stored audio/video	Loss-tolerant	Same as above	Yes: few seconds
Interactive games	Loss-tolerant	Few kbps–10 kbps	Yes: 100s of ms
Instant messaging	No loss	Elastic	Yes and no

1.1.4.1 TCP Services

- An application using transport-protocol TCP, receives following 2 services.
    1) Connection-Oriented Service
    - Before the start of communication, client & server need to exchange control-information.
    - This phase is called handshaking phase.
    - Then, the two processes can send messages to each other over the connection.
    - After the end of communication, the applications must tear down the connection.

    2) Reliable Data Transfer Service
    - The communicating processes must deliver all data sent without error & in the proper order.
- TCP also includes a congestion-control.
- The congestion-control throttles a sending-process when the network is congested.

1.1.4.2 UDP Services

- UDP is a lightweight transport-protocol, providing minimal services.
- UDP is connectionless, so there is no handshaking before the 2 processes start to communicate.
- UDP provides an unreliable data transfer service.
- Unreliable means providing no guarantee that the message will reach the receiving-process.
- Furthermore, messages that do arrive at the receiving-process may arrive out-of-order.
- UDP does not include a congestion-control.
- UDP can pump data into the network-layer at any rate.

1.2 The Web & HTTP

- The appearance of Web dramatically changed the Internet.
- Web has many advantages for a lot of applications.
- It operates on demand so that the users receive what they want when they want it.
- It provides an easy way for everyone make information available over the world.
- Hyperlinks and search engines help us navigate through an ocean of Web-sites.

- Forms, JavaScript, Java applets, and many other devices enable us to interact with pages and sites.
- The Web serves as a platform for many killer applications including YouTube, Gmail, and Facebook.

1.2.1 Overview of HTTP

1.2.1.1 Web

- A web-page consists of objects (HTML : Hyper Text Markup Language).
- An object is a file such as an HTML file, a JPEG image, a Java applet, a video chip.
- The object is addressable by a single URL (URL : Uniform Resource Locator).
- Most Web-pages consist of a base HTML file & several referenced objects.
- For example:

  If a Web-page contains HTML text and five JPEG images; then the Web-page has six objects:

  1) Base HTML file and
  2) Five images.
- The base HTML file references the other objects in the page with the object's URLs.
- URL has 2 components:

  1) The hostname of the server that houses the object and
  2) The object's path name.
- For example:

  "http://www.someSchool.edu/someDepartment/picture.gif"

  In above URL,

  1) Hostname = "www.someSchool.edu "
  2) Path name = "/someDepartment/picture.gif".
- The web browsers implement the client-side of HTTP. For ex: Google Chrome, Internet Explorer
- The web-servers implement the server-side of HTTP. For ex: Apache

1.2.1.2 HTTP

- HTTP is Web's application-layer protocol (Figure 1.3) (HTTP : HyperText Transfer Protocol).
- HTTP defines

  → how clients request Web-pages from servers and

  → how servers transfer Web-pages to clients.

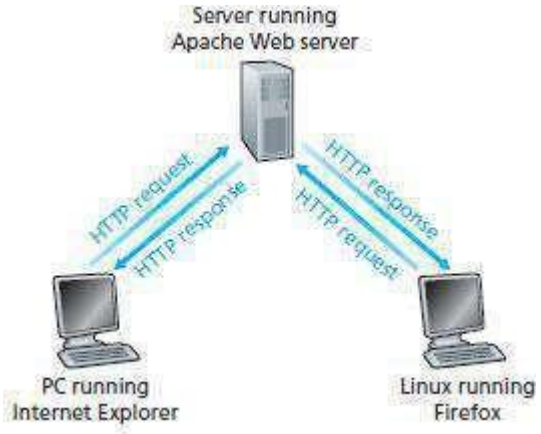


Figure 1.3: HTTP request-response behavior

- When a user requests a Web-page, the browser sends HTTP request to the server.
- Then, the server responds with HTTP response that contains the requested-objects.
- HTTP uses TCP as its underlying transport-protocol.

- The HTTP client first initiates a TCP connection with the server.
- HTTP is a stateless protocol.
- Stateless means the server sends requested-object to client w/o storing state-info about the client.
- HTTP uses the client-server architecture:
  1) **Client**
     - Browser that requests receive and displays Web objects.
  2) **Server**
     - Web-server sends objects in response to requests.

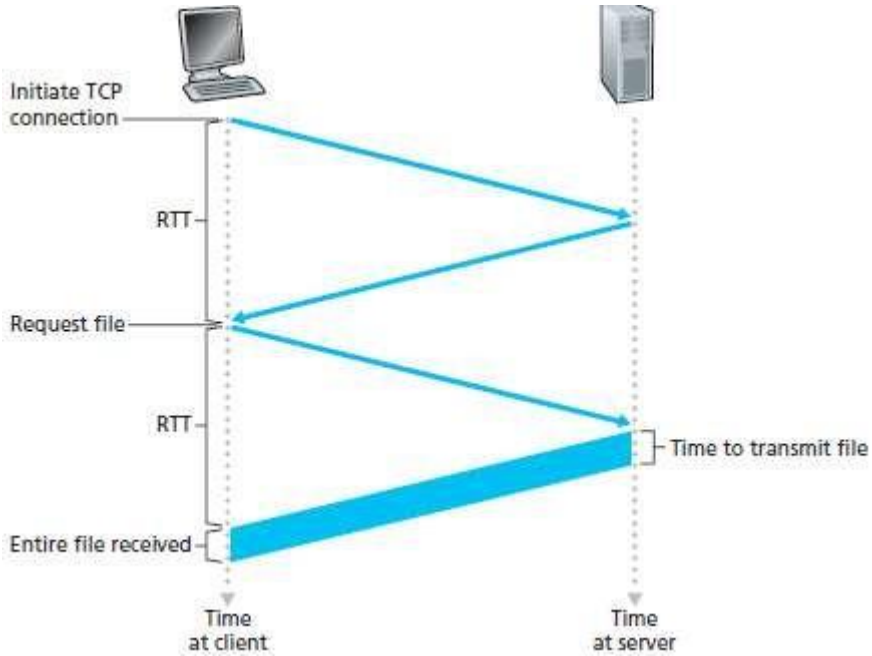
1.2.2 Non-Persistent & Persistent Connections

- In many internet applications, the client and server communicate for an extended period of time.
- When this client-server interaction takes place over TCP, a decision should be made:
  1) Should each request/response pair be sent over a separate TCP connection or
  2) Should all requests and their corresponding responses be sent over same TCP connection?
- These different connections are called non-persistent connections (1) or persistent connections (2).
- Default mode: HTTP uses persistent connections.

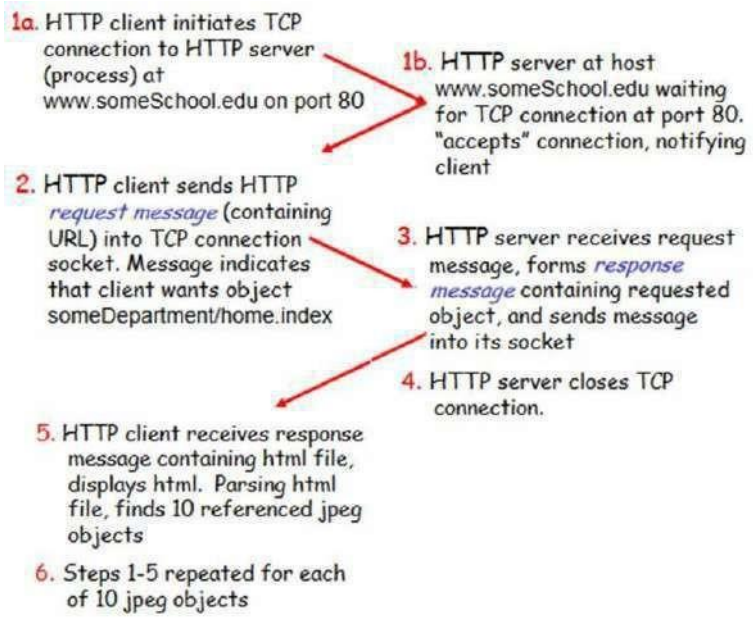
1.2.2.1 HTTP with Non-Persistent Connections

- A non-persistent connection is closed after the server sends the requested-object to the client.
- In other words, the connection is used exactly for one request and one response.
- For downloading multiple objects, multiple connections must be used.
- Suppose user enters URL:
  "http://www.someSchool.edu/someDepartment/home.index"
- Assume above link contains text and references to 10 jpeg images.

Figure 1.4: Back-of-the-envelope calculation for the time needed to request and receive an HTML file



- Here is how it works:



- RTT is the time taken for a packet to travel from client to server and then back to the client.
- The total response time is sum of following (Figure 1.4):
  i) One RTT to initiate TCP connection (RTT : Round Trip Time).
  ii) One RTT for HTTP request and first few bytes of HTTP response to return.
  iii) File transmission time.

  i.e. Total response time = (i) + (ii) + (iii) = 1 RTT+ 1 RTT+ File transmission time
  = 2(RTT) + File transmission time

1.2.2.2 HTTP with Persistent Connections

- Problem with Non-Persistent Connections:
  1) A new connection must be established and maintained for each requested-object.
     - Hence, buffers must be allocated and state info must be kept in both the client and server.
     - This results in a significant burden on the server.
  2) Each object suffers a delivery delay of two RTTs:
     i) One RTT to establish the TCP connection and
     ii) One RTT to request and receive an object.
- Solution: Use persistent connections.
- With persistent connections, the server leaves the TCP connection open after sending responses.
- Hence, subsequent requests & responses b/w same client & server can be sent over same connection
- The server closes the connection only when the connection is not used for a certain amount of time.
- Default mode of HTTP: Persistent connections with pipelining.
- Advantages:
  1) This method requires only one RTT for all the referenced-objects.
  2) The performance is improved by 20%.

1.2.3 HTTP Message Format

- Two types of HTTP messages: 1) Request-message and 2) Response-message.

1.2.3.1 HTTP Request Message

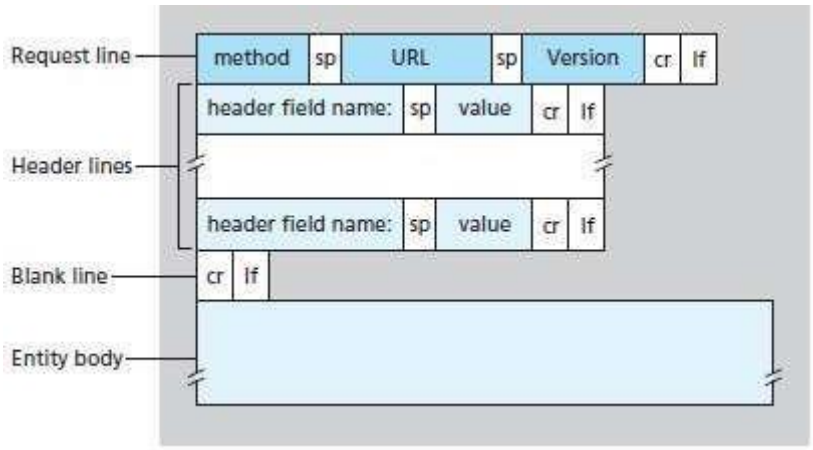


Figure 1.5: General format of an HTTP request-message

- An example of request-message is as follows:

```
GET   /somedir/page.html    HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: eng
```

- The request-message contains 3 sections (Figure 1.5):
  1) Request-line
  2) Header-line and
  3) Carriage return.
- The first line of message is called the request-line. The subsequent lines are called the header-lines.
- The request-line contains 3 fields. The meaning of the fields is as follows:
  1) Method
     - "GET": This method is used when the browser requests an object from the server.
  2) URL
     - "/somedir/page.html": This is the object requested by the browser.
  3) Version
     - "HTTP/1.1": This is version used by the browser.
- The request-message contains 4 header-lines. The meaning of the header-lines is as follows:
  1) "Host: www.someschool.edu" specifies the host on which the object resides.
  2) "Connection: close" means requesting a non-persistent connection.
  3) "User-agent:Mozilla/5.0" means the browser used is the Firefox.
  4) "Accept-language:eng" means English is the preferred language.
- The method field can take following values: GET, POST, HEAD, PUT and DELETE.
  1) GET is used when the browser requests an object from the server.
  2) POST is used when the user fills out a form & sends to the server.
  3) HEAD is identical to GET except the server must not return a message-body in the response.
  4) PUT is used to upload objects to servers.
  5) DELETE allows an application to delete an object on a server.

1.2.3.2 HTTP Response Message

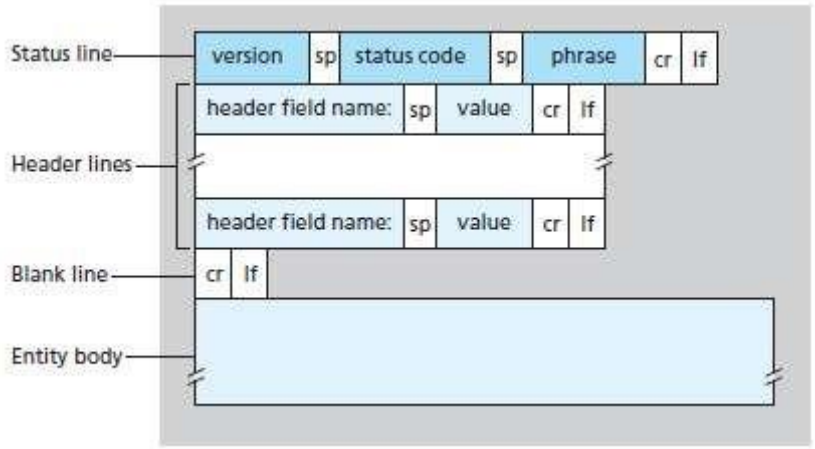


Figure 1.6: General format of an HTTP response-message

- An example of response-message is as follows:

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 09 Aug 2011 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 09 Aug 2011 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html
(data data data data data ...)
```

- The response-message contains 3 sections (Figure 1.6):
  1) Status line
  2) Header-lines and
  3) Data (Entity body).
- The status line contains 3 fields:
  1) Protocol version
  2) Status-code and
  3) Status message.
- Some common status-codes and associated messages include:
  1) 200 OK: Standard response for successful HTTP requests.
  2) 400 Bad Request: The server cannot process the request due to a client error.
  3) 404 Not Found: The requested resource cannot be found.
- The meaning of the Status line is as follows:
  "HTTP/1.1 200 OK": This line indicates the server is using HTTP/1.1 & that everything is OK.
- The response-message contains 6 header-lines. The meaning of the header-lines is as follows:
  1) Connection: This line indicates browser requesting a non-persistent connection.
  2) Date: This line indicates the time & date when the response was sent by the server.
  3) Server: This line indicates that the message was generated by an Apache Web-server.
  4) Last-Modified: This line indicates the time & date when the object was last modified.
  5) Content-Length: This line indicates the number of bytes in the sent-object.

6) Content-Type: This line indicates that the object in the entity body is HTML text.

1.2.4 User-Server Interaction: Cookies

- Cookies refer to a small text file created by a Web-site that is stored in the user's computer.
- Cookies are stored either temporarily for that session only or permanently on the hard disk.
- Cookies allow Web-sites to keep track of users.
- Cookie technology has four components:
  1) A cookie header-line in the HTTP response-message.
  2) A cookie header-line in the HTTP request-message.
  3) A cookie file kept on the user's end-system and managed by the user's browser.
  4) A back-end database at the Web-site.

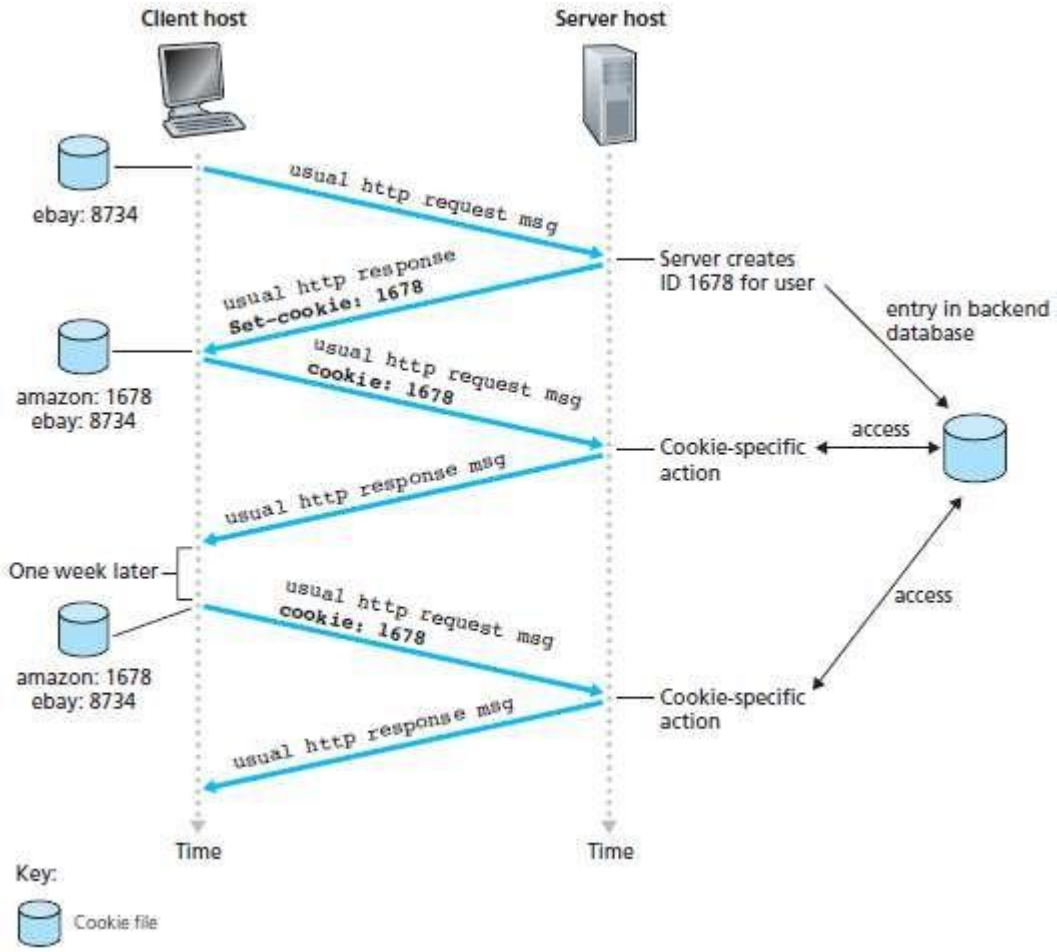


Figure 1.7: Keeping user state with cookies

- Here is how it works (Figure 1.7):
  1) When a user first time visits a site, the server
     → creates a unique identification number (1678) and
     → creates an entry in its back-end database by the identification number.
  2) The server then responds to user's browser.
  - HTTP response includes Set-cookie: header which contains the identification number (1678)
  3) The browser then stores the identification number into the cookie-file.
  4) Each time the user requests a Web-page, the browser
     → extracts the identification number from the cookie file, and
     → puts the identification number in the HTTP request.

5) In this manner, the server is able to track user's activity at the web-site.

1.2.5 Web Caching

- A Web-cache is a network entity that satisfies HTTP requests on the behalf of an original Web-server.
- The Web-cache has disk-storage.
- The disk-storage contains copies of recently requested-objects.

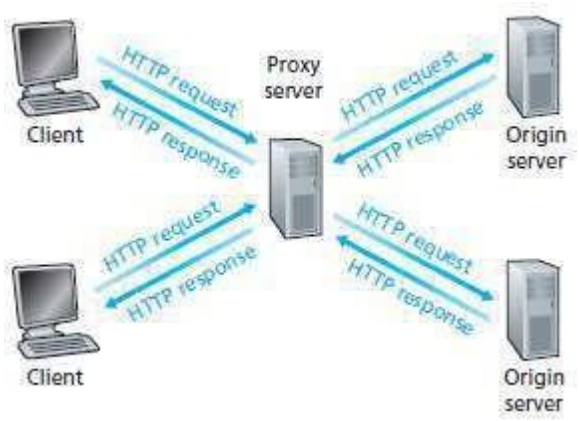


Figure 1.8: Clients requesting objects through a Web-cache (or Proxy Server)

- Here is how it works (Figure 1.8):
  1) The user's HTTP requests are first directed to the web-cache.
  2) If the cache has the object requested, the cache returns the requested-object to the client.
  3) If the cache does not have the requested-object, then the cache
     → connects to the original server and
     → asks for the object.
  4) When the cache receives the object, the cache
     → stores a copy of the object in local-storage and
     → sends a copy of the object to the client.
- A cache acts as both a server and a client at the same time.
  1) The cache acts as a server when the cache
     → receives requests from a browser and
     → sends responses to the browser.
  2) The cache acts as a client when the cache
     → requests to an original server and
     → receives responses from the origin server.
- Advantages of caching:
  1) To reduce response-time for client-request.
  2) To reduce traffic on an institution's access-link to the Internet.
  3) To reduce Web-traffic in the Internet.

1.2.6 The Conditional GET

- Conditional GET refers a mechanism that allows a cache to verify that the objects are up to date.
  - An HTTP request-message is called conditional GET if
    1) Request-message uses the GET method and
    2) Request-message includes an If-Modified-Since: header-line.
- The following is an example of using conditional GET:

```
GET /fruit/kiwi.fig HTTP1.1
Host: www.exoriguecuisine.com
If-modified-since: Wed, 7 Sep 2011 09:23:24
```

- The response is:

```
HTTP/1.1 304 Not Modified
Date: Sat, 15 Oct 2011 15:39:29
```

1.3 File Transfer: FTP

- FTP is used by the local host to transfer files to or from a remote-host over the network.
- FTP uses client-server architecture (Figure 1.9).
- FTP uses 2 parallel TCP connections (Figure 1.10):

1) Control Connection

- The control-connection is used for sending control-information b/w local and remote-hosts.
- The control-information includes:
  - → user identification
  - → password
  - → commands to change directory and
  - → commands to put & get files.

2) Data Connection

- The data-connection is used to transfer files.

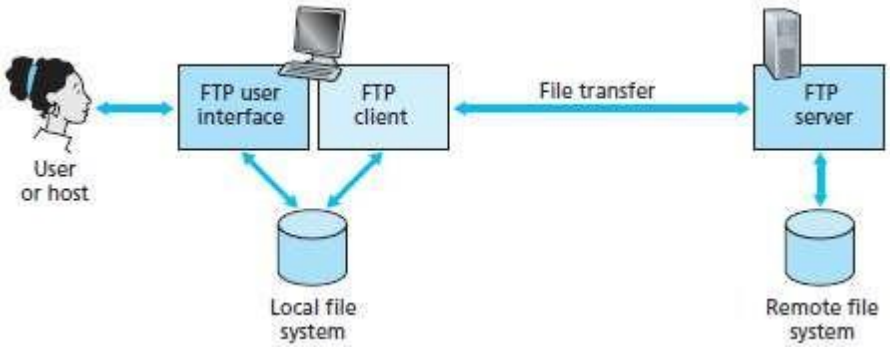


Figure 1.9: FTP moves files between local and remote file systems

Figure 1.10: Control and data-connections

- Here is how it works:
  1) When session starts, the client initiates a control-connection with the server on port 21.
  2) The client sends user-identity and password over the control-connection.
  3) Then, the server initiates data-connection to the client on port 20.
  4) FTP sends exactly one file over the data-connection and then closes the data-connection.
  5) Usually, the control-connection remains open throughout the duration of the user-session.

6) But, a new data-connection is created for each file transferred within a session.

- During a session, the server must maintain the state-information about the user.
- For example:
  The server must keep track of the user's current directory.
- Disadvantage:
  Keeping track of state-info limits the no. of sessions maintained simultaneously by a server.

1.3.1 FTP Commands & Replies

- The commands are sent from client to server.
- The replies are sent from server to client.
- The commands and replies are sent across the control-connection in 7-bit ASCII format.
- Each command consists of 4-uppercase ASCII characters followed by optional arguments.
- For example:
  1) USER username
  - Used to send the user identification to the server.
  2) PASS password
  - Used to send the user password to the server.
  3) LIST
  - Used to ask the server to send back a list of all the files in the current remote directory.
  4) RETR filename
  - Used to retrieve a file from the current directory of the remote-host.
  5) STOR filename
  - Used to store a file into the current directory of the remote-host.
- Each reply consists of 3-digit numbers followed by optional message.
- For example:
  1) 331 Username OK, password required
  2) 125 Data-connection already open; transfer starting
  3) 425 Can't open data-connection
  4) 452 Error writing file

1.4 Electronic Mail in the Internet

- e-mail is an asynchronous communication medium in which people send and read messages.
- e-mail is fast, easy to distribute, and inexpensive.
- e-mail has features such as
  → messages with attachments
  → hyperlinks
  → HTML-formatted text and
  → embedded photos.
- Three major components of an e-mail system (Figure 1.11):
  1) User Agents
  - User-agents allow users to read, reply to, forward, save and compose messages.
  - For example: Microsoft Outlook and Apple Mail

  2) Mail Servers
  - Mail-servers contain mailboxes for users.
  - A message is first sent to the sender's mail-server.
  - Then, the sender's mail-server sends the message to the receiver's mail-server.
  - If the sender's server cannot deliver mail to receiver's server, the sender's server
    → holds the message in a message queue and
    → attempts to transfer the message later.

  3) SMTP (Simple Mail Transfer Protocol)
  - SMTP is an application-layer protocol used for email.
  - SMTP uses TCP to transfer mail from the sender's mail-server to the recipient's mail-server.

- SMTP has two sides:
  1) A client-side, which executes on the sender's mail-server.
  2) A server-side, which executes on the recipient's mail-server.
- Both the client and server-sides of SMTP run on every mail-server.
- When a mail-server receives mail from other mail-servers, the mail-server acts as a server.When a mail-server sends mail to other mail-servers, the mail-server acts as a client.

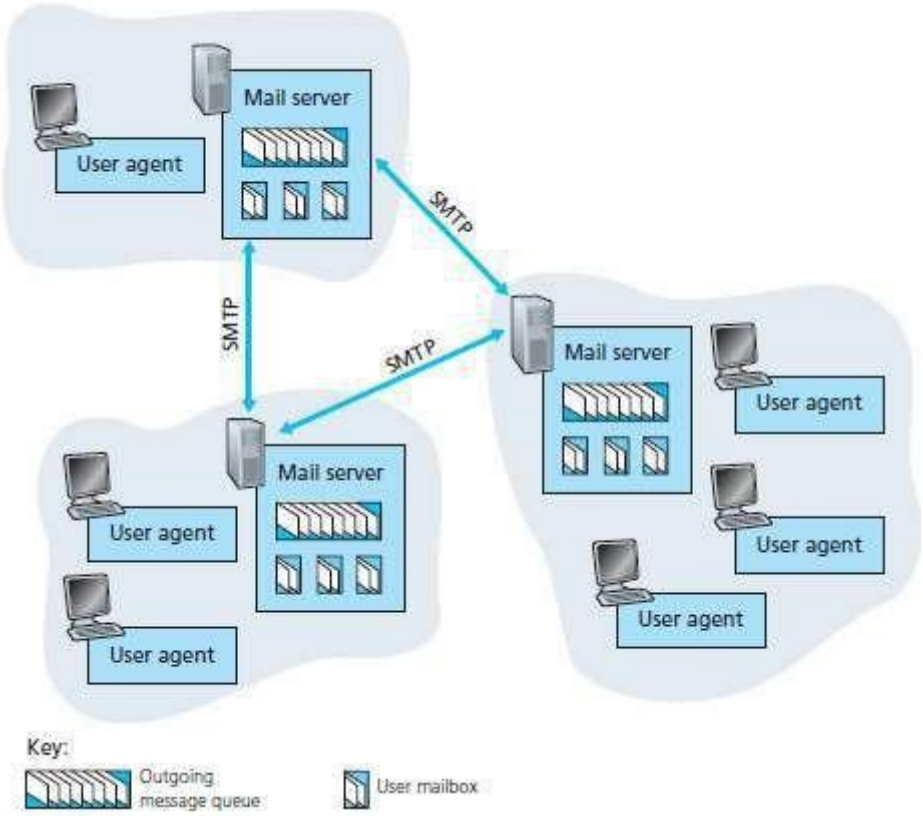


Figure 1.11: A high-level view of the Internet e-mail system

1.4.1 SMTP

- SMTP is the most important protocol of the email system.
- Three characteristics of SMTP (that differs from other applications):
  1) Message body uses 7-bit ASCII code only.
  2) Normally, no intermediate mail-servers used for sending mail.
  3) Mail transmissions across multiple networks through mail relaying.
- Here is how it works:
  1) Usually, mail-servers are listening at port 25.
  2) The sending server initiates a TCP connection to the receiving mail-server.
  3) If the receiver's server is down, the sending server will try later.
  4) If connection is established, the client & the server perform application-layer handshaking.
  5) Then, the client indicates the e-mail address of the sender and the recipient.
  6) Finally, the client sends the message to the server over the same TCP connection.

1.4.2 Comparison of SMTP with HTTP

1) HTTP is mainly a pull protocol. This is because
   → someone loads information on a web-server and
   → users use HTTP to pull the information from the server.

- On the other hand, SMTP is primarily a push protocol. This is because
  → the sending mail-server pushes the file to receiving mail-server.

2) SMTP requires each message to be in seven-bit ASCII format.
- If message contains binary-data, the message has to be encoded into 7-bit ASCII format.
- HTTP does not have this restriction.

3) HTTP encapsulates each object of message in its own response-message.
- SMTP places all of the message's objects into one message.

1.4.3 Mail Access Protocols

- It is not realistic to run the mail-servers on PC & laptop. This is because
  → mail-servers must be always-on and
  → mail-servers must have fixed IP addresses
- Problem: How a person can access the email using PC or laptop?
- Solution: Use mail access protocols.
- Three mail access protocols:
  1) Post Office Protocol (POP)
  2) Internet Mail Access Protocol (IMAP) and
  3) HTTP.

1.4.3.1 POP

- POP is an extremely simple mail access protocol.
- POP server will listen at port 110.
- Here is how it works:
  - The user-agent at client's computer opens a TCP connection to the main server.
  - POP then progresses through three phases:

  1) Authentication
  - The user-agent sends a user name and password to authenticate the user.

  2) Transaction
  - The user-agent retrieves messages.
  - Also, the user-agent can
    → mark messages for deletion
    → remove deletion marks &
    → obtain mail statistics.
  - The user-agent issues commands, and the server responds to each command with a reply.
  - There are two responses:
    i) +OK: used by the server to indicate that the previous command was fine.
    ii) –ERR: used by the server to indicate that something is wrong.

  3) Update
  - After user issues a quit command, the mail-server removes all messages marked for deletion.
- Disadvantage:
  The user cannot manage the mails at remote mail-server. For ex: user cannot delete messages.

1.4.3.2 IMAP

- IMAP is another mail access protocol, which has more features than POP.
- An IMAP server will associate each message with a folder.
- When a message first arrives at server, the message is associated with recipient's INBOX folder
- Then, the recipient can
  → move the message into a new, user-created folder
  → read the message

→ delete the message and
→ search remote folders for messages matching specific criteria.

- An IMAP server maintains user state-information across IMAP sessions.
- IMAP permits a user-agent to obtain components of messages.
  For example, a user-agent can obtain just the message header of a message.

1.4.3.3 Web-Based E-Mail

- HTTPs are now used for Web-based email accessing.
- The user-agent is an ordinary Web browser.
- The user communicates with its remote-server via HTTP.
- Now, Web-based emails are provided by many companies including Google, Yahoo etc.

1.5 DNS — The Internet's Directory Service

- DNS is an internet service that translates domain-names into IP addresses.
  For ex: the domain-name "www.google.com" might translate to IP address "198.105.232.4".
- Because domain-names are alphabetic, they are easier to remember for human being.
- But, the Internet is really based on IP addresses (DNS Domain Name System).

1.5.1 Services Provided by DNS

- The DNS is
  1) A distributed database implemented in a hierarchy of DNS servers.
  2) An application-layer protocol that allows hosts to query the distributed database.
- DNS servers are often UNIX machines running the BIND software.
- The DNS protocol runs over UDP and uses port 53. (BIND Berkeley Internet Name Domain)
- DNS is used by application-layer protocols such as HTTP, SMTP, and FTP.
- Assume a browser requests the URL www.someschool.edu/index.html.
- Next, the user's host must first obtain the IP address of www.someschool.edu
- This is done as follows:
  1) The same user machine runs the client-side of the DNS application.
  2) The browser
     → extracts the hostname "www.someschool.edu" from the URL and
     → passes the hostname to the client-side of the DNS application.
  3) The client sends a query containing the hostname to a DNS server.
  4) The client eventually receives a reply, which includes the IP address for the hostname.
  5) After receiving the IP address, the browser can initiate a TCP connection to the HTTP server.
- DNS also provides following services:
  1) Host Aliasing
  - A host with a complicated hostname can have one or more alias names.

  2) Mail Server Aliasing
  - For obvious reasons, it is highly desirable that e-mail addresses be mnemonic.

  3) Load Distribution
  - DNS is also used to perform load distribution among replicated servers.
  - Busy sites are replicated over multiple servers & each server runs on a different system.

1.5.2 Overview of How DNS Works

- Distributed database design is more preferred over centralized design because:
  1) A Single Point of Failure
  - If the DNS server crashes then the entire Internet will not stop.

  2) Traffic Volume
  - A Single DNS Server cannot handle the huge global DNS traffic.

- But with distributed system, the traffic is distributed and reduces overload on server.

3) Distant Centralized Database

- A single DNS server cannot be "close to" all the querying clients.
- If we put the single DNS server in Mysore,
  then all queries from USA must travel to the other side of the globe.
- This can lead to significant delays.

4) Maintenance

- The single DNS server would have to keep records for all Internet hosts.
- This centralized database has to be updated frequently to account for every new host.

1.5.2.1 A Distributed, Hierarchical Database

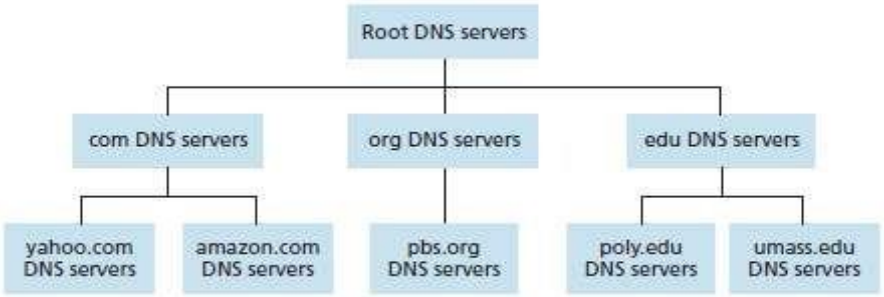


Figure 1.12: Portion of the hierarchy of DNS servers

- Suppose a client wants to determine IP address for hostname "www.amazon.com" (Figure 1.12):
  1) The client first contacts one of the root servers, which returns IP addresses for TLD servers
  2) Then, the client contacts one of these TLD servers.
     - The TLD server returns the IP address of an authoritative-server for "amazon.com".
  3) Finally, the client contacts one of the authoritative-servers for amazon.com.
     - The authoritative-server returns the IP address for the hostname "www.amazon.com".

1.5.2.1.1 Recursive Queries & Iterative Queries

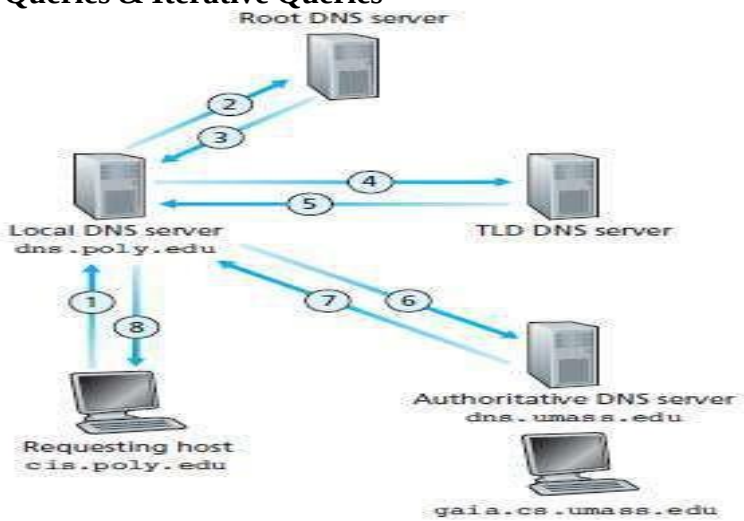


Figure 1.13: Interaction of the various DNS servers

- The example shown in Figure 1.13 makes use of both recursive queries and iterative queries.
- The query 1 sent from cis.poly.edu to dns.poly.edu is a recursive query. This is because
  → the query asks dns.poly.edu to obtain the mapping on its behalf.
- But the subsequent three queries 2, 4 and 6 are iterative. This is because
  → all replies are directly returned to dns.poly.edu.

1.5.3 DNS Records & Messages

- The DNS server stores resource-records (RRs).
- RRs provide hostname-to-IP address mappings.
- Each DNS reply message carries one or more resource-records.
- A resource-record is a 4-tuple that contains the following fields: (Name, Value, Type, TTL)
- TTL (time to live) determines when a resource should be removed from a cache.
- The meaning of Name and Value depend on Type:
  1) If Type=A, then Name is a hostname and Value is the IP address for the hostname.
  - Thus, a Type A record provides the standard hostname-to-IP address mapping.For ex: (relay1.bar.foo.com, 145.37.93.126, A)

  2) If Type=NS, then
     i) Name is a domain (such as foo.com) and
     ii) Value is the hostname of an authoritative DNS server.
  - This record is used to route DNS queries further along in the query chain.For ex: (foo.com, dns.foo.com, NS) is a Type NS record.

  3) If Type=CNAME, then Value is a canonical hostname for the alias hostname Name.
  - This record can provide querying hosts the canonical name for a hostname.For ex: (foo.com, relay1.bar.foo.com, CNAME) is a CNAME record.

  4) If Type=MX, Value is the canonical name of a mail-server that has an alias hostname Name.
  - MX records allow the hostnames of mail-servers to have simple aliases.For ex: (foo.com, mail.bar.foo.com, MX) is an MX record.

1.5.3.1 DNS Messages

- Two types of DNS messages: 1) query and 2) reply.
- Both query and reply messages have the same format.

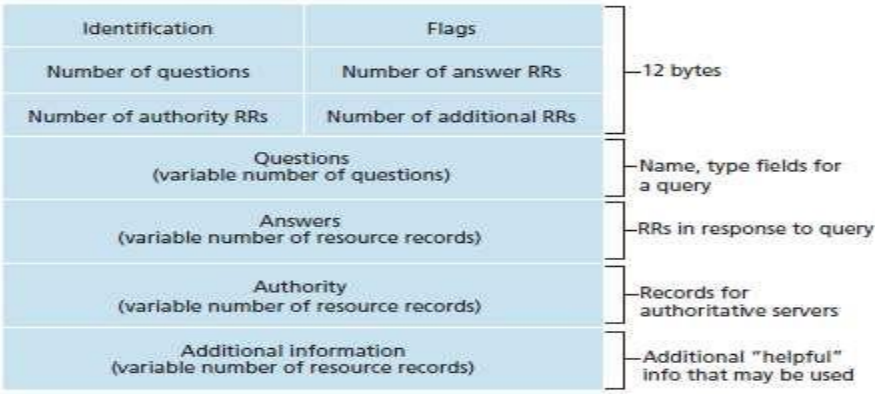


Figure 1.14: DNS message format

- The various fields in a DNS message are as follows (Figure 1.14):

1) Header Section

- The first 12 bytes is the header-section.
- This section has following fields:

i) Identification

- This field identifies the query.
- This identifier is copied into the reply message to a query.
- This identifier allows the client to match received replies with sent queries.

ii) Flag

- This field has following 3 flag-bits:

a) Query/Reply

¤ This flag-bit indicates whether the message is a query (0) or a reply (1).

b) Authoritative

¤ This flag-bit is set in a reply message when a DNS server is an authoritative-server.

c) Recursion Desired

¤ This flag-bit is set when a client desires that the DNS server perform recursion.

iii) Four Number-of-Fields

- These fields indicate the no. of occurrences of 4 types of data sections that follow the header.

2) Question Section

- This section contains information about the query that is being made.
- This section has following fields:

i) Name

- This field contains the domain-name that is being queried.

ii) Type

- This field indicates the type of question being asked about the domain-name.

3) Answer Section

- This section contains a reply from a DNS server.
- This section contains the resource-records for the name that was originally queried.
- A reply can return multiple RRs in the answer, since a hostname can have multiple IP addresses.

4) Authority Section

- This section contains records of other authoritative-servers.

5) Additional Section

- This section contains other helpful records.

1.6 Peer-to-Peer Applications

- Peer-to-peer architecture is different from client-server architecture.
- In P2P, each node (called peers) acts as a client and server at the same time.
- The peers are not owned by a service-provider.
- The peers not supposed to be always listening on the Internet.
- The peers are dynamic, i.e., some peers will join some peers will leave from time to time.

1.6.1 P2P File Distribution

- One popular P2P file distribution protocol is BitTorrent
- Consider the following scenarios:

Suppose a server has a large file and 'N' computers want to download the file (Figure 1.15).

1) In client-server architecture, each of the N computers will
   → connect to the server &
   → download a copy of the file to local-host.
2) In P2P architecture, a peer need not necessarily download a copy from the server.

- Rather, the peer may download from other peers.

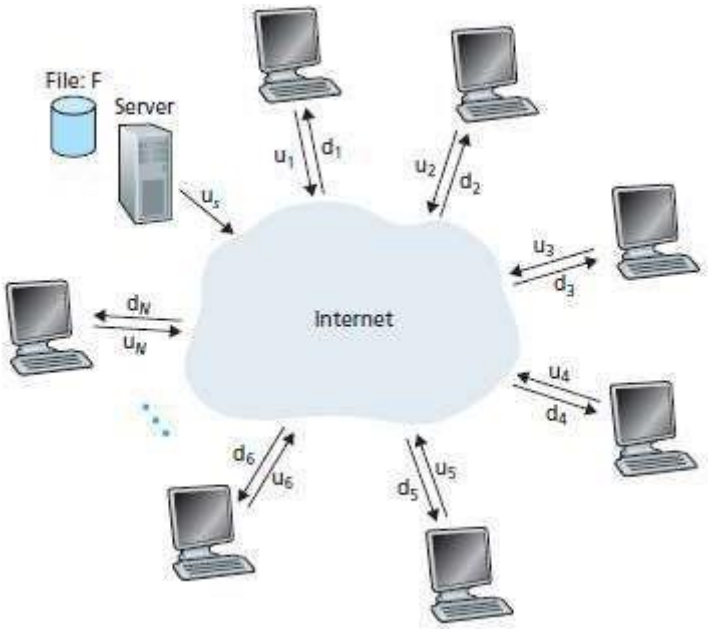


Figure 1.15: An illustrative file distribution problem

Let us define the following:
Length of the file = F
Upload rate for the server = u_S
Upload rate for ith computer = u_i
Download rate for ith computer = d_i
Distribution-time for the client-server architecture = D_{CS}
Server needs transmit = N_F bits.
Lowest download-rate of peer = d_{min}
Distribution-time for the P2P architecture
= DP2P

Case 1: Client-Server Architecture

- We have 2 observations:
  1) The server must transmit one copy of the file to each of the N peers.
     - Since the server's upload rate is u_S, the distribution-time is at least N_F/u_S.
  2) The peer with the lowest download-rate cannot obtain all F bits of the file in less than F/d_{min}.
     - Thus, the minimum distribution-time is at least F/ d_{min}.
- Putting above 2 observations together, we have

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

Case 2: P2P Architecture

- We have 2 observations:
  1) At the beginning of the distribution, only the server has the file.
     - So, the minimum distribution-time is at least F/u_S.
  2) The peer with the lowest download-rate cannot obtain all F bits of the file in less than F/d_{min}.
     - Thus, the minimum distribution-time is at least F/ d_{min}.
  3) The total upload capacity of the system as a whole is $u_{total} = u_S + u1 + u2 \ldots + uN$.
     - The system must deliver F bits to each of the N peers.
     - Thus, the minimum distribution-time is at least $NF/(u_S + u1 + u2 \ldots + uN)$.

- Putting above 3 observations together, we have

$$D_{P2P} \geq \max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\}$$

- Figure 1.16 compares the minimum distribution-time for the client-server and P2P architectures.

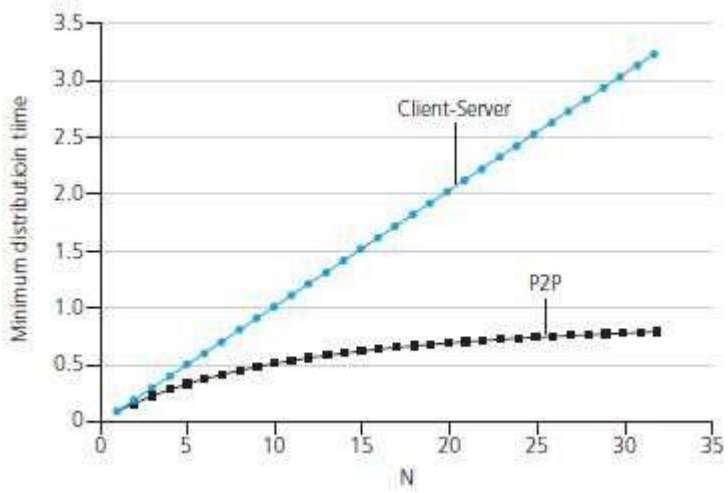


Figure 1.16: Distribution-time for P2P and client-server architectures

- The above comparison shows that when N is large,
  i) P2P architecture is consuming less distribution-time.
  ii) P2P architecture is self-scaling.

1.6.1.1 BitRorrent

- The collection of all peers participating in the distribution of a particular file is called a torrent.
- Peers download equal-size chunks of the file from one another. Chunk size = 256 KBytes.
- The peer also uploads chunks to other peers.
- Once a peer has acquired the entire file, the peer may leave the torrent or remain in the torrent.
- Each torrent has an infrastructure node called tracker.
- Here is how it works (Figure 1.17):
  1) When a peer joins a torrent, the peer
     → registers itself with the tracker and
     → periodically informs the tracker that it is in the torrent.
  2) When a new peer joins the torrent, the tracker
     → randomly selects a subset of peers from the set of participating peers and
     → sends the IP addresses of these peers to the new peer.
  3) Then, the new peer tries to establish concurrent TCP connections with all peers on this list.
  ➢ All peers on the list are called neighboring-peers.
  4) Periodically, the new peer will ask each of the neighboring-peers for the set of chunks.
- To choose the chunks to download, the peer uses a technique called rarest-first.
- Main idea of rarest-first:
  → Determine the chunks that are the rarest among the neighbors and
  → Request then those rarest chunks first.

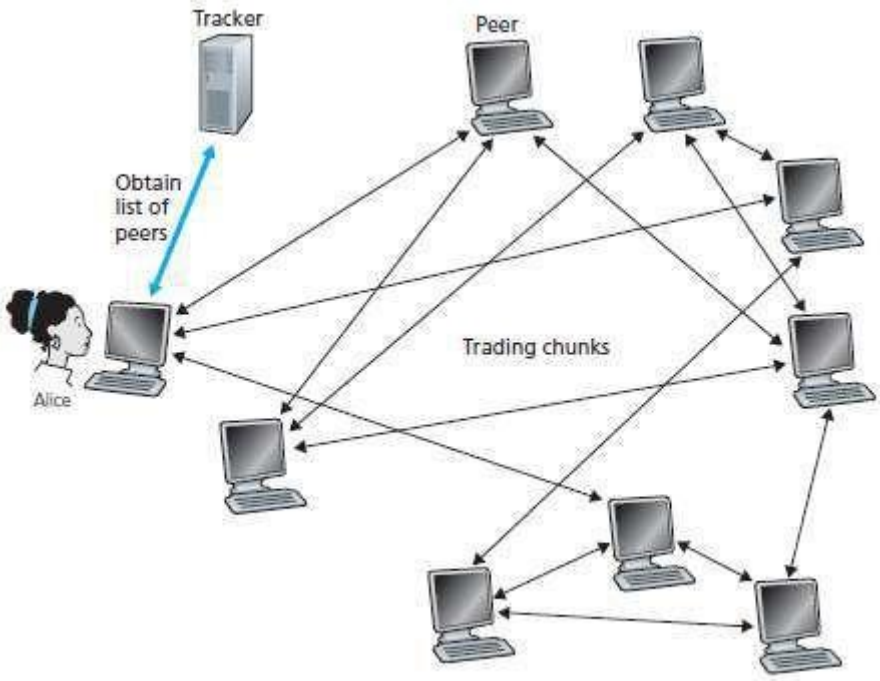


Figure 1.17: File distribution with BitTorrent

1.6.2 Distributed Hash Table

- We can use P2P architecture to form a distributed database.
- We consider a simple database, which contains (key, value) pairs.
- Each peer will only hold a small subset of the data.
- Any peer can query the distributed database with a particular key.
- Then, the database will
  → locate the peers that have the corresponding (key, value) pairs and
  → return the key-value to the querying peer.
- Any peer will also be allowed to insert new key-value pairs into the database.
- Such a distributed database is referred to as a distributed hash table (DHT).
- To construct the database, we need to use some hash-functions.
- The input of a hash-function can be a large number.
  But the output of the hash-function is of fixed-size bit-string.
- The outline of building a DHT is as follows:
  1) Assign an identifier to each peer, where the identifier is an n-bit string.
  ➢ So, we can view the identifier as an integer at the range from 0 to 2^n-1.
  2) For a data pair (key, value), the hash value of the key is computed.
  ➢ Then the data is stored in the peer whose identifier is closest to the key.
  3) To insert or retrieve data, first we need to find the appropriate peer.
  ➢ Problem: It is not realistic to let the peer to store all of the other peer's identifiers.Solution: Use a circular arrangement.

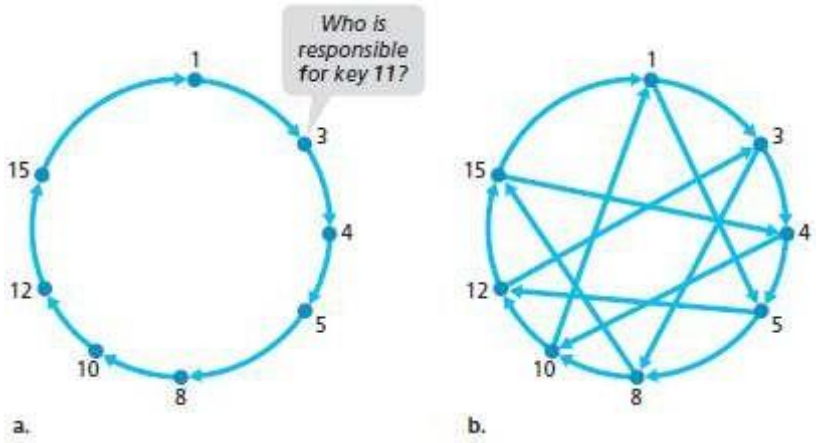


Figure 1.18: (a) A circular DHT. Peer 3 wants to determine who is responsible for key 11.
(b) A circularDHT with shortcuts

1.6.2.1 Circular Arrangement

- In this example, the identifiers are range [0, 15] (4-bit strings) and there are eight peers.
- Each peer is only aware of its immediate successor and predecessor (Figure 1.18a).
- So, each peer just needs to track two neighbors.
- When a peer asks for a key, the peer sends the message clockwise around the circle.
- For example:
  - The peer 4 sends a message saying "Who is responsible for key 11?"
  - The message will forward through peers 5, 8, 10 and reach peer 12.
  - The peer 12 determines that it is the closest peer to key 11.
  - At this point, peer 12 sends a message back to the querying peer 4.
- But when the circle is too large, a message may go through a large number of peers to get answer.
- Problem: There is trade off between
  i) Number of neighbors each peer has to track and
  ii) Number of message to be sent to resolve a single query.
- Solution: To reduce the query time, add "shortcuts" to the circular arrangement (Figure 1.18b).

1.7 Socket Programming: Creating Network Applications

- Two types of network-applications:
  1) First type is an implementation whose operation is specified in a protocol standard (RFC)
  - Such an application is referred to as "open".
  - The client & server programs must conform to the rules dictated by the RFC.
  2) Second type is a proprietary network-application.
  - The client & server programs use an application-layer protocol not openly published in a RFC.
  - A single developer creates both the client and server programs.
  - The developer has complete control over the code.
- During development phase, the developer must decide whether the application uses TCP or UDP.

1.7.1 Socket Programming with UDP

- Consider client-server application in which following events occurs:
  1) The client

→ reads a line of characters (data) from the keyboard and
→ sends the data to the server.

2) The server receives the data and converts the characters to uppercase.
3) The server sends the modified data to the client.
4) The client receives the modified data and displays the line on its screen.

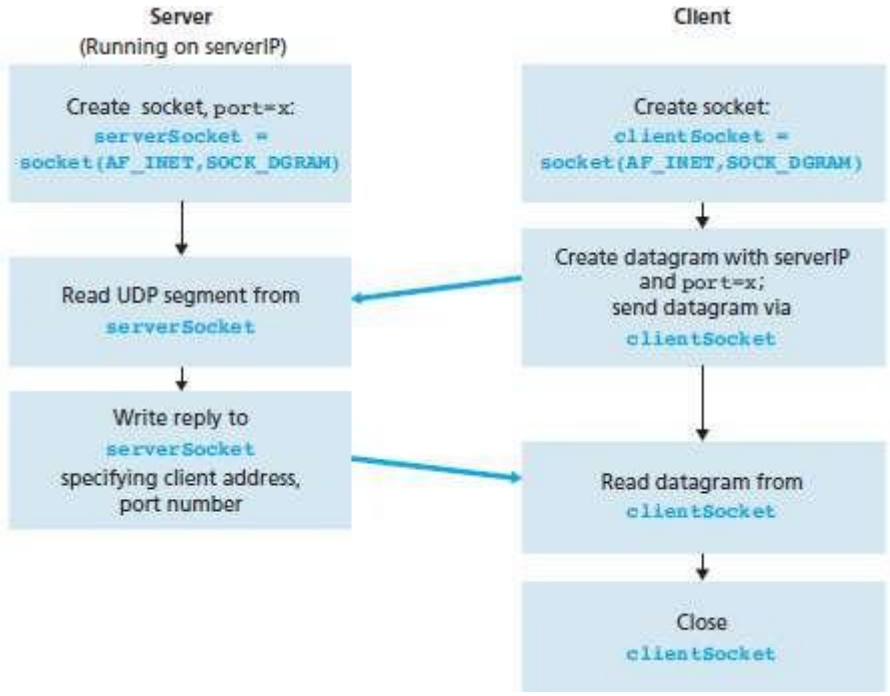


Figure 1.19: The client-server application using UDP

- The client-side of the application is as follows (Figure 1.19):

```
from socket import *    //This line declares socket within the program.
serverName = 'hostname'      // This line sets the server name to "hostname".
serverPort = 12000           // This line sets the server port# to "12000".
clientSocket = socket(socket.AF_INET, socket.SOCK_DGRAM) //This line creates the client's socket
message = raw_input('Input lowercase sentence:') //This line reads the data at the client
clientSocket.sendto(message,(serverName, serverPort)) //This line sends data into the socket
modifiedMessage, serverAddress = clientSocket.recvfrom(2048) // This line receives data from socket
print modifiedMessage //This line displays received-data on the screen
clientSocket.close() //This line closes the socket. The process then terminates.
```

Here, AF_INET indicates address family
SOCK_DGRAM indicates UDP as socket type contains server's IP address & port#

- The server-side of the application is as follows:

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort)) // This line assigns the port# 12000 to the server's socket.
print "The server is ready to receive"
while 1:
message, clientAddress = serverSocket.recvfrom(2048)
modifiedMessage = message.upper() // This line converts data to upper case
serverSocket.sendto(modifiedMessage, clientAddress)
```

1.7.2 Socket Programming with TCP

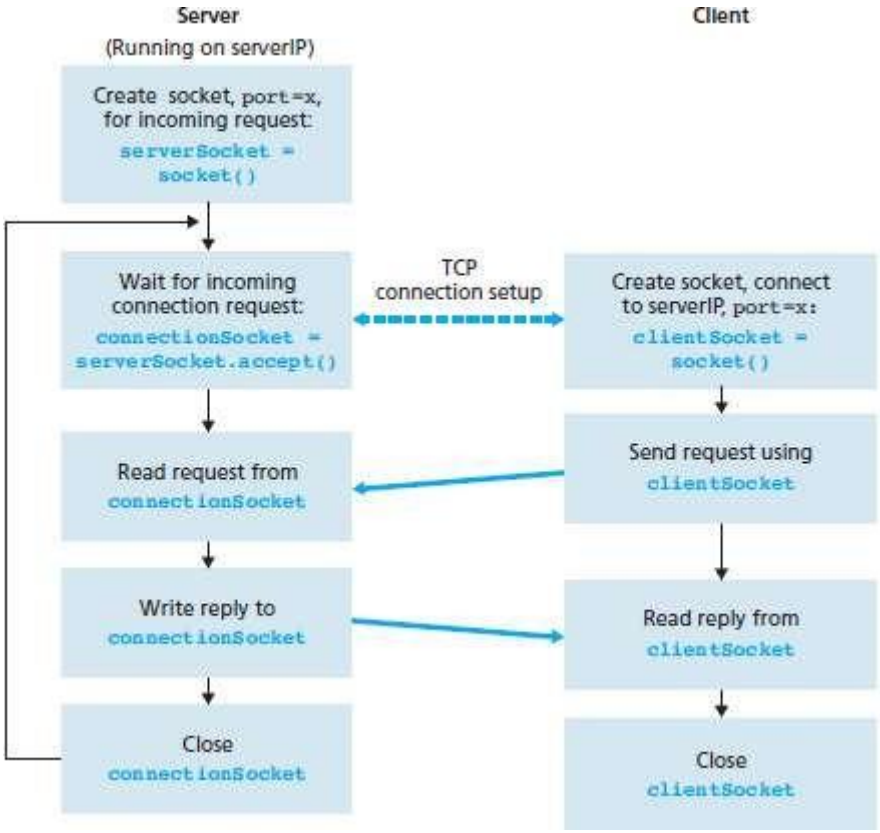


Figure 1.20: The client-server application using TCP

- The client-side of the application is as follows (Figure 1.20):

```
from socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort)) // This line initiates TCP connection b/w client & server
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence)
modifiedSentence = clientSocket.recv(1024)
print 'From Server:', modifiedSentence
clientSocket.close()
```

- The server-side of the application is as follows:

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(1) // This line specifies no. of connection-requests from the client to server
print 'The server is ready to receive'
while 1:
connectionSocket, addr=serverSocket.accept() //allows server to accept connection request from client
sentence = connectionSocket.recv(1024)
capitalizedSentence = sentence.upper()
connectionSocket.send(capitalizedSentence)
connectionSocket.close()
```

MODULE 2: TRANSPORT LAYER

2.1 Introduction and Transport Layer Services

- A transport-layer protocol provides logical-communication b/w application-processes running on different hosts.
- Transport-layer protocols are implemented in the end-systems but not in network-routers.
- On the sender, the transport-layer
  → receives messages from an application-process
  → converts the messages into the segments and
  → passes the segment to the network-layer.
- On the receiver, the transport-layer
  → receives the segment from the network-layer
  → converts the segments into to the messages and
  → passes the messages to the application-process.
- The Internet has 2 transport-layer protocols: TCP and UDP

2.1.1 Relationship between Transport and Network Layers

- A transport-layer protocol provides logical-communication b/w processes running on different hosts.
  Whereas, a network-layer protocol provides logical-communication between hosts.
- Transport-layer protocols are implemented in the end-systems but not in network-routers.
- Within an end-system, a transport protocol
  → moves messages from application-processes to the network-layer and vice versa.
  → but doesn't say anything about how the messages are moved within the network-core.
- The routers do not recognize any info. which is appended to the messages by the transport-layer.

2.1.2 Overview of the Transport Layer in the Internet

- When designing a network-application, we must choose either TCP or UDP as transport protocol.

1) UDP (User Datagram Protocol)

- UDP provides a connectionless service to the invoking application.
- The UDP provides following 2 services:
  i) Process-to-process data delivery and
  ii) Error checking.
- UDP is an unreliable service i.e. it doesn't guarantee data will arrive to destination-process.

2) TCP (Transmission Control Protocol)

- TCP provides a connection-oriented service to the invoking application.
- The TCP provides following 3 services:
  1) Reliable data transfer i.e. guarantees data will arrive to destination-process correctly.
  2) Congestion control and
  3) Error checking.

2.2 Multiplexing and Demultiplexing

- A process can have one or more sockets.
- The sockets are used to pass data from the network to the process and vice versa.

1) Multiplexing

- At the sender, the transport-layer
  → gathers data-chunks at the source-host from different sockets
  → encapsulates data-chunk with header to create segments and
  → passes the segments to the network-layer.
- The job of combining the data-chunks from different sockets to create a segment is called multiplexing.

2) Demultiplexing

- At the receiver, the transport-layer
  → examines the fields in the segments to identify the receiving-socket and
  → directs the segment to the receiving-socket.

- The job of delivering the data in a segment to the correct socket is called demultiplexing.

- In Figure 2.1,
  - In the middle host, the transport-layer must demultiplex segments arriving from the network-layer to either process P1 or P2.
  - The arriving segment's data is directed to the corresponding process's socket.

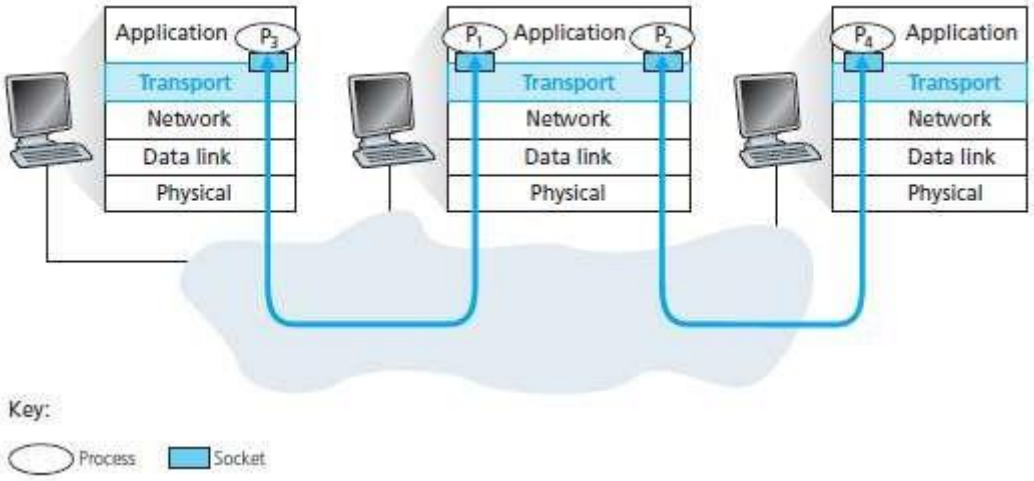


Figure 2.1: Transport-layer multiplexing and demultiplexing

2.2.1 Endpoint Identification

- Each socket must have a unique identifier.
- Each segment must include 2 header-fields to identify the socket (Figure 2.2):
  1) Source-port-number field and
  2) Destination-port-number field.
- Each port-number is a 16-bit number: 0 to 65535.
- The port-numbers ranging from 0 to 1023 are called well-known port-numbers and are restricted.For example: HTTP uses port-no 80
  FTP uses port-no 21
- When we develop a new application, we must assign the application a port-number, which are knownas ephemeral ports (49152–65535).

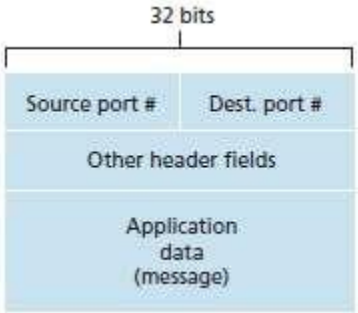


Figure 2.2: Source and destination-port-no fields in a transport-layer segment

- How the transport-layer implements the de-multiplexing service?
- Answer:
  - Each socket in the host will be assigned a port-number.
  - When a segment arrives at the host, the transport-layer
    → examines the destination-port-no in the segment
    → directs the segment to the corresponding socket and
    → passes then the segment to the attached process.

2.2.2 Connectionless Multiplexing and De-multiplexing

- At client side of the application, the transport-layer automatically assigns the port-number.
  Whereas, at the server side, the application assigns a specific port-number.
- Suppose process on Host-A (port 19157) wants to send data to process on Host-B (port 46428).

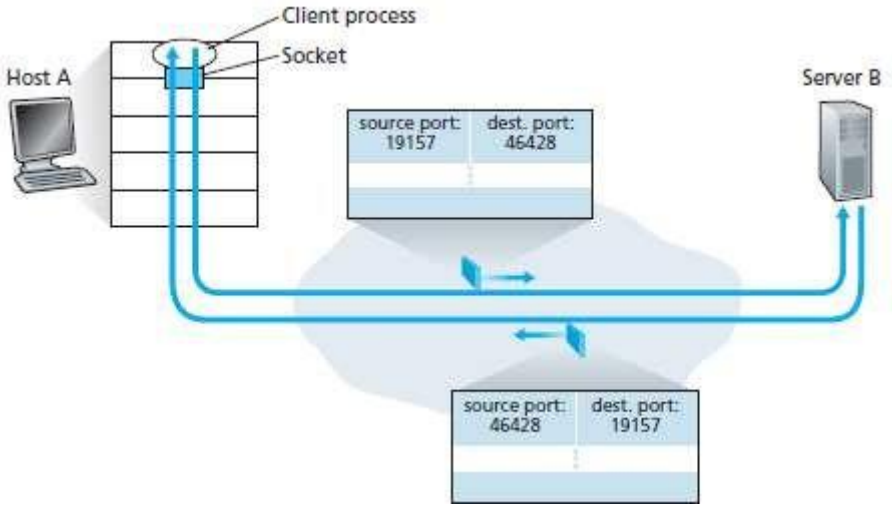


Figure 2.3: The inversion of source and destination-port-nos

- At the sender A, the transport-layer
  → creates a segment containing source-port 19157, destination-port 46428 & data and
  → passes then the resulting segment to the network-layer.
- At the receiver B, the transport-layer
  → examines the destination-port field in the segment and
  → delivers the segment to the socket identified by port 46428.
- A UDP socket is identified by a two-tuple:
  1) Destination IP address &
  2) Destination-port-no.
- As shown in Figure 2.3,
  Source-port-no from Host-A is used at Host-B as "return address" i.e. when B wants to send a segment back to A.

2.2.3 Connection Oriented Multiplexing and De-multiplexing

- Each TCP connection has exactly 2 end-points. (Figure 2.4).
- Thus, 2 arriving TCP segments with different source-port-nos will be directed to 2 different sockets,even if they have the same destination-port-no.
- A TCP socket is identified by a four-tuple:
  1) Source IP address
  2) Source-port-no
  3) Destination IP address &
  4) estination-port-no.

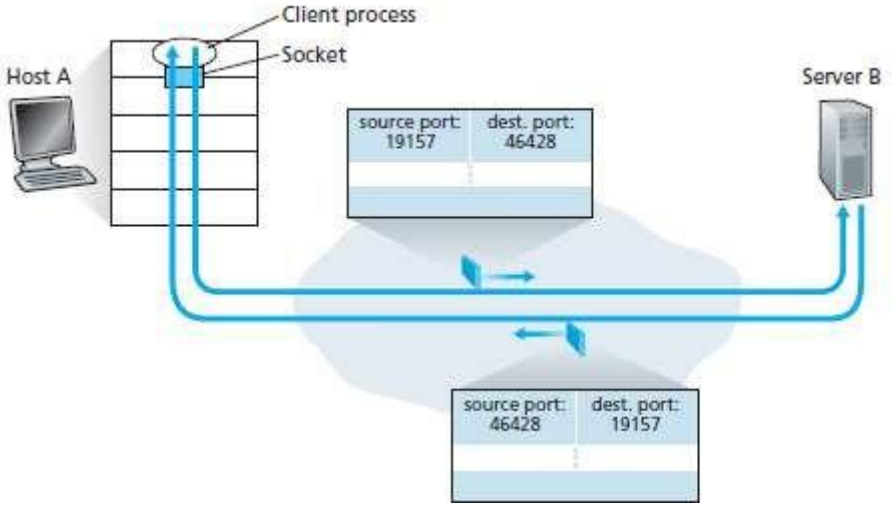


Figure 2.4: The inversion of source and destination-port-nos

- The server-host may support many simultaneous connection-sockets.
- Each socket will be
  → attached to a process.
  → identified by its own four tuple.
- When a segment arrives at the host, all 4 fields are used to direct the segment to the appropriatesocket. (i.e. Demultiplexing).

2.2.4 Web Servers and TCP

- Consider a host running a Web-server (ex: Apache) on port 80.
- When clients (ex: browsers) send segments to the server, all segments will have destination-port 80.
- The server distinguishes the segments from the different clients using two-tuple:
  1) Source IP addresses &
  2) Source-port-nos.
- Figure 2.5 shows a Web-server that creates a new process for each connection.
- The server can use either i) persistent HTTP or ii) non-persistent HTTP

i) Persistent HTTP

➢ Throughout the duration of the persistent connection the client and server exchange HTTP messages via the same server socket.

ii) Non-persistent HTTP

➢ A new TCP connection is created and closed for every request/response.
➢ Hence, a new socket is created and closed for every request/response.
➢ This can severely impact the performance of a busy Web-server.

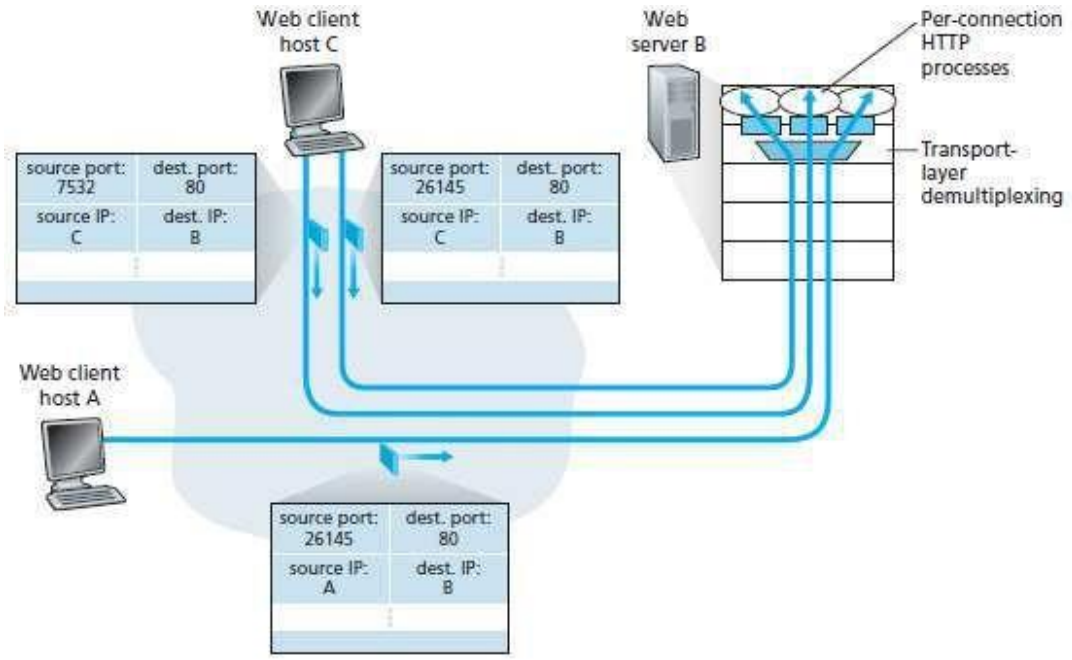


Figure 2.5: Two clients, using the same destination-port-no (80) to communicate with the same Web-server application

2.3 Connectionless Transport: UDP

- UDP is an unreliable, connectionless protocol.
  - Unreliable service means UDP doesn't guarantee data will arrive to destination-process.
  - Connectionless means there is no handshaking b/w sender & receiver before sending data.
- It provides following 2 services:
  - i) Process-to-process data delivery and
  - ii) Error checking.
- It does not provide flow, error, or congestion control.
- At the sender, UDP
  - → takes messages from the application-process
  - → attaches source- & destination-port-nos and
  - → passes the resulting segment to the network-layer.
- At the receiver, UDP
  - → examines the destination-port-no in the segment and
  - → delivers the segment to the correct application-process.
- It is suitable for application program that
  - → needs to send short messages &
  - → cannot afford the retransmission.
- UDP is suitable for many applications for the following reasons:

1) Finer Application Level Control over what Data is Sent, and when.

- When an application-process passes data to UDP, the UDP
  - → packs the data inside a segment and
  - → passes immediately the segment to the network-layer.
- On the other hand,
  - In TCP, a congestion-control mechanism throttles the sender when the n/w is congested

2) No Connection Establishment.

- TCP uses a three-way handshake before it starts to transfer data.
- UDP just immediately passes the data without any formal preliminaries.
- Thus, UDP does not introduce any delay to establish a connection.
- That's why, DNS runs over UDP rather than TCP.

3) No Connection State.

- TCP maintains connection-state in the end-systems.

- This connection-state includes
  → receive and send buffers
  → congestion-control parameters and
  → sequence- and acknowledgment-number parameters.
- On the other hand,
  In UDP, no connection-state is maintained.

4) Small Packet Header Overhead.

- The TCP segment has 20 bytes of header overhead in every segment.
- On the other hand, UDP has only 8 bytes of overhead.

Table 2.1: Popular Internet applications and their underlying transport protocols

Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP	TCP
Remote terminal access	Telnet	TCP
Web	HTTP	TCP
File transfer	FTP	TCP
Remote file server	NFS	Typically UDP
Streaming multimedia	typically proprietary	UDP or TCP
Internet telephony	typically proprietary	UDP or TCP
Network management	SNMP	Typically UDP
Routing protocol	RIP	Typically UDP
Name translation	DNS	Typically UDP

2.3.1 UDP Segment Structure

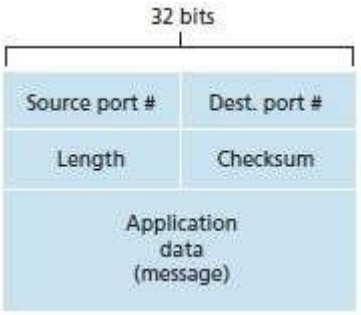


Figure 2.6: UDP segment structure

- UDP Segment contains following fields (Figure 2.6):
  1) **Application Data:** This field occupies the data-field of the segment.
  2) **Destination Port No:** This field is used to deliver the data to correct process running on the destination-host. (i.e. demultiplexing function).
  3) **Length:** This field specifies the number of bytes in the segment (header plus data).
  4) **Checksum:** This field is used for error-detection.

2.3.2 UDP Checksum

- The checksum is used for error-detection.
- The checksum is used to determine whether bits within the segment have been altered.
- How to calculate checksum on the sender:
  1) All the 16-bit words in the segment are added to get a sum.
  2) Then, the 1's complement of the sum is obtained to get a result.

3) Finally, the result is added to the checksum-field inside the segment.

- How to check for error on the receiver:
  1) All the 16-bit words in the segment (including the checksum) are added to get a sum.
     i) For no errors: In the sum, all the bits are 1. (Ex: 1111111)
     ii) For any error: In the sum, at least one of the bits is a 0. (Ex: 1011111)

Example:

- On the sender:
  - Suppose that we have the following three 16-bit words:

    0110011001100000
    0101010101010101 → three 16 bits words
    1000111100001100
  - The sum of first two 16-bit words is:

    0110011001100000
    0101010101010101
    1011101110110101
  - Adding the third word to the above sum gives:

    1011101110110101 → sum of 1st two 16 bit words
    1000111100001100 → third 16 bit word
    0100101011000010 → sum of all three 16 bit words
  - Taking 1's complement for the final sum:

    0100101011000010 → sum of all three 16 bit words
    1011010100111101 → 1's complement for the final sum
- The 1's complement value is called as checksum which is added inside the segment.
- On the receiver
  - All four 16-bit words are added, including the checksum.
    i) If no errors are introduced into the packet, then clearly the sum will be 1111111111111111.
    ii) If one of the bits is a 0, then errors have been introduced into the packet.

2.4 Principles of Reliable Data Transfer

- Figure 2.7 illustrates the framework of reliable data transfer protocol.

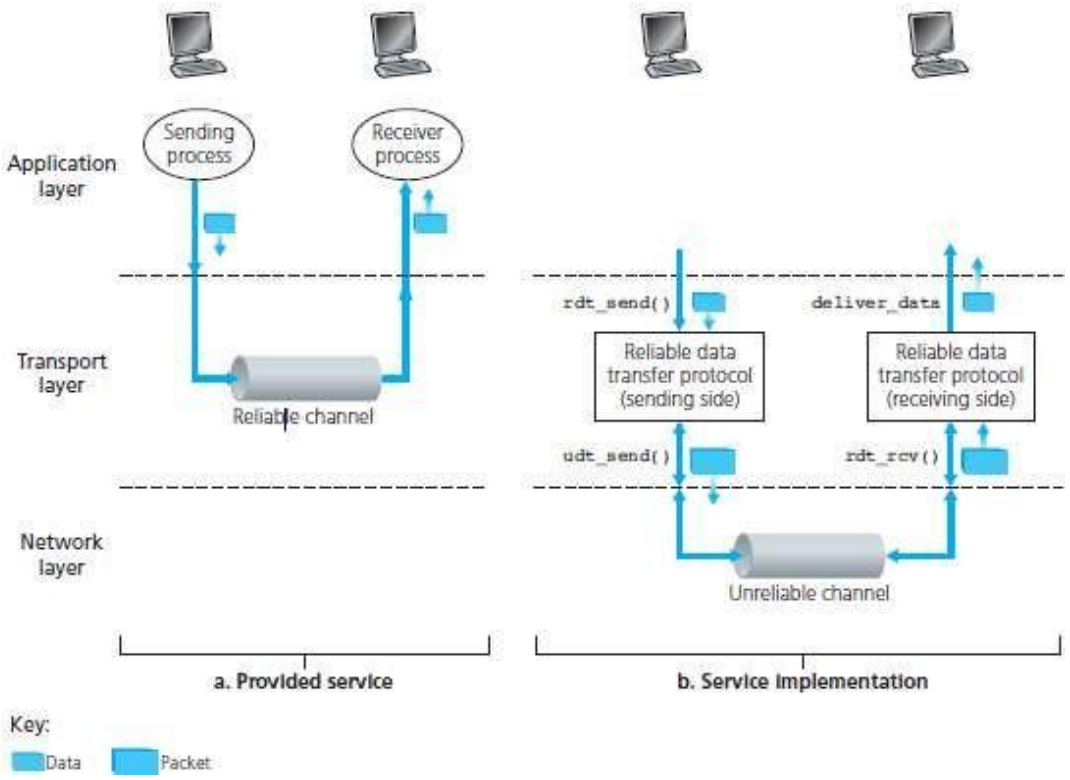


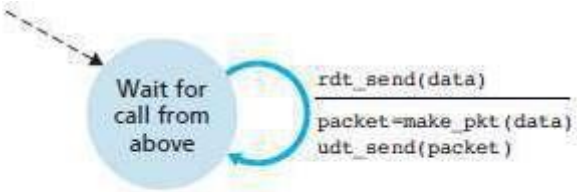
Figure 2.7: Reliable data transfer: Service model and service implementation

- On the sender, rdt_send() will be called when a packet has to be sent on the channel.
- On the receiver,
  i) rdt_rcv()will be called when a packet has to be recieved on the channel.
  ii) deliver_data() will be called when the data has to be delivered to the upper layer

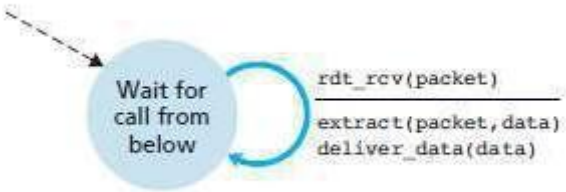
2.4.1 Building a Reliable Data Transfer Protocol

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

- Consider data transfer over a perfectly reliable channel.
- We call this protocol as rdt1.0.



a. rdt1.0: sending side



b. rdt1.0: receiving side

Figure 2.8: rdt1.0 – A protocol for a completely reliable channel

- The finite-state machine (FSM) definitions for the rdt1.0 sender and receiver are shown in Figure 2.8.
- The sender and receiver FSMs have only one state.
- In FSM, following notations are used:
  i) The arrows indicate the transition of the protocol from one state to another.
  ii) The event causing the transition is shown above the horizontal line labelling the transition.
  iii) The action taken when the event occurs is shown below the horizontal line.
  iv) The dashed arrow indicates the initial state.
- On the sender, rdt
  → accepts data from the upper layer via the rdt_send(data) event
  → creates a packet containing the data (via the action make_pkt(data)) and
  → sends the packet into the channel.
- On the receiver, rdt
  → receives a packet from the underlying channel via the rdt_rcv(packet) event
  → removes the data from the packet (via the action extract (packet, data)) and
  → passes the data up to the upper layer (via the action deliver_data(data)).

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

- Consider data transfer over an unreliable channel in which bits in a packet may be corrupted.
- We call this protocol as rdt2.0.
- The message dictation protocol uses both
  → positive acknowledgements (ACK) and
  → negative acknowledgements (NAK).
- The receiver uses these control messages to inform the sender about
  → what has been received correctly and
  → what has been received in error and thus requires retransmission.
- Reliable data transfer protocols based on the retransmission are known as ARQ protocols.
- Three additional protocol capabilities are required in ARQ protocols:

1) Error Detection

- A mechanism is needed to allow the receiver to detect when bit-errors have occurred.
- UDP uses the checksum field for error-detection.
- Error-correction techniques allow the receiver to detect and correct packet bit-errors.

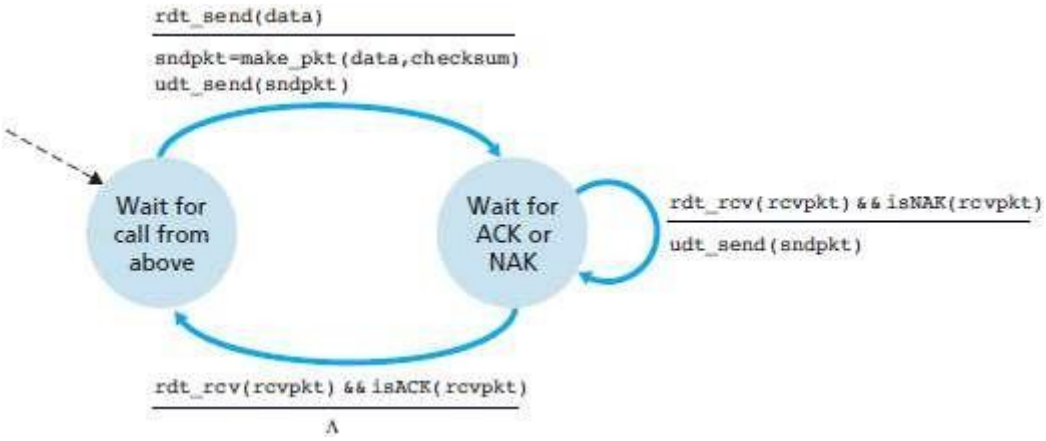
2) Receiver Feedback

- Since the sender and receiver are typically executing on different end-systems.
- The only way for the sender to learn about status of the receiver is by the receiver providing explicit feedback to the sender.
- For example: ACK & NAK

3) Retransmission

- A packet that is received in error at the receiver will be retransmitted by the sender.

- Figure 2.9 shows the FSM representation of rdt2.0.



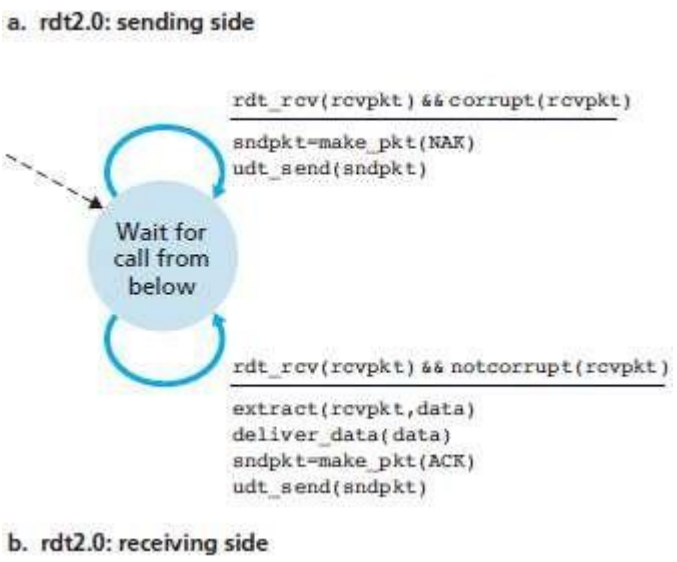


Figure 2.9: rdt2.0–A protocol for a channel with bit-errors

Sender FSM

- The sender of rdt2.0 has 2 states:
  1) In one state, the protocol is waiting for data to be passed down from the upper layer.
  2) In other state, the protocol is waiting for an ACK or a NAK from the receiver.
     i) If an ACK is received, the protocol
        → knows that the most recently transmitted packet has been received correctly
        → returns to the state of waiting for data from the upper layer.
     ii) If a NAK is received, the protocol
        → retransmits the last packet and
        → waits for an ACK or NAK to be returned by the receiver.
- The sender will not send a new data until it is sure that the receiver has correctly received the current packet.
- Because of this behaviour, protocol rdt2.0 is known as stop-and-wait protocols.

Receiver FSM

- The receiver of rdt2.0 has a single state.
- On packet arrival, the receiver replies with either an ACK or a NAK, depending on the received packetis corrupted or not.

2.4.1.2.1 Sender Handles Garbled ACK/NAKs: rdt2.1

- Problem with rdt2.0:
  If an ACK or NAK is corrupted, the sender cannot know whether the receiver has correctly received the data or not.
- Solution: The sender resends the current data packet when it receives garbled ACK or NAK packet.
  - Problem: This approach introduces duplicate packets into the channel.
  - Solution: Add sequence-number field to the data packet.
  - The receiver has to only check the sequence-number to determine whether the received packet is a retransmission or not.
- For a stop-and-wait protocol, a 1-bit sequence-number will be sufficient.
- A 1-bit sequence-number allows the receiver to know whether the sender is sending
  → previously transmitted packet (0) or
  → new packet (1).
- We call this protocol as rdt2.1.
- Figure 2.10 and 2.11 shows the FSM description for rdt2.1.

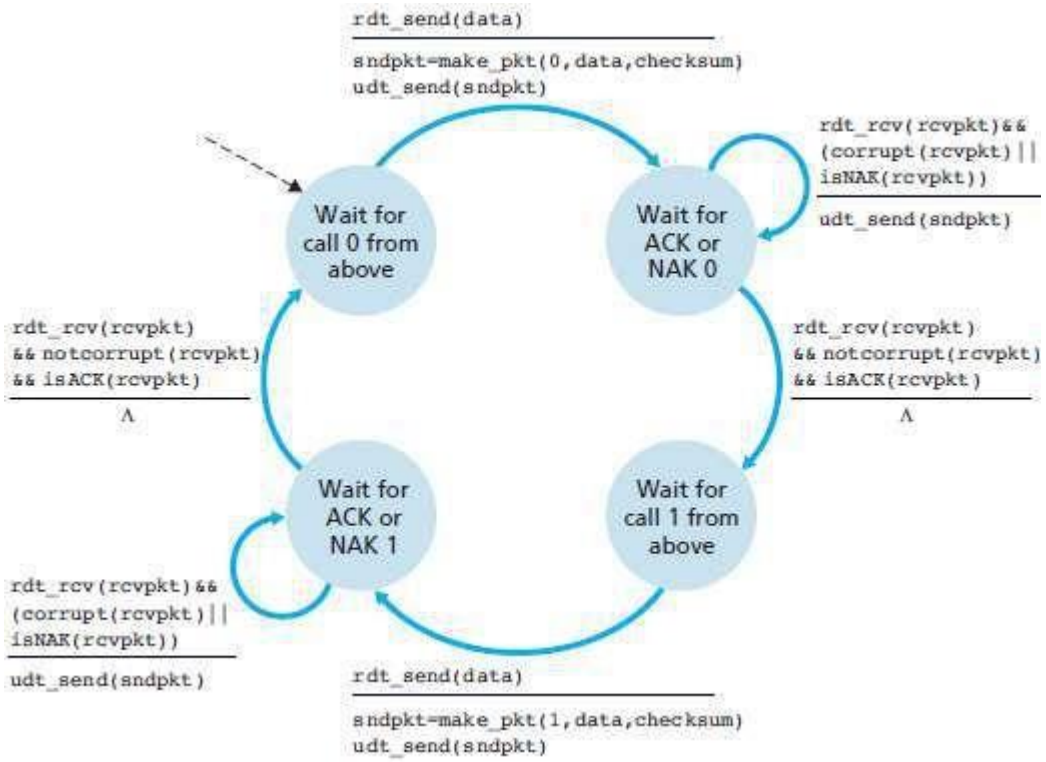


Figure 2.10: rdt2.1 sender

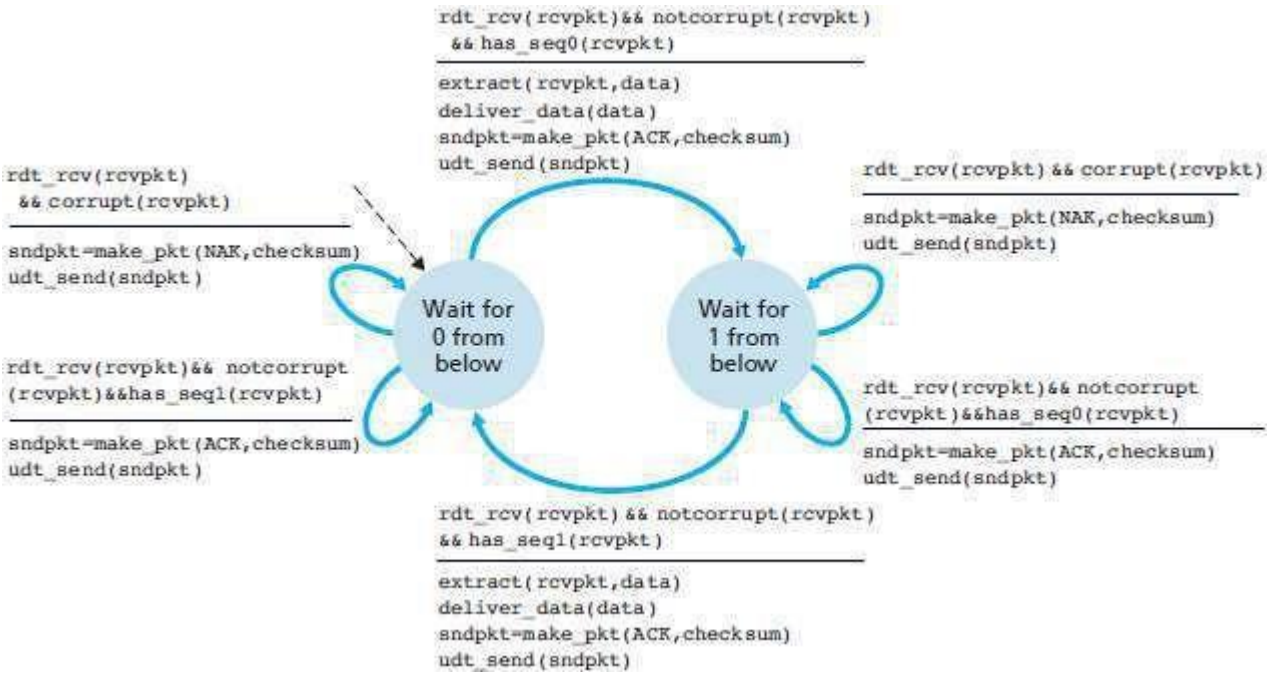


Figure 2.11: rdt2.1 receiver

2.4.1.2.2 Sender uses ACK/NAKs: rdt2.2

- Protocol rdt2.2 uses both positive and negative acknowledgments from the receiver to the sender.
  - i) When out-of-order packet is received, the receiver sends a positive acknowledgment (ACK).
  - ii) When a corrupted packet is received, the receiver sends a negative acknowledgment (NAK).

- We call this protocol as rdt2.2. (Figure 2.12 and 2.13).

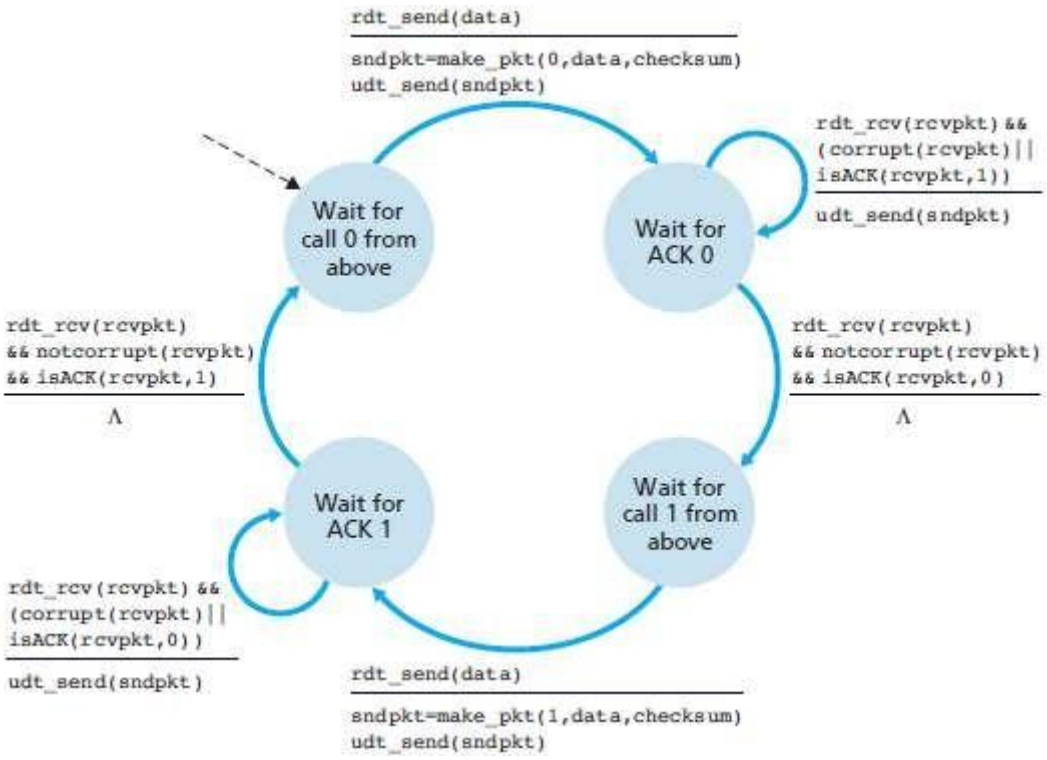


Figure 2.12: rdt2.2 sender

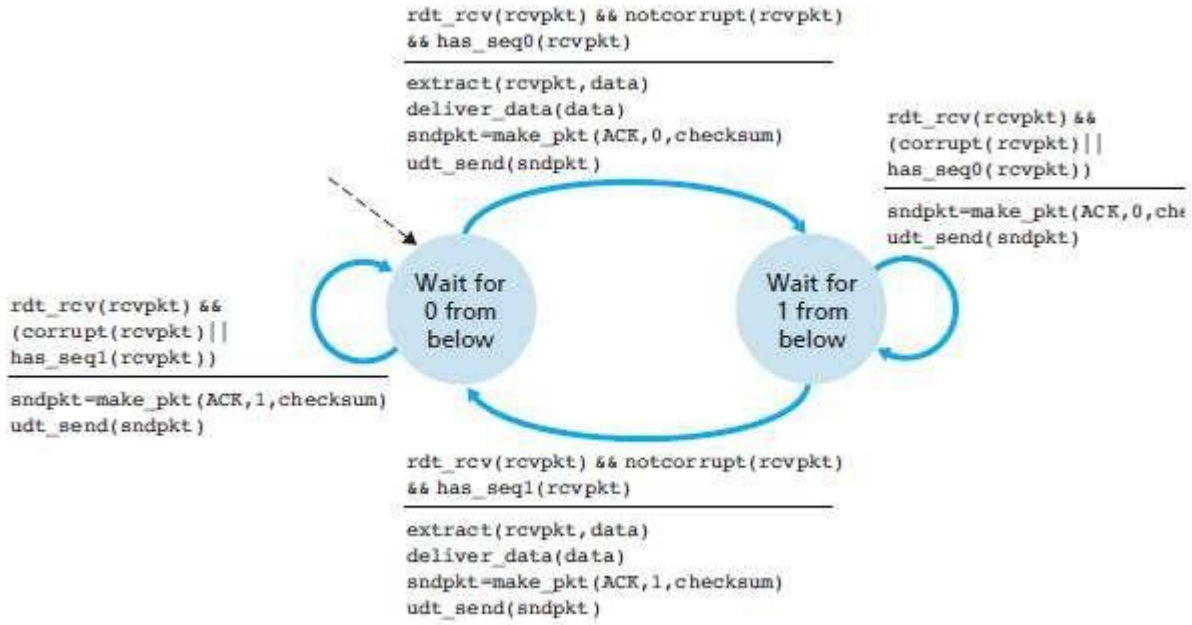


Figure 2.13: rdt2.2 receiver

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

- Consider data transfer over an unreliable channel in which packet lose may occur.
- We call this protocol as rdt3.0.
- Two problems must be solved by the rdt3.0:
  1) How to detect packet loss?
  2) What to do when packet loss occurs?
- Solution:

- The sender
  - → sends one packet & starts a timer and
  - → waits for ACK from the receiver (okay to go ahead).
- If the timer expires before ACK arrives, the sender retransmits the packet and restarts thetimer.

- The sender must wait at least as long as
  1) A round-trip delay between the sender and receiver plus
  2) Amount of time needed to process a packet at the receiver.
- Implementing a time-based retransmission mechanism requires a countdown timer.
- The timer must interrupt the sender after a given amount of time has expired.
- Figure 2.14 shows the sender FSM for rdt3.0, a protocol that reliably transfers data over a channelthat can corrupt or lose packets;
- Figure 2.15 shows how the protocol operates with no lost or delayed packets and how it handles lostdata packets.
- Because sequence-numbers alternate b/w 0 & 1, protocol rdt3.0 is known as alternating-bit protocol.

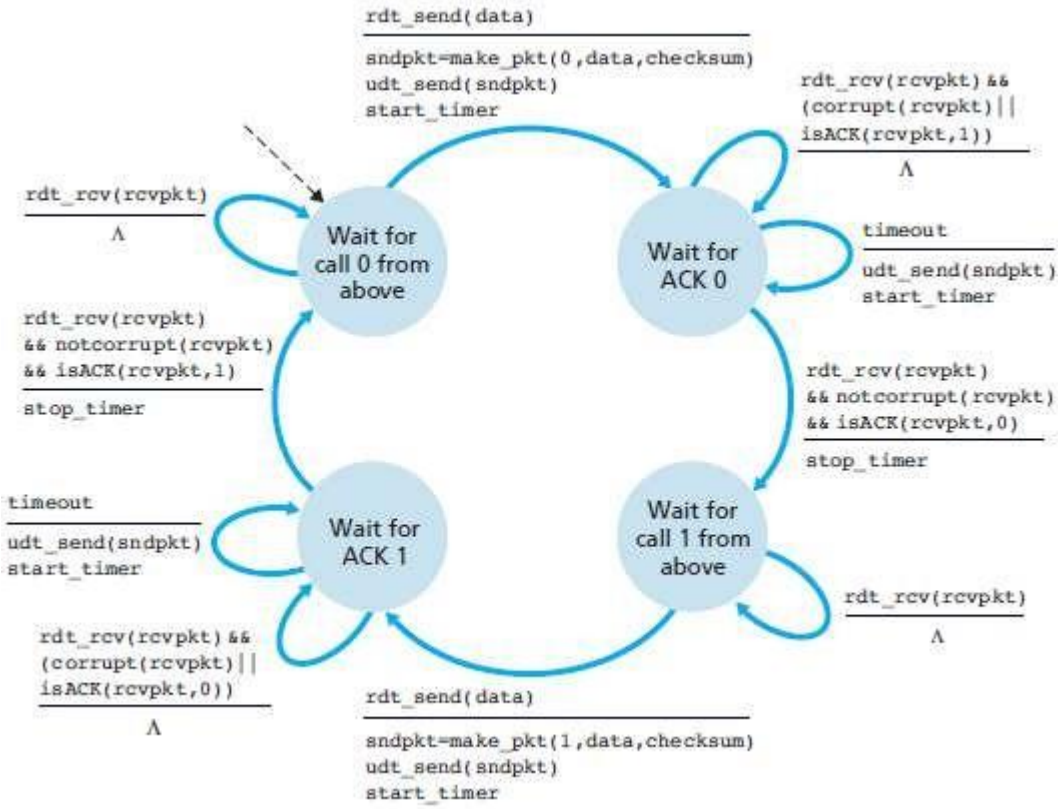


Figure 2.14: rdt3.0 sender

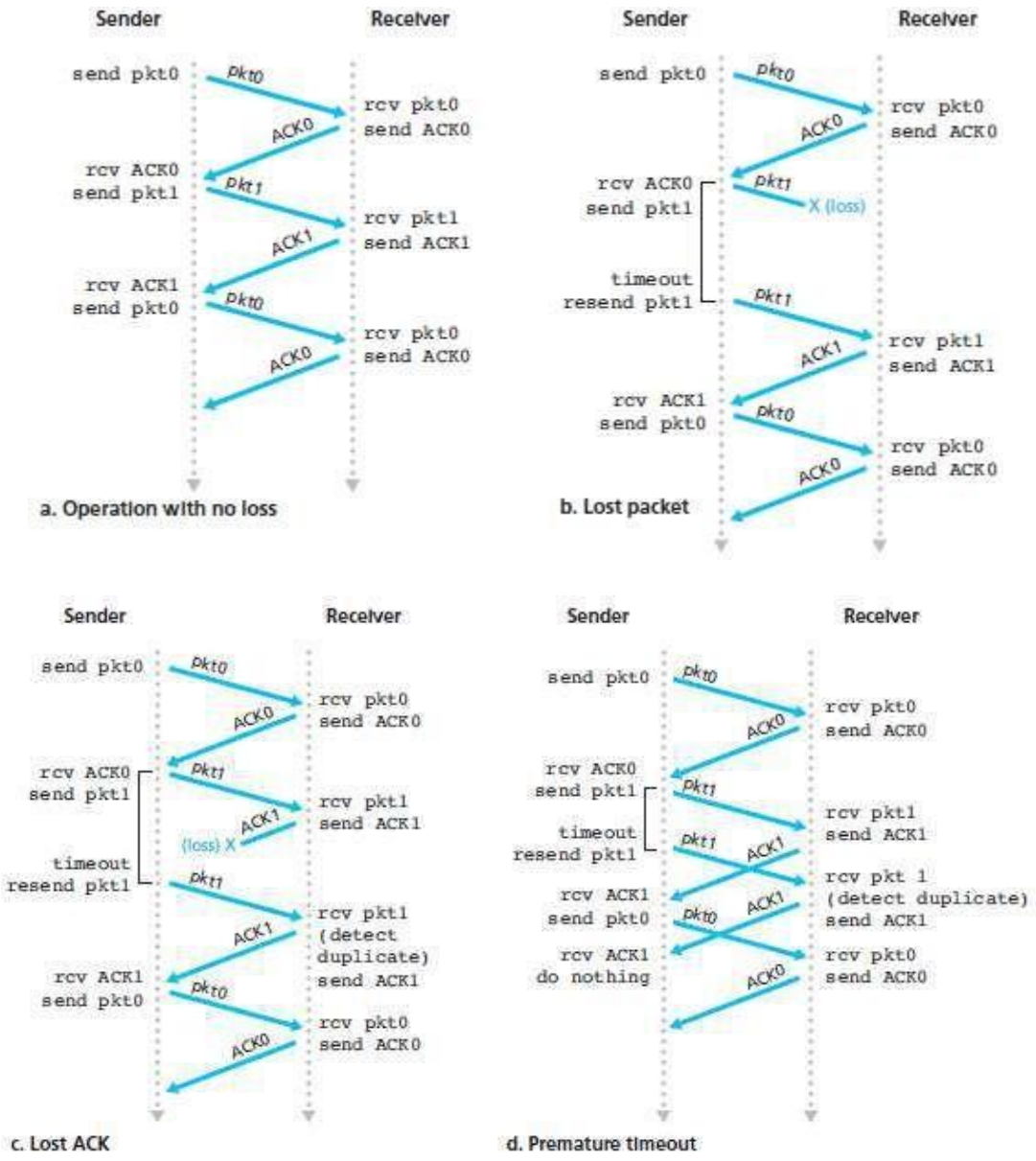


Figure 2.15: Operation of rdt3.0, the alternating-bit protocol

2.4.2 Pipelined Reliable Data Transfer Protocols

- The sender is allowed to send multiple packets without waiting for acknowledgments.
- This is illustrated in Figure 2.16 (b).
- Pipelining has the following consequences:
  1) The range of sequence-numbers must be increased.
  2) The sender and receiver may have to buffer more than one packet.
- Two basic approaches toward pipelined error recovery can be identified:
  1) Go-Back-N and 2) Selective repeat.

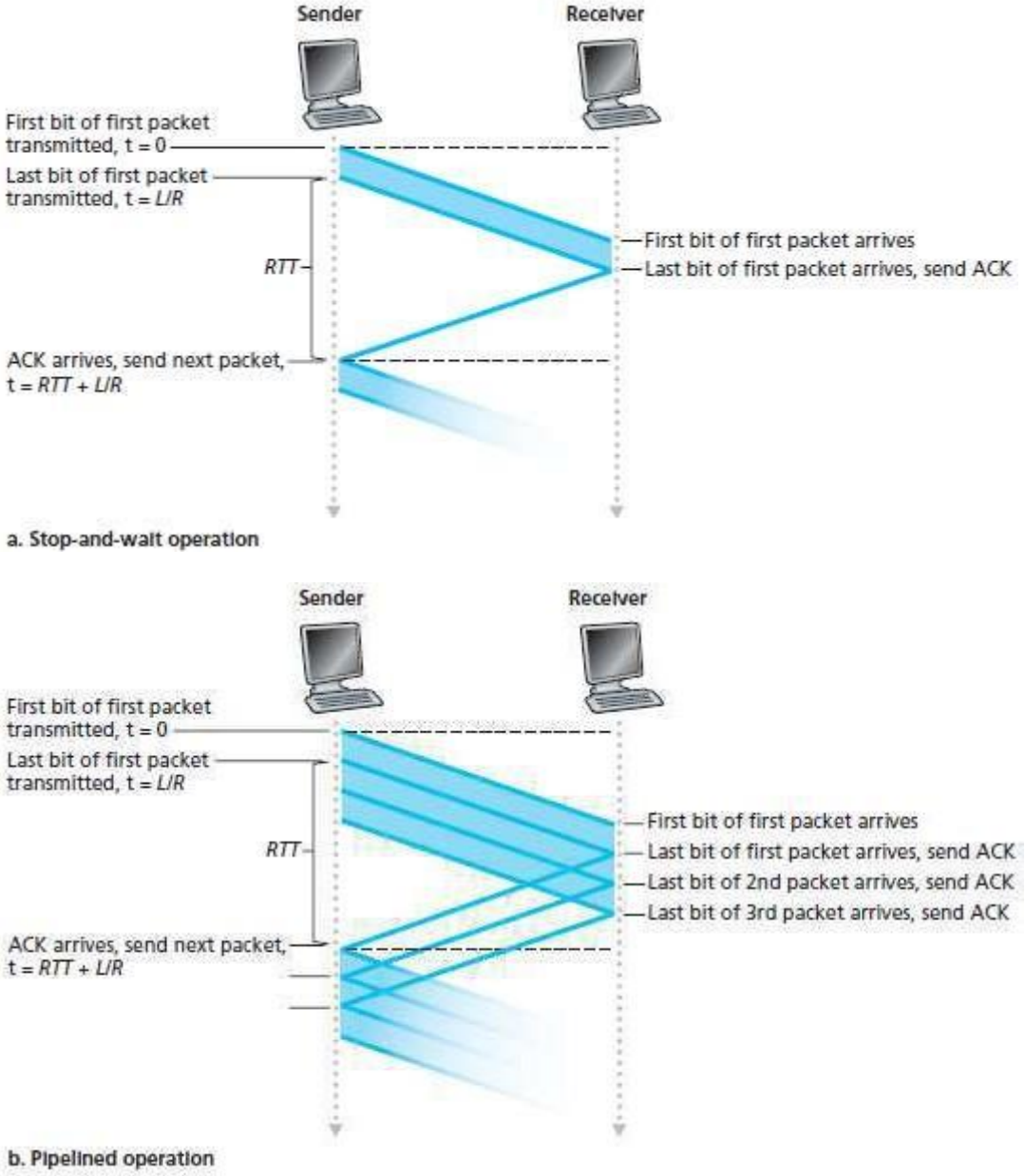


Figure 2.16: Stop-and-wait and pipelined sending

2.4.3 Go-Back-N (GBN)

- The sender is allowed to transmit multiple packets without waiting for an acknowledgment.
- But, the sender is constrained to have at most N unacknowledged packets in the pipeline.
  Where N = window-size which refers maximum no. of unacknowledged packets in the pipeline
- GBN protocol is called a sliding-window protocol.
- Figure 2.17 shows the sender's view of the range of sequence-numbers.

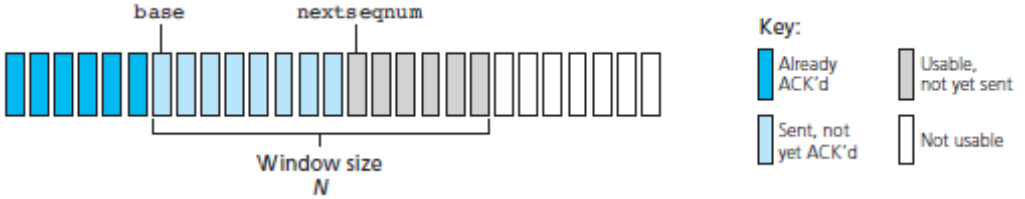


Figure 2.17: Sender's view of sequence-numbers in Go-Back-N

- Figure 2.18 and 2.19 give a FSM description of the sender and receivers of a GBN protocol.

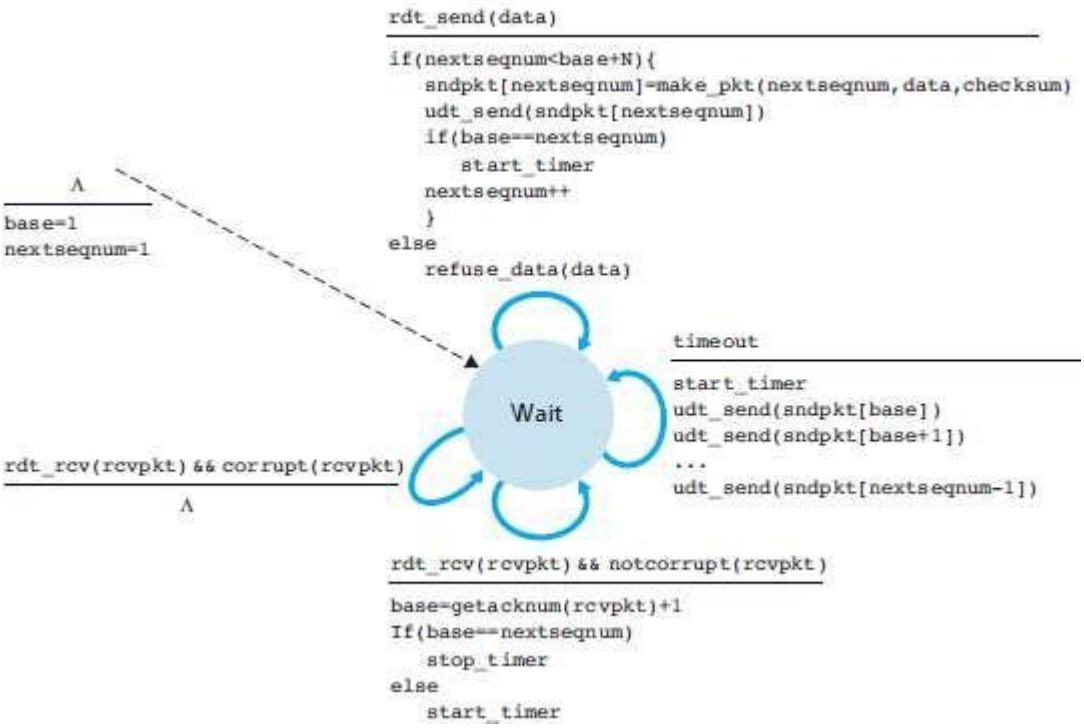


Figure 2.18: Extended FSM description of GBN sender

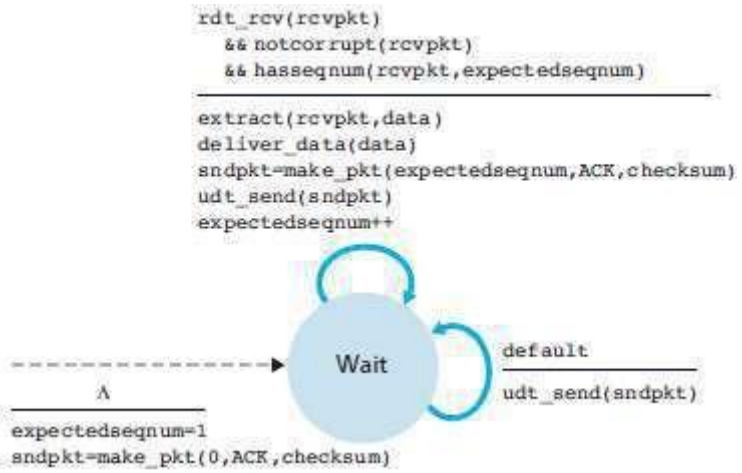


Figure 2.19: Extended FSM description of GBN receiver

2.4.3.1 GBN Sender

- The sender must respond to 3 types of events:

1) Invocation from above.

➢ When rdt_send() is called from above, the sender first checks to see if the window is full i.e. whether there are N outstanding, unacknowledged packets.

i) If the window is not full, the sender creates and sends a packet.

ii) If the window is full, the sender simply returns the data back to the upper layer. Thisis an implicit indication that the window is full.

2) Receipt of an ACK.

➢ An acknowledgment for a packet with sequence-number n will be taken to be a cumulative acknowledgment.
➢ All packets with a sequence-number up to n have been correctly received at the receiver.

3) A Timeout Event.

➢ A timer will be used to recover from lost data or acknowledgment packets.

i) If a timeout occurs, the sender resends all packets that have been previously sent butthat have not yet been acknowledged.

ii) If an ACK is received but there are still additional transmitted but not yetacknowledged packets, the timer is restarted.

iii) If there are no outstanding unacknowledged packets, the timer is stopped.

2.4.3.2 GBN Receiver

- If a packet with sequence-number n is received correctly and is in order, the receiver
  → sends an ACK for packet n and
  → delivers the packet to the upper layer.
- In all other cases, the receiver
  → discards the packet and
  → resends an ACK for the most recently received in-order packet.

2.4.3.3 Operation of the GBN Protocol

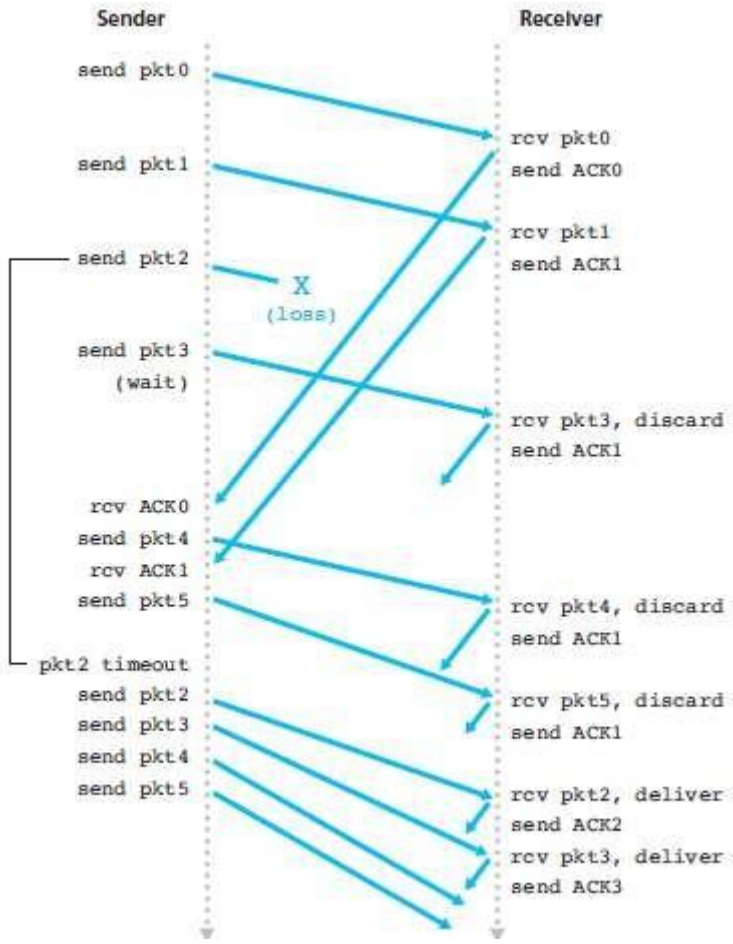


Figure 2.20: Go-Back-N in operation

- Figure 2.20 shows the operation of the GBN protocol for the case of a window-size of four packets.
- The sender sends packets 0 through 3.
- The sender then must wait for one or more of these packets to be acknowledged before proceeding.
- As each successive ACK (for ex, ACK0 and ACK1) is received, the window slides forward and the sender transmits one new packet (pkt4 and pkt5, respectively).

- On the receiver, packet 2 is lost and thus packets 3, 4, and 5 are found to be out of order and are discarded.

2.4.4 Selective Repeat (SR)

- Problem with GBN:
  - GBN suffers from performance problems.
  - When the window-size and bandwidth-delay product are both large, many packets can be inthe pipeline.
  - Thus, a single packet error results in retransmission of a large number of packets.
- Solution: Use Selective Repeat (SR).

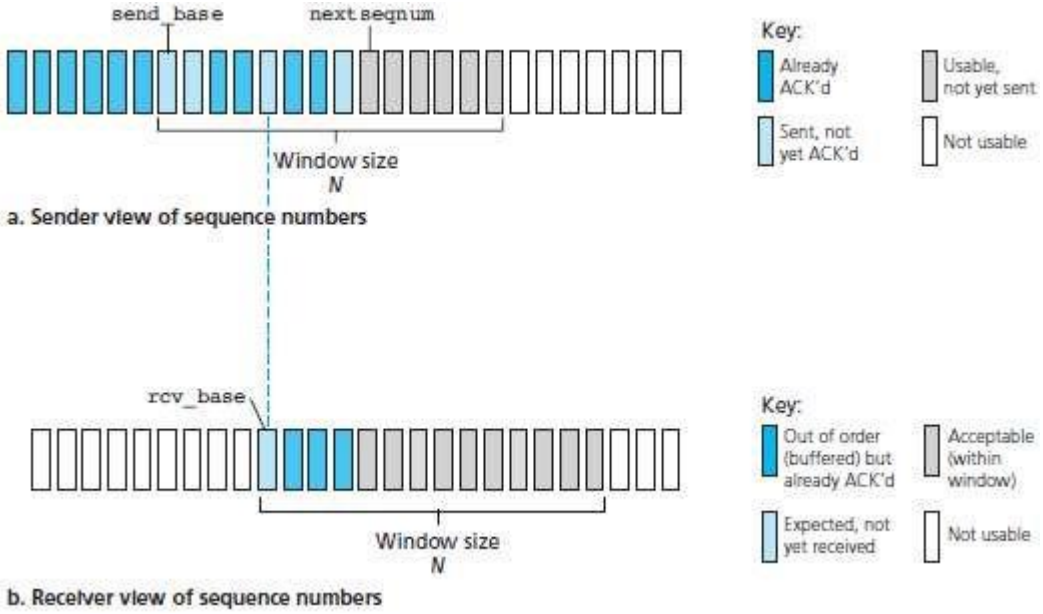


Figure 2.21: Selective-repeat (SR) sender and receiver views of sequence-number space

- The sender retransmits only those packets that it suspects were erroneous.
- Thus, avoids unnecessary retransmissions. Hence, the name "selective-repeat".
- The receiver individually acknowledge correctly received packets.
- A window-size N is used to limit the no. of outstanding, unacknowledged packets in the pipeline.
- Figure 2.21 shows the SR sender's view of the sequence-number space.

2.4.4.1 SR Sender

- The various actions taken by the SR sender are as follows:

1) Data Received from above.

- When data is received from above, the sender checks the next available sequence-number forthe packet.
- If the sequence-number is within the sender's window;
  Then, the data is packetized and sent;
  Otherwise, the data is buffered for later transmission.

2) Timeout.

- Timers are used to protect against lost packets.
- Each packet must have its own logical timer. This is because
  → only a single packet will be transmitted on timeout.

3) ACK Received.

- If an ACK is received, the sender marks that packet as having been received.
- If the packet's sequence-number is equal to send_base, the window base is increased by the smallest sequence-number.
- If there are untransmitted packets with sequence-numbers that fall within the window, these

packets are transmitted.

2.4.4.2 SR Receiver

- The various actions taken by the SR receiver are as follows:

1) Packet with sequence-number in [rcv_base, rcv_base+N-1] is correctly received.

- In this case,
  - → received packet falls within the receiver's window and
  - → selective ACK packet is returned to the sender.
- If the packet was not previously received, it is buffered.
- If this packet has a sequence-number equal to rcv_base, then this packet, and any previously buffered and consecutively numbered packets are delivered to the upper layer.
- The receive-window is then moved forward by the no. of packets delivered to the upper layer.
- For example: consider Figure 2.22.
  - ¤ When a packet with a sequence-number of rcv_base=2 is received, it and packets 3, 4,and 5 can be delivered to the upper layer.

2) Packet with sequence-number in [rcv_base-N, rcv_base-1] is correctly received.

- In this case, an ACK must be generated, even though this is a packet that the receiver has previously acknowledged.

3) Otherwise.

- Ignore the packet.

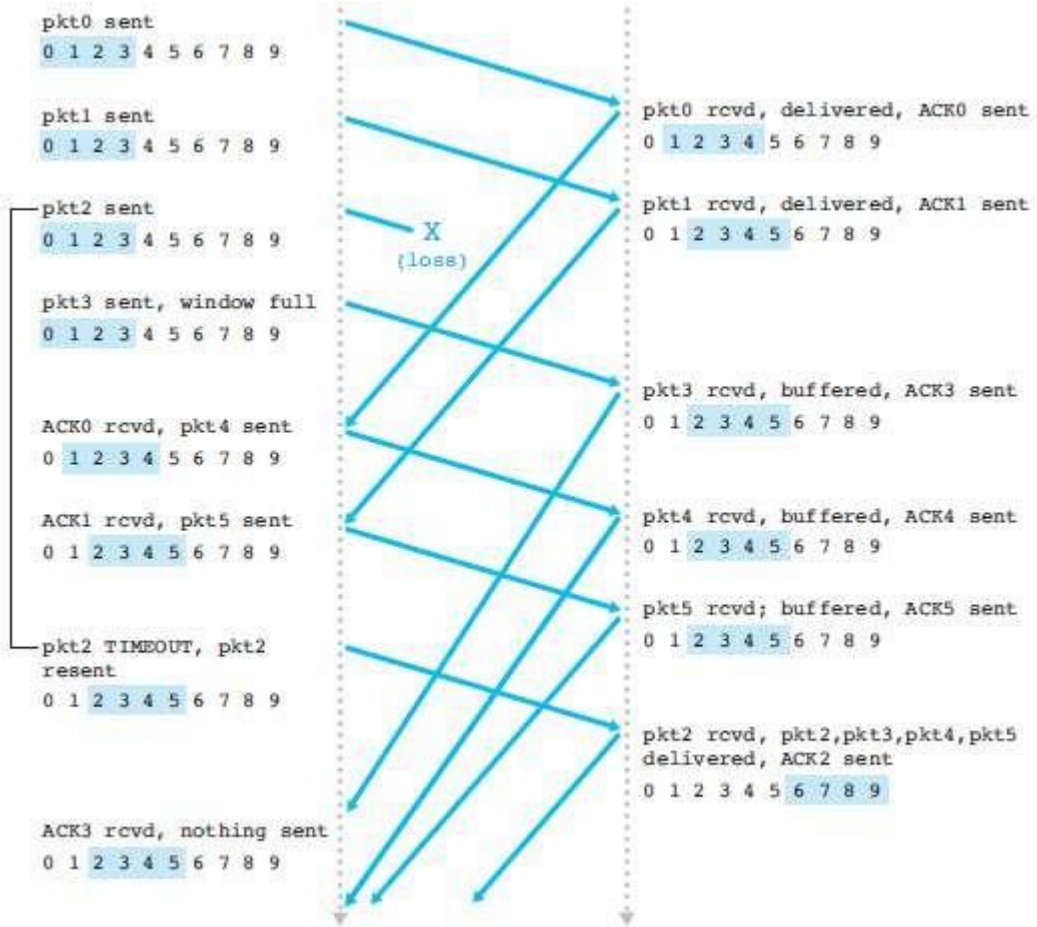


Figure 2.22: SR operation

2.4.5 Summary of Reliable Data Transfer Mechanisms and their Use

Table 2.2: Summary of reliable data transfer mechanisms and their use

Mechanism	Use, Comments
Checksum	Used to detect bit errors in a transmitted packet.
Timer	Used to timeout/retransmit a packet because the packet (or its ACK) waslost. Because timeouts can occur when a packet is delayed but not lost,duplicate copies of a packet may be received by a receiver.
Sequence-number	Used for sequential numbering of packets of data flowing from sender to receiver. Gaps in the sequence-numbers of received packets allow the receiver to detect a lost packet. Packets with duplicate sequence-numbers allow the receiver to detect duplicate copies of a packet.
Acknowledgment	Used by the receiver to tell the sender that a packet or set of packets hasbeen received correctly. Acknowledgments will typically carry the sequence-number of the packet or packets being acknowledged. Acknowledgments may be individual or cumulative, depending on theprotocol.
Negative acknowledgment	Used by the receiver to tell the sender that a packet has not been received correctly. Negative acknowledgments will typically carry the sequence-number of the packet that was not received correctly.
Window, pipelining	The sender may be restricted to sending only packets with sequence-numbers that fall within a given range. By allowing multiple packets to be transmitted but not yet acknowledged, sender utilization can be increased over a stop-and-wait mode of operation.

2.5 Connection-Oriented Transport: TCP

- TCP is a reliable connection-oriented protocol.
  - Connection-oriented means a connection is established b/w sender & receiver before sendingthe data.
  - Reliable service means TCP guarantees that the data will arrive to destination-process correctly.
- TCP provides flow-control, error-control and congestion-control.

2.5.1 The TCP Connection

- The features of TCP are as follows:

1) Connection Oriented

- TCP is said to be connection-oriented. This is because
  The 2 application-processes must first establish connection with each other before theybegin communication.
- Both application-processes will initialize many state-variables associated with the connection.

2) Runs in the End Systems

- TCP runs only in the end-systems but not in the intermediate routers.
- The routers do not maintain any state-variables associated with the connection.

3) Full Duplex Service

- TCP connection provides a full-duplex service.
- Both application-processes can transmit and receive the data at the same time.

4) Point-to-Point

- A TCP connection is point-to-point i.e. only 2 devices are connected by a dedicated-link
- So, multicasting is not possible.

5) Three-way Handshake

- Connection-establishment process is referred to as a three-way handshake. This is because3 segments are sent between the two hosts:
  - **i)** The client sends a first-segment.
  - **ii)** The server responds with a second-segment and
  - **iii)** Finally, the client responds again with a third segment containing payload (or data).

6) Maximum Segment Size (MSS)

- MSS limits the maximum amount of data that can be placed in a segment.For example: MSS = 1,500 bytes for Ethernet

7) Send & Receive Buffers

- As shown in Figure 2.23, consider sending data from the client-process to the server-process.

At Sender

i) The client-process passes a stream-of-data through the socket.
ii) Then, TCP forwards the data to the send-buffer.
iii) Each chunk-of-data is appended with a header to form a segment.
iv) The segments are sent into the network.

At Receiver

i) The segment's data is placed in the receive-buffer.
ii) The application reads the stream-of-data from the receive-buffer.

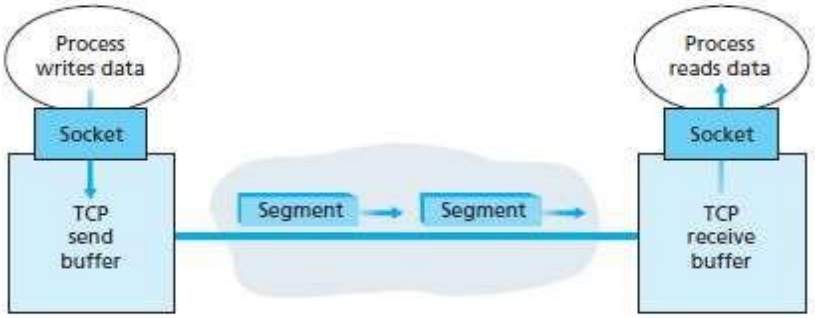


Figure 2.23: TCP send and receive-buffers

2.5.2 TCP Segment Structure

- The segment consists of header-fields and a data-field.
- The data-field contains a chunk-of-data.
- When TCP sends a large file, it breaks the file into chunks of size MSS.
- Figure 2.24 shows the structure of the TCP segment.

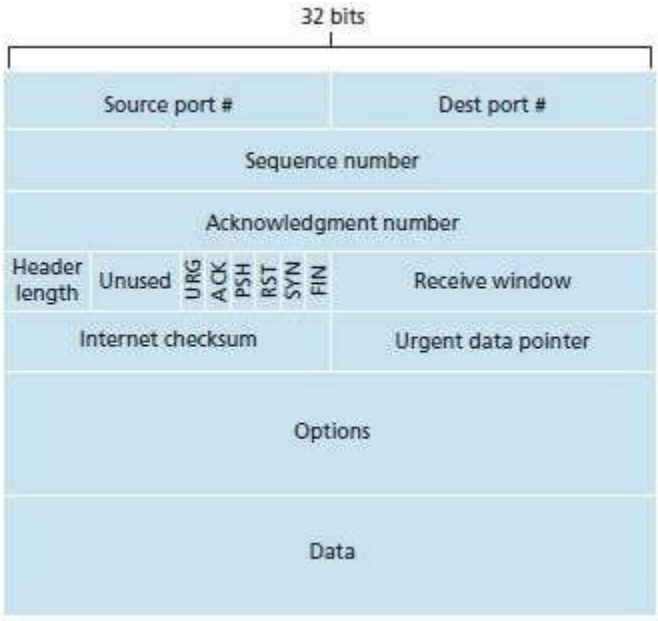


Figure 2.24: TCP segment structure

- The fields of TCP segment are as follows:
  - **1) Source and Destination Port Numbers**
    - These fields are used for multiplexing/demultiplexing data from/to upper-layer applications.
  - **2) Sequence Number & Acknowledgment Number**
    - These fields are used by sender & receiver in implementing a reliable data-transfer-service.
  - **3) Header Length**
    - This field specifies the length of the TCP header.
  - **4) Flag**
    - This field contains 6 bits.
      - **i) ACK**
        - ¤ This bit indicates that value of acknowledgment field is valid.
      - **ii) RST, SYN & FIN**
        - ¤ These bits are used for connection setup and teardown.
      - **iii) PSH**
        - ¤ This bit indicates the sender has invoked the push operation.
      - **iv) URG**
        - ¤ This bit indicates the segment contains urgent-data.
  - **5) Receive Window**
    - This field defines receiver's window size
    - This field is used for flow control.
  - **6) Checksum**
    - This field is used for error-detection.
  - **7) Urgent Data Pointer**
    - This field indicates the location of the last byte of the urgent data.
  - **8) Options**
    - This field is used when a sender & receiver negotiate the MSS for use in high-speed networks.

2.5.2.1 Sequence Numbers and Acknowledgment Numbers

Sequence Numbers

- The sequence-number is used for sequential numbering of packets of data flowing from sender to receiver.
- Applications:
  - **1)** Gaps in the sequence-numbers of received packets allow the receiver to detect a lost packet.

2) Packets with duplicate sequence-numbers allow the receiver to detect duplicate copies of a packet.

Acknowledgment Numbers

- The acknowledgment-number is used by the receiver to tell the sender that a packet has been received correctly.
- Acknowledgments will typically carry the sequence-number of the packet being acknowledged.

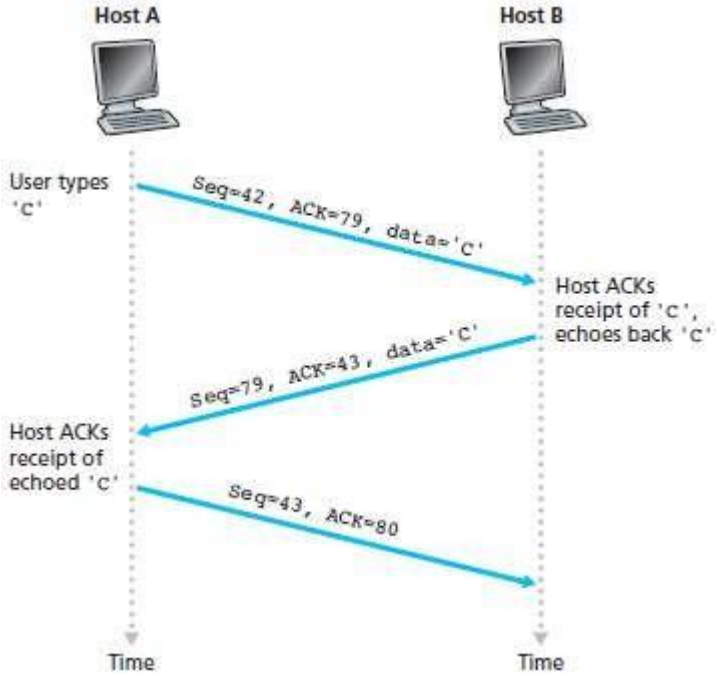


Figure 2.25: Sequence and acknowledgment-numbers for a simple Telnet application over TCP

- Consider an example (Figure 2.25):
  - A process in Host-A wants to send a stream-of-data to a process in Host-B.
  - In Host-A, each byte in the data-stream is numbered as shown in Figure 2.26.

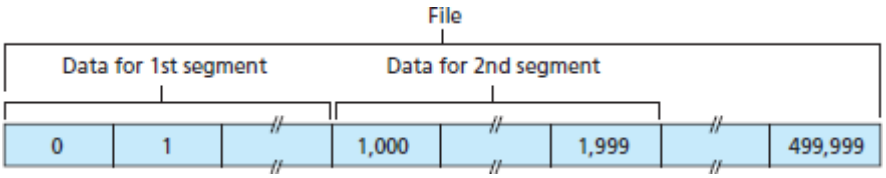


Figure 2.26: Dividing file data into TCP segments

  - The first segment from A to B has a sequence-number 42 i.e. Seq=42.
  - The second segment from B to A has a sequence-number 79 i.e. Seq=79.
  - The second segment from B to A has acknowledgment-number 43, which is the sequence-number of the next byte, Host-B is expecting from Host-A. (i.e. ACK=43).
  - What does a host do when it receives out-of-order bytes?
    Answer: There are two choices:
    1) The receiver immediately discards out-of-order bytes.
    2) The receiver
       → keeps the out-of-order bytes and
       → waits for the missing bytes to fill in the gaps.

2.5.2.2 Telnet: A Case Study for Sequence and Acknowledgment Numbers

- Telnet is a popular application-layer protocol used for remote-login.
- Telnet runs over TCP.

- Telnet is designed to work between any pair of hosts.
- As shown in Figure 2.27, suppose client initiates a Telnet session with server.
- Now suppose the user types a single letter, 'C'.
- Three segments are sent between client & server:

1) First Segment

- The first-segment is sent from the client to the server.
- The segment contains
  - → letter 'C'
  - → sequence-number 42
  - → acknowledgment-number 79

2) Second Segment

- The second-segment is sent from the server to the client.
- Two purpose of the segment:
  - i) It provides an acknowledgment of the data the server has received.
  - ii) It is used to echo back the letter 'C'.
- The acknowledgment for client-to-server data is carried in a segment carrying server-to-clientdata.
- This acknowledgment is said to be piggybacked on the server-to-client data-segment.

3) Third Segment

- The third segment is sent from the client to the server.
- One purpose of the segment:
  - i) It acknowledges the data it has received from the server.

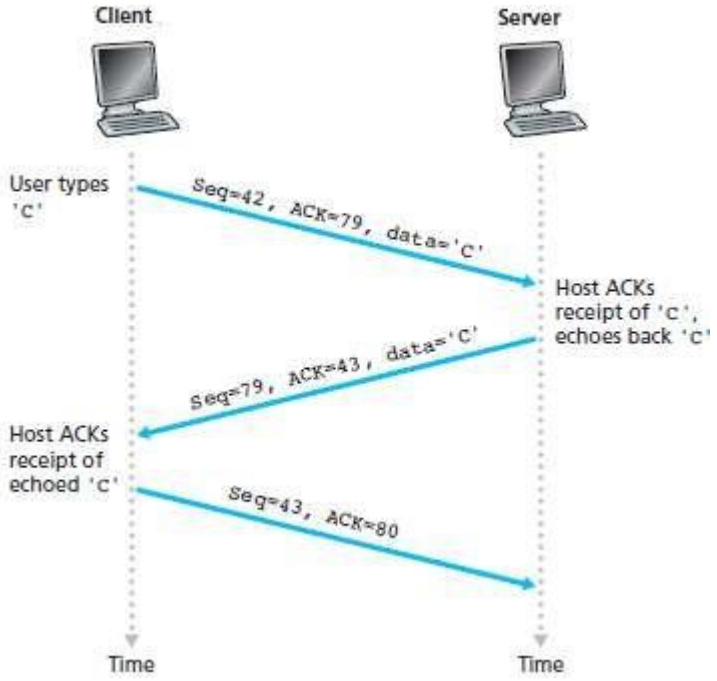


Figure 2.27: Sequence and acknowledgment-numbers for a simple Telnet application over TCP

2.5.3 Round Trip Time Estimation and Timeout

- TCP uses a timeout/retransmit mechanism to recover from lost segments.
- Clearly, the timeout should be larger than the round-trip-time (RTT) of the connection.

2.5.3.1 Estimating the Round Trip Time

- SampleRTT is defined as
  "The amount of time b/w when the segment is sent and when an acknowledgment is received."
- Obviously, the SampleRTT values will fluctuate from segment to segment due to congestion.
- TCP maintains an average of the SampleRTT values, which is referred to as EstimatedRTT.

$$\text{EstimatedRTT} = (1 - \alpha) \cdot \text{EstimatedRTT} + \alpha \cdot \text{SampleRTT}$$

- DevRTT is defined as

  "An estimate of how much SampleRTT typically deviates from EstimatedRTT."

$$DevRTT = (1 - \beta) \cdot DevRTT + \beta \cdot |SampleRTT - EstimatedRTT|$$

- If the SampleRTT values have little fluctuation, then DevRTT will be small.
  If the SampleRTT values have huge fluctuation, then DevRTT will be large.

2.5.3.2 Setting and Managing the Retransmission Timeout Interval

- What value should be used for timeout interval?
- Clearly, the interval should be greater than or equal to EstimatedRTT.
- Timeout interval is given by:

$$TimeoutInterval = EstimatedRTT + 4 \cdot DevRTT$$

2.5.4 Reliable Data Transfer

- IP is unreliable i.e. IP does not guarantee data delivery.
  IP does not guarantee in-order delivery of data.IP does not guarantee the integrity of the data.
- TCP creates a reliable data-transfer-service on top of IP's unreliable-service.
- At the receiver, reliable-service means
  → data-stream is uncorrupted
  → data-stream is without duplication and
  → data-stream is in sequence.

2.5.4.1 A Few Interesting Scenarios

2.5.4.1.1 First Scenario

- As shown in Figure 2.28, Host-A sends one segment to Host-B.
- Assume the acknowledgment from B to A gets lost.
- In this case, the timeout event occurs, and Host-A retransmits the same segment.
- When Host-B receives retransmission, it observes that the sequence-no has already been received.
- Thus, Host-B will discard the retransmitted-segment.

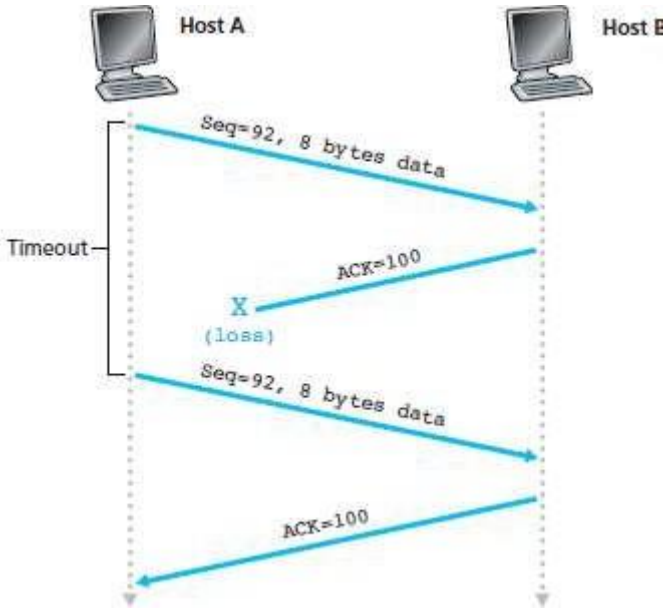


Figure 2.28: Retransmission due to a lost acknowledgment

2.5.4.1.2 Second Scenario

- As shown in Figure 2.29, Host-A sends two segments back-to-back.
- Host-B sends two separate acknowledgments.
- Suppose neither of the acknowledgments arrives at Host-A before the timeout.
- When the timeout event occurs, Host-A resends the first-segment and restarts the timer.
- The second-segment will not be retransmitted until ACK for the second-segment arrives before thenew timeout.

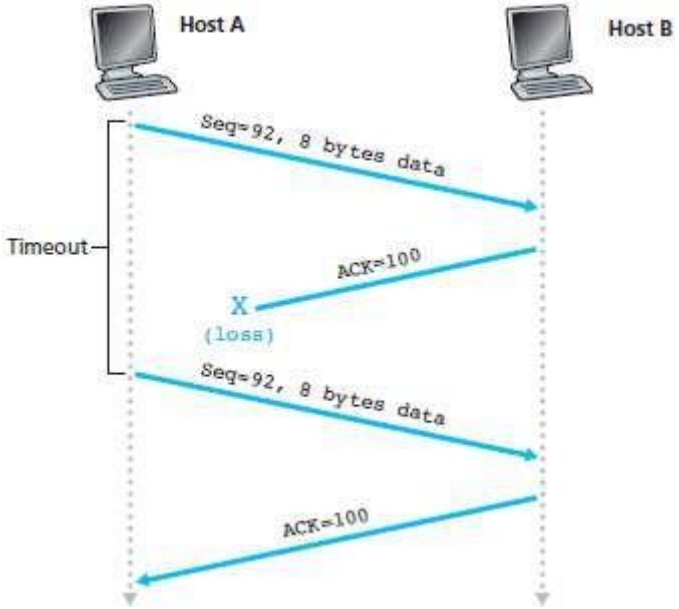
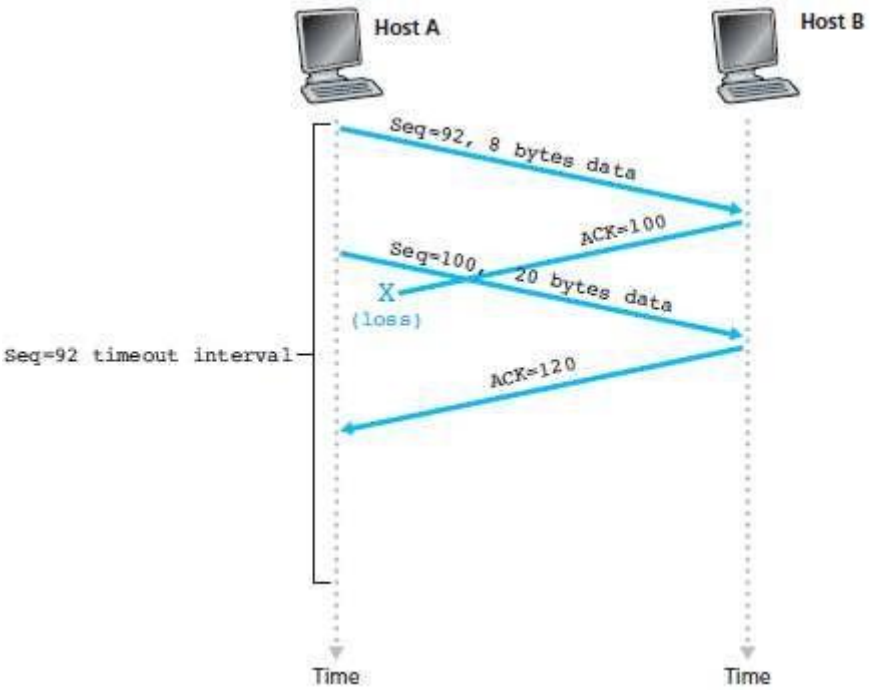


Figure 2.29: Segment 100 not retransmitted

2.5.4.1.3 Third Scenario

- As shown in Figure 2.30, Host-A sends the two segments.
- The acknowledgment of the first-segment is lost.
- But just before the timeout event, Host-A receives an acknowledgment-no 120.
- Therefore, Host-A knows that Host-B has received all the bytes up to 119.
- So, Host-A does not resend either of the two segments.



2.5.4.2 Fast Retransmit

- The timeout period can be relatively long.
- The sender can often detect packet-loss well before the timeout occurs by noting duplicate ACKs.
- A duplicate ACK refers to ACK the sender receives for the second time. (Figure 2.31).

Table 2.3: TCP ACK Generation Recommendation

Event	TCP Receiver Action
Arrival of in-order segment with expected sequence-number. All up to expected sequence-number already acknowledged.	Delayed ACK. Wait up to 500 msec for arrival of another in-order segment. If next in-order segment does not arrive in this interval, send an ACK.
Arrival of in-order segment with expected sequence-number. One other in-order segment waiting for ACK transmission.	Immediately send single cumulative ACK, ACKing both in-order segments.
Arrival of out-of-order segment with higher-than-expected sequence-number. Gap detected.	Immediately send duplicate ACK, indicating sequence-number of next expected-byte.
Arrival of segment that partially or completely fills in gap in received-data.	Immediately send ACK.

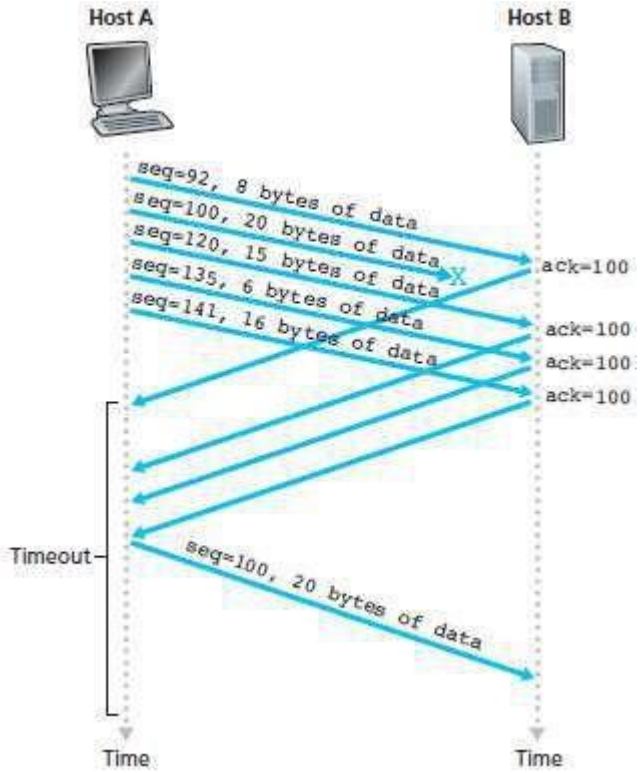


Figure 2.31: Fast retransmit: retransmitting the missing segment before the segment's timer expires

2.5.5 Flow Control

- TCP provides a flow-control service to its applications.
- A flow-control service eliminates the possibility of the sender overflowing the receiver-buffer.

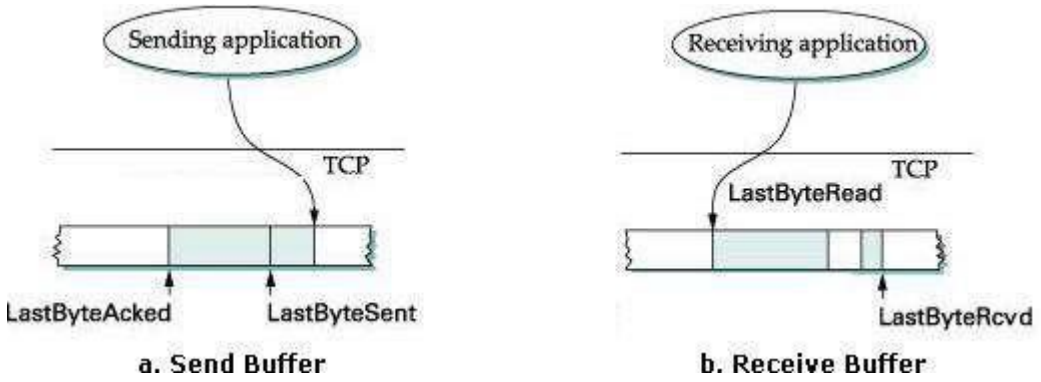


Figure 2.32: a) Send buffer and b) Receive Buffer

- As shown in Figure 2.32, we define the following variables:
  1) MaxSendBuffer: A send-buffer allocated to the sender.
  2) MaxRcvBuffer: A receive-buffer allocated to the receiver.
  3) LastByteSent: The no. of the last bytes sent to the send-buffer at the sender.
  4) LastByteAcked: The no. of the last bytes acknowledged in the send-buffer at the sender.
  5) LastByteRead: The no. of the last bytes read from the receive-buffer at the receiver.
  6) LastByteRcvd: The no. of the last bytes arrived & placed in receive-buffer at the receiver.

Send Buffer

- Sender maintains a send buffer, divided into 3 segments namely
  1) Acknowledged data
  2) Unacknowledged data and
  3) Data to be transmitted
- Send buffer maintains 2 pointers: LastByteAcked and LastByteSent. The relation b/w these two is:

$$\text{LastByteAcked} \leq \text{LastByteSent}$$

Receive Buffer

- Receiver maintains receive buffer to hold data even if it arrives out-of-order.
- Receive buffer maintains 2 pointers: LastByteRead and LastByteRcvd. The relation b/w these two is:

$$\text{LastByteRead} \leq \text{LastByteRcvd} + 1$$

Flow Control Operation

- Sender prevents overflowing of send buffer by maintaining

$$\text{LastByteWritten} - \text{LastByteAcked} \leq \text{MaxSendBuffer}$$

- Receiver avoids overflowing receive buffer by maintaining

$$\text{LastByteRcvd} - \text{LastByteRead} \leq \text{MaxRcvBuffer}$$

- Receiver throttles the sender by advertising a window that is smaller than the amount of free spacethat it can buffer as:

$$\text{AdvertisedWindow} = \text{MaxRcvBuffer} - (\text{LastByteRcvd} - \text{LastByteRead})$$

2.5.6 TCP Connection Management

2.5.6.1 Connection Setup & Data Transfer

- To setup the connection, three segments are sent between the two hosts. Therefore, this process is referred to as a three-way handshake.
- Suppose a client-process wants to initiate a connection with a server-process.
- Figure 2.33 illustrates the steps involved:

Step 1: Client sends a connection-request segment to the Server

- The client first sends a connection-request segment to the server.
- The connection-request segment contains:

1) SYN bit is set to 1.
2) Initial sequence-number (client_isn).

- The SYN segment is encapsulated within an IP datagram and sent to the server.

Step 2: Server sends a connection-granted segment to the Client

- Then, the server
  → extracts the SYN segment from the datagram
  → allocates the buffers and variables to the connection and
  → sends a connection-granted segment to the client.
- The connection-granted segment contains:
  1) SYN bit is set to 1.
  2) Acknowledgment field is set to client_isn+1.
  3) Initial sequence-number (server_isn).

Step 3: Client sends an ACK segment to the Server

- Finally, the client
  → allocates buffers and variables to the connection and
  → sends an ACK segment to the server
- The ACK segment acknowledges the server.
- SYN bit is set to zero, since the connection is established.

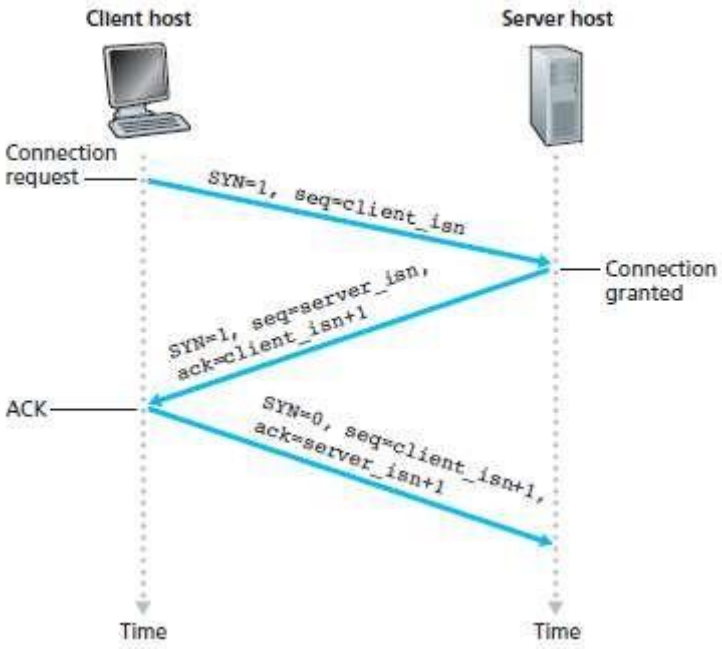


Figure 2.33: TCP three-way handshake: segment exchange

2.5.6.2 Connection Release

- Either of the two processes in a connection can end the connection.
- When a connection ends, the "resources" in the hosts are de-allocated.
- Suppose the client decides to close the connection.
- Figure 2.34 illustrates the steps involved:
  1) The client-process issues a close command.
     ¤ Then, the client sends a shutdown-segment to the server.
     ¤ This segment has a FIN bit set to 1.
  2) The server responds with an acknowledgment to the client.
  3) The server then sends its own shutdown-segment.
     ¤ This segment has a FIN bit set to 1.
  4) Finally, the client acknowledges the server's shutdown-segment.

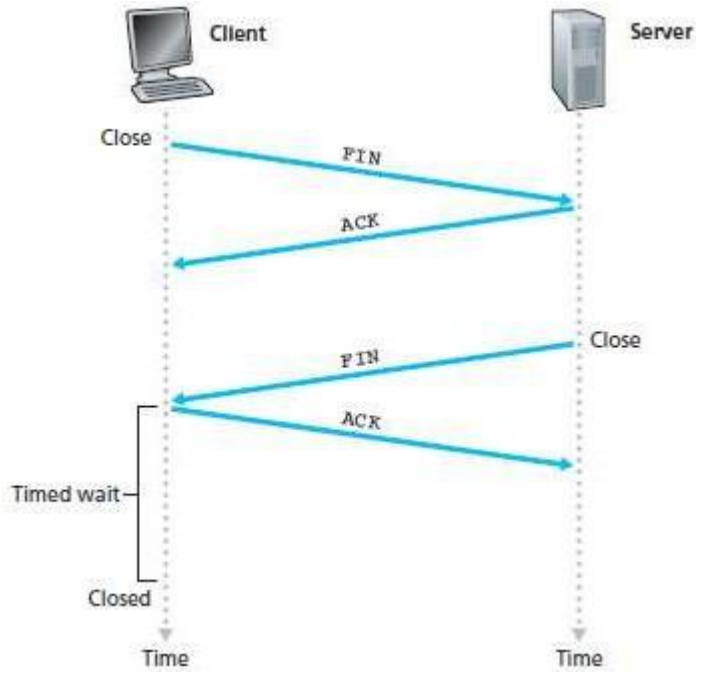


Figure 2.34**:** Closing a TCP connection

2.6 Principles of Congestion Control

2.6.1 The Causes and the Costs of Congestion

2.6.1.1 Scenario 1: Two Senders, a Router with Infinite Buffers

- Two hosts (A & B) have a connection that shares a single-hop b/w source & destination.
- This is illustrated in Figure 2.35.

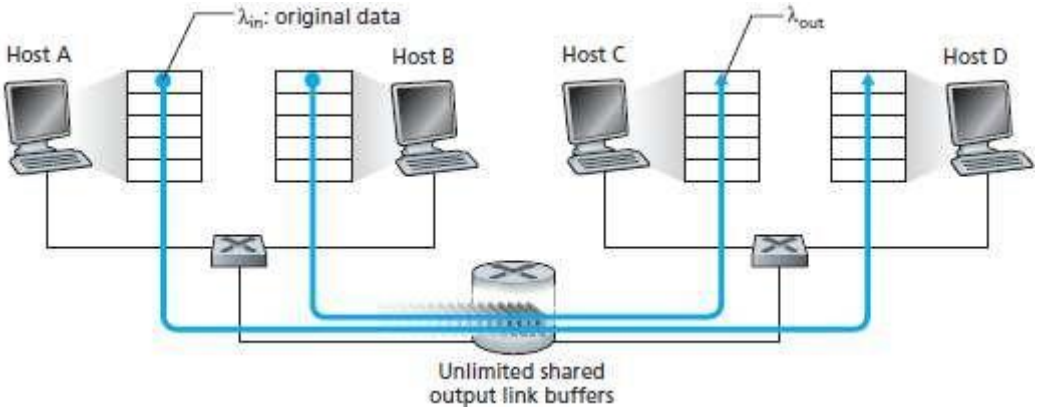


Figure 2.35: Congestion scenario 1: Two connections sharing a single hop with infinite buffers

- Let

  Sending-rate of Host-A = λ_{in} bytes/sec
  Outgoing Link's capacity = R

- Packets from Hosts A and B pass through a router and over a shared outgoing link.
- The router has buffers.
- The buffers stores incoming packets when packet-arrival rate exceeds the outgoing link's capacity.

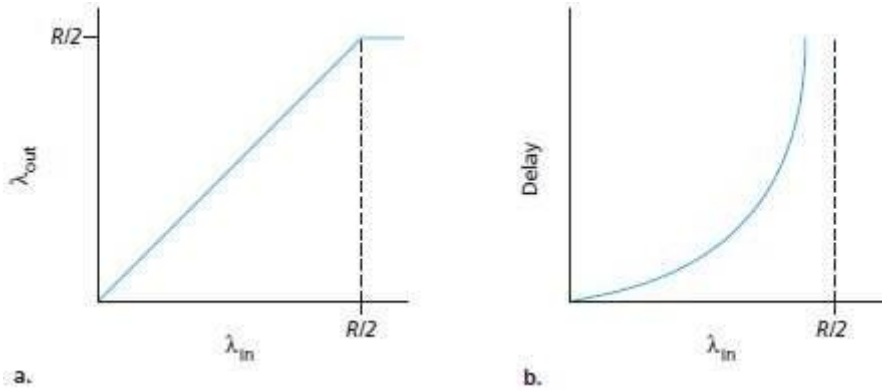


Figure 2.36: Congestion scenario 1: Throughput and delay as a function of host sending-rate

- Figure 2.36 plots the performance of Host-A's connection.

Left Hand Graph

- The left graph plots the per-connection throughput as a function of the connection-sending-rate.
- For a sending-rate b/w 0 and R/2, the throughput at the receiver equals the sender's sending-rate.
  However, for a sending-rate above R/2, the throughput at the receiver is only R/2. (Figure 2.36a)
- Conclusion: The link cannot deliver packets to a receiver at a steady-state rate that exceeds R/2.

Right Hand Graph

- The right graph plots the average delay as a function of the connection-sending-rate (Figure 2.36b).
- As the sending-rate approaches R/2, the average delay becomes larger and larger.
  However, for a sending-rate above R/2, the average delay becomes infinite.
- Conclusion: Large queuing delays are experienced as the packet arrival rate nears the link capacity.

2.6.1.2 Scenario 2: Two Senders and a Router with Finite Buffers

- Here, we have 2 assumptions (Figure 2.37):
  1) The amount of router buffering is finite.
  - Packets will be dropped when arriving to an already full buffer.

  2) Each connection is reliable.
  - If a packet is dropped at the router, the sender will eventually retransmit it.
- Let
  Application's sending-rate of Host-A = λ_{in} bytes/sec
  Transport-layer's sending-rate of Host-A = λ_{in}' bytes/sec (also called offered-load to network)
  Outgoing Link's capacity = R

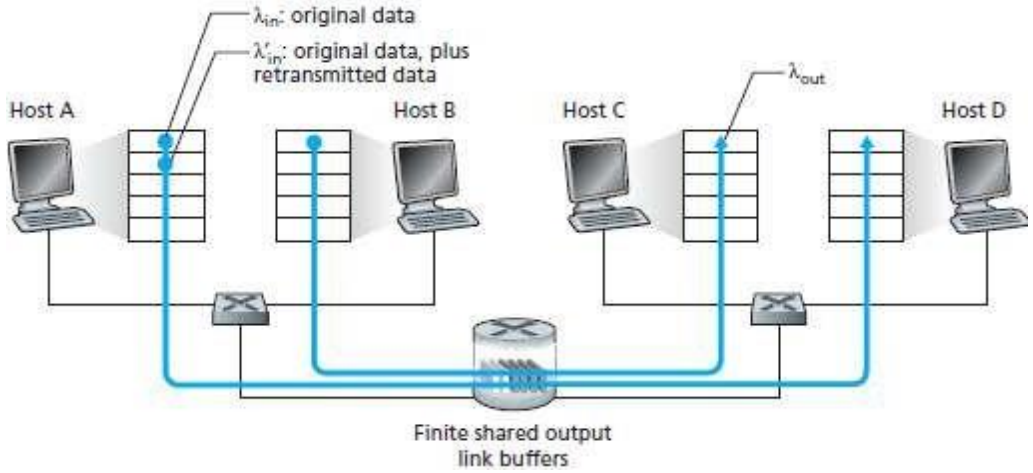


Figure 2.37: Scenario 2: Two hosts (with retransmissions) and a router with finite buffers

Case 1 (Figure 2.38(a)):

- Host-A sends a packet only when a buffer is free.
- In this case,
  - → no loss occurs
  - → λ_{in} will be equal to λ_{in}', and
  - → throughput of the connection will be equal to λ_{in}.
- The sender retransmits only when a packet is lost.
- Consider the offered-load $\lambda_{in}' = R/2$.
- The rate at which data are delivered to the receiver application is R/3.
- The sender must perform retransmissions to compensate for lost packets due to buffer overflow.

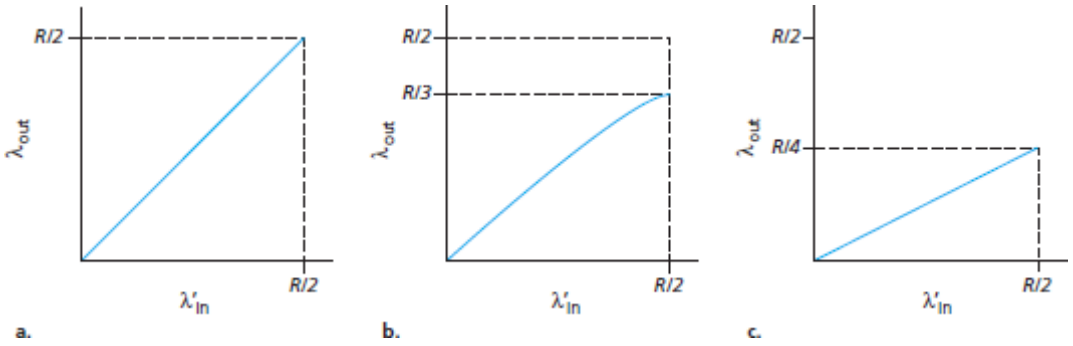


Figure 2.38: Scenario 2 performance with finite buffers

Case 3 (Figure 2.38(c)):

- The sender may time out & retransmit a packet that has been delayed in the queue but not yet lost.
- Both the original data packet and the retransmission may reach the receiver.
- The receiver needs one copy of this packet and will discard the retransmission.
- The work done by the router in forwarding the retransmitted copy of the original packet was wasted.

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

- Four hosts transmit packets, each over overlapping two-hop paths.
- This is illustrated in Figure 2.39.

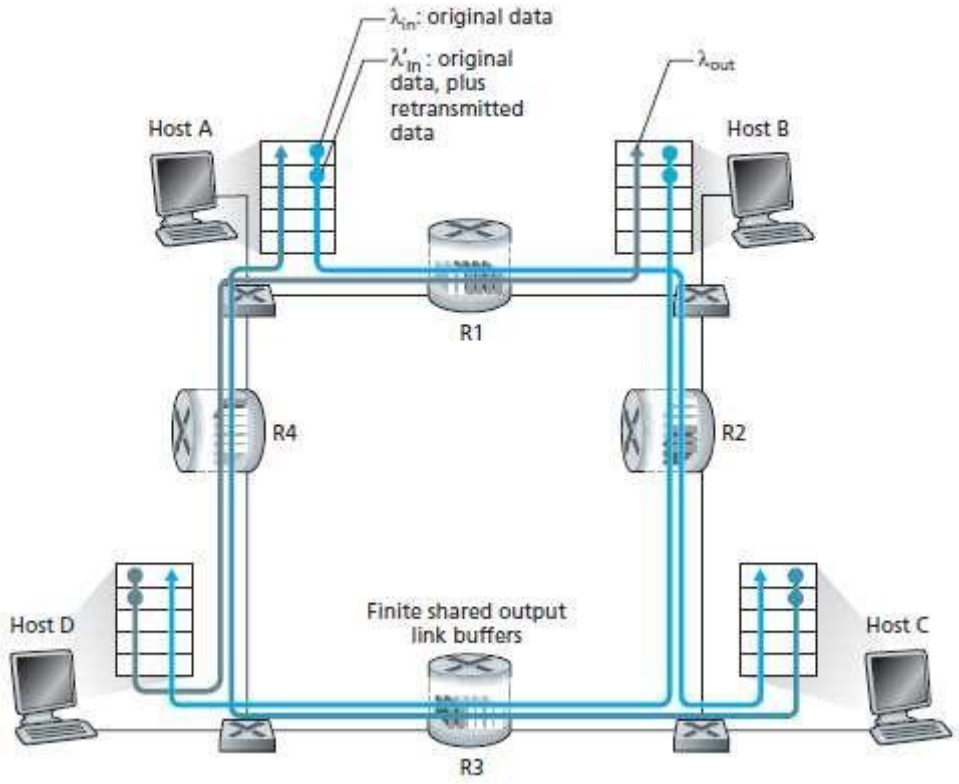


Figure 2.39: Four senders, routers with finite buffers, and multihop paths

- Consider the connection from Host-A to Host C, passing through routers R1 and R2.
- The A–C connection
  → shares router R1 with the D–B connection and
  → shares router R2 with the B–D connection.
- Case-1: For extremely small values of λ_{in},
  → buffer overflows are rare (as in congestion scenarios 1 and 2) and
  → the throughput approximately equals the offered-load.
- Case-2: For slightly larger values of λ_{in}, the corresponding throughput is also larger. This is because
  → more original data is transmitted into the network
  → data is delivered to the destination and
  → overflows are still rare.
- Case-3: For extremely larger values of λ_{in}.
  - Consider router R2.
  - The A–C traffic arriving to router R2 can have an arrival rate of at most R regardless of the value of λ_{in}.
    where R = the capacity of the link from R1 to R2,.
  - If λ_{in}' is extremely large for all connections, then the arrival rate of B–D traffic at R2 can be much larger than that of the A–C traffic.
  - The A–C and B–D traffic must compete at router R2 for the limited amount of buffer-space.
  - Thus, the amount of A–C traffic that successfully gets through R2 becomes smaller and smaller as the offered-load from B–D gets larger and larger.
  - In the limit, as the offered-load approaches infinity, an empty buffer at R2 is immediately filled by a B–D packet, and the throughput of the A–C connection at R2 goes to zero.
  - When a packet is dropped along a path, the transmission capacity ends up having been wasted.

2.6.2 Approaches to Congestion Control

• Congestion-control approaches can be classified based on whether the network-layer provides any explicit assistance to the transport-layer:

1) End-to-end Congestion Control

➢ The network-layer provides no explicit support to the transport-layer for congestion-control.

➢ Even the presence of congestion must be inferred by the end-systems based only on observed network-behavior.

➢ Segment loss is taken as an indication of network-congestion and the window-size is decreased accordingly.

2) Network Assisted congestion Control

➢ Network-layer components provide explicit feedback to the sender regarding congestion.

➢ This feedback may be a single bit indicating congestion at a link.

➢ Congestion information is fed back from the network to the sender in one of two ways:

i) Direct feedback may be sent from a network-router to the sender (Figure 2.40).

¤ This form of notification typically takes the form of a choke packet.

ii) A router marks a field in a packet flowing from sender to receiver to indicate congestion.

¤ Upon receipt of a marked packet, the receiver then notifies the sender of thecongestion indication.

¤ This form of notification takes at least a full round-trip time.

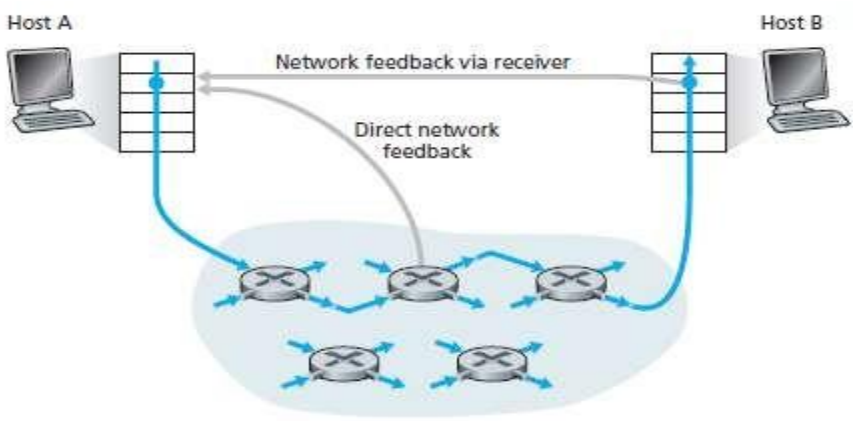


Figure 2.40: Two feedback pathways for network-indicated congestion information

2.6.3 Network Assisted Congestion Control Example: ATM ABR Congestion Control

• ATM (Asynchronous Transfer Mode) protocol uses network-assisted approach for congestion-control.

• ABR (Available Bit Rate) has been designed as an elastic data-transfer-service.

i) When the network is underloaded, ABR has to take advantage of the spare available bandwidth.

ii) When the network is congested, ABR should reduce its transmission-rate.

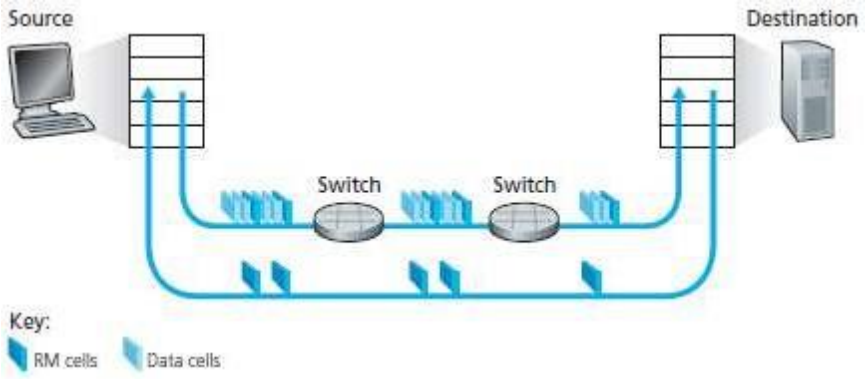


Figure 2.41: Congestion-control framework for ATM ABR service

- Figure 2.41 shows the framework for ATM ABR congestion-control.
- Data-cells are transmitted from a source to a destination through a series of intermediate switches.
- RM-cells are placed between the data-cells. (RM Resource Management).
- The RM-cells are used to send congestion-related information to the hosts & switches.
- When an RM-cell arrives at a destination, the cell will be sent back to the sender
- Thus, RM-cells can be used to provide both
  → direct network feedback and
  → network feedback via the receiver.

2.6.3.1 Three Methods to indicate Congestion

- ATM ABR congestion-control is a rate-based approach.
- ABR provides 3 mechanisms for indicating congestion-related information:

1) EFCI Bit

- Each data-cell contains an EFCI bit. (EFCI : Explicit forward congestion indication)
- A congested-switch sets the EFCI bit to 1 to signal congestion to the destination.
- The destination must check the EFCI bit in all received data-cells.
- If the most recently received data-cell has the EFCI bit set to 1, then the destination
  → sets the CI bit to 1 in the RM-cell (CI : congestion indication)
  → sends the RM-cell back to the sender.
- Thus, a sender can be notified about congestion at a network switch.

2) CI and NI Bits

- The rate of RM-cell interspersion is a tunable parameter.
- The default value is one RM-cell every 32 data-cells. (NI : No Increase)
- The RM-cells have a CI bit and a NI bit that can be set by a congested-switch.
- A switch
  → sets the NI bit to 1 in a RM-cell under mild congestion and
  → sets the CI bit to 1 under severe congestion conditions.

3) ER Setting

- Each RM-cell also contains an ER field. (ER : explicit rate)
- A congested-switch may lower the value contained in the ER field in a passing RM-cell.
- In this manner, ER field will be set to minimum supportable rate of all switches on the path.

2.7 TCP Congestion Control

2.7.1 TCP Congestion Control

- TCP has congestion-control mechanism.
- TCP uses end-to-end congestion-control rather than network-assisted congestion-control
- Here is how it works:
  - Each sender limits the rate at which it sends traffic into its connection as a function of perceived congestion.
    i) If sender perceives that there is little congestion, then sender increases its data-rate.
    ii) If sender perceives that there is congestion, then sender reduces its data-rate.
- This approach raises three questions:
  1) How does a sender limit the rate at which it sends traffic into its connection?
  2) How does a sender perceive that there is congestion on the path?
  3) What algorithm should the sender use to change its data-rate?
- The sender keeps track of an additional variable called the congestion-window (cwnd).
- The congestion-window imposes a constraint on the data-rate of a sender.
- The amount of unacknowledged-data at a sender will not exceed minimum of (cwnd & rwnd), that is:

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

- The sender's data-rate is roughly cwnd/RTT bytes/sec.
- Explanation of Loss event:
  - A "loss event" at a sender is defined as the occurrence of either
    → timeout or
    → receipt of 3 duplicate ACKs from the receiver.
  - Due to excessive congestion, the router-buffer along the path overflows. This causes a

datagram to be dropped.
- The dropped datagram, in turn, results in a loss event at the sender.
- The sender considers the loss event as an indication of congestion on the path.

- How congestion is detected?
  - Consider the network is congestion-free.
  - Acknowledgments for previously unacknowledged segments will be received at the sender.
  - TCP
    → will take the arrival of these acknowledgments as an indication that all is well and
    → will use acknowledgments to increase the window-size (& hence data-rate).
  - TCP is said to be self-clocking because
    → acknowledgments are used to trigger the increase in window-size
  - Congestion-control algorithm has 3 major components:
    1) Slow start
    2) Congestion avoidance and
    3) Fast recovery.

2.7.1.1 Slow Start

- When a TCP connection begins, the value of cwnd is initialized to 1 MSS.
- TCP doubles the number of packets sent every RTT on successful transmission.
- Here is how it works:
  - As shown in Figure 2.42, the TCP
    → sends the first-segment into the network and
    → waits for an acknowledgment.
  - When an acknowledgment arrives, the sender
    → increases the congestion-window by one MSS and
    → sends out 2 segments.
  - When two acknowledgments arrive, the sender
    → increases the congestion-window by one MSS and
    → sends out 4 segments.
  - This process results in a doubling of the sending-rate every RTT.
- Thus, the TCP data-rate starts slow but grows exponentially during the slow start phase.

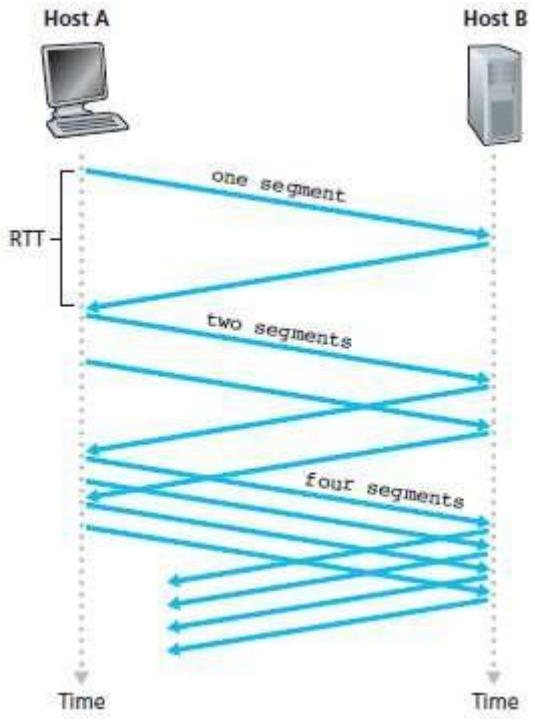


Figure 2.42: TCP slow start

- When should the exponential growth end?
  - Slow start provides several answers to this question.
    - **1)** If there is a loss event, the sender
      - → sets the value of cwnd to 1 and
      - → begins the slow start process again. (ssthresh : "slow start threshold")
      - → sets the value of ssthresh to cwnd/2.
    - **2)** When the value of cwnd equals ssthresh, TCP enters the congestion avoidance state.
    - **3)** When three duplicate ACKs are detected, TCP
      - → performs a fast retransmit and
      - → enters the fast recovery state.
- TCP's behavior in slow start is summarized in FSM description in Figure 2.43.

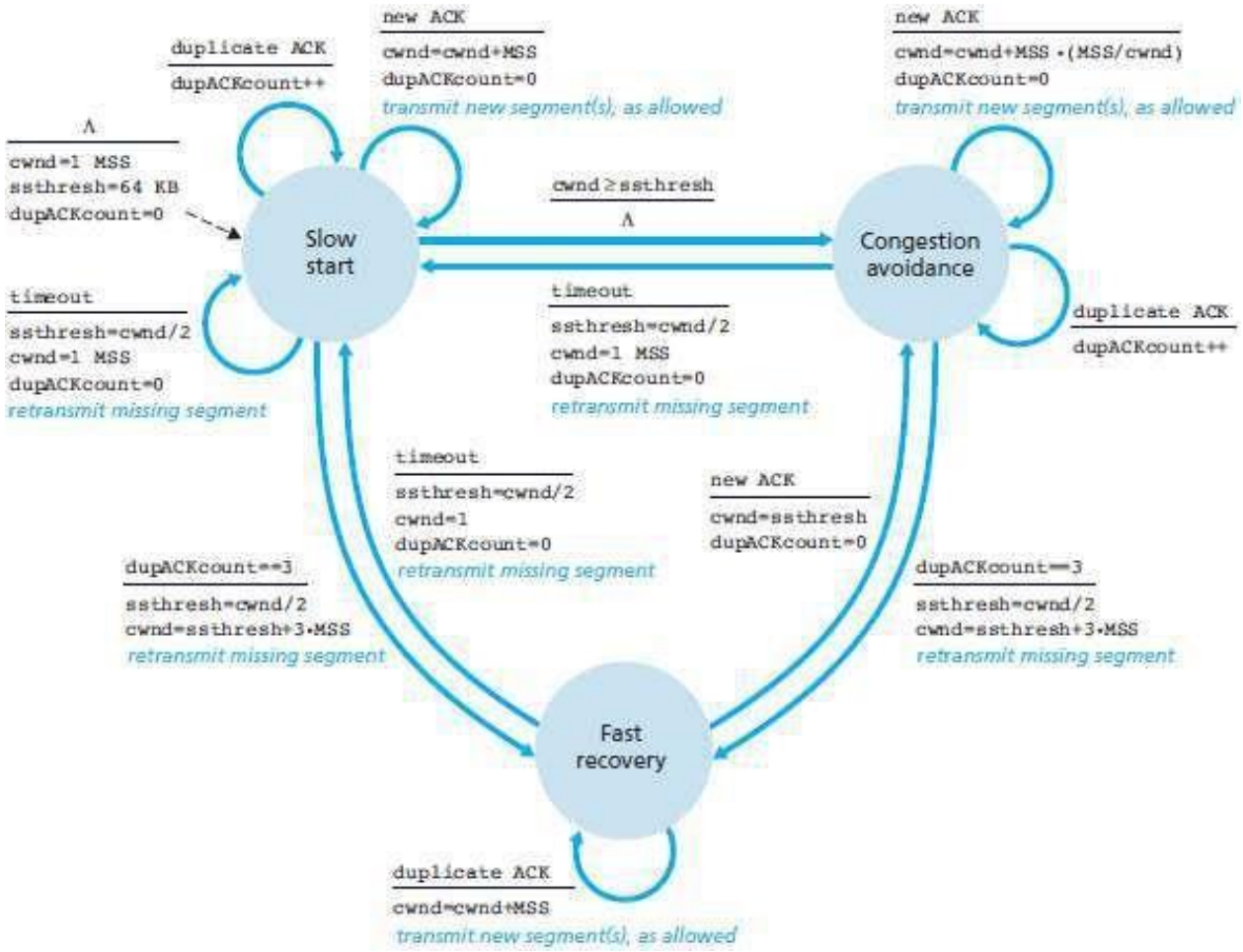


Figure 2.43: FSM description of TCP congestion-control

2.7.1.2 Congestion Avoidance

- On entry to congestion-avoidance state, the value of cwnd is approximately half its previous value.
- Thus, the value of cwnd is increased by a single MSS every RTT.
- The sender must increases cwnd by MSS bytes (MSS/cwnd) whenever a new acknowledgment arrives
- When should linear increase (of 1 MSS per RTT) end?
  - **1) When a timeout occurs.**
    - When the loss event occurred,
      - → value of cwnd is set to 1 MSS and
      - → value of ssthresh is set to half the value of cwnd.
  - **2) When triple duplicate ACK occurs.**
    - When the triple duplicate ACKs were received,

→ value of cwnd is halved.
→ value of ssthresh is set to half the value of cwnd.

2.7.1.3 Fast Recovery

- The value of cwnd is increased by 1 MSS for every duplicate ACK received.
- When an ACK arrives for the missing segment, the congestion-avoidance state is entered.
- If a timeout event occurs, fast recovery transitions to the slow-start state.
- When the loss event occurred
  → value of cwnd is set to 1 MSS, and
  → value of ssthresh is set to half the value of cwnd.
- There are 2 versions of TCP:

1) TCP Tahoe

- An early version of TCP was known as TCP Tahoe.
- TCP Tahoe
  → cut the congestion-window to 1 MSS and
  → entered the slow-start phase after either
  i) timeout-indicated or
  ii) triple-duplicate-ACK-indicated loss event.

2) TCP Reno

- The newer version of TCP is known as TCP Reno.
- TCP Reno incorporated fast recovery.
- Figure 2.44 illustrates the evolution of TCP's congestion-window for both Reno and Tahoe.

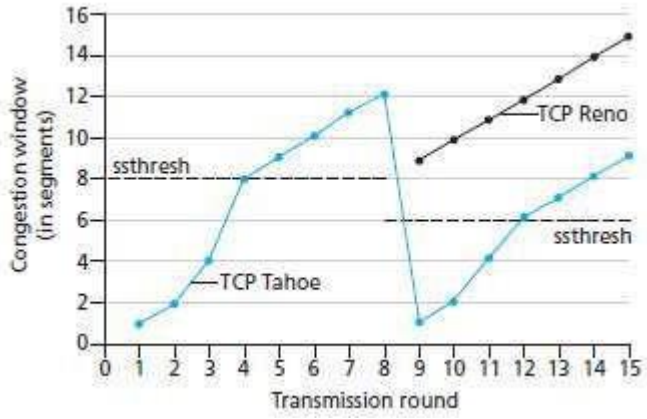


Figure 2.44: Evolution of TCP's congestion-window (Tahoe and Reno)

2.7.1.4 TCP Congestion Control: Retrospective

- TCP's congestion-control consists of (AIMD: additive increase, multiplicative decrease)
  → Increasing linearly (additive) value of cwnd by 1 MSS per RTT and
  → Halving (multiplicative decrease) value of cwnd on a triple duplicate-ACK event.
- For this reason, TCP congestion-control is often referred to as an AIMD.
- AIMD congestion-control gives rise to the "saw tooth" behavior shown in Figure 2.45.
- TCP
  → increases linearly the congestion-window-size until a triple duplicate-ACK event occurs and
  → decreases then the congestion-window-size by a factor of 2

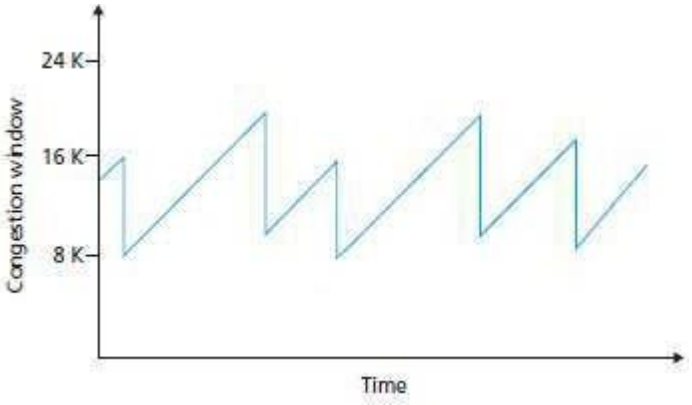


Figure 2.45: Additive-increase, multiplicative-decrease congestion-control

2.7.2 Fairness

- Congestion-control mechanism is fair if each connection gets equal share of the link-bandwidth.
- As shown in Figure 2.46, consider 2 TCP connections sharing a single link with transmission-rate R.
- Assume the two connections have the same MSS and RTT.

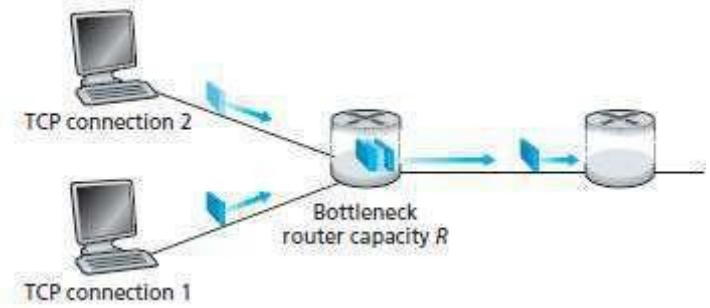


Figure 2.46: Two TCP connections sharing a single bottleneck link

- Figure 2.47 plots the throughput realized by the two TCP connections.
  - If TCP shares the link-bandwidth equally b/w the 2 connections, then the throughput falls along the 45-degree arrow starting from the origin.

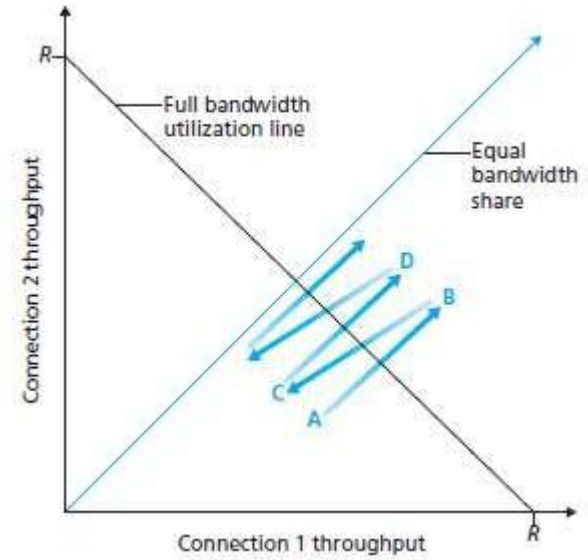


Figure 2.47: Throughput realized by TCP connections 1 and 2

2.7.2.1 Fairness and UDP

- Many multimedia-applications (such as Internet phone) often do not run over TCP.
- Instead, these applications prefer to run over UDP. This is because
  → applications can pump their audio into the network at a constant rate and
  → occasionally lose packets.

2.7.2.2 Fairness and Parallel TCP Connections

- Web browsers use multiple parallel-connections to transfer the multiple objects within a Web page.
- Thus, the application gets a larger fraction of the bandwidth in a congested link.
- '.' Web-traffic is so pervasive in the Internet; multiple parallel-connections are common nowadays.

MODULE 3: THE NETWORK LAYER

3.1 Introduction

3.1.1 Forwarding & Routing

- The role of the network-layer is to move packets from a sending-host to a receiving-host.
- Two important functions of network-layer:
  - **1) Forwarding**
    - Forwarding refers to transferring a packet from incoming-link to outgoing-link within a router.
    - Forwarding is a router-local action.
  - **2) Routing**
    - Routing means determining the path taken by packets from a sender to a receiver.
    - Routing is a network-wide process.

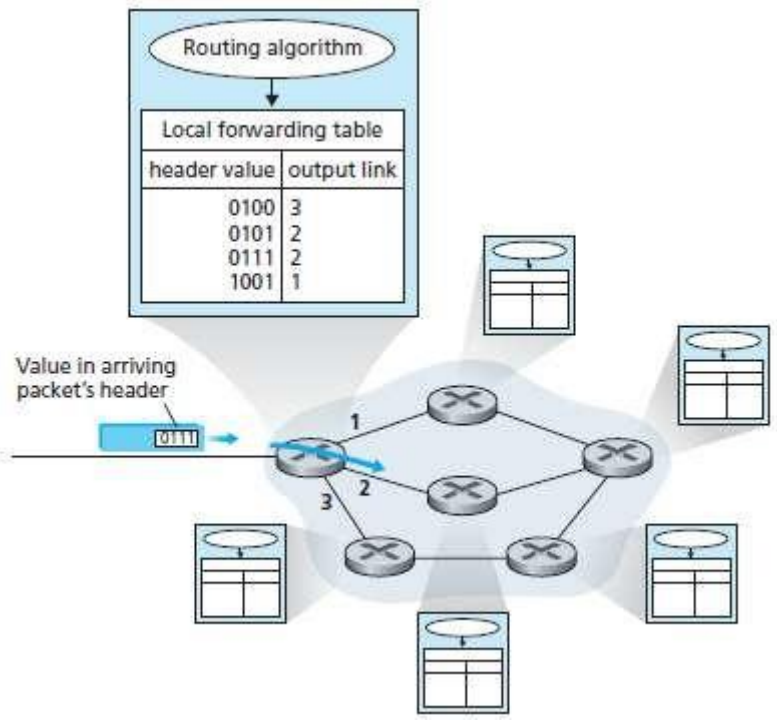


Figure 3.1: Routing-algorithms determine values in forwarding-tables

- The algorithms that determine the paths are referred to as routing-algorithms.
- Each router has a forwarding-table.
- As shown in Figure 3.1, a forwarding-table contains 2 columns:
  1) Header value and
  2) Output link.
- How forwarding is done?
  1) Firstly, a router examines the header-value of an arriving packet.
  2) Then, the router uses the header-value to index into the forwarding-table.
  3) Finally, the router forwards the packet.

3.1.2 Network Service Models

- This defines the characteristics of end-to-end transport of packets b/w sending & receiving systems.
- The network-layer provides following services:
  - **1) Guaranteed Delivery**
    - This service guarantees that the packet will eventually arrive at its destination.
  - **2) Guaranteed Delivery with Bounded Delay**
    - This service guarantees delivery of the packet within a specified host-to-host delay bound.
  - **3) In-order Packet Delivery**

- This service guarantees packets arrive at the destination in the order that they were sent.

4) Guaranteed Minimal Bandwidth

- This service imitates the behavior of a link of a specified bit rate b/w sender & receiver.

5) Guaranteed Maximum Jitter

- This service guarantees

  The amount of time b/w the transmissions of 2 successive packets at the sender is equalto the amount of time b/w their receipts at the destination.

6) Security Services

- The network-layer provides security-services such as
  → confidentiality
  → data-integrity and
  → source-authentication.

- How confidentiality is provided?
  1) Using a secret key, the sender encrypts the data being sent to the receiver.
  2) Then, the receiver decrypts the data using the same secret key.

3.2 Virtual Circuit & Datagram Networks

- A network-layer provides 2 types of services:
  1) Connectionless service and
  2) Connection-oriented service.
- Two categories of computer-networks:
  1) Virtual Circuit (VC) Networks
  - This provides only a connection-oriented service at the network-layer.

  2) Datagram Networks
  - This provides only a connectionless service at the network-layer.
  - For example: The Internet.

3.2.1 Virtual Circuit Networks

- A VC consists of
  1) A path between the source and destination.
  2) VC number: This is one number for each link along the path.
  3) Entries in the forwarding-table in each router.
- A packet belonging to a virtual-circuit will carry a VC number in its header.
- At intervening router, the VC number of traversing packet is replaced with a new VC number.
- The new VC number is obtained from the forwarding-table.

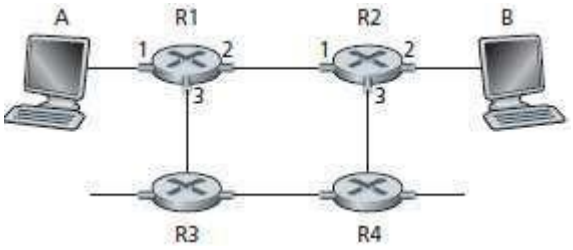


Figure 3.2: A simple virtual-circuit network

Incoming Interface	Incoming VC #	Outgoing Interface	Outgoing VC #
1	12	2	22
2	63	1	18
3	7	2	17
1	97	3	87

Table 3.1: Forwarding-table in R1

- Q: How does router determine the replacement VC number for a packet traversing the router?
  Answer: Each router's forwarding-table includes VC number translation.
- The forwarding-table in R1 is shown in Table 3.1.

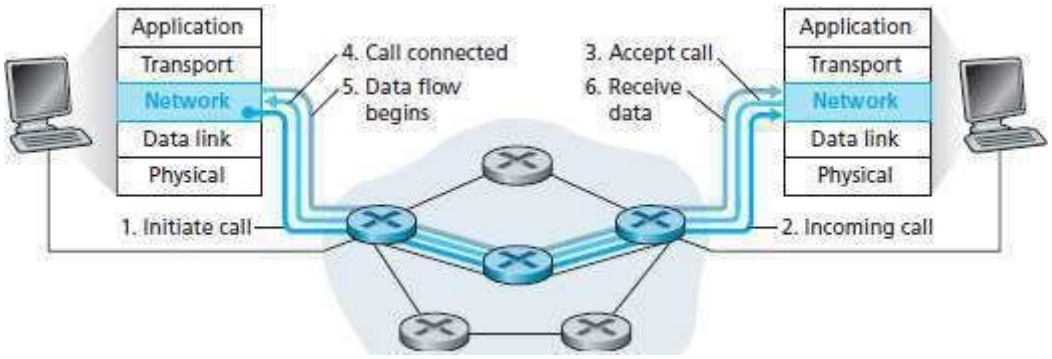


Figure 3.3: Virtual-circuit setup

- Why a packet does not use the same VC number on each link along the path?

Answer: 1) Replacing the number from link to link reduces length of the VC field in the packet-header.
2) VC setup is simplified by permitting a different VC number at each link along the path.

- Disadvantage:
  The routers must maintain connection state information for the ongoing connections.
- Three phases in a virtual-circuit (Figure 3.3):

1) VC Setup

- During the setup phase, the sending transport-layer
  → contacts the network-layer
  → specifies the receiver's address and
  → waits for the network to set-up the VC.
- The network-layer determines the path between sender and receiver.
- The network-layer also determines the VC number for each link along the path.
- Finally, the network-layer adds an entry in the forwarding-table in each router.
- During VC setup, the network-layer may also reserve resources.

2) Data Transfer

- Once the VC has been established, packets can begin to flow along the VC.

3) VC Teardown

- This is initiated when the sender/receiver wants to terminate the VC.
- The network-layer
  → informs the other end-system of the call termination and
  → removes the appropriate entries in the forwarding-table in each router.

3.2.2 Datagram Networks

- The source attaches the packet with the address of the destination.
- The packets are injected into the network.
- The packets are routed independent of each other.
- No advance circuit setup is needed. So, routers do not maintain any connection state information.
- As a packet is transmitted from source to destination, it passes through a series of routers.
- Each router uses the packet's destination-address to forward the packet.

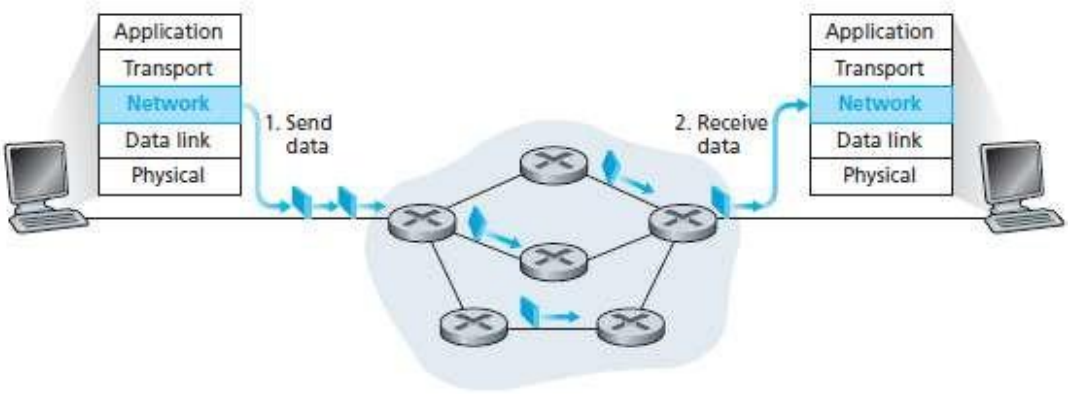


Figure 3.4: Datagram network

- Suppose the router R1 has four links, numbered 0 through 3 (Figure 3.4).
- Forwarding-table of R1 is as follows (Table 3.2):

Prefix Match	Link Interface
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
otherwise	3

Table 3.2: Forwarding-table of R1

- The router matches a prefix of the packet's destination-address with the entries in the table;
  1) If both are equal, the router forwards the packet to an associated link. (0, 1 or 2)
  2) If both are unequal, the router forwards the packet to a default link (otherwise 3).
- When there are multiple matches, the router uses the longest prefix matching rule.

3.2.3 Comparison of Virtual Circuit & Datagram

Issue	Datagram	Virtual Circuit
Connection Setup	None	Required
Addressing	Packet contains full source and destination-address	Packet contains short virtual-circuit number identifier
State Information	None other than router table containing destination-network	Each virtual-circuit number entered totable on setup, used for routing
Routing	Packets routed independently	Route established at setup, all packets follow same route
Effect of Router Failure	Only on packets lost during crash	All virtual circuits passing through failed router terminated

3.3 What's Inside a Router?

- The router is used for transferring packets from an incoming-links to the appropriate outgoing-links.

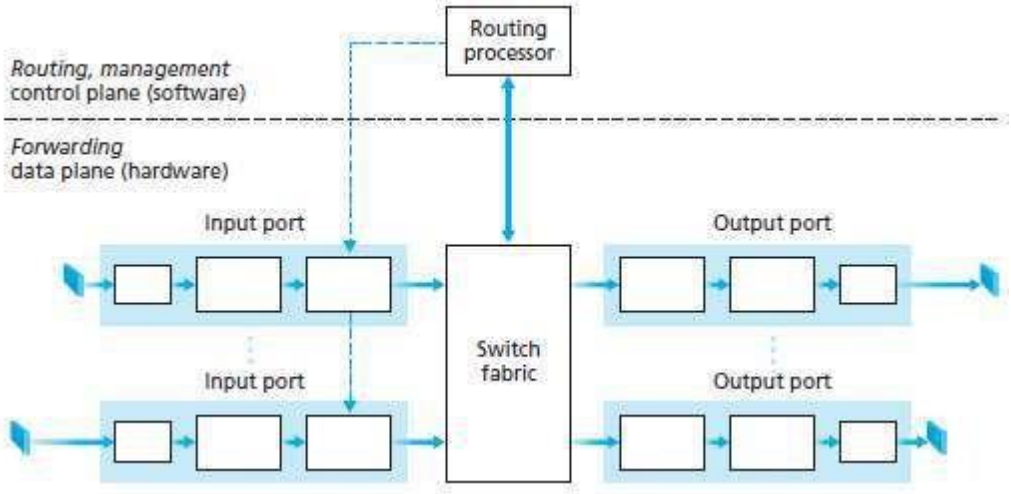


Figure 3.5: Router architecture

- Four components of router (Figure 3.5):

1) Input Ports

- An input-port is used for terminating an incoming physical link at a router (Figure 3.6).
- It is used for interoperating with the link layer at the other side of the incoming-link.
- It is used for lookup function i.e. searching through forwarding-table looking for longest prefix match.
- It contains forwarding-table.
- Forwarding-table is consulted to determine output-port to which arriving packet will be forwarded.
- Control packets are forwarded from an input-port to the routing-processor.
- Many other actions must be taken:
  - i) Packet's version number, checksum and time-to-live field must be checked.
  - ii) Counters used for network management must be updated.

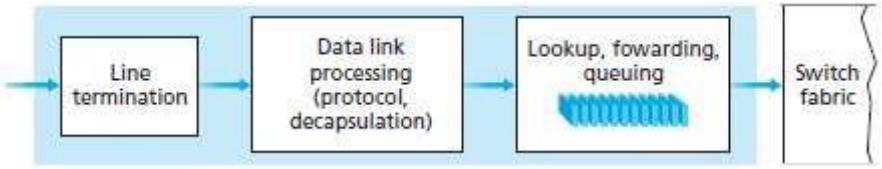


Figure 3.6: Input port processing

2) Switching Fabric

- The switching fabric connects the router's input-ports to its output-ports.
- In fabric, the packets are switched (or forwarded) from an input-port to an output-port.
- In fact, fabric is a network inside of a router.
- A packet may be temporarily blocked if packets from other input-ports are currently using the fabric.
- A blocked packet will be queued at the input-port & then scheduled to send at a later point in time.

3) Output Ports

- An output-port
  - → stores packets received from the switching fabric and
  - → transmits the packets on the outgoing-link.
- For a bidirectional link, an output-port will typically be paired with the input-port.

4) Routing Processor

- The routing-processor
  - → executes the routing protocols

→ maintains routing-tables & attached link state information and
→ computes the forwarding-table.

- It also performs the network management functions.

3.3.1 Switching

- Three types of switching fabrics (Figure 3.7)
  - **1)** Switching via memory
  - **2)** Switching via a bus and
  - **3)** Switching via an interconnection network.

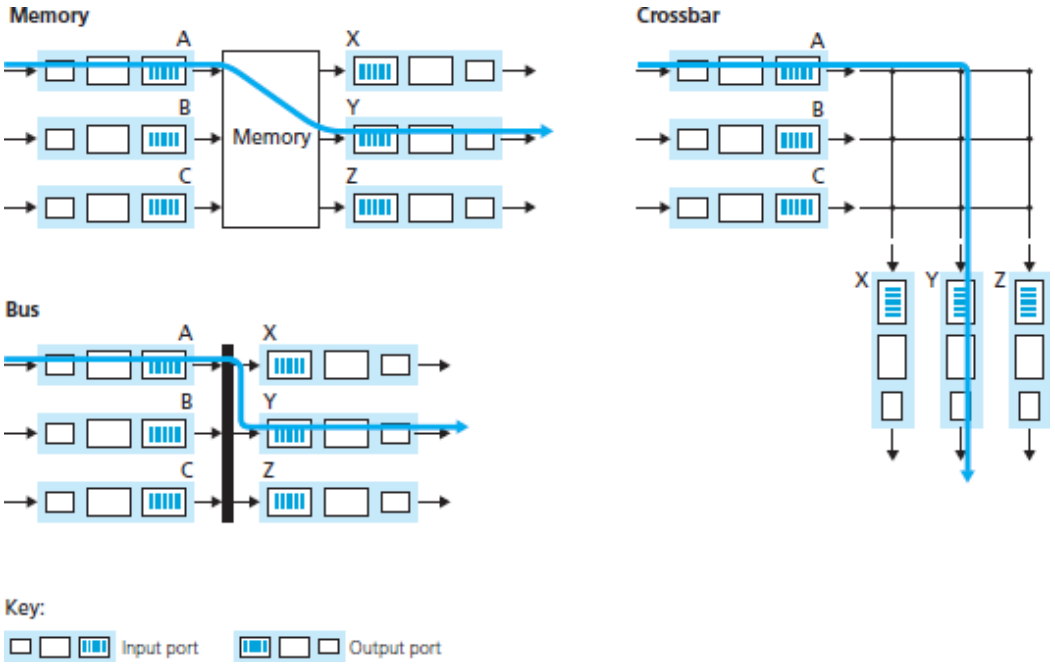


Figure 3.7: Three switching techniques

3.3.1.1 Switching via Memory

- Switching b/w input-ports & output-ports is done under direct control of CPU i.e. routing-processor.
- Input and output-ports work like a traditional I/O devices in a computer.
- Here is how it works (Figure 3.7a):
  - i) On arrival of a packet, the input-port notifies the routing-processor via an interrupt.
  - ii) Then, the packet is copied from the input-port to processor-memory.
  - iii) Finally, the routing-processor
    - → extracts the destination-address from the header
    - → looks up the appropriate output-port in the forwarding-table and
    - → copies the packet into the output-port's buffers.
- Let memory-bandwidth = B packets per second.
  Thus, the overall forwarding throughput must be less than B/2.
- Disadvantage:
  - ➢ Multiple packets cannot be forwarded at the same time. This is because
    - → only one memory read/write over the shared system bus can be done at a time.

3.3.1.2 Switching via a Bus

- Switching b/w input-ports & output-ports is done without intervention by the routing-processor.
- Here is how it works (Figure 3.7b)
  - i) The input-port appends a switch-internal label (header) to the packet.
  - ➢ The label indicates the local output-port to which the packet must be transferred.
  - ii) Then, the packet is received by all output-ports.
  - ➢ But, only the port that matches the label will keep the packet.
  - iii) Finally, the label is removed at the output-port.

- Disadvantages:
  i) Multiple packets cannot be forwarded at the same time. This is because
  → only one packet can cross the bus at a time.
  ii) The switching speed of the router is limited to the bus-speed.

3.3.1.3 Switching via an Interconnection Network

- A crossbar switch is an interconnection network.
- The network consists of 2N buses that connect N input-ports to N output-ports.
- Each vertical bus intersects each horizontal bus at a cross point.
- The cross point can be opened or closed at any time by the switch-controller.
- Here is how it works (Figure 3.7c)
  1) To move a packet from port A to port Y, the switch-controller closes the crosspoint at the intersection of buses A and Y.
  2) Then, port A sends the packet onto its bus, which is picked up by bus Y.
- Advantage:
  - Crossbar networks are capable of forwarding multiple packets in parallel.
  - For ex: A packet from port B can be forwarded to port X at the same time. This is because
    → A-to-Y and B-to-X packets use different input and output buses.
- Disadvantage:
  - If 2 packets have to use same output-port, then one packet has to wait. This is because
    → only one packet can be sent over any given bus at a time.

3.3.2 Output Processing

- Output-port processing
  → takes the packets stored in the output-port's memory and
  → transmits the packets over the output link (Figure 3.8).
- This includes
  → selecting and de-queueing packets for transmission and
  → performing the link layer and physical-layer transmission functions.

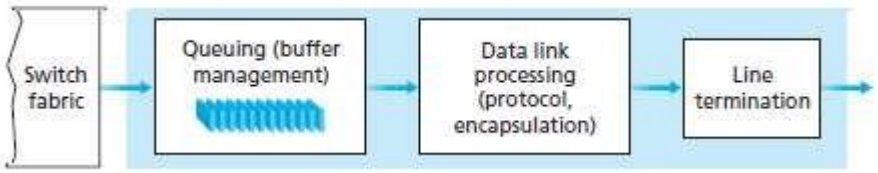


Figure 3.8: Output port processing

3.3.3 Where Does Queueing Occur?

- Packet queues may form at both the input-ports & the output-ports (Figure 3.9).
- As the queues grow large, the router's memory can be exhausted and packet loss will occur.
- The location and extent of queueing will depend on
  1) The traffic load
  2) The relative speed of the switching fabric and
  3) The line speed
- Switching fabric transfer rate R_{switch} is defined as
  "The rate at which packets can be moved from input-port to output-port".
- If R_{switch} is N times faster than R_{line}, then only negligible queuing will occur at the input-ports.

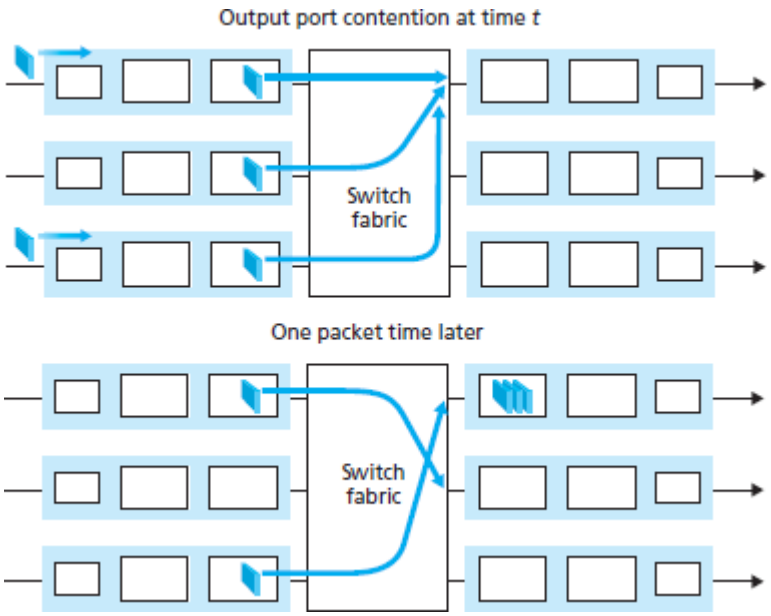


Figure 3.9: Output port queuing

- At output-port, packet-scheduler is used to choose one packet among those queued for transmission.
- The packet-scheduling can be done using
  → first-come-first-served (FCFS) or
  → weighted fair queuing (WFQ).
- Packet scheduling plays a crucial role in providing QoS guarantees.
- If there is less memory to hold an incoming-packet, a decision must be made to either
  1) Drop the arriving packet (a policy known as drop-tail) or
  2) Remove one or more already-queued packets to make room for the newly arrived packet.

3.4 IP: Forwarding & Addressing in the Internet

- IP(Internet Protocol) is main protocol responsible for packetizing, forwarding & delivery of a packet at network-layer.
- It is a connection-less & unreliable protocol.
  i) Connection-less means there is no connection setup b/w the sender and the receiver.
  ii) Unreliable protocol means
  → IP does not make any guarantee about delivery of the data.
  → Packets may get dropped during transmission.
- It provides a best-effort delivery service.
- Best effort means IP does its best to get the packet to its destination, but with no guarantees.
- If reliability is important, IP must be paired with a TCP which is reliable transport-layer protocol.
- IP does not provide following services
  → flow control
  → error control
  → congestion control services.

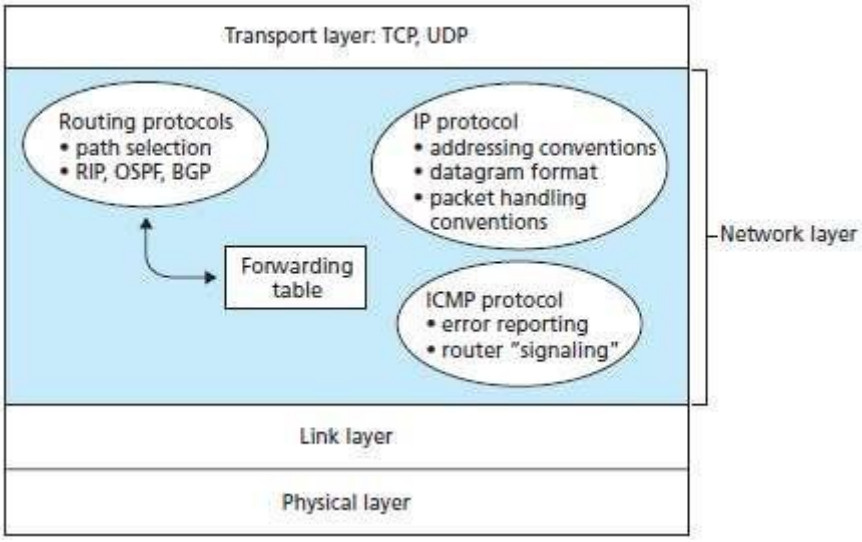


Figure 3.10: A look inside the Internet's network-layer

- Two important components of IP:
  1) Internet addressing and
  2) Forwarding
- There are two versions of IP in use today.
  1) IP version 4 (IPv4) and
  2) IP version 6 (IPv6)
- As shown in Figure 3.10, the network-layer has three major components:
  1) IP protocol
  2) Routing component determines the path a data follows from source to destination
  3) Network-layer is a facility to report errors in datagrams

3.4.1 IPv4 Datagram Format

- IP uses the packets called datagrams.
- A datagram consist of 2 parts: 1) Payload (or Data) 2) Header.

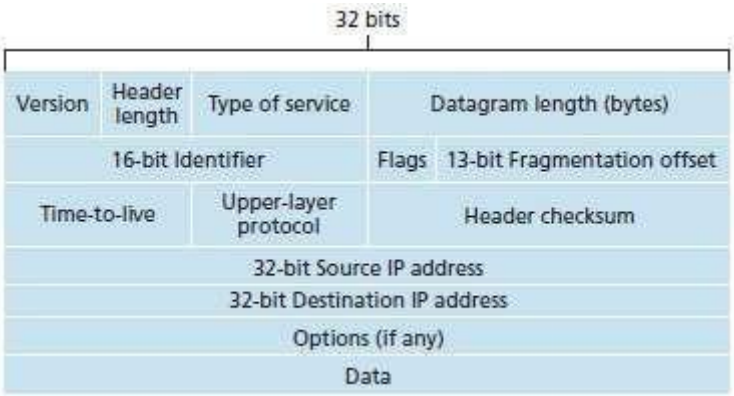


Figure 3.11: IPv4 datagram format

1) Payload (or Data)

- This field contains the data to be delivered to the destination.

2) Header

- Header contains information essential to routing and delivery.
- IP header contains following fields (Figure 3.11):

1) Version
- This field specifies version of the IPv4 datagram, i.e. 4.

2) Header Length
- This field specifies length of header.
- Without options field, header-length = 5 bytes.

3) Type of Service (TOS)
- This field specifies priority of packet based on parameters such as delay, throughput,reliability & cost.

4) Datagram Length
- This field specifies the total length of the datagram (header + data).
- Maximum length = 65535 bytes.

5) Identifier, Flags, Fragmentation Offset
- These fields are used for fragmentation and reassembly.
- Fragmentation occurs when the size of the datagram is larger than the MTU of the network.

i) Identifier: This field uniquely identifies a datagram packet.

ii) Flags: It is a 3-bit field. The first bit is not used.
The second bit D is called the do not fragment bit.
The third bit M is called the more fragment bit.

iii) Fragmentation Offset: This field identifies location of a fragment in a datagram.

6) Time-To-Live (TTL)
- This defines lifetime of the datagram (default value 64) in hops.
- Each router decrements TTL by 1 before forwarding. If TTL is zero, the datagram is discarded.

7) Protocol
- This field specifies upper-layer protocol used to receive the datagram at the destination-host.
- For example, TCP=6 and UDP=17.

8) Header Checksum
- This field is used to verify integrity of header only.
- If the verification process fails, the packet is discarded.

9) Source IP Address & Destination IP Address
- These fields contain the addresses of source and destination respectively.

10) Options
- This field allows the packet to request special features such as
  → security level
  → route to be taken by packet at each router.

3.4.2 Fragmentation

3.4.2.1 Maximum Transfer Unit

- Each network imposes a restriction on maximum size of packet that can be carried. This is called the MTU (maximum transmission unit).
- For example:
  MTU Ethernet = 1500 bytes
  MTU FDDI = 4464 bytes
- Fragmentation means
  "The datagram is divided into smaller fragments when size of a datagram is larger than MTU"
- Each fragment is routed independently (Figure 3.12).
- A fragmented datagram may be further fragmented, if it encounters a network with a smaller MTU.
- Source/router is responsible for fragmentation of original datagram into the fragments.
  Only destination is responsible for reassembling the fragments into the original datagram.

3.4.2.2 Fields Related to Fragmentation & Reassembly

- Three fields in the IP header are used to manage fragmentation and reassembly:
  1) Identification 2) Flags 3) Fragmentation offset.

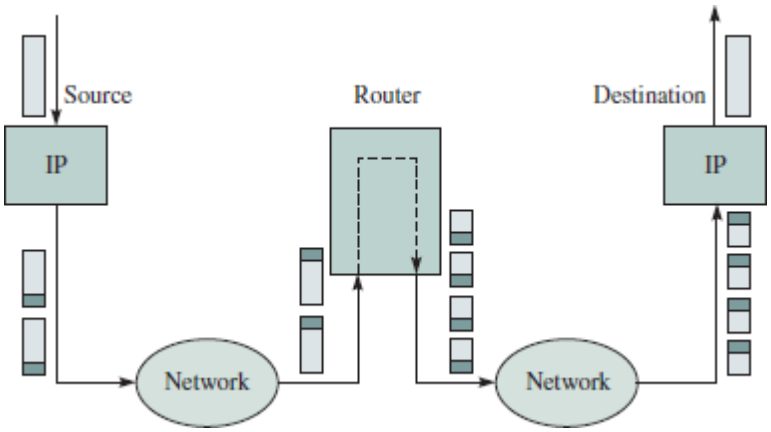


Figure 3.12: IP fragmentation and reassembly

1) Identification

- This field is used to identify to which datagram a particular fragment belongs to (so that fragmentsfor different packets do not get mixed up).
- When a datagram is created, the source attaches the datagram with an identification-number.
- When a datagram is fragmented, the value in the identification-field is copied into all fragments.
- The identification-number helps the destination in reassembling the datagram.

2) Flags

- This field has 3 bits.
  - i) The first bit is not used.
  - ii) DF bit (Don't Fragment):
    - a) If DF=1, the router should not fragment the datagram. Then, the router discards the datagram.
    - b) If DF=0, the router can fragment the datagram.
  - iii) MF bit (More Fragment):
    - a) If MF=1, there are some more fragments to come.
    - b) If MF=0, this is last fragment.

3) Fragmentation Offset

- This field identifies location of a fragment in a datagram.
- This field is the offset of the data in the original datagram.

3.4.3 IPv4 Addressing

3.4.4

- IP address is a numeric identifier assigned to each machine on the internet.
- IP address consists of two parts: network ID(NID) and host ID(HID).
  - **1)** NID identifies the network to which the host is connected. All the hosts connected to the same network have the same NID.
  - **2)** HID is used to uniquely identify a host on that network.
- HID is assigned by the network-administrator at the local site.
  NID for an organization may be assigned by the ISP (Internet Service Provider).
- IPv4 uses 32-bit addresses, i.e., approximately 4 billion addresses (2^{32}).
- IP addresses are usually written in dotted-decimal notation. The address is broken into four bytes.For example, an IP address of
  10000000 10000111 01000100 00000101
  is written as
  128.135.68.5
- IP address can be classified as
  1) Classful IP addressing &

2) Classless IP addressing (CIDR : Classless Inter Domain Routing)

3.4.3.1 IPv4 Classful Addressing

- In classful addressing, the address space is divided into five classes: A, B, C, D and E.
- IP address class is identified by MSBs in binary.
- Classes A, B and C are used for unicast addressing. (Figure 3.13).
- Class D was designed for multicasting and class E is reserved.

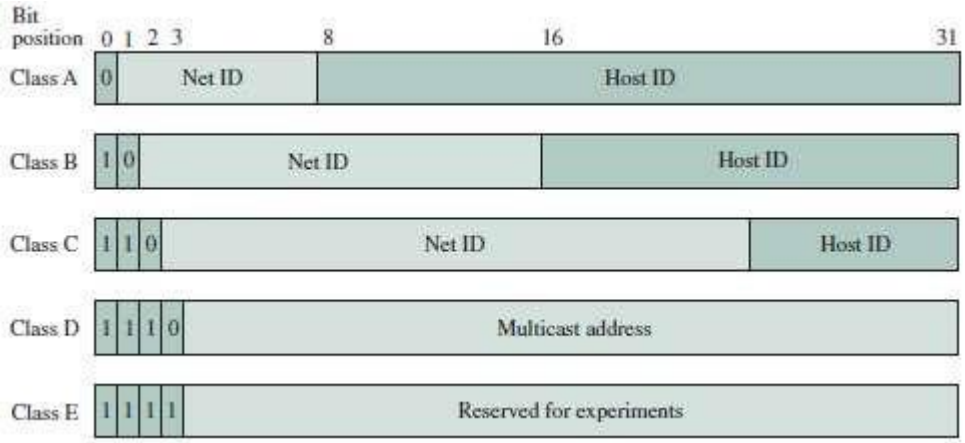


Figure 3.13: The five classes of IP addresses

Class	No. of networks	Max. No. of hosts per network	Designed for
A	126	$2^{24} - 2$	WAN
B	16,382	65,534	Campus networks
C	2^{21}	254	LAN

Table 3.3: Classful Addressing

- Analysis:
  - In c1assful addressing, a large part of the available addresses were wasted, since Class Aand B were too large for most organizations (Table 3.3).
  - Class C is suited only for small organization and reserved addresses were sparingly used.
  -
  -
  -
  - Consider an organization has a Class B address which can support about 64,000 hosts.
  - It will be a huge task for the network-administrator to manage all 64,000 hosts.
- Solution: Use subnet addressing.
- Subnetting reduces the total number of network-numbers by assigning a single network-number tomany adjacent physical networks.
- Each adjacent physical network is referred to as subnet. (Figure 3.14).
- All nodes on a subnet are configured with a subnet mask. For example: 255.255.255.0.
- The 1's in the subnet-mask represent the positions that refer to the network or subnet-numbers.The 0's represent the positions that refer to the host part of the address.
- The bitwise AND of IP address and its subnet mask gives the subnet number.
- Advantage:
  - The subnet-addressing scheme is oblivious to the network outside the organization.
  - Inside the organization the network-administrator is free to choose any combination of

lengths for the subnet & host ID fields.

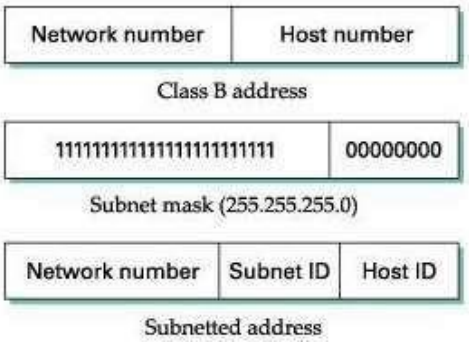


Figure 3.14: Subnet addressing

Question: If a packet with a destination IP address of 150.100.12.176 arrives at site from the outside network, which subnet should a router forward this packet to? Assume subnet mask is 255.255.255.128 (Figure 3.15).

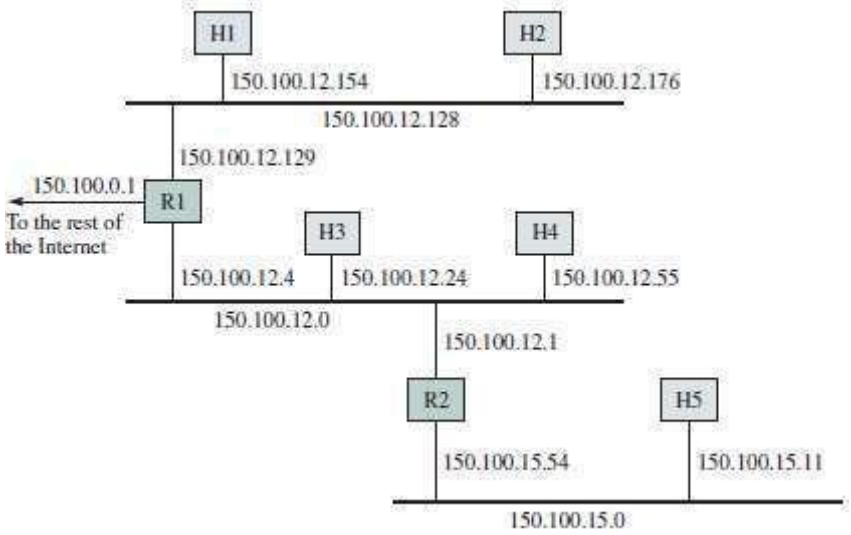


Figure 3.15: Example of address assignment with sub-netting

Solution: The router can determine the subnet number by performing a binary AND between thesubnet mask and the IP address.

IP address:	10010110	01100100	00001100	10110000(150.100.12.176)
Subnet mask:	11111111	11111111	11111111	10000000(255.255.255.128)
Subnet number:	10010110	01100100	00001100	10000000(150.100.12.128)

This number (150.100.12.128) is used to forward the packet to the correct subnet work inside the organization.

3.4.3.2 CIDR

- Problem with class full IP addressing:
  - Consider an organization needs about 500 hosts.
  - Obviously, the organization will get a Class B license, even though it has far fewer than64,000 hosts.
  - At most, over 64,000 addresses can go unused.
  - This results in inefficient usage of the available address-space.
- Solution: Use CIDR (Classless Inter Domain Routing).
  - A single IP address can be used to designate many unique IP addresses. This is called super netting.
  - A CIDR IP address looks like a normal IP address except that
    → the address ends with a slash followed by a number, called the IP network prefix.
    For ex: 205.100.0.0/22

- CIDR addresses
  → reduce the size of routing-tables and
  → make more IP addresses available within organizations.

3.4.3.3 Obtaining a Block of Addresses

- To obtain a block of IP addresses for use within an organization's subnet, a network-administrator contacts the ISP.
- IP addresses are managed under the authority of the ICANN.
- The responsibility of the ICANN (Internet Corporation for Assigned Names and Numbers):
  → to allocate IP addresses,
  → to manage the DNS root servers.
  → to assign domain names and resolve domain name disputes.
  → to allocate addresses to regional Internet registries.

3.4.3.4 Obtaining a Host Address: DHCP

- Two ways to assign an IP address to a host:

1) Manual Configuration

- Operating systems allow system-administrator to manually configure IP address.

2) Dynamic Host Configuration Protocol (DHCP)

- DHCP enables auto-configuration of IP address to host.

3.4.3.4.1 DHCP Protocol

- DHCP enables auto-configuration of IP address to host.
- DHCP assigns dynamic IP addresses to devices on a network.
- Dynamic address allocation is required
  → when a host moves from one network to another or
  → when a host is connected to a network for the first time.

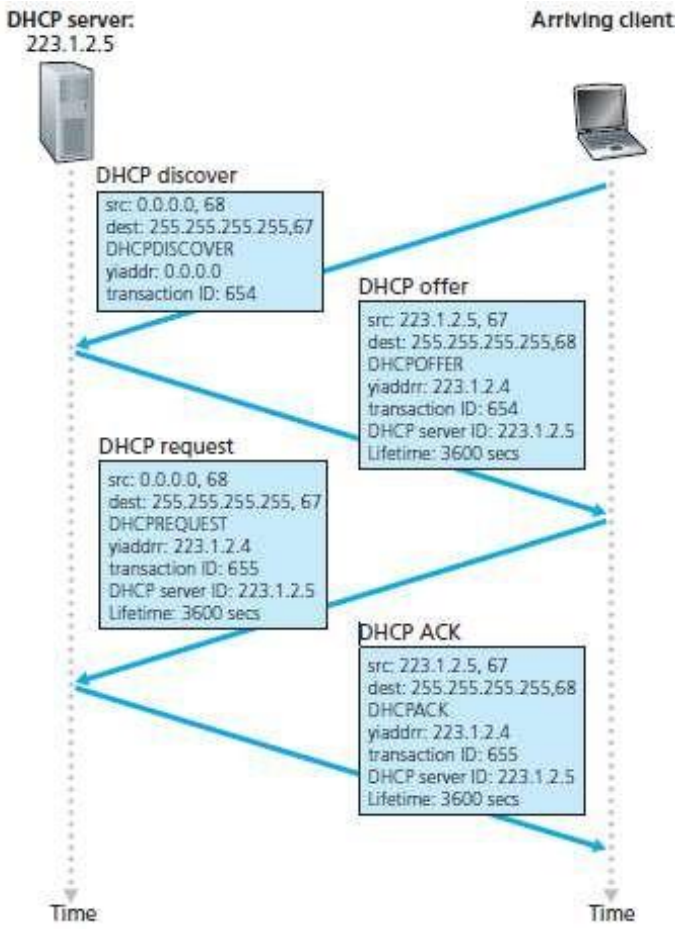


Figure 3.16: DHCP client-server interaction

- Four steps in DHCP protocol (Figure 3.16):

1) DHCP Server Discovery

➢ DHCP server contains a range of unassigned addresses to be assigned to hosts on-demand.

➢ To contact DHCP server, a client broadcasts a DHCPDISCOVER message with destination IP address 255.255.255.255.

2) DHCP Server Offer

➢ DHCP server broadcasts DHCPOFFER message containing

→ client's IP address

→ network mask and

→ IP address lease time (i.e. the amount of time for which the IP address will be valid).

3) DHCP Request

➢ The client sends a DHCPREQUEST message, requesting the offered address.

4) DHCP ACK

➢ The DHCP server acknowledges with a DHCPACK message containing the requested configuration.

3.4.3.5 NAT

- Network Address Translation (NAT) enables hosts to use Internet without the need to have globally unique addresses.
- NAT enables organization to have a large set of addresses internally and one address externally.
- The organization must have single connection to the Internet through a NAT-enabled router.
- NAT allows a single device (such as a router) to act as an agent b/w

1) Internet (or "public network") and

2) Local (or "private") network.

- This means only a single, unique IP address is required to represent an entire group of computers.
- Figure 3.17 shows the operation of a NAT-enabled router.

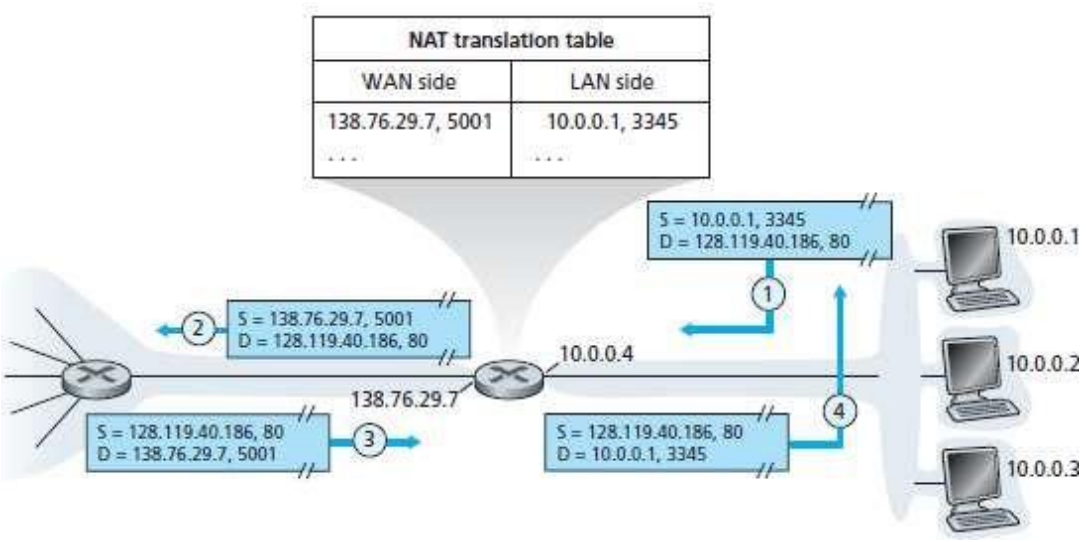


Figure 3.17: Network address translation

- The private addresses only have meaning to devices within a given network.
- The NAT-enabled router does not look like a router to the outside world.
- Instead, the NAT-enabled router behaves to the outside world as a single device with a single IP address.
- In Figure 3.17,
  1) All traffic leaving the home-router for the Internet has a source-address of 138.76.29.7.
  2) All traffic entering the home-router must have a destination-address of 138.76.29.7.
- The NAT-enabled router is hiding the details of the home-network from the outside world.
- At the NAT router, NAT translation-table includes
  1) Port numbers and
  2) IP addresses.
- IETF community is against the use of NAT. This is because of following reasons:
  1) They argue, port numbers are to be used for addressing processes, not for addressing hosts.
  2) They argue routers are supposed to process packets only up to layer 3.
  3) They argue the NAT protocol violates the end-to-end argument.
  4) They argue, we should use IPv6 to solve the shortage of IP addresses.
  5) NAT interferes with P2P applications. If Peer B is behind NAT, Peer B cannot act as a server.

3.4.4 ICMP

- ICMP is a network-layer protocol. (ICMP : Internet Control Message Protocol).
- This is used to handle error and other control messages.
- Main responsibility of ICMP: To report errors that occurs during the processing of the datagram.
- ICMP does not correct errors; ICMP simply reports the errors to the source.
- 12 types of ICMP messages are defined as shown in Table 3.4.
- Each ICMP message type is encapsulated in an IP packet.

ICMP Type	Code	Description
0	0	echo reply (to ping)
3	0	destination network unreachable
3	1	destination host unreachable
3	2	destination protocol unreachable
3	3	destination port unreachable
3	6	destination network unknown
3	7	destination host unknown
4	0	source quench (congestion control)
8	0	echo request
9	0	router advertisement
10	0	router discovery
11	0	TTL expired
12	0	IP header bad

Table 3.4: ICMP message types

1) Destination Unreachable (Type=3)

- This message is related to problem reaching the destinations.
- This message uses different codes (0 to 15) to define type of error-message.
- Possible values for code field:

  Code 0 = network unreachable Code 1 = host unreachable
  Code 2 = protocol unreachable Code 3 = port unreachable

2) Source Quench (Type=4)

- The main purpose is to perform congestion control.
- This message
  → informs the sender that network has encountered congestion & datagram has been dropped.
  → informs the sender to reduce its transmission-rate.

3) Echo Request & Echo Reply (Type=8 & Type=0)

- These messages are used to determine whether a remote-host is alive.
- A source sends an echo request-message to destination;
  If the destination is alive, the destination responds with an echo reply message.
- Type=8 is used for echo request;
  Type=0 is used for echo reply.
- These messages can be used in two debugging tools: ping and traceroute.

i) Ping

- The ping program can be used to find if a host is alive and responding.
- The source-host sends ICMP echo-request-messages.
- The destination, if alive, responds with ICMP echo-reply messages.

ii) Traceroute

- The traceroute program can be used to trace the path of a packet from source to destination.
- It can find the IP addresses of all the routers that are visited along the path.
- The program is usually set to check for the maximum of 30 hops (routers) to be visited.

3.4.5 IPv6

- CIDR, subnetting and NAT could not solve address-space exhaustion faced by IPv4.
- IPv6 was evolved to solve this problem.

3.4.5.1 Changes from IPv4 to IPv6 (Advantages of IPv6)

1) Expanded Addressing Capabilities

- IPv6 increases the size of the IP address from 32 to 128 bits (Supports upto 3.4×10^{38} nodes).
- In addition to unicast & multicast addresses, IPv6 has an anycast address.
- Anycast address allows a datagram to be delivered to only one member of the group.

2) A Streamlined 40-byte Header

- A number of IPv4 fields have been dropped or made optional.
- The resulting 40-byte fixed-length header allows for faster processing of the IP datagram.
- A new encoding of options field allows for more flexible options processing.

3) Flow Labeling & Priority

- A flow can be defined as

  "Labeling of packets belonging to particular flows for which the sender requests special handling".
- For example:

  Audio and video transmission may be treated as a flow.

3.4.5.2 IPv6 Datagram Format

- The format of the IPv6 datagram is shown in Figure 3.18.

Figure 3.18: IPv6 datagram format

- The following fields are defined in IPv6:

1) Version

- This field specifies the IP version, i.e., 6.

2) Traffic Class

- This field is similar to the TOS field in IPv4.
- This field indicates the priority of the packet.

3) Flow Label

- This field is used to provide special handling for a particular flow of data.

4) Payload Length

- This field shows the length of the IPv6 payload.

5) Next Header

- This field is similar to the options field in IPv4 (Figure 3.19).
- This field identifies type of extension header that follows the basic header.

6) Hop Limit

- This field is similar to TTL field in IPv4.
- This field shows the maximum number of routers the packet can travel.
- The contents of this field are decremented by 1 by each router that forwards the datagram.
- If the hop limit count reaches 0, the datagram is discarded.

7) Source & Destination Addresses

- These fields show the addresses of the source & destination of the packet.

8) Data

- This field is the payload portion of the datagram.
- When the datagram reaches the destination, the payload will be

  → removed from the IP datagram and

→ passed on to the upper layer protocol (TCP or UDP).

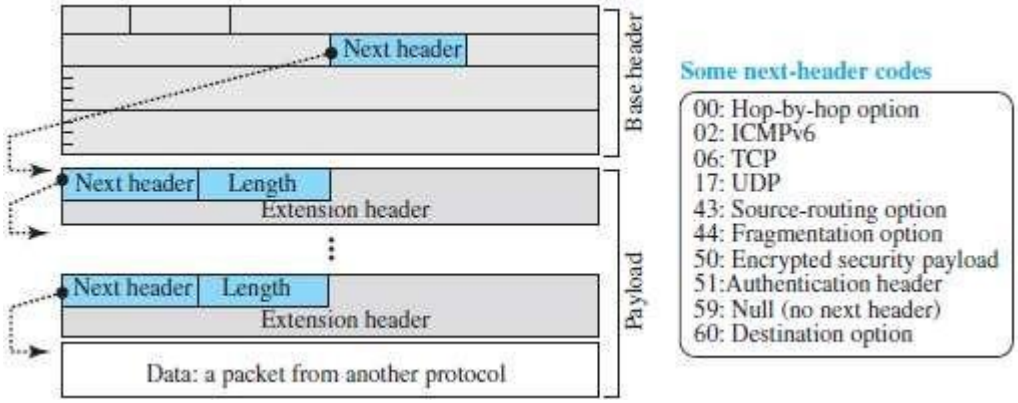


Figure 3.19: Payload in IPv6 datagram

3.4.5.3 IPv4 Fields not present in IPv6

1) Fragmentation/Reassembly

- Fragmentation of the packet is done only by the source, but not by the routers.The reassembling is done by the destination.
- Fragmentation & reassembly is a time-consuming operation.
- At routers, the fragmentation is not allowed to speed up the processing in the router.
- If packet-size is greater than the MTU of the network, the router
  → drops the packet.
  → sends an error message to inform the source.

2) Header Checksum

- In the Internet layers, the transport-layer and link-layer protocols perform check summing.
- This functionality was redundant in the network-layer.
- So, this functionality was removed to speed up the processing in the router.

3) Options

- In, IPv6, next-header field is similar to the options field in IPv4.
- This field identifies type of extension header that follows the basic header.
- To support extra functionalities, extension headers can be placed b/w base header and payload.

3.4.5.4 Difference between IPv4 & IPv6

	IPv4	IPv6
1	IPv4 addresses are 32 bit length	IPv6 addresses are 128 bit length
2	Fragmentation is done by sender and forwarding routers	Fragmentation is done only by sender
3	Does not identify packet flow for QoS handling	Contains Flow Label field that specifies packet flow for QoS handling
4	Includes Options up to 40 bytes	Extension headers used for optional data
5	Includes a checksum	Does not includes a checksum
6	Address Resolution Protocol (ARP) is available to map IPv4 addresses to MAC addresses	Address Resolution Protocol (ARP) is replaced with Neighbor Discovery Protocol (NDP)
7	Broadcast messages are available	Broadcast messages are not available
8	Manual configuration (Static) of IP addresses or DHCP (Dynamic configuration) is required to configure IP addresses	Auto-configuration of addresses is available

3.4.5.5 Transitioning from IPv4 to IPv6

- IPv4-capable systems are not capable of handling IPv6 datagrams.
- Two strategies have been devised for transition from IPv4 to IPv6:
  1) Dual stack and
  2) Tunneling.

3.4.5.5.1 Dual Stack Approach

- IPv6-capable nodes also have a complete IPv4 implementation. Such nodes are referred to as IPv6/IPv4 nodes.
- IPv6/IPv4 node has the ability to send and receive both IPv4 and IPv6 datagrams.
- When interoperating with an IPv4 node, an IPv6/IPv4 node can use IPv4 datagrams. When interoperating with an IPv6 node, an IPv6/IPv4 node can use IPv6 datagrams.
- IPv6/IPv4 nodes must have both IPv6 and IPv4 addresses.
- IPv6/IPv4 nodes must be able to determine whether another node is IPv6-capable or IPv4-only.
- This problem can be solved using the DNS.
  If the node name is resolved to IPv6-capable, then the DNS returns an IPv6 address
  Otherwise, the DNS return an IPv4 address.
- If either the sender or the receiver is only IPv4-capable, an IPv4 datagram must be used.
- Two IPv6-capable nodes can send IPv4 datagrams to each other.

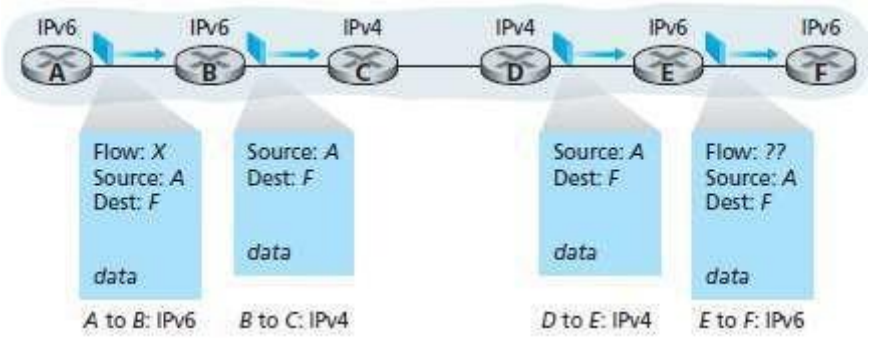


Figure 3.20: A dual-stack approach

- Dual stack is illustrated in Figure 3.20.
- Here is how it works:
  1) Suppose IPv6-capable Node-A wants to send a datagram to IPv6-capable Node-F.
  2) IPv6-capable Node-B creates an IPv4 datagram to send to IPv4-capable Node-C.
  3) At IPv6-capable Node-B, the IPv6 datagram is copied into the data field of the IPv4 datagramand appropriate address mapping can be done.
  4) At IPv6-capable Node-E, the IPv6 datagram is extracted from the data field of the IPv4 datagram.
  5) Finally, IPv6-capable Node-E forwards an IPv6 datagram to IPv6-capable Node-F.
- Disadvantage: During transition from IPv6 to IPv4, few IPv6-specific fields will be lost.

3.4.5.5.2 Tunneling

- Tunneling is illustrated in Figure 3.21.
- Suppose two IPv6-nodes B and E
  → want to interoperate using IPv6 datagrams and
  → are connected by intervening IPv4 routers.
- The intervening-set of IPv4 routers between two IPv6 routers are referred as a tunnel.
- Here is how it works:
  - On the sending side of the tunnel:
    → IPv6-node B takes & puts the IPv6 datagram in the data field of an IPv4 datagram.
    → The IPv4 datagram is addressed to the IPv6-node E.
  - On the receiving side of the tunnel: The IPv6-node E
    → receives the IPv4 datagram
    → extracts the IPv6 datagram from the data field of the IPv4 datagram and
    → routes the IPv6 datagram to IPv6-node F

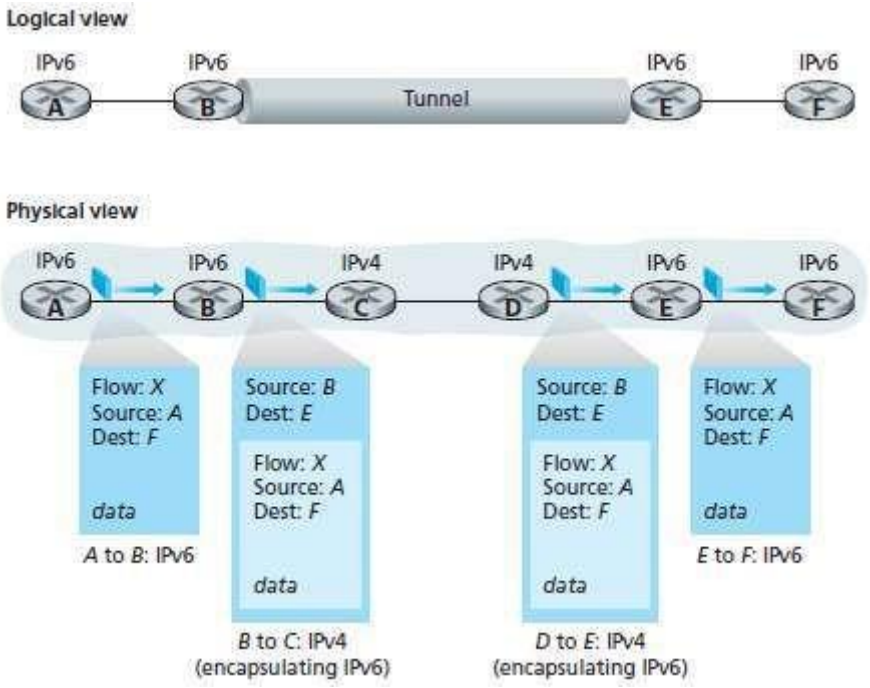


Figure 3.21: Tunneling

3.4.6 A Brief Foray into IP Security

- IPsec is a popular secure network-layer protocol.
- It is widely deployed in Virtual Private Networks (VPNs).
- It has been designed to be backward compatible with IPv4 and IPv6.
- It can be used to create a connection-oriented service between 2 entities.
- In transport mode, 2 hosts first establish an IPsec session between themselves.
- All TCP and UDP segments sent between the two hosts enjoy the security services provided by IPsec.
- On the source-side,
  1) The transport-layer passes a segment to IPsec.
  2) Then, IPsec
     → encrypts the segment
     → appends additional security fields to the segment and
     → encapsulates the resulting payload in a IP datagram.
  3) Finally, the sending-host sends the datagram into the Internet.
  - The Internet then transports the datagram to the destination-host.
- On the destination-side,
  1) The destination receives the datagram from the Internet.
  2) Then, IPsec
     → decrypts the segment and
     → passes the unencrypted segment to the transport-layer.
- Three services provided by an IPsec:

  1) Cryptographic Agreement
  - This mechanism allows 2 communicating hosts to agree on cryptographic algorithms & keys.

  2) Encryption of IP Datagram Payloads
  - When the sender receives a segment from the transport-layer, IPsec encrypts the payload.
  - The payload can only be decrypted by IPsec in the receiver.

  3) Data Integrity
  - IPsec allows the receiver to verify that the datagram's header fields.
  - The encrypted payload is not modified after transmission of the datagram into the n/w.

  4) Origin Authentication
  - The receiver is assured that the source-address in datagram is the actual source of datagram.

3.5 Routing Algorithms

- A routing-algorithm is used to find a "good" path from source to destination.
- Typically, a good path is one that has the least cost.
- The least-cost problem: Find a path between the source and destination that has least cost.

3.5.1 Routing Algorithm Classification

- A routing-algorithm can be classified as follows:
    1) Global or decentralized
    2) Static or dynamic
    3) Load-sensitive or Load-insensitive

3.5.1.1 Global or Decentralized

Global Routing Algorithm

- The calculation of the least-cost path is carried out at one centralized site.
- This algorithm has complete, global knowledge about the network.
- Algorithms with global state information are referred to as link-state (LS) algorithms.

Decentralized Routing Algorithm

- The calculation of the least-cost path is carried out in an iterative, distributed manner.
- No node has complete information about the costs of all network links.
- Each node has only the knowledge of the costs of its own directly attached links.
- Each node performs calculation by exchanging information with its neighboring nodes.

3.5.1.2 Static or Dynamic

Static Routing Algorithms

- Routes change very slowly over time, as a result of human intervention.
- For example: a human manually editing a router's forwarding-table.

Dynamic Routing Algorithms

- The routing paths change, as the network-topology or traffic-loads change.
- The algorithm can be run either
    → periodically or
    → in response to topology or link cost changes.
- Advantage: More responsive to network changes.
- Disadvantage: More susceptible to routing loop problem.

3.5.1.3 Load Sensitive or Load Insensitive

Load Sensitive Algorithm

- Link costs vary dynamically to reflect the current level of congestion in the underlying link.
- If high cost is associated with congested-link, the algorithm chooses routes around congested-link.

Load Insensitive Algorithm

- Link costs do not explicitly reflect the current level of congestion in the underlying link.
- Today's Internet routing-algorithms are load-insensitive. For example: RIP, OSPF, and BGP

3.5.2 LS Routing Algorithm

3.5.2.1 Dijkstra's Algorithm

- Dijkstra's algorithm computes the least-cost path from one node to all other nodes in the network.
- Let us define the following notation:
    1) u: source-node
    2) D(v): cost of the least-cost path from the source u to destination v.
    3) p(v): previous node (neighbor of v) along the current least-cost path from the source to v.
    4) N': subset of nodes; v is in N' if the least-cost path from the source to v is known.

Link-State (LS) Algorithm for Source Node u

```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v is a neighbor of u
5        then D(v) = c(u,v)
6      else D(v) = ∞
7
8  Loop
9    find w not in N' such that D(w) is a minimum
10   add w to N'
11   update D(v) for each neighbor v of w and not in N':
12         D(v) = min( D(v), D(w) + c(w,v) )
13   /* new cost to v is either old cost to v or known
14    least path cost to w plus cost from w to v */
15 until N'= N
```

- Example: Consider the network in Figure 3.22 and compute the least-cost paths from u to all possible destinations.

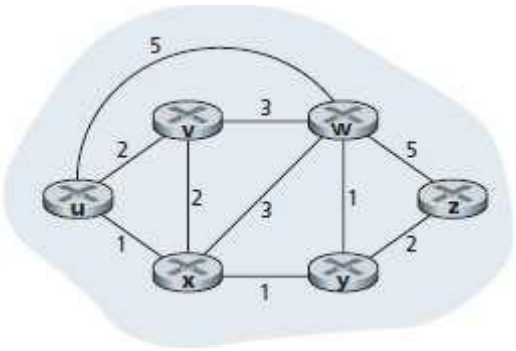


Figure 3.22: Abstract graph model of a computer network

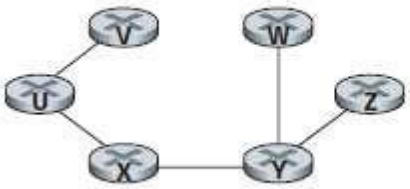
Solution:

- Let's consider the few first steps in detail.
  1) In the initialization step, the currently known least-cost paths from u to its directly attached neighbors, v, x, and w, are initialized to 2, 1, and 5, respectively.
  2) In the first iteration, we
     → look among those nodes not yet added to the set N' and
     → find that node with the least cost as of the end of the previous iteration.
  3) In the second iteration,
     → nodes v and y are found to have the least-cost paths (2) and
     → we break the tie arbitrarily and
     → add y to the set N' so that N' now contains u, x, and y.
  4) And so on. . . .
  5) When the LS algorithm terminates,
     We have, for each node, its predecessor along the least-cost path from the source.
- A tabular summary of the algorithm's computation is shown in Table 3.5.

step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

Table 3.5: Running the link-state algorithm on the network in Figure 3.20

- Figure 3.23 shows the resulting least-cost paths for u for the network in Figure 3.22.



Destination	Link
v	(u, v)
w	(u, x)
x	(u, x)
y	(u, x)
z	(u, x)

Figure 3.23: Least cost path and forwarding-table for node u

3.5.3 DV Routing Algorithm

3.5.3.1 Bellman Ford Algorithm

- Distance vector (DV) algorithm is 1) iterative, 2) asynchronous, and 3) distributed.
  1) It is distributed. This is because each node
     → receives some information from one or more of its directly attached neighbors
     → performs the calculation and
     → distributes then the results of the calculation back to the neighbors.
  2) It is iterative. This is because
     → the process continues on until no more info is exchanged b/w neighbors.
  3) It is asynchronous. This is because
     → the process does not require all of the nodes to operate in lockstep with each other.
- The basic idea is as follows:
  1) Let us define the following notation:
     $D_x(y)$ = cost of the least-cost path from node x to node y, for all nodes in N.
     $D_x = [D_x(y): y \text{ in } N]$ be node x's distance vector of cost estimates from x to all other nodes y in N.
  2) Each node x maintains the following routing information:
     i) For each neighbor v, the cost c(x,v) from node x to directly attached neighbor v
     ii) Node x's distance vector, that is, $D_x = [D_x(y): y \text{ in } N]$, containing x's estimate of its cost to all destinations y in N.
     iii) The distance vectors of each of its neighbors, that is, $D_v = [D_v(y): y \text{ in } N]$ for each neighbor v of x.
  3) From time to time, each node sends a copy of its distance vector to each of its neighbors.
  4) The least costs are computed by the Bellman-Ford equation:
     $D_x(y) = \min_v\{c(x,v) + D_v(y)\}$ for each node y in N
  5) If node x's distance vector has changed as a result of this update step, node x will then sendits updated distance vector to each of its neighbors.

Distance-Vector (DV) Algorithm

At each node, x:

```
1  Initialization:
2     for all destinations y in N:
3        D_x(y) = c(x,y)    /* if y is not a neighbor then c(x,y) = ∞ */
4     for each neighbor w
5        D_w(y) = ? for all destinations y in N
6     for each neighbor w
7        send distance vector D_x = [D_x(y): y in N] to w
8
9  loop
10    wait (until I see a link cost change to some neighbor w or
11          until I receive a distance vector from some neighbor w)
12
13    for each y in N:
14       D_x(y) = min_v{c(x,v) + D_v(y)}
15
16    if D_x(y) changed for any destination y
17       send distance vector D_x = [D_x(y): y in N] to all neighbors
18
19 forever
```

- Figure 3.24 illustrates the operation of the DV algorithm for the simple three node network.

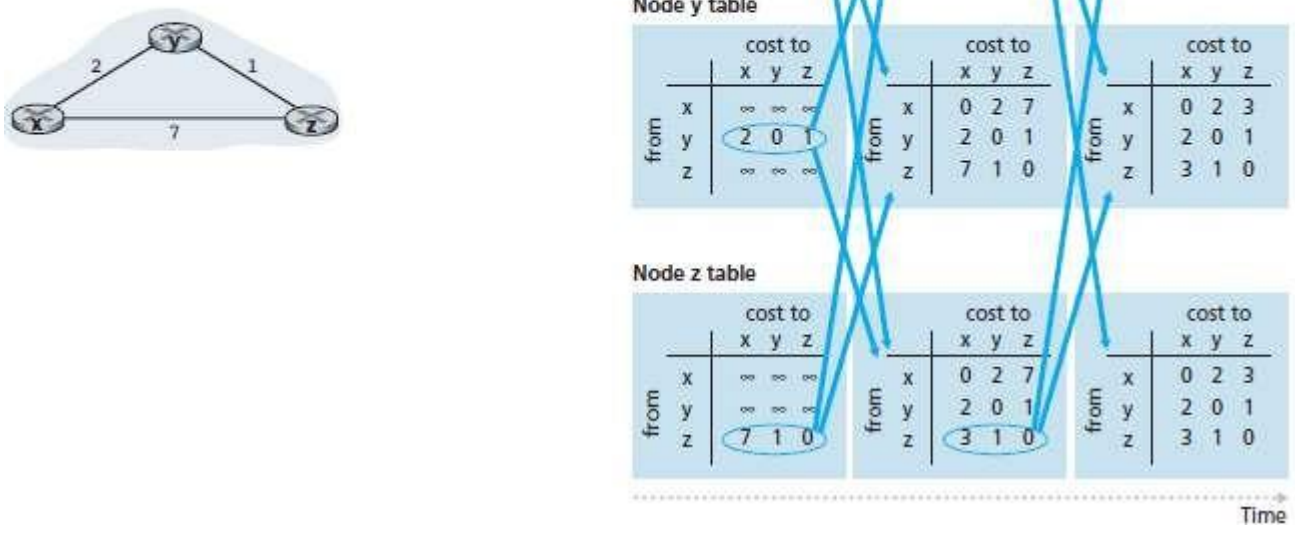


Figure 3.24: Distance-vector (DV) algorithm

- The operation of the algorithm is illustrated in a synchronous manner. Here, all nodes simultaneously
  - → receive distance vectors from their neighbours
  - → compute their new distance vectors, and
  - → inform their neighbours if their distance vectors have changed.

- The table in the upper-left corner is node x's initial routing-table.
- In this routing-table, each row is a distance vector.
- The first row in node x's routing-table is $D_x = [D_x(x), D_x(y), D_x(z)] = [0, 2, 7]$.
- After initialization, each node sends its distance vector to each of its two neighbours.
- This is illustrated in Figure 3.24 by the arrows from the first column of tables to the second column of tables.
- For example, node x sends its distance vector Dx = [0, 2, 7] to both nodes y and z. After receivingthe updates, each node recomputes its own distance vector.
- For example, node x computes

$D_x(x) = 0$
$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\} = \min\{2 + 0, 7 + 1\} = 2$
$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\} = \min\{2 + 1, 7 + 0\} = 3$

- The second column therefore displays, for each node, the node's new distance vector along with distance vectors just received from its neighbours.
- Note, that node x's estimate for the least cost to node z, $D_x(z)$, has changed from 7 to 3.
- The process of receiving updated distance vectors from neighbours, recomputing routing-table entries, and informing neighbours of changed costs of the least-cost path to a destination continues until no update messages are sent.
- The algorithm remains in the quiescent state until a link cost changes.

3.5.4 A Comparison of LS and DV Routing-algorithms

Distance Vector Protocol	Link State Protocol
Entire routing-table is sent as an update	Updates are incremental & entire routing-table is not sent as update
Distance vector protocol send periodic update at every 30 or 90 second	Updates are triggered not periodic
Updates are broadcasted	Updates are multicasted
Updates are sent to directly connected neighbour only	Update are sent to entire network &to just directly connected neighbour
Routers don't have end to end visibility of entire network.	Routers have visibility of entire network of that area only.
Prone to routing loops	No routing loops
Each node talks to only its directly connected neighbors	Each node talks with all other nodes (via broadcast)

3.5.5 Hierarchical Routing

- Two problems of a simple routing-algorithm:
  1) **Scalability**
     - As no. of routers increases, overhead involved in computing & storing routing info increases.
  2) **Administrative Autonomy**
     - An organization should be able to run and administer its network.
     - At the same time, the organization should be able to connect its network to internet.
  - Both of these 2 problems can be solved by organizing routers into autonomous-system (AS).
- An autonomous system (AS) is a group of routers under the authority of a single administration.For example: same ISP or same company network.
- Two types of routing-protocol:
  1) Intra-AS routing protocol: refers to routing inside an autonomous system.
  2) Inter-AS routing protocol: refers to routing between autonomous systems.

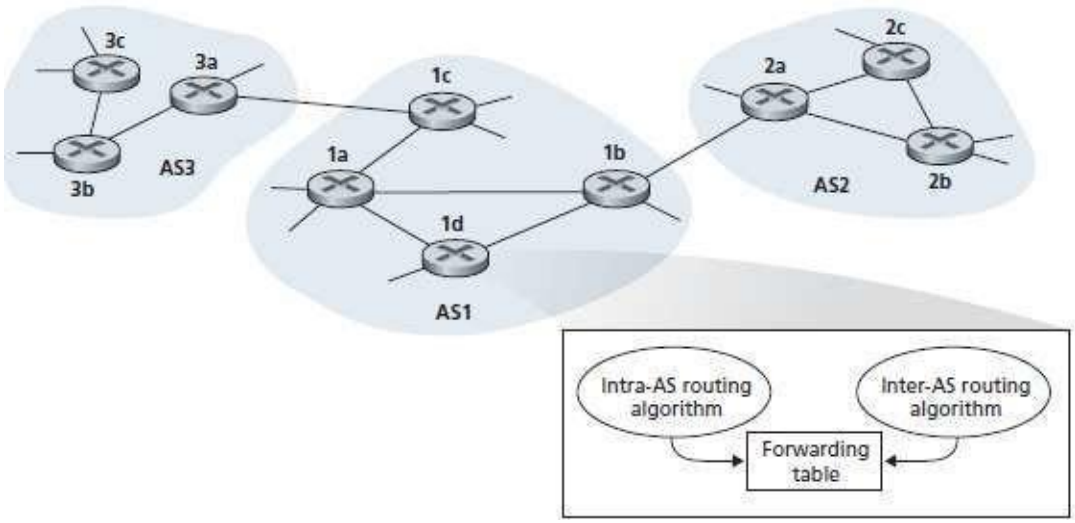


Figure 3.25: An example of interconnected autonomous-systems

3.5.5.1 Intra-AS Routing Protocol

- The routing-algorithm running within an autonomous-system is called intra-AS routing protocol.
- All routers within the same AS must run the same intra-AS routing protocol. For ex: RIP and OSPF
- Figure 3.25 provides a simple example with three ASs: AS1, AS2, and AS3.
- AS1 has four routers: 1a, 1b, 1c, and 1d. These four routers run the intra-AS routing protocol.
- Each router knows how to forward packets along the optimal path to any destination within AS1.

3.5.5.2 ~~Intra-AS~~ Routing Protocol

- The routing-algorithm running between 2 autonomous-systems is called inter-AS routing protocol.
- Gateway-routers are used to connect ASs to each other.
- Gateway-routers are responsible for forwarding packets to destinations outside the AS.
- Two main tasks of inter-AS routing protocol:
    1) Obtaining reachability information from neighboring Ass.
    2) Propagating the reachability information to all routers internal to the AS.
- The 2 communicating ASs must run the same inter-AS routing protocol. For ex: BGP.
- Figure 3.26 summarizes the steps in adding an outside-AS destination in a router's forwarding-table.

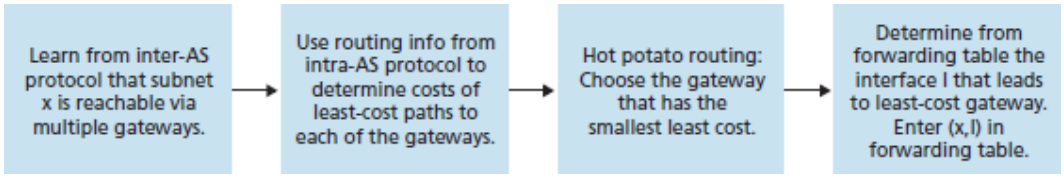


Figure 3.26: Steps in adding an outside-AS destination in a router's forwarding-table

3.6 Routing in the Internet

- Purpose of Routing protocols:
  To determine the path taken by a datagram between source and destination.
- An autonomous-system (AS) is a collection of routers under the same administrative control.
- In AS, all routers run the same routing protocol among themselves.

3.6.1 Intra-AS Routing in the Internet: RIP

- Intra-AS routing protocols are also known as interior gateway protocols.
- An intra-AS routing protocol is used to determine how routing is performed within an AS.
- Most common intra-AS routing protocols:
  1) Routing-information Protocol (RIP) and 2) Open Shortest Path First (OSPF)
- OSPF deployed in upper-tier ISPs whereas RIP is deployed in lower-tier ISPs & enterprise-networks.

3.6.1.1 RIP Protocol

- RIP is widely used for intra-AS routing in the Internet.
- RIP is a distance-vector protocol.
- RIP uses hop count as a cost metric. Each link has a cost of 1.
- Hop count refers to the no. of subnets traversed along the shortest path from source to destination.
- The maximum cost of a path is limited to 15.
- The distance vector is the current estimate of shortest path distances from router to subnets in AS.
- Consider an AS shown in Figure 3.27.

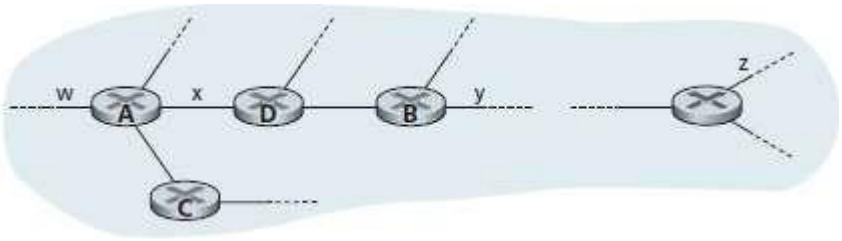


Figure 3.27: A portion of an autonomous-system

- Each router maintains a RIP table known as a routing-table.
- Figure 3.28 shows the routing-table for router D.

Destination Subnet	Next Router	Number of Hops to Destination
w	A	2
y	B	2
z	B	7
x	—	1

Figure 3.28: Routing-table in router D before receiving advertisement from router A

- Routers can send types of messages: 1) Response-message & 2) Request-message

  1) Response Message

  - Using this message, the routers exchange routing updates with their neighbors every 30 secs.
  - If a router doesn't hear from its neighbor every 180 secs, then that neighbor is not reachable.
  - When this happens, RIP
    → modifies the local routing-table and
    → propagates then this information by sending advertisements to its neighbors.
  - The response-message contains
    → list of up to 25 destination subnets within the AS and
    → sender's distance to each of those subnets.
  - Response-messages are also known as advertisements.

  2) Request Message

  - Using this message, router requests info about its neighbor's cost to a given destination.
- Both types of messages are sent over UDP using port# 520.
- The UDP segment is carried between routers in an IP datagram.

3.6.2 Intra-AS Routing in the Internet: OSPF

- OSPF is widely used for intra-AS routing in the Internet.
- OSPF is a link-state protocol that uses
  → flooding of link-state information and
  → Dijkstra least-cost path algorithm.
- Here is how it works:
  1) A router constructs a complete topological map (a graph) of the entire autonomous-system.
  2) Then, the router runs Dijkstra's algorithm to determine a shortest-path tree to all subnets.
  3) Finally, the router broadcasts link state info to all other routers in the autonomous-system.Specifically, the router broadcasts link state information
     → periodically at least once every 30 minutes and
     → whenever there is a change in a link's state. For ex: a change in up/down status.
- Individual link costs are configured by the network-administrator.
- OSPF advertisements are contained in OSPF messages that are carried directly by IP.
- HELLO message can be used to check whether the links are operational.
- The router can also obtain a neighboring router's database of network-wide link state.
- Some of the advanced features include:

1) Security

- Exchanges between OSPF routers can be authenticated.
- With authentication, only trusted routers can participate within an AS.
- By default, OSPF packets between routers are not authenticated.
- Two types of authentication can be configured: 1) Simple and 2) MD5.

i) Simple Authentication

¤ The same password is configured on each router.
¤ Clearly, simple authentication is not very secure.

ii) MD5 Authentication

¤ This is based on shared secret keys that are configured in all the routers.
¤ Here is how it works:
1) The sending router
   → computes a MD5 hash on the content of packet
   → includes the resulting hash value in the packet and
   → sends the packet
2) The receiving router
   → computes an MD5 hash of the packet
   → compares computed-hash value with the hash value carried in packet and
   → verifies the packet's authenticity

2) Multiple Same Cost Paths

- When multiple paths to a destination have same cost, OSPF allows multiple paths to be used.

3) Integrated Support for Unicast & Multicast Routing

- Multicast OSPF (MOSPF) provides simple extensions to OSPF to provide for multicast-routing.
- MOSPF
  → uses the existing OSPF link database and
  → adds a new type of link-state advertisement to the existing broadcast mechanism.

4) Support for Hierarchy within a Single Routing Domain

- An autonomous-system can be configured hierarchically into areas.
- In area, an area-border-router is responsible for routing packets outside the area.

- Exactly one OSPF area in the AS is configured to be the backbone-area.
- The primary role of the backbone-area is to route traffic between the other areas in the AS.

Inter-AS Routing: BGP

- BGP is widely used for inter-AS routing in the Internet.
- Using BGP, each AS can
  4) Obtain subnet reachability-information from neighboring ASs.
  5) Propagate the reachability-information to all routers internal to the AS.
  6) Determine good routes to subnets based on i) reachability-information and ii) AS policy.
- Using BGP, each subnet can advertise its existence to the rest of the Internet.

3.6.3.1 Basics

- Pairs of routers exchange routing-information over semi-permanent TCP connections using port-179.
- One TCP connection is used to connect 2 routers in 2 different autonomous-systems. Semipermanent TCP connection is used to connect among routers within an autonomous-system.
- Two routers at the end of each connection are called peers. The messages sent over the connection is called a session.
- Two types of session:
  1) External BGP (eBGP) session
     - This refers to a session that spans 2 autonomous-systems.
  2) Internal BGP (iBGP) session
     - This refers to a session between routers in the same AS.

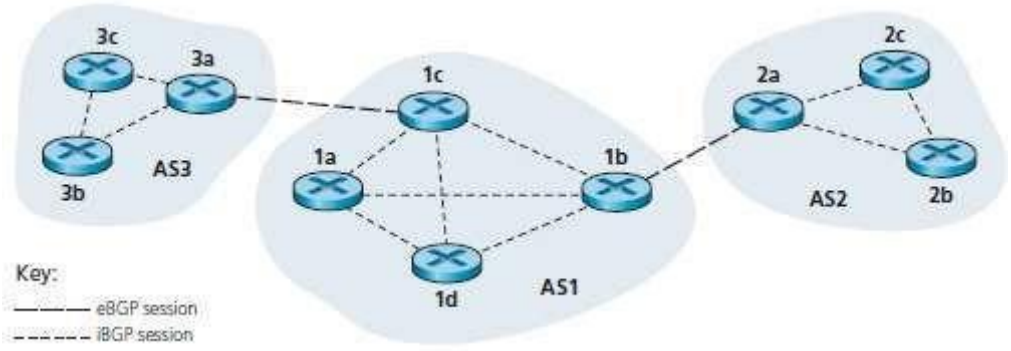


Figure 3.29: eBGP and iBGP sessions

- BGP operation is shown in Figure 3.29.
- The destinations are not hosts but instead are CIDRized prefixes.
- Each prefix represents a subnet or a collection of subnets.

3.6.3.2 Path Attributes & Routes

- An autonomous-system is identified by its globally unique ASN (Autonomous-System Number).
- A router advertises a prefix across a session.
- The router includes a number of attributes with the prefix.
- Two important attributes: 1) AS-PATH and 2) NEXT-HOP
  1) AS-PATH
  - This attribute contains the ASs through which the advertisement for the prefix has passed.
  - When a prefix is passed into an AS, the AS adds its ASN to the ASPATH attribute.

- Routers use the AS-PATH attribute to detect and prevent looping advertisements.
- Routers also use the AS-PATH attribute in choosing among multiple paths to the same prefix.

2) NEXT-HOP

- This attribute provides the critical link between the inter-AS and intra-AS routing protocols.
- This attribute is the router-interface that begins the AS-PATH.

- BGP also includes
  → attributes which allow routers to assign preference-metrics to the routes.
  → attributes which indicate how the prefix was inserted into BGP at the origin AS.
- When a gateway-router receives a route-advertisement, the gateway-router decides
  → whether to accept or filter the route and
  → whether to set certain attributes such as the router preference metrics.

3.6.3.3 Route Selection

- For 2 or more routes to the same prefix, the following elimination-rules are invoked sequentially:
  1) Routes are assigned a local preference value as one of their attributes.
  2) The local preference of a route
     → will be set by the router or
     → will be learned by another router in the same AS.
  3) From the remaining routes, the route with the shortest AS-PATH is selected.
  4) From the remaining routes, the route with the closest NEXT-HOP router is selected.
  5) If more than one route still remains, the router uses BGP identifiers to select the route.

3.6.3.4 Routing Policy

- Routing policy is illustrated as shown in Figure 3.30.
- Let A, B, C, W, X & Y = six interconnected autonomous-systems.W, X & Y = three stub-networks.
  A, B & C = three backbone provider networks.
- All traffic entering a stub-network must be destined for that network.
  All traffic leaving a stub-network must have originated in that network.
- Clearly, W and Y are stub-networks.
- X is a multihomed stub-network, since X is connected to the rest of the n/w via 2 different providers
- X itself must be the source/destination of all traffic leaving/entering X.
- X will function as a stub-network if X has no paths to other destinations except itself.
- There are currently no official standards that govern how backbone ISPs route among themselves.

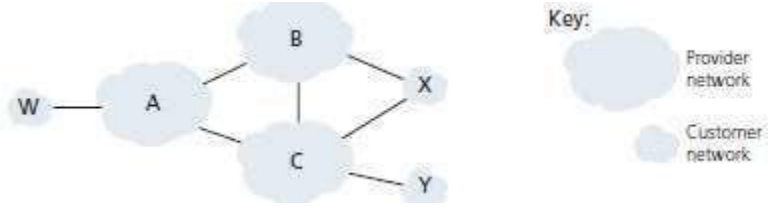


Figure 3.30: A simple BGP scenario

3.7 Broadcast & Multicast Routing

3.7.1 Broadcast Routing Algorithms

- Broadcast-routing means delivering a packet from a source-node to all other nodes in the network.

3.7.1.1 N-way Unicast

- Given N destination-nodes, the source-node
  → makes N copies of the packet and
  → transmits then the N copies to the N destinations using unicast routing (Figure 3.31).
- Disadvantages:
  1) Inefficiency
  - If source is connected to the n/w via single link, then N copies of packet will traverse this link.

  2) More Overhead & Complexity
  - An implicit assumption is that the sender knows broadcast recipients and their addresses.
  - Obtaining this information adds more overhead and additional complexity to a protocol.

  3) Not suitable for Unicast Routing
  - It is not good idea to depend on the unicast routing infrastructure to achieve broadcast.

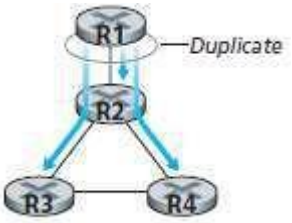


Figure 3.31: Duplicate creation/transmission

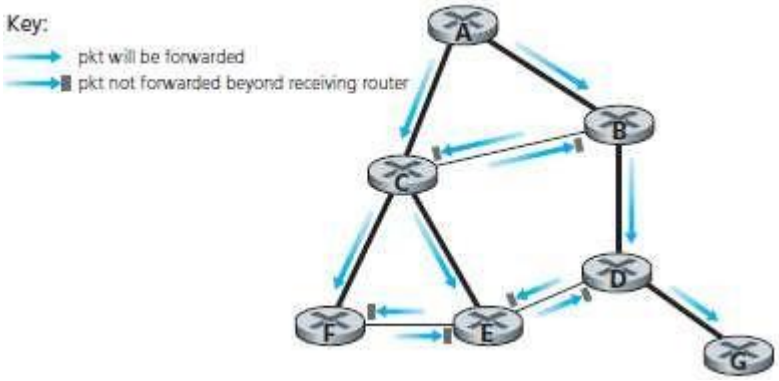


Figure 3.32: Reverse path forwarding

3.7.1.2 Uncontrolled Flooding

- The source-node sends a copy of the packet to all the neighbors.
- When a node receives a broadcast-packet, the node duplicates & forwards packet to all neighbors.
- In connected-graph, a copy of the broadcast-packet is delivered to all nodes in the graph.
- Disadvantages:
  1) If the graph has cycles, then copies of each broadcast-packet will cycle indefinitely.
  2) When a node is connected to 2 other nodes, the node creates & forwards multiple copies of packet
- Broadcast-storm refers to
  "The endless multiplication of broadcast-packets which will eventually make the network useless."

3.7.1.3 Controlled Flooding

- A node can avoid a broadcast-storm by judiciously choosing
  → when to flood a packet and when not to flood a packet.
- Two methods for controlled flooding:

1) Sequence Number Controlled Flooding

- A source-node
  → puts its address as well as a broadcast sequence-number into a broadcast-packet
  → sends then the packet to all neighbors.
- Each node maintains a list of the source-address & sequence# of each broadcast-packet.
- When a node receives a broadcast-packet, the node checks whether the packet is in this list.
- If so, the packet is dropped; if not, the packet is duplicated and forwarded to all neighbors.

2) Reverse Path Forwarding (RPF)

- If a packet arrived on the link that has a path back to the source;Then the router transmits the packet on all outgoing-links.
  Otherwise, the router discards the incoming-packet.
- Such a packet will be dropped. This is because
  → the router has already received a copy of this packet (Figure 3.32).

3.7.1.4 Spanning - Tree Broadcast

- This is another approach to providing broadcast. (MST Minimum Spanning Tree).
- Spanning-tree is a tree that contains each and every node in a graph.
- A spanning-tree whose cost is the minimum of all of the graph's spanning-trees is called a MST.
- Here is how it works (Figure 3.33):
  1) Firstly, the nodes construct a spanning-tree.
  2) The node sends broadcast-packet out on all incident links that belong to the spanning-tree.
  3) The receiving-node forwards the broadcast-packet to all neighbors in the spanning-tree.

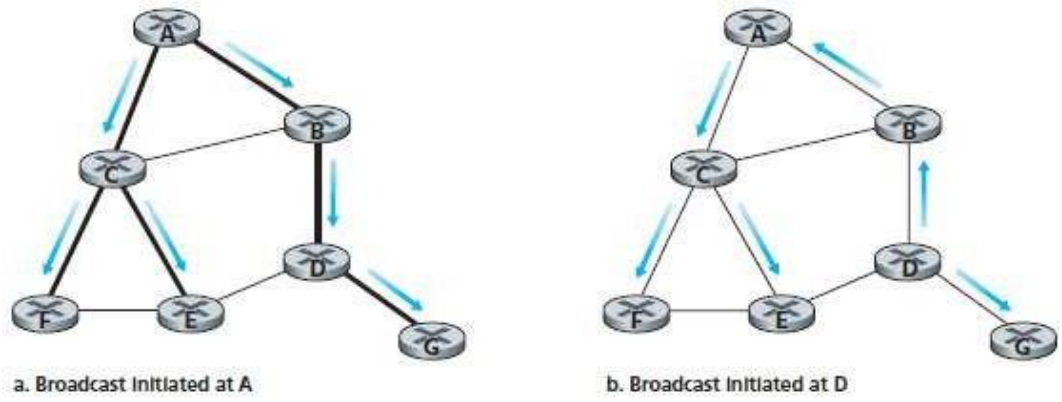


Figure 3.33: Broadcast along a spanning-tree

- Disadvantage:
  Complex: The main complexity is the creation and maintenance of the spanning-tree.

3.7.1.4.1 Center Based Approach

- This is a method used for building a spanning-tree.
- Here is how it works:
    1) A center-node (rendezvous point or a core) is defined.
    2) Then, the nodes send unicast tree-join messages to the center-node.
    3) Finally, a tree-join message is forwarded toward the center until the message either
        → arrives at a node that already belongs to the spanning-tree or
        → arrives at the center.
- Figure 3.34 illustrates the construction of a center-based spanning-tree.

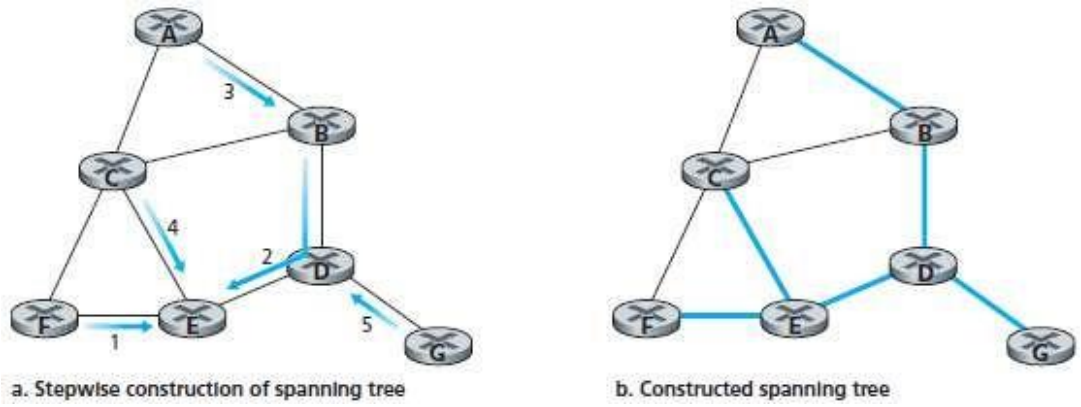


Figure 3.34: Center-based construction of a spanning-tree

3.7.2 Multicast

- Multicasting means a multicast-packet is delivered to only a subset of network-nodes.
- A number of emerging network applications requires multicasting. These applications include
    1) Bulk data transfer (for ex: the transfer of a software upgrade)
    2) Streaming continuous media (for ex: the transfer of the audio/video)
    3) Shared data applications (for ex: a teleconferencing application)
    4) Data feeds (for ex: stock quotes)
    5) Web cache updating and
    6) Interactive gaming (for ex: multiplayer games).
- Two problems in multicast communication:
    1) How to identify the receivers of a multicast-packet.
    2) How to address a packet sent to these receivers.
- A multicast-packet is addressed using address indirection.
- A single identifier is used for the group of receivers.
- Using this single identifier, a copy of the packet is delivered to all multicast receivers.
- In the Internet, class-D IP address is the single identifier used to represent a group of receivers.
- The multicast-group abstraction is illustrated in Figure 3.35.

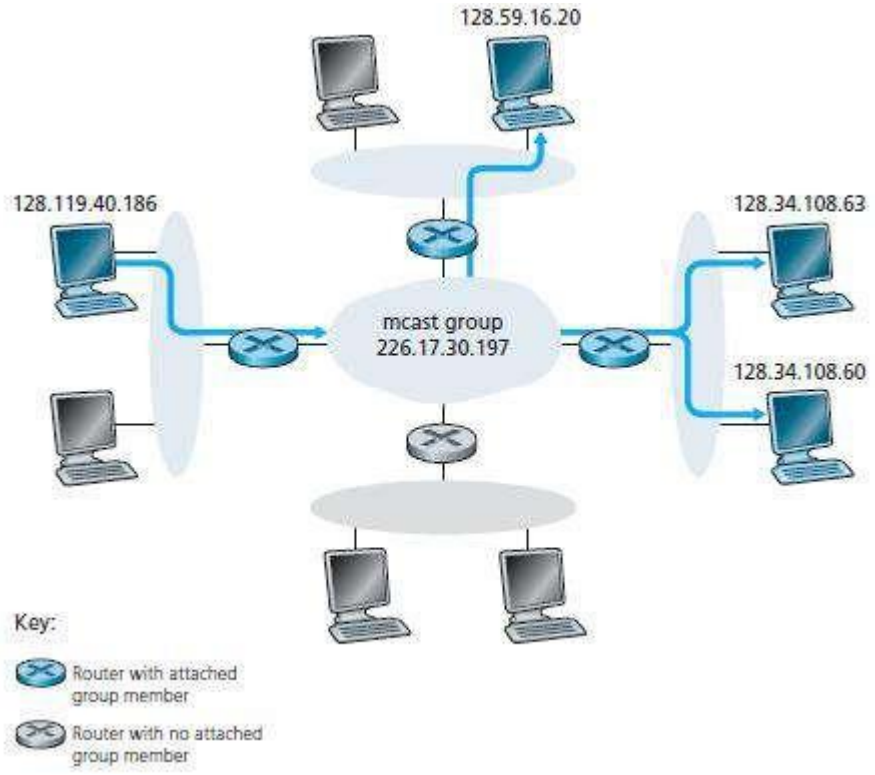


Figure 3.35: The multicast group: A datagram addressed to the group is delivered to all members ofthe multicast group.

3.7.2.1 IGMP

- In the Internet, the multicast consists of 2 components:
  1) **IGMP (Internet Group Management Protocol)**
     - IGMP is a protocol that manages group membership.
     - It provides multicast-routers info about the membership-status of hosts connected to the n/w
     - The operations are i) Joining/Leaving a group and ii) monitoring membership
  2) **Multicast Routing Protocols**
     - These protocols are used to coordinate the multicast-routers throughout the Internet.
     - A host places a multicast address in the destination address field to send packets to a set ofhosts belonging to a group.
- The IGMP protocol operates between a host and its attached-router.
- Figure 3.36 shows three first-hop multicast-routers.

Figure 3.36: The two components of network-layer multicast in the Internet: IGMP and multicast-routing protocols

- IGMP messages are encapsulated within an IP datagram.
- Three types of message: 1) membership query 2) membership report 3) leave group

1) Membership query
- A host sends a membership-query message to find active group-members in the network.

2) Membership report
- A host sends membership report message when an application first joins a multicast-group.
- The host sends this message w/o waiting for a membership query message from the router.

3) Leave group
- This message is optional.
- The host sends this message to leave the multicast-group.

- How does a router detect when a host leaves the multicast-group?

Answer: The router infers that a host is no longer in the multicast-group if it no longer responds to amembership query message. This is called soft state.

3.7.2.2 Multicast Routing Algorithms

- The multicast-routing problem is illustrated in Figure 3.37.
- Two methods used for building a multicast-routing tree:
  1) Single group-shared tree.
  2) Source-specific routing tree.

1) Multicast Routing using a Group Shared Tree

- A single group-shared tree is used to distribute the traffic for all senders in the group.
- This is based on

  Building a tree that includes all edge-routers & attached-hosts belonging to the multicast-group.
- In practice, a center-based approach is used to construct the multicast-routing tree.
- Edge-routers send join messages addressed to the center-node.
- Here is how it works:
  1) A center-node (rendezvous point or a core) is defined.
  2) Then, the edge-routers send unicast tree-join messages to the center-node.
  3) Finally, a tree-join message is forwarded toward the center until it either
     → arrives at a node that already belongs to the multicast tree or
     → arrives at the center.

2) Multicast Routing using a Source Based Tree

- A source-specific routing tree is constructed for each individual sender in the group.
- In practice, an RPF algorithm is used to construct a multicast forwarding tree.
- The solution to the problem of receiving unwanted multicast-packets under RPF is known as pruning.
- A multicast-router that has no attached-hosts will send a prune message to its upstream router.

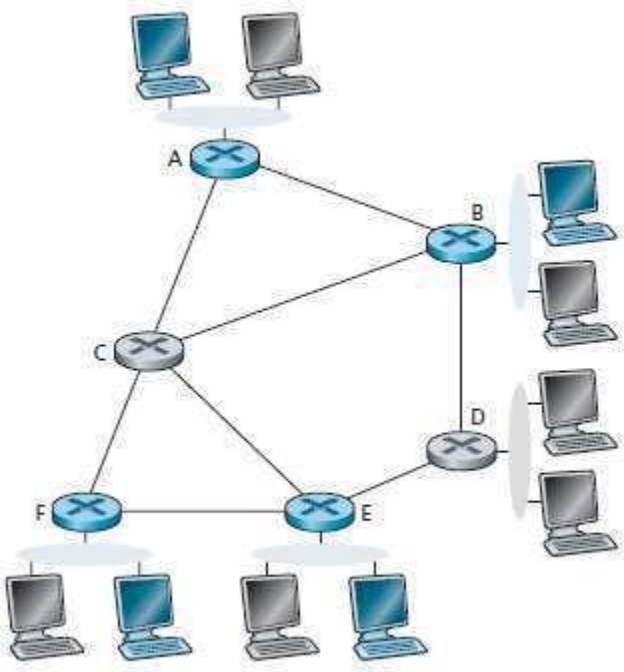


Figure 3.37: Multicast hosts, their attached routers, and other routers.

3.7.2.3 Multicast Routing in the Internet

- Three multicast routing protocols are:
  1) Distance Vector Multicast Routing Protocol (DVMRP)
  2) Protocol Independent Multicast (PIM) and
  3) Source Specific Multicast (SSM)

1) DVMRP

- DVMRP was the first multicast-routing protocol used in the Internet.
- DVMRP uses an RPF algorithm with pruning. (Reverse Path Forwarding).

2) PIM

- PIM is the most widely used multicast-routing protocol in the Internet.
- PIM divides multicast routing into sparse and dense mode.

i) Dense Mode

- Group-members are densely located.
- Most of the routers in the area need to be involved in routing the data.
- PIM dense mode is a flood-and-prune reverse path forwarding technique.

i) Sparse Mode

- The no. of routers with attached group-members is small with respect to total no. of routers.
- Group-members are widely dispersed.
- This uses rendezvous points to set up the multicast distribution tree.

3) SSM

- Only a single sender is allowed to send traffic into the multicast tree. This simplifies tree construction& maintenance.

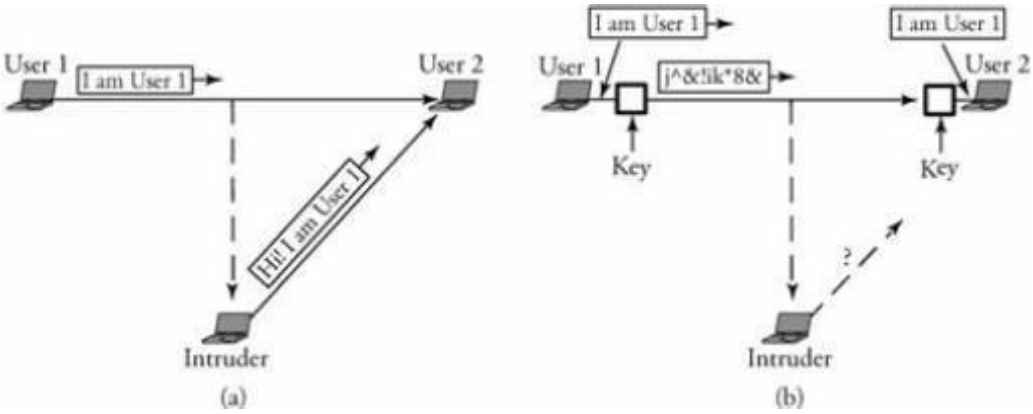
MODULE 4: Network Security

Overview of Network Security

- Network security is required by the users to communicate on the network.
- If medium is insecure then an intruder may intercept, read and modify the transmitted-data from sender to receiver.

Elements of Network Security

1) **Confidentiality:** Information should be available only to those who have rightful access to it
2) **Authenticity and integrity:** The sender of a message and the message itself should be verified at the receiving-point



(a) Message content and sender identity falsified by intruder; (b) a method of applied security

- In figure a, user 1 sends a message ("i am user 1") to user 2. Since the network lacks any security system, an intruder can receive the message and change its content to a different message ("hi i am user 1") and send it to user 2. User 2 may not know that this falsified message is really from user 1(authentication).
- In figure 10.1b, a security block is added to each side of the communication, and a secret key that only users 1 and 2 would know about is included. Therefore, the message is changed to a form that cannot be altered by the intruder.

Threats To Network Security

Internet infrastructure attacks are broadly classified into 4 categories

1) DNS hacking

2) Routing table poisoning
3) Packet mistreatment
4) Denial of Service (DOS)

DNS HACKING ATTACKS

- DNS server is a distributed hierarchical and global directory that translates domain names into numerical IP address.
- DNS is a critical infrastructure, and all hosts contact DNS to access servers and start connections.
- Name-resolution services in the modern Internet environment are essential for email transmission, navigation to web sites, or data transfer. Thus, an attack on DNS can potentially affect a large portion of the Internet.
- A DNS hacking attack can appear in any of the following forms
  1) **Masquerading Attack:** The attacker poses as a trusted entity and obtains all the secret information. The attacker can stop any message from being transmitted further or can change the content or redirect the packet to bogus servers. This action is also known as a middle-man attack.
  2) **Domain Highjacking Attack:** Whenever a user enters a domain address, he is forced to enter into the attacker's Web site.
  3) **Information Leakage Attack:** The attacker sends a query to all hosts identifies which IP addresses are not used and uses those IP address to make other types of attacks
  4) **Information-Level Attack(Cache Poisoning):** This forces a server to correspond with other than the correct answer. The hacker tricks a remote name-servers into caching the answer for a third-party domain by providing malicious information and redirects traffic to a preselected site.

ROUTING TABLE POISONING

- This is the undesired modification of routing tables. This results in a lower throughput of the network.
- Two types of attacks are: i) link attack and ii)router attack.

Link Attack

- Link attack occurs when a hacker gets access to a link and thereby intercepts, interrupts or modifies routing messages. This act similarly on both the link-state and the distance-vector protocols.
- If an attacker succeeds in placing an attack in a link-state routing protocol, a router may send incorrect updates about its neighbors or remain silent even if the link state of its neighbor has changed

Router Attack

- Router Attack may affect the link-state protocol or even the distance-vector protocol.
- In link-state protocol, if routers are attacked, they become malicious. As a result, routers may add a non existing link to a routing table delete an existing link or change the cost of a link.
- In the distance-vector protocol, an attacker may cause routers to send wrong updates about any node in the network, thereby misleading a router and resulting in network problems.

PACKET MISTREATMENT ATTACKS

- Packet mistreatment attacks can occur during any data transmission.
- A hacker may capture certain data packets and mistreat them.
- The attack may result in congestion lowering throughput & DOS attacks
- Link-attack causes interruption, modification or replication of data packets. Whereas, a router-attack can misroute all packets and may result in congestion or DOS

Following are some examples:

1) **Interruption:** If an attacker intercepts packets, they may not be allowed to be propagated to their destinations.
2) **Modification:** Attackers may succeed in accessing the content of a packet. They can then change the address of the packet or change the data of the packet. This kind of attack can be detected by digital signature mechanism.
3) **Replication**: An attacker may trap a packet and duplicate it. This kind of attack can be detected by using the sequence number for each packet.
4) **Malicious Misrouting of Packets**: A hacker may attack a router and change its routing table, resulting in misrouting of data packets.

5) **Ping of death:** An attacker may send a ping message, which is large and therefore must be fragmented for transport. The receiver then starts to reassemble the fragments as the ping fragments arrive. The total packet length becomes too large and might cause a system crash.

DOS ATTACKS (DENIAL OF SERVICE)

- DOS is a type of security breach that prohibits a user from accessing normally provided services.
- DOS can cost the target person a large amount of time and money.
- DOS affects the destination rather than a data-packet or router.
- They take important servers out of action for few hours, thereby denying service to allusers.

Two types of attacks are:

1) *Single-source:* An attacker sends a large number of packets to a target system to overwhelm & disable it
2) *Distributed:* A large number of hosts are used to flood unwanted traffic to a single target. The target cannot then be accessible to other users in the network.

Overview of Security Methods

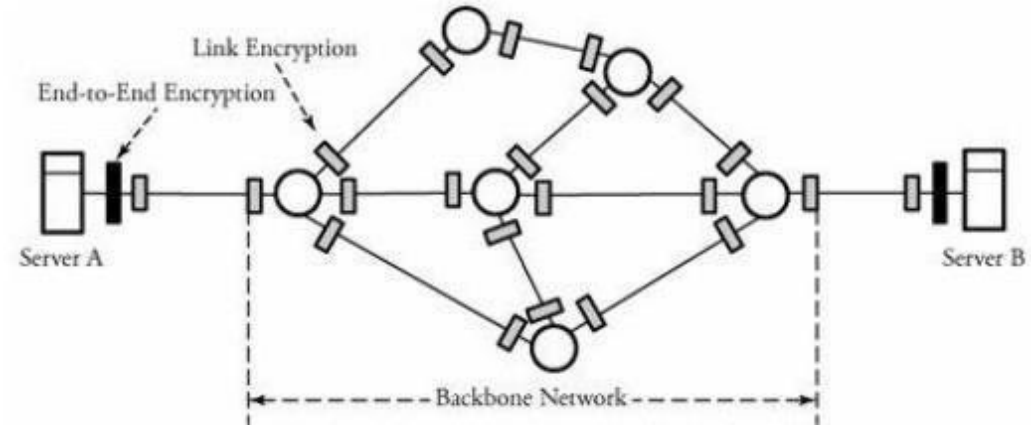
Common solutions that can protect computer communication networks from attacks are classified are cryptographic techniques or authentication techniques(verification).

Cryptographic Techniques

- Cryptography is the process of transforming a piece of information or message shared by two parties into some sort of code.
- The message is scrambled before transmission so that it is undetectable by outside watchers.
- The scrambled-message needs to be decoded at the receiving-end before any further processing.
- The main tool used to encrypt a message M is a secret-key K.
- The fundamental operation used to encrypt a message is the exclusive-OR($\oplus$).
- Assume that we have one-bit M and a secret-bit K. A simple encryption is carried out using $M \oplus K$.

- To decrypt this message, the second party can detect M by performing the following operation: (M ⊕ K) ⊕ K = M
- In *end-to-end encryption*, secret coding is carried out at both end systems. In *link encryption*, all the traffic passing over that link is secured.
- Two types of encryption techniques are secret-key & public-keyencryption
  1) In *secret-key model*, both sender & receiver conventionally use same key for an encryption process.
  2) In *public-key model*, a sender and a receiver each use a different key.
- The public-key system is more powerful than the secret key system & provides better security and message privacy.

  Drawbacks of public-key system: slow speed and more complex computationally



Authentication Techniques

Encryption methods offer the assurance of message confidentiality. A networking-system must be able to verify the authenticity of the message and the sender of the message. These forms of security techniques are known as authentication techniques.

Authentication techniques are categorized as

i) authentication with message digest

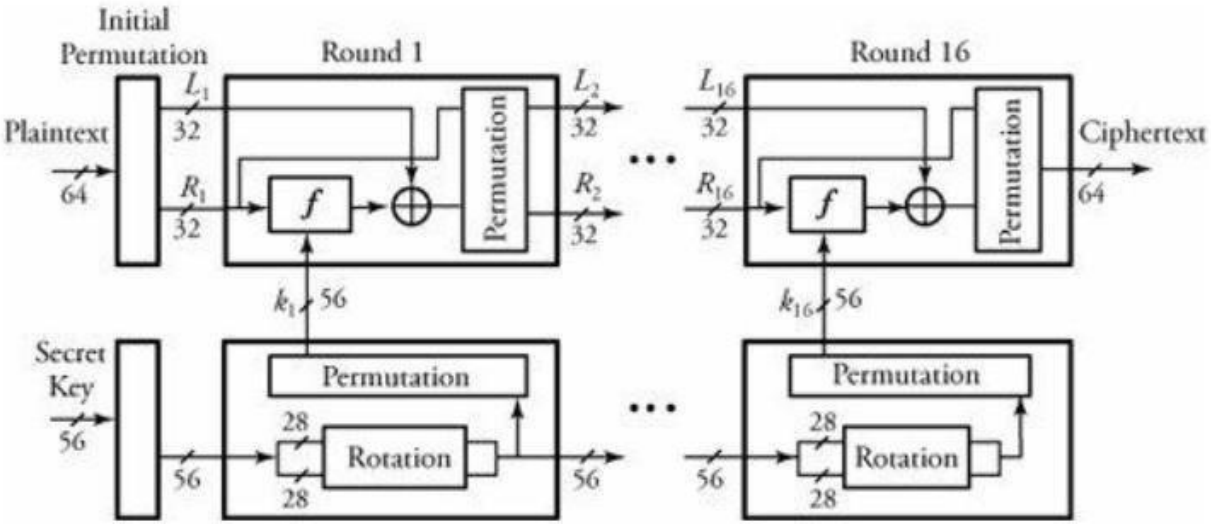
ii) authentication with digital signature.

Secret Key Encryption Protocols

- This is also called as symmetric encryption or single-key encryption.
- Sender and receiver conventionally use the same key for an encryption process.
- This consist of an encryption-algorithm, a key and a decryption-algorithm
- The encrypted-message is called **cipher text.**
- Two popular protocols are:
  1) DES (Data Encryption Standard)
  2) AES (Advanced Encryption Standard)
- A shared secret-key between a transmitter and a receiver is assigned at the transmitter and receiver points.
- At the receiving end, the encrypted information can be transformed back to the original data by using decryption algorithm and secret key.

DES (Data Encryption Standard)

- Plaintext messages are converted into 64-bit blocks & each block is encrypted using a key.
- The key length is 56 bits.
- DES consists of 16 identical rounds of an operation.



Begin DES Algorithm

1) Initialize. Before round 1 begins, all 64 bits of the message and all 56 bits of the secret key are separately permuted (shuffled).

2) Each incoming 64-bit message is broken into two 32-bit halves denoted by Li and Ri respectively.
3) The 56 bits of the key are also broken into two 28-halves, and each half is rotated one or two bit positions, depending on the round.
4) All 56 bits of the key are permuted, producing version ki of the key on round i.
5) Li and Ri are determined by

   $L_i = R_{i-1}$

   and

   $R_i = L_{i-1} \oplus F(R_{i-1}, k_i)$

6) All 64 bits of a message are permuted.

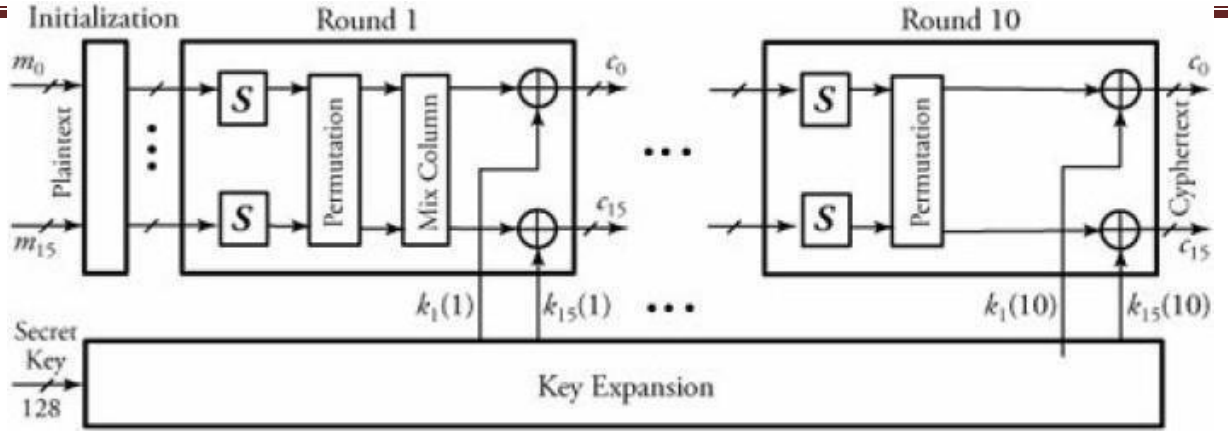
Operation of function F()

- Out of 56 bits of key k_i, function F( ) chooses 48 bits.
- The 32-bit R_{i-1} is expanded from 32 bits to 48 bits so that it can be combined with 48-bit k_i. The expansion of R_{i-1} is carried out by first breaking R_{i-1} into eight 4-bit chunks and then expanding each chunk by copying the leftmost bit and the rightmost bit from left and right adjacent chunks , respectively.
- F( ) also partitions the 48 bits of ki into eight 6-bit chunks.
- The corresponding eight chunks of R_{i-1} and eight chunks of kiare combined as follows

  $R_{i-1} = Ri\text{-}1 \oplus k_i$

AES (Advanced Encryption Standard)

- AES has better security strength than DES.
- In AES message is divided into 128-bit block, and it uses 128 or 192 or 256 bit key.
- Based on the key size number of rounds can be 10,12 or 14.
- The plaintext is formed as 16 bytes m0 through m15 and is fed into round 1 after an initialization stage.
- In this round, substitute-units(S) perform a byte-by-byte substitution of blocks.
- The ciphers move through a permutation-stage to shift rows to mix-columns.
- At the end of this round, all 16 blocks of ciphers are Exclusive-ORed with the 16 bytes of round 1 key k0(1) through k15(1).



Public Key Encryption Protocols

- This is also called as asymmetric or two key encryption.
- A sender/receiver pair use different keys.
- This is based on mathematical functions rather than on substitution or permutation.
- Two popular protocols are:
  i) RSA protocol
  ii) Diffie-Hillman key-exchange protocol.
- Either of the two related keys can be used for encryption; the other one for decryption.
- Each system publishes its encryption key by placing it in a public-register & sorts out key as public one. The companion key is kept private.
- If A wishes to send a message to B, A encrypts the message by using B's public key.
- At receiving end, B decrypts the message by using its private key.
- No other recipients can decrypt the message, since only B knows its private key.
- The public-key system is more powerful than the secret key system & provides better
- Drawbacks of public-key system:
  - slow speed
  - more complex computationally

RSA ALGORITHM

• Assume that a plaintext m must be encrypted to a cipher text c.

• This has three phases: key generation, encryption and decryption.

Key Generation Algorithm

1) Choose two prime numbers a and b and compute n=a.b
2) *Find x.* Select encryption-key x such that x and (a-1)(b-1) are relatively prime.
3) *Find y.* Calculate decryption-key y.

   xy mod (a-1)(b-1) = 1
4) At this point, a and b can be discarded.
5) The public key = {x, n}
6) The private key = {y, n}

Encryption

1) Both sender and receiver must know the value of n.
2) The sender knows the value of x and only the receiver knows the value of y.
3) Ciphertext c is constructed by

   $c=m^x \bmod n$

Decryption

Given the ciphertext c, the plaintext m is extracted by

$m=c^y \bmod n$.

DIFFIE-HILLMAN KEY-EXCHANGE PROTOCOL

• Two end users can agree on a shared secret-code without any information shared in advance.

• This protocol is normally used for VPN (virtual private network).

• Assume that user-1 wishes to communicate with user-2.

Key Generation Algorithm

1) User-1 selects a prime number 'a', random integer number 'x1', and a generator 'g'. Then creates 'y1' such that

   $y^1 = g^{x1} \bmod a$
2) User-2 performs the same function and creates y2 such that

   $y^2 = g^{x2} \bmod a$
3) User-1 then sends y1 to user-2. Now, user-1 forms its key k1 using the information its partner sent as

   $k_1 = y_2{}^{x1} \bmod a$
4) User-2 forms its key ka using the information its partner send it as

$k_2 = y_1^{x2} \bmod a$

5) The two keys k1 and k2 are equal. The two users can now encrypt their messages, each using its own key

AUTHENTICATION

- Message-authentication verifies the authenticity of both the message-sender and themessage-content.
- Message-sender is authenticated through implementation of a digital signature.
- Message-content is authenticated through implementation of a hash function and encryption of the resulting message-digest.
- Hash-function is used to produce a "fingerprint" of a message.
- The hash-value is added at the end of message before transmission.
- The receiver re-computes the hash-value from the received message and compares it to the received hash value.
- If the two hash-values are the same, the message was not altered during transmission.
- Once a hash-function is applied on a message m, the result is known as a message-digest h(m).
- The hash-function has the following properties

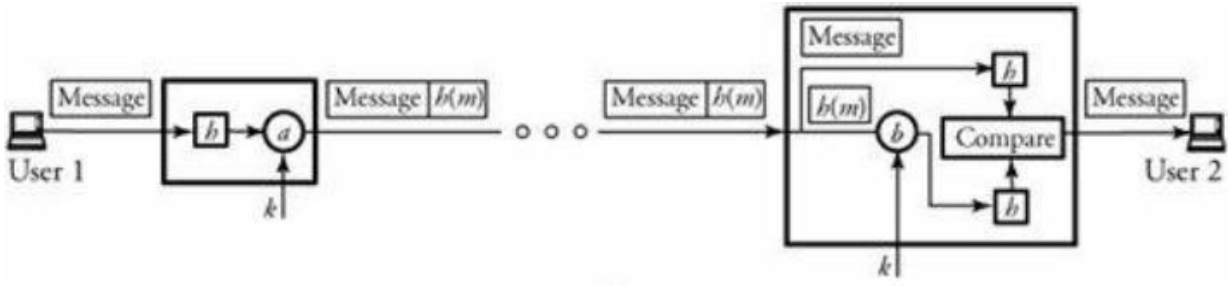
  1) Unlike the encryption-algorithm, the authentication algorithm is not required to be reversible.

  2) Given a message-digest h(m),it is computationally infeasible to find m.

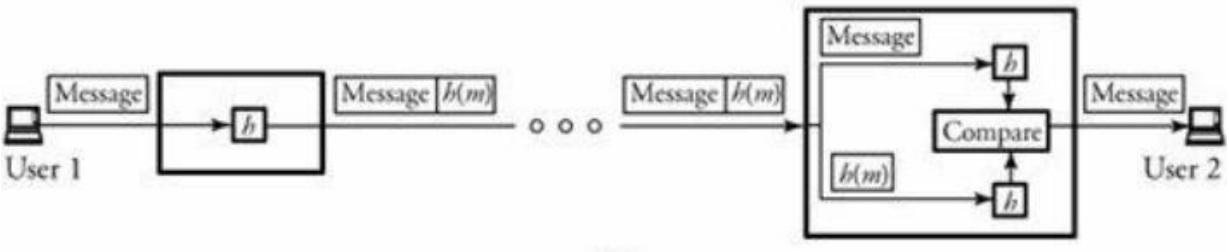
  3) This is computationally infeasible to find two different messages m1 and m2 such that h(m1)=h(m2).

• Message-authentication can be implemented by two methods.

1) In first method, a hash-function is applied on a message and then a process of encryption is implemented. At the receiver site, the received message-digest is decrypted and the comparison is made between the decrypted h(m) and the message-digest made locally from the received message. compare it with the one made locally at its site for any judgments on the integrity of the message.



2) In second method, no encryption is involved. The two parties share a secret key. Hence, at the receiving site, the comparison is made between the received h(m) and the message-digest made locally from the received message.



Secure Hash Algorithm (SHA)

- The Secure Hash Algorithm (SHA) was proposed as part of the digital signature standard. SHA-1, the first version of this standard, takes messages with a maximum length of 2^{24} and produces a 160-bit digest.
- With this algorithm, SHA-1 uses five registers, R_1 through R_5, to maintain a "state" of 20 bytes.
- The first step is to pad a message m with length l_m. The message length is forced to $l_m = 448 \bmod 512$. In other words, the length of the padded message becomes 64 bits less than the multiple of 512 bits.
- After padding, the second step is to expand each block of 512-bit (16 32 bits) words $\{m_0, m_1, ..., m_{15}\}$ to words of 80 32 bits using:

  $$w_i = m_i \text{ for } 0 \le i \le 15$$

  And

  $$w_i = w_{i-3} \oplus w_{i-8} \oplus w_{i-14} \oplus w_{i-16} \hookleftarrow 1 \text{ for } 16 \le i \le 79,$$

  where $\hookleftarrow$ means left rotation by j bits.
- Then, the 80 steps (i = 0, 1, 2, ..., 79) of the four rounds are described as follows

  $$\delta = (R_1 \hookleftarrow 5) + F_i(R_2, R_3, R_4) + R_5 + w_i + C_i$$

$$R_5 = R_4$$
$$R_4 = R_3$$
$$R_3 = R_2 \hookleftarrow 30$$
$$R_2 = R_1$$

Where C_i is a constant value specified by the standard for round i .

$$R_1 = \delta,$$
$$F_i(a,b,c) = \begin{cases} (a \cap b) \cup (\bar{a} \cap c) & 0 \le i \le 19 \\ a \oplus b \oplus c & 20 \le i \le 39 \\ (a \cap b) \cup (a \cap c) \cup (b \cap c) & 40 \le i \le 59 \\ a \oplus b \oplus c & 60 \le i \le 79 \end{cases}$$

The message digest is produced by concatenation of the values in R_1 through R_5.

Authentication and Digital Signature

- A digital signature on a message is required for the authentication and identification of the right sender.
- RSA algorithm can be used to implement digital signature.
- The message is encrypted with the sender's private key. Thus, the entire encrypted message serves as a digital signature.
- At the receiving end, the receiver can decrypt the message using the public key. This authenticates that the packet comes from the right user.

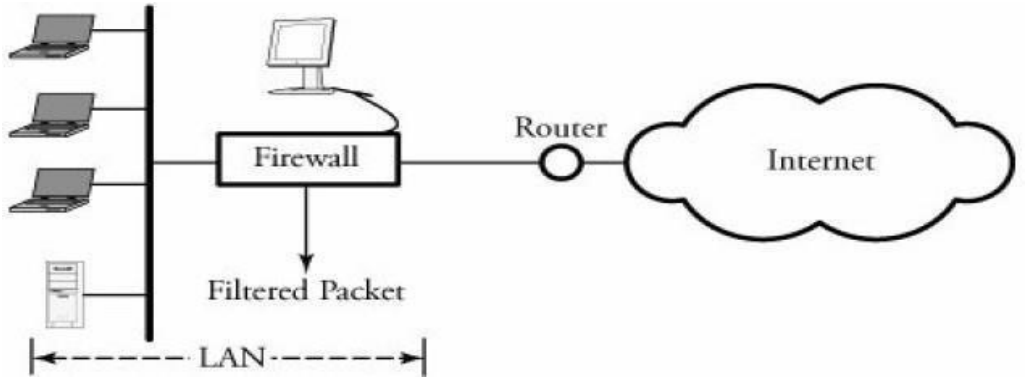
Firewalls

- Firewall is placed between hosts of a certain network and the outside world.
- Firewall is used to protect the network from unwanted web sites and potential hackers.
- The main objective is to monitor and filter packets coming from unknown sources.
- Firewall can also be used to control data traffic.
- Firewall can be a software program or a hardware device.

  1) Software firewalls can be installed in home computers by using an Internet connection with gateways.
- 2) Hardware firewalls are more secure than software firewalls are not expensive.

A firewall controls the flow of traffic by one of the following three methods:

1) Packet filtering: A firewall filters those packets that pass through. If packets can get through the filter, they reach their destinations: otherwise, they are discarded
2) A firewall filters packets based on the source IP address. This filtering is helpful when a host has to be protected from any unwanted external packets.



MODULE 5: MULTIMEDIA NETWORKING

Multimedia Networking Applications

➔ Properties of Video

- Most salient characteristic of video is its high bit rate.
  - Video distributed over the Internet typically ranges from 100 kbps for low-quality video conferencing to over 3 Mbps for streaming high-definition movies.
  - Video streaming consumes most bandwidth, having a bit rate of more than ten times greater than that of the normal HTTP and music-streaming applications.
- Video can be compressed.
  - A video is a sequence of images, typically being displayed at a constant rate, for example, at 24 or 30 images per second.
  - An uncompressed, digitally encoded image consists of an array of pixels, with each pixel encoded into a number of bits to represent luminance and color.
  - There are two types of redundancy in video, both of which can be exploited by **video compression**.
  - **Spatial redundancy** is the redundancy within a given image. Intuitively, an image that consists of mostly white space has a high degree of redundancy and can be efficiently compressed without significantly sacrificing image quality.
  - **Temporal redundancy** reflects repetition from image to subsequent image. If, for example, an image and the subsequent image are exactly the same, there is no reason to reencode the subsequent image; it is instead more efficient simply to indicate during encoding that the subsequent image is exactly the same.
  - We can also use compression to create multiple versions of the same video, each at a different quality level. For example, we can use compression to create, say, three versions of the same video, at rates of 300 kbps, 1 Mbps, and 3 Mbps.

➔ Properties of Audio

- Digital audio has significantly lower bandwidth requirements than video.

- Analog audio can be converted to a digital signal using pulse code modulation with the following steps:
  - The analog audio signal is **sampled** at some fixed rate.
  - Each of the samples is then rounded to one of a finite number of values. This operation is referred to as **quantization**. The number of such finite values called quantization values.
  - Each of the quantization values are **encoded** by representing with a fixed number ofbits.
- PCM-encoded speech and music, however, are rarely used in the Internet. Instead, as with video, compression techniques are used to reduce the bit rates of the stream.
- A popular compression technique for near CD-quality stereo music is MPEG 1 layer 3, more commonly known as MP3.
- MP3 encoders can compress to many different rates; 128 kbps is the most common encoding rate and produces very little sound degradation.
- As with video, multiple versions of a prerecorded audio stream can be created, each at a different bit rate.

➔ Types of Multimedia Network Applications

Multimedia applications are classified into three broad categories:

(i) Streaming stored audio/video

(ii) Conversational voice/video-over-IP

(iii) Streaming live audio/video

1) Streaming Stored Audio and Video

- In this class of applications, the underlying medium is prerecorded video, such as a movie, a television show, a prerecorded sporting event, or a prerecorded user generated video (such as those commonly seen on YouTube).
- These prerecorded videos are placed on servers, and users send requests to the servers to view the videos on demand.
- Many Internet companies today provide streaming video, including YouTube (Google), Netflix, and Hulu.
- By some estimates, streaming stored video makes up over 50 percent of the downstream traffic in the Internet access networks today.

Streaming stored video has three key distinguishing features.

- **Streaming:** In a streaming stored video application, the client typically begins video playout within a few seconds after it begins receiving the video from the server. This means that the client will be playing out from one location in the video while at the same time receiving later parts of the video from the server. This technique, known as streaming, avoids having to download the entire video file before playout begins.
- **Interactivity:** Because the media is prerecorded, the user may pause, reposition forward, reposition backward, fast-forward, and so on through the video content. The time from when the user makes such a request until the action manifests itself at the client should be less than a few seconds for acceptable responsiveness.
- **Continuous playout:** Once playout of the video begins, it should proceed according to the original timing of the recording. Therefore, data must be received from the server in time for its playout at the client; otherwise, users experience video frame freezing or frame skipping.

2) Conversational Voice- and Video-over-IP

- Real-time conversational voice over the Internet is often referred to as Internet telephony. It is also commonly called Voice-over-IP (VoIP).
- Conversational video is similar, except that it includes the video of the participants as well as their voices.
- Most of today's voice and video conversational systems allow users to create conferences with three or more participants.
- Conversational voice and video are widely used in the Internet today, with the Internet companies Skype, QQ, and Google Talk boasting hundreds of millions of daily users.
- Timing considerations and tolerance of data loss are important for conversational voice and video applications.
- Timing considerations are important because audio and video conversational applications are highly delay-sensitive. For a conversation with two or more interacting speakers, the delay from when a user speaks or moves until the action is manifested at the other end should be less than a few hundred milliseconds.

- On the other hand, conversational multimedia applications are loss-tolerant— occasional loss only causes occasional glitches in audio/video playback, and these losses can often be partially or fully concealed.

3) Streaming Live Audio and Video

- This third class of applications is similar to traditional broadcast radio and television, except that transmission takes place over the Internet.
- These applications allow a user to receive a live radio or television transmission—such as a live sporting event or an ongoing news event—transmitted from any corner of the world.
- Today, thousands of radio and television stations around the world are broadcasting content over the Internet.
- Live, broadcast-like applications often have many users who receive the same audio/video program at the same time.
- Although the distribution of live audio/video to many receivers can be efficiently accomplished using the IP multicasting techniques, multicast distribution is more often accomplished today via application-layer multicast (using P2P networks or CDNs) or through multiple separate unicast streams.
- As with streaming stored multimedia, the network must provide each live multimedia flow with an average throughput that is larger than the video consumption rate. Because the event is live, delay can also be an issue, although the timing constraints are much less stringent than those for conversational voice.

Streaming Stored Video

- For streaming video applications, prerecorded videos are placed on servers, and users send requests to these servers to view the videos on demand.
- The user may watch the video from beginning to end without interruption, may stop watching the video well before it ends, or interact with the video by pausing or repositioning to a future or past scene.
- Streaming video systems can be classified into three categories:
  1. UDP streaming

2. HTTP streaming
3. Adaptive HTTP streaming.

- A common characteristic of all three forms of video streaming is the extensive use of client-side application buffering to mitigate the effects of varying end-to-end delays and varying amounts of available bandwidth between server and client.
- When the video starts to arrive at the client, the client need not immediately begin playout, but can instead build up a reserve of video in an application buffer. Once the client has built up a reserve of several seconds of buffered-but-not-yet-played video, the client can then begin video playout.
- There are two important advantages provided by such client buffering. First, client side buffering can absorb variations in server-to-client delay. Second, if the server-to-client bandwidth briefly drops below the video consumption rate, a use can continue to enjoy continuous playback, again as long as the client application buffer does not become completely drained.

➔ UDP Streaming

- With UDP streaming, the server transmits video at a rate that matches the client's video consumption rate by clocking out the video chunks over UDP at a steady rate.
- For example, if the video consumption rate is 2 Mbps and each UDP packet carries 8,000 bits of video, then the server would transmit one UDP packet into its socket every (8000 bits)/(2 Mbps) = 4 msec.
- UDP does not employ a congestion-control mechanism, the server can push packets into the network at the consumption rate of the video without the rate-control restrictions of TCP.
- Before passing the video chunks to UDP, the server will encapsulate the video chunks within transport packets specially designed for transporting audio and video, using the Real-Time Transport Protocol (RTP).
- The client and server also maintain, in parallel, a separate control connection over which the client sends commands regarding session state changes (such as pause, resume, reposition,

and so on). The Real-Time Streaming Protocol is a popular open protocol for such a control connection.

Limitation:

- Due to the unpredictable and varying amount of available bandwidth between server and client, constant-rate UDP streaming can fail to provide continuous playout.
- It requires a media control server, such as an RTSP server, to process client-to-server interactivity requests and to track client state for each ongoing client session.
- Many firewalls are configured to block UDP traffic, preventing the users behind these firewalls from receiving UDP video.

➔ HTTP Streaming

- In HTTP streaming, the video is simply stored in an HTTP server as an ordinary file with a specific URL.
- When a user wants to see the video, the client establishes a TCP connection with the server and issues an HTTP GET request for that URL.
- The server then sends the video file, within an HTTP response message, as quickly as possible, that is, as quickly as TCP congestion control and flow control will allow.
- On the client side, the bytes are collected in a client application buffer. Once the number of bytes in this buffer exceeds a predetermined threshold, the client application begins playback—specifically, it periodically grabs video frames from the client application buffer, decompresses the frames, and displays them on the user's screen.

Advantages:

- The use of HTTP over TCP also allows the video to traverse firewalls and NATs more easily.
- Streaming over HTTP also obviates the need for a media control server, such as an RTSP server, reducing the cost of a large-scale deployment over the Internet.

Limitation and solution:

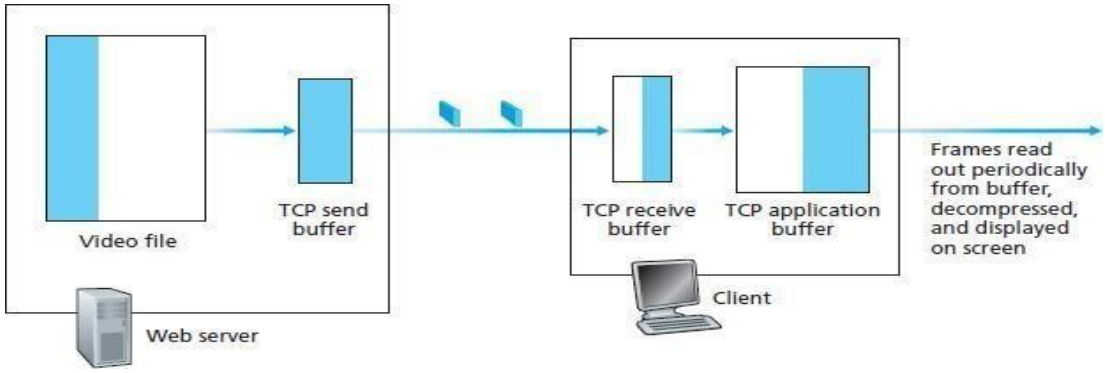
When transferring a file over TCP, the server-to client transmission rate can vary significantly due to TCP's congestion control mechanism. Packets can also be significantly delayed due to

TCP's retransmission mechanism. Because of these characteristics of TCP, it was believed that video streaming would never work well over TCP. Over time, however, designers of streaming video systems learned that TCP's congestion control and reliable-data transfer mechanisms do not necessarily preclude continuous playout when client buffering and prefetching are used.

Prefetching Video:

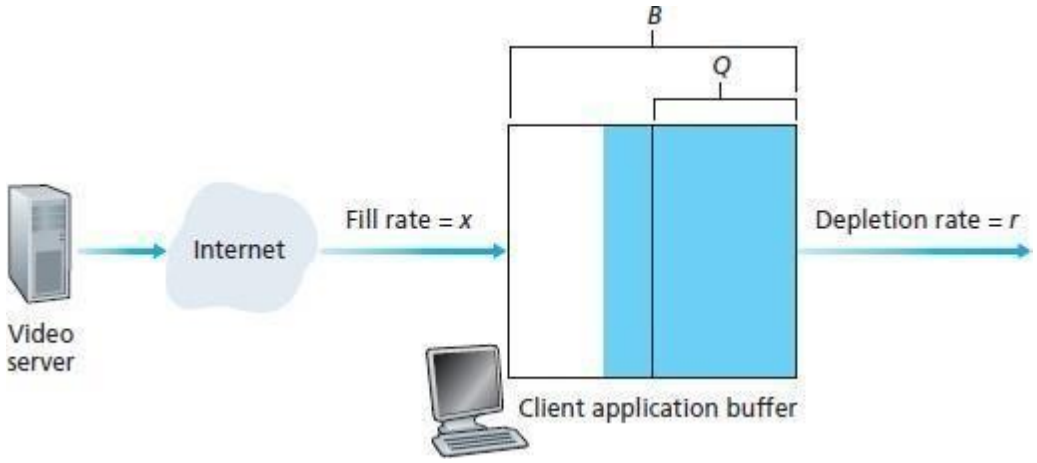
The client can attempt to download the video at a rate higher than the consumption rate, thereby prefetching video frames that are to be consumed in the future. This prefetched video is naturally stored in the client application buffer.

Client Application Buffer and TCP Buffers:



- Here TCP send buffer is shown to be full, the server is momentarily prevented from sending more bytes from the video file into the socket.
- On the client side, the client application reads bytes from the TCP receive buffer and places the bytes into the client application buffer.
- At the same time, the client application periodically grabs video frames from the client application buffer, decompresses the frames, and displays them on the user's screen.
- Consider now what happens when the user pauses the video during the streaming process. During the pause period, bits are not removed from the client application buffer, even though bits continue to enter the buffer from the server. If the client application buffer is finite, it may eventually become full, which will cause "back pressure" all the way back to the server.

Analysis of Video Streaming:



- Let B denote the size (in bits) of the client's application buffer, and let Q denote the number of bits that must be buffered before the client application begins playout.
- Let r denote the video consumption rate—the rate at which the client draws bits out of the client application buffer during playback.
- Let's assume that the server sends bits at a constant rate x whenever the client buffer is not full.
- If x < r (that is, if the server send rate is less than the video consumption rate), then the client buffer will never become full.
- When the available rate in the network is more than the video rate, after the initial buffering delay, the user will enjoy continuous playout until the video ends.

Early Termination and Repositioning the Video:

➔ HTTP streaming systems often make use of the **HTTP byte-range** header in the HTTP GET request message, which specifies the specific range of bytes the client currently wants to retrieve from the desired video.

➔ This is particularly useful when the user wants to reposition (that is, jump) to a future point in time in the video.

➔ When the user repositions to a new position, the client sends a new HTTP request, indicating with the byte-range header from which byte in the file should the server send data.

➔ When the server receives the new HTTP request, it can forget about any earlier request and instead send bytes beginning with the byte indicated in the byterange request.

➔ Adaptive Streaming and DASH

Shortcoming of HTTP Streaming:

All clients receive the same encoding of the video, despite the large variations in the amount of bandwidth available to a client, both across different clients and also over time for the same client.

Solution: **DASH**

- In DASH - Dynamic Adaptive Streaming over HTTP, the video is encoded into several different versions, with each version having a different bit rate and, correspondingly, a different quality level. The client dynamically requests chunks of video segments of a few seconds in length from the different versions.
- With DASH, each video version is stored in the HTTP server, each with a different URL.
- The HTTP server also has a manifest file, which provides a URL for each version along with its bit rate.
- The client first requests the manifest file and learns about the various versions.
- The client then selects one chunk at a time by specifying a URL and a byte range in an HTTP GET request message for each chunk.
- While downloading chunks, the client also measures the received bandwidth and runs a rate determination algorithm to select the chunk to request next.
- Naturally, if the client has a lot of video buffered and if the measured receive bandwidth is high, it will choose a chunk from a high-rate version. And naturally if the client has little video buffered and the measured received bandwidth is low, it will choose a chunk from a low-rate version.
- By dynamically monitoring the available bandwidth and client buffer level, and adjusting the transmission rate with version switching, DASH can often achieve continuous playout at the best possible quality level without frame freezing or skipping.

Content Distribution Networks

- Streaming stored video to locations all over the world while providing continuous playout and high interactivity is clearly a challenging task.

- For an Internet video company, the most straightforward approach to providing streaming video service is to build a single massive data center which stores all of its videos in the data center, and stream the videos directly from the data center to clients worldwide.
- But this approach faces some problems:
  - Single massive date center is single point of failure
  - It leads long path to distant clients
  - It may create network congestion.
  - Popular video will likely be sent many times over the same communication links. Not only does this waste network bandwidth, but the Internet video company itself will be paying its provider ISP (connected to the data center) for sending the same bytes into the Internet over and over again.
- In order to meet the challenge of distributing massive amounts of video data to users distributed around the world, almost all major video-streaming companies make use of Content Distribution Networks (CDNs).
- A CDN manages servers in multiple geographically distributed locations, stores copies of the videos (and other types of Web content, including documents, images, and audio) in its servers, and attempts to direct each user request to a CDN location that will provide the best user experience.
- The CDN may be a private CDN,that is, owned by the content provider itself; for example, Google's CDN distributes YouTube videos and other types of content.
- The CDN may alternatively be a third-party CDN that distributes content on behalf of multiple content providers; for example, Akamai's CDN is a third party CDN that distributes Netflix and Hulu content, among others.
- CDNs typically adopt one of two different server placement philosophies:
  - **Enter Deep:** One philosophy, pioneered by Akamai, is to enter deep into the access networks of Internet Service Providers, by deploying server clusters in access ISPs all over the world. The goal is to get close to end users, thereby improving user-perceived delay and throughput by decreasing the number of links and routers between the end user and the CDN cluster from which it receives content.
  - **Bring Home:** A second design philosophy, taken by Limelight and many other CDN companies, is to bring the ISPs home by building large clusters at a smaller number (for

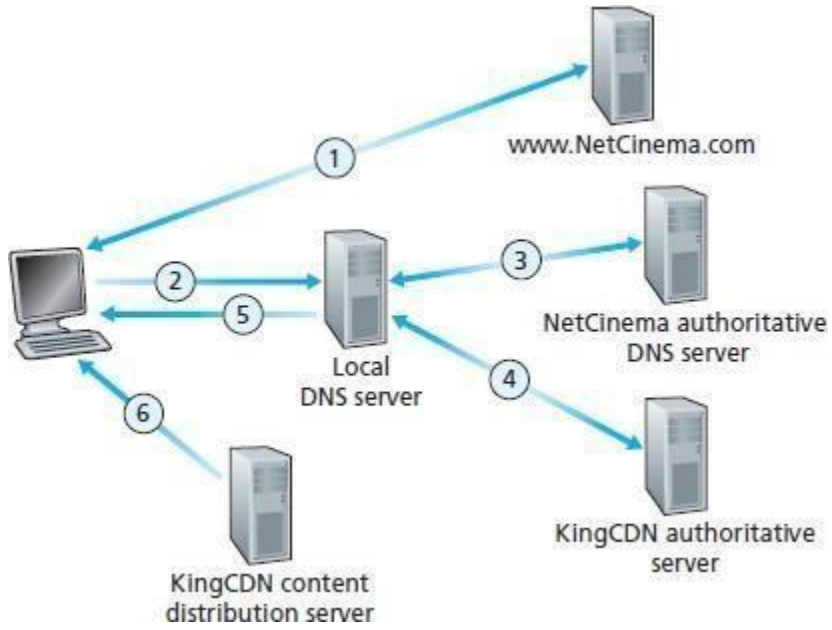
example, tens) of key locations and connecting these clusters using a private high-speed network. Instead of getting inside the access ISPs, these CDNs typically place each cluster at a location that is simultaneously near the PoPs of many tier-1 ISPs

➔ CDN Operation

When a browser in a user's host is instructed to retrieve a specific video (identified by a URL), the CDN must intercept the request so that it can

(1) Determine a suitable CDN server cluster for that client at that time.

(2) Redirect the client's request to a server in that cluster.



1. The user visits the Web page at NetCinema.
2. When the user clicks on the link http://video.netcinema.com/6Y7B23V, the user's host sends a DNS query for video.netcinema.com.
3. The user's Local DNS Server (LDNS) relays the DNS query to an authoritative DNS server for NetCinema, which observes the string "video" in the hostname video.netcinema.com. To "hand over" the DNS query to KingCDN, instead of returning an IP address, the NetCinema authoritative DNS server returns to the LDNS a hostname in the KingCDN's domain, for example, a1105.kingcdn.com.
4. From this point on, the DNS query enters into KingCDN's private DNS infrastructure. The user's LDNS then sends a second query, now for a1105.kingcdn.com, and KingCDN's DNS system eventually returns the IP addresses of a KingCDN content server to the LDNS. It is

thus here, within the KingCDN's DNS system, that the CDN server from which the client will receive its content is specified.

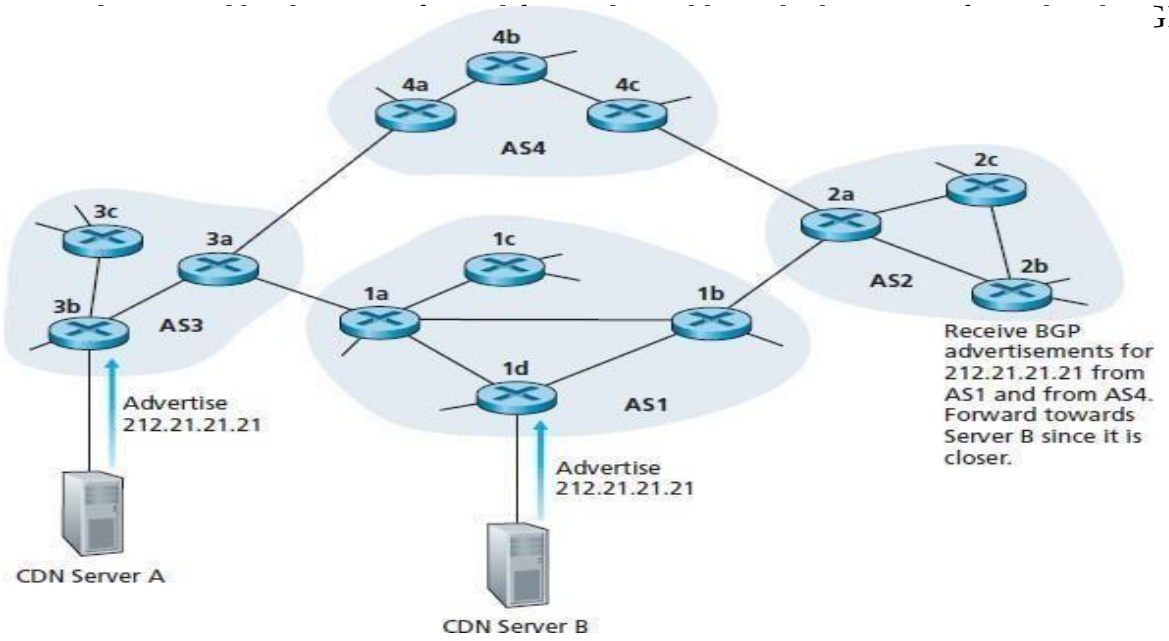
5. The LDNS forwards the IP address of the content-serving CDN node to the user's host.
6. Once the client receives the IP address for a KingCDN content server, it establishes a direct TCP connection with the server at that IP address and issues an HTTP GET request for the video. If DASH is used, the server will first send to the client a manifest file with a list of URLs, one for each version of the video, and the client will dynamically select chunks from the different versions.

➔ Cluster Selection Strategies

- Cluster Selection Strategies is a mechanism for dynamically directing clients to a server cluster or a data center within the CDN.
- The CDN learns the IP address of the client's LDNS server via the client's DNS lookup. After learning this IP address, the CDN needs to select an appropriate cluster based on this IP address.
- One simple strategy is to assign the client to the cluster that is geographically closest. Using commercial geo-location databases each LDNS IP address is mapped to a geographic location. When a DNS request is received from a particular LDNS, the CDN chooses the geographically closest cluster.
- For some clients, the solution may perform poorly, since the geographically closest cluster may not be the closest cluster along the network path.
- In order to determine the best cluster for a client based on the current traffic conditions, CDNs can instead perform periodic real-time measurements of delay and loss performance between their clusters and clients.
- An alternative to sending extraneous traffic for measuring path properties is to use the characteristics of recent and ongoing traffic between the clients and CDN servers.
- Such solutions, however, require redirecting clients to (possibly) suboptimal clusters from time to time in order to measure the properties of paths to these clusters.
- A very different approach to matching clients with CDN servers is to use IP anycast. The idea behind IP anycast is to have the routers in the Internet route the client's packets to the "closest" cluster, as determined by BGP.

- During the IP-anycast configuration stage, the CDN company assigns the same IP address to each of its clusters, and uses standard BGP to advertise this IP address from each of the different cluster locations.
- When a BGP router receives multiple route advertisements for this same IP address, it treats these advertisements as providing different paths to the same physical location.
- Following standard operating procedures, the BGP router will then pick the "best" route to the IP address according to its local route selection mechanism.
- After this initial configuration phase, the CDN can do its main job of distributing content. When any client wants to see any video, the CDN's DNS returns the anycast address, no matter where the client is located.
- When the client sends a packet to that IP address, the packet is routed to the "closest" cluster

GP.



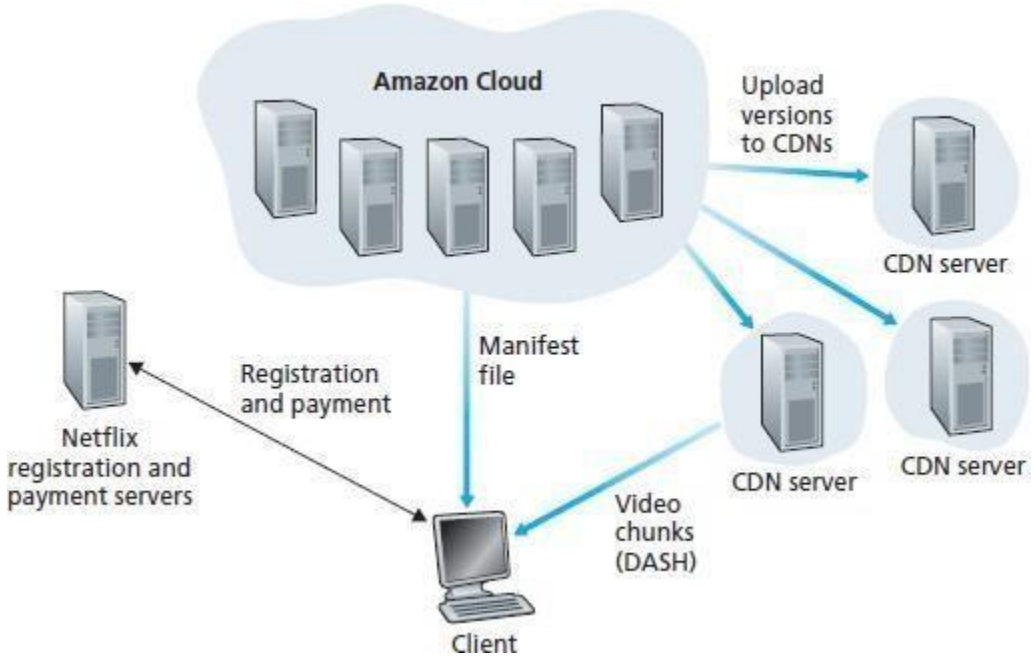
➔ Case Studies: Netflix, YouTube, and Kankan

Netflix

- Netflix is the leading service provider for online movies and TV shows in the United States.
- In order to rapidly deploy its large-scale service, Netflix has made extensive use of third-party cloud services and CDNs. Indeed, Netflix is an interesting example of a company

deploying a large-scale online service by renting servers, bandwidth, storage, and database services from third parties while using hardly any infrastructure of its own.

- Basic Architecture:



- Netflix has four major components: the registration and payment servers, the Amazon cloud, multiple CDN providers, and clients.
- In its own hardware infrastructure, Netflix maintains registration and payment servers, which handle registration of new accounts and capture credit-card payment information.
- Netflix runs its online service by employing machines (or virtual machines) in the Amazon cloud. Some of the functions taking place in the Amazon cloud include:
  - **Content ingestion:** Before Netflix can distribute a movie to its customers, it must first ingest and process the movie. Netflix receives studio master versions of movies and uploads them to hosts in the Amazon cloud.
  - **Content processing:** The machines in the Amazon cloud create many different formats for each movie, suitable for a diverse array of client video players running on desktop computers, smartphones, and game consoles connected to televisions. A different version is created for each of these formats and at multiple bit rates, allowing for adaptive streaming over HTTP using DASH.

- **Uploading versions to the CDNs:** Once all of the versions of a movie have been created, the hosts in the Amazon cloud upload the versions to the CDNs.

- The Web pages for browsing the Netflix video library are served from servers in the Amazon cloud.
- When the user selects a movie to "Play Now," the user's client obtains a manifest file, also from servers in the Amazon cloud. The manifest file includes a variety of information, including a ranked list of CDNs and the URLs for the different versions of the movie, which are used for DASH playback.
- The ranking of the CDNs is determined by Netflix, and may change from one streaming session to the next.
- Typically the client will select the CDN that is ranked highest in the manifest file.
- After the client selects a CDN, the CDN leverages DNS to redirect the client to a specific CDN server.
- The client and that CDN server then interact using DASH.

Youtube:

- With approximately half a billion videos in its library and half a billion video views per day, YouTube is indisputably the world's largest video-sharing site.
- YouTube began its service in April 2005 and was acquired by Google in November 2006.
- Google does not employ third-party CDNs but instead uses its own private CDN to distribute
- YouTube videos.
- Google has installed server clusters in many hundreds of different locations. From a subset of about 50 of these locations, Google distributes YouTube video.
- Google uses DNS to redirect a customer request to a specific cluster.
- Most of the time,
- Google's cluster selection strategy directs the client to the cluster for which the RTT between client and cluster is the lowest; however, in order to balance the load across clusters, sometimes the client is directed (via DNS) to a more distant cluster.
- If a cluster does not have the requested video, instead of fetching it from somewhere else and relaying it to the client, the cluster may return an HTTP redirect message, thereby redirecting the client to another cluster.

- YouTube employs HTTP streaming. YouTube often makes a small number of different versions available for a video, each with a different bit rate and corresponding quality level.
- YouTube processes each video it receives, converting it to a YouTube video format and creating multiple versions at different bit rates. This processing takes place entirely within Google data centers.

Kankan

- Kankan allows the service provider to significantly reduce its infrastructure and bandwidth costs.
- This approach uses P2P delivery instead of client-server (via CDNs) delivery. P2P video delivery is used with great success by several companies in China, including Kankan (owned and operated by Xunlei), PPTV (formerly PPLive), and PPs (formerly PPstream).
- Kankan, currently the leading P2P-based video-on-demand provider in China, has over 20 million unique users viewing its videos every month.
- At a high level, P2P video streaming is very similar to BitTorrent file downloading.
- When a peer wants to see a video, it contacts a tracker (which may be centralized or peer-based using a DHT) to discover other peers in the system that have a copy of that video.
- This peer then requests chunks of the video file in parallel from these other peers that have the file.
- Different from downloading with BitTorrent, however, requests are preferentially made for chunks that are to be played back in the near future in order to ensure continuous playback.
- The Kankan design employs a tracker and its own DHT for tracking content.

Network Support for Multimedia

There exist three broad approaches towards providing network-level support for multimedia applications.

Approach	Granularity	Guarantee	Mechanisms	Complexity	Deployment to date
Making the best of best-effort service.	all traffic treated equally	none, or soft	application-layer support, CDNs, overlays, network-level resource provisioning	minimal	everywhere
Differentiated service	different classes of traffic treated differently	none, or soft	packet marking, policing, scheduling	medium	some
Per-connection Quality-of-Service (QoS) Guarantees	each source-destination flows treated differently	soft or hard, once flow is admitted	packet marking, policing, scheduling; call admission and signaling	light	little

- **Making the best of best-effort service:** The application-level mechanisms and infrastructure can be successfully used in a well-dimensioned network where packet loss and excessive end-to-end delay rarely occur. When demand increases are forecasted, the ISPs deploy additional bandwidth and switching capacity to continue to ensure satisfactory delay and packet-loss performance.
- **Differentiated service:** With differentiated service, one type of traffic might be given strict priority over another class of traffic when both types of traffic are queued at a router.
- **Per-connection Quality-of-Service (QoS) Guarantees:** With per-connection QoS guarantees, each instance of an application explicitly reserves end-to-end bandwidth and thus has a guaranteed end-to-end performance. A hard guarantee means the application will receive its requested quality of service (QoS) with certainty. A soft guarantee means the application will receive its requested quality of service with high probability.

➔ Dimensioning Best-Effort Networks

- A first approach to improving the quality of multimedia applications is through providing enough link capacity throughout the network so that network congestion, and its consequent

packet delay and loss, never (or only very rarely) occurs. With enough link capacity, packets could zip through today's Internet without queuing delay or loss.

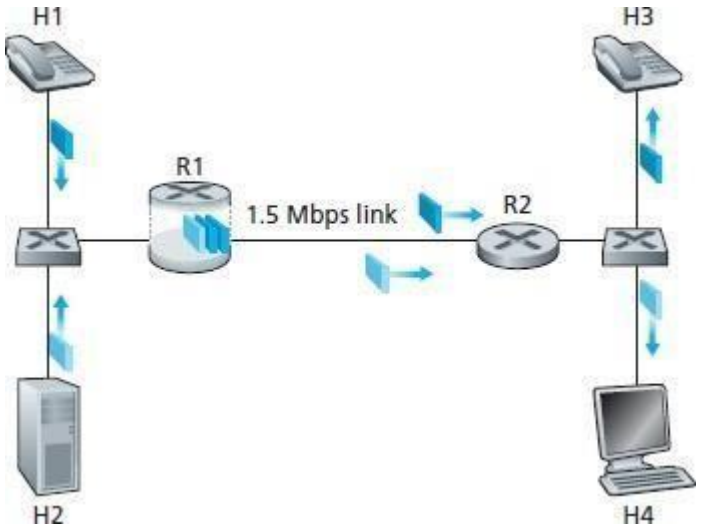
- Challenges:
  - The question of how much capacity to provide at network links in a given topology to achieve a given level of performance is often known as **bandwidth provisioning**.
  - The even more complicated problem of how to design a network topology (where to place routers, how to interconnect routers with links, and what capacity to assign to links) to achieve a given level of end-to-end performance is a network design problem often referred to as **network dimensioning**.

➔ Providing Multiple Classes of Service

The simplest enhancement to the one-size-fits-all best-effort service in today's Internet is to divide traffic into classes, and provide different levels of service to these different classes of traffic.

The type-of-service (ToS) field in the IPv4 header can be used for this purpose.

Motivating Scenarios



Here H1 and H3 are using audio application, H2 and H4 are using HTTP web application.

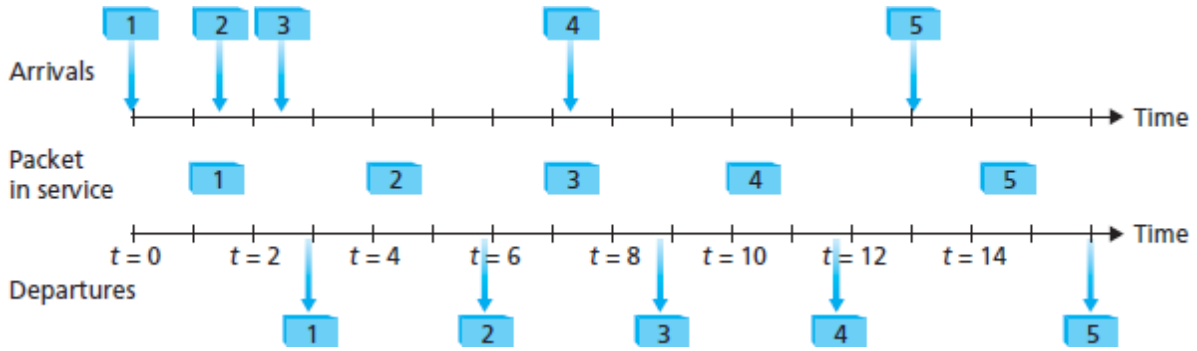
- In the best-effort Internet, the audio and HTTP packets are mixed in the output queue at R1 and (typically) transmitted in a first-in-first-out (FIFO) order.

- In this scenario, a burst of packets from the Web server could potentially fill up the queue, causing IP audio packets to be excessively delayed or lost due to buffer overflow at R1.
- Solution for this is differentiating traffic class and assigning suitable priority to it.
- **Packet marking** allows a router to distinguish among packets belonging to different classes of traffic.
- Now suppose that the router is configured to give priority to packets marked as belonging to the 1 Mbps audio application. Since the outgoing link speed is 1.5 Mbps, even though the HTTP packets receive lower priority, they can still, on average, receive 0.5 Mbps of transmission service. But if the audio application starts sending packets at a rate of 1.5 Mbps or higher, the HTTP packets will starve, that is, they will not receive any service on the R1-to-R2 link.
- Therefore it is desirable to provide a degree of traffic isolation among classes so that one class is not adversely affected by another class of traffic that misbehaves.
- If a traffic class or flow must meet certain criteria, then a policing mechanism can be put into place to ensure that these criteria are indeed observed. If the policed application misbehaves, the policing mechanism will take some action so that the traffic actually entering the network conforms to the criteria.
- A complementary approach for providing isolation among traffic classes is for the link-level packet-scheduling mechanism to explicitly allocate a fixed amount of link bandwidth to each class.
- While providing isolation among classes or flows, it is desirable to use resources (for example, link bandwidth and buffers) as efficiently as possible.

➔ Scheduling Mechanisms

First-In-First-Out (FIFO)

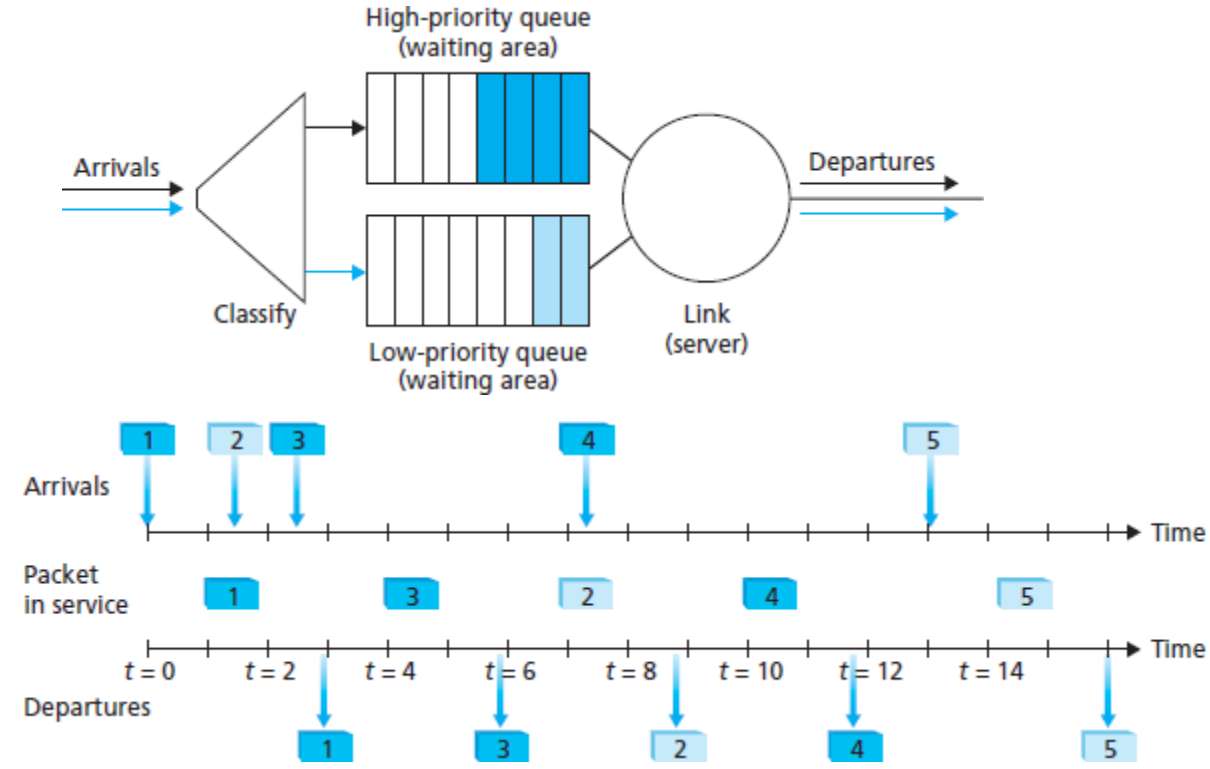
The FIFO (also known as first-come-first-served, or FCFS) scheduling discipline selects packets for link transmission in the same order in which they arrived at the output link queue.



Priority Queuing

Under priority queuing, packets arriving at the output link are classified into priority classes at the output queue.

Each priority class typically has its own queue. When choosing a packet to transmit, the priority queuing discipline will transmit a packet from the highest priority class that has a nonempty queue. The choice among packets in the same priority class is typically done in a FIFO manner.



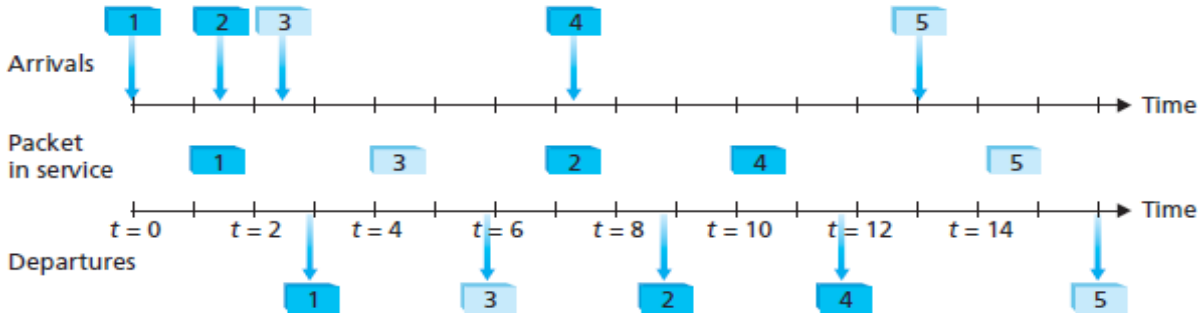
Round Robin

Under the round robin queuing discipline, packets are sorted into classes as with priority queuing.

However, rather than there being a strict priority of service among classes, a round robin scheduler alternates service among the classes.

In the simplest form of round robin scheduling, a class 1 packet is transmitted, followed by a class 2 packet, followed by a class 1 packet, followed by a class 2 packet, and so on.

A work-conserving round robin discipline that looks for a packet of a given class but finds none will immediately check the next class in the round robin sequence.

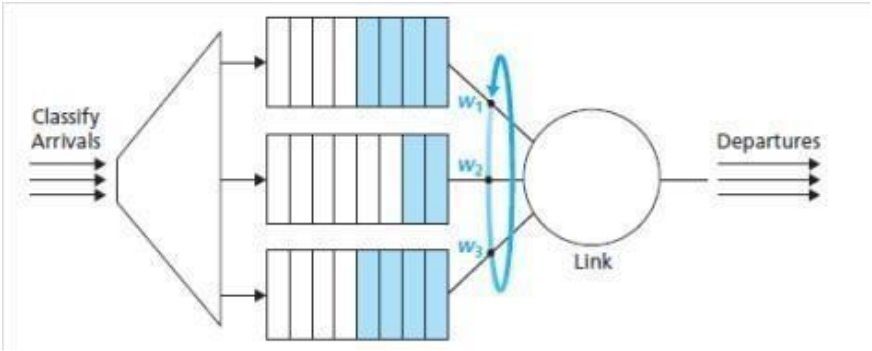


Weighted Fair Queuing (WFQ)

A generalized abstraction of round robin queuing that has found considerable use in QoS architectures is weighted fair queuing (WFQ) discipline.

Here arriving packets are classified and queued in the appropriate per-class waiting area. As in round robin scheduling, a WFQ scheduler will serve classes in a circular manner—first serving class 1, then serving class 2, then serving class 3, and then (assuming there are three classes) repeating the service pattern.

WFQ is also a work-conserving queuing discipline and thus will immediately move on to the next class in the service sequence when it finds an empty class queue.

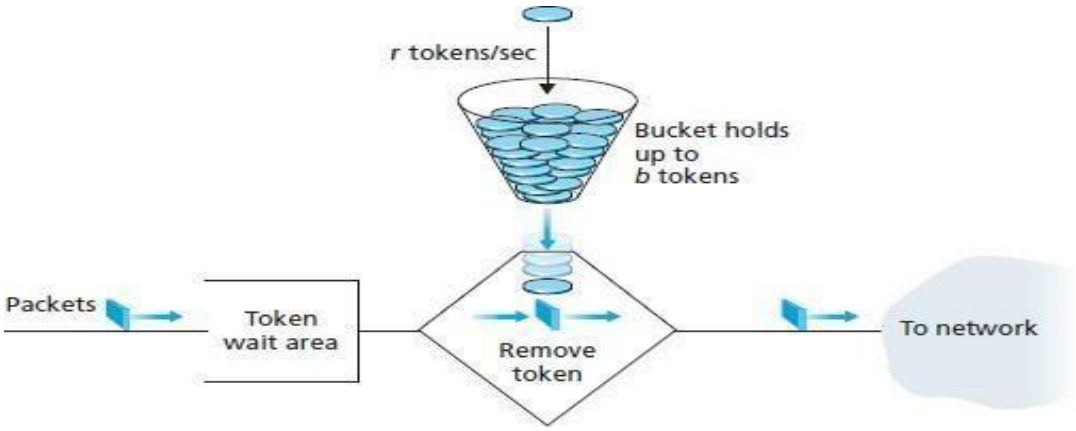


➔ Policing: The Leaky Bucket

Three important policing criteria:

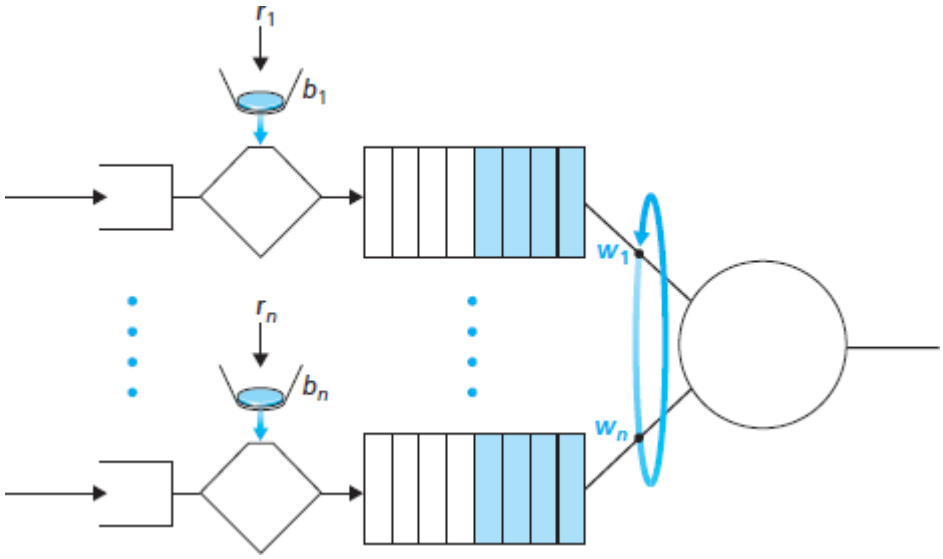
- **Average rate:** The network may wish to limit the long-term average rate (packets per time interval) at which a flow's packets can be sent into the network. A crucial issue here is the interval of time over which the average rate will be policed.
- **Peak rate:** While the average-rate constraint limits the amount of traffic that can be sent into the network over a relatively long period of time, a peak-rate constraint limits the maximum number of packets that can be sent over a shorter period of time.
- **Burst size:** The network may also wish to limit the maximum number of packets (the "burst" of packets) that can be sent into the network over an extremely short interval of time.

The leaky bucket mechanism is an abstraction that can be used to characterize these policing limits.



- A leaky bucket consists of a bucket that can hold up to b tokens.
- Tokens are added to this bucket as follows. New tokens, which may potentially be added to the bucket, are always being generated at a rate of r tokens per second.
- If the bucket is filled with less than b tokens when a token is generated, thenewly generated token is added to the bucket; otherwise the newly generated token is ignored, and the token bucket remains full with b tokens.
- Suppose that before a packet is transmitted into the network, it must first remove a token from the token bucket. If the token bucket is empty, the packet must wait for a token.

- Because there can be at most b tokens in the bucket, the maximum burst size for a leaky-bucket policed flow is b packets. Furthermore, because the token generation rate is r, the maximum number of packets that can enter the network of any interval of time of length t is rt + b.
- Leaky Bucket + Weighted Fair Queuing = Provable Maximum Delay in a Queue



➔ Diffserv

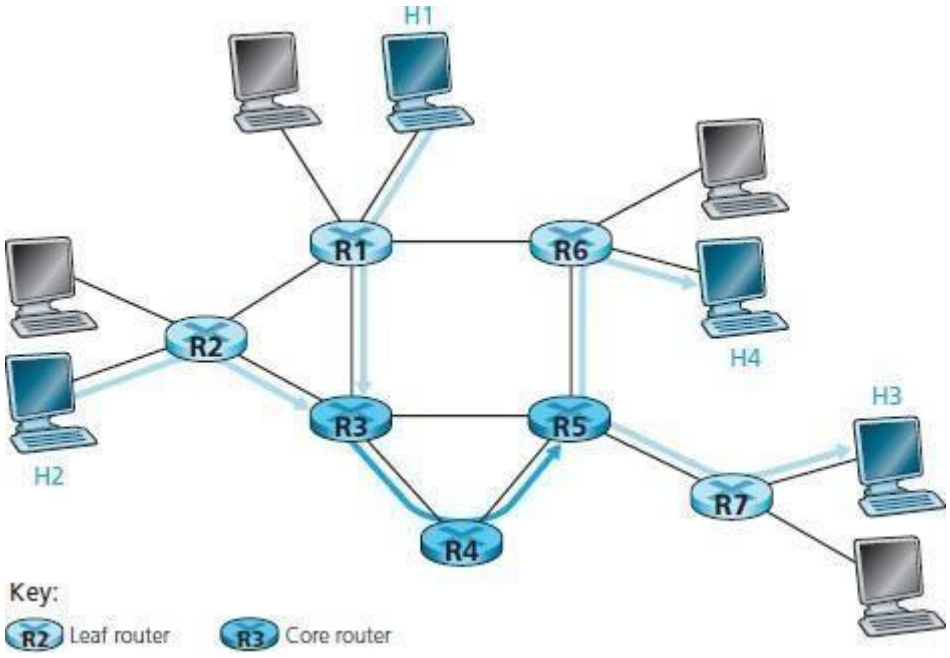
Diffserv provides service differentiation—that is, the ability to handle different classes of traffic in different ways within the Internet in a scalable manner.

The need for scalability arises from the fact that millions of simultaneous source-destination traffic flows may be present at a backbone router.

The Diffserv architecture consists of two sets of functional elements:

1) Edge functions: packet classification and traffic conditioning. At the incoming edge of the network arriving packets are marked. The mark that a packet receives identifies the class of traffic to which it belongs. Different classes of traffic will then receive different service within the core network.

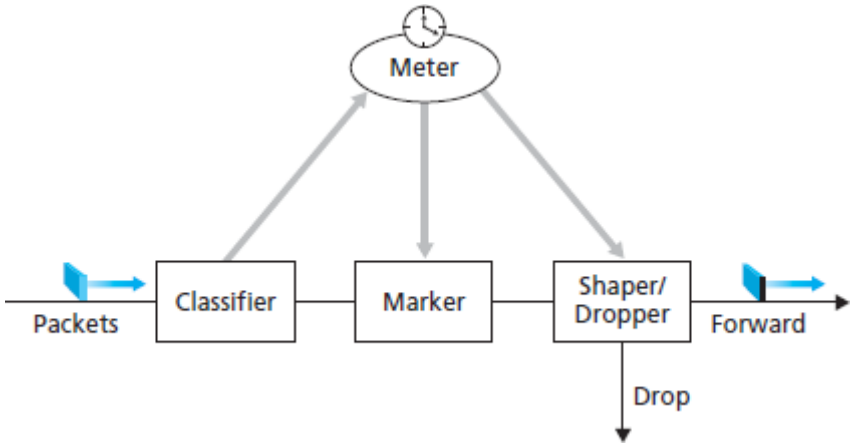
2) Core function: forwarding. When a DS-marked packet arrives at a Diffserv capable router, the packet is forwarded onto its next hop according to the so-called per-hop behavior (PHB) associated with that packet's class. The per-hop behavior influences how a router's buffers and link bandwidth are shared among the competing classes of traffic.



- Packets arriving to the edge router are first classified. The classifier selects packets based on the values of one or more packet header fields (for example, source address, destination address, source port, destination port, and protocol ID) and steers the packet to the appropriate marking function.
- In some cases, an end user may have agreed to limit its packet-sending rate to conform to a declared traffic profile. The traffic profile might contain a limit on the peak rate, as well as the burstiness of the packet flow.
- As long as the user sends packets into the network in a way that conforms to the negotiated traffic profile, the packets receive their priority marking and are forwarded along their route to the destination.
- On the other hand, if the traffic profile is violated, out-of-profile packets might be marked differently, might be shaped (for example, delayed so that a maximum rate constraint would be observed), or might be dropped at the network edge.
- The role of the metering function, is to compare the incoming packet flow with the negotiated traffic profile and to determine whether a packet is within the negotiated traffic profile.
- The second key component of the Diffserv architecture involves the per-hop behavior (PHB) performed by Diffserv-capable routers. PHB is rather cryptically, but carefully, defined as “a

description of the externally observable forwarding behavior of a Diffserv node applied to a particular Diffserv behavior aggregate".

- A PHB can result in different classes of traffic receiving different performance.

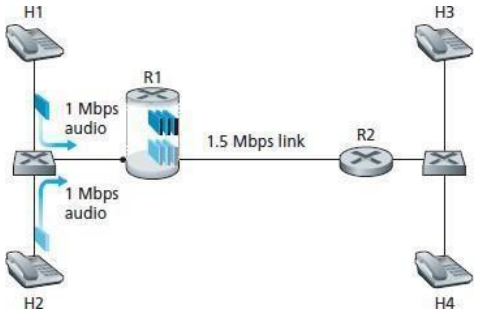


- The **expedited forwarding** PHB specifies that the departure rate of a class of traffic from a router must equal or exceed a configured rate.
- The **assured forwarding** PHB divides traffic into four classes, where each AF class is guaranteed to be provided with some minimum amount of bandwidth and buffering.

Per-Connection Quality-of-Service (QoS) Guarantees: Resource Reservation and Call Admission

Consider two 1 Mbps audio applications transmitting their packets over the 1.5 Mbps link. The combined data rate of the two flows (2 Mbps) exceeds the link capacity.

There is simply not enough bandwidth to accommodate the needs of both applications at the same time. If the two applications equally share the bandwidth, each application would lose 25 percent of its transmitted packets.



If sufficient resources will not always be available, and QoS is to be guaranteed, a call admission process is needed in which flows declare their QoS requirements and are then either admitted to the network (at the required QoS) or blocked from the network (if the required QoS cannot be provided by the network).

The process of having a flow declare its QoS requirement, and then having the network either accept the flow (at the required QoS) or block the flow is referred to as the call admission process.

Resource reservation: The only way to guarantee that a call will have the resources (link bandwidth, buffers) needed to meet its desired QoS is to explicitly allocate those resources to the call—a process known in networking parlance as resource reservation. Once resources are reserved, the call has on-demand access to these resources throughout its duration, regardless of the demands of all other calls. If a call reserves and receives a guarantee of x Mbps of link bandwidth, and never transmits at a rate greater than x, the call will see loss- and delay-free performance.

Call admission: If resources are to be reserved, then the network must have a mechanism for calls to request and reserve resources. Since resources are not infinite, a call making a call admission request will be denied admission, that is, be blocked, if the requested resources are not available. Such a call admission is performed by the telephone network—we request resources when we dial a number. If the circuits (TDMA slots) needed to complete the call are available, the circuits are allocated and the call is completed. If the circuits are not available, then the call is blocked, and we receive a busy signal. A blocked call can try again to gain admission to the network, but it is not allowed to send traffic into the network until it has successfully completed the call admission process. Of course, a router that allocates link bandwidth should not allocate more than is available at that link. Typically, a call may reserve only a fraction of the link's bandwidth, and so a router may allocate link bandwidth to more than one call. However, the sum of the allocated bandwidth to all calls should be less than the link capacity if hard quality of service guarantees are to be provided.

Call setup signaling: The call admission process described above requires that a call be able to reserve sufficient resources at each and every network router on its source-to-destination path to ensure that its end-to-end QoS requirement is met. Each router must determine the local resources required by the session, consider the amounts of its resources that are already

committed to other ongoing sessions, and determine whether it has sufficient resources to satisfy the per-hop QoS requirement of the session at this router without violating local QoS guarantees made to an already-admitted session. A signaling protocol is needed to coordinate these various activities—the per-hop allocation of local resources, as well as the overall end-to-end decision of whether or not the call has been able to reserve sufficient resources at each and every router on the end-to-end path. The RSVP protocol was proposed for this purpose within an Internet architecture for providing qualityof- service guarantees.