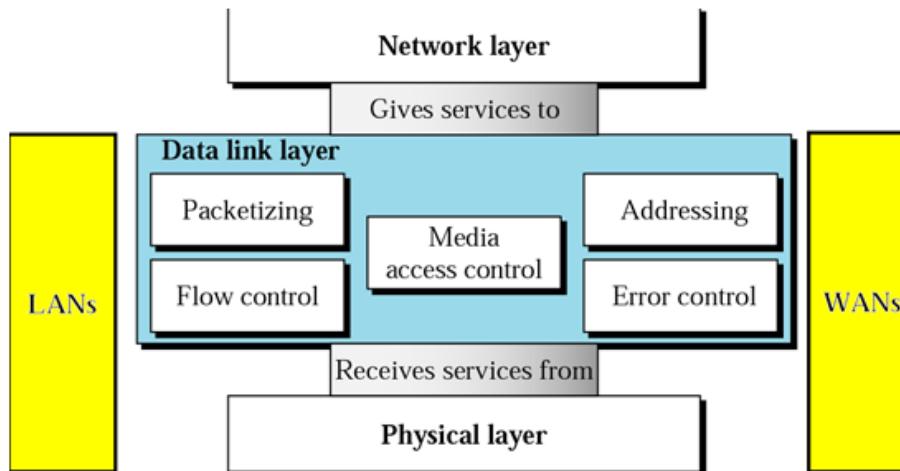


Overview of DLL

- The data link layer transform the physical layer, a raw transmission facility to a link responsible for node-to-node transmission
- Specific responsibilities of data link layer includes framing, addressing, flow control, error control and media access control



Design Issues in Data Link Layer

- Services provided
 - The data link layer act as a service interface to the network layer
 - The principle service is transferring data from network layer on sending machine to the network layer on destination machine
- Frame Synchronization
 - The source machine sends data in the form of blocks called frames to the destination machine
 - The starting and ending of each frame should be identified so that the frame can be recognized by the destination machine

Design Issues in Data Link Layer

- Flow control
 - Flow control is done to prevent the flow of data frame at the receiver end
 - The source machine must not send data frames at a rate faster than the capacity of destination machine to accept them
- Error control
 - Error control is done to prevent duplication of frames
 - The errors introduced during transmission from source to destination machines must be detected and corrected at the destination machine

Framing

- Data transmission in the physical layer means moving bits in the form of a signal from the source to the destination
- The physical layer provides bit synchronization to ensure that the sender and receiver use the same bit durations and timing
- The data link layer on the other hand needs to pack bits into frames, so that each frame is distinguishable from another

Why whole message is not packed in single frame?

- Although the whole message could be packed in one frame, generally it is not normally done
- One reason is that a frame can be very large, making flow and error control very inefficient
- When a message is carried in one very large frame, even a single-bit error would require the retransmission of the whole message
- When a message is divided into smaller frames, a single-bit error affects only that small frame

Parts of a Frame

- A frame has the following parts
 - Frame Header: It contains the source and the destination addresses
 - Payload field: It contains the message to be delivered
 - Trailer: It contains the error detection and error correction bits
 - Flag: It marks the beginning and end of the frame



Types of Framing

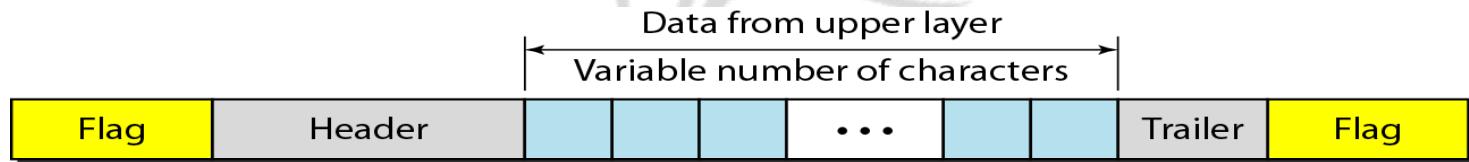
- Fixed size
 - In fixed-size framing, there is no need for defining the boundaries of the frames
 - The size itself can be used as a delimiter
- Drawback
 - It suffers from internal fragmentation if data size is less than frame size

Types of Framing

- Variable-Size Framing
 - In variable-size framing, we need a way to define the end of the frame and the beginning of the next
- Historically, two approaches were used for this purpose
 - Character-oriented approach
 - Bit-oriented approach

Character-oriented approach

- To separate one frame from the next, an 8-bit (1-byte) flag is added at the beginning and end of a frame
- The flag composed of protocol-dependent special characters, which signals the start or end of a frame

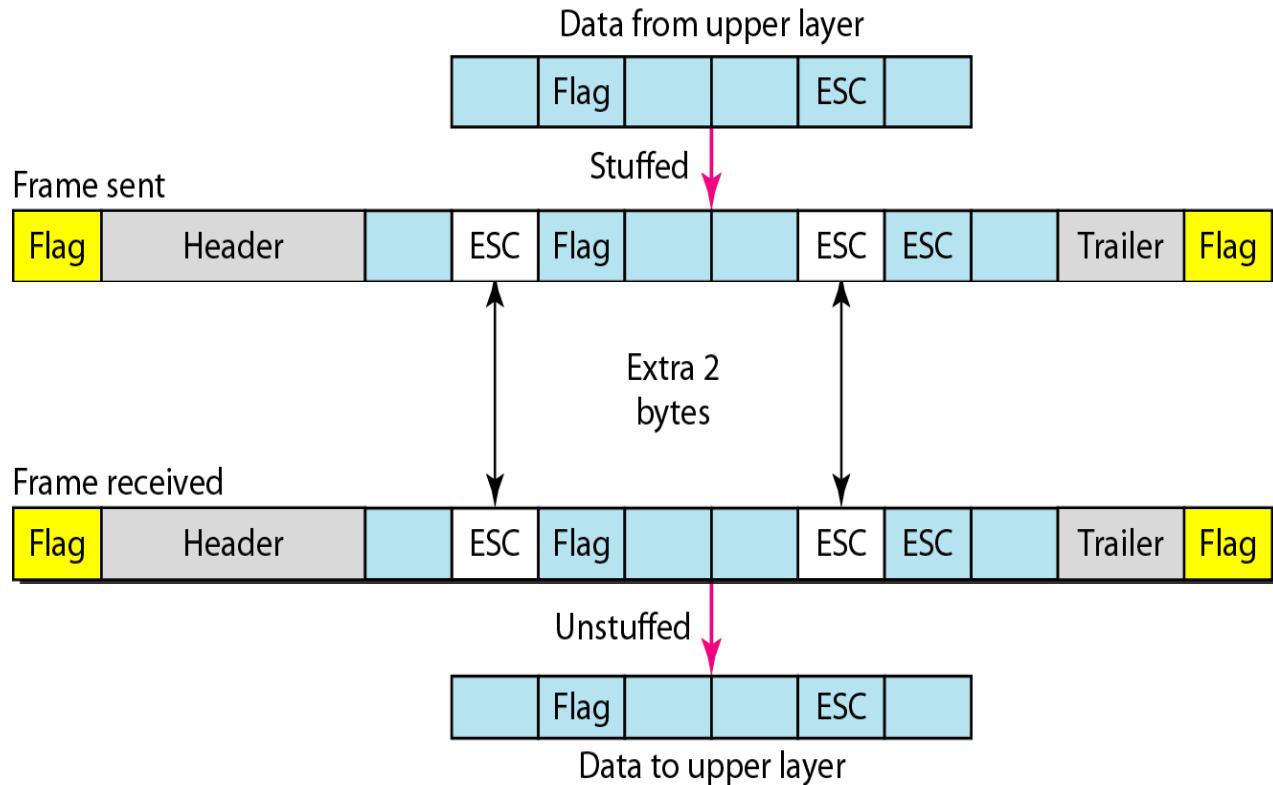


- Problem???

Character-oriented protocols

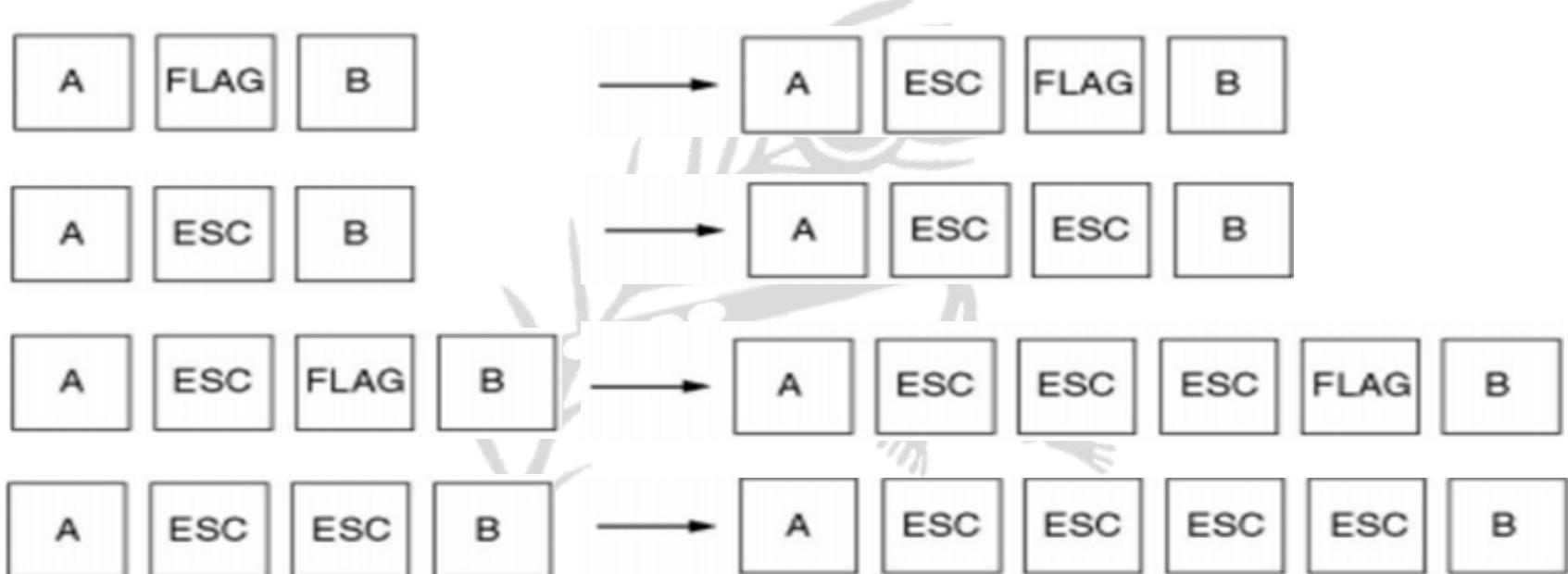
- In byte stuffing, a special byte is added to the data section of the frame when there is a character with the same pattern as the flag
- The data section is stuffed with an extra byte
- This byte is usually called the escape character (ESC), which has a predefined bit pattern
- Whenever the receiver encounters the ESC character, it removes it from the data section and treats the next character as data, not a delimiting flag

Character-oriented protocols



Character-oriented protocols

- Eg:



Character-oriented protocols

- Eg:
 - Considering “f” as flag delimiter and “e” as DLE (escape) character, Apply the byte stuffing on the single frame data

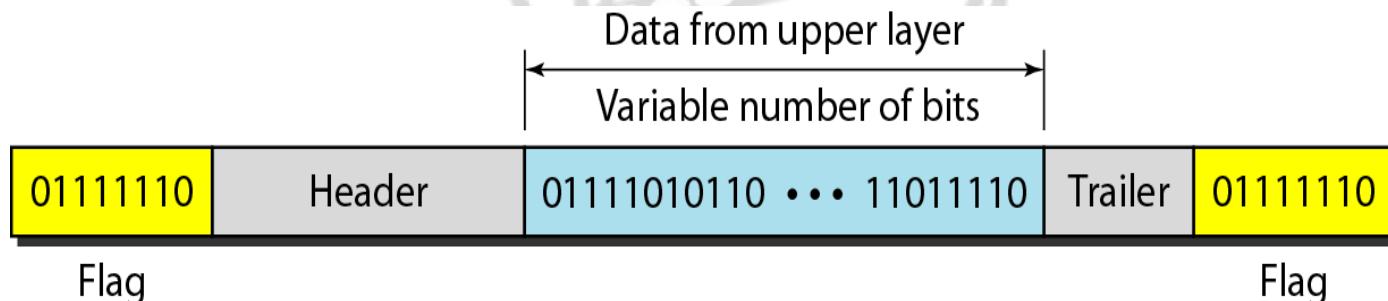
five and four

Solution

f efivee and efour f

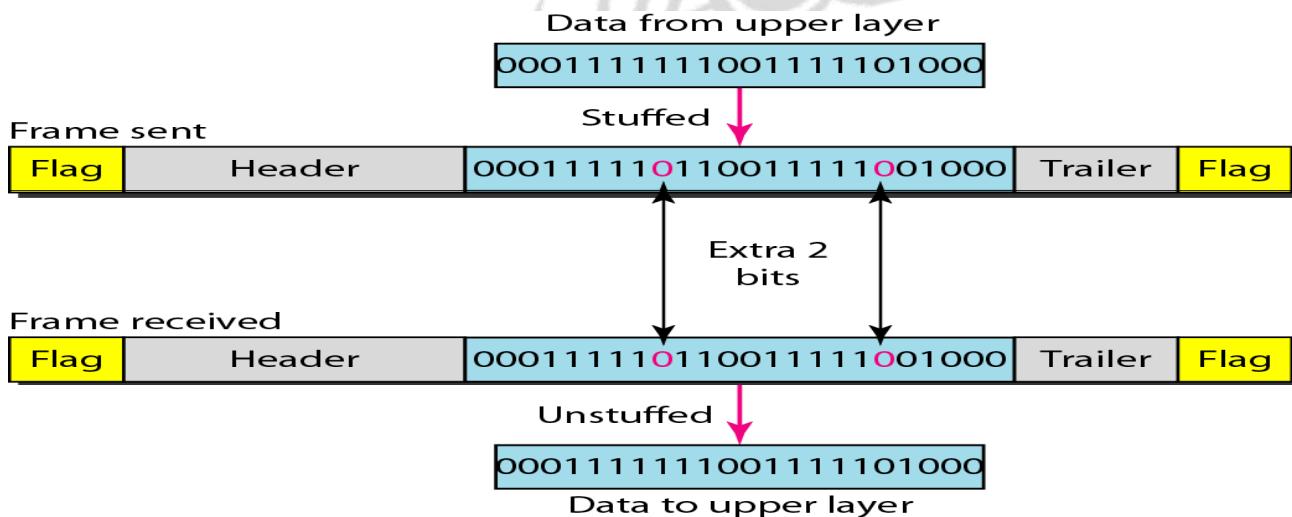
Bit-oriented protocols

- In a bit-oriented protocol, the data section of a frame is a sequence of bits to be interpreted by the upper layer
- However, in addition to headers and trailers, we still need a delimiter to separate one frame from the other.
- Most protocols use a special 8-bit pattern flag 01111110 as the delimiter to define the beginning and the end of the frame



Bit-oriented protocols

- The data may contain the special bit-pattern from the upper layer
- Bit stuffing is the process of adding one extra 0 whenever five consecutive 1's are received, so that the receiver does not mistake the pattern 01111110 for a flag



Bit-oriented protocols

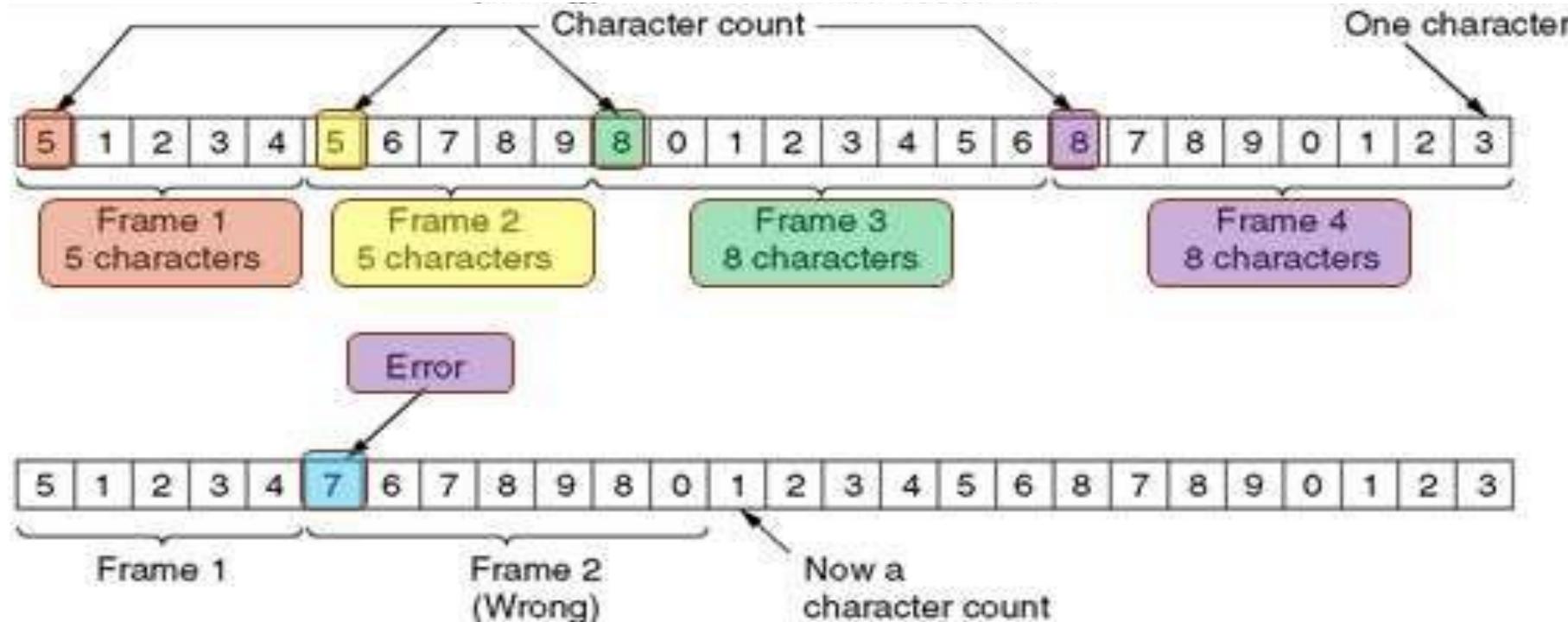
- Eg:
 - Assume, we send a frame of 2 8-bit characters: char1 (01111111) and char2 (01111101)
- Sender rule
 - if 5 1s in data, then add (stuff) a 0
- On the line, we'll send
 - 01111110 011111011 011111001 01111110
- Receiver rule
 - if a 0 occurs after 5 1s, then remove it

01111110 011111-11 011111-01 01111110

Framing

- The framing method uses a field in the header to specify the number of characters in the frame
- When the data link layer at the destination sees the character count, it knows how many characters to follow and hence where the end of the frame
- The trouble with this algorithm is that the count can be corrupted by a transmission error

Framing



ERROR DETECTION AND CORRECTION

Error Detection and Correction

- Data can be corrupted during transmission
- Some applications may be required to detect and correct the errors
- There are many reasons such as noise, cross-talk etc., which is induced in data to get corrupted during transmission
- Data-link layer uses some error control mechanism to ensure that frames (data bit streams) are transmitted with certain level of accuracy

Error

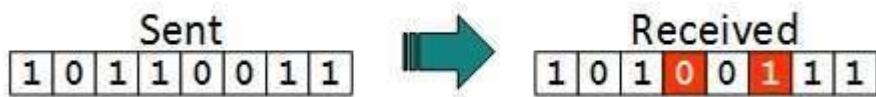
- A condition when the receiver's information does not match with the sender's information
- During transmission, digital signals suffer from noise that can introduce errors in the binary bits travelling from sender to receiver
- This means a 0 may change to 1 or vice-versa
- Note: Error Detecting Codes (Implemented either at Data link layer or Transport Layer of OSI Model)

Types of Error

- Single bit error
 - In a frame, there is only one bit, anywhere, which is corrupt



- Multiple bits error
 - Frame is received with more than one bits corrupted

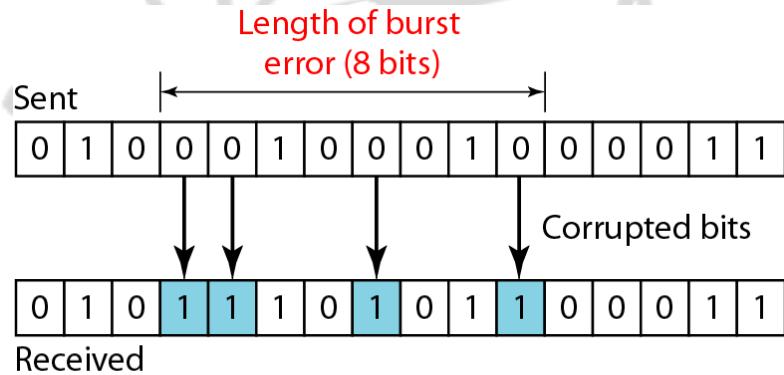


- Burst error
 - Frame contains more than 1 consecutive bits corrupted



Burst Error

- The term burst error means that two or more consecutive bits in the data unit have changed from 1 to 0 or from 0 to 1
- Burst errors does not necessarily mean that the errors occur in consecutive bits, the length of the burst is measured from the first corrupted bit to the last corrupted bit. Some bits in between may not have been corrupted
- NOTE: To detect or correct errors, we need to send extra (redundant) bits with data



Error control mechanism

- Error detection
- Error correction



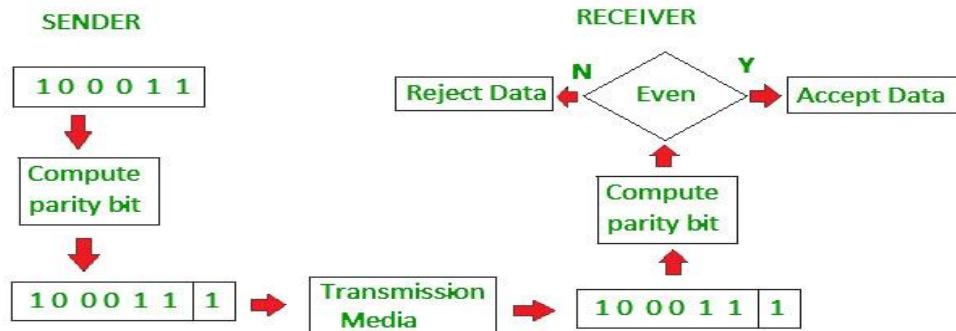
Error control mechanism

- Error detection:
 - Whenever a message is transmitted, it may get scrambled by noise or data may get corrupted
 - To avoid this, we use error-detecting codes which are additional data added to a given digital message to help us detect if any error has occurred during transmission of the message
 - Basic approach used for error detection is the use of redundancy bits, where additional bits are added to facilitate detection of errors
- Some popular techniques for error detection are:
 - Simple parity check
 - Two-dimensional parity check
 - Checksum
 - Cyclic redundancy check

Error control mechanism

Simple Parity check

- Blocks of data from the source are subjected to a check bit or parity bit generator, where a parity of:
 - 1 is added to the block if it contains odd number of 1's, and
 - 0 is added if it contains even number of 1's
- This scheme makes the total number of 1's even, that is why it is called even parity checking



Error control mechanism

Two Parity check

- Parity check bits are calculated for each row, which is equivalent to a simple parity check bit
- Parity check bits are also calculated for all columns, then both are sent along with the data
- At the receiving end these are compared with the parity bits calculated on the received data

Error control mechanism

Original Data

10011001	11100010	00100100	10000100
-----------------	-----------------	-----------------	-----------------

Row parities

1 0 0 1 1 0 0 1	0
1 1 1 0 0 0 1 0	0
0 0 1 0 0 1 0 0	0
1 0 0 0 0 1 0 0	0
1 1 0 1 1 0 1 1	0

Column parities →

100110010	111000100	001001000	100001000	110110110
------------------	------------------	------------------	------------------	------------------

Data to be sent

Error control mechanism

Check Sum

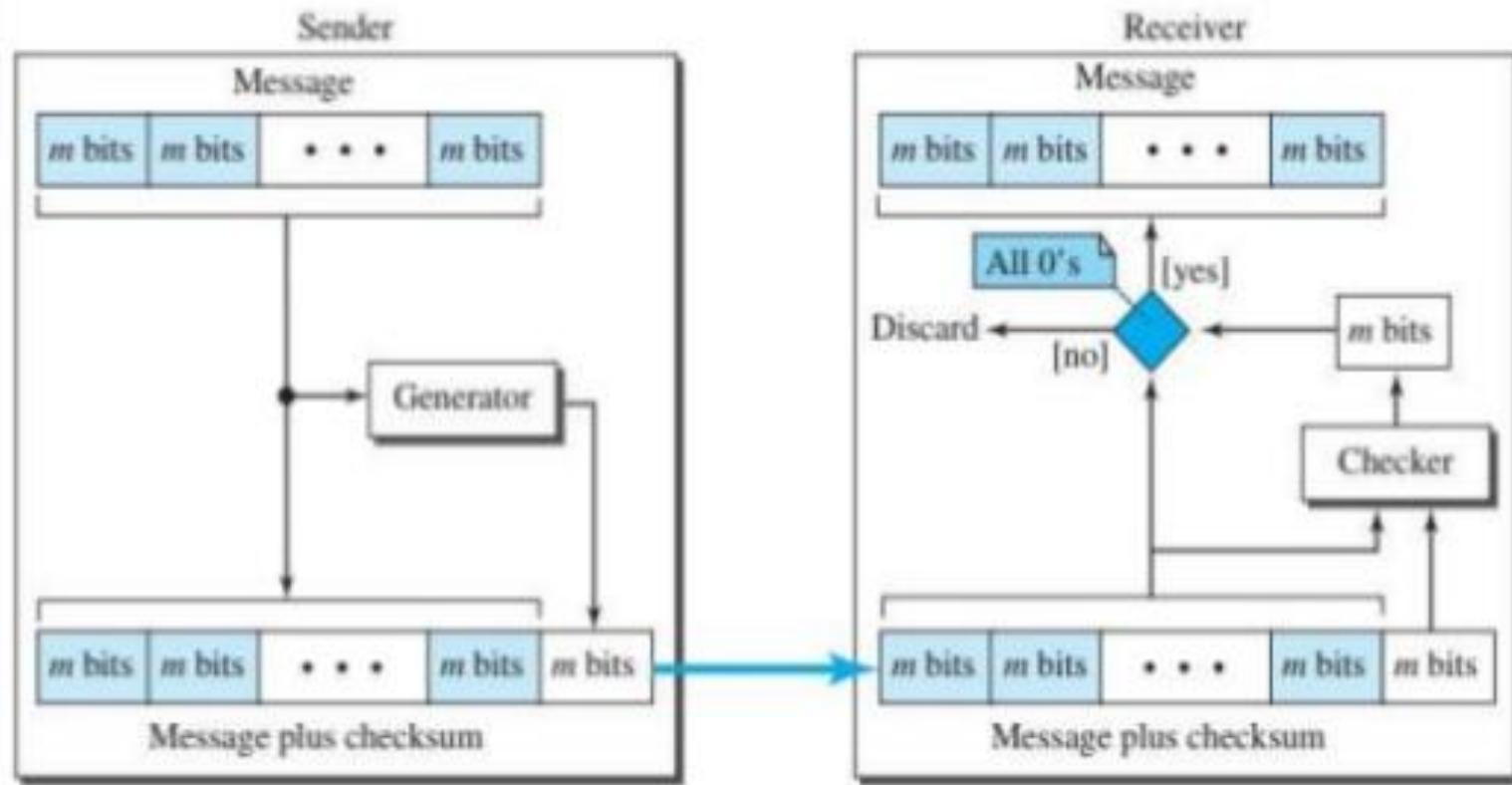
- Error detecting technique
- Applied to a message of any length
- Mostly used at network and transport layer rather to data link layer

Error control mechanism

Check Sum - Approach

- At the source, the message is divided into m-bits unit
- The generator than creates a checksum (an extra m-bit unit)
- At the destination, the checker creates a new checksum from the combination of message and sent checksum
- If checksum is all 0s
 - Message is accepted
 - Else
 - Message is discarded

Error control mechanism



Error control mechanism

- Eg:
 - if the set of numbers is (7, 11, 12, 0, 6)
 - we send (7, 11, 12, 0, 6, 36), where 36 is the sum of the original numbers
 - The receiver adds the five numbers and compares the result with the sum
 - If the two are the same, the receiver assumes no error, accepts the five numbers, and discards the sum
 - Otherwise, there is an error somewhere and the data are not accepted

Error control mechanism

- Eg:
 - Suppose that the sender wants to send 4 frames each of 8 bits, where the frames are 11001100, 10101010, 11110000 and 11000011

Frame 1:	11001100
Frame 2:	+ 10101010
Partial Sum:	1 01110110
	+ 1
	01110111
Frame 3:	+ 11110000
Partial Sum:	1 01100111
	+ 1
	01101000
Frame 4:	+ 11000011
Partial Sum:	1 00101011
	+ 1
Sum:	00101100
Checksum:	11010011

Frame 1:	11001100
Frame 2:	+ 10101010
Partial Sum:	1 01110110
	+ 1
	01110111
Frame 3:	+ 11110000
Partial Sum:	1 01100111
	+ 1
	01101000
Frame 4:	+ 11000011
Partial Sum:	1 00101011
	+ 1
Sum:	00101100
Checksum:	11010011
Sum:	11111111
Complement:	00000000
Hence accept frames.	

Error control mechanism

Cyclic Redundancy Check (CRC)

- CRC is a method of detecting accidental changes/errors in the communication channel
- CRC is used in networks such as LANs and WANs
- CRC involves binary division of the data bits being sent by a predetermined divisor agreed upon by the communicating system
- The divisor is generated using polynomials so, CRC is also called polynomial code checksum

$$0 \oplus 0 = 0$$

$$1 \oplus 1 = 0$$

a. Two bits are the same, the result is 0.

$$0 \oplus 1 = 1$$

$$1 \oplus 0 = 1$$

b. Two bits are different, the result is 1.

$$\begin{array}{r}
 & 1 & 0 & 1 & 1 & 0 \\
 \oplus & 1 & 1 & 1 & 0 & 0 \\
 \hline
 & 0 & 1 & 0 & 1 & 0
 \end{array}$$

c. Result of XORing two patterns

Error control mechanism

Cyclic Redundancy Check (CRC) - Approach

- Sender Side
 - The binary data is first augmented by adding $k-1$ zeros in the end of the data
 - Use modulo-2 binary division to divide binary data by the key and store remainder of division
 - Append the remainder at the end of the data to form the encoded data and send the same
- n : Number of bits in data to be sent from sender side
- k : Number of bits in the key obtained from generator polynomial

Error control mechanism

Cyclic Redundancy Check (CRC) - Approach

- Receiver Side
 - Perform modulo-2 division again and if the remainder is 0, then there are no errors
- Modulo 2 Division: The process of modulo-2 binary division is the same as the familiar division process we use for decimal numbers.
- Instead of subtraction, we use XOR here

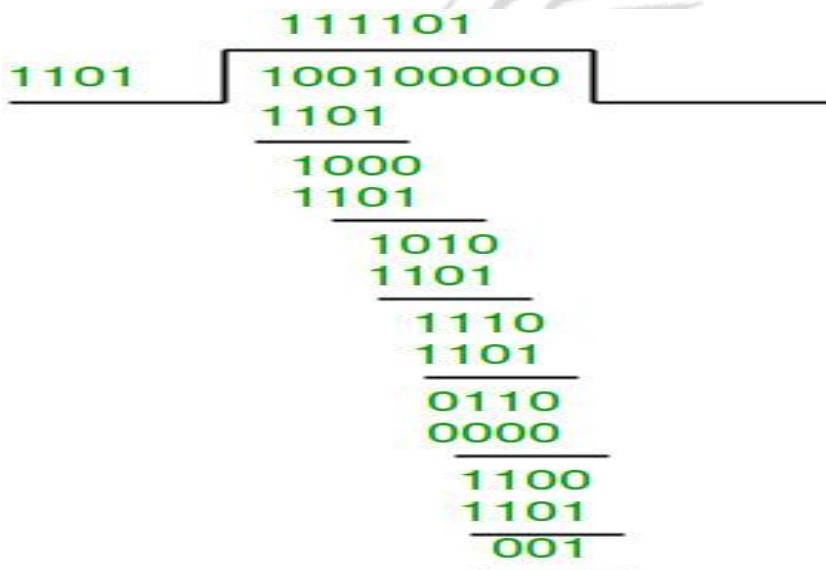
Error control mechanism

Cyclic Redundancy Check (CRC) - Approach

- Case - 01: Remainder = 0
 - If the remainder is zero
 - Receiver assumes that no error occurred in the data during the transmission
 - Receiver accepts the data
- Case - 02: Remainder \neq 0
 - If the remainder is non-zero
 - Receiver assumes that some error occurred in the data during the transmission
 - Receiver rejects the data and asks the sender for retransmission

Error control mechanism

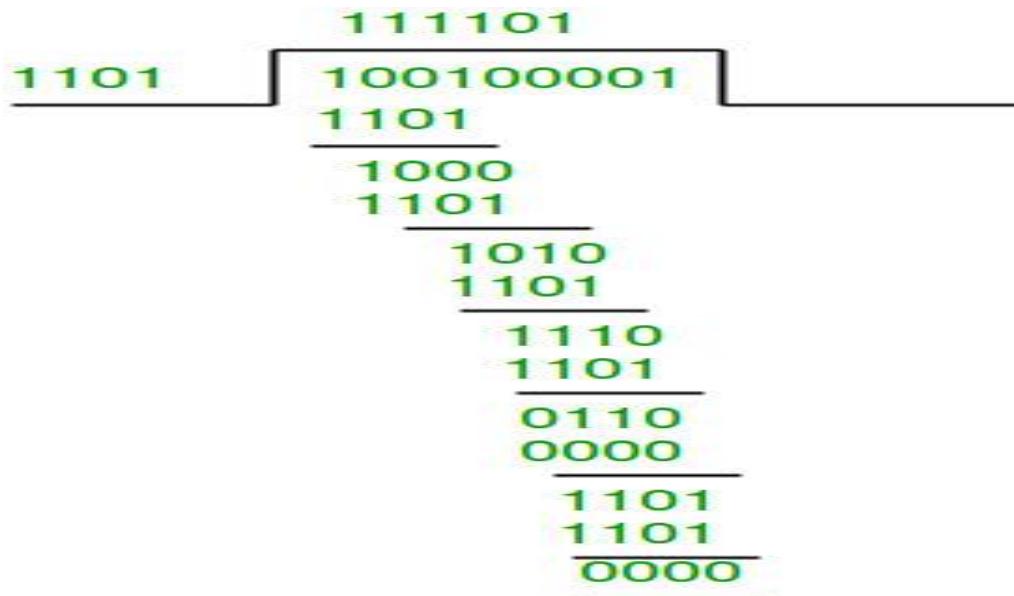
- Eg:
 - Data word to be sent - 100100 Key - 1101 [Or generator polynomial $x^3 + x^2 + 1$]



- The remainder is 001 and hence the encoded data sent is 100100001

Error control mechanism

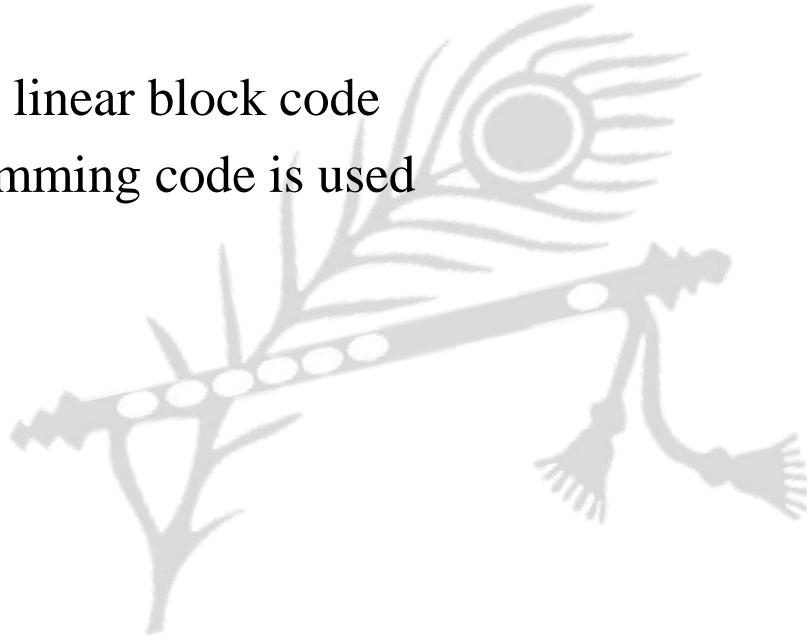
- Code word received at the receiver side 100100001



- The remainder is all zeros. Hence, the data received has no error

Hamming Code

- Hamming code is a set of error-correction codes that can be used to detect and correct the errors
- Hamming codes are linear block code
- Commonly 7-bit hamming code is used



Hamming Code

- Redundant bits
 - Redundant bits are extra binary bits that are generated and added to the information-carrying bits of data transfer to ensure that no bits were lost during the data transfer
- The number of redundant bits can be calculated using the following formula:
 - $2^r \geq m + r + 1$
 - where, r = redundant bit, m = data bit
- Suppose the number of data bits are 7, then the number of redundant bits can be calculated using:
- $2^4 \geq 7 + 4 + 1$
- Thus, the number of redundant bits = 4

Hamming Code

- General algorithm
 - Write the bit positions starting from 1 in binary form (1, 10, 11, 100, etc)
 - All the bit positions that are a power of 2 are marked as parity bits (1, 2, 4, 8, etc)
 - Each data bit is included in a unique set of parity bits, as determined its bit position in binary form

Hamming Code

- Selection of parity bits



- Parity bit 1 covers all the bits positions whose binary representation includes a 1 in the least significant position (1, 3, 5, 7 etc)
- Parity bit 2 covers all the bits positions whose binary representation includes a 1 in the second position from the least significant bit (2, 3, 6, 7 etc)
- Parity bit 4 covers all the bits positions whose binary representation includes a 1 in the third position from the least significant bit (4,5,6,7 etc)

Hamming Code

- Eg:
 - A bit word 1011 is to be transmitted construct even parity 7 bit hamming code.
- Step-1



- $P_1 = 1,3,5,7 = [P_1,111]$ as nos. of 1's are odd so P_1 will be set to 1, [1111]
- $P_2 = 2,3,6,7 = [P_2,101]$ as nos. of 1's are even so P_2 will be set to 0, [0101]
- $P_4 = 4,5,6,7 = [P_4,101]$ as nos. of 1's are even so P_3 will be set to 0, [0101]
- So, required 7 bit hamming code is 1010101

Detection of Error in Hamming Code

- At Receiver end decoded data is checked for the following groups;
- Group-1[1,3,5,7]
- Group-2[2,3,6,7]
- Group-3[4,5,6,7]
- If all group contains even parity then there is no error

Error Correction in Hamming code

- Eg:
 - Codeword received 1011011, assume even parity. State whether received codeword is correct or wrong. Locate the bit which has error

Solution

- 1(D7) 0(D6) 1(D5) 1(P4) 0(D3) 1(P2) 1(P1)
- Check for error for even parity
- Data at 1,3,5,7=1011
- Data at 2,3,6,7=1001
- Data at 4,5,6,7=1101
- Write Error :

P4	P2	P1
1	0	1
- 101=5, so error is in 5th bit that means 5th bit is 1, must be 0 to be corrected

Hamming distance

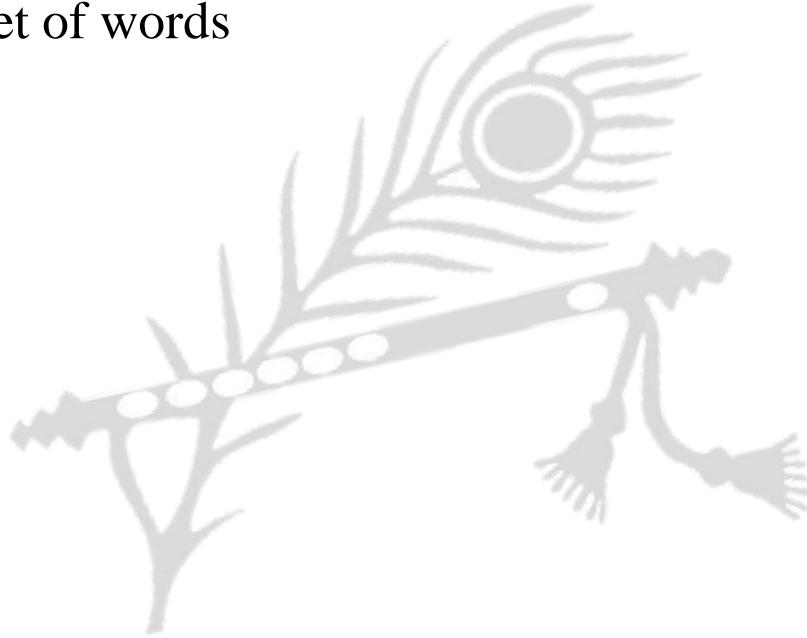
- The Hamming distance between two words is the number of differences between corresponding bits
- Let us find the Hamming distance between two pairs of words
 - The hamming distance $d(000, 011)$ is 2
 - The hamming distance $d(10101, 11110)$ is 3

$000 \oplus 011$ is 011 (two 1s)

$10101 \oplus 11110$ is 01011 (three 1s)

Hamming distance

- The minimum hamming distance is the smallest hamming distance between all possible pairs in a set of words



Hamming distance

- The minimum hamming distance is the smallest hamming distance between all possible pairs in a set of words
- Eg:
 - Find the minimum hamming distance



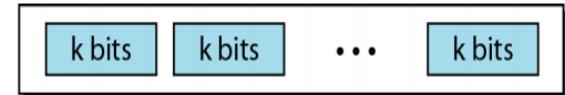
$$\begin{array}{lll}
 d(00000, 01011) = 3 & d(00000, 10101) = 3 & d(00000, 11110) = 4 \\
 d(01011, 10101) = 4 & d(01011, 11110) = 3 & d(10101, 11110) = 3
 \end{array}$$



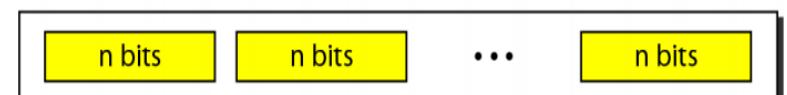
The d_{min} in this case is 3.

Block coding

- In block coding, we divide our message into blocks, each of k bits, called datawords
- We add r redundant bits to each block to make the length $n = k + r$
- The resulting n -bit blocks are called codewords
- Block coding is normally referred as mb/nb coding
- It replaces each m -bit group with n -bit group

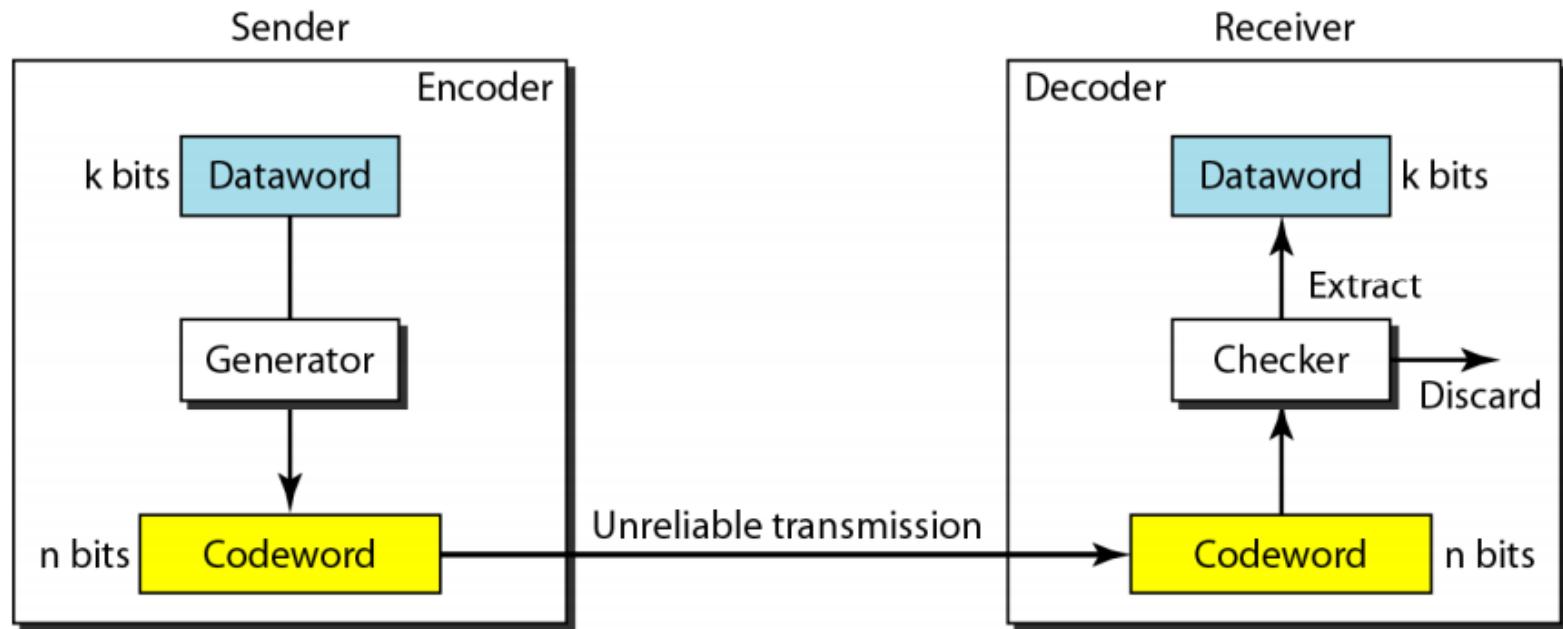


2^k Datawords, each of k bits



2^n Codewords, each of n bits (only 2^k of them are valid)

Process of error detection in block coding



$$R = \frac{n}{k} \quad R = \sqrt{1 + \frac{1}{k}}$$

Block coding

- Eg:
 - Let us assume that $k = 2$ and $n = 3$ list of datawords and codewords
 - Assume the sender encodes the dataword 01 as 011 and sends it to the receiver. Consider the following cases:
 - 1: The receiver receives 011. It is a valid codeword. The receiver extracts the dataword 01 from it
 - 2: The codeword is corrupted during transmission, and 111 is received. This is not a valid codeword and is discarded
 - 3: The codeword is corrupted during transmission, and 000 is received. This is a valid codeword. The receiver incorrectly extracts the dataword 00. Two corrupted bits have made the error undetectable

Block coding

- If we are having a block code of (4, 3) even parity

- Solution
- Information bit (k) = 3
- Codeword (n) = 4
- Now we will make table with following
 - 1: Information bit
 - 2: Parity bit
 - 3: Codeword

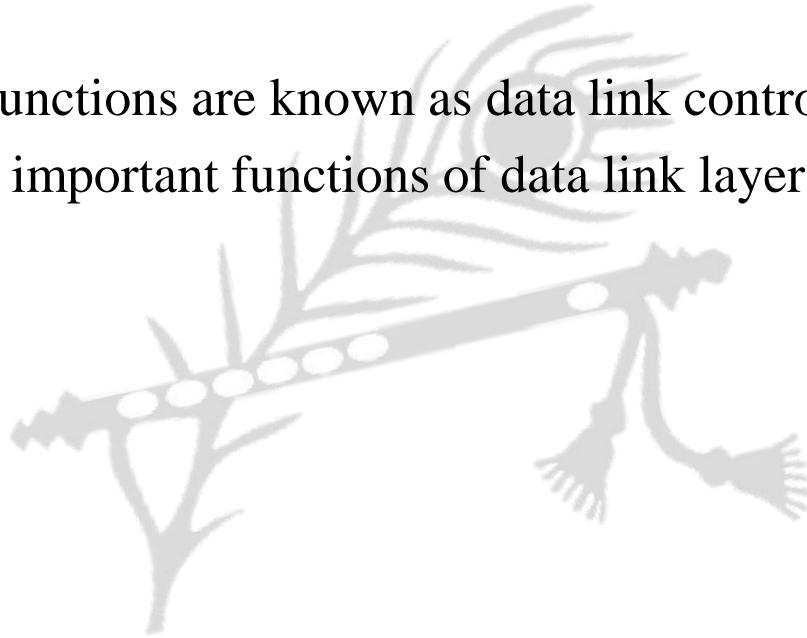
Block coding

Information bit	Parity Bit	Codeword
000-----	0	0000
-----1		0001
001-----	0	0010
-----1		0011
010-----	0	0100
-----1		0101
011-----	0	0110
-----1		0111
100-----	0	1000
-----1		1001
101-----	0	1010
-----1		1011
110-----	0	1100
-----1		1101
111-----	0	1110
-----1		1111

FLOW CONTROL

Flow Control

- The most important responsibilities of the data link layer are flow control and error control
- Collectively, these functions are known as data link control
- It is one of the most important functions of data link layer



Flow Control

- Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment
- Flow control coordinates the amount of data that can be sent before receiving acknowledgement
- Flow control is a set of procedures that tells the sender how much data it can transmit before it must wait for an acknowledgement from the receiver

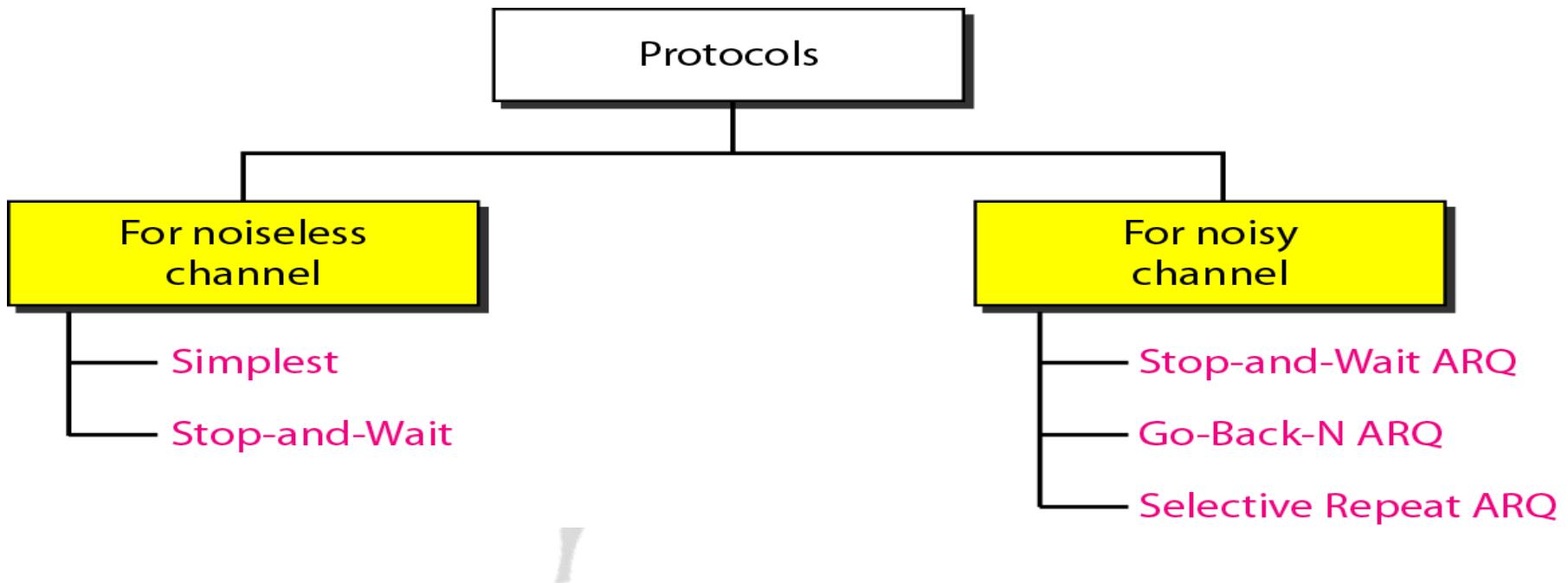
Flow Control

- Receiver has a limited speed at which it can process incoming data and a limited amount of memory in which to store incoming data
- Receiver must inform the sender before the limits are reached and request that the transmitter to send fewer frames or stop temporarily
- Since the rate of processing is often slower than the rate of transmission, receiver has a block of memory (buffer) for storing incoming data until they are processed

Flow Control

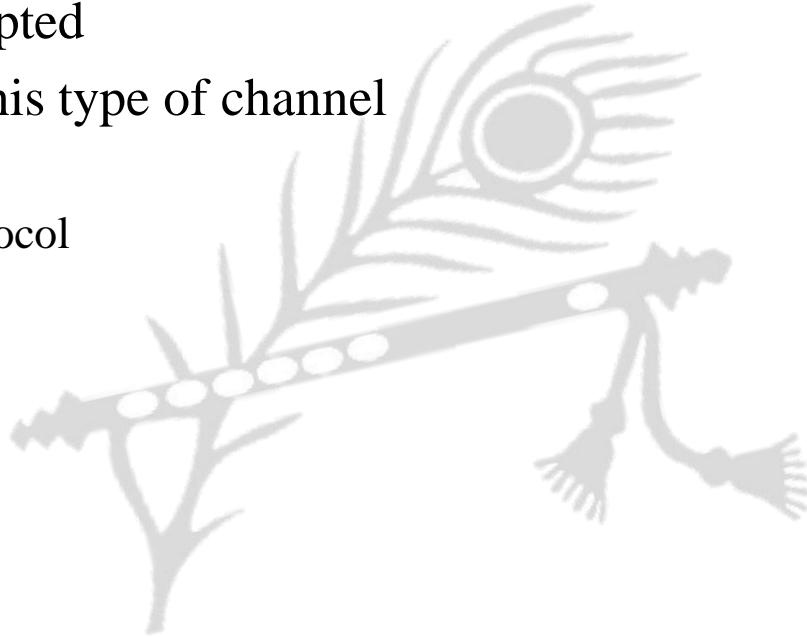
- Two Approaches:
 - Feedback based flow control
 - The receiver sends back information to the sender giving it permission to send more data or at least telling the sender how the receiver is doing
 - Rate based flow control
 - The protocol has a built in mechanism that limits the rate at which senders may transmit data, without using feedback from the receiver
- Various flow control schemes uses a common protocol that contains well-defined rules about when a sender may transmit the next frame
- These rules often prohibit frames from being sent until the receiver has granted permission, either implicitly or explicitly

Flow Control

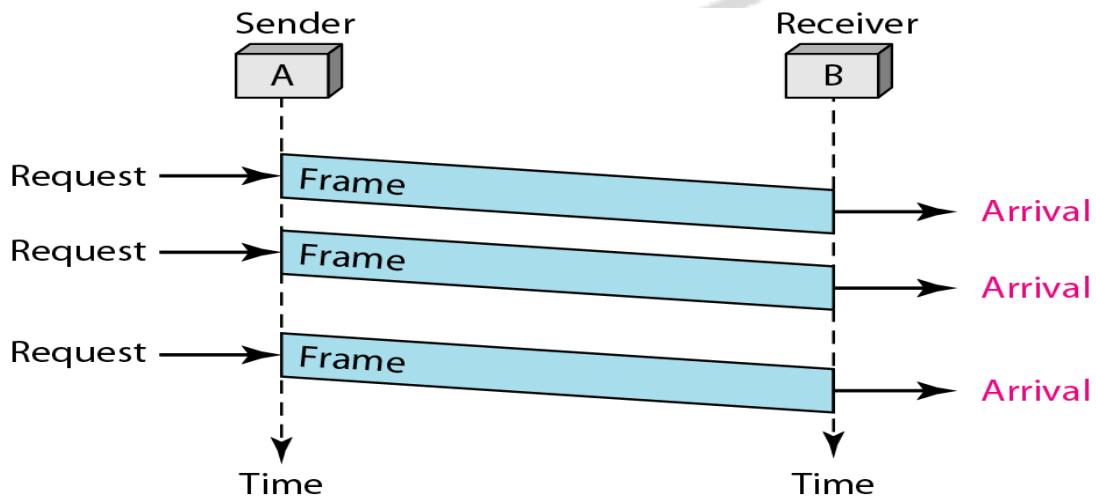


Noiseless channel

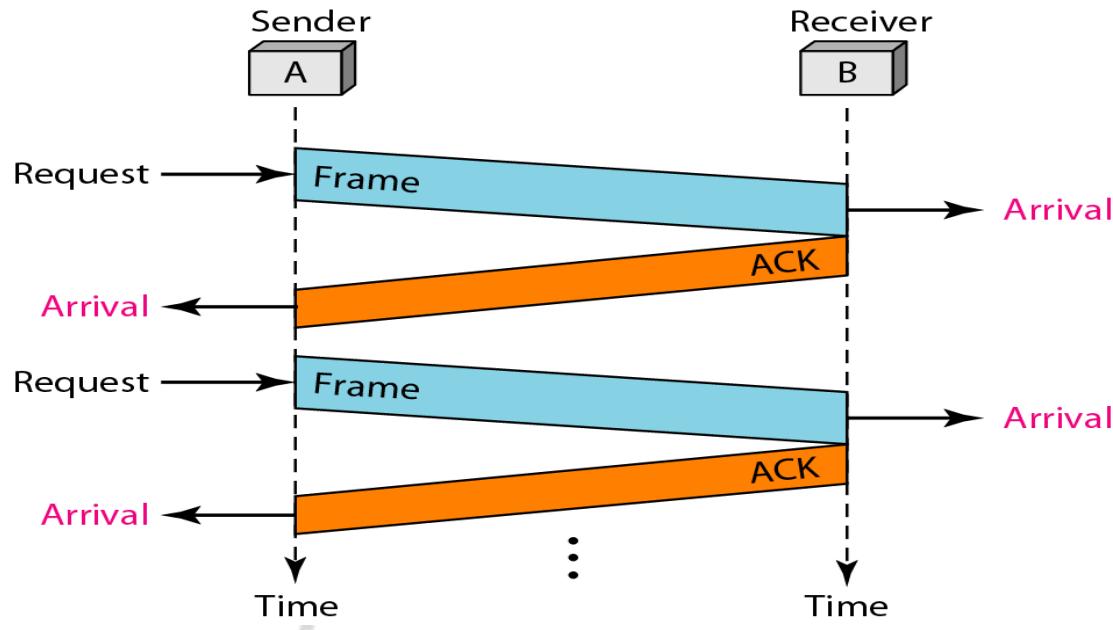
- Let us first assume we have an ideal channel in which no frames are lost, duplicated, or corrupted
- Two protocols for this type of channel
 - Simplest Protocol
 - Stop-and-Wait Protocol



Simplest Protocol



Simplest Protocol

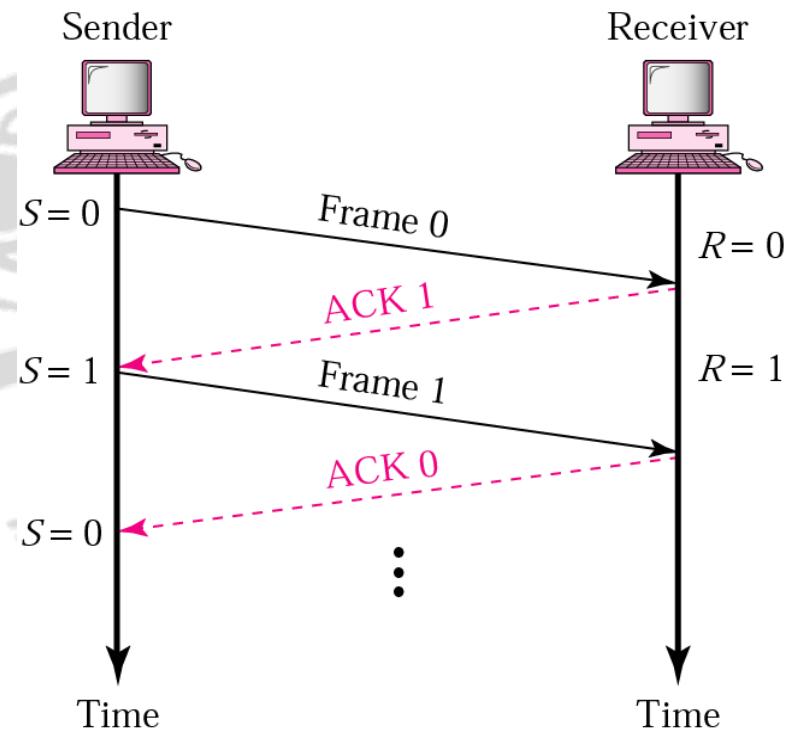


Noisy channel

- Three protocols
 - Stop-and-Wait Automatic Repeat Request
 - Go-Back-N Automatic Repeat Request
 - Selective Repeat Automatic Repeat Request
- NOTE:
- Round Trip Time (RTT)
 - Time taken by frame to travel from sender to receiver + Time taken by corresponding frame ACK to travel from receiver to sender
- Thus,
 - $RTT = 2 * T_p$ (Twice Propagation Time)

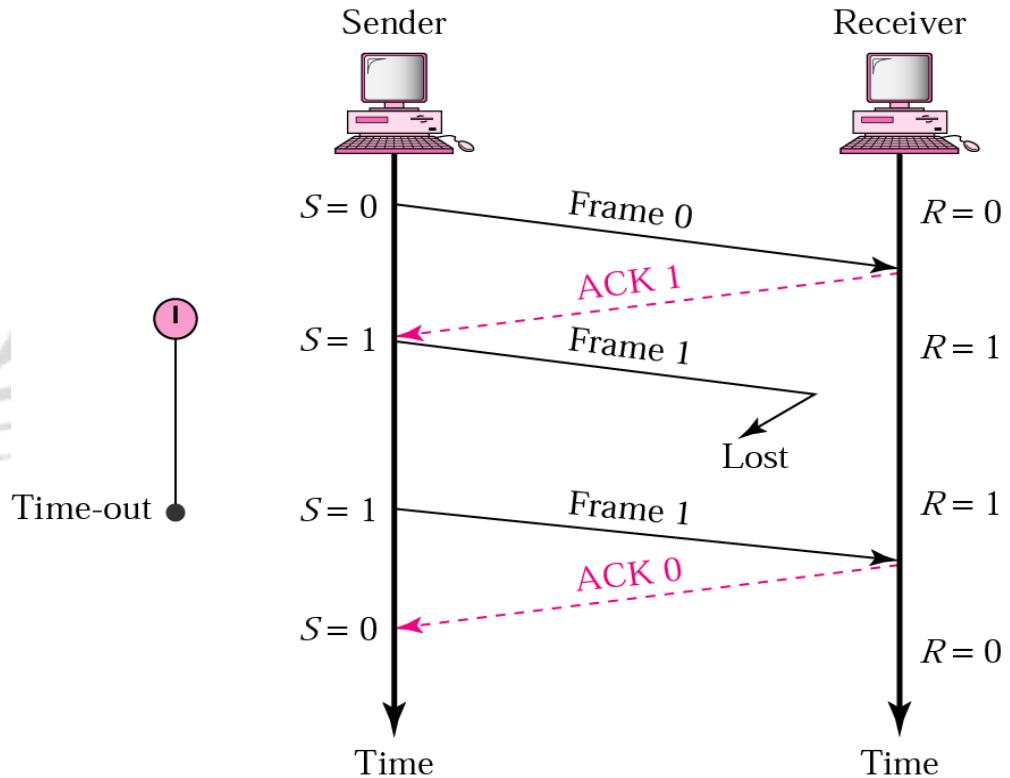
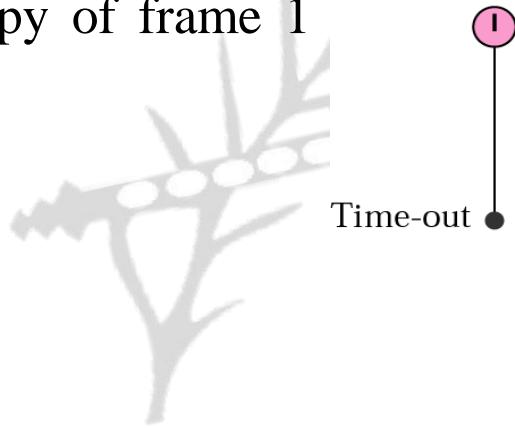
Stop and Wait ARQ (General Scenario)

- Sender keeps a copy of the last frame until it receives an acknowledgement
- For identification, data frame and acknowledgement (ACK) frames are numbered alternatively 0 and 1
- Sender has a control variable “S” that holds the number of the recently sent frame. (0 or 1)
- Receiver has a control variable “R” that holds the number of the next frame expected (0 or 1)
- Sender starts a timer when it sends a frame. If an ACK is not received within a allocated time period, the sender assumes that the frame was lost or damaged and resends it
- Receiver send only positive ACK if the frame is received
- ACK number always defines the number of the next expected frame



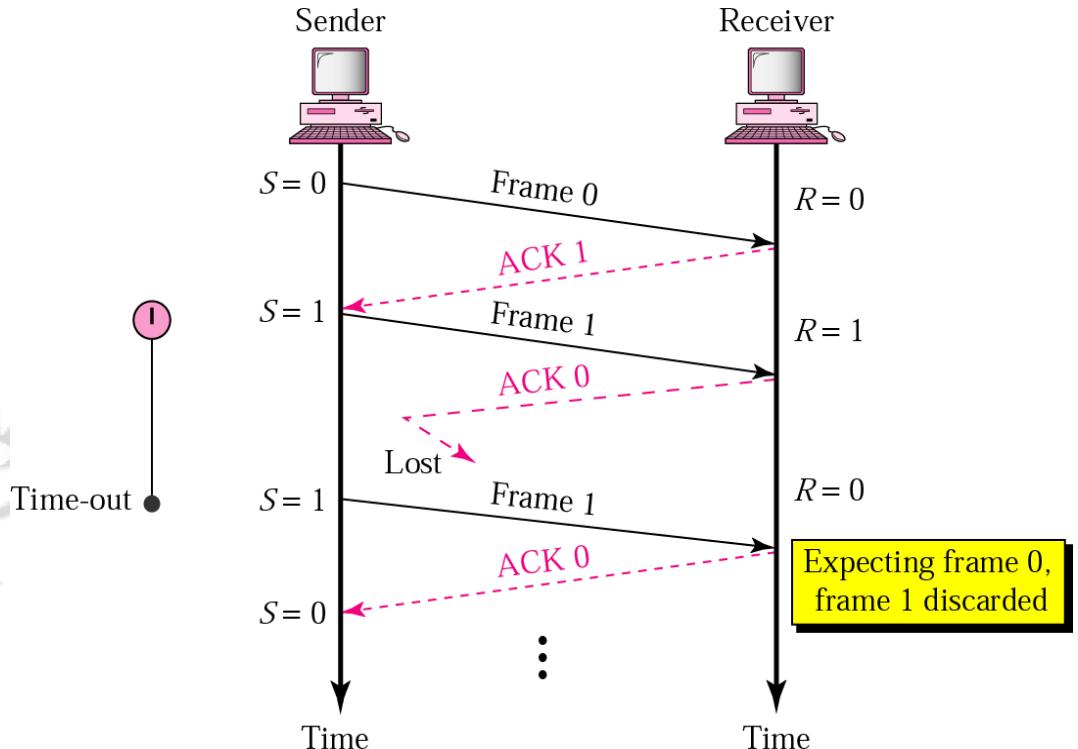
Stop and Wait ARQ (Damaged frame)

- When a receiver receives a damaged frame, it discards it and keeps its value of R
- After the timer at the sender expires, another copy of frame 1 is sent



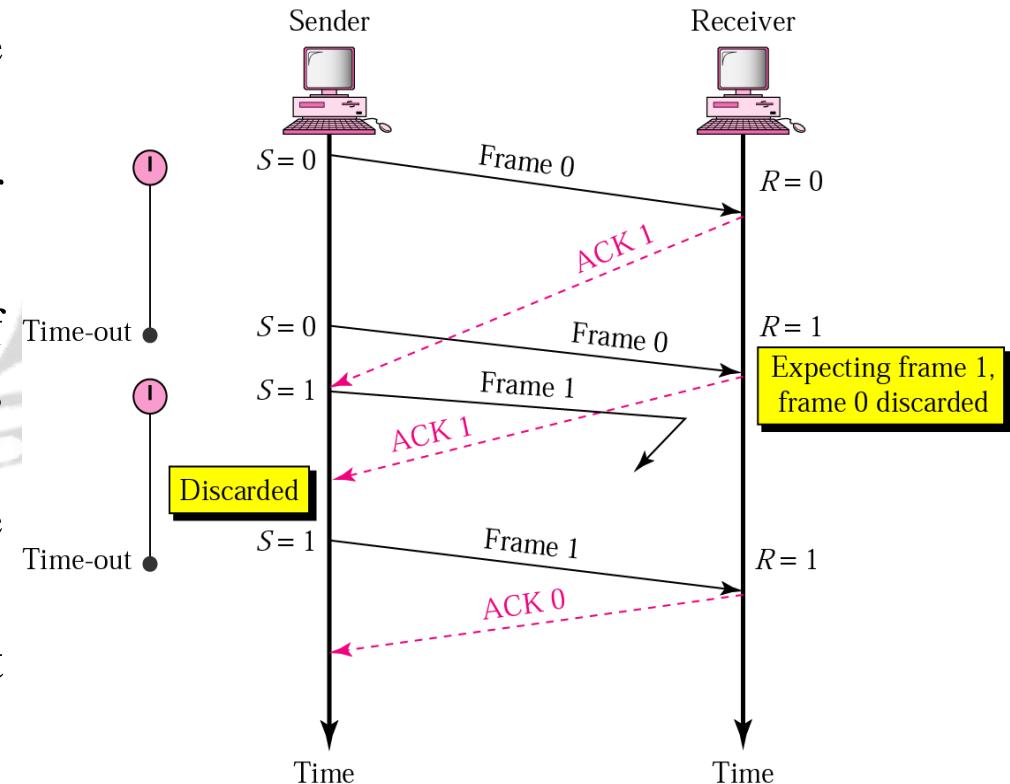
Stop and Wait ARQ (Damaged ACK)

- If the sender receives a damaged ACK, it discards it
- When the timer of the sender expires, the sender retransmits frame 1
- Receiver has already received frame 1 and expecting to receive frame 0 ($R=0$). Therefore it discards the second copy of frame 1



Stop and Wait ARQ (ACK delayed)

- The ACK can be delayed at the receiver or due to some problem
- It is received after the timer for frame 0 has expired
- Sender retransmitted a copy of frame 0. However, $R = 1$ means receiver expects to see frame 1. Receiver discards the duplicate frame 0
- Sender receives 2 ACKs, it discards the second ACK

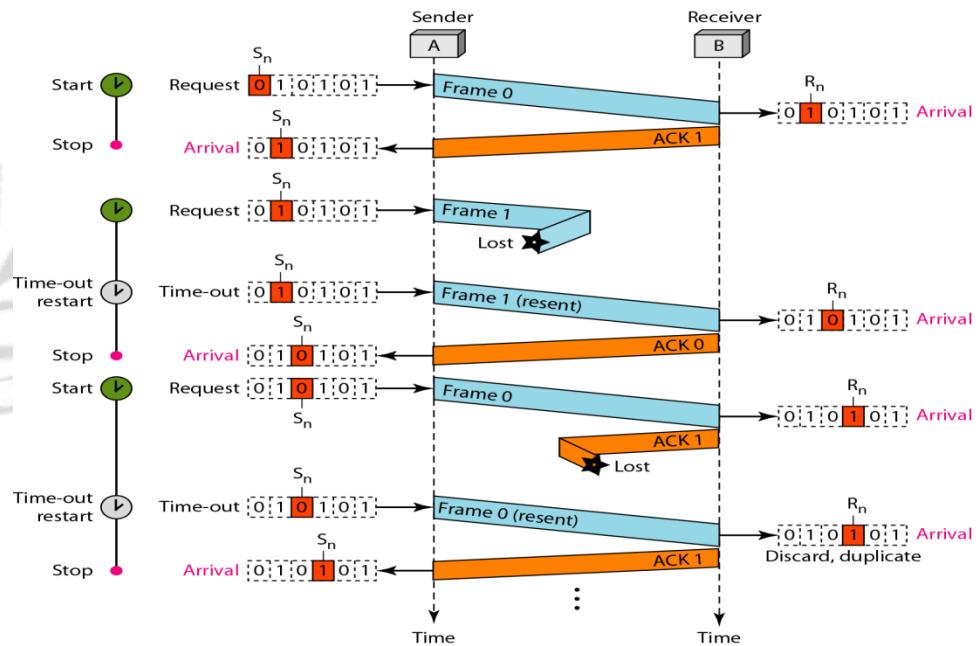


Stop and Wait ARQ

- Error correction in Stop-and-Wait ARQ is done by keeping a copy of the sent frame and retransmitting of the frame when the timer expires
- In Stop-and-Wait ARQ, we use sequence numbers to number the frames
- The sequence numbers are based on modulo-2 arithmetic
- In Stop-and-Wait ARQ, the acknowledgment number always announces in modulo-2 arithmetic the sequence number of the next frame expected

Stop and Wait ARQ

- Eg:
 - As seen from the figure, Frame 0 is sent and acknowledged
 - Frame 1 is lost and resent after the time-out
 - The resent frame 1 is acknowledged and the timer stops
 - Frame 0 is sent and acknowledged, but the acknowledgment is lost
 - The sender has no idea if the frame or the acknowledgment is lost, so after the time-out it resends frame 0, which is acknowledged



Stop and Wait ARQ

- Eg:
 - Assume that, in a Stop-and-Wait ARQ system, the bandwidth of the line is 1 Mbps, and 1 bit takes 20 ms to make a round trip. What is the bandwidth-delay product?

Solution

- The bandwidth-delay product is

$$(1 \times 10^6) \times (20 \times 10^{-3}) = 20,000 \text{ bits}$$

Stop and Wait ARQ

- Eg: (contd)
 - If the system data frames are 1000 bits in length, what is the utilization percentage of the link?

Solution

- The system can send 20,000 bits during the time it takes for the data to go from the sender to the receiver and then back again
- However, the system sends only 1000 bits
- We can say that the link utilization is only $1000/20,000$, or 5 percent
- For this reason, a link with a high bandwidth or long delay, the use of Stop-and-Wait ARQ wastes the capacity of the link

Stop and Wait ARQ

- Eg: (contd)
 - What is the utilization percentage of the link if we have a protocol that can send up to 15 frames before stopping and worrying about the acknowledgments?

Solution

- The bandwidth-delay product is still 20,000 bits
- The system can send up to 15 frames or 15,000 bits during a round trip
- This means the utilization is $15,000/20,000$, or 75 percent
- If there are damaged frames, the utilization percentage is much less because frames have to be resent

Stop and Wait ARQ

- Disadvantages
 - At any point in time, there is only one frame that is sent and waiting to be acknowledged
 - This is not a good use of transmission medium
 - Efficiency is very low
 - To improve efficiency, multiple frames should be in transition while waiting for ACK
- Two protocol use the above concept
 - Go back N ARQ
 - Selective repeat ARQ

Stop and Wait ARQ

Total time taken to send one packet,

$$= T_{t(data)} + T_p(data) + T_q + T_{pro} + T_{t(ack)} + T_p(ack)$$

Since,

$$T_p(ack) = T_p(data)$$

And,

$$T_{t(ack)} \ll T_{t(data)}.$$

So we can neglect $T_{t(ack)}$

$$T_q = 0 \text{ and } T_{pro} = 0$$

Hence,

$$\text{Total time} = T_{t(data)} + 2 * T_p$$

Stop and Wait ARQ

- Efficiency (η) = Useful time / Total cycle time

$$= T_t / (T_t + 2*T_p)$$

$$= 1 / (1+2*(T_p/T_t))$$

$$= 1 / (1+2*a)$$

where, $a = T_p / T_t$

- Throughput
 - Number of bits send per second, which is also known as Effective Bandwidth or Bandwidth utilization
 - Throughput = $\eta * BW$
- where, BW is Bandwidth

Stop and Wait ARQ

- Eg:
 - If the bandwidth of the line is 1.5 Mbps, RTT is 45 msec and packet size is 1 KB, then find the link utilization in stop and wait

Solution

$$\begin{aligned}\text{Transmission delay } (T_t) &= \text{Packet size} / \text{Bandwidth} \\ &= 1 \text{ KB} / 1.5 \text{ Mbps} \\ &= (2^{10} \times 8 \text{ bits}) / (1.5 \times 10^6 \text{ bits/sec}) \\ &= 5.461 \text{ msec}\end{aligned}$$

Stop and Wait ARQ

$$\begin{aligned}\text{Propagation delay (Tp)} &= \text{Round Trip Time} / 2 \\ &= 45 \text{ msec}/2 \\ &= 22.5 \text{ msec}\end{aligned}$$

Calculating Value Of 'a'

$$\begin{aligned}a &= Tp / Tt \\ a &= 22.5 / 5.461 \\ a &= 4.12 \text{ msec}\end{aligned}$$

$$\begin{aligned}\text{Link Utilization or Efficiency } (\eta) &= 1 / (1 + 2a) \\ &= 1 / (1 + 2 \times 4.12) \\ &= 1 / 9.24 \\ &= 0.108 \\ &= 10.8 \%\end{aligned}$$

Stop and Wait ARQ

- Eg:
 - A channel has a bit rate of 4 Kbps and one way propagation delay of 20 msec. The channel uses stop and wait protocol. The transmission time of the acknowledgement frame is negligible. To get a channel efficiency of at least 50%, the minimum frame size should be
 - 80 bytes
 - 80 bits
 - 160 bytes
 - 160 bits

Solution

Bandwidth = 4 Kbps, Propagation delay (T_p) = 20 msec, Efficiency $\geq 50\%$

Let the required frame size = L bits

$$\begin{aligned}
 \text{Transmission delay (T}_t\text{)} &= \text{Packet size / Bandwidth} \\
 &= L \text{ bits / 4 Kbps}
 \end{aligned}$$

Stop and Wait ARQ

Condition for efficiency to be at least 50 %

For efficiency to be at least 50%, we must have

$$\begin{aligned} 1 / (1+2a) &\geq 1/2 \\ a &\leq 1/2 \end{aligned}$$

Substituting the value of ‘a’, we get

$$(20 \text{ msec} \times 4 \text{ Kbps}) / L \text{ bits} \leq 1/2$$

$$L \text{ bits} \geq (20 \text{ msec} \times 4 \text{ Kbps}) \times 2$$

$$L \text{ bits} \geq (20 \times 10^{-3} \text{ sec} \times 4 \times 10^3 \text{ bits per sec}) \times 2$$

$$L \text{ bits} \geq 20 \times 4 \text{ bits} \times 2$$

$$L \geq 160$$

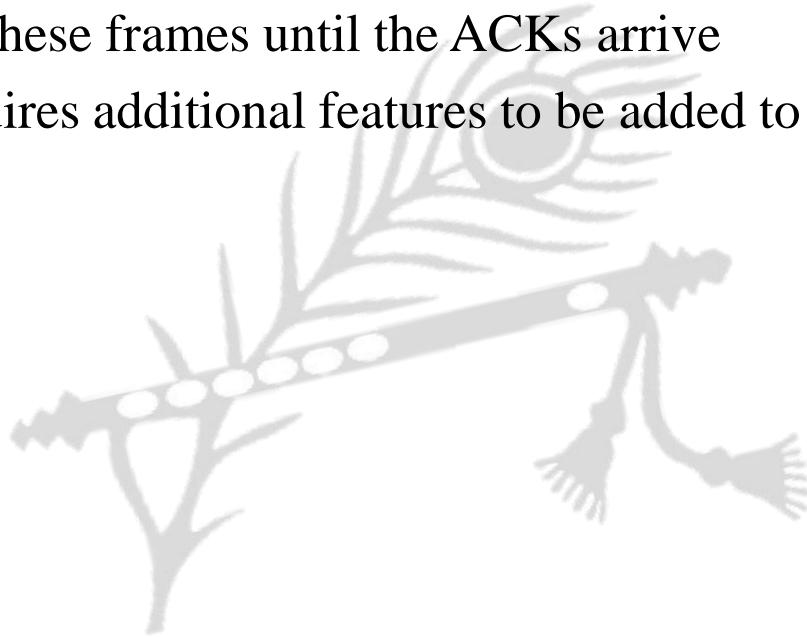
From here, frame size must be at least 160 bits

Disadvantage of Stop and Wait

- In stop-and-wait, at any point in time, there is only one frame that is sent and waiting to be acknowledged
- This is not a good use of transmission medium
- To improve efficiency, multiple frames should be in transition while waiting for ACK
- Two protocol use the above concept,
 - Go-Back-N ARQ
 - Selective Repeat ARQ

Go Back N ARQ

- We can send up to W frames before worrying about ACKs
- We keep a copy of these frames until the ACKs arrive
- This procedure requires additional features to be added to Stop and Wait ARQ

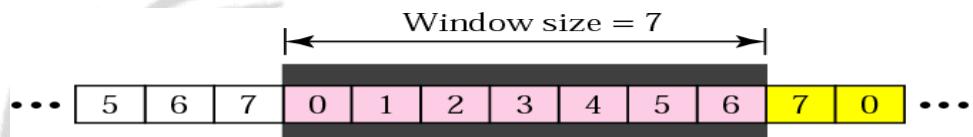


Sequence Numbers

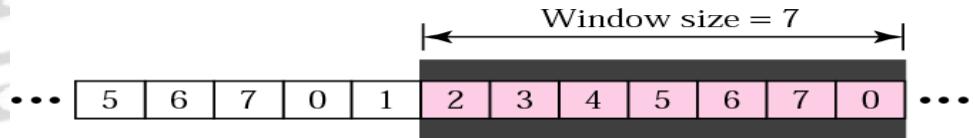
- Frames from a sender are numbered sequentially
- We need to set a limit since we need to include the sequence number of each frame in the header
- If the header of the frame allows m bits for sequence number, the sequence numbers range from 0 to $2^m - 1$
- For $m = 3$, sequence numbers are: 1, 2, 3, 4, 5, 6, 7
- We can repeat the sequence number
- Sequence numbers are 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, ...

Sender Sliding Window

- At the sending site, to hold the outstanding frames until they are acknowledged, we use the concept of a window
- The size of the window is at most $2^m - 1$, where m is the number of bits for the sequence number
- Size of the window can be variable, e.g. TCP
- The window slides to include new unsent frames when the correct ACKs are received



a. Before sliding



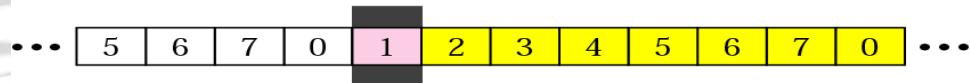
b. After sliding two frames

Receiver Sliding Window

- Size of the window at the receiving site is always 1 in this protocol
- Receiver is always looking for a specific frame to arrive in a specific order
- Any frame arriving out of order is discarded and needs to be resent.
- Receiver window slides as shown in fig. Receiver is waiting for frame 0 in part a



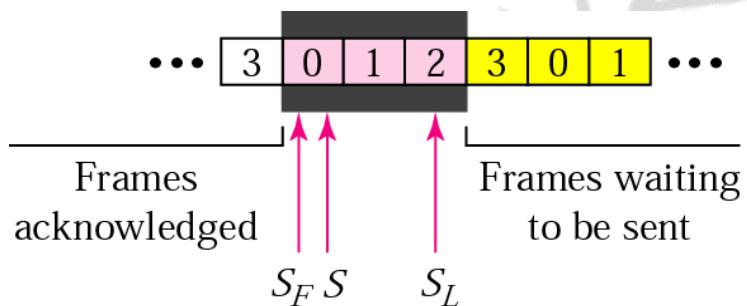
a. Before sliding



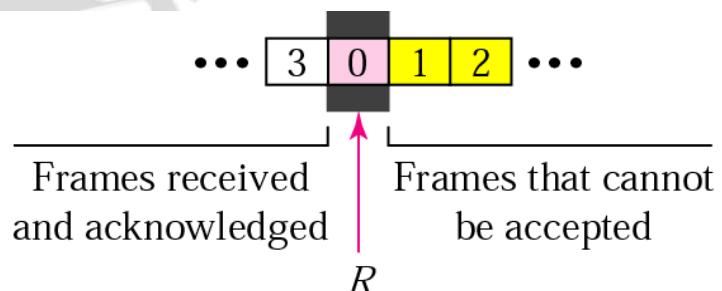
b. After sliding

Control Variables

- Sender has 3 variables: S, SF, and SL
- S holds the sequence number of recently sent frame
- SF holds the sequence number of the first frame
- SL holds the sequence number of the last frame
- Receiver only has the one variable, R, that holds the sequence number of the frame it expects to receive. If the seq. no. is the same as the value of R, the frame is accepted, otherwise rejected



a. Sender window



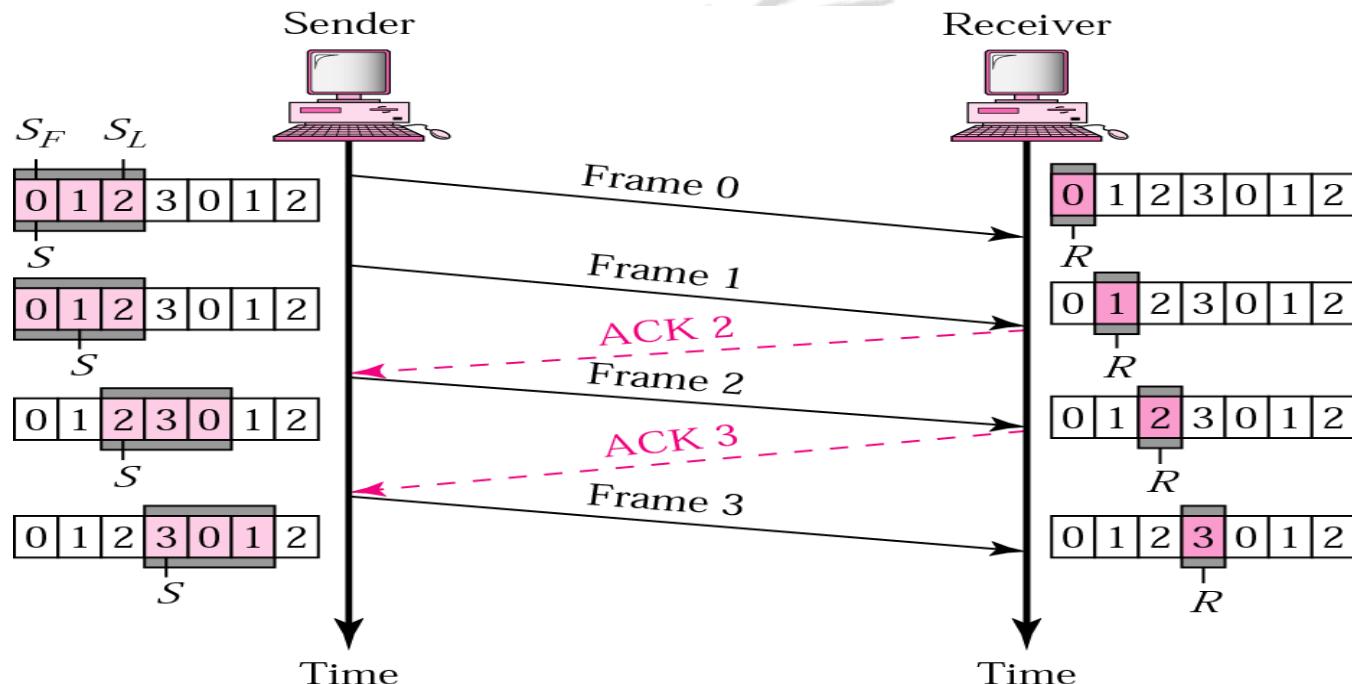
b. Receiver window

Acknowledgement

- Receiver sends positive ACK if a frame arrived safe and in order
- If the frames are damaged/out of order, receiver is silent and discard all subsequent frames until it receives the one it is expecting
- The silence of the receiver causes the timer of the unacknowledged frame to expire
- Then the sender resends all frames, beginning with the one with the expired timer
- For example, suppose the sender has sent frame 6, but the timer for frame 3 expires (i.e. frame 3 has not been acknowledged), then the sender goes back and sends frames 3, 4, 5, 6 again. Thus it is called Go-Back-N-ARQ
- The receiver does not have to acknowledge each frame received, it can send one cumulative ACK for several frames

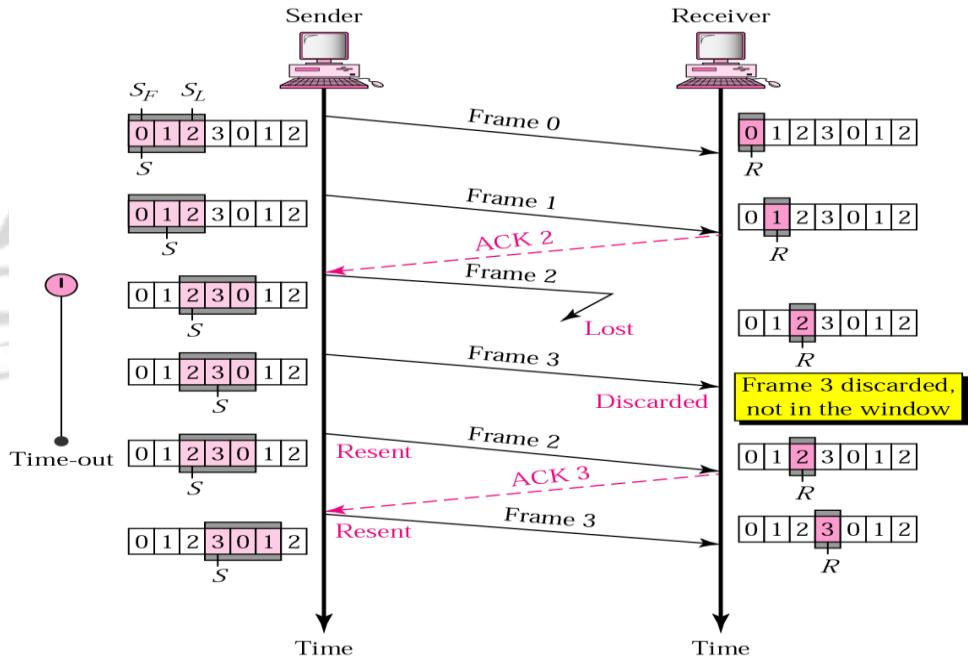
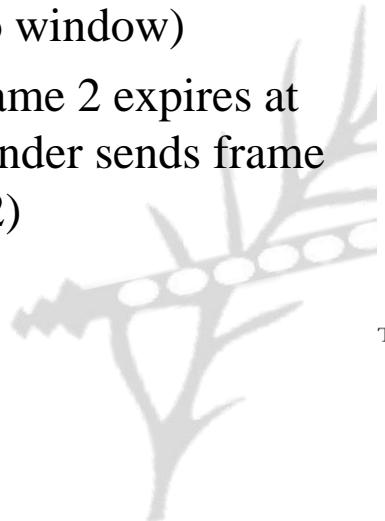
Go Back N ARQ, normal operation

- The sender keeps track of the outstanding frames and updates the variables and windows as the ACKs arrive



Go Back N ARQ, lost frame

- Frame 2 is lost
- When the receiver receives frame 3, it discards frame 3 as it is expecting frame 2 (according to window)
- After the timer for frame 2 expires at the sender site, the sender sends frame 2 and 3. (go back to 2)

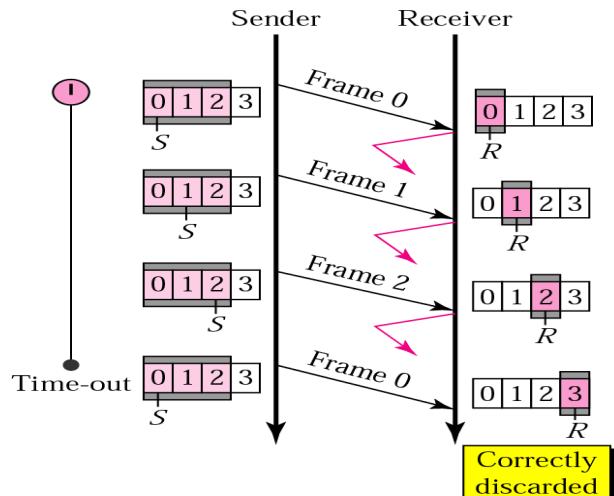


Go Back N ARQ, damaged/lost/delayed ACK

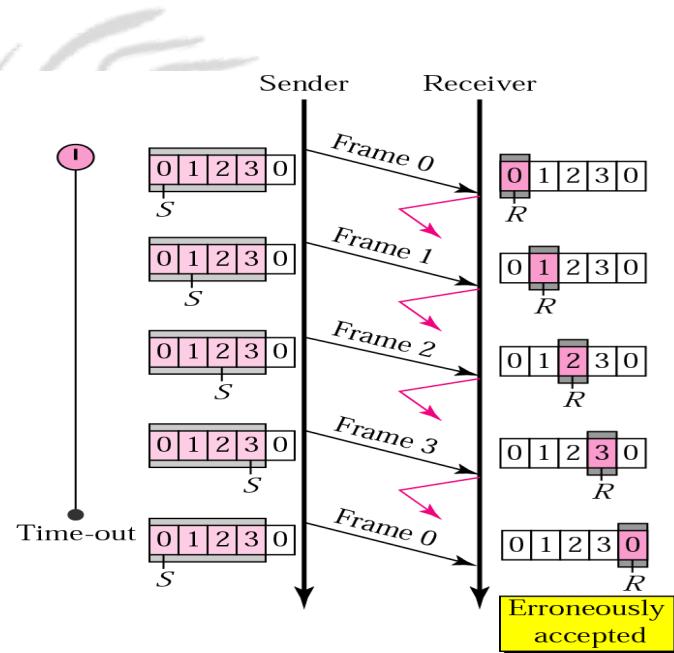
- If an ACK is damaged/lost, we can have two situations
- If the next ACK arrives before the expiration of any timer, there is no need for retransmission of frames because ACKs are cumulative in this protocol
- If ACK1, ACK2, and ACK3 are lost, ACK4 covers them if it arrives before the timer expires
- If ACK4 arrives after time-out, the last frame and all the frames after that are resent
- Receiver never resends an ACK
- A delayed ACK also triggers the resending of frames

Go Back N ARQ, sender window size

- Size of the sender window must be less than 2^m
- Size of the receiver is always 1

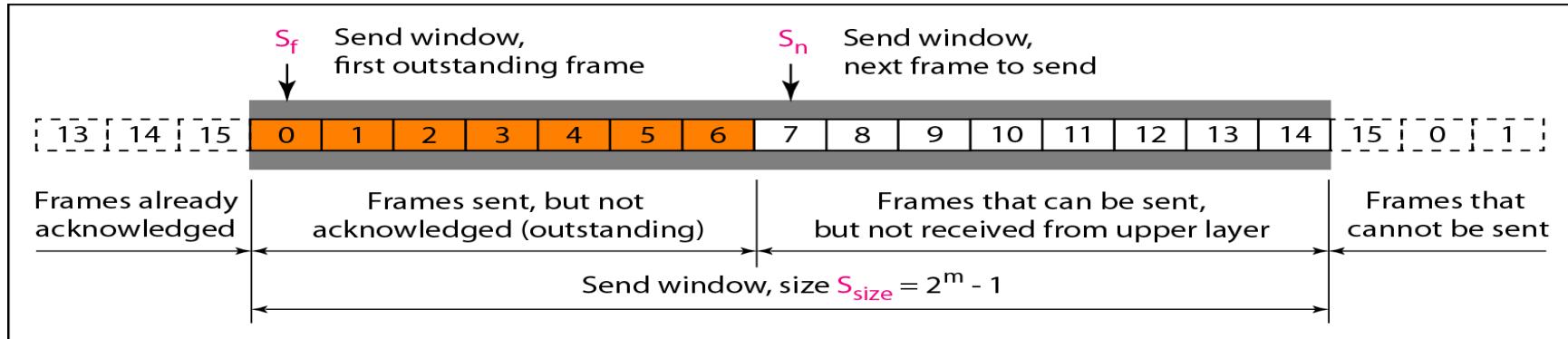


a. Window size $< 2^m$

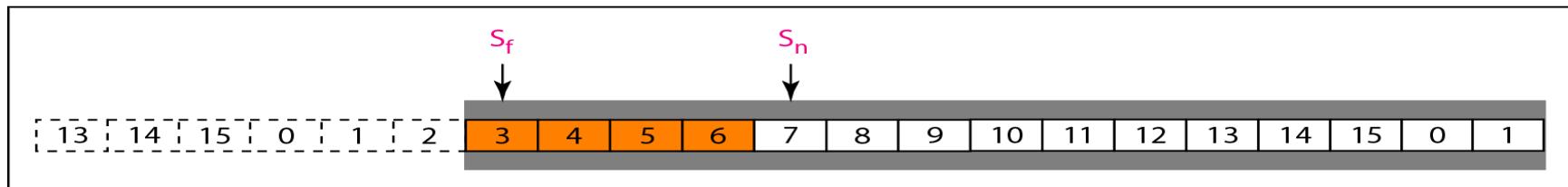


b. Window size $= 2^m$

Go Back N ARQ (sending window)

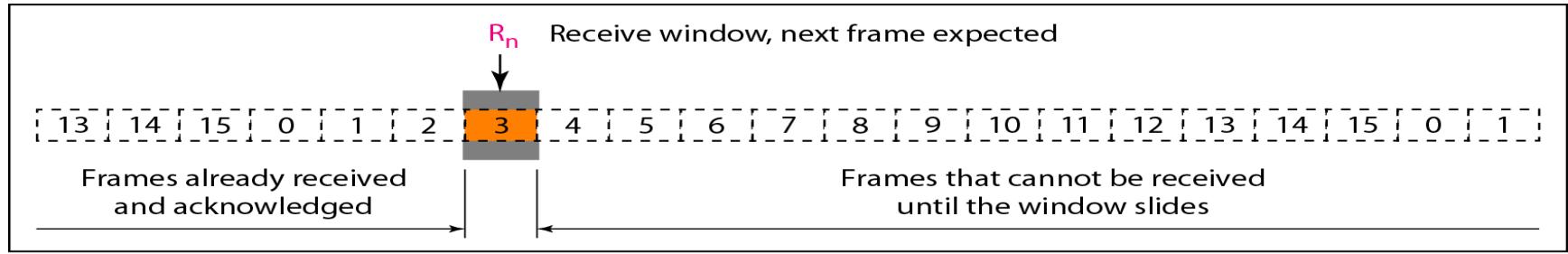


a. Send window before sliding

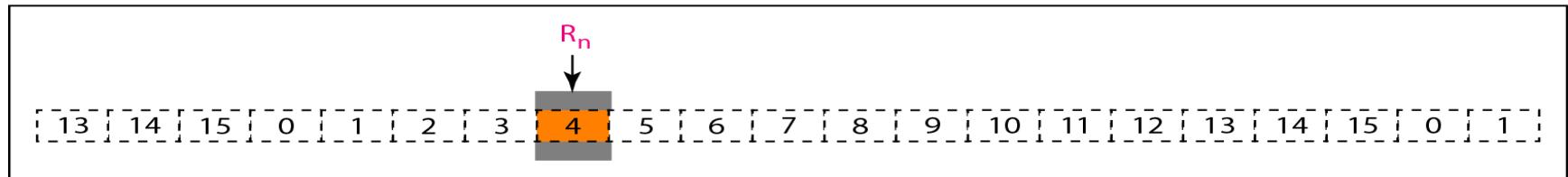


b. Send window after sliding

Go Back N ARQ (receiving window)



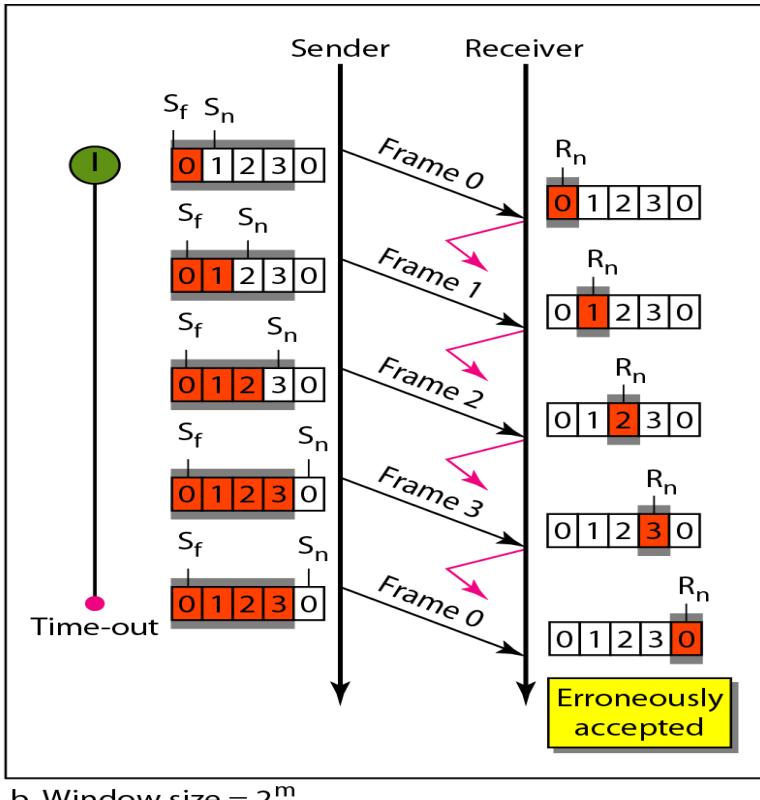
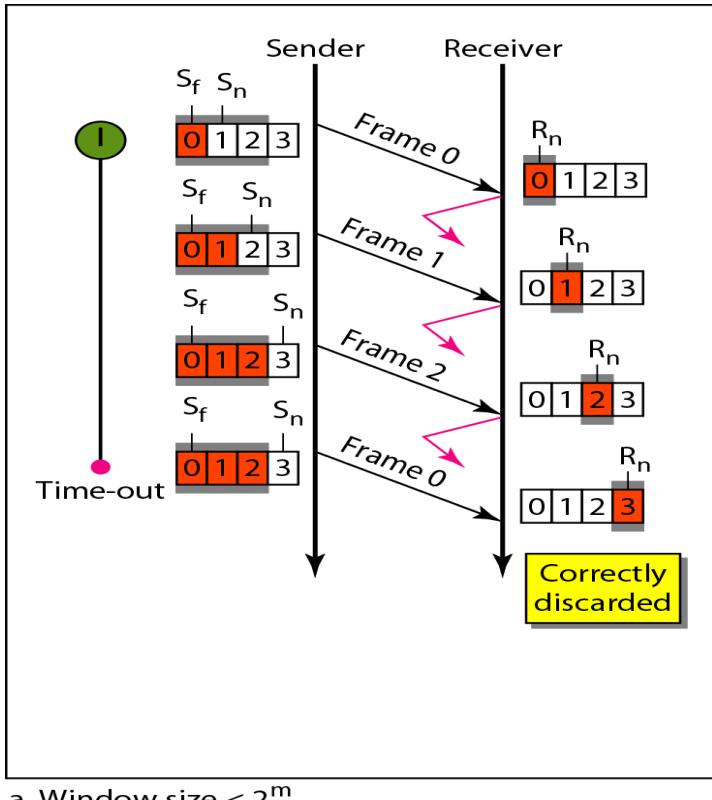
a. Receive window



b. Window after sliding



Go Back N ARQ (sending and receiving window)

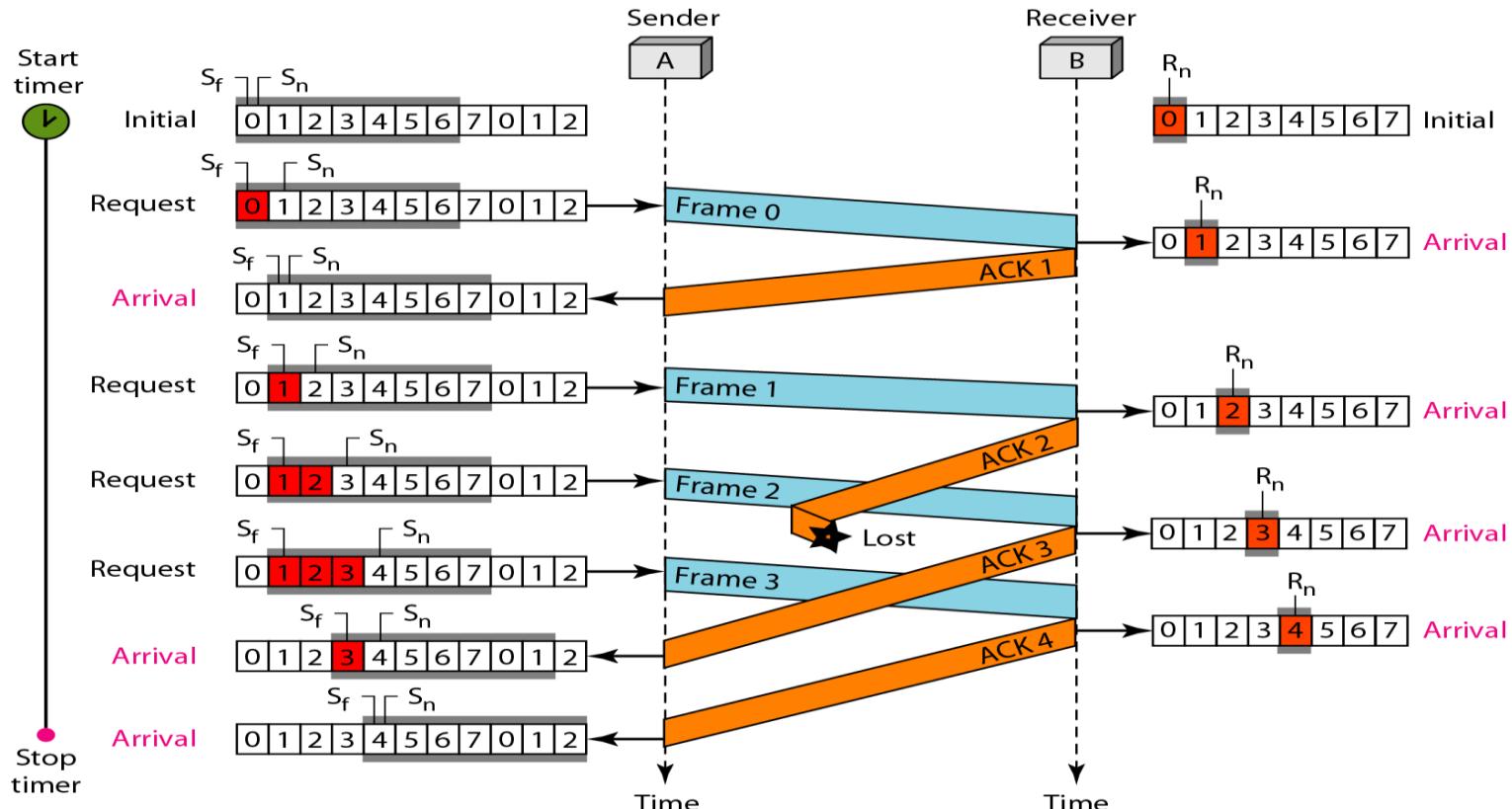


Go Back N ARQ

- Eg:

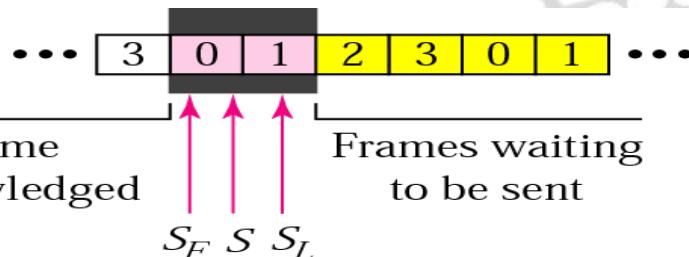
- Figure shows an example of Go Back N. This is an example of a case where the forward channel is reliable, but the reverse is not. No data frames are lost, but some ACKs are delayed and one is lost. The example also shows how cumulative acknowledgments can help if acknowledgments are delayed or lost. After initialization, there are seven sender events. Request events are triggered by data from the network layer; arrival events are triggered by acknowledgments from the physical layer. There is no time-out event here because all outstanding frames are acknowledged before the timer expires. Note that although ACK 2 is lost, ACK 3 serves as both ACK 2 and ACK 3.

Go Back N ARQ

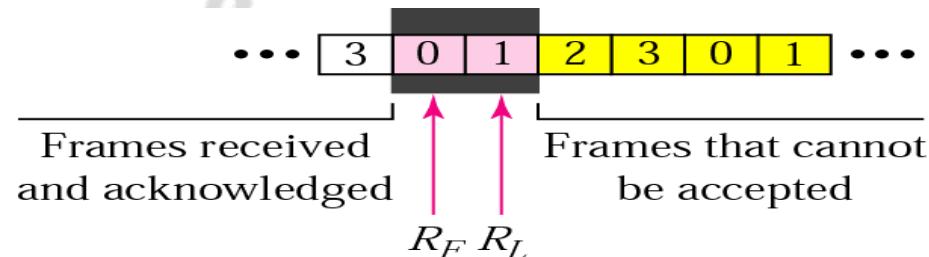


Selective Repeat ARQ, sender and receiver windows

- Go Back N ARQ simplifies the process at the receiver site. Receiver only keeps track of only one variable, and there is no need to buffer out-of-order frames, they are simply discarded. However, Go Back N ARQ protocol is inefficient for noisy link
- In Selective Repeat ARQ, only the damaged frame is resent. More bandwidth efficient but more complex processing at receiver
- It defines a negative ACK (NAK) to report the sequence number of a damaged frame before the timer expires



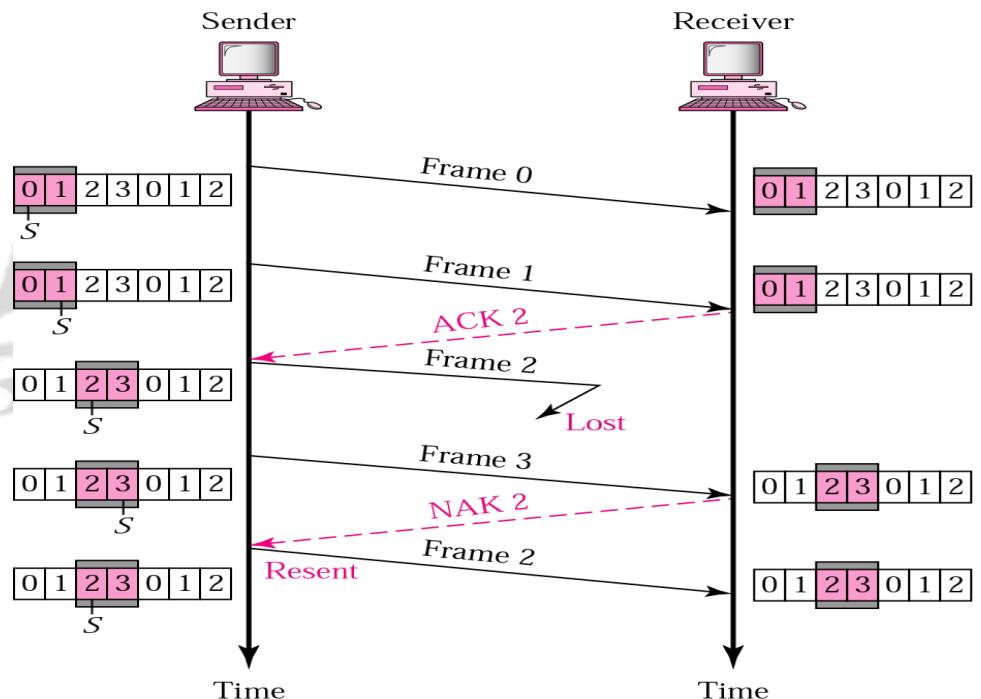
a. Sender window



b. Receiver window

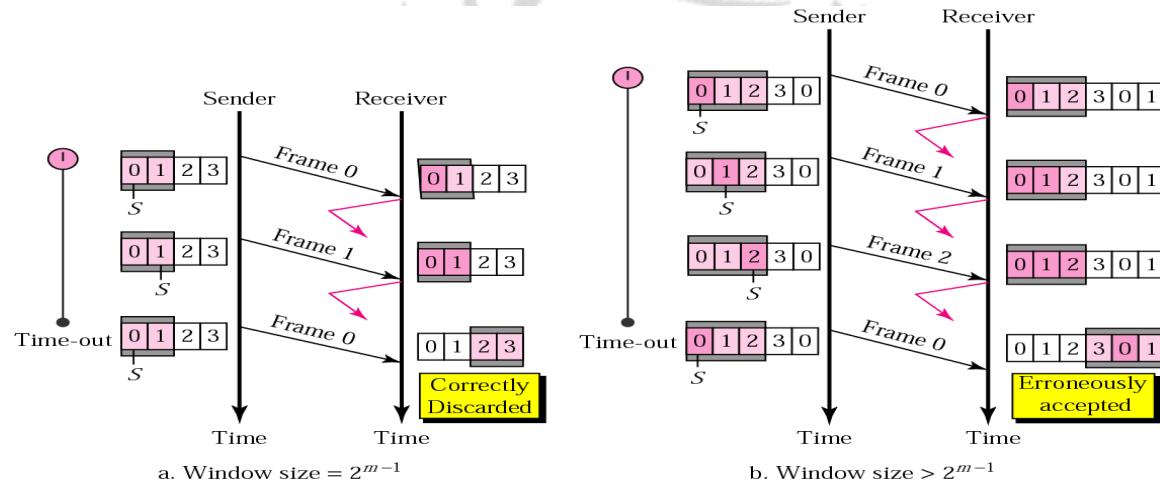
Selective Repeat ARQ, lost frame

- Frames 0 and 1 are accepted when received because they are in the range specified by the receiver window. Same for frame 3
- Receiver sends a NAK2 to show that frame 2 has not been received and then sender resends only frame 2 and it is accepted as it is in the range of the window

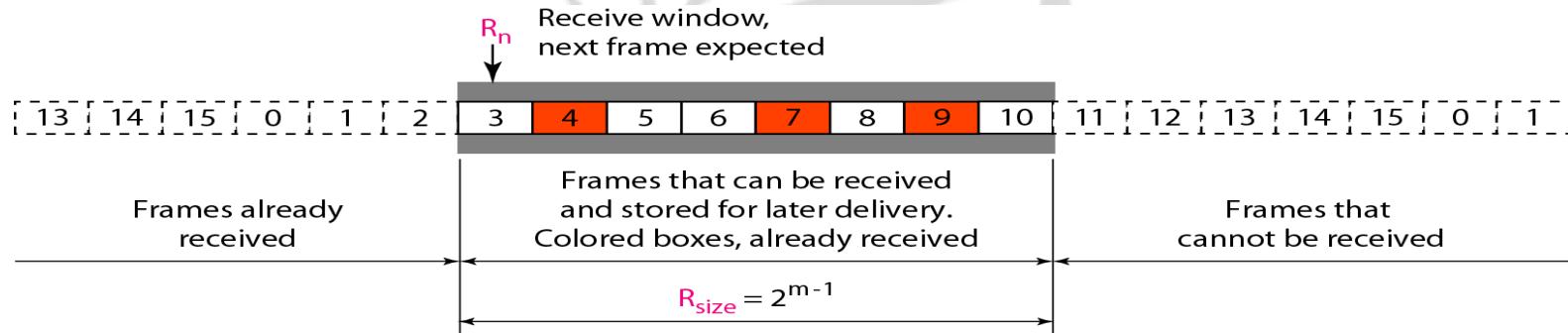
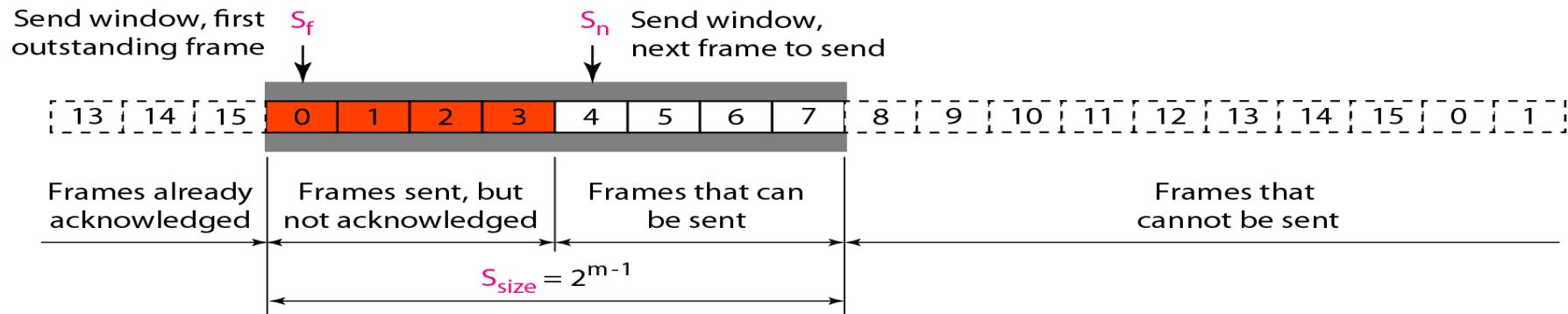


Selective Repeat ARQ, sender window size

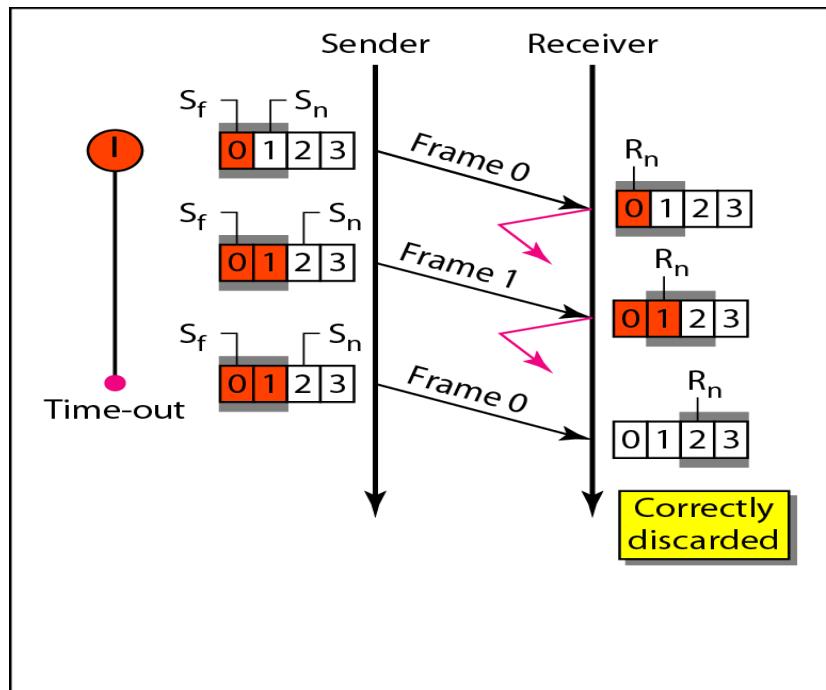
- Size of the sender and receiver windows must be at most one-half of 2^m . If $m = 2$, window size should be $2^m/2 = 2$
- Fig compares a window size of 2 with a window size of 3. Window size is 3 and all ACKs are lost, sender sends duplicate of frame 0, window of the receiver expect to receive frame 0, so accepts frame 0, as the 1st frame of the next cycle – an error



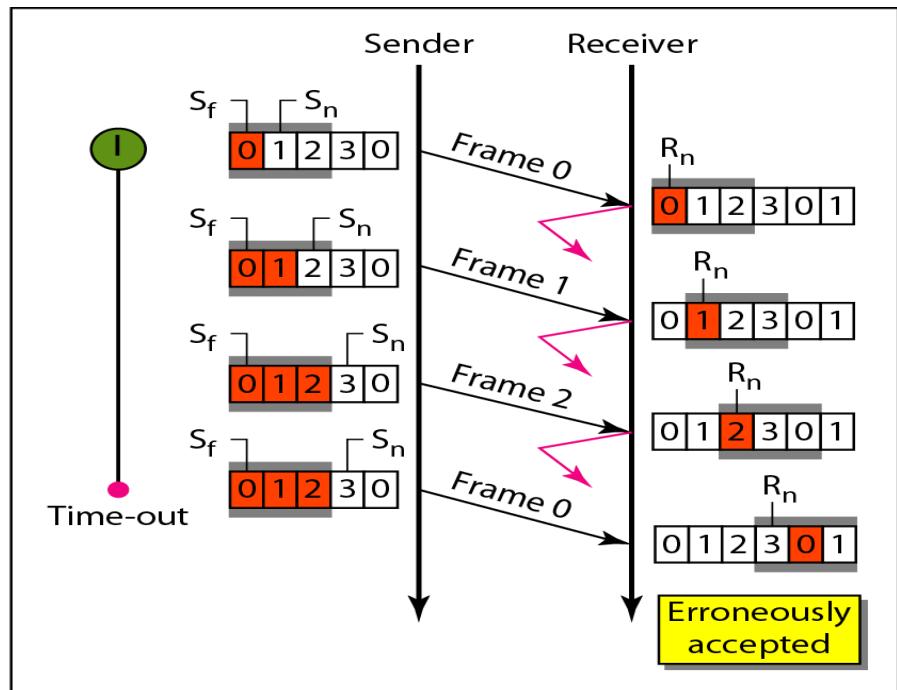
Window for Selective Repeat ARQ



Window for Selective Repeat ARQ

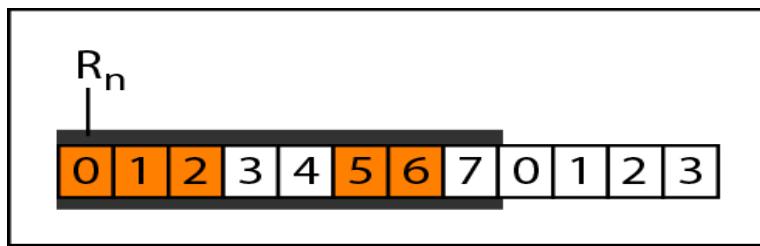


a. Window size = 2^{m-1}

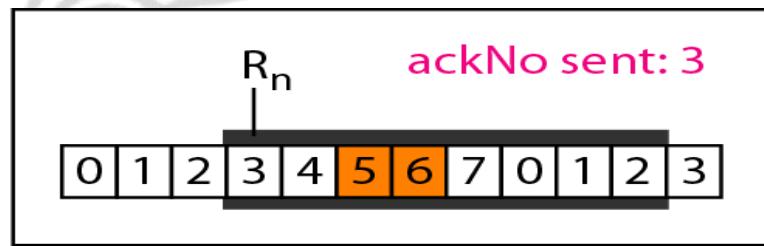


b. Window size $> 2^{m-1}$

Delivery of data in Selective Repeat ARQ



a. Before delivery



b. After delivery

Selective Repeat ARQ

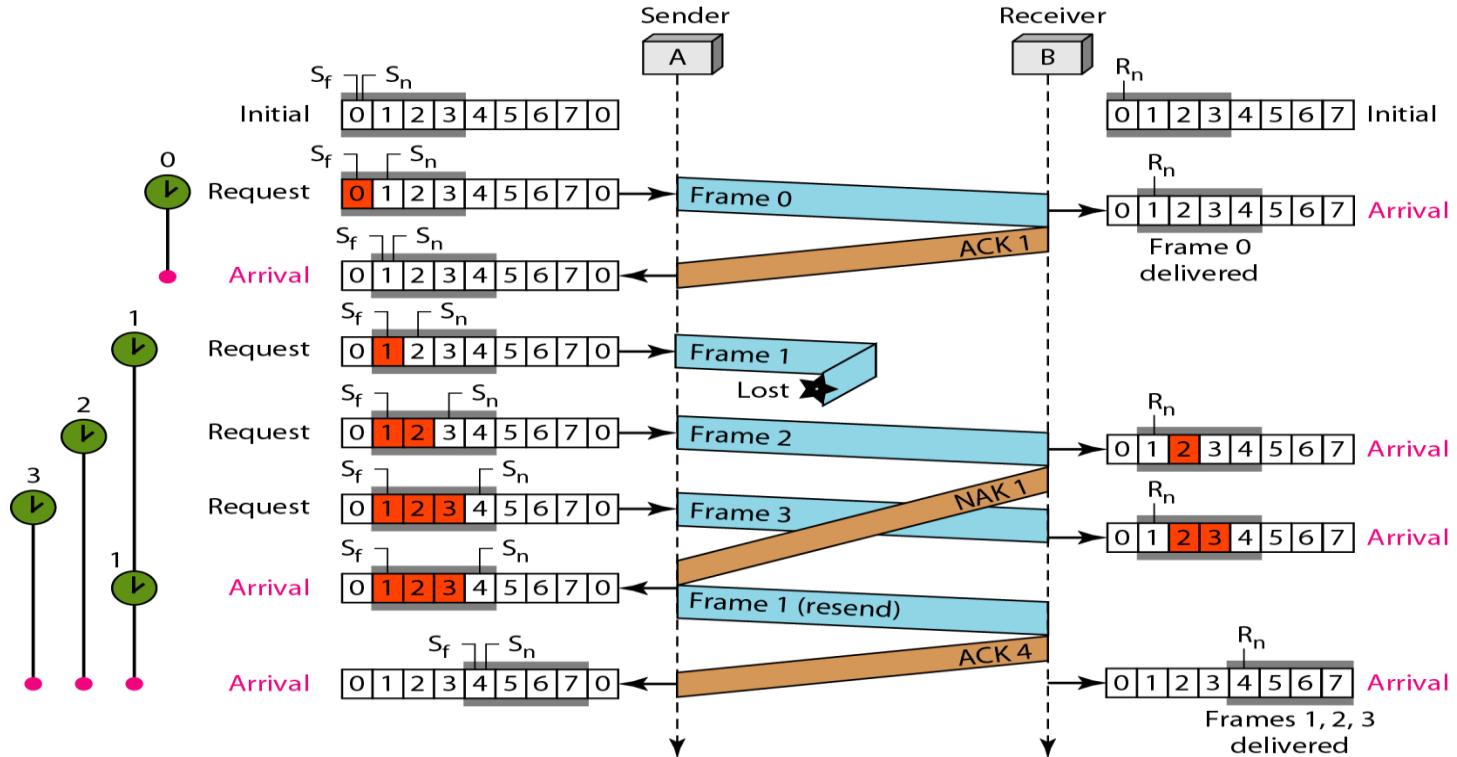
- Eg:

- This example is similar to previous example in which frame 1 is lost. We show how Selective Repeat behaves in this case. Figure 11.23 shows the situation. One main difference is the number of timers. Here, each frame sent or resent needs a timer, which means that the timers need to be numbered (0, 1, 2, and 3). The timer for frame 0 starts at the first request, but stops when the ACK for this frame arrives. The timer for frame 1 starts at the second request, restarts when a NAK arrives, and finally stops when the last ACK arrives. The other two timers start when the corresponding frames are sent and stop at the last arrival event.
- At the receiver site we need to distinguish between the acceptance of a frame and its delivery to the network layer. At the second arrival, frame 2 arrives and is stored and marked, but it cannot be delivered because frame 1 is missing. At the next arrival, frame 3 arrives and is marked and stored, but still none of the frames can be delivered. Only at the last arrival, when finally a copy of frame 1 arrives, can frames 1, 2, and 3 be delivered to the network layer. There are two conditions for the delivery of frames to the network layer: First, a set of consecutive frames must have arrived. Second, the set starts from the beginning of the window.

Selective Repeat ARQ

- Eg: (contd...)
 - Another important point is that a NAK is sent after the second arrival, but not after the third, although both situations look the same. The reason is that the protocol does not want to crowd the network with unnecessary NAKs and unnecessary resent frames. The second NAK would still be NAK1 to inform the sender to resend frame 1 again; this has already been done. The first NAK sent is remembered (using the nakSent variable) and is not sent again until the frame slides. A NAK is sent once for each window position and defines the first slot in the window.
 - The next point is about the ACKs. Notice that only two ACKs are sent here. The first one acknowledges only the first frame; the second one acknowledges three frames. In Selective Repeat, ACKs are sent when data are delivered to the network layer. If the data belonging to n frames are delivered in one shot, only one ACK is sent for all of them.

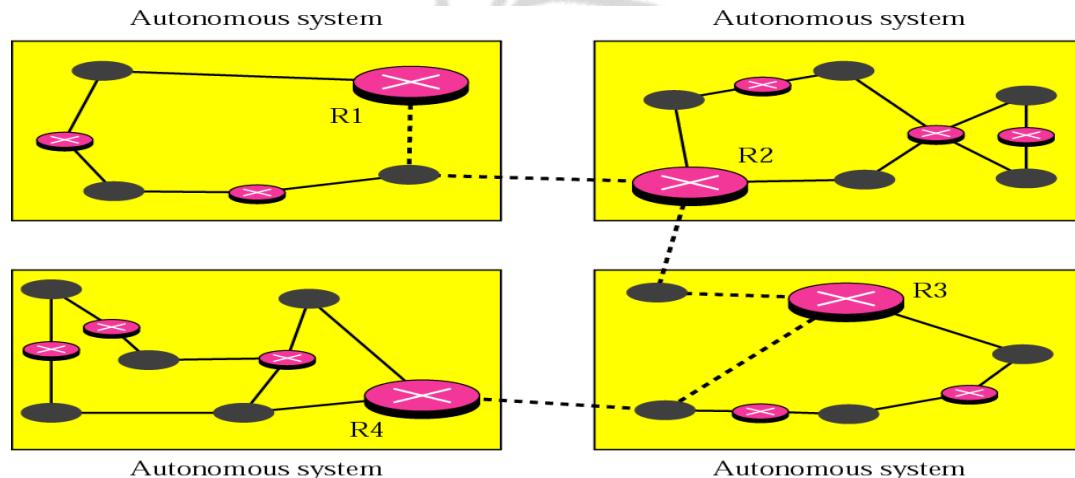
Selective Repeat ARQ



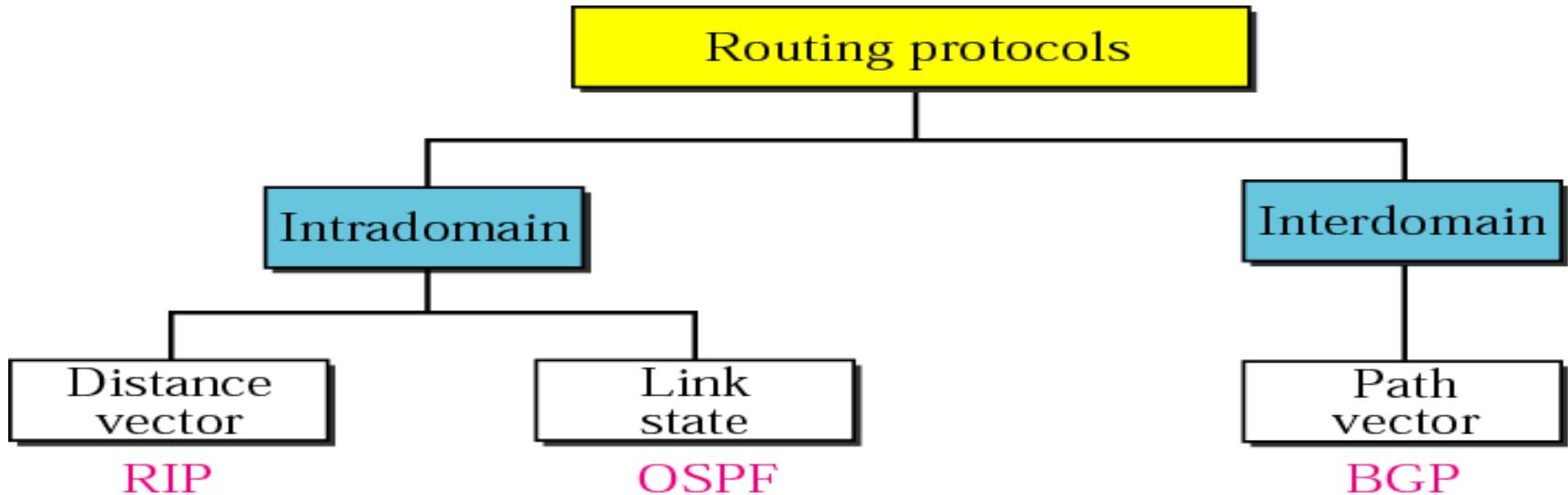
NETWORK LAYER

Autonomous systems

- An autonomous system is a set of networks and routers under the control of a single administrative authority
- Routing within an autonomous system is intra-domain routing
- Routing between autonomous systems is inter-domain routing

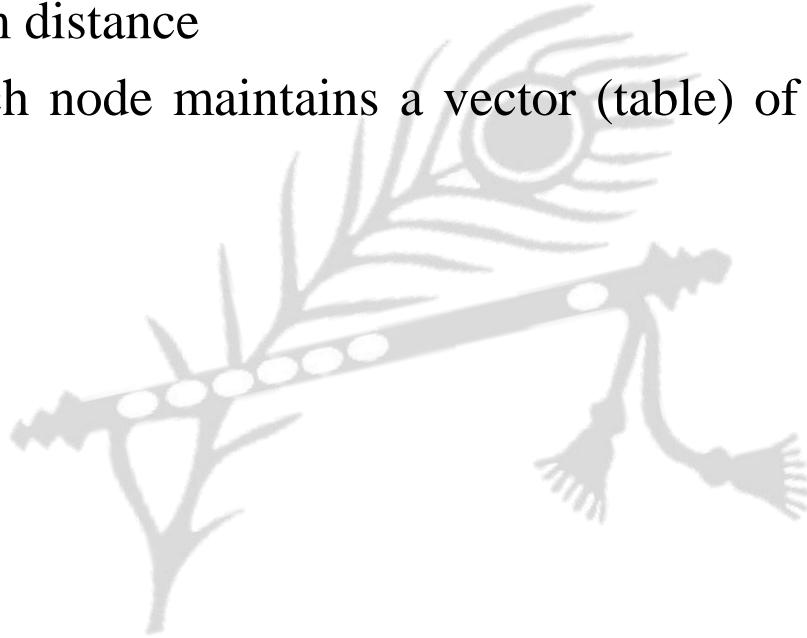


Popular routing protocols



Distance Vector Routing

- In distance vector routing, the least cost route between any two nodes is the route with minimum distance
- In this protocol each node maintains a vector (table) of minimum distances to every node



Distance Vector Routing tables

To	Cost	Next
A	0	—
B	5	—
C	2	—
D	3	—
E	6	C

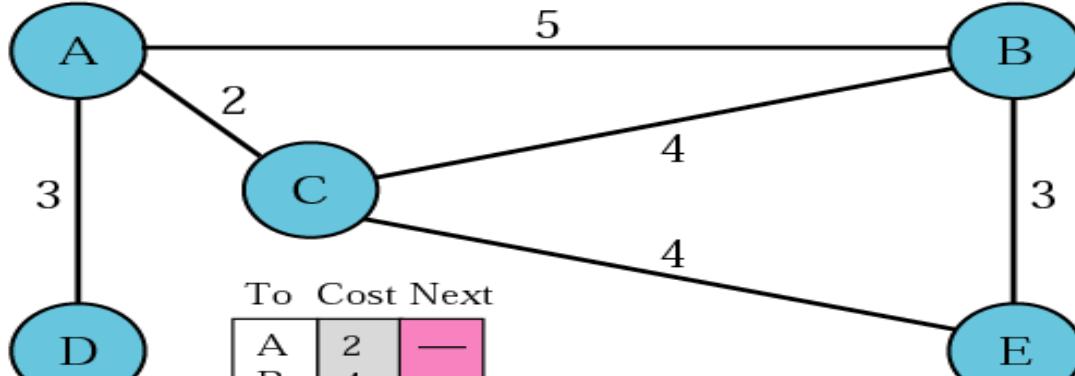
A's table

To	Cost	Next
A	3	—
B	8	A
C	5	A
D	0	—
E	9	A

D's table

To	Cost	Next
A	2	—
B	4	—
C	0	—
D	5	A
E	4	—

C's table



To	Cost	Next
A	5	—
B	0	—
C	4	—
D	8	A
E	3	—

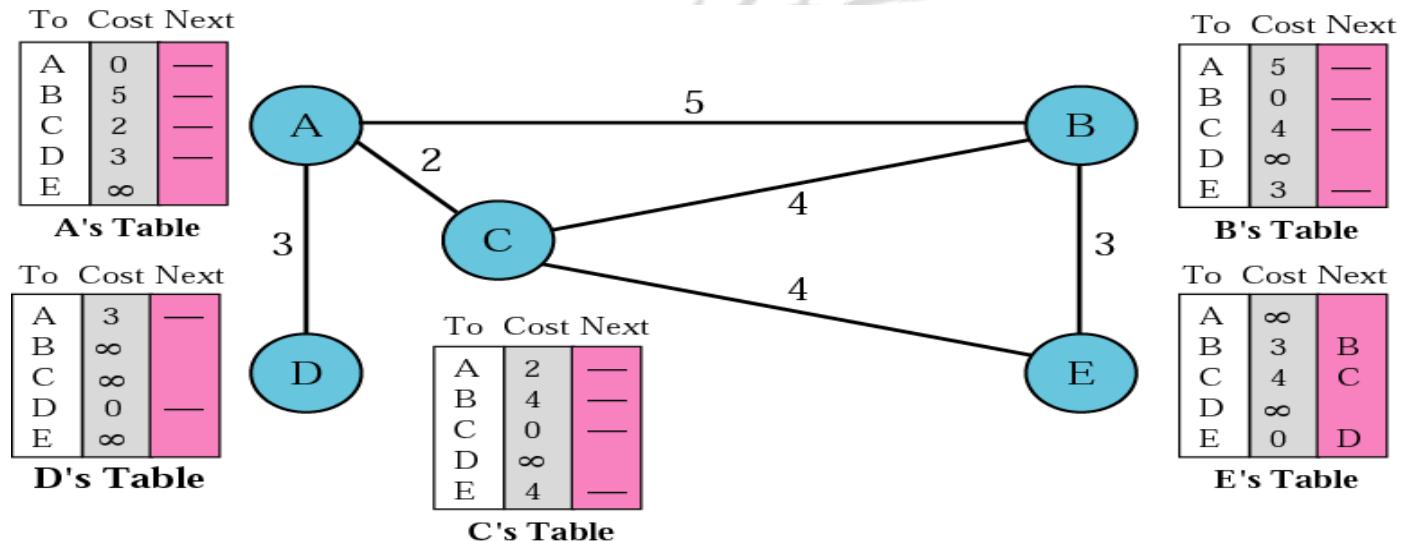
B's table

To	Cost	Next
A	6	C
B	3	—
C	4	—
D	9	C
E	0	—

E's table

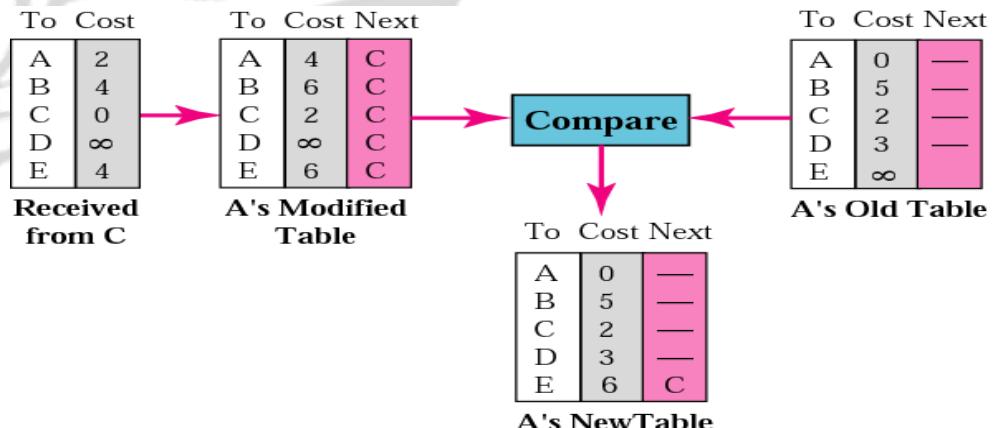
Initialization of tables in distance vector routing

- In distance vector routing, each node shares its table with its immediate neighbor periodically (eg every 30s) and when there is a change

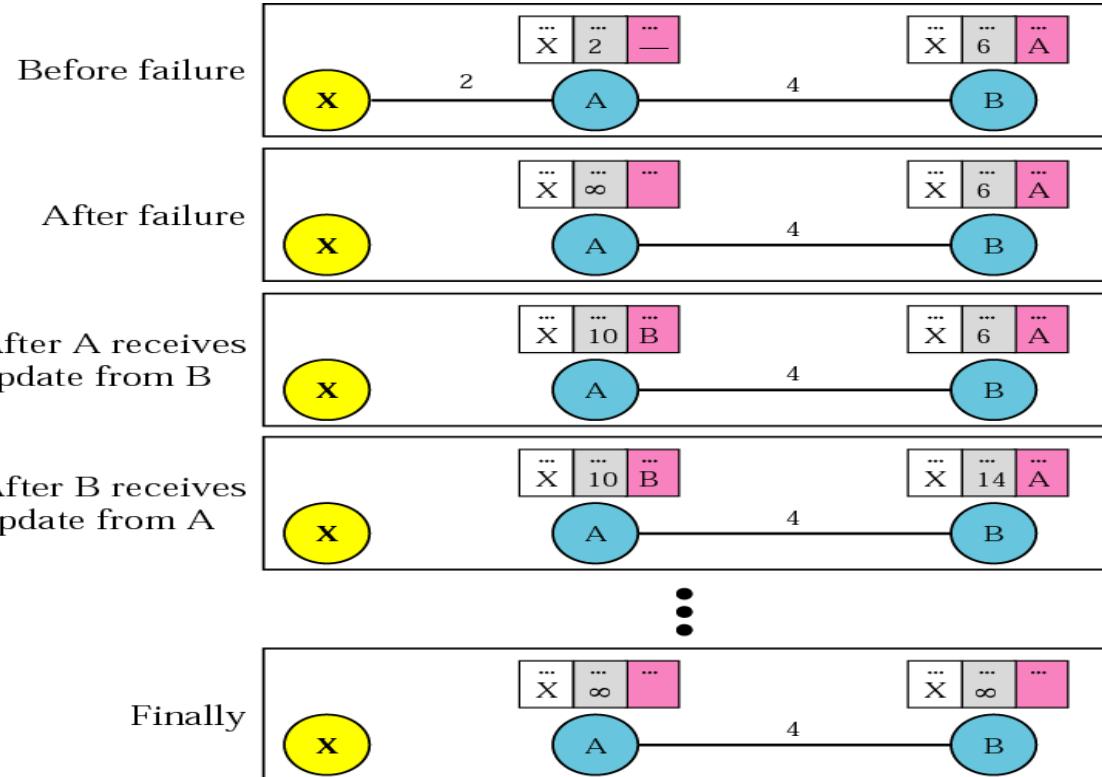


Updating in distance vector routing

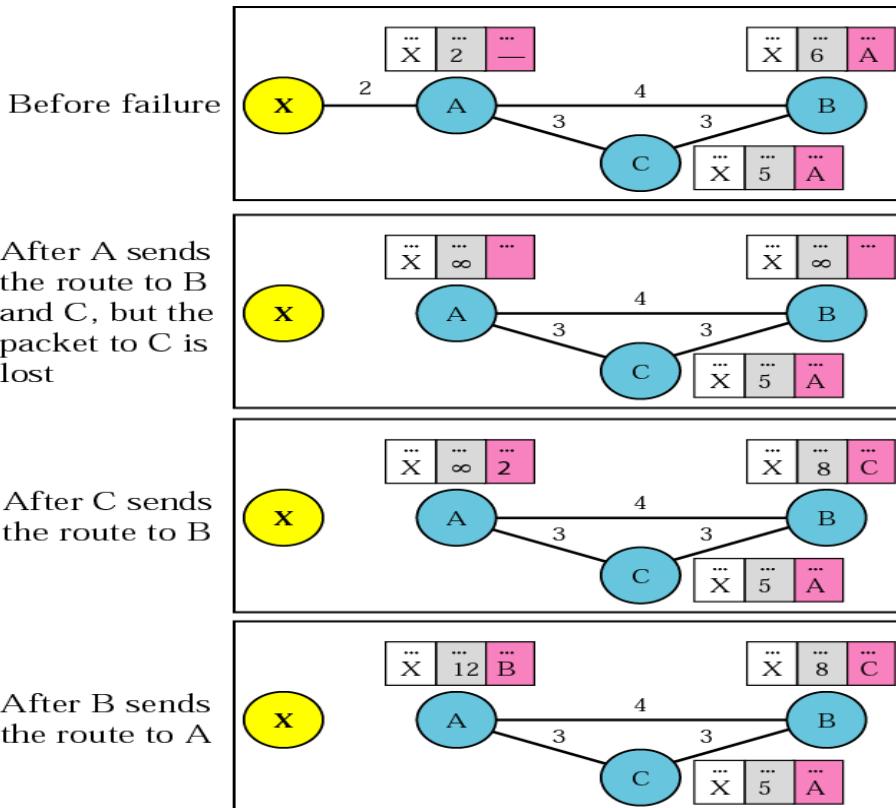
- Step 1: Add cost (2) to table received from neighbor (C)
- Step 2: Compare Modified Table with Old Table (row by row)
- If Next node entry is different, select the row with the smaller cost, if tie, keep the old one
- If next node entry the same, select the new row value



Two-node instability

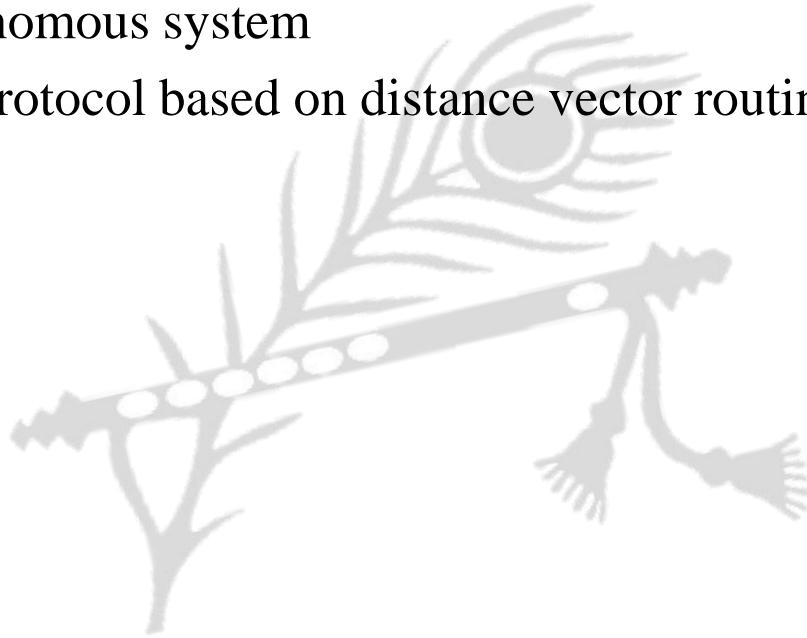


Three-node instability



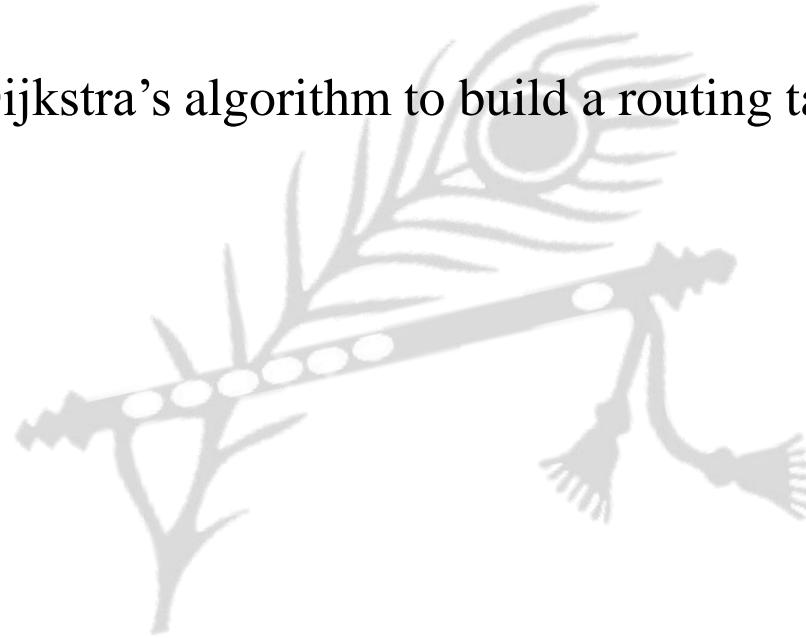
Routing Information Protocol

- The Routing Information Protocol (RIP) is an intra-domain routing protocol used inside an autonomous system
- It is a very simple protocol based on distance vector routing



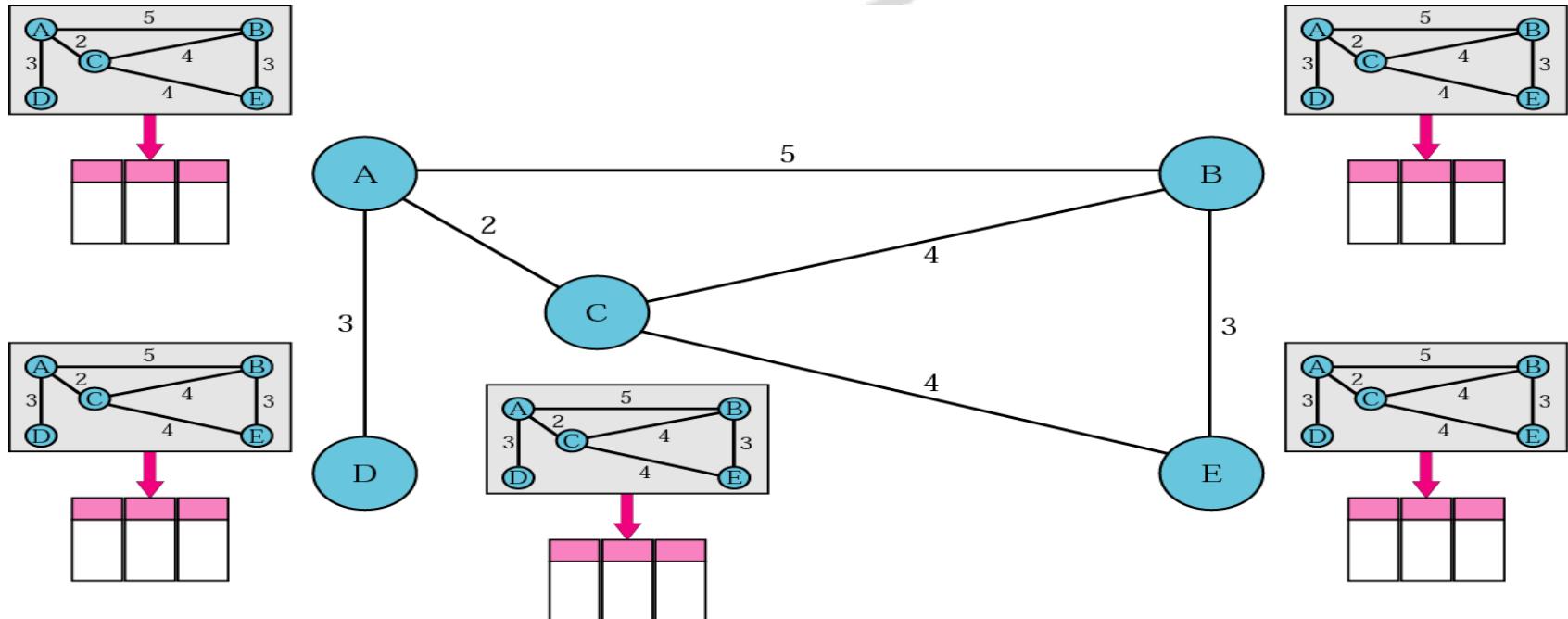
Link state routing

- In link state routing, each node in the domain has the entire topology of the domain
- The node can use Dijkstra's algorithm to build a routing table



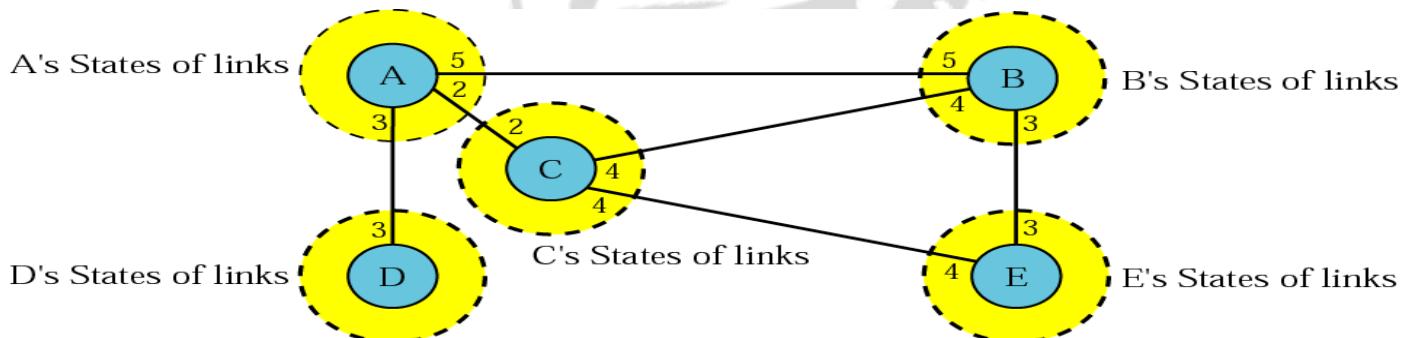
Concept of link state routing

Every router has knowledge about the network, but from its own perspective

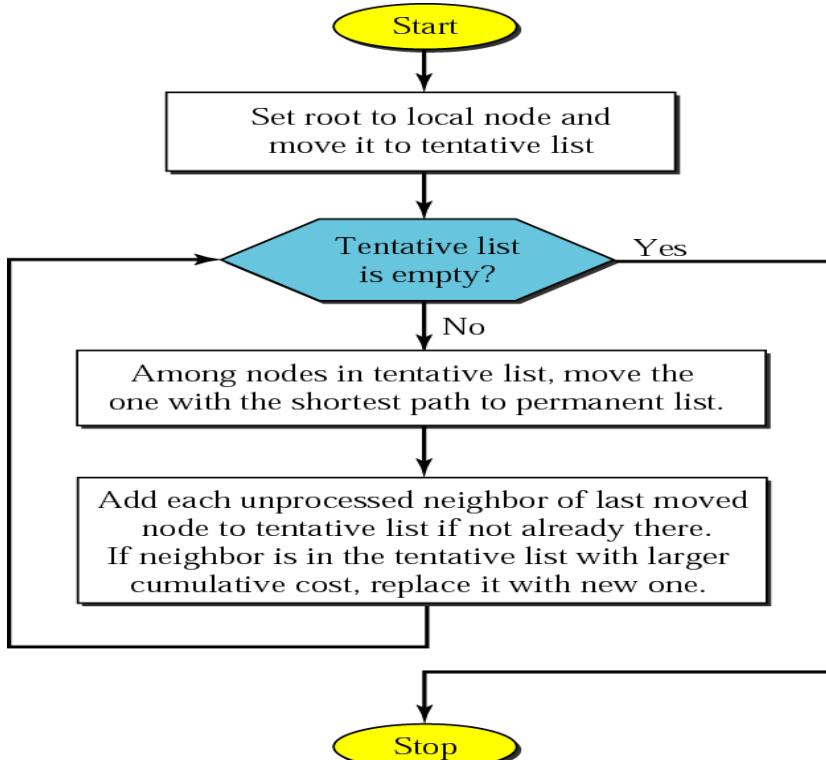


Link state knowledge

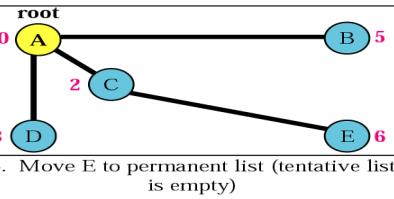
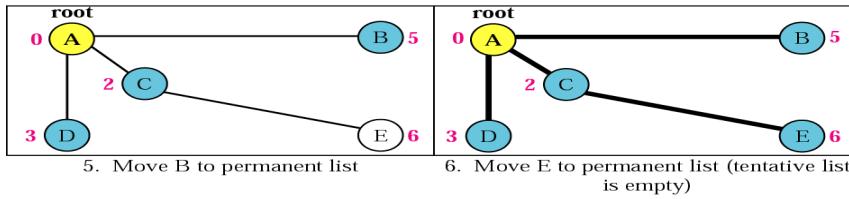
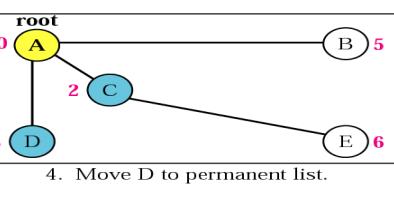
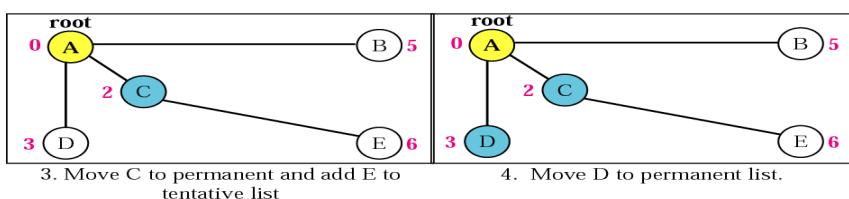
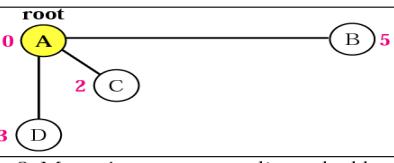
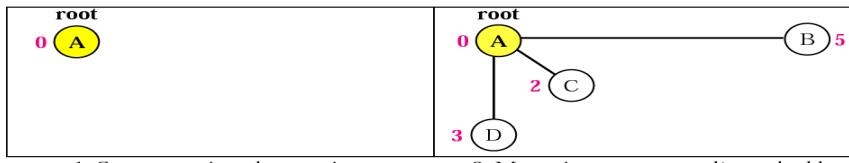
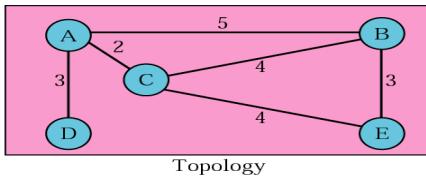
- Each router knows (maintains) its states of its links
- Each router floods this info (via a Link State Packet) to other routers periodically (to check the change in topology)
- Each router takes this data and using Dijkstra's algorithm, creates the shortest path tree and corresponding routing table



Dijkstra algorithm



Example of formation of shortest path tree



Sample Routing table

<i>Node</i>	<i>Cost</i>	<i>Next Router</i>
A	0	—
B	5	—
C	2	—
D	3	—
E	6	C



Comparison of protocols

Link State

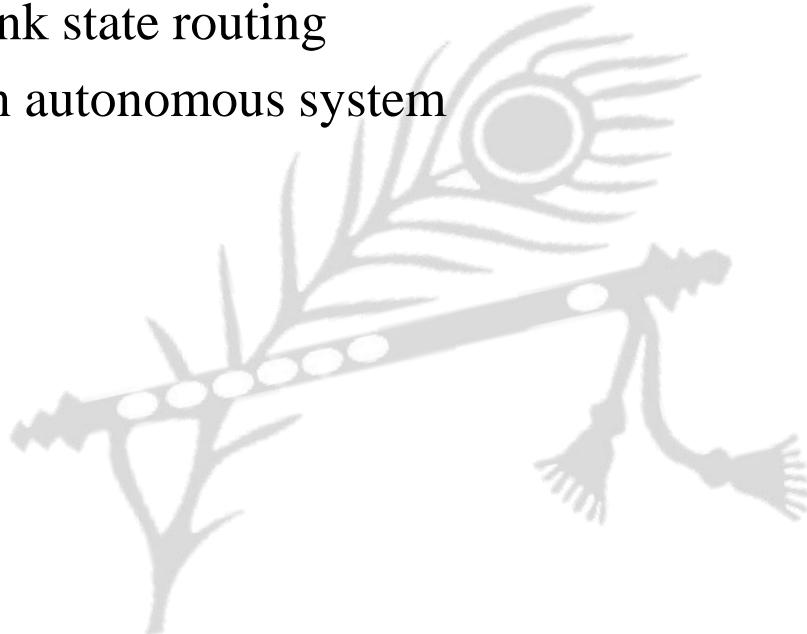
- Knowledge of every router's links (entire graph)
- Every router has $O(\# \text{ edges})$
- Trust a peer's info, do routing computation yourself
- Use Dijkstra's algorithm
- Send updates on any link-state changes
- Ex: OSPF, IS-IS
- Adv: Fast to react to changes

Distance Vector

- Knowledge of neighbors' distance to destinations
- Every router has $O (\# \text{neighbors} * \# \text{nodes})$
- Trust a peer's routing computation
- Use Bellman-Ford algorithm
- Send updates periodically or routing decision change
- Ex: RIP, IGRP
- Adv: Less info & lower computational overhead

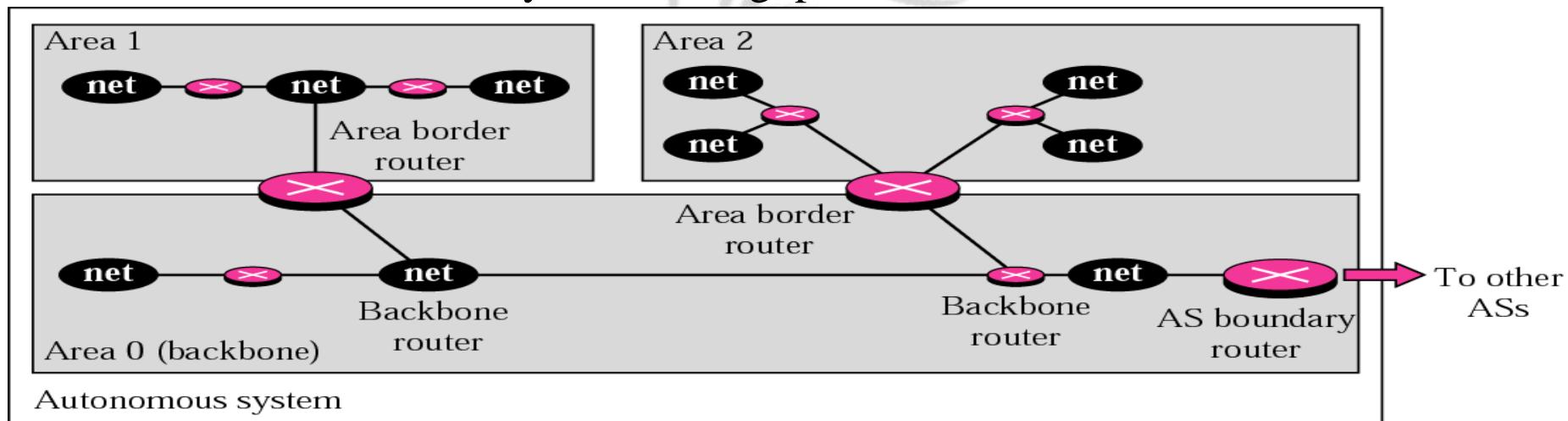
OSPF

- The Open Shortest Path First (OSPF) protocol is an intra-domain routing protocol based on link state routing
- Its domain is also an autonomous system

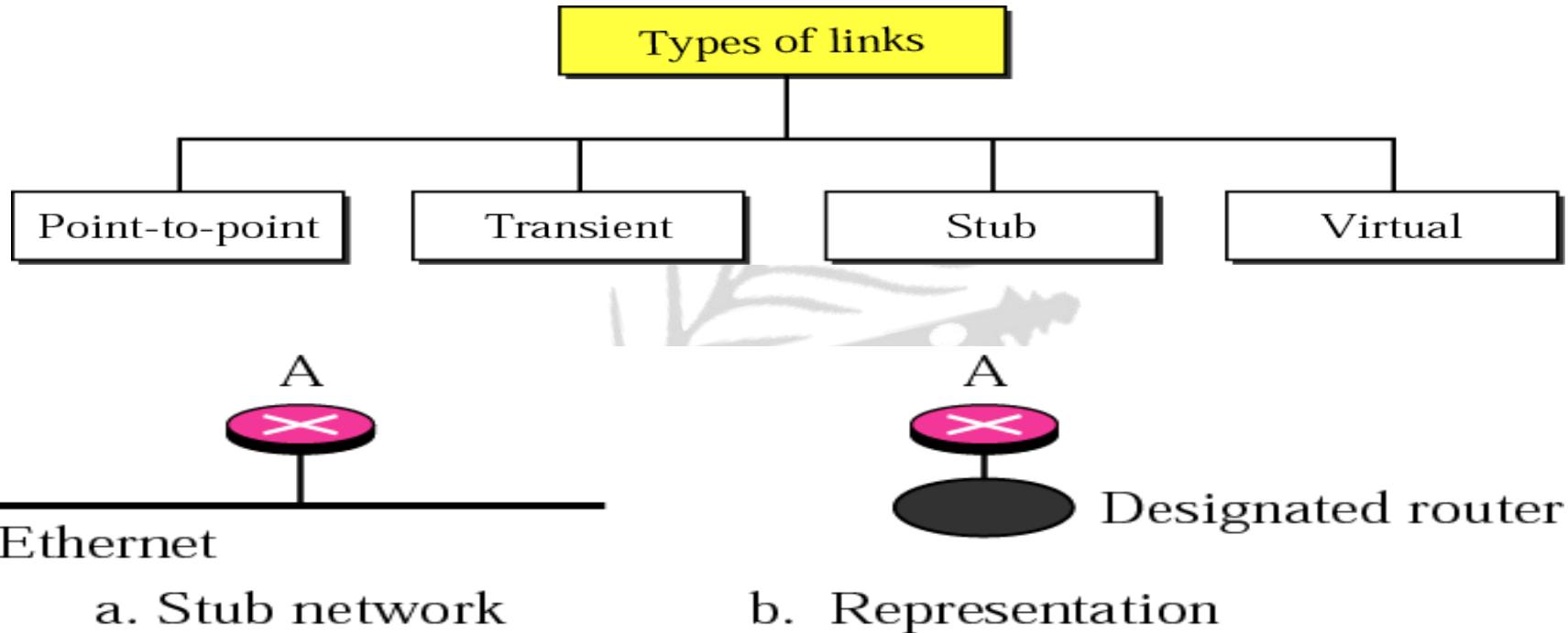


Areas in an autonomous system

- OSPF divides an autonomous system into areas
- All networks inside an area must be connected
- The cost associated with a route is called the metric
- Metric could be min delay, max throughput, etc



Point-to-point link



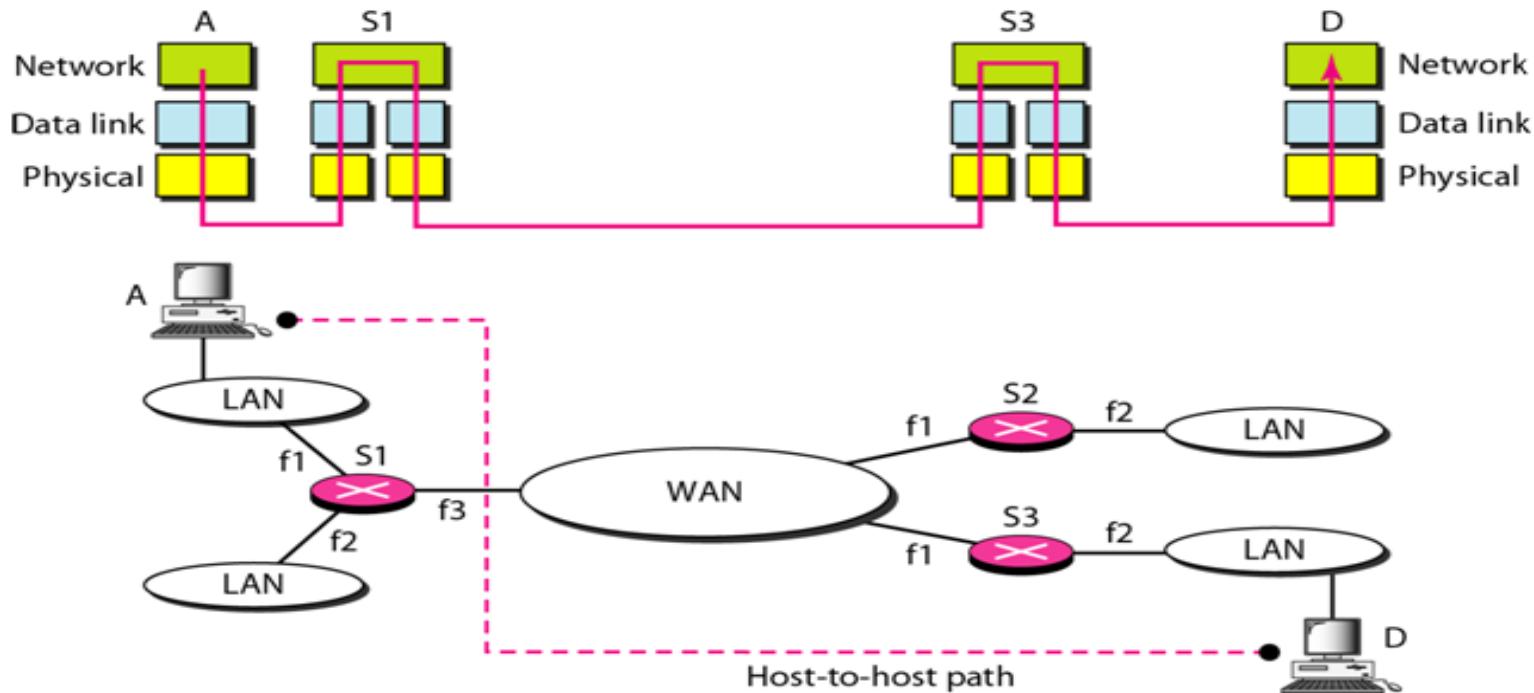
Path Vector Routing

- Path vector routing is similar to distance vector routing
- There is at least one node, called the speaker node in each AS that creates a routing table and advertises it to speaker nodes in the neighboring ASs
- Only the speaker nodes communicate
- The speaker node advertises the path, not the metric of the nodes

BGP

- Border Gateway Protocol (BGP) is an inter-domain routing protocol using path vector routing
- It first appeared in 1989 and has gone through four versions
- BGP interconnects three different types of AS
 - Stub AS, e.g. a corporate network
 - Multihomed AS, e.g. a large corporate network with connections to multiple ASs, but does not allow traffic to pass thru (transient)
 - Transit AS - one that allows transient traffic, such as an Internet backbone

Transmission using Network Layer

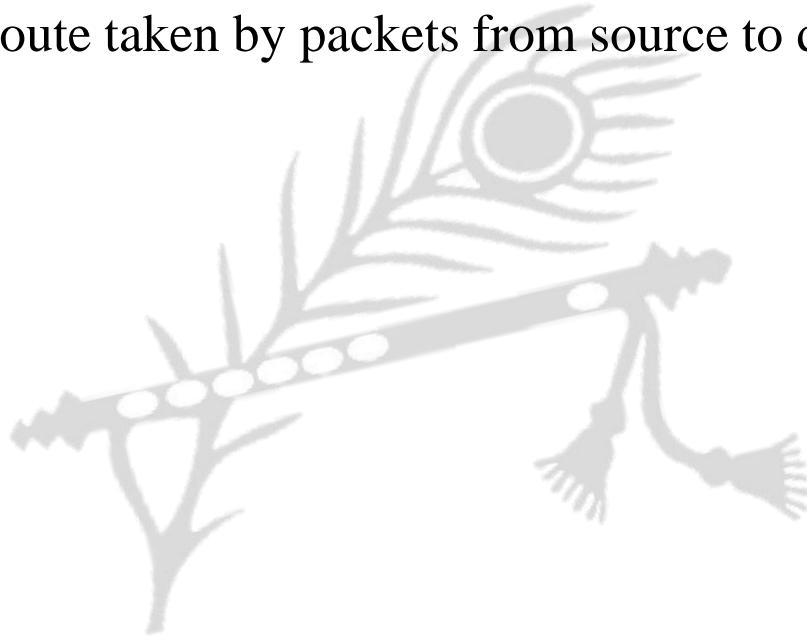


Network Layer

- Services
 - Routing and Forwarding
 - Host to Host Delivery (Using IP Addressing)
 - transports segment from sending to receiving host
 - on sending side encapsulates segments into datagrams
 - on receiving side, delivers segments to transport layer
 - network layer protocols in every host, router
 - router examines header fields in all IP datagrams passing through it

Two key network-layer functions

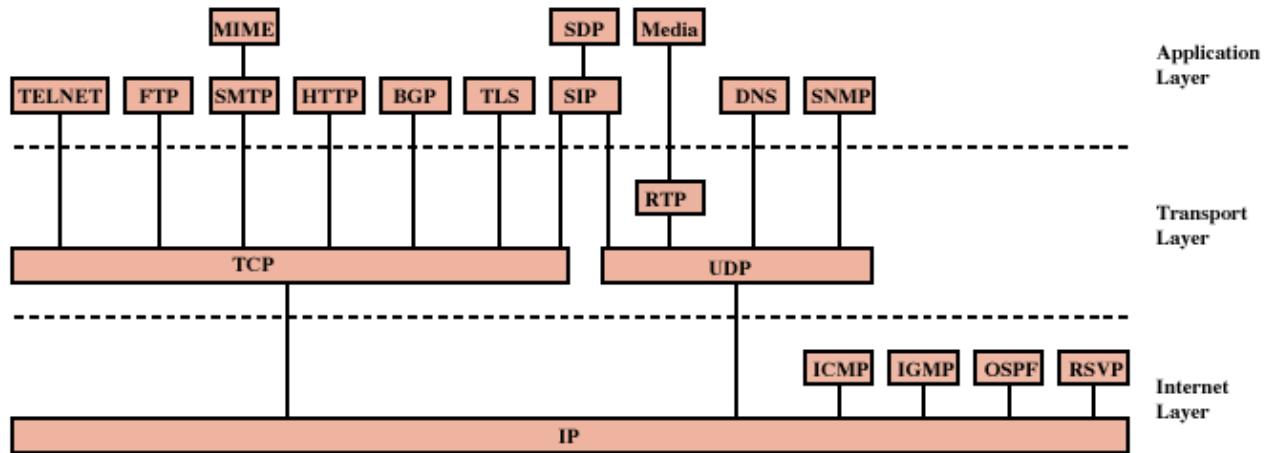
- forwarding: move packets from router's input to appropriate router output
- routing: determine route taken by packets from source to destination



Some basics of IP

- The term internet is short for “internetworking”
 - interconnection of networks with different network access mechanisms, addressing, different routing techniques, etc.
- An internet
 - Collection of communications networks interconnected by layer 3 switches and/or routers
- The Internet - note the uppercase I
 - The global collection of individual machines and networks
- IP (Internet Protocol)
 - most widely used internetworking protocol
 - foundation of all internet-based applications

Protocols of TCP/IP Protocol Suite

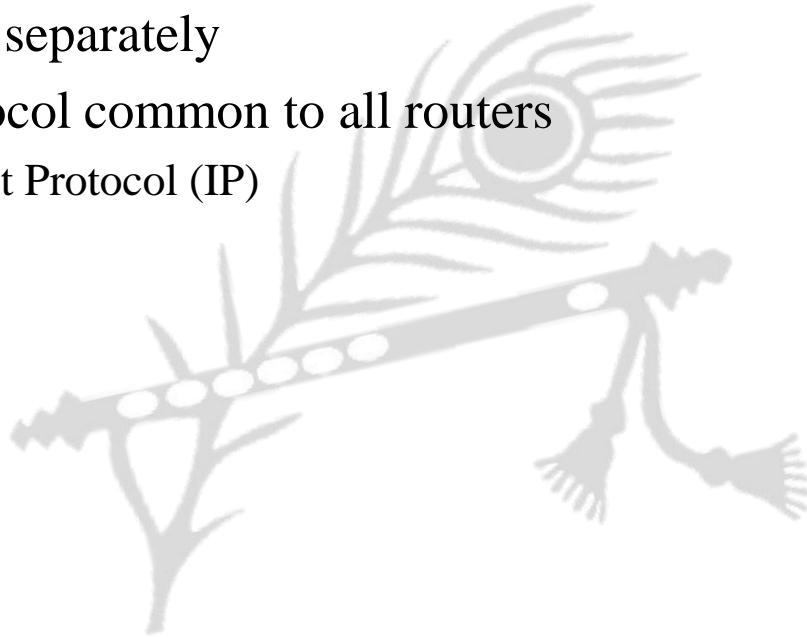


BGP	=	Border Gateway Protocol
DNS	=	Domain Name System
FTP	=	File Transfer Protocol
HTTP	=	Hypertext Transfer Protocol
ICMP	=	Internet Control Message Protocol
IGMP	=	Internet Group Management Protocol
IP	=	Internet Protocol
MIME	=	Multi-Purpose Internet Mail Extension
OSPF	=	Open Shortest Path First

RSVP	=	Resource ReSerVation Protocol
RTP	=	Real-Time Transport Protocol
SDP	=	Session Description Protocol
SIP	=	Session Initiation Protocol
SMTP	=	Simple Mail Transfer Protocol
SNMP	=	Simple Network Management Protocol
TCP	=	Transmission Control Protocol
TLS	=	Transport Layer Security
UDP	=	User Datagram Protocol

Internet Protocol (IP)

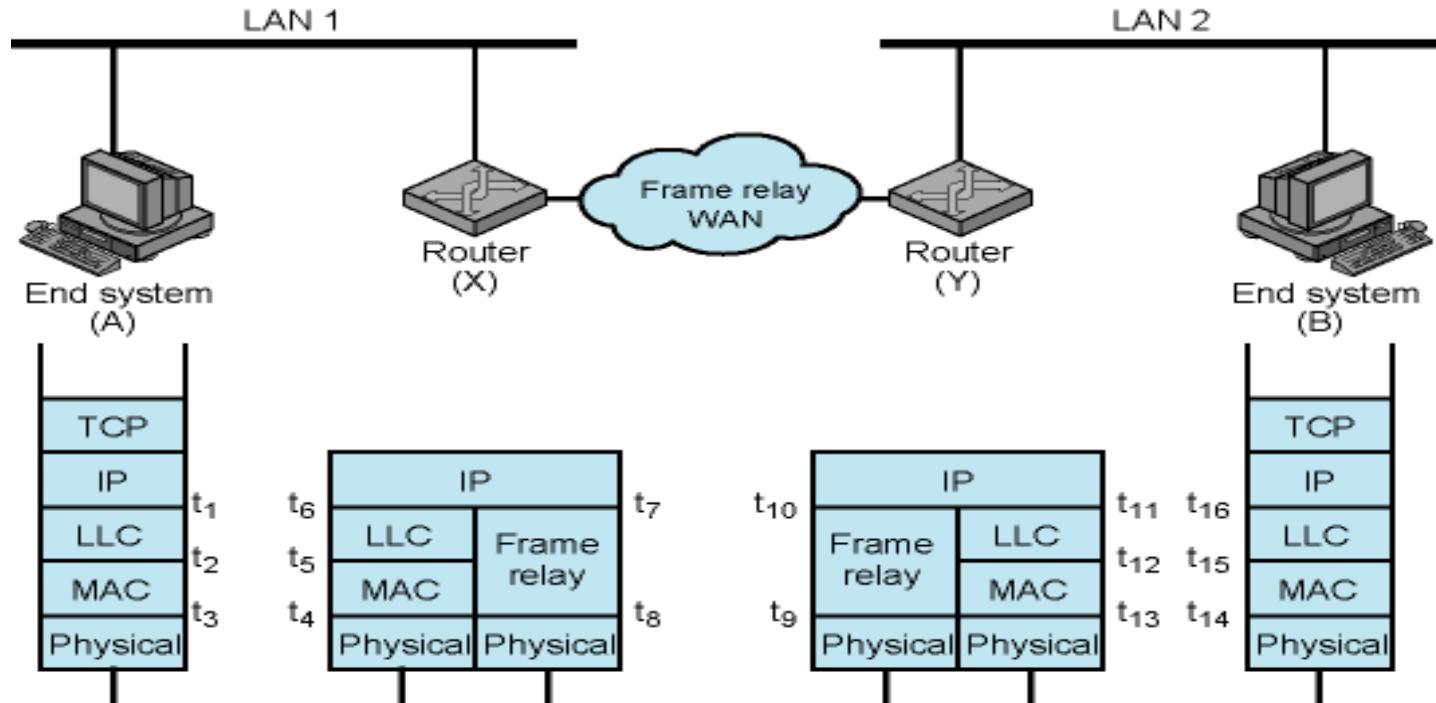
- IP provides connectionless (datagram) service
- Each packet treated separately
- Network layer protocol common to all routers
 - which is the Internet Protocol (IP)



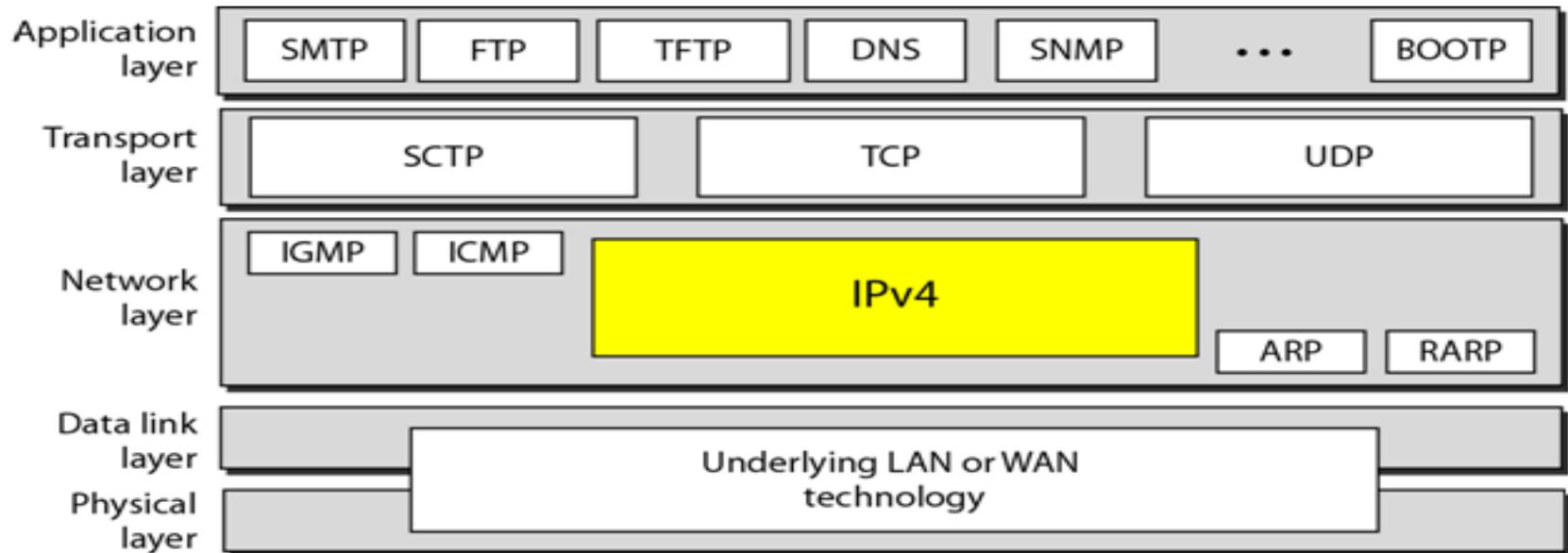
Connectionless Internetworking (General)

- Advantages
 - Flexible and robust
 - e.g. in case of congestion or node failure, packets find their way easier than connection-oriented services
 - No unnecessary overhead for connection setup
 - Can work with different network types
- Disadvantage
 - Unreliable
 - Not guaranteed delivery
 - Not guaranteed order of delivery
 - Reliability is responsibility of next layer up (e.g. TCP)

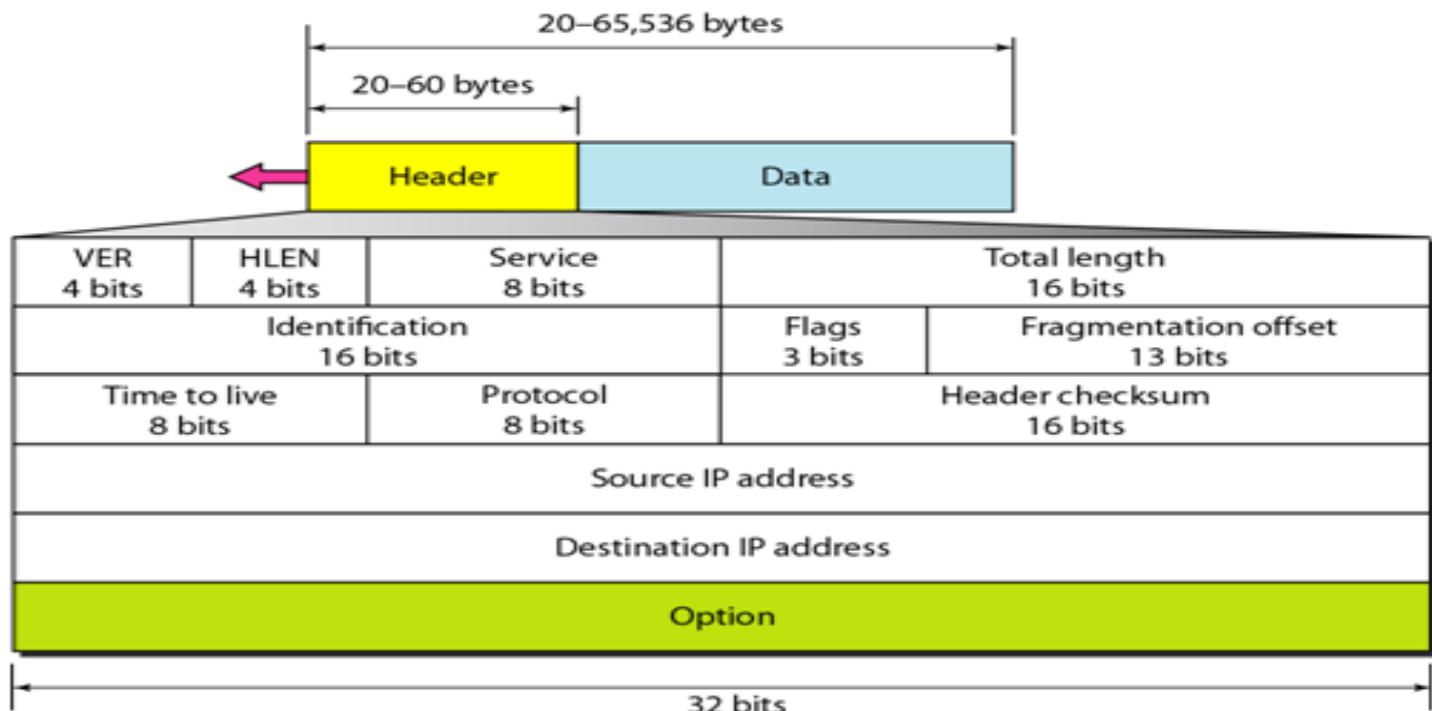
Example Internet Protocol Operation



Internet Protocol (IP) Version 4



Header + Data using IPv4

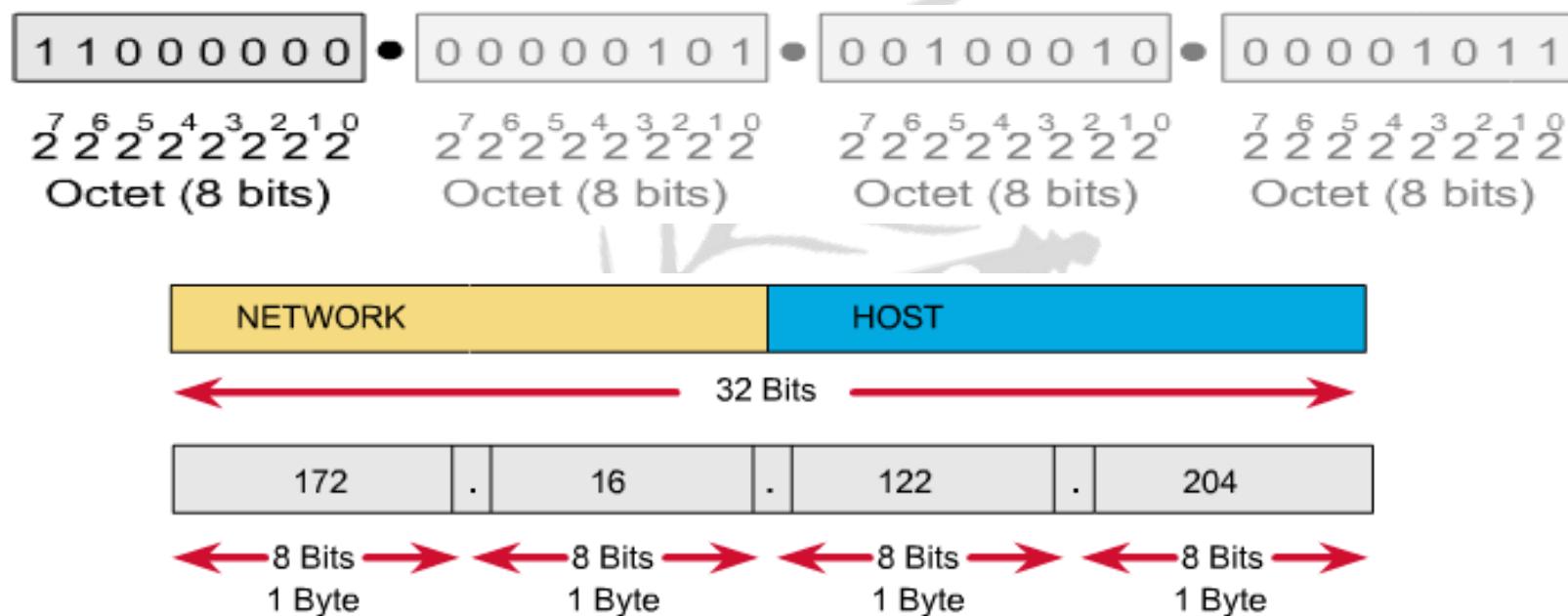


Header + Data using IPv4

- An IPv4 address is 32 bits long
- The IPv4 addresses are unique and universal
- The address space of IPv4 is 2^{32} or 4,294,967,296
- 32 bit address is divided into 4 octet (also known as byte)
- To connect on internet each device requires a unique IP address

IPv4 Addressing

- IP Address as a 32-bit binary number (Four Octet)



Binary to DDN conversion

10000000

00001011

00000011

00011111

128.11.3.31

IPv4

- Eg:
 - Change the following IPv4 addresses from binary notation to dotted-decimal notation
 - 10000001 00001011 00001011 11101111

Solution

- 129.11.11.239

IPv4

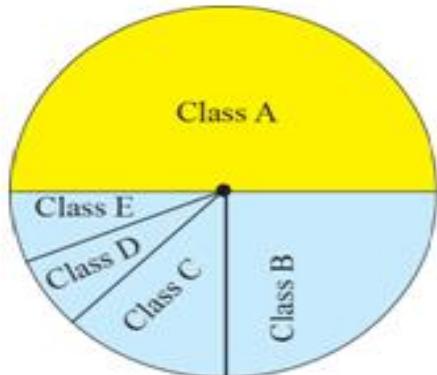
- Eg:
 - Change the following IPv4 addresses from dotted-decimal notation to binary notation
 - 111.56.45.78
- Solution
 - 01101111 00111000 00101101 01001110

IPv4

- Eg:
 - Find the error, if any, in the following IPv4 addresses
 - a. 111.56.045.78
 - b. 221.34.7.8.20
 - c. 75.45.301.14
 - d. 11100010.23.14.67
- Solution
 - a. There should be no leading zeroes (045)
 - b. We may not have more than 4 bytes in an IPv4 address
 - c. Each byte should be less than or equal to 255
 - d. A mixture of binary notation and dotted-decimal notation

IPv4 Addressing Classification

- IPv4 addresses can be classified on 2 ways
 - Classful Addressing
 - Classless Addressing



Class A: $2^{31} = 2,147,483,648$ addresses, 50%

Class B: $2^{30} = 1,073,741,824$ addresses, 25%

Class C: $2^{29} = 536,870,912$ addresses, 12.5%

Class D: $2^{28} = 268,435,456$ addresses, 6.25%

Class E: $2^{28} = 268,435,456$ addresses, 6.25%

Classful addressing

- In classful addressing, the address space is divided into five classes: A, B, C, D, and E

	First byte	Second byte	Third byte	Fourth byte
Class A	0			
Class B	10			
Class C	110			
Class D	1110			
Class E	1111			

a. Binary notation

	First byte	Second byte	Third byte	Fourth byte
Class A	0–127			
Class B	128–191			
Class C	192–223			
Class D	224–239			
Class E	240–255			

b. Dotted-decimal notation

Classful addressing

Class	Net ID bits	Host ID bits	Binary	DDN	No of networks	No of host/Nw	Default mask	CIDR
A (2^{31})	8	24	0	0-127	$2^7=128$	2^{24}	255.0.0.0	/8
B (2^{30})	16	16	10	128-191	2^{14}	2^{16}	255.255.0.0	/16
C (2^{29})	24	8	110	192-223	2^{21}	2^8	255.255.255.0	/24
D (2^{28})	NA	NA	1110	224-239	NA	NA	NA	NA
E (2^{28})	NA	NA	1111	240-255	NA	NA	NA	NA

CIDR = classless Interdomain Routing Notation
 DDN = Dotted Decimal Notation

IPv4

- Eg:
 - Find the class of each address
 - a. 00000001 00001011 00001011 11101111
 - b. 11000001 10000011 00011011 11111111
 - c. 14.23.120.8
 - d. 252.5.15.111
- Solution
 - a. The first bit is 0. This is a class A address
 - b. The first 2 bits are 1; the third bit is 0. This is a class C address
 - c. The first byte is 14; the class is A
 - d. The first byte is 252; the class is E

Classes and Blocks

- The classful addressing wastes a large part of the address space



<i>Class</i>	<i>Number of Blocks</i>	<i>Block Size</i>	<i>Application</i>
A	128	16,777,216	Unicast
B	16,384	65,536	Unicast
C	2,097,152	256	Unicast
D	1	268,435,456	Multicast
E	1	268,435,456	Reserved



Structure of IPv4 Address

- Consists of Net ID and Host ID

<i>Class</i>	<i>Binary</i>	<i>Dotted-Decimal</i>	<i>CIDR</i>
A	11111111 00000000 00000000 00000000	255 .0.0.0	/8
B	11111111 11111111 00000000 00000000	255.255 .0.0	/16
C	11111111 11111111 11111111 00000000	255.255.255 .0	/24

- Mask
 - 32-bit number of contiguous 1's followed by contiguous 0's
 - To help to find the net ID and the host ID

Use of IPv4 Address

- Subnetting
 - Divide a large address block into smaller sub-groups
 - Use of flexible net mask
- Supernetting
 - Exhausted class A and B address space
 - Huge demand for class B address space
 - To combine several contiguous address spaces into a larger single address space

Classless Addressing

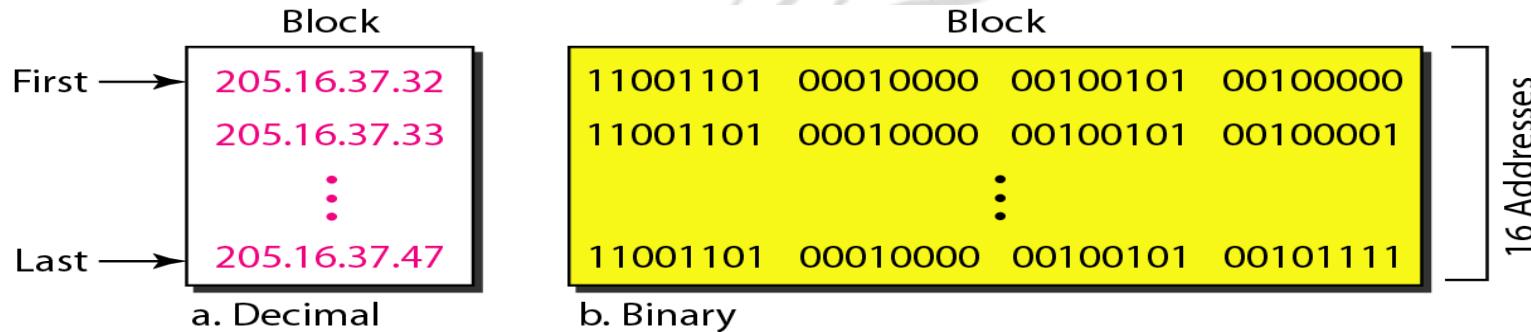
- To overcome the depletion of address space
- Restriction
 - The addresses in a block must be contiguous
 - The number of addresses in a block must be a power of 2
 - The first address must be evenly divisible by the number of address
- Mask
 - Consists of n consecutive 1's followed by zeros
 - n can be any number b/w 0 and 32

Classless Addressing

- In IPv4 addressing, a block of addresses can be defined as x.y.z.t /n, in which x.y.z.t defines one of the addresses and the /n defines the mask
- The first address in the block can be found by setting the rightmost $32 - n$ bits to 0s
- The last address in the block can be found by setting the rightmost $32 - n$ bits to 1s
- The number of addresses in the block can be found by using the formula 2^{32-n}

Eg

- Figure shows a block of addresses, in both binary and dotted-decimal notation, granted to a small business that needs 16 addresses



- We can see that the restrictions are applied to this block. The addresses are contiguous. The number of addresses is a power of 2 ($16 = 2^4$), and the first address is divisible by 16.

Eg:

- A block of addresses is granted to a small organization. We know that one of the addresses is 205.16.37.39/28. What is the first address in the block?

Solution

- The binary representation of the given address is

11001101 00010000 00100101 00100111

- If we set 32–28 rightmost bits to 0, we get

11001101 00010000 00100101 00100000

or

205.16.37.32

Eg:

- Find the last address for the block in previous example

Solution

- The binary representation of the given address is

11001101 00010000 00100101 00100111

- If we set 32 – 28 rightmost bits to 1, we get

11001101 00010000 00100101 00101111

or

205.16.37.47

Eg:

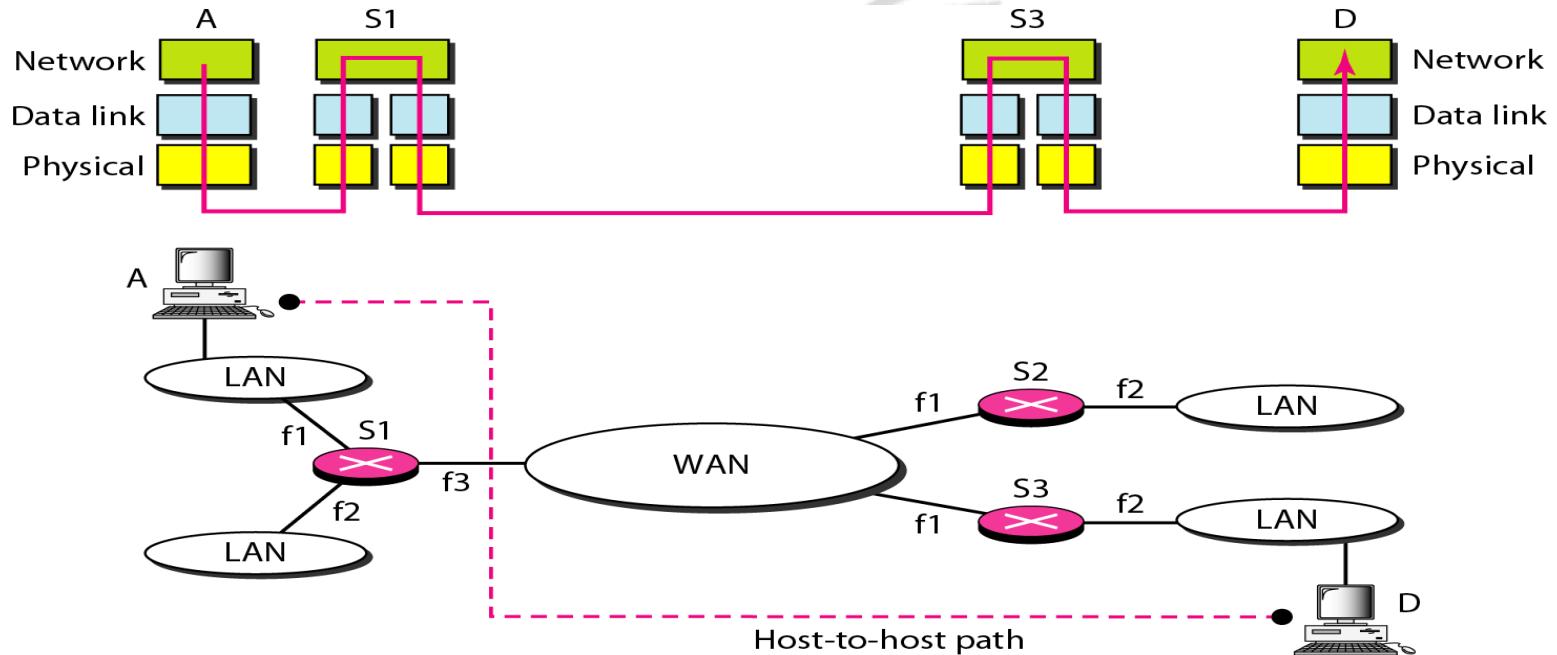
- Find the number of addresses in the previous example

Solution

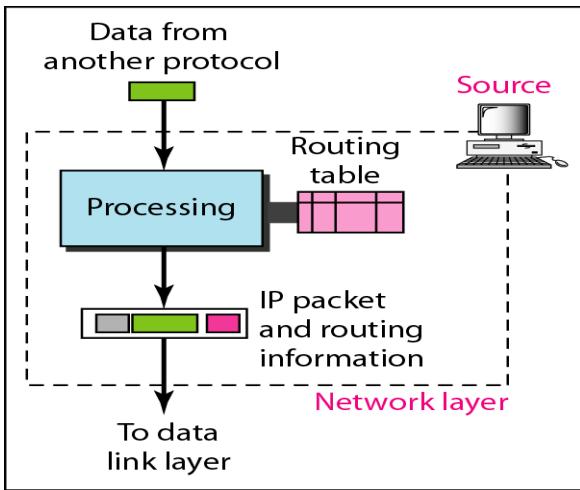
- The value of n is 28, which means that number of addresses is 2^{32-28} or 16

Internetworking

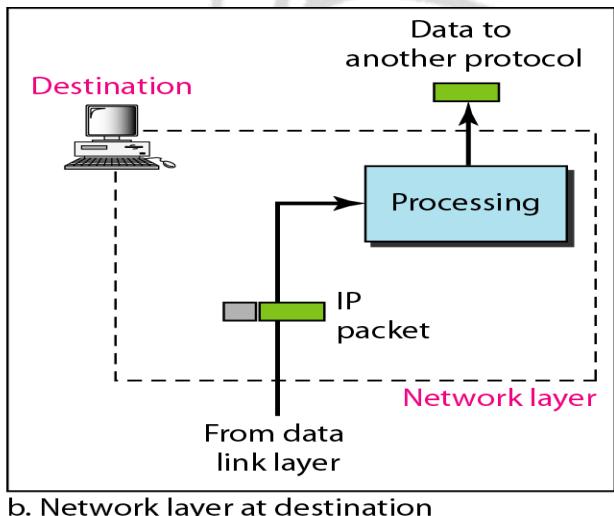
- Connecting networks together to make an internetwork or an internet



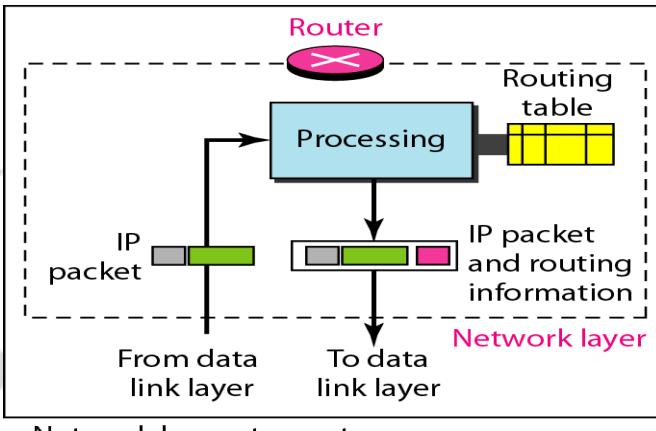
Network layer at the source, destination and router



a. Network layer at source

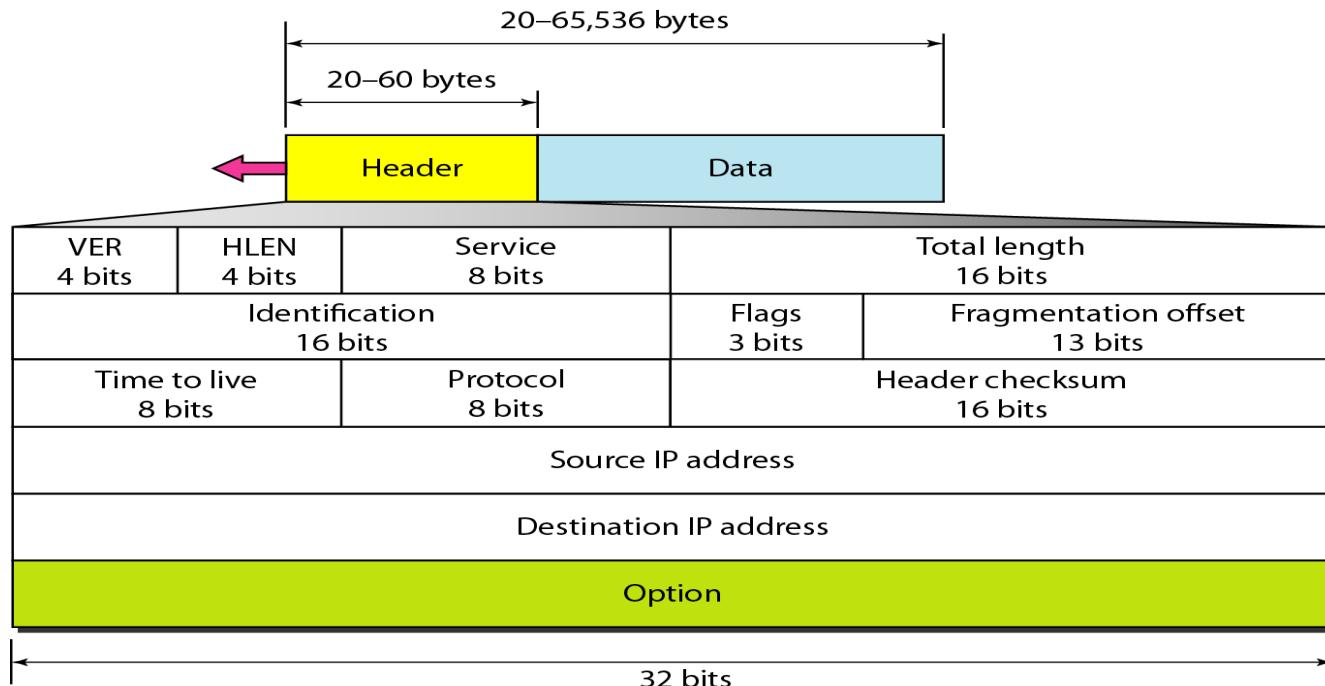


b. Network layer at destination

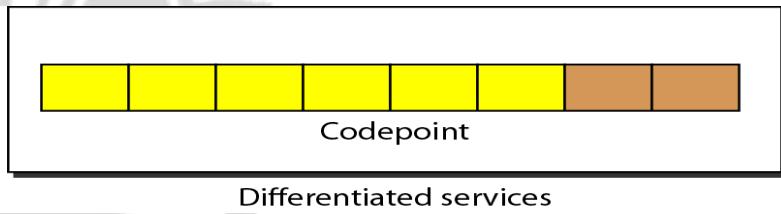
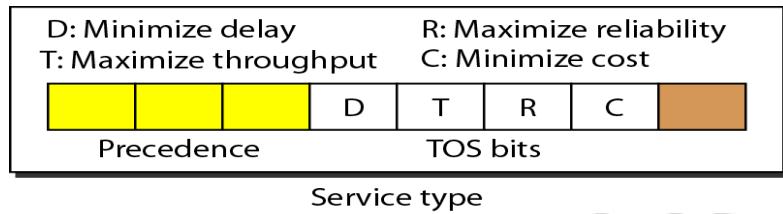


c. Network layer at a router

IPv4 datagram format



Service type or differentiated services



Types of service

<i>TOS Bits</i>	<i>Description</i>
0000	Normal (default)
0001	Minimize cost
0010	Maximize reliability
0100	Maximize throughput
1000	Minimize delay



Default types of service

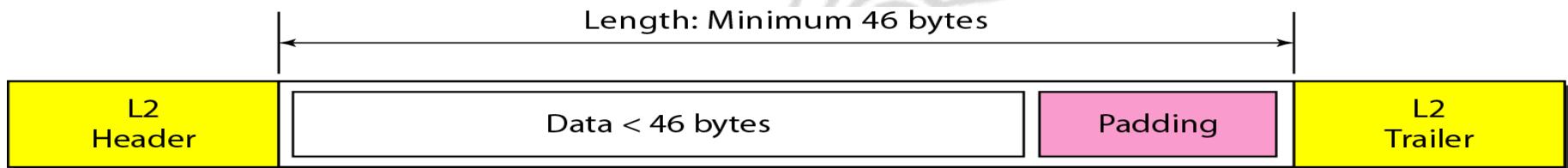
<i>Protocol</i>	<i>TOS Bits</i>	<i>Description</i>
ICMP	0000	Normal
BOOTP	0000	Normal
NNTP	0001	Minimize cost
IGP	0010	Maximize reliability
SNMP	0010	Maximize reliability
TELNET	1000	Minimize delay
FTP (data)	0100	Maximize throughput
FTP (control)	1000	Minimize delay
TFTP	1000	Minimize delay
SMTP (command)	1000	Minimize delay
SMTP (data)	0100	Maximize throughput
DNS (UDP query)	1000	Minimize delay
DNS (TCP query)	0000	Normal
DNS (zone)	0100	Maximize throughput

Protocol values

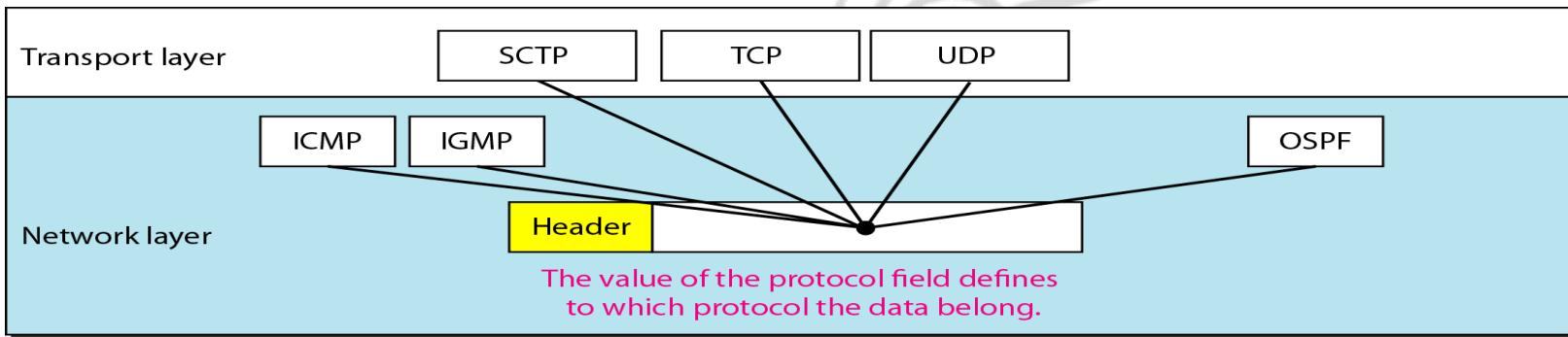
<i>Value</i>	<i>Protocol</i>
1	ICMP
2	IGMP
6	TCP
17	UDP
89	OSPF

Encapsulation of a small datagram in an Ethernet frame

- The total length field defines the total length of the datagram including the header



Protocol field and encapsulated data



Example

- An IPv4 packet has arrived with the first 8 bits as shown:

01000010

- The receiver discards the packet. Why?

Solution

- There is an error in this packet. The 4 leftmost bits (0100) show the version, which is correct. The next 4 bits (0010) show an invalid header length ($2 \times 4 = 8$)
- The minimum number of bytes in the header must be 20. The packet has been corrupted in transmission

Example

- In an IPv4 packet, the value of HLEN is 1000 in binary. How many bytes of options are being carried by this packet?

Solution

- The HLEN value is 8, which means the total number of bytes in the header is 8×4 , or 32 bytes
- The first 20 bytes are the base header, the next 12 bytes are the options.

Example

- In an IPv4 packet, the value of HLEN is 5, and the value of the total length field is 0x0028. How many bytes of data are being carried by this packet?

Solution

- The HLEN value is 5, which means the total number of bytes in the header is 5×4 , or 20 bytes (no options)
- The total length is 40 bytes, which means the packet is carrying 20 bytes of data ($40 - 20$)

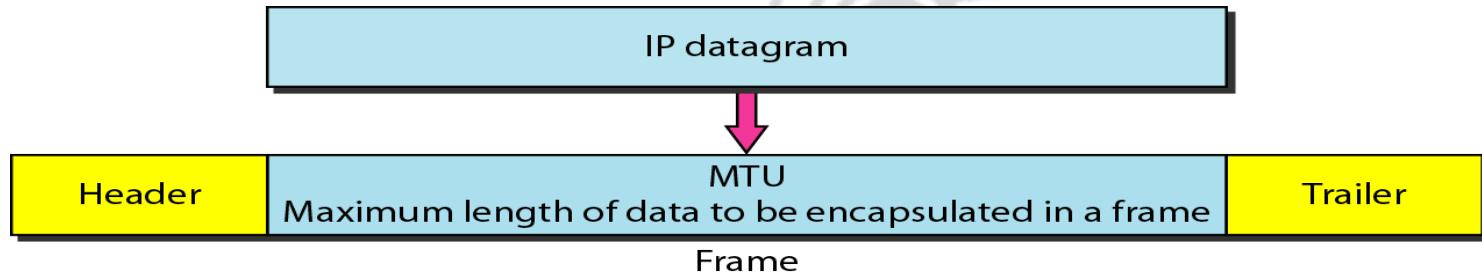
Example

- An IPv4 packet has arrived with the first few hexadecimal digits as shown.
 $0x45000028000100000102 \dots$
- How many hops can this packet travel before being dropped? The data belong to what upper-layer protocol?

Solution

- To find the time-to-live field, we skip 8 bytes. The time-to-live field is the ninth byte, which is 01
- This means the packet can travel only one hop. The protocol field is the next byte (02), which means that the upper-layer protocol is IGMP

Maximum transfer unit (MTU)



MTUs for some networks

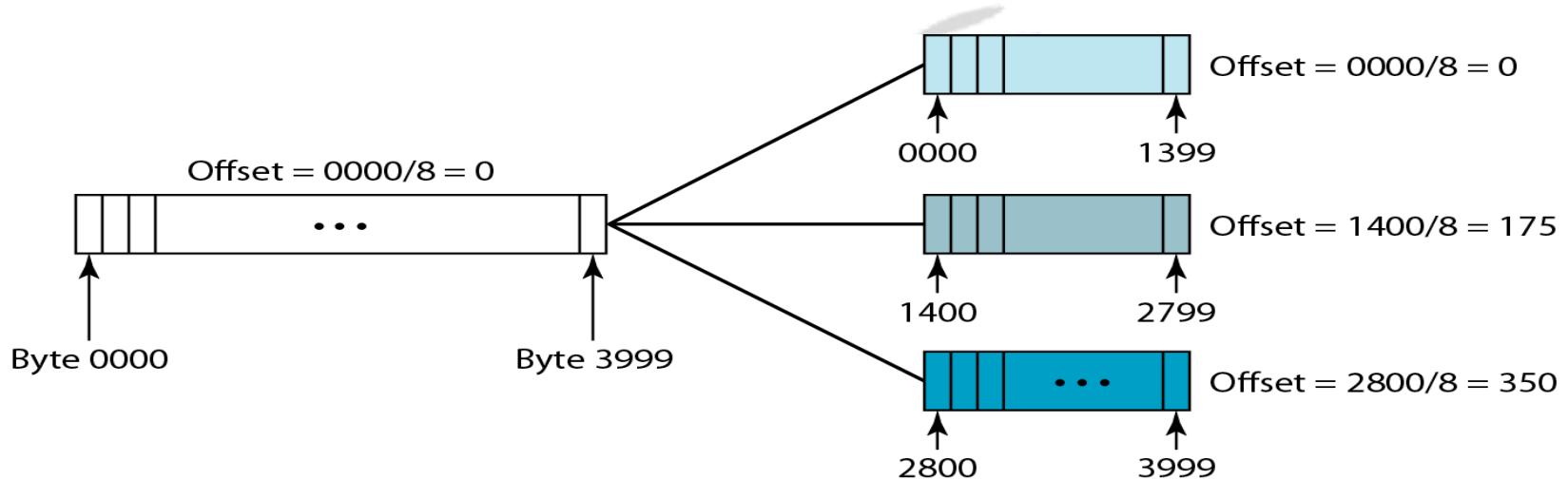
<i>Protocol</i>	<i>MTU</i>
Hyperchannel	65,535
Token Ring (16 Mbps)	17,914
Token Ring (4 Mbps)	4,464
FDDI	4,352
Ethernet	1,500
X.25	576
PPP	296

Flags used in fragmentation

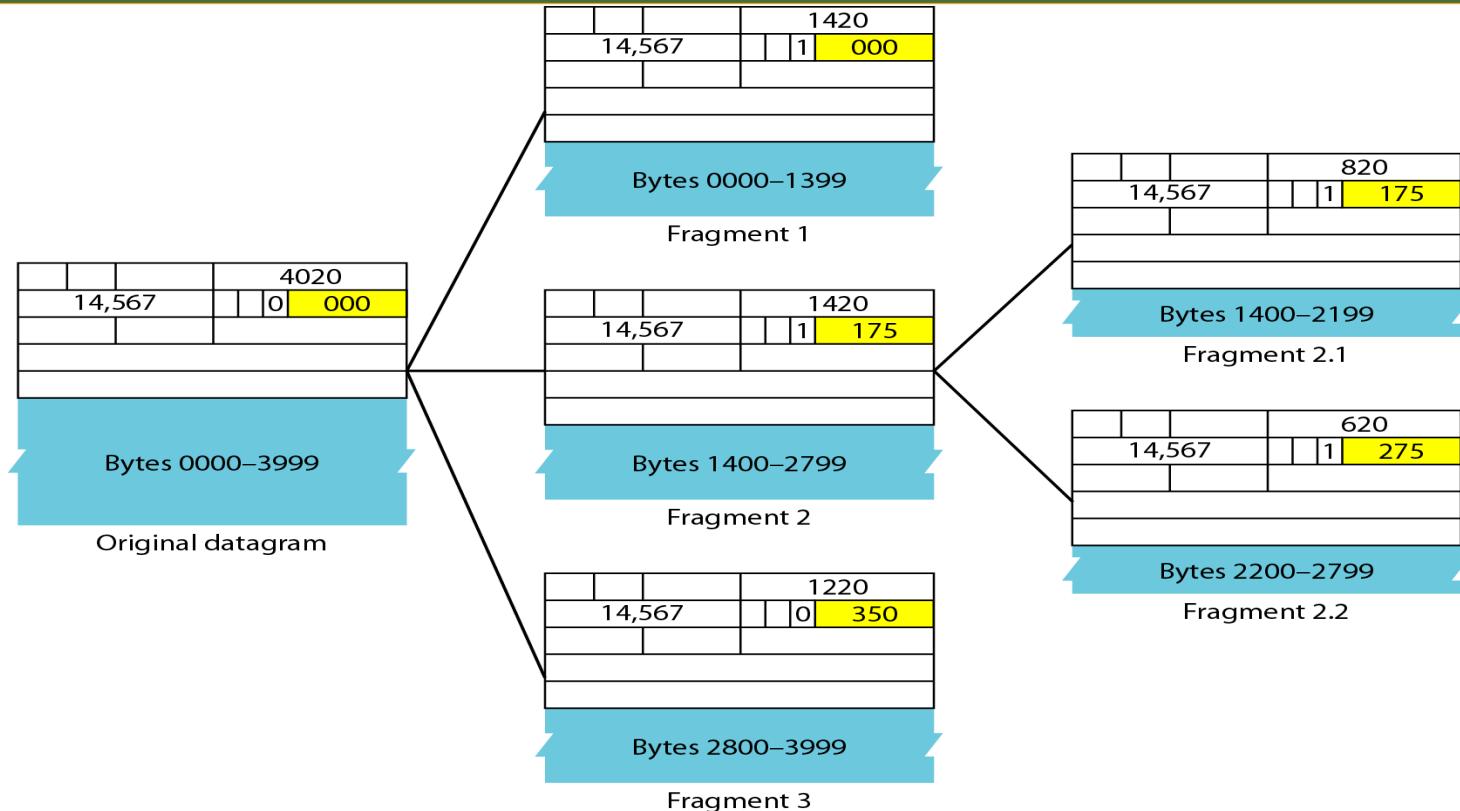


D: Do not fragment
M: More fragments

Fragmentation example



Detailed fragmentation example



Example

- A packet has arrived with an M bit value of 0. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?

Solution

- If the M bit is 0, it means that there are no more fragments; the fragment is the last one
- However, we cannot say if the original packet was fragmented or not. A non-fragmented packet is considered the last fragment

Example

- A packet has arrived with an M bit value of 1. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?

Solution

- If the M bit is 1, it means that there is at least one more fragment. This fragment can be the first one or a middle one, but not the last one
- We don't know if it is the first one or a middle one; we need more information (the value of the fragmentation offset)

Example

- A packet has arrived with an M bit value of 1 and a fragmentation offset value of 0. Is this the first fragment, the last fragment, or a middle fragment?

Solution

- Because the M bit is 1, it is either the first fragment or a middle one
- Because the offset value is 0, it is the first fragment

Example

- A packet has arrived in which the offset value is 100. What is the number of the first byte? Do we know the number of the last byte?

Solution

- To find the number of the first byte, we multiply the offset value by 8. This means that the first byte number is 800
- We cannot determine the number of the last byte unless we know the length

Example

- A packet has arrived in which the offset value is 100, the value of HLEN is 5, and the value of the total length field is 100. What are the numbers of the first byte and the last byte?

Solution

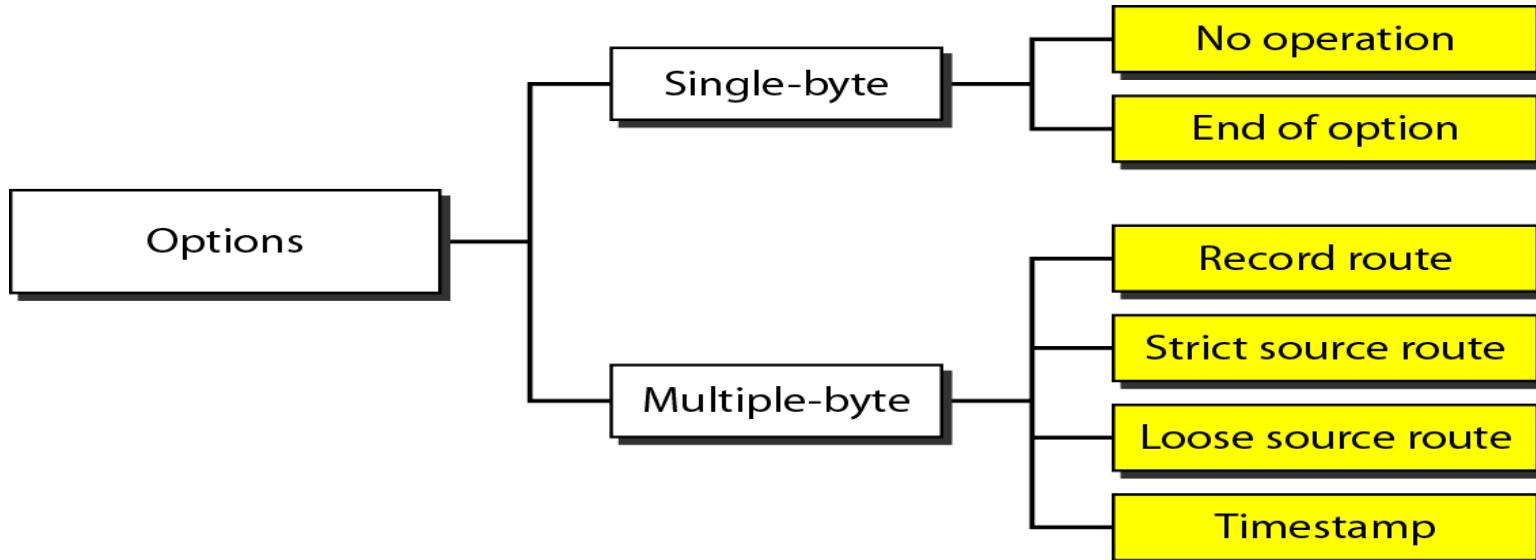
- The first byte number is $100 \times 8 = 800$
- The total length is 100 bytes, and the header length is 20 bytes (5×4), which means that there are 80 bytes in this datagram. If the first byte number is 800, the last byte number must be 879

Example

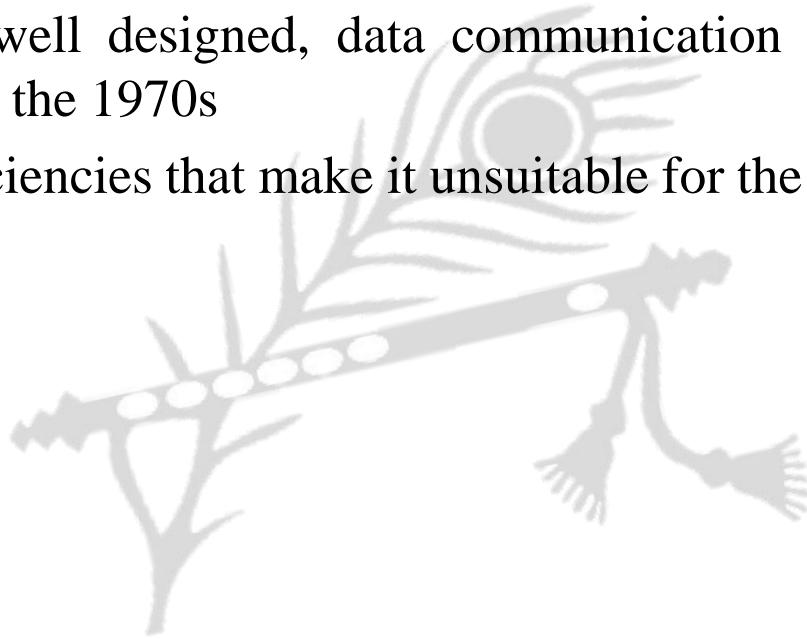
- An example of a checksum calculation for an IPv4 header without options. The header is divided into 16-bit sections. All the sections are added and the sum is complemented. The result is inserted in the checksum field

4	5	0	28			
1		0 0				
4	17	0				
10.12.14.5						
12.6.7.9						
4, 5, and 0	4	5	0	0		
28	0	0	1	C		
1	0	0	0	1		
0 and 0	0	0	0	0		
4 and 17	0	4	1	1		
0	0	0	0	0		
10.12	0	A	0	C		
14.5	0	E	0	5		
12.6	0	C	0	6		
7.9	0	7	0	9		
Sum	7	4	4	E		
Checksum	8	B	B	1		

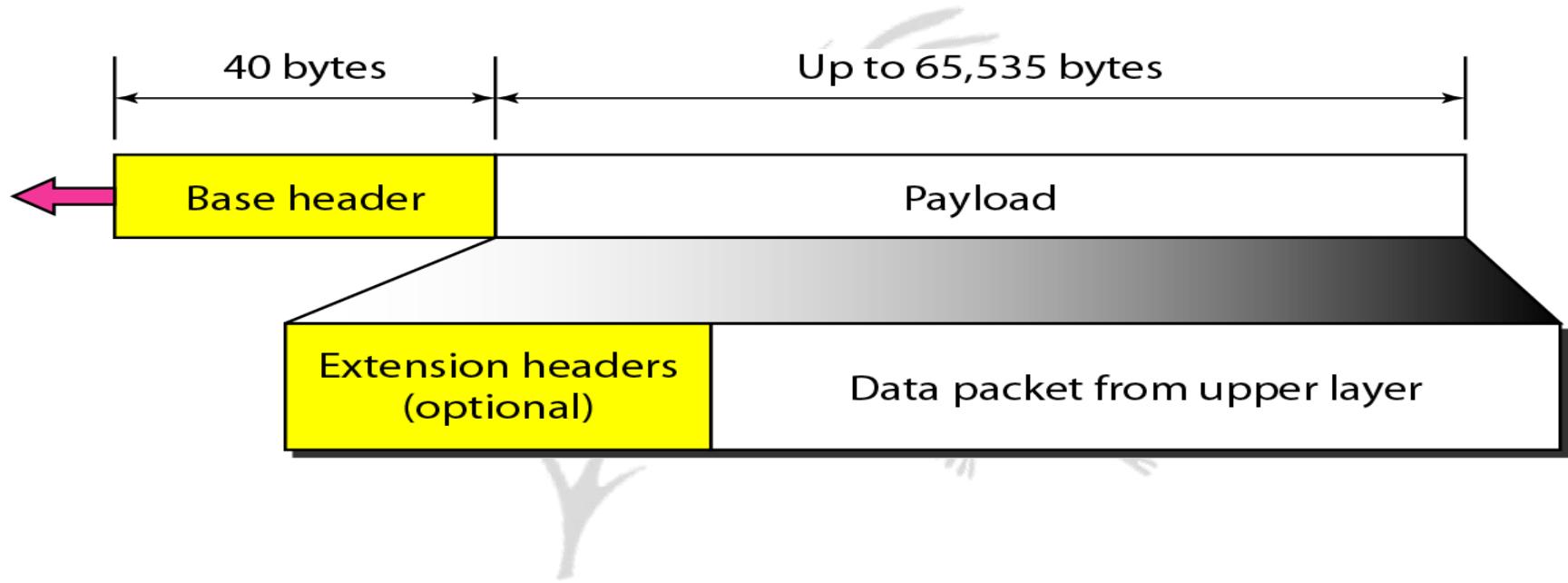
Taxonomy of options in IPv4



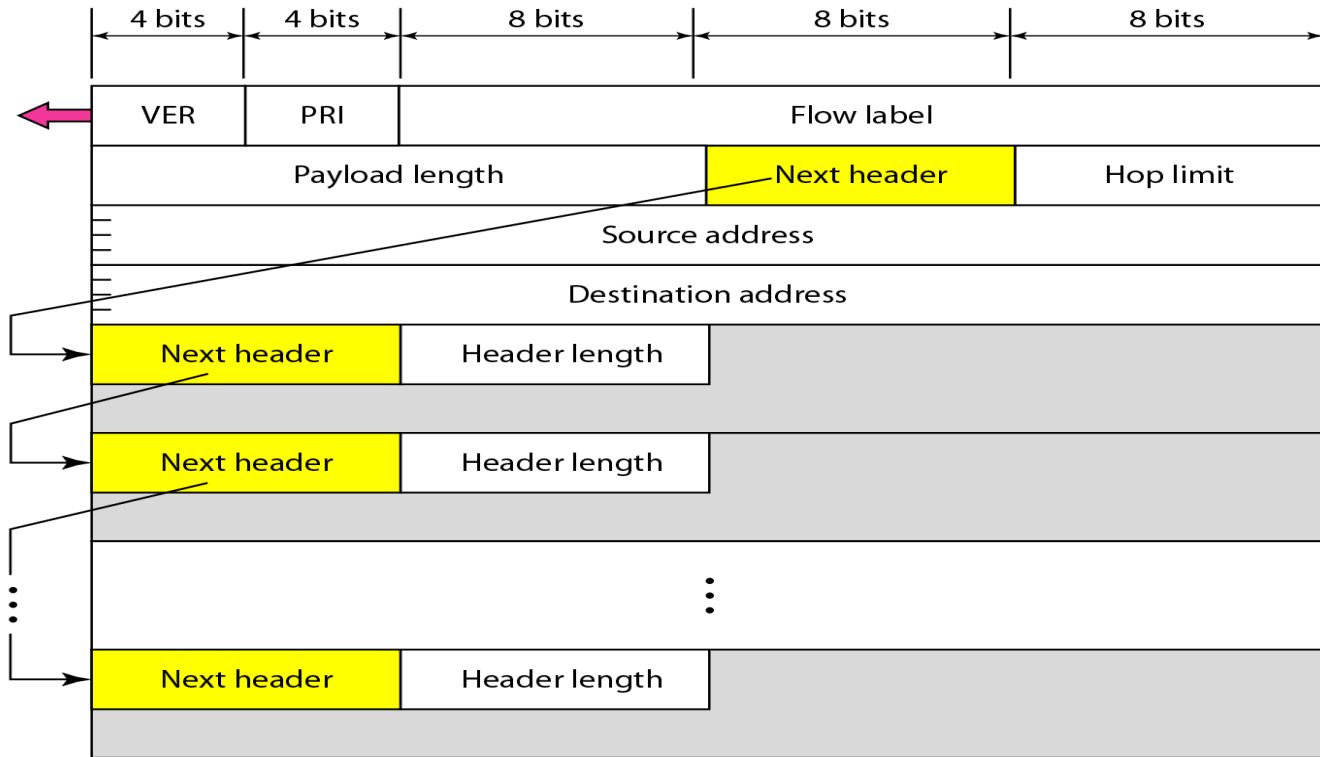
- The network layer protocol in the TCP/IP protocol suite is currently IPv4
- Although IPv4 is well designed, data communication has evolved since the inception of IPv4 in the 1970s
- IPv4 has some deficiencies that make it unsuitable for the fast-growing Internet



IPv6 datagram header and payload



Format of an IPv6 datagram



Next header codes for IPv6

<i>Code</i>	<i>Next Header</i>
0	Hop-by-hop option
2	ICMP
6	TCP
17	UDP
43	Source routing
44	Fragmentation
50	Encrypted security payload
51	Authentication
59	Null (no next header)
60	Destination option

Priorities for congestion-controlled traffic

<i>Priority</i>	<i>Meaning</i>
0	No specific traffic
1	Background data
2	Unattended data traffic
3	Reserved
4	Attended bulk data traffic
5	Reserved
6	Interactive traffic
7	Control traffic

Priorities for noncongestion-controlled traffic

<i>Priority</i>	<i>Meaning</i>
8	Data with greatest redundancy
...	...
15	Data with least redundancy

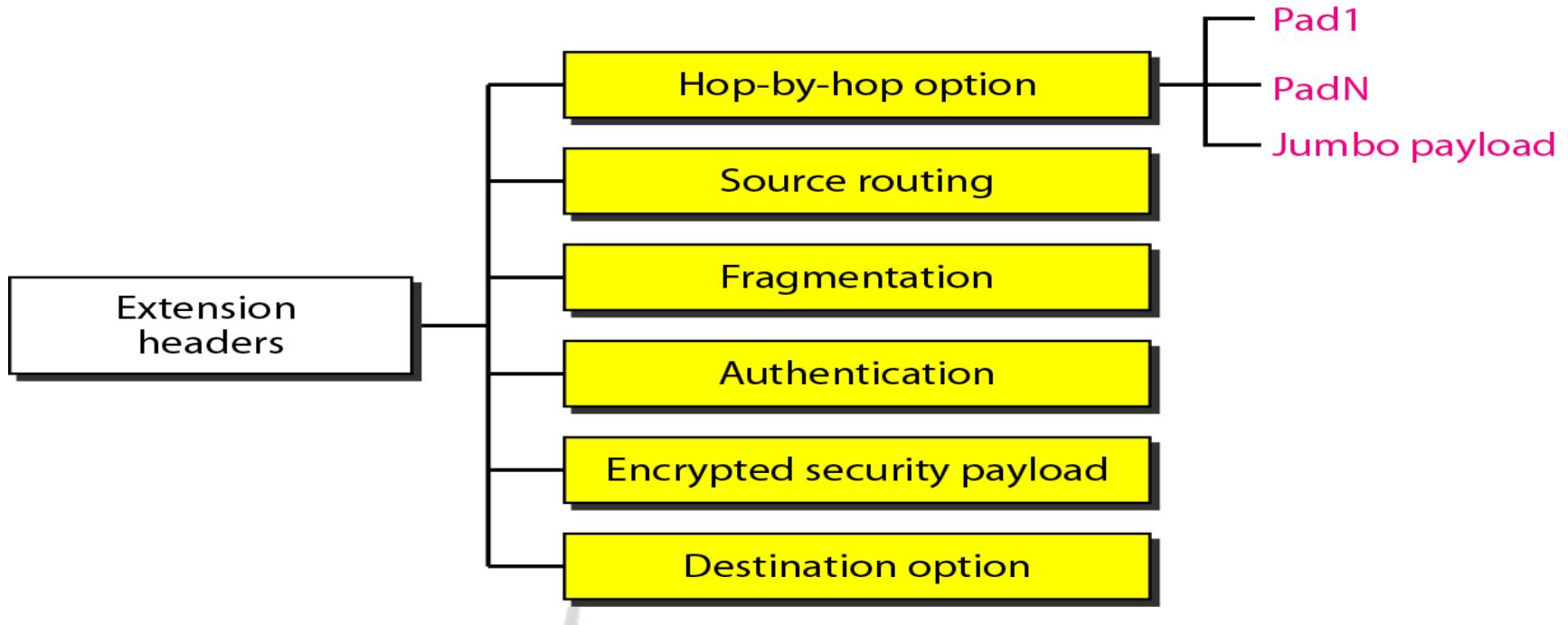


Comparison between IPv4 and IPv6 packet headers

Comparison

1. The header length field is eliminated in IPv6 because the length of the header is fixed in this version.
2. The service type field is eliminated in IPv6. The priority and flow label fields together take over the function of the service type field.
3. The total length field is eliminated in IPv6 and replaced by the payload length field.
4. The identification, flag, and offset fields are eliminated from the base header in IPv6. They are included in the fragmentation extension header.
5. The TTL field is called hop limit in IPv6.
6. The protocol field is replaced by the next header field.
7. The header checksum is eliminated because the checksum is provided by upper-layer protocols; it is therefore not needed at this level.
8. The option fields in IPv4 are implemented as extension headers in IPv6.

Extension header types



Comparison between IPv4 options and IPv6 extension headers

Comparison

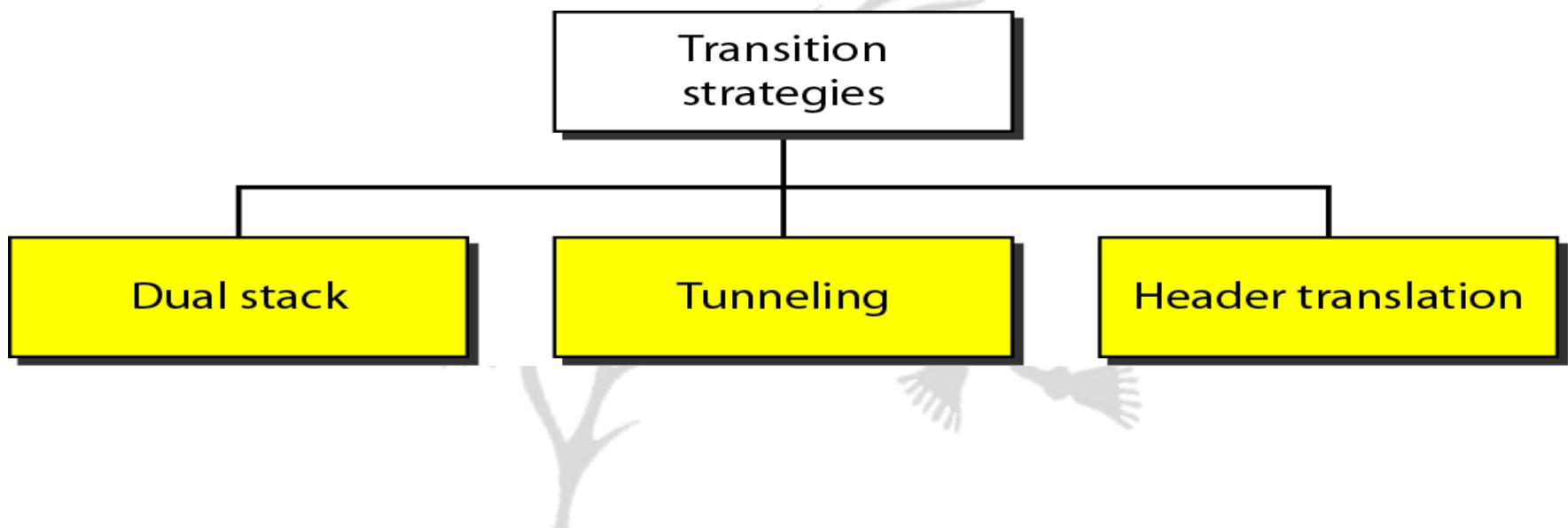
1. The no-operation and end-of-option options in IPv4 are replaced by Pad1 and PadN options in IPv6.
2. The record route option is not implemented in IPv6 because it was not used.
3. The timestamp option is not implemented because it was not used.
4. The source route option is called the source route extension header in IPv6.
5. The fragmentation fields in the base header section of IPv4 have moved to the fragmentation extension header in IPv6.
6. The authentication extension header is new in IPv6.
7. The encrypted security payload extension header is new in IPv6.



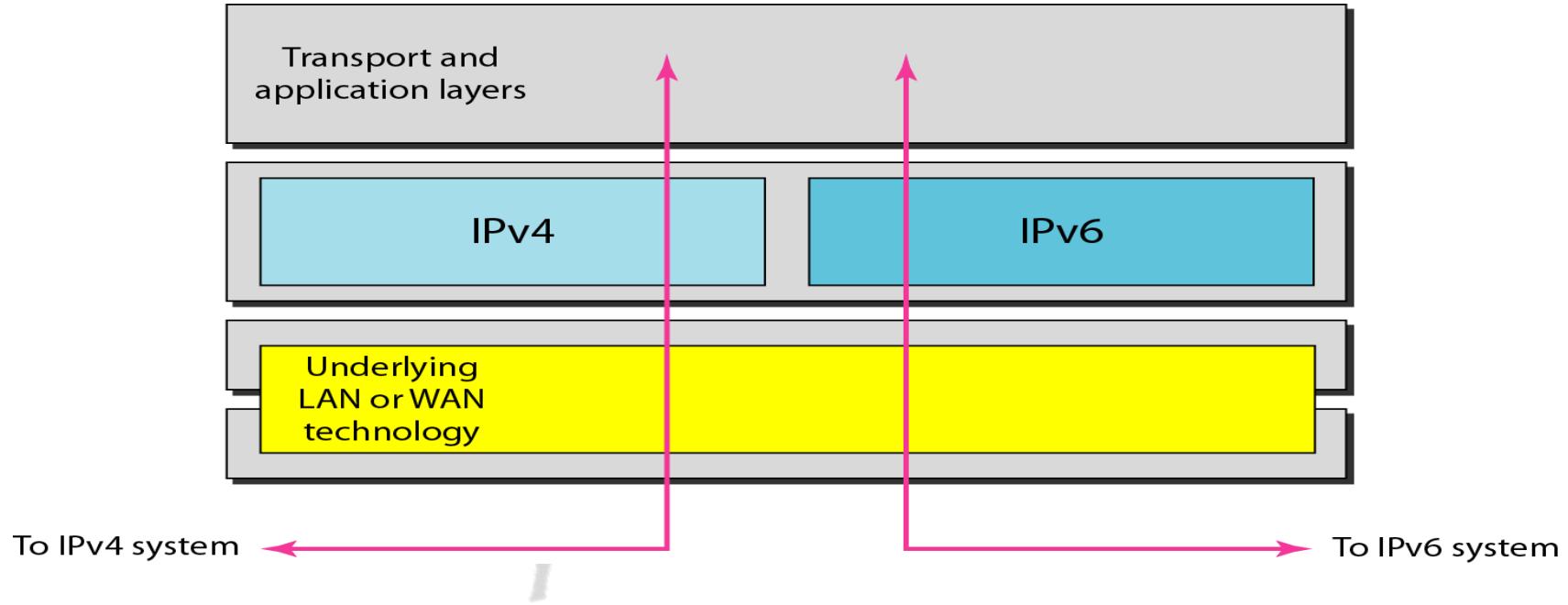
Transition from IPv4 to IPv6

- Because of the huge number of systems on the Internet, the transition from IPv4 to IPv6 cannot happen suddenly
- It takes a considerable amount of time before every system in the Internet can move from IPv4 to IPv6
- The transition must be smooth to prevent any problems between IPv4 and IPv6 systems

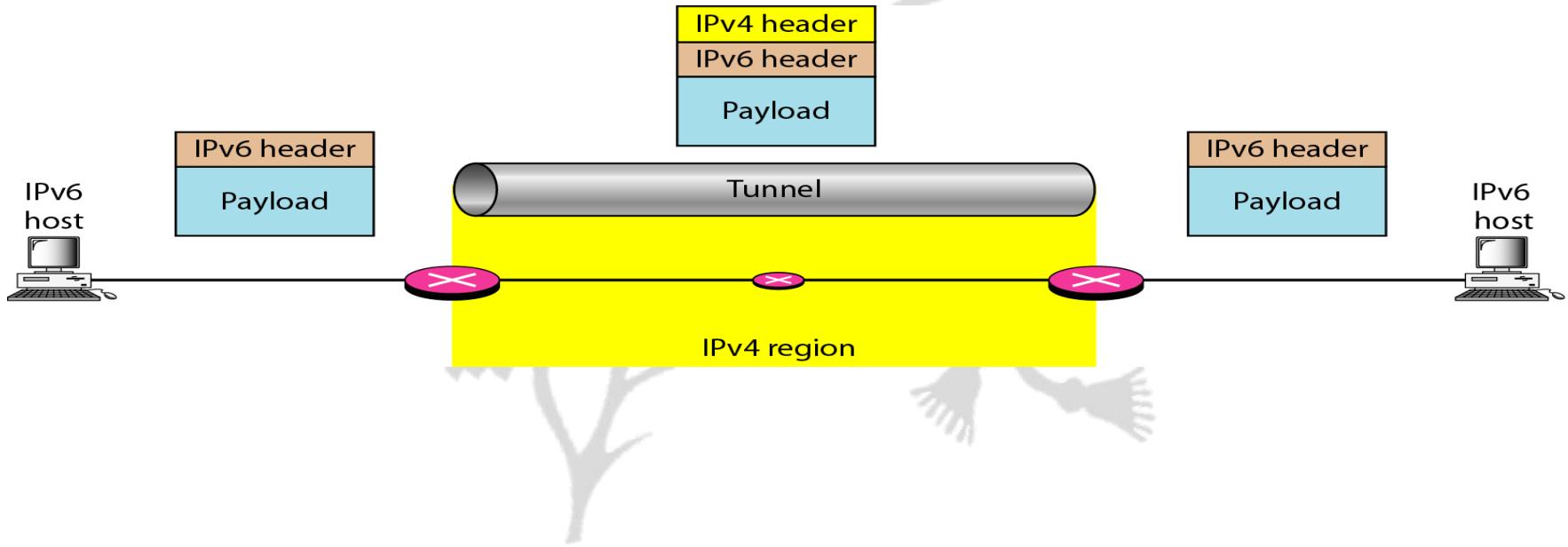
Three transition strategies



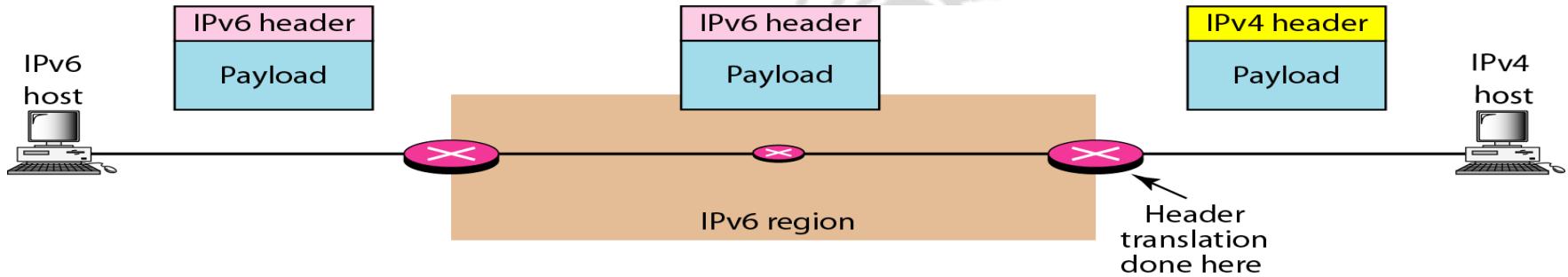
Dual stack



Tunneling strategy



Header translation strategy



Header translation

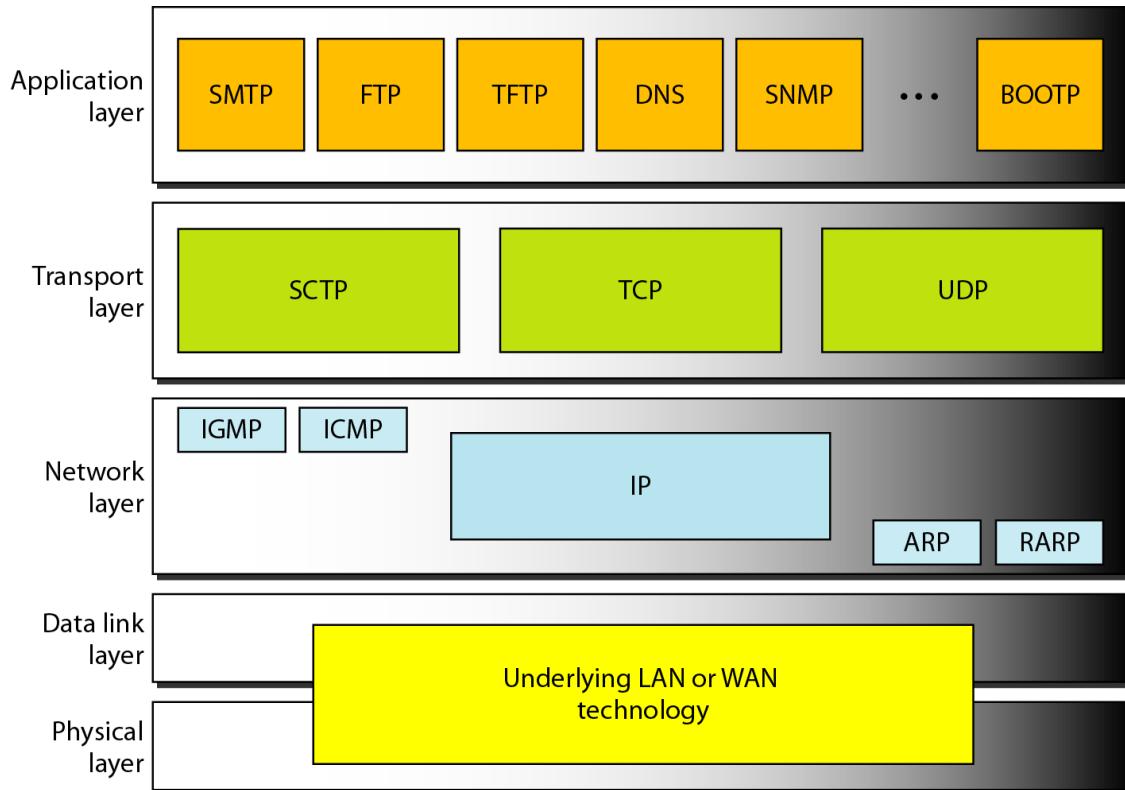
Header Translation Procedure

1. The IPv6 mapped address is changed to an IPv4 address by extracting the rightmost 32 bits.
2. The value of the IPv6 priority field is discarded.
3. The type of service field in IPv4 is set to zero.
4. The checksum for IPv4 is calculated and inserted in the corresponding field.
5. The IPv6 flow label is ignored.
6. Compatible extension headers are converted to options and inserted in the IPv4 header.
Some may have to be dropped.
7. The length of IPv4 header is calculated and inserted into the corresponding field.
8. The total length of the IPv4 packet is calculated and inserted in the corresponding field.



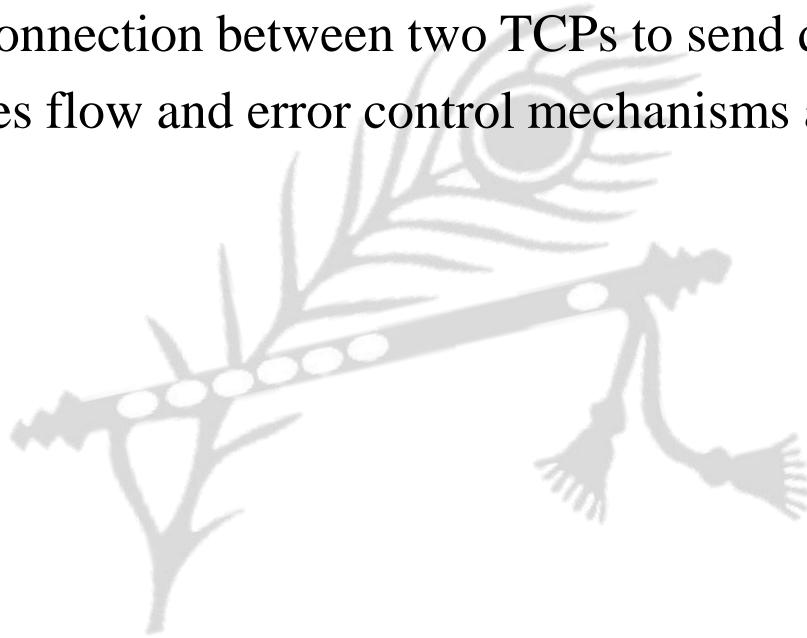
TRANSPORT LAYER

Introduction

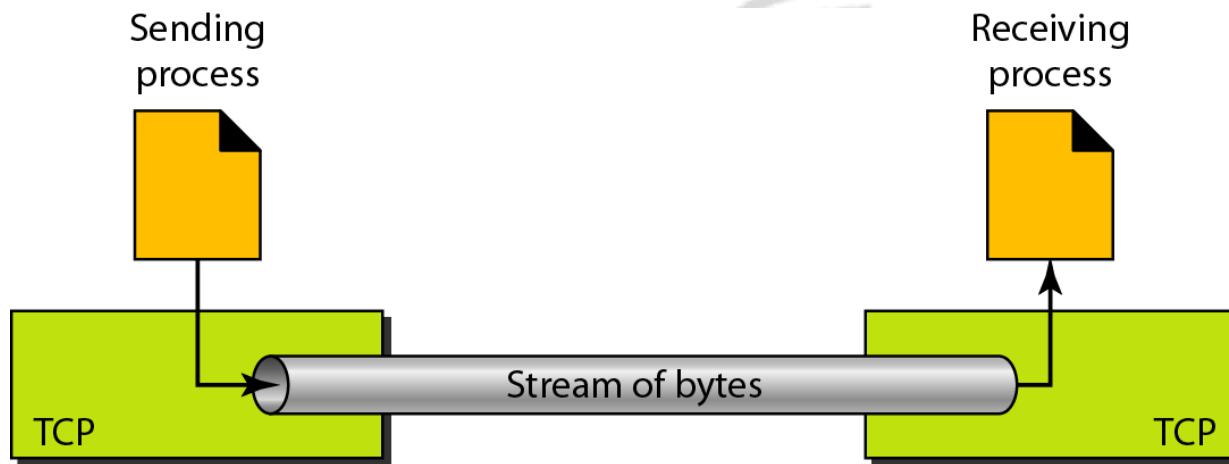


TCP

- TCP is a connection-oriented protocol
- It creates a virtual connection between two TCPs to send data
- In addition, TCP uses flow and error control mechanisms at the transport level

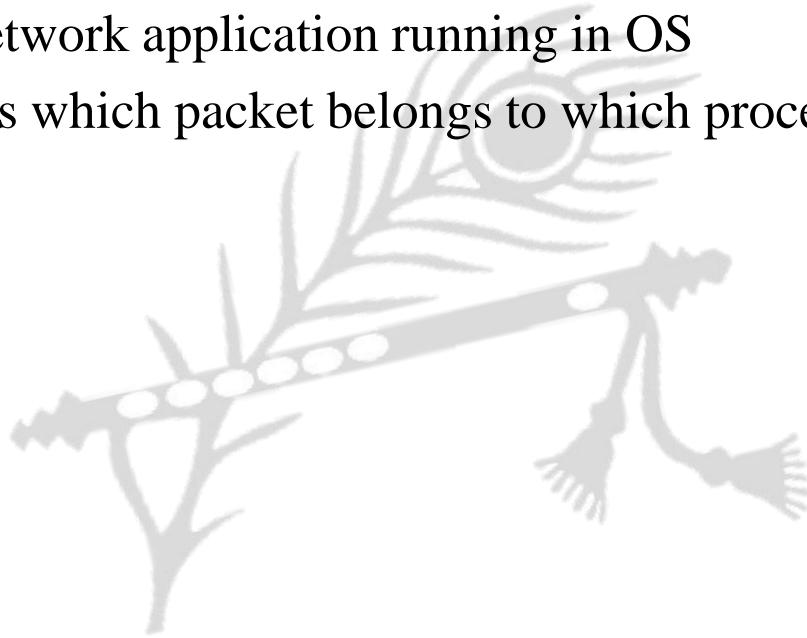


Stream delivery



Why we need transport layer?

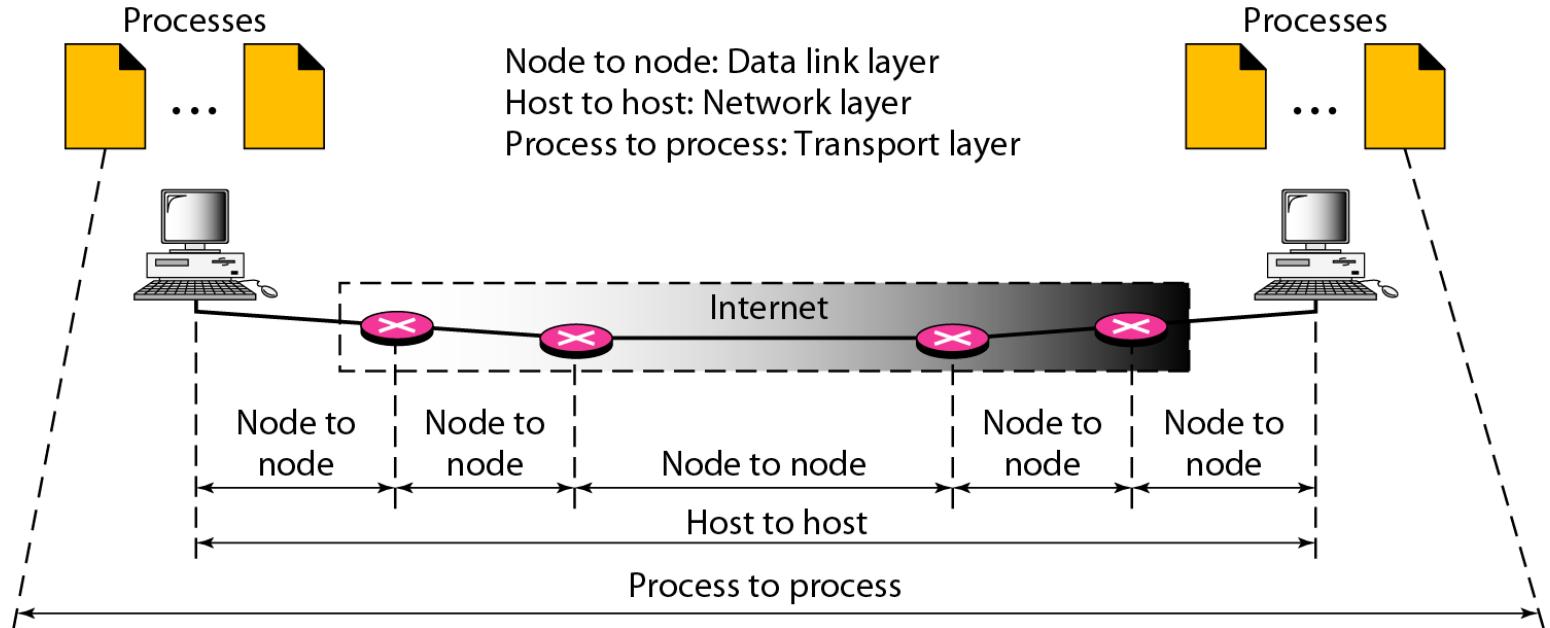
- Network layer is responsible for host to host communication (IP address)
- There are several network application running in OS
- How did NIC knows which packet belongs to which process/application?



Why we need transport layer?

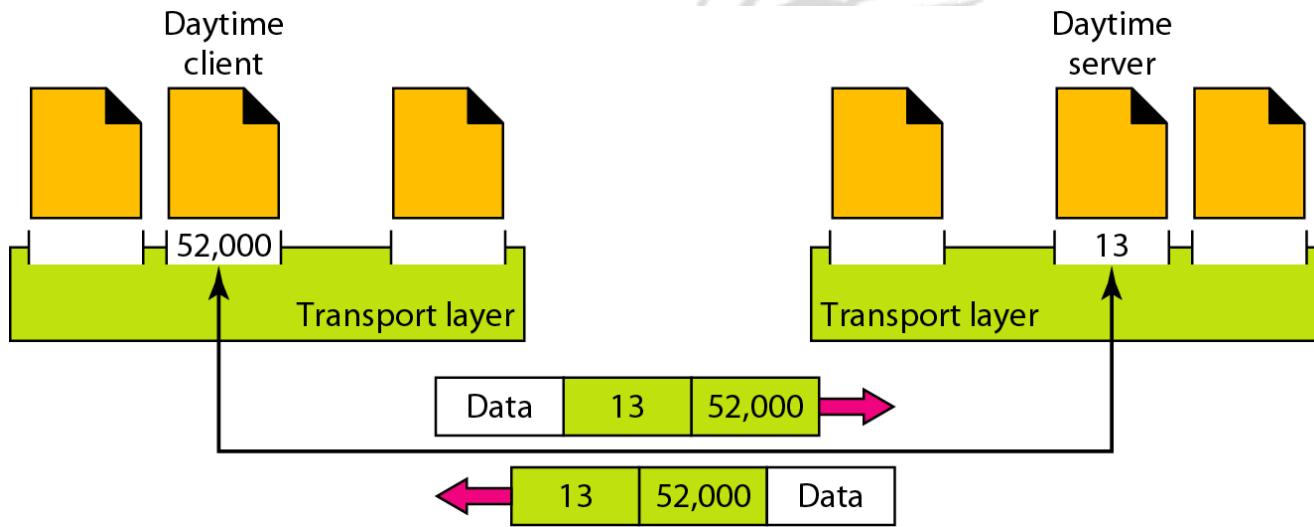
- The transport layer is responsible for process-to-process delivery
 - the delivery of a packet, part of a message, from one process to another
- It provides logical communication between app processes running on different hosts
- transport protocols run in end systems
 - send side: breaks app messages into segments, passes to network layer
 - rcv side: reassembles segments into messages, passes to app layer
- more than one transport protocol available to apps
 - Internet: TCP and UDP

Types of data deliveries: Internet Stack

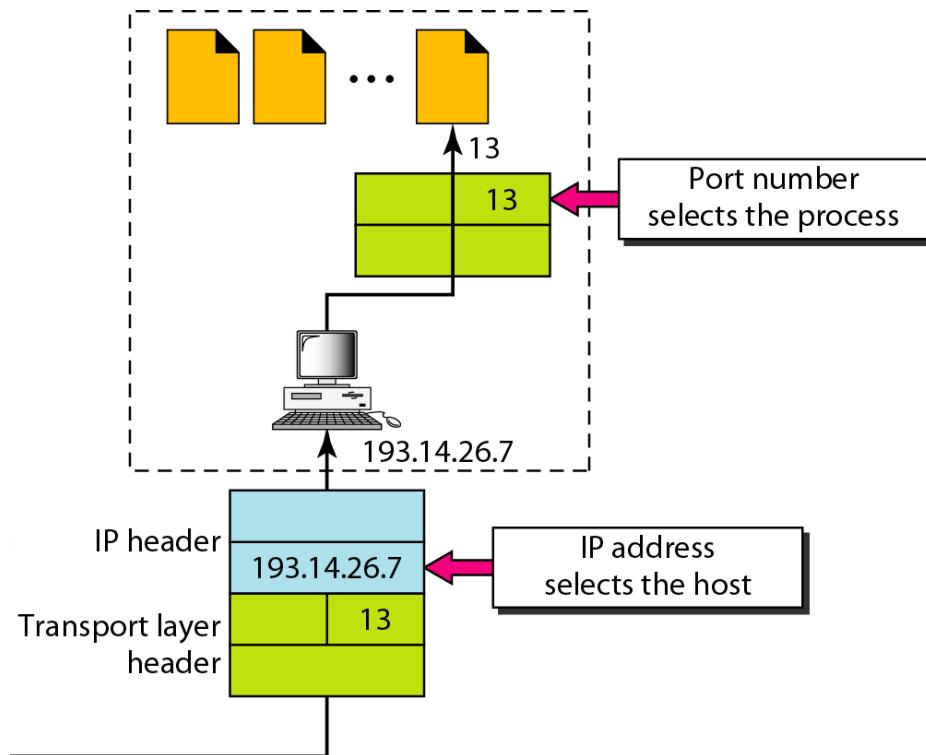


How it delivers messages to specific process

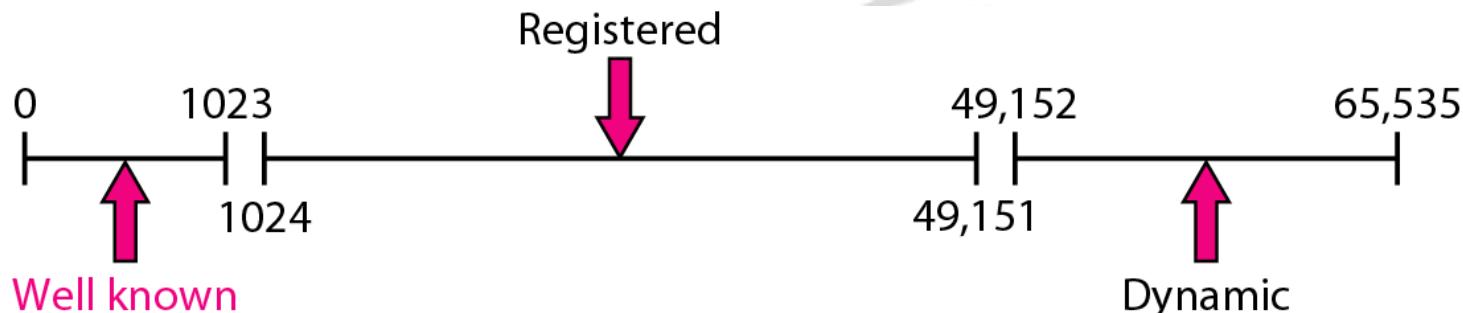
Using Port address (16 bit)
Ranges from 0 to 65,535



IP addresses v/s port numbers



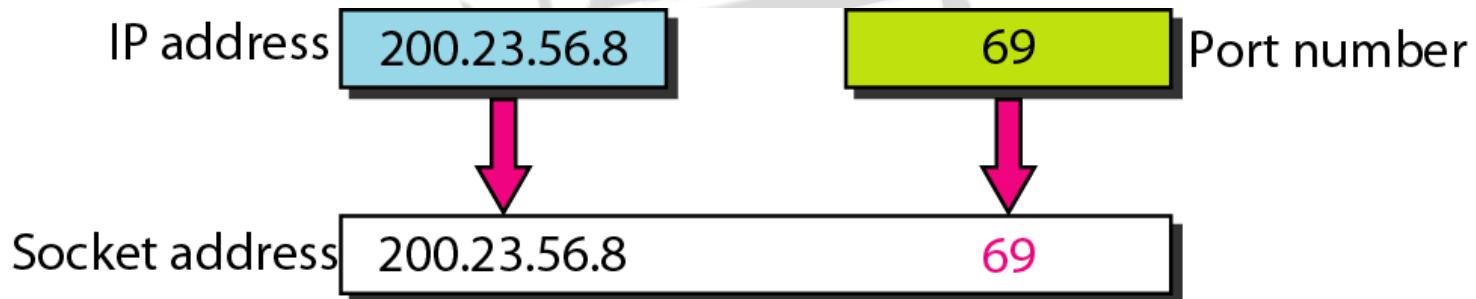
LANA ranges



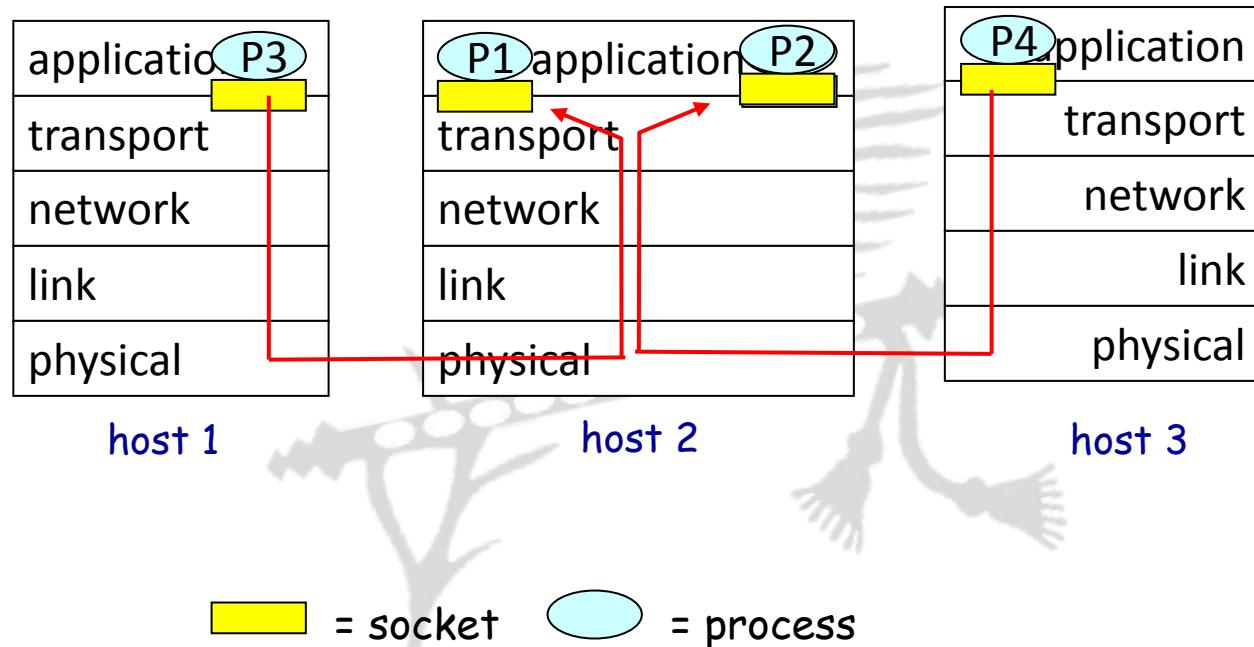
Socket address (IP address + Port Address)

What is the use of socket?

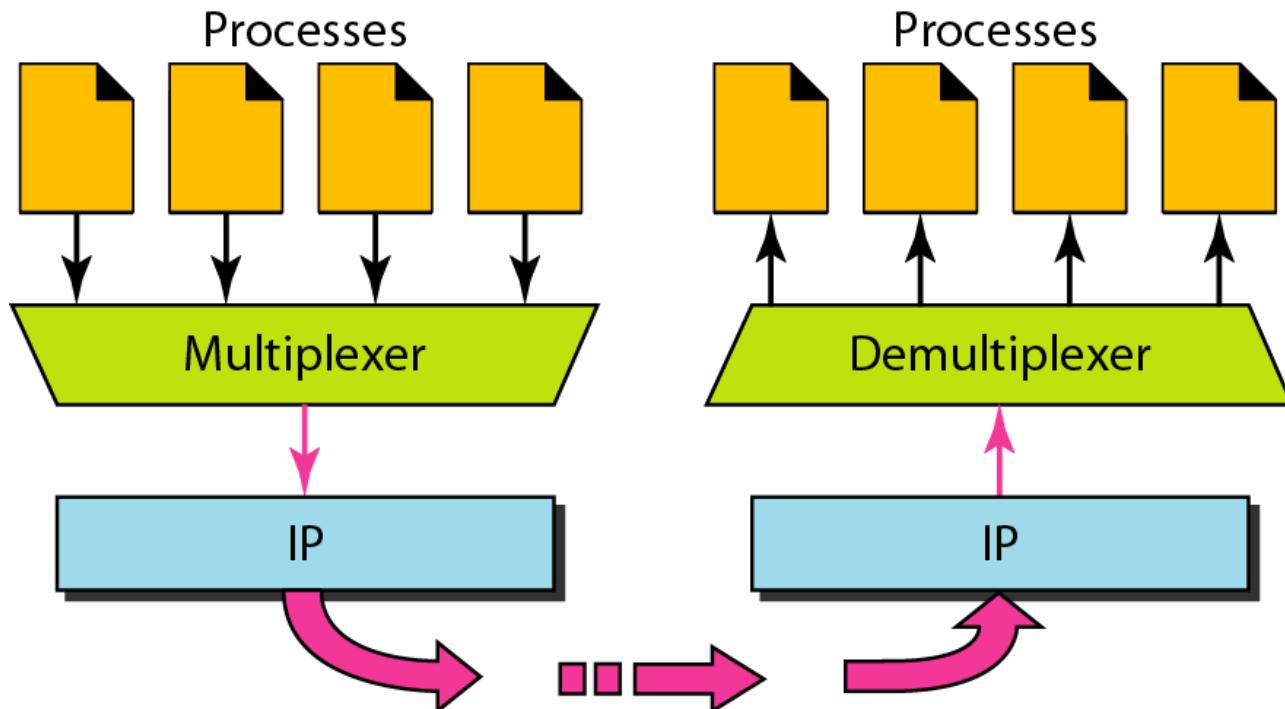
- The socket mechanism provides a means of inter-process communication (IPC)
- Socket is basically an API for enabling communication between two end points
- A socket is one endpoint of a two way communication link between two programs running on the network



Socket API



Multiplexing and De-multiplexing



Multiplexing and De-multiplexing

Demultiplexing at rcv host:

delivering received segments
to correct socket



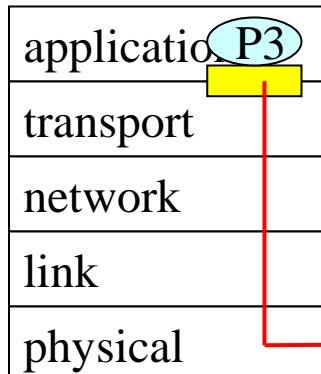
= socket



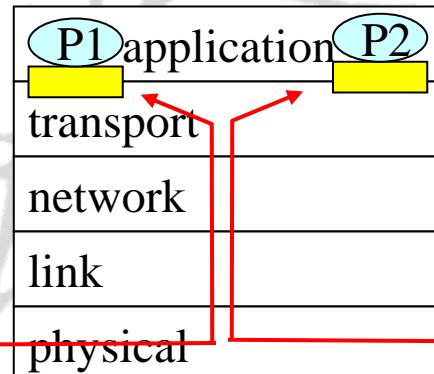
= process

Multiplexing at send host:

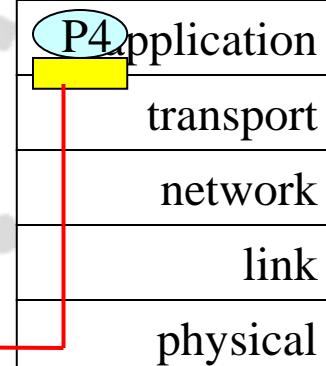
gathering data from multiple
sockets, enveloping data with
header (later used for
De-multiplexing)



host 1

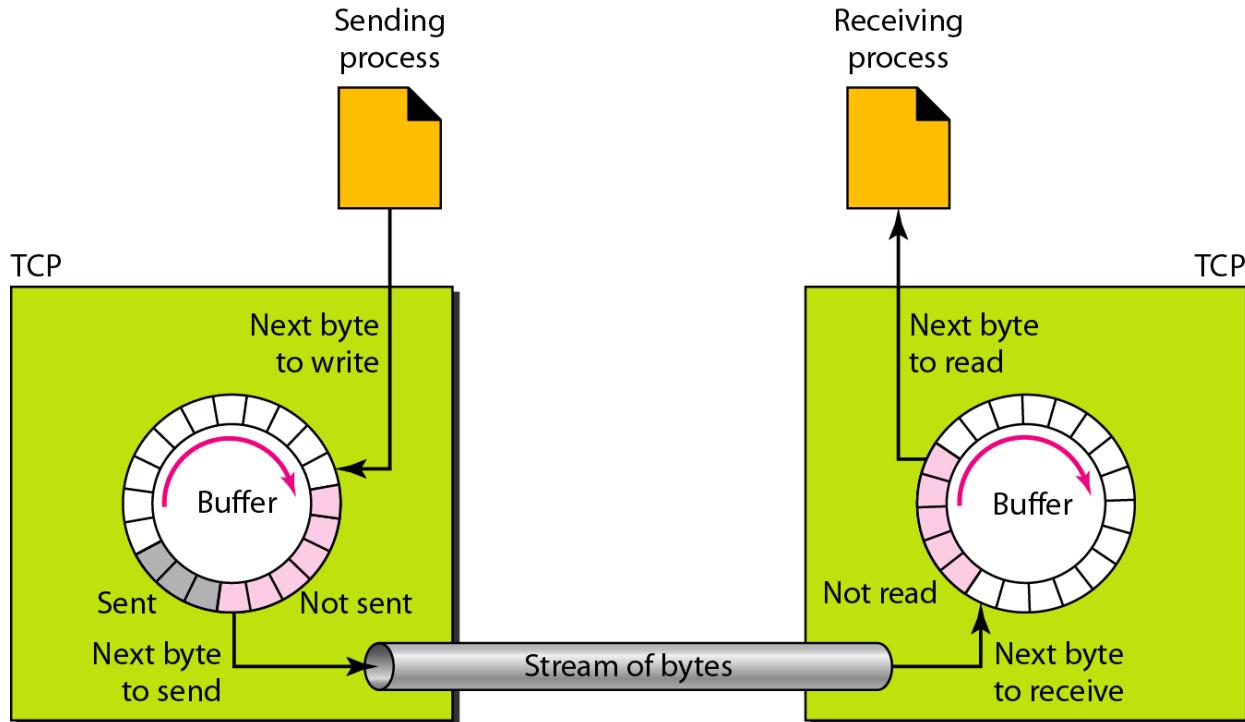


host 2

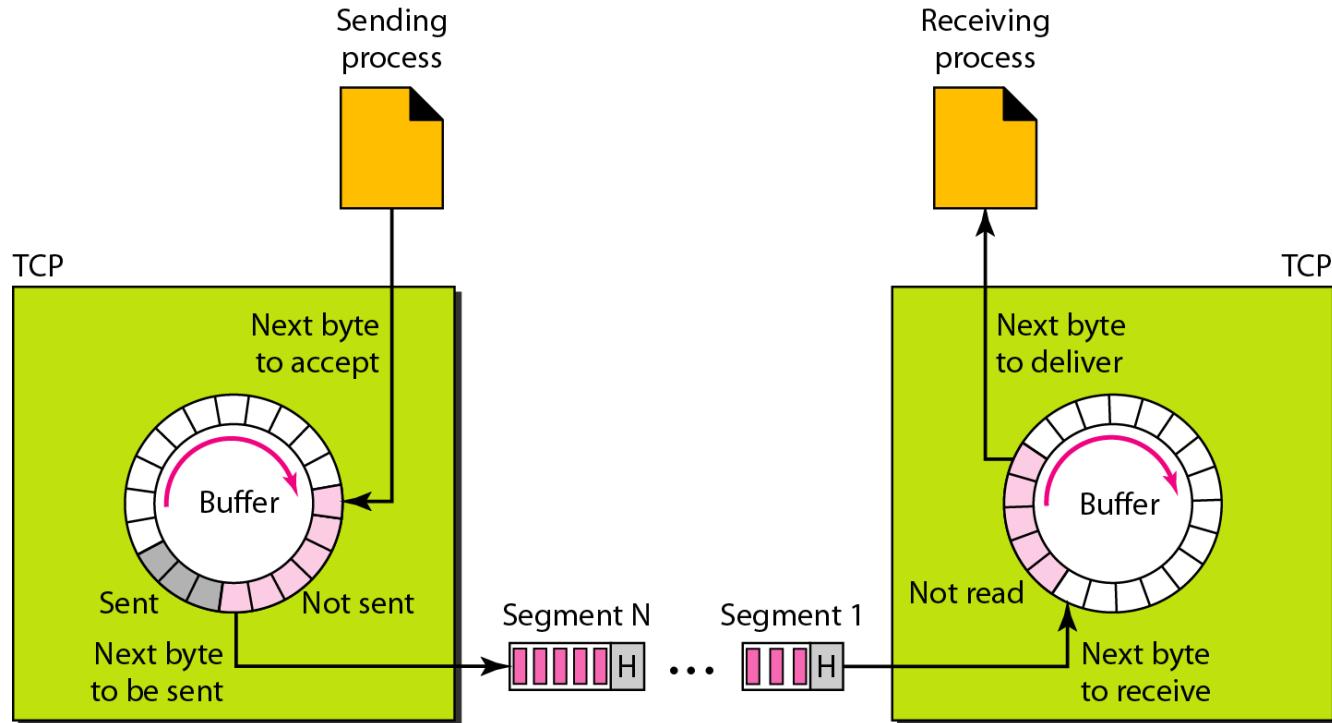


host 3

Multiplexing and demultiplexing

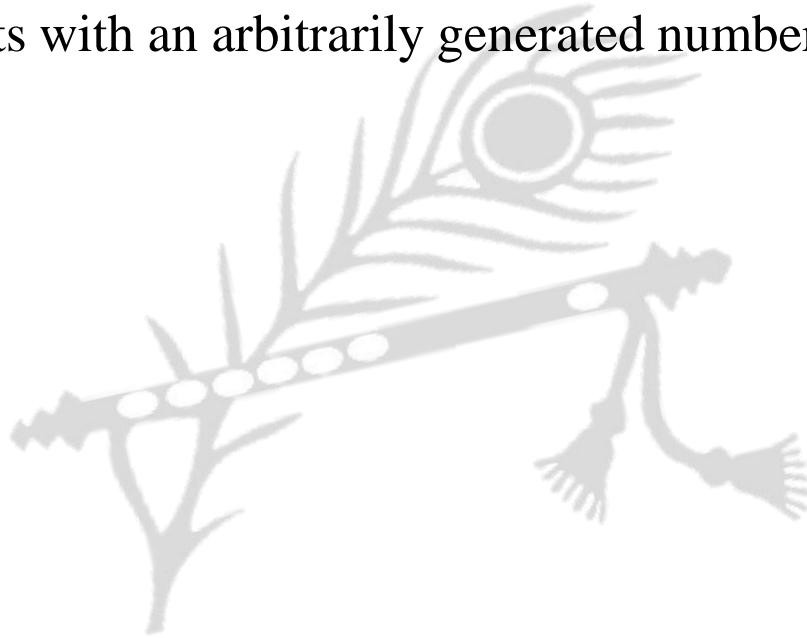


TCP segments



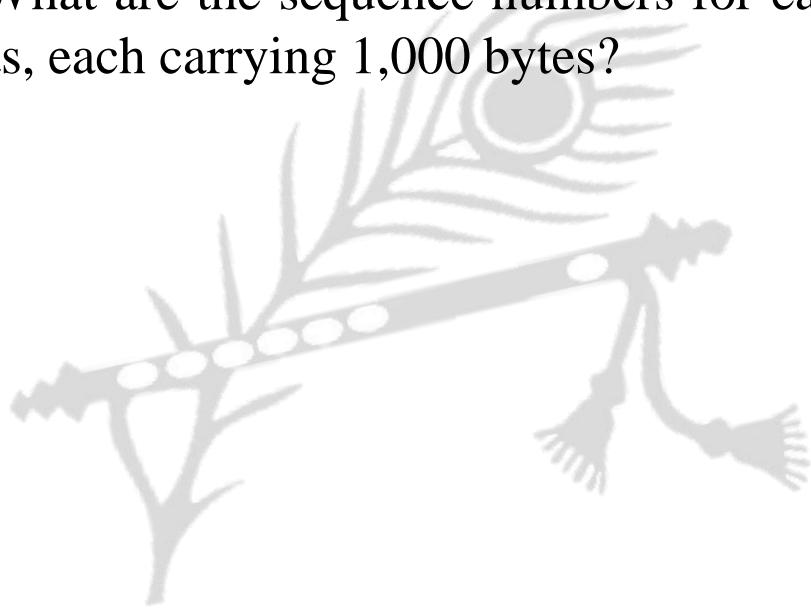
TCP segments

- The bytes of data being transferred in each connection are numbered by TCP
- The numbering starts with an arbitrarily generated number



Example

- Suppose a TCP connection is transferring a file of 5,000 bytes. The first byte is numbered 10,001. What are the sequence numbers for each segment if data are sent in five segments, each carrying 1,000 bytes?

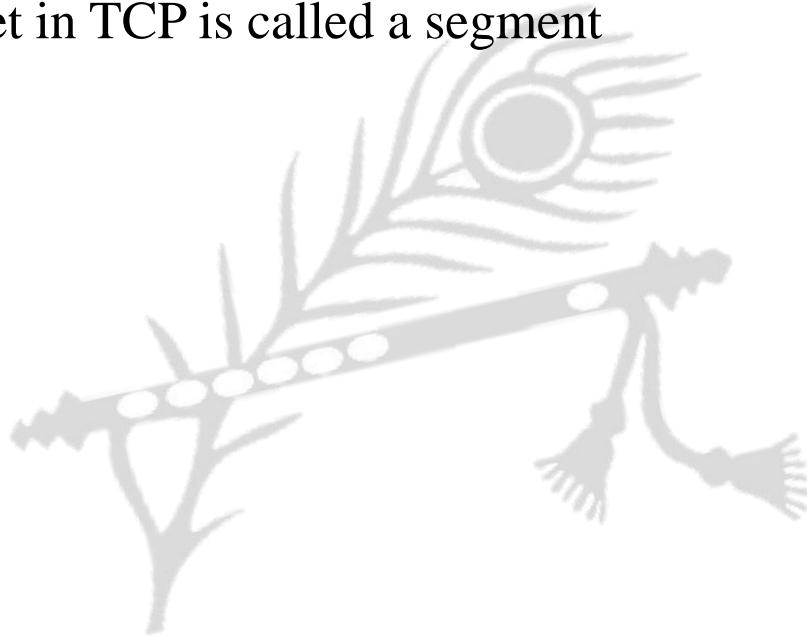


Example

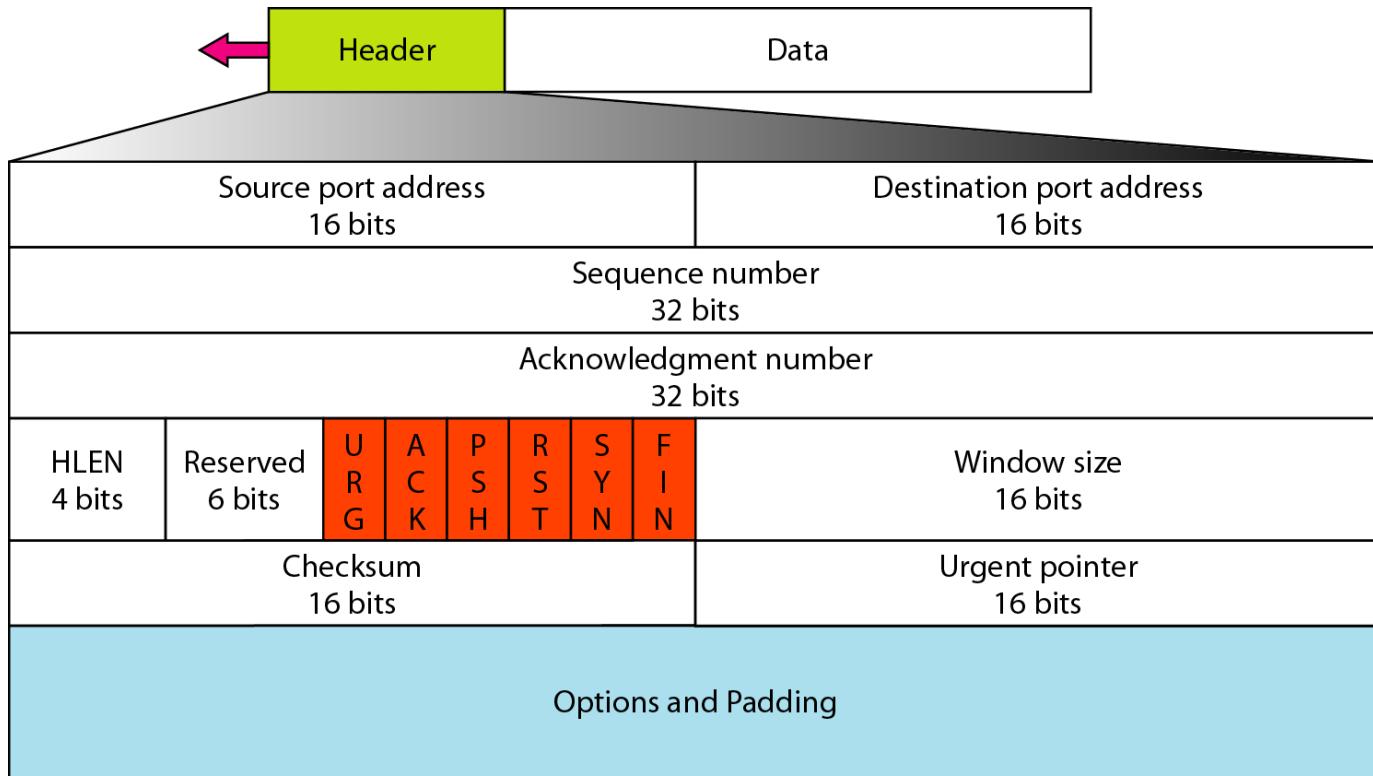
- The value in the sequence number field of a segment defines the number assigned to the first data byte contained in that segment
- The value of the acknowledgment field in a segment defines the number of the next byte a party expects to receive
- The acknowledgment number is cumulative

Segment

- Before discussing TCP in more detail, let us discuss the TCP packets themselves. A packet in TCP is called a segment



TCP segment format



Control field

URG: Urgent pointer is valid

ACK: Acknowledgment is valid

PSH: Request for push

RST: Reset the connection

SYN: Synchronize sequence numbers

FIN: Terminate the connection

URG	ACK	PSH	RST	SYN	FIN
-----	-----	-----	-----	-----	-----

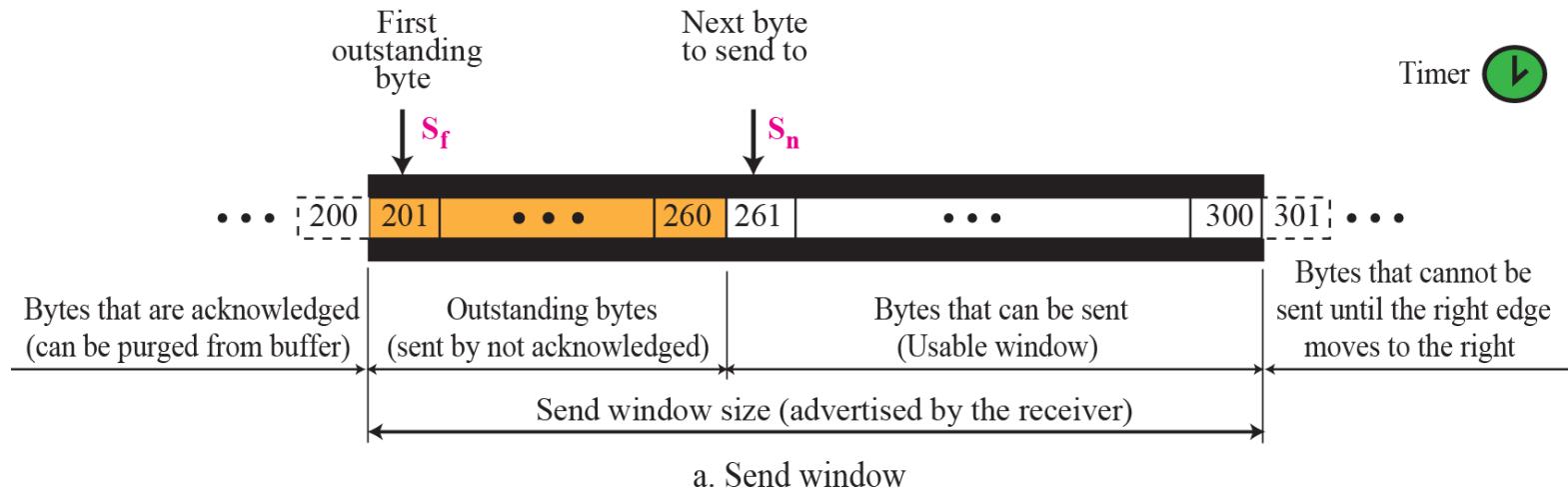
Description of flags in the control field

<i>Flag</i>	<i>Description</i>
URG	The value of the urgent pointer field is valid.
ACK	The value of the acknowledgment field is valid.
PSH	Push the data.
RST	Reset the connection.
SYN	Synchronize sequence numbers during connection.
FIN	Terminate the connection.

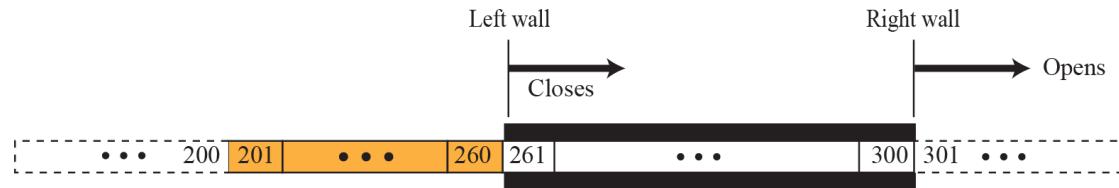
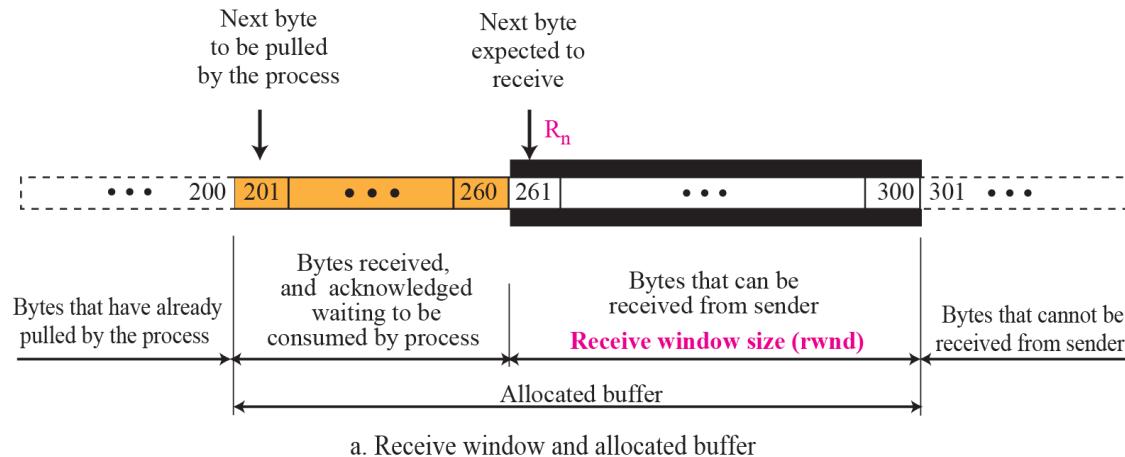
TCP Window Management

- TCP uses two windows (send window and receive window) for each direction of data transfer, which means four windows for a bidirectional communication
- To make the discussion simple, we make an assumption that communication is only unidirectional
- The bidirectional communication can be inferred using two unidirectional communications with piggybacking

Send window in TCP



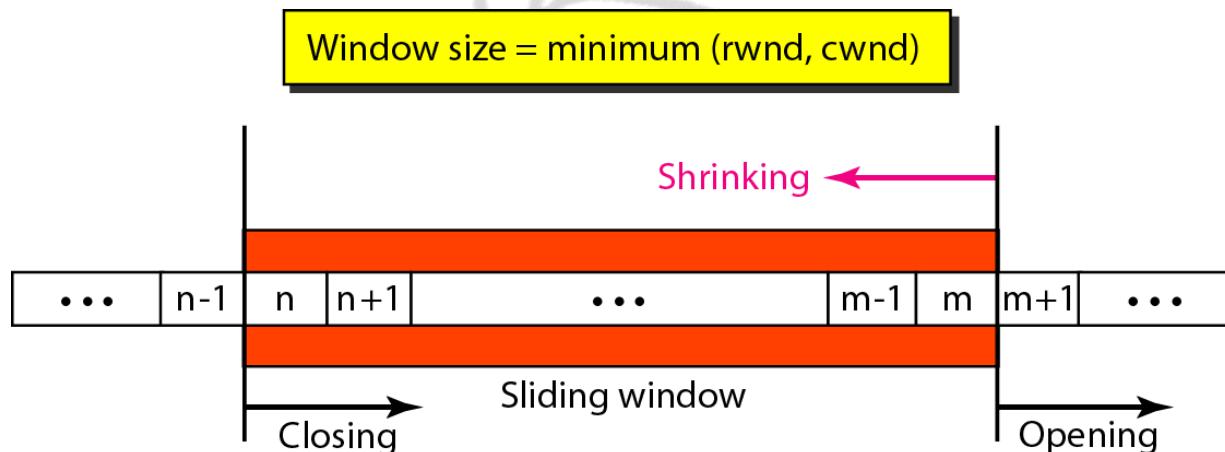
Receive window in TCP



b. Opening and closing of receive window

Sliding window

- A sliding window is used to make transmission more efficient as well as to control the flow of data so that the destination does not become overwhelmed with data
- TCP sliding windows are byte-oriented



Example

- What is the value of the receiver window (rwnd) for host A if the receiver host B has a buffer size of 5000 bytes and 1000 bytes of received and unprocessed data?

Solution

- The value of rwnd = $5000 - 1000 = 4000$
- Host B can receive only 4000 bytes of data before overflowing its buffer. Host B advertises this value in its next segment to A

Example

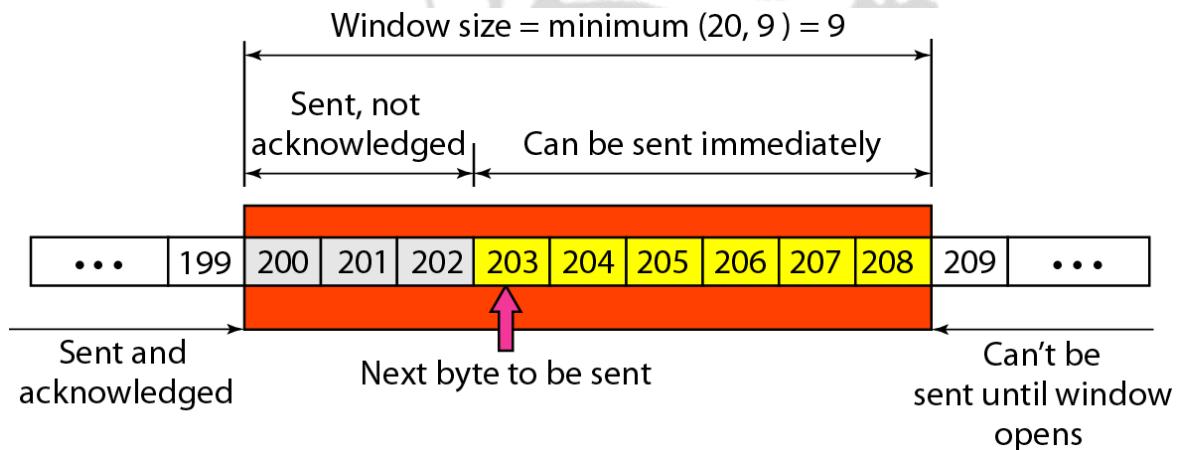
- What is the size of the window for host A if the value of rwnd is 3000 bytes and the value of cwnd is 3500 bytes?

Solution

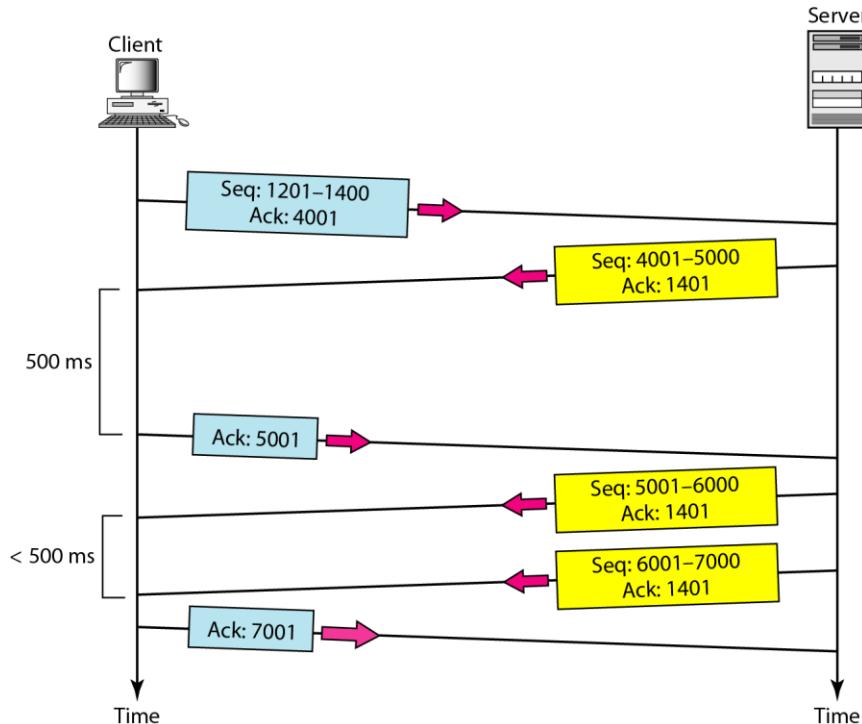
- The size of the window is the smaller of rwnd and cwnd, which is 3000 bytes

Example

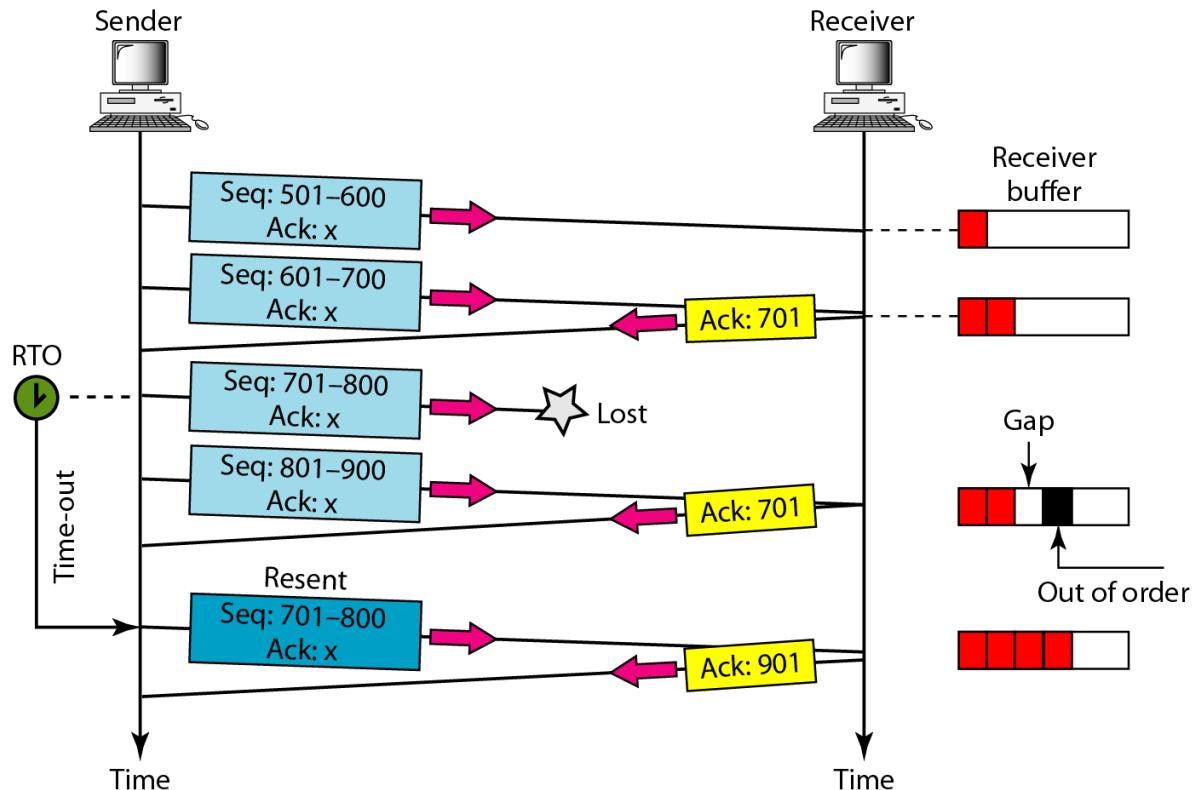
- The sender has sent bytes up to 202. We assume that cwnd is 20. The receiver has sent an acknowledgment number of 200 with an rwnd of 9 bytes. The size of the sender window is the minimum of rwnd and cwnd, or 9 bytes. Bytes 200 to 202 are sent, but not acknowledged. Bytes 203 to 208 can be sent without worrying about acknowledgment. Bytes 209 and above cannot be sent.



Normal operation

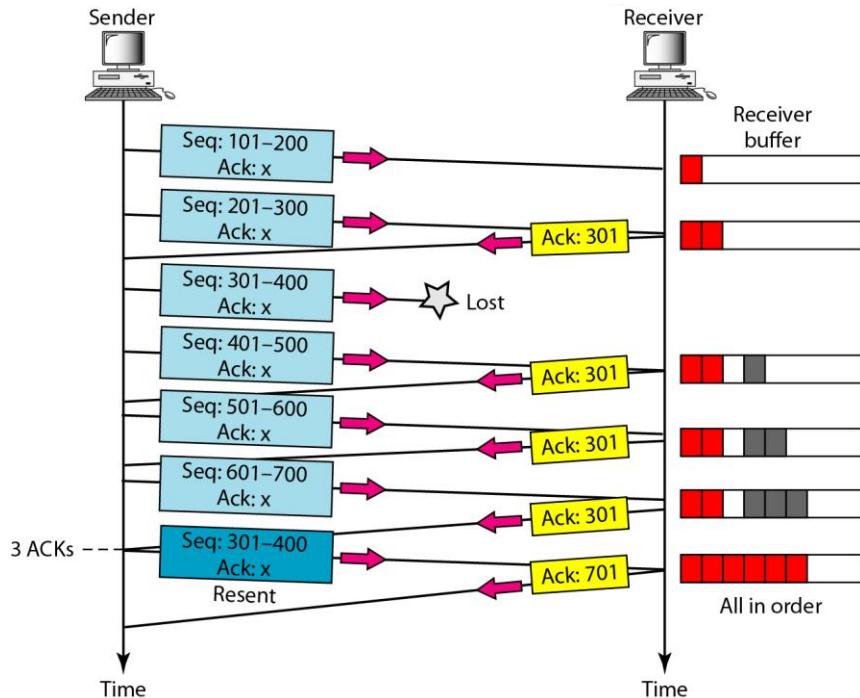


Lost segment



Fast retransmission

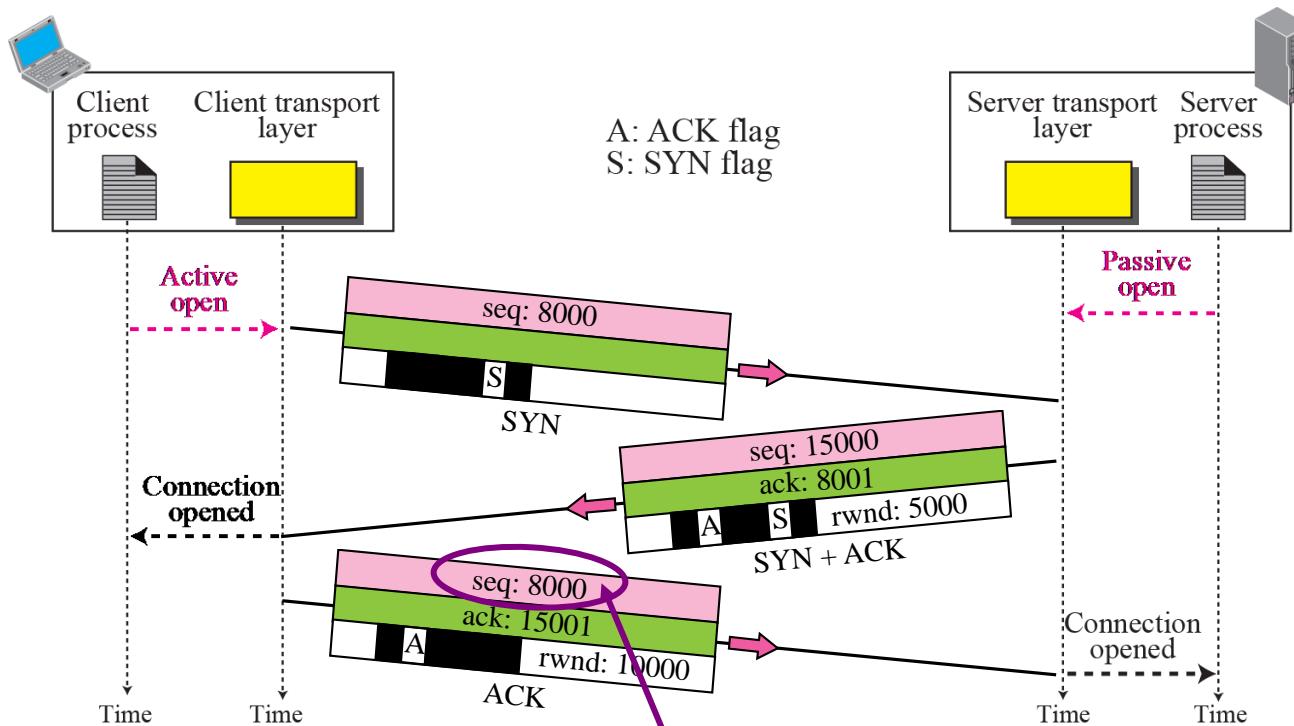
- The receiver TCP delivers only ordered data to the process



TCP connection

- TCP is connection-oriented
- It establishes a virtual path between the source and destination
- All of the segments belonging to a message are sent over this virtual path
- TCP, which uses the services of IP, a connectionless protocol, can be connection-oriented
- The point is that a TCP connection is virtual, not physical
- TCP operates at a higher level
- TCP uses the services of IP to deliver individual segments to the receiver, but it controls the connection itself
- If a segment is lost or corrupted, it is retransmitted

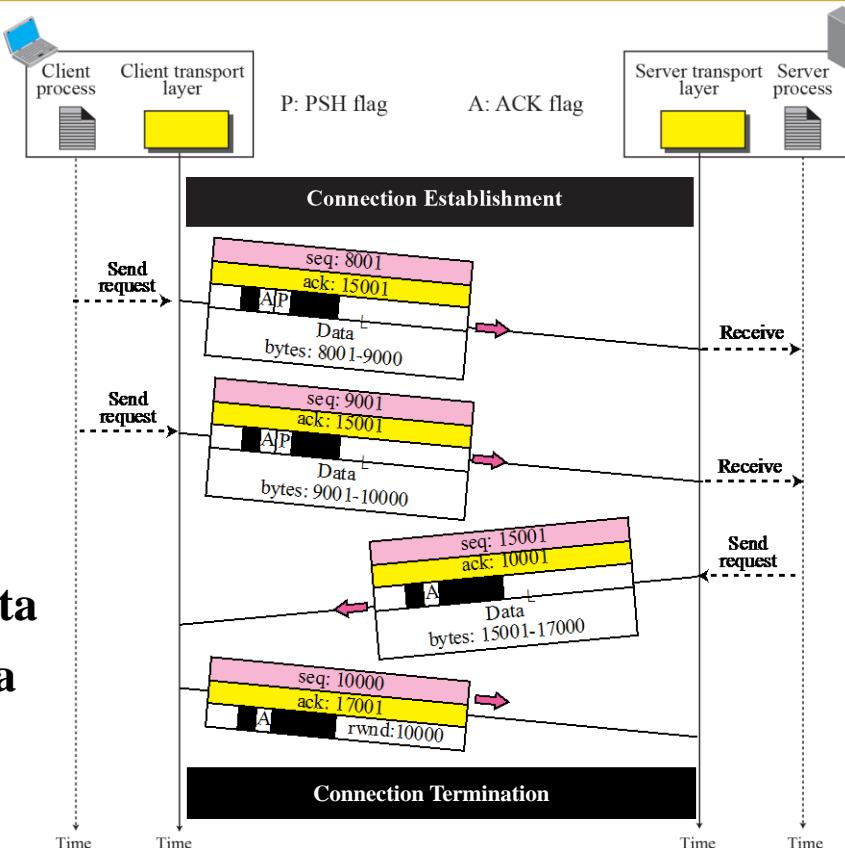
Connection establishment using three-way handshake



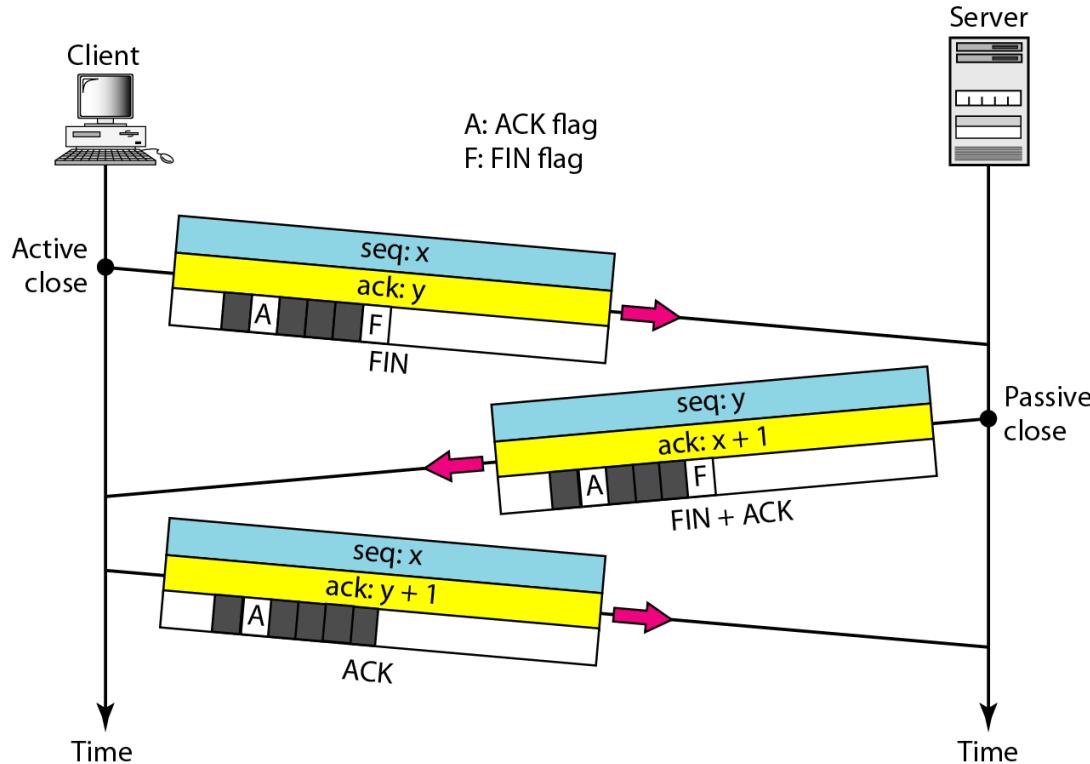
Means “no data” !
seq: 8001 if piggybacking

Data transfer

Pushing data
Urgent data

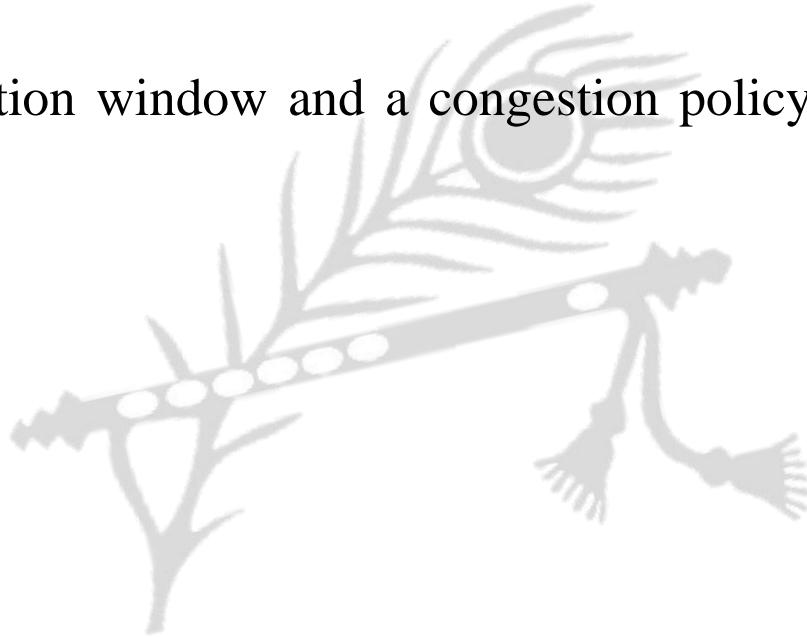


Connection termination using three-way handshaking



Congestion control

- Congestion control in TCP is based on both open loop and closed-loop mechanisms
- TCP uses a congestion window and a congestion policy that avoid congestion and detect



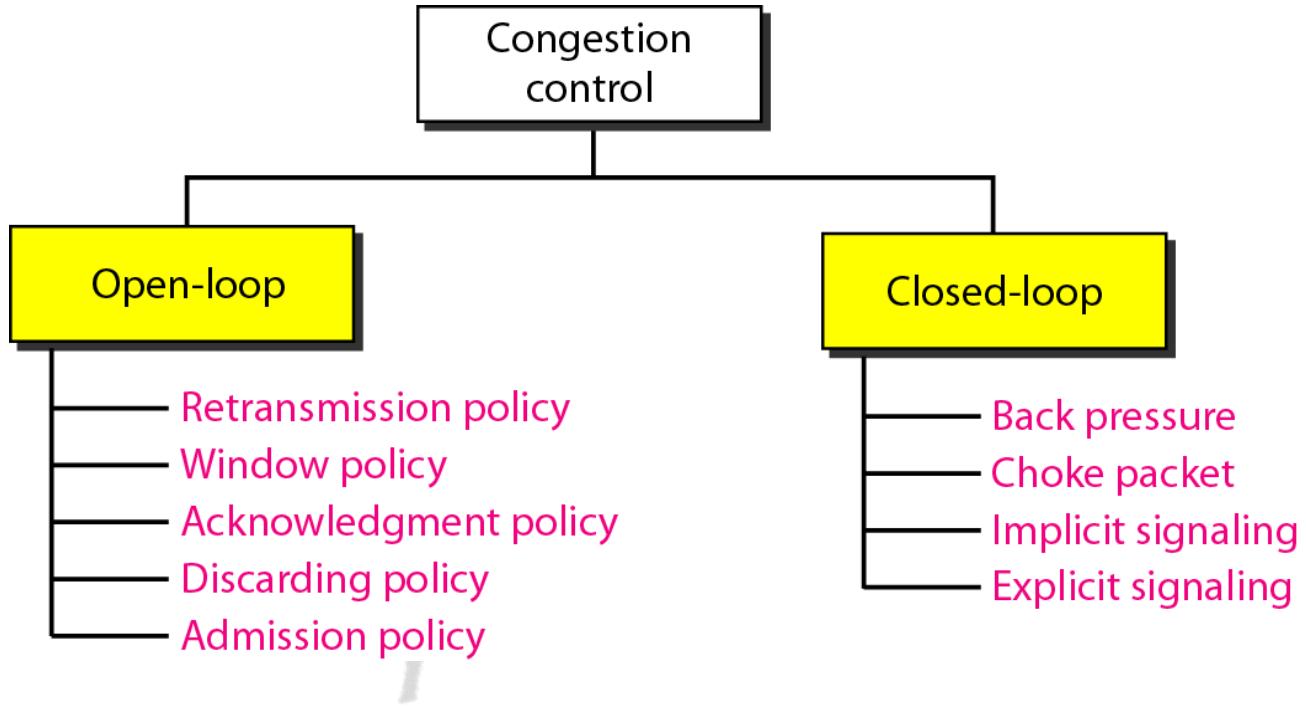
Congestion control

- When too many packets are present in the subnet, performance degrades, this situation is called congestion
- At very high traffic, performance collapses completely and almost no packets are delivered
- Reasons of Congestion:
 - Slow Processor
 - High stream of packets sent from one of the sender
 - Insufficient memory
 - Low bandwidth lines
- Congestion control and flow control are often confused but both helps reduce congestion

Congestion control

- Dividing all algorithms into
 - open loop
 - They further divide the open loop algorithms into ones that act at the source versus ones that act at the destination
 - closed loop
 - The closed loop algorithms are also divided into two subcategories:
 - In explicit feedback algorithms, packets are sent back from the point of congestion to warn the source
 - In implicit algorithms, the source deduces the existence of congestion by making local observations, such as the time needed for acknowledgements to come back
- The presence of congestion means that the load is (temporarily) greater than the resources can handle

Congestion control categories



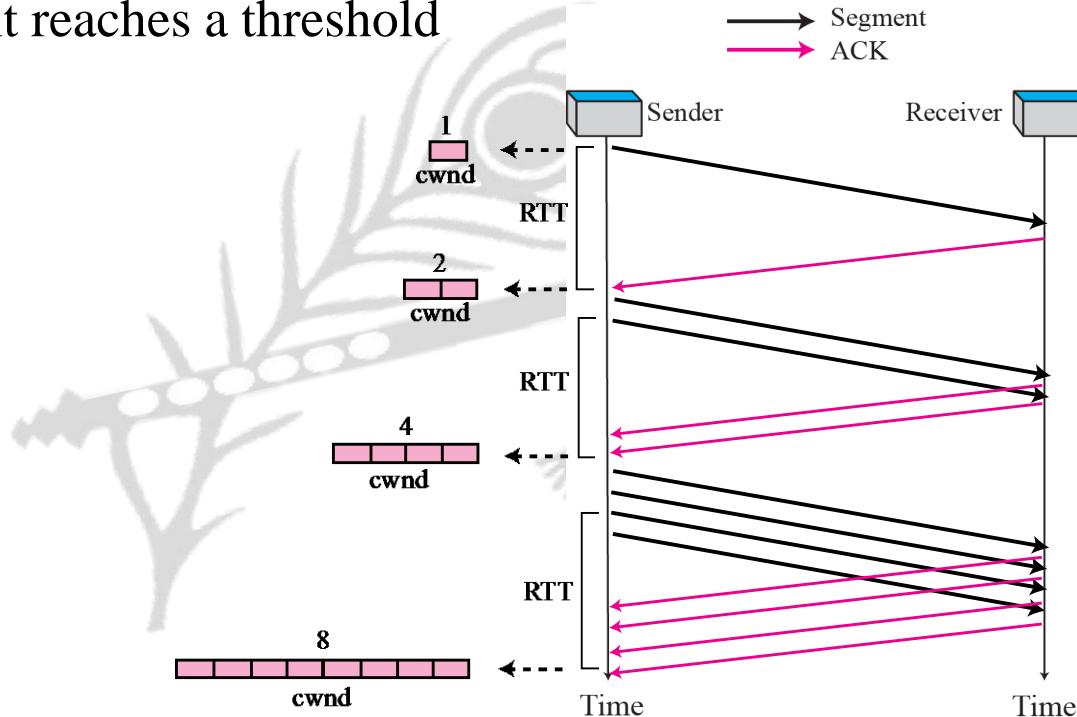
Congestion control in TCP

- Slow Start
- Additive Increase (Congestion Avoidance)
- Multiplicative decrease



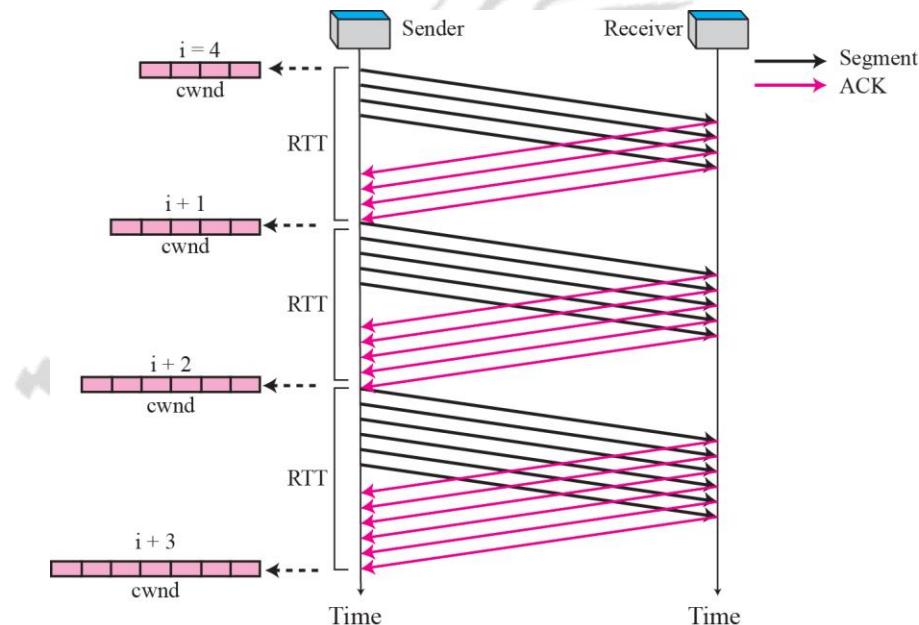
Slow start, exponential increase

- In the slow start algorithm, the size of the congestion window increases exponentially until it reaches a threshold

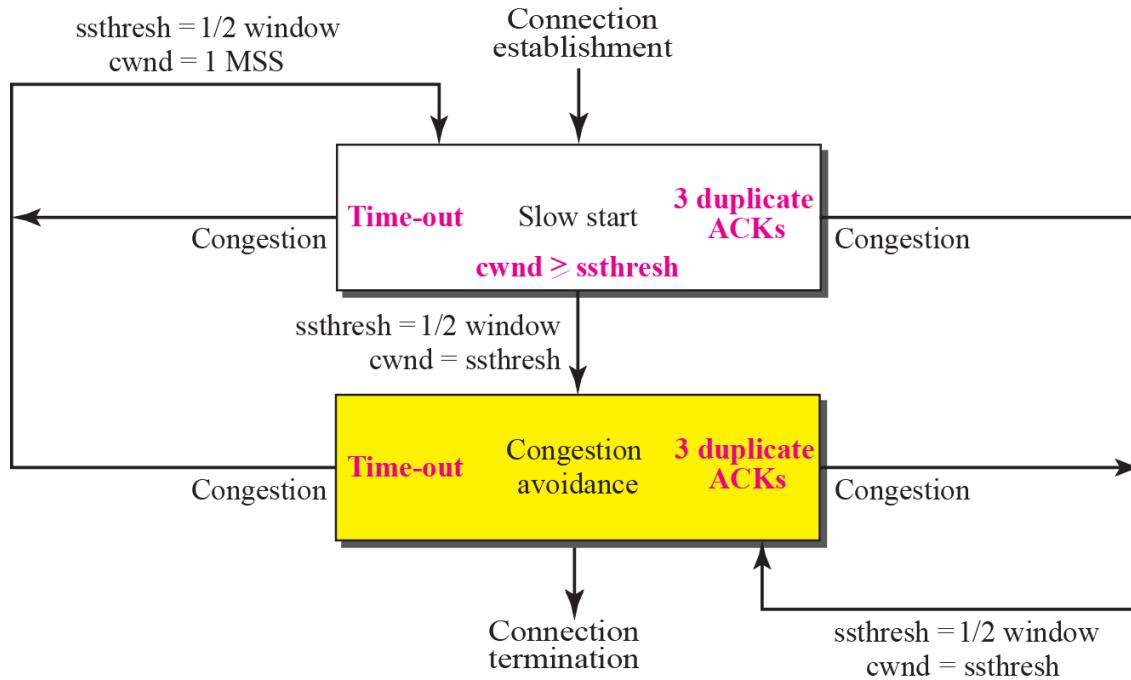


Congestion avoidance, additive increase

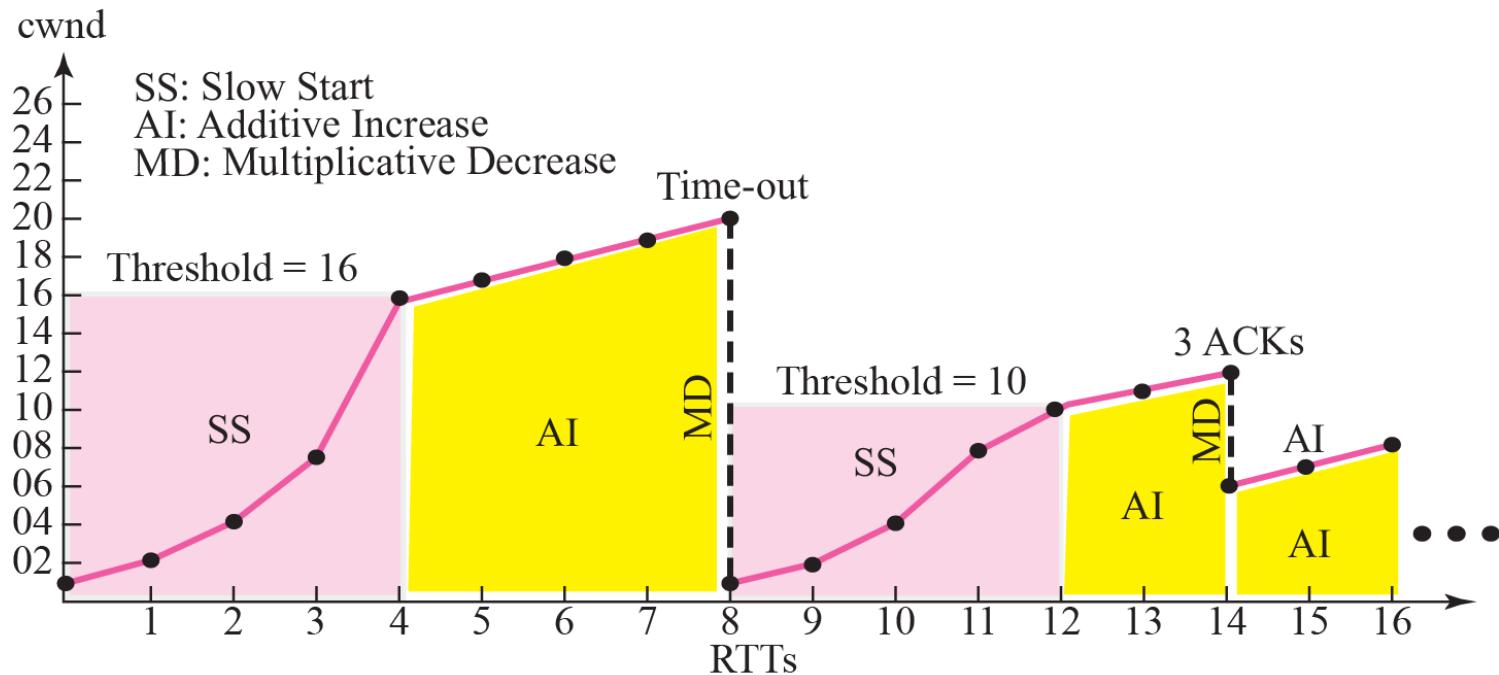
- In the congestion avoidance algorithm the size of the congestion window increases additively until congestion is detected



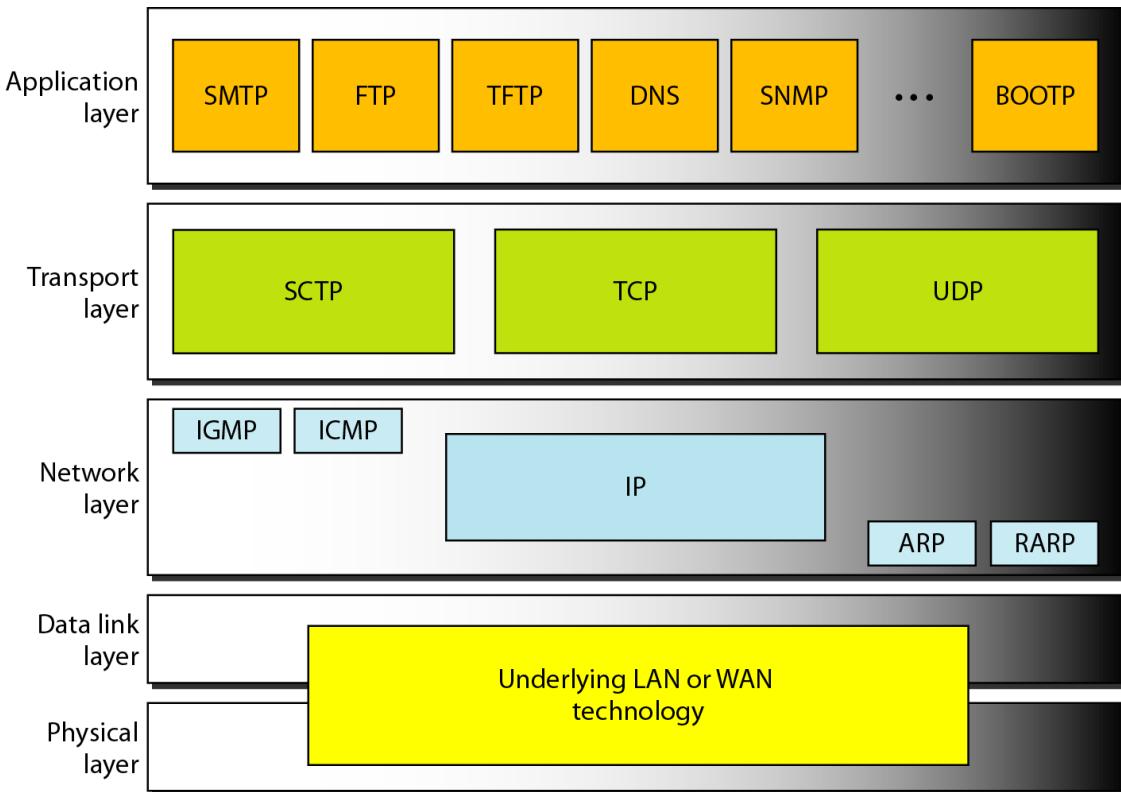
TCP Congestion policy summary



Congestion example



UDP



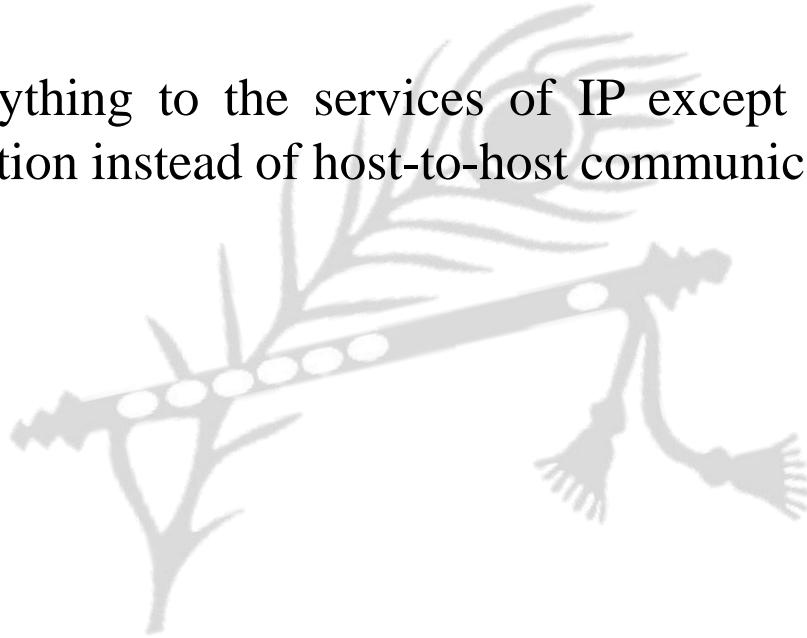
UDP

- UDP provides
 - best effort delivery
 - Connectionless
 - Unreliable
 - Out of order delivery
- TCP
 - Reliable
 - In-order delivery
 - Congestion control
 - Flow control
 - Connection setup



UDP

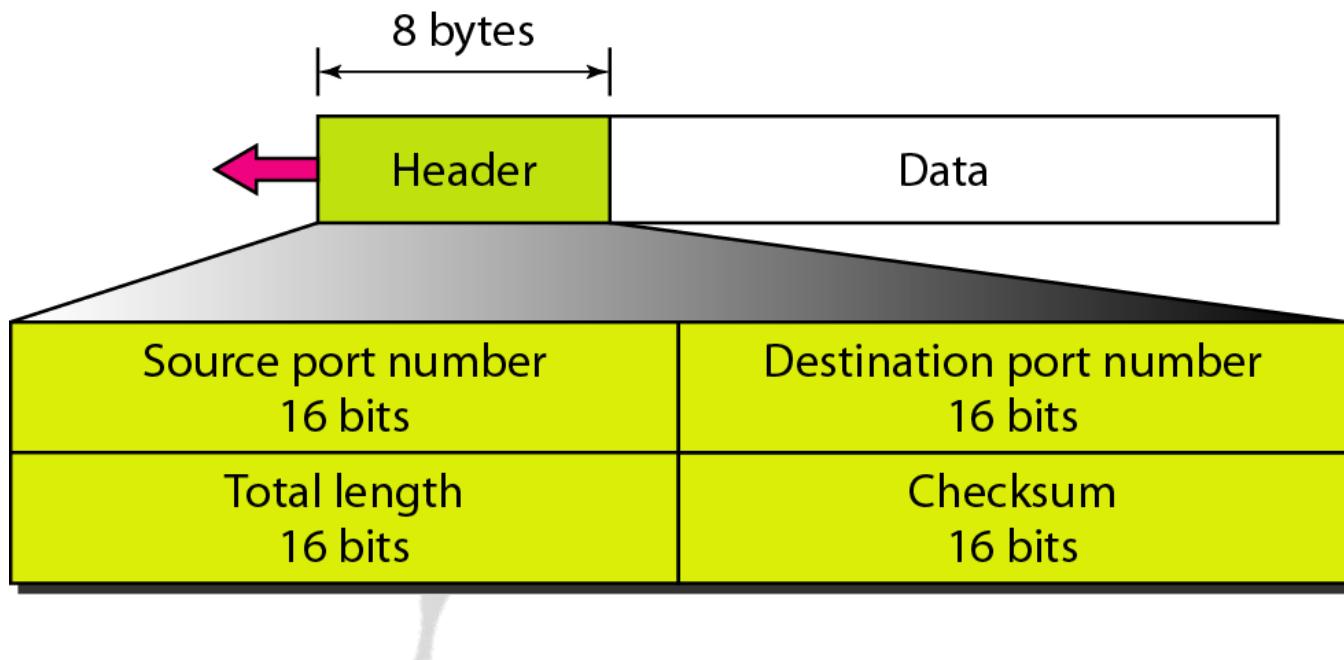
- The User Datagram Protocol (UDP) is called a connectionless, unreliable transport protocol
- It does not add anything to the services of IP except to provide process-to-process communication instead of host-to-host communication



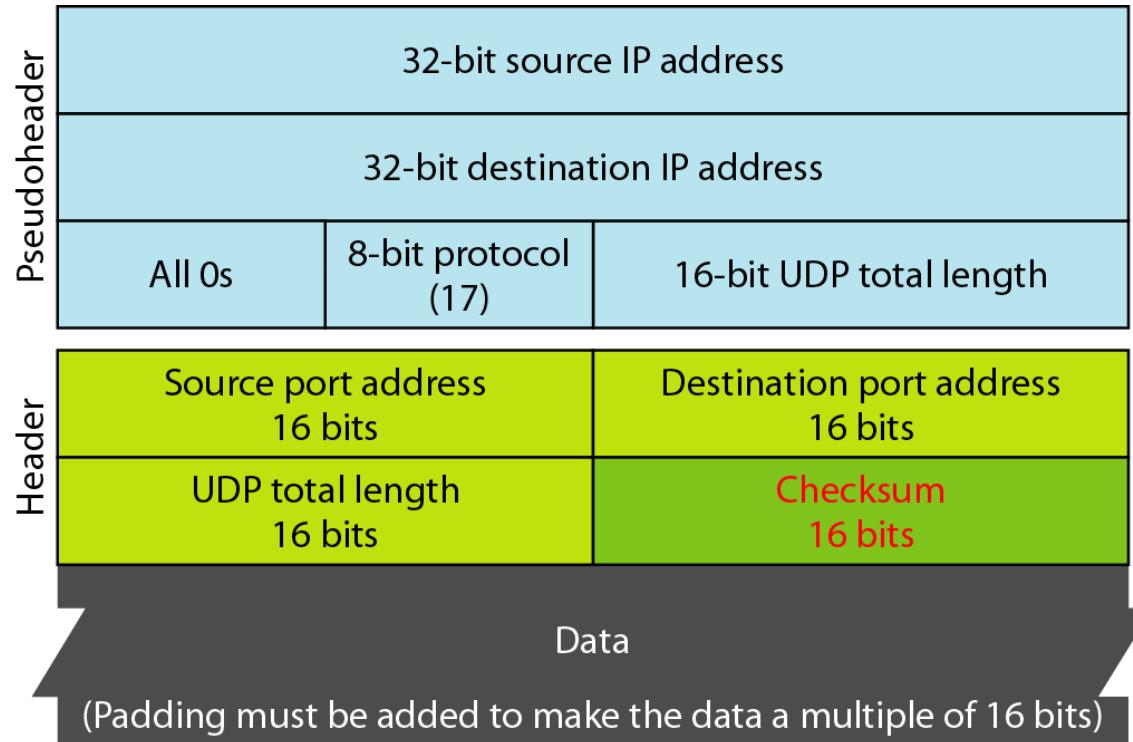
Well-known ports used with UDP

<i>Port</i>	<i>Protocol</i>	<i>Description</i>
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Nameserver	Domain Name Service
67	BOOTPs	Server port to download bootstrap information
68	BOOTPc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

UDP format



Pseudo-header for checksum calculation



APPLICATION LAYER

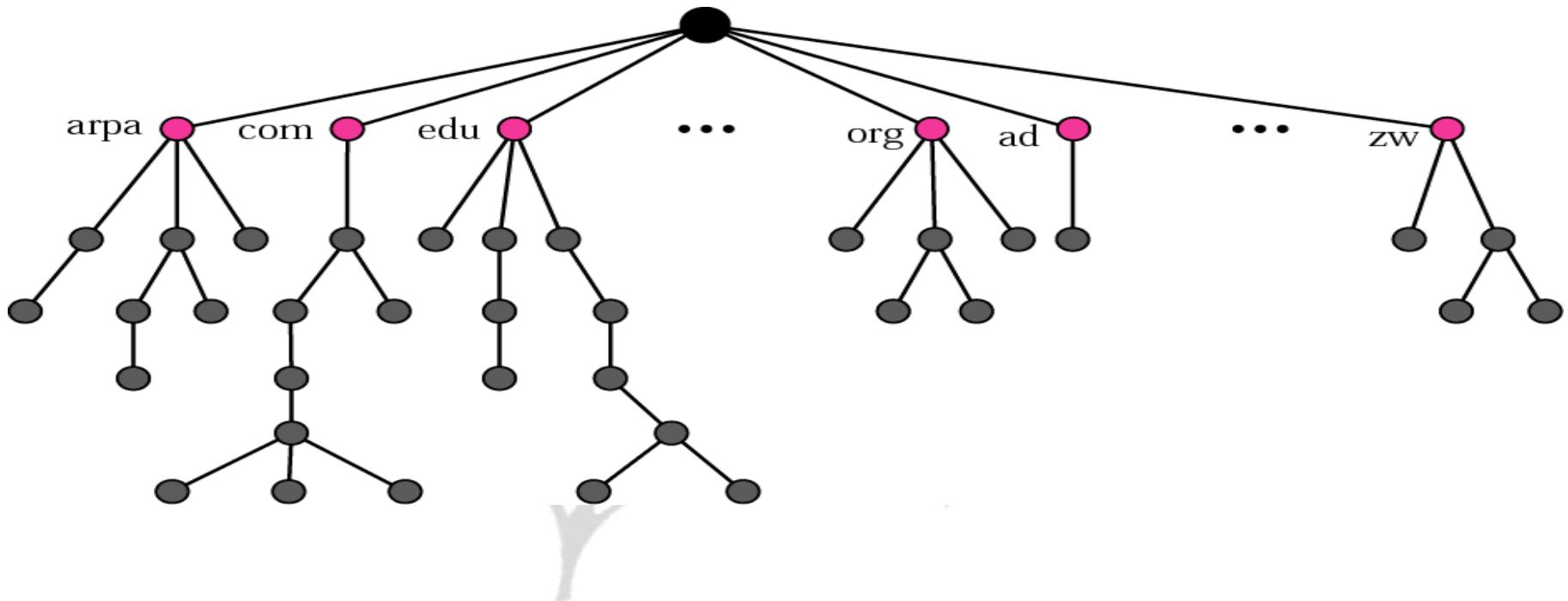
Name space

- The names assigned to machines must be unique because the addresses are unique
- A name space that maps each address to a unique name can be organized in two ways
 - flat
 - hierarchical

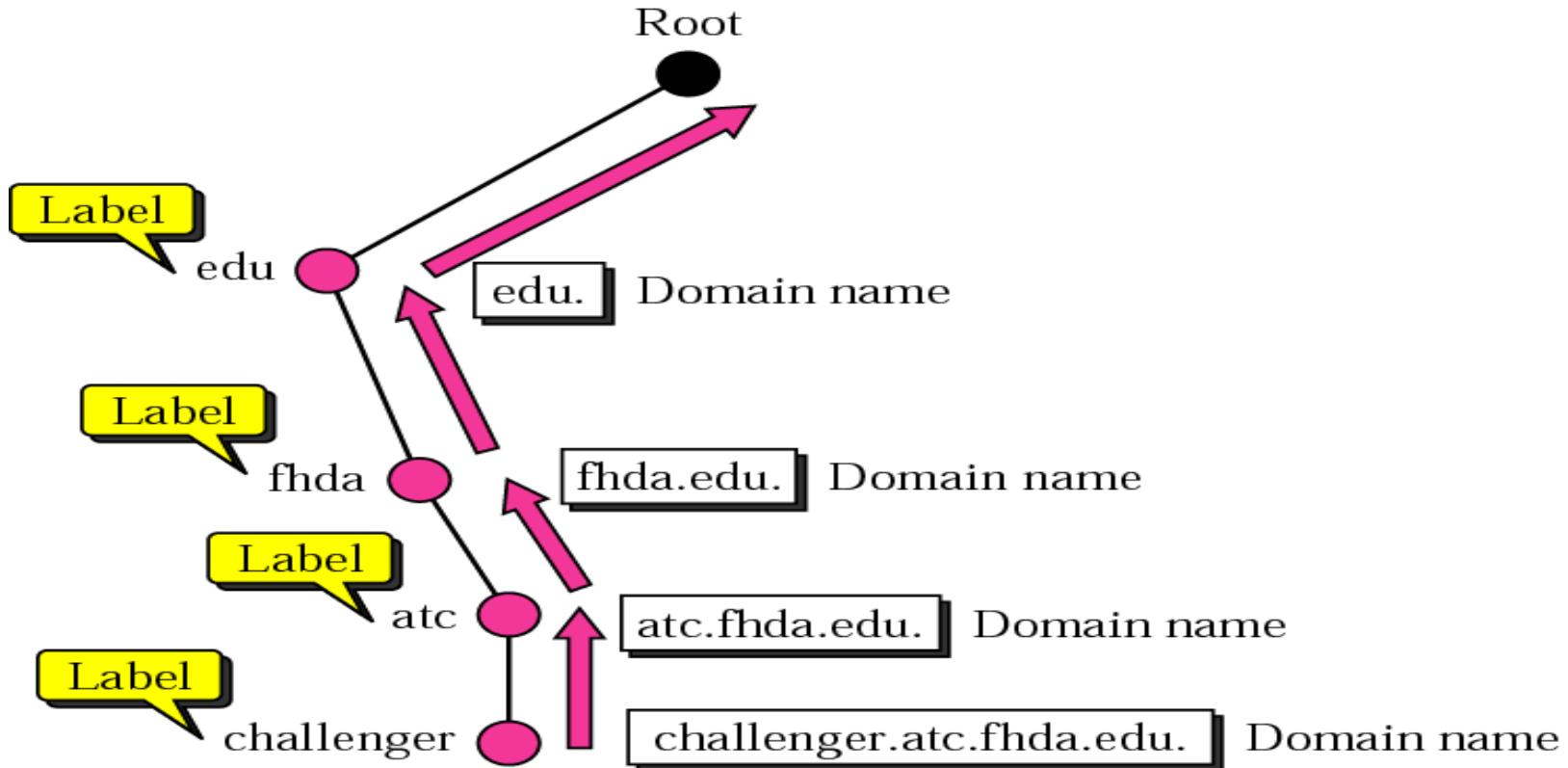
Domain Name Space (DNS)

- The domain name space is hierarchical in design
- The names are defined in an inverted-tree structure with the root at the top
- The tree can have 128 levels: level 0 (root) to level 127

Domain Name Space (DNS)



Domain Name and Labels

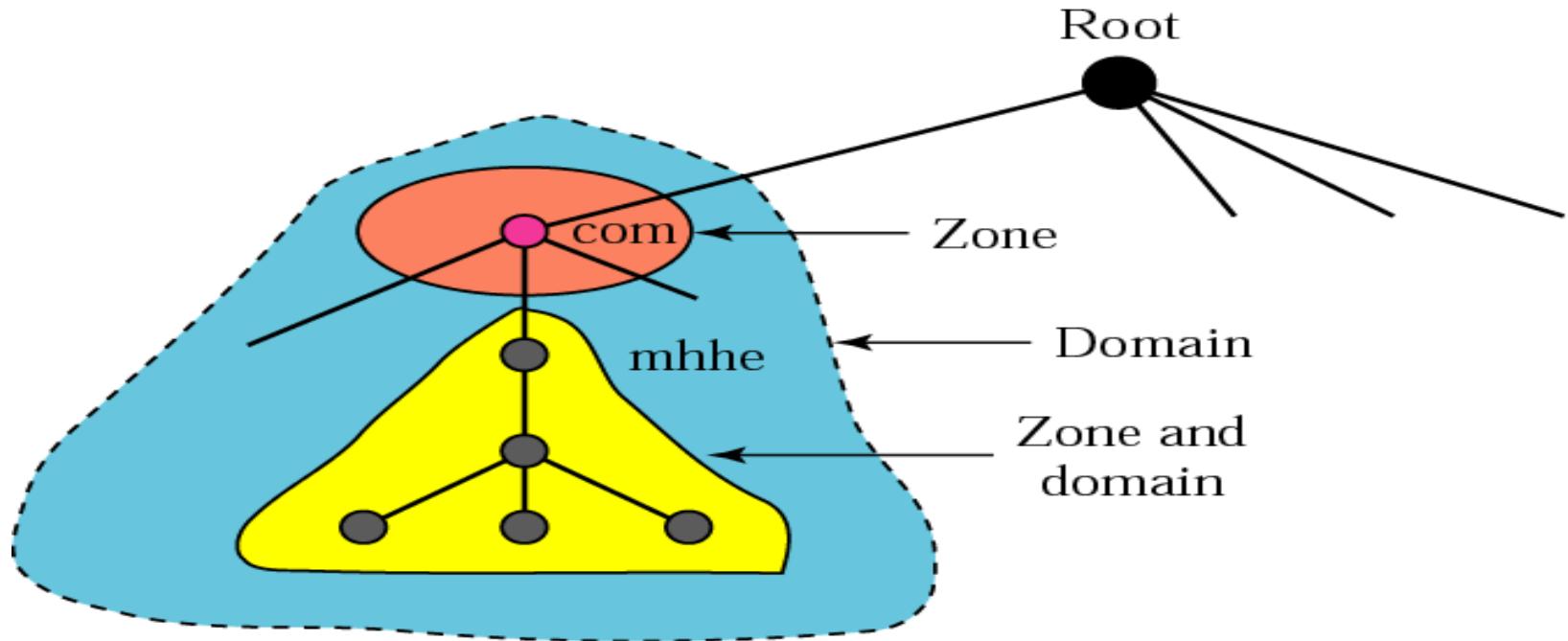


Distribution of name space

- The information contained in the domain name space is distributed among many computers called DNS servers

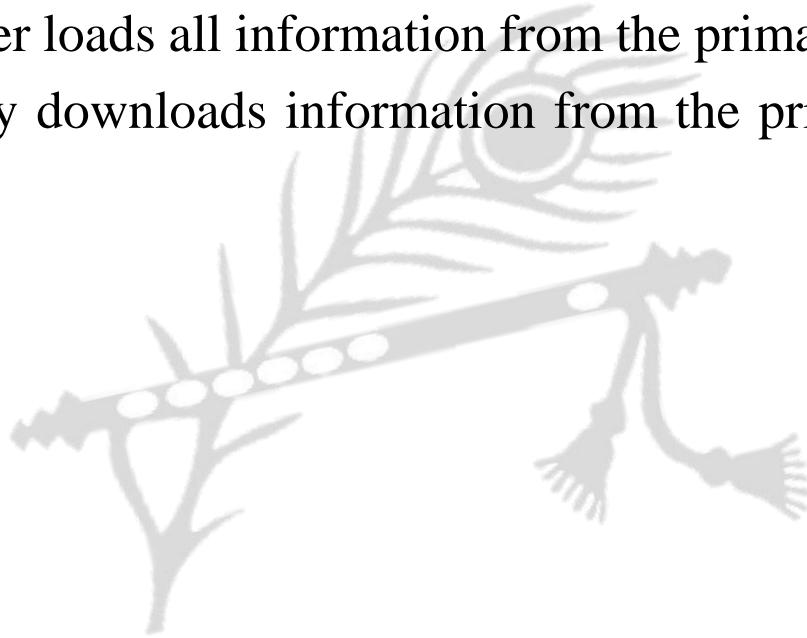


Zones and domains



Zones and domains

- A primary server loads all information from the disk file
- The secondary server loads all information from the primary server
- When the secondary downloads information from the primary, it is called zone transfer

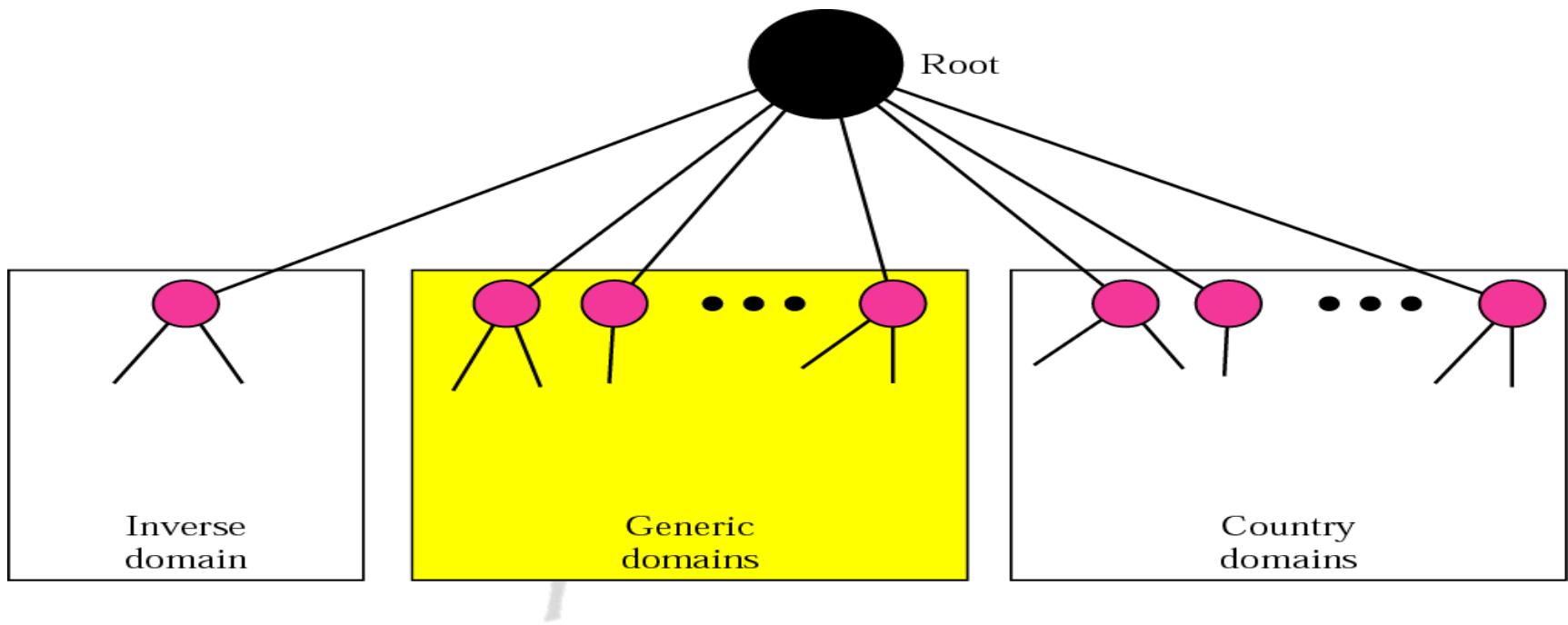


DNS in the internet

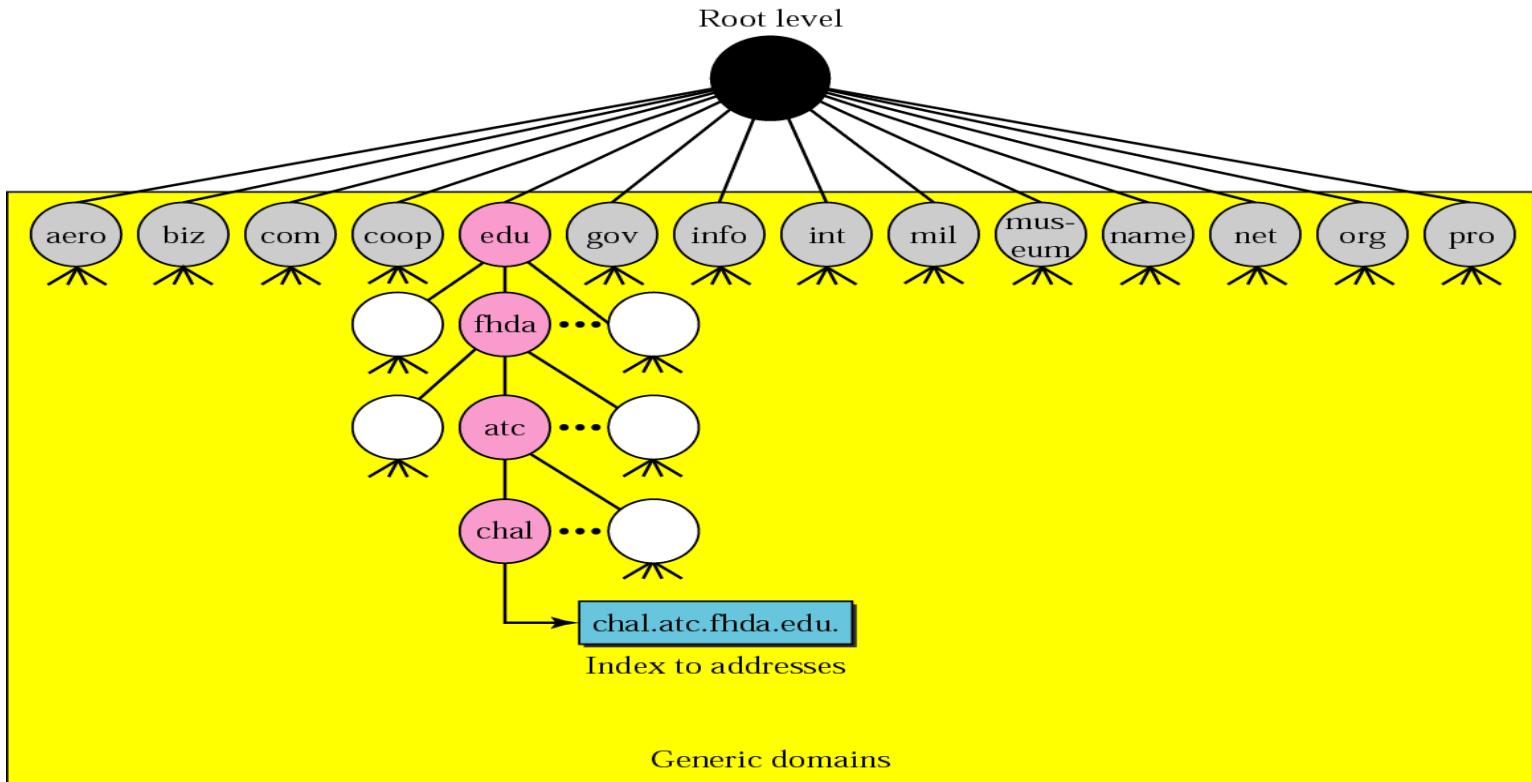
- The domain name space (tree) is divided into three different sections
 - generic domains
 - country domains
 - inverse domain



DNS used in the Internet



Generic domain

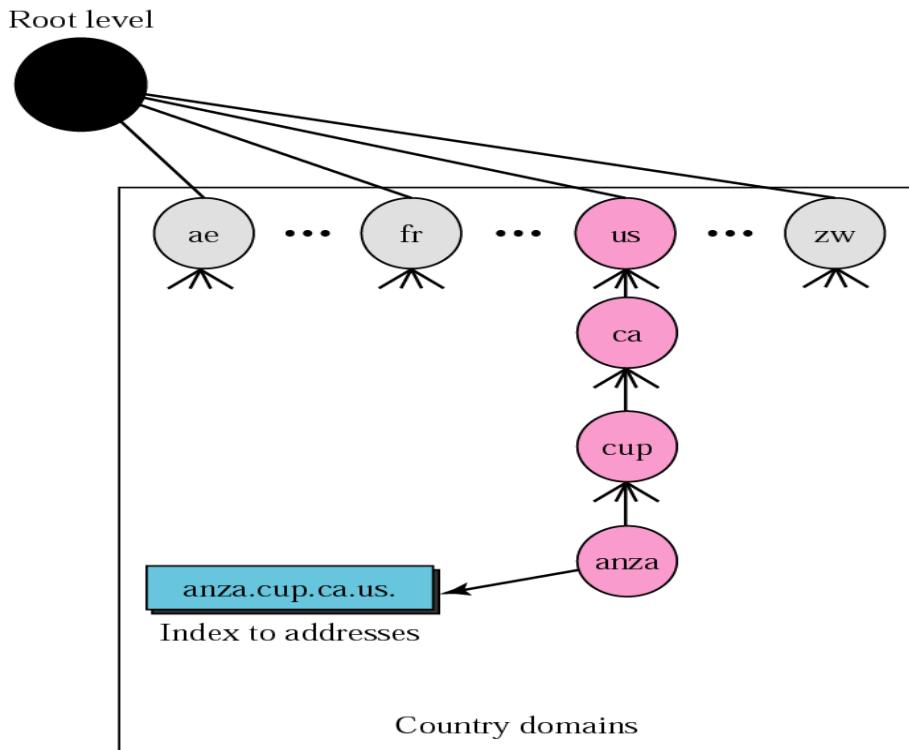


Generic domain labels

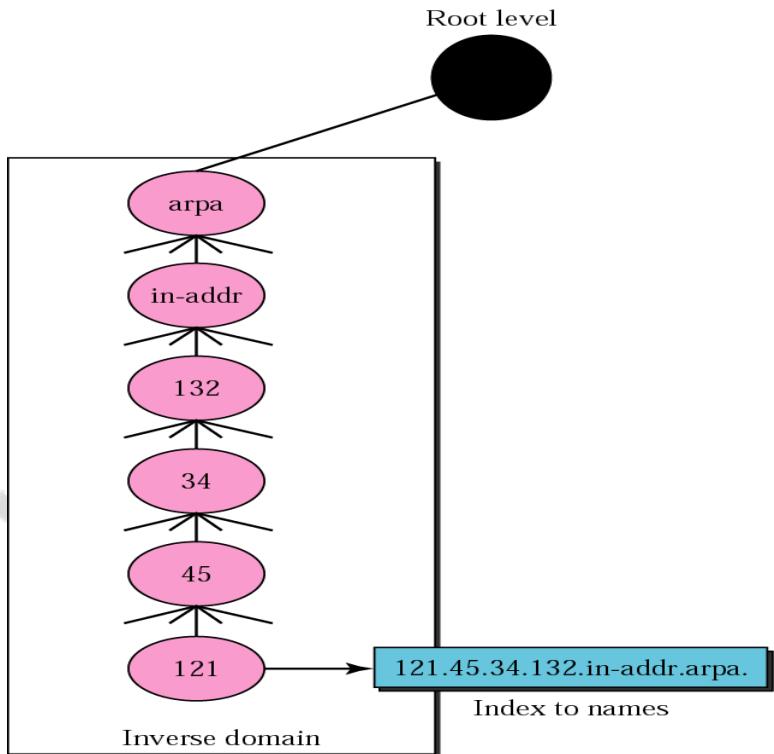
<i>Label</i>	<i>Description</i>
aero	Airlines and aerospace companies
biz	Businesses or firms (similar to “com”)
com	Commercial organizations
coop	Cooperative business organizations
edu	Educational institutions
gov	Government institutions
info	Information service providers

<i>Label</i>	<i>Description</i>
int	International organizations
mil	Military groups
museum	Museums and other non-profit organizations
name	Personal names (individuals)
net	Network support centers
org	Nonprofit organizations
pro	Professional individual organizations

Country domain



Inverse domain

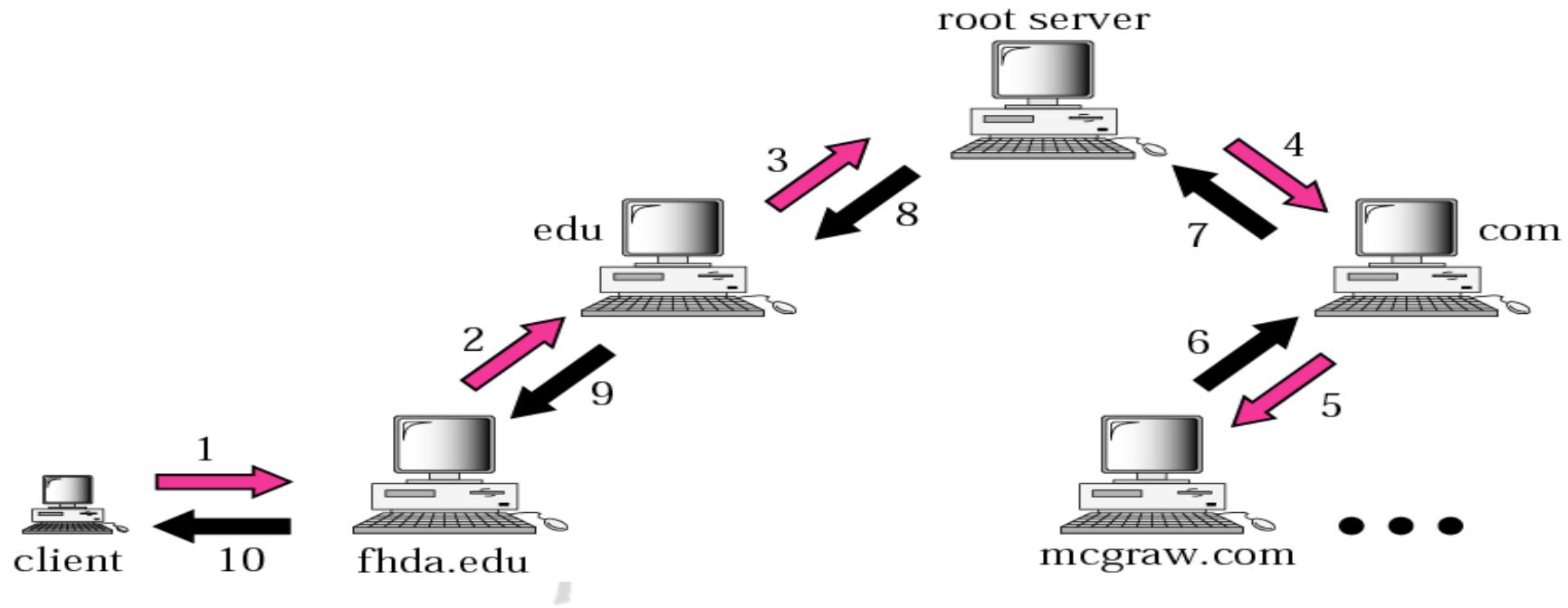


Resolution

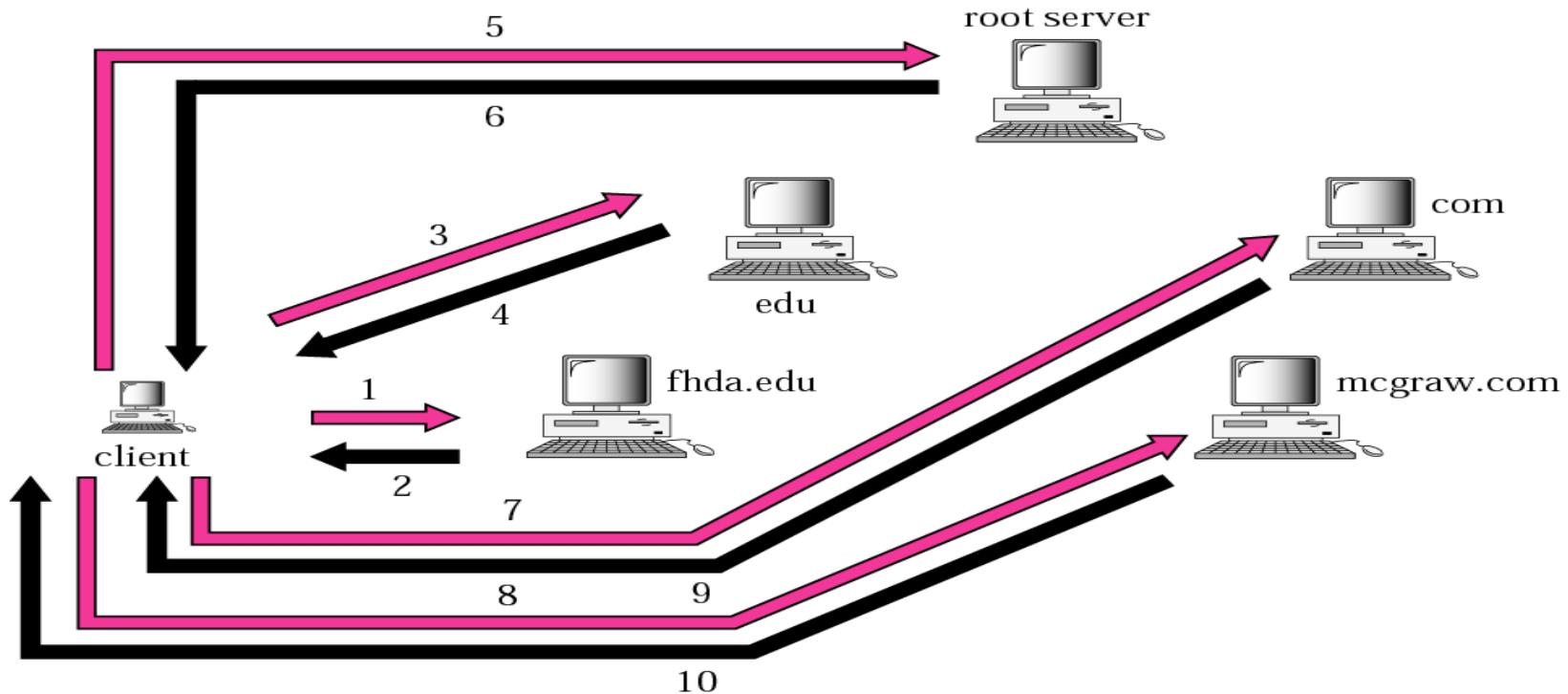
- Mapping a name to an address or an address to a name is called name-address resolution



Recursive resolution

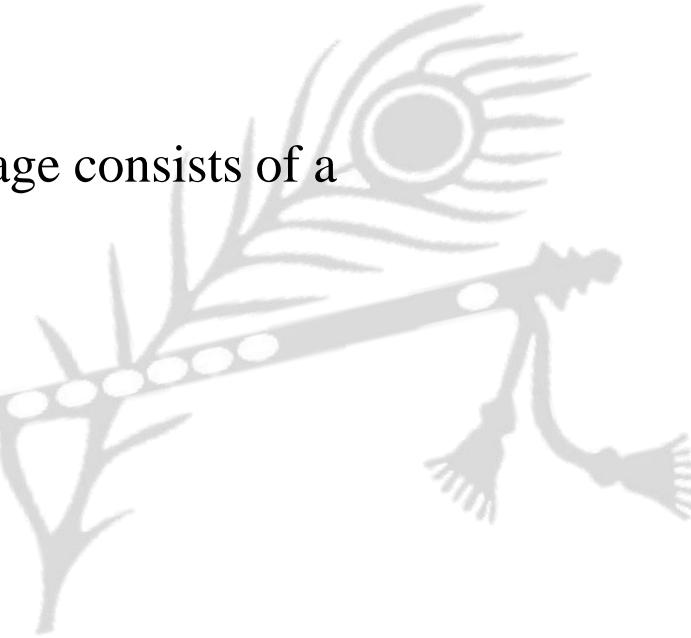


Iterative resolution

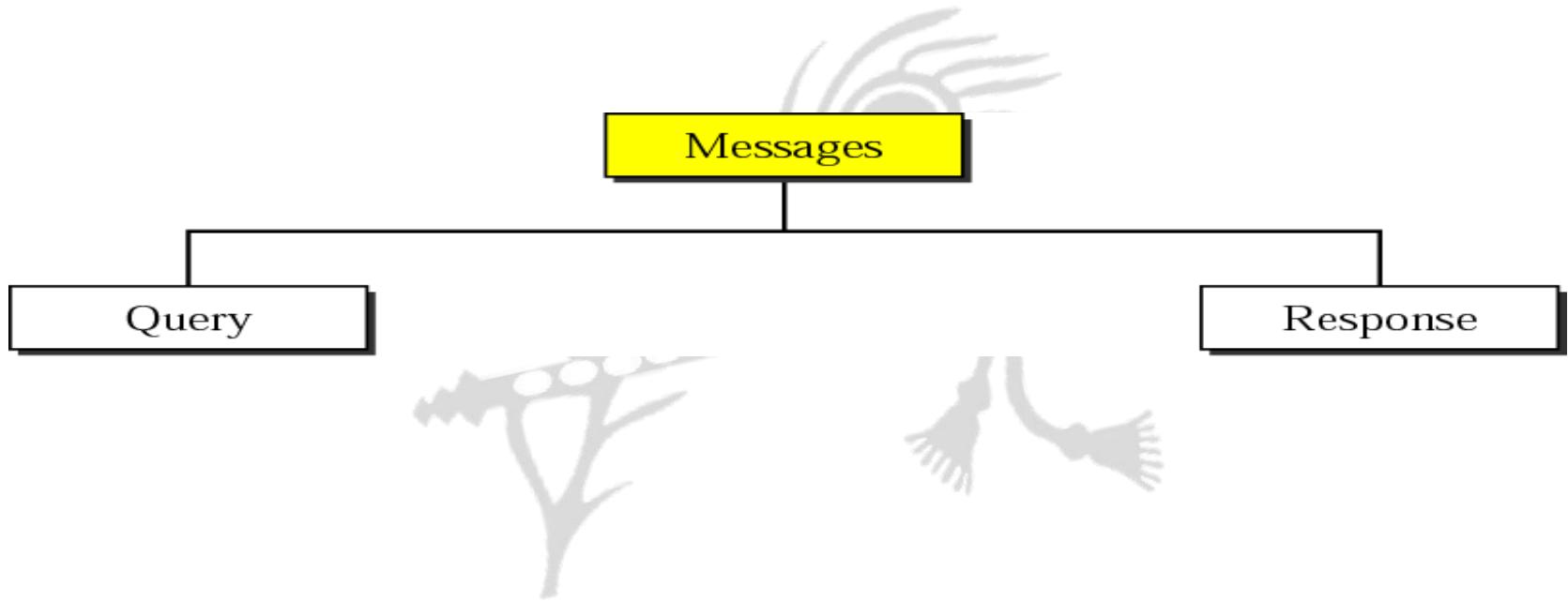


DNS messages

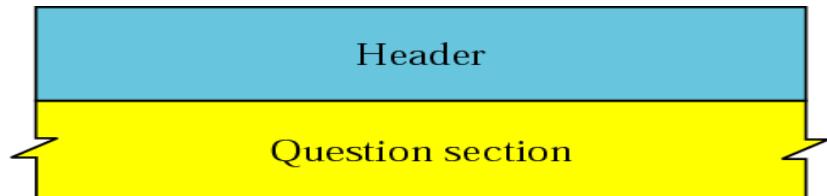
- The DNS query message consists of a
 - Header
 - Question records
- The DNS response message consists of a
 - Header
 - question records
 - answer records
 - authoritative records
 - additional records



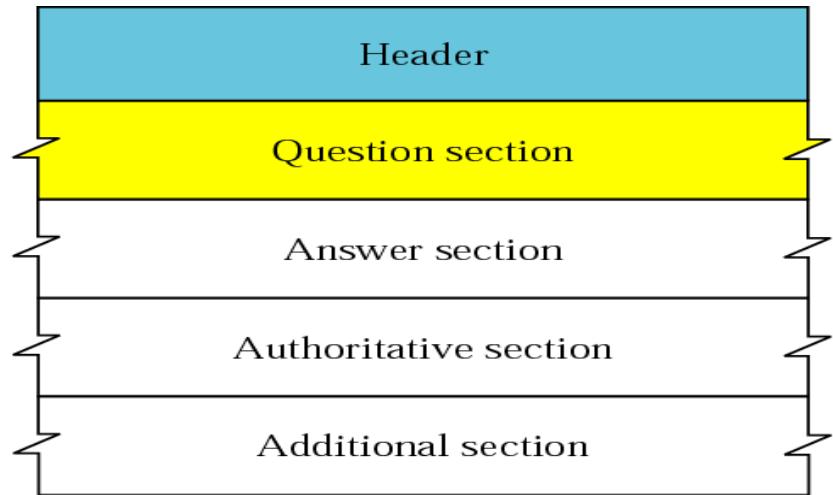
DNS messages



Query and response messages



a. Query



b. Response

Header format

Identification	Flags
Number of question records	Number of answer records (All 0s in query message)
Number of authoritative records (All 0s in query message)	Number of additional records (All 0s in query message)

Flag fields



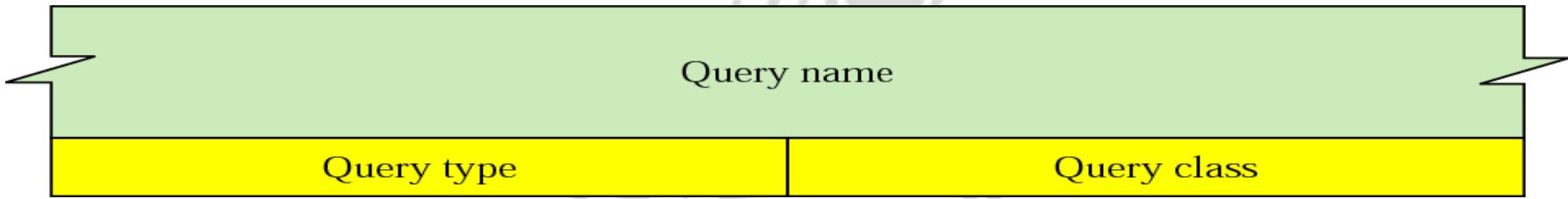
Values of rCode

<i>Value</i>	<i>Meaning</i>
0	No error
1	Format error
2	Problem at name server
3	Domain reference problem
4	Query type not supported
5	Administratively prohibited
6–15	Reserved

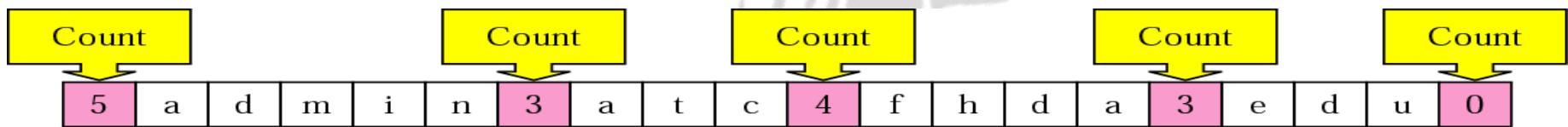
Types of records

- Two types of records are used in DNS
- The question records are used in the question section of the query and response messages
- The resource records are used in the answer, authoritative, and additional information sections of the response message

Question record format



Query name format



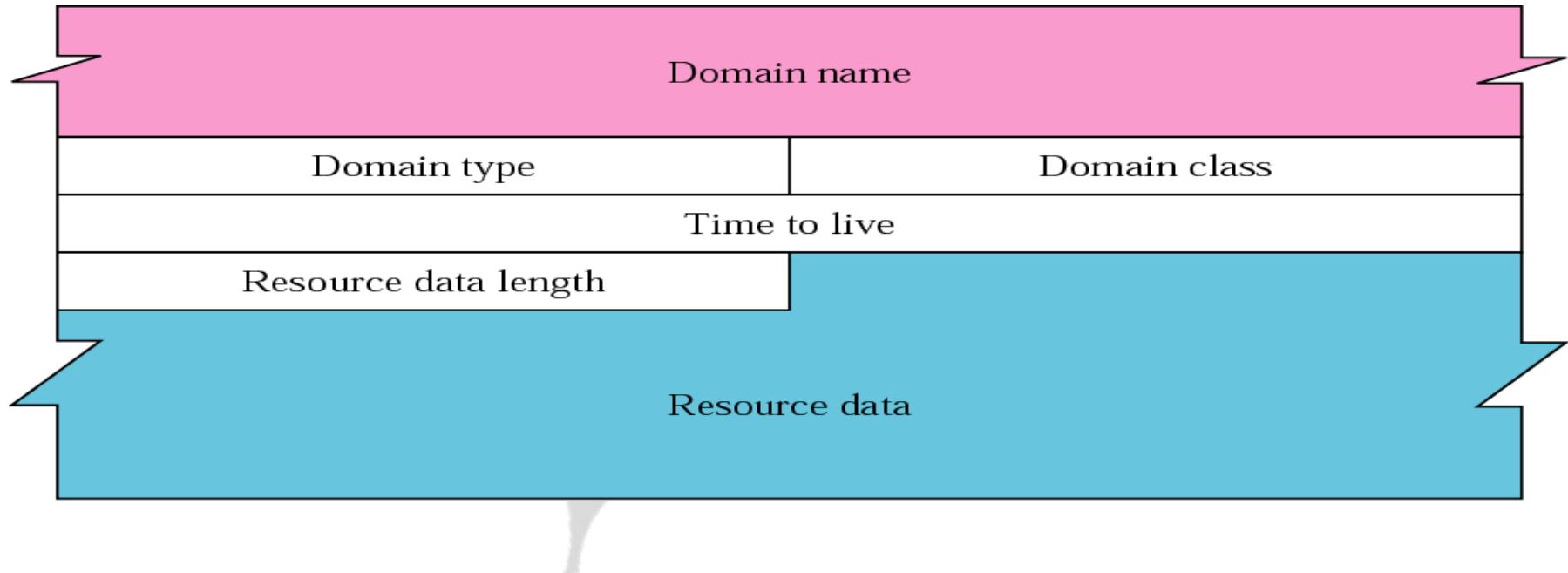
Types

Type	Mnemonic	Description
1	A	Address. A 32-bit IPv4 address. It is used to convert a domain name to an IPv4 address.
2	NS	Name server. It identifies the authoritative servers for a zone.
5	CNAME	Canonical name. It defines an alias for the official name of a host.
6	SOA	Start of authority. It marks the beginning of a zone. It is usually the first record in a zone file.
11	WKS	Well-known services. It defines the network services that a host provides.
12	PTR	Pointer. It is used to convert an IP address to a domain name.
13	HINFO	Host information. It gives the description of the hardware and the operating system used by a host.
15	MX	Mail exchange. It redirects mail to a mail server.
28	AAAA	Address. An IPv6 address (see Chapter 27).
252	AXFR	A request for the transfer of the entire zone.
255	ANY	A request for all records.

Classes

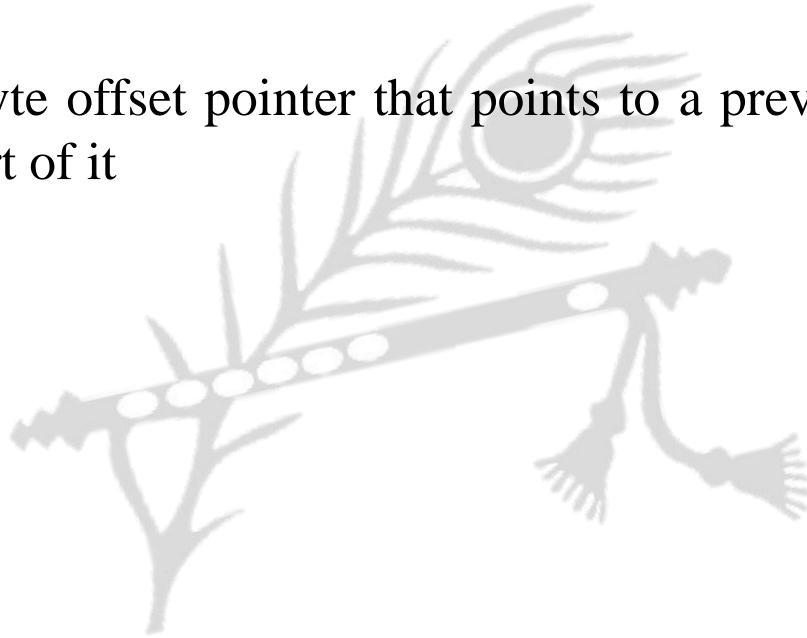
<i>Class</i>	<i>Mnemonic</i>	<i>Description</i>
1	IN	Internet
2	CSNET	CSNET network (obsolete)
3	CS	The COAS network
4	HS	The Hesiod server developed by MIT

Resource record format

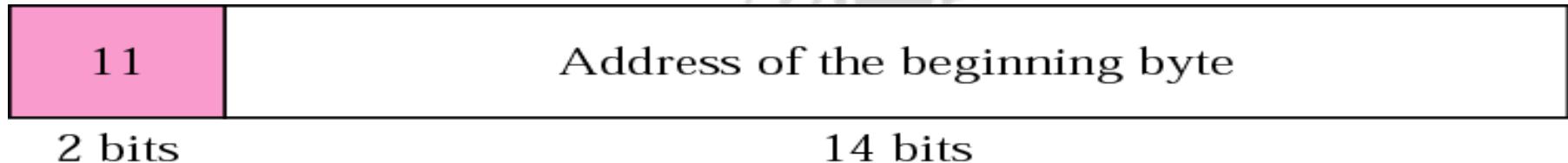


Compression

- DNS requires that a domain name be replaced by an offset pointer if it is repeated
- DNS defines a 2-byte offset pointer that points to a previous occurrence of the domain name or part of it



Format of an offset pointer



DDNS

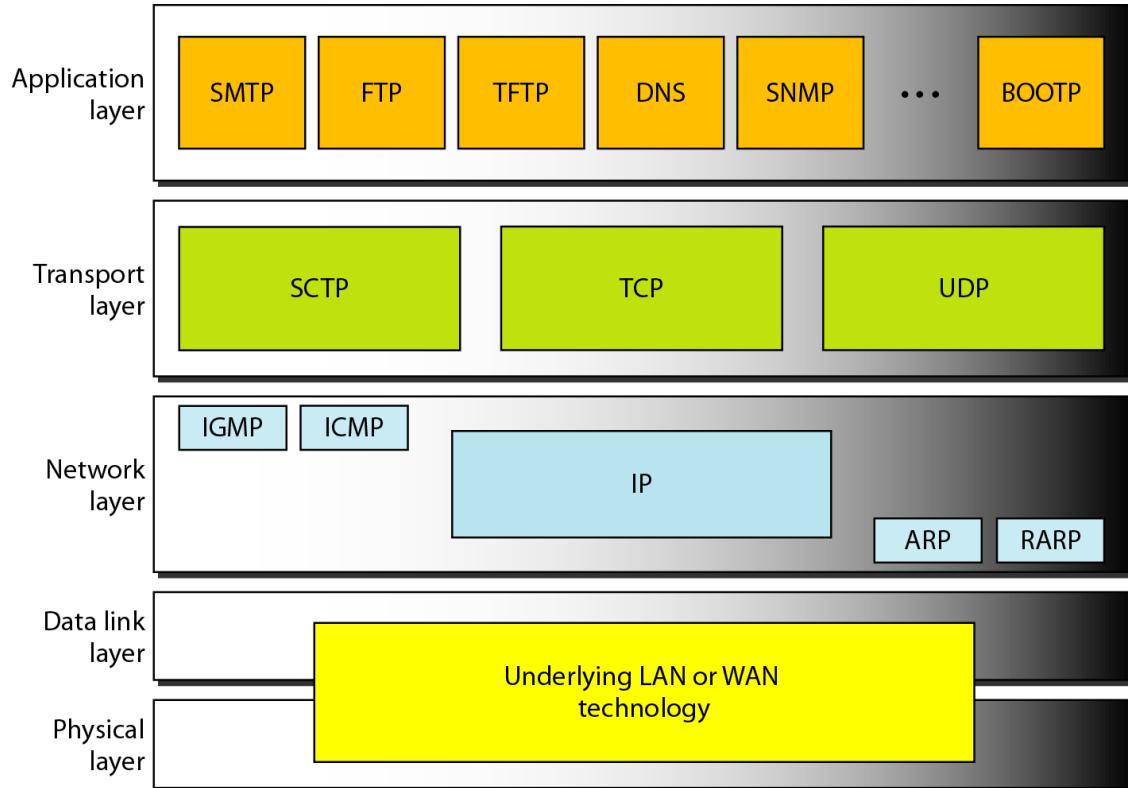
- The Dynamic Domain Name System (DDNS) updates the DNS master file dynamically



Encapsulation

- DNS uses UDP as the transport protocol when the size of the response message is less than 512 bytes
- If the size of the response message is more than 512 bytes, a TCP connection is used
- DNS can use the services of UDP or TCP using the well-known port 53

FTP



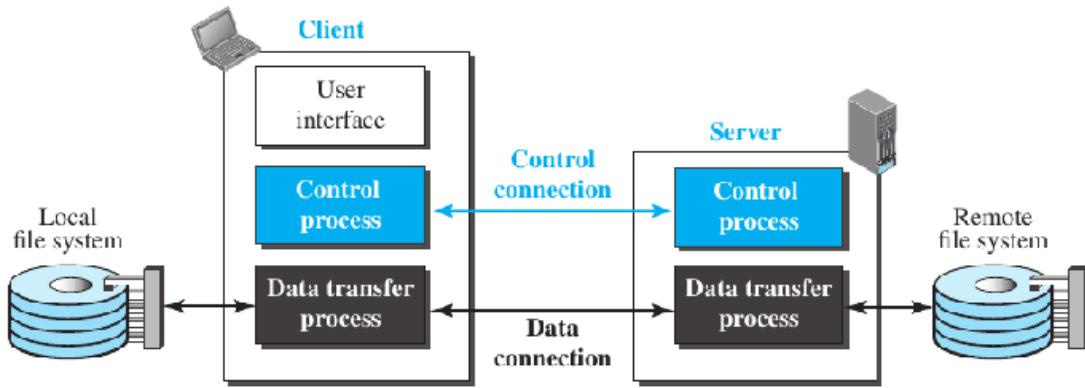
FTP

- File Transfer Protocol (FTP) is the standard mechanism provided by TCP/IP for copying a file from one host to another

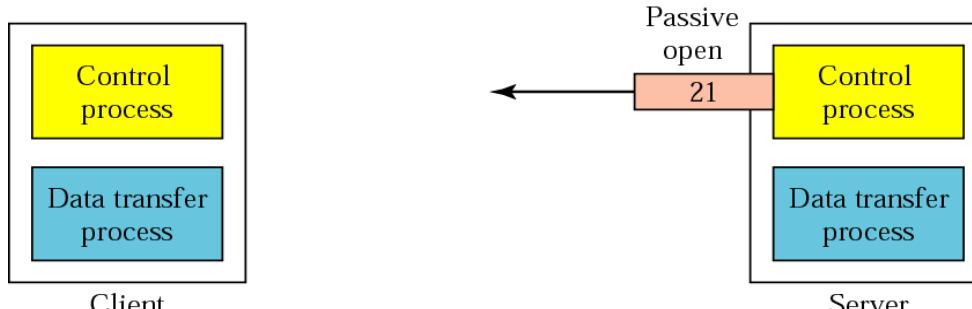


FTP

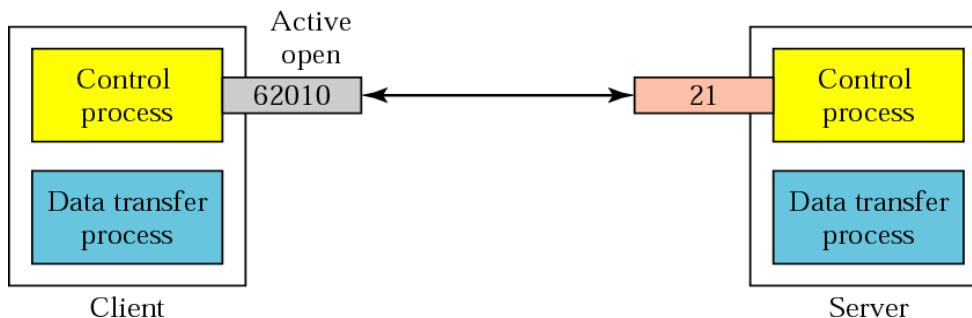
- FTP uses the services of TCP
- It needs two TCP connections
- The well-known port 21 is used for the control connection and the well-known port 20 for the data connection



Opening the control connection

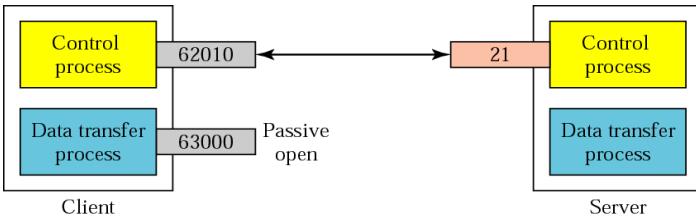


a. Passive open by server

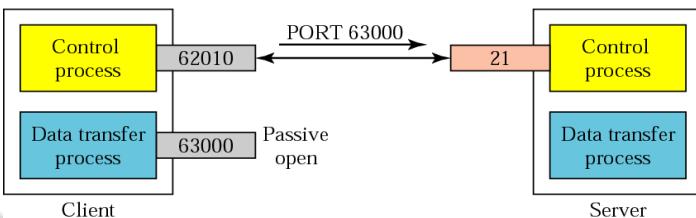


b. Active open by client

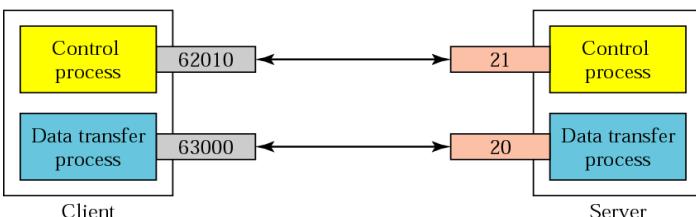
Creating the data connection



a. Passive open by client

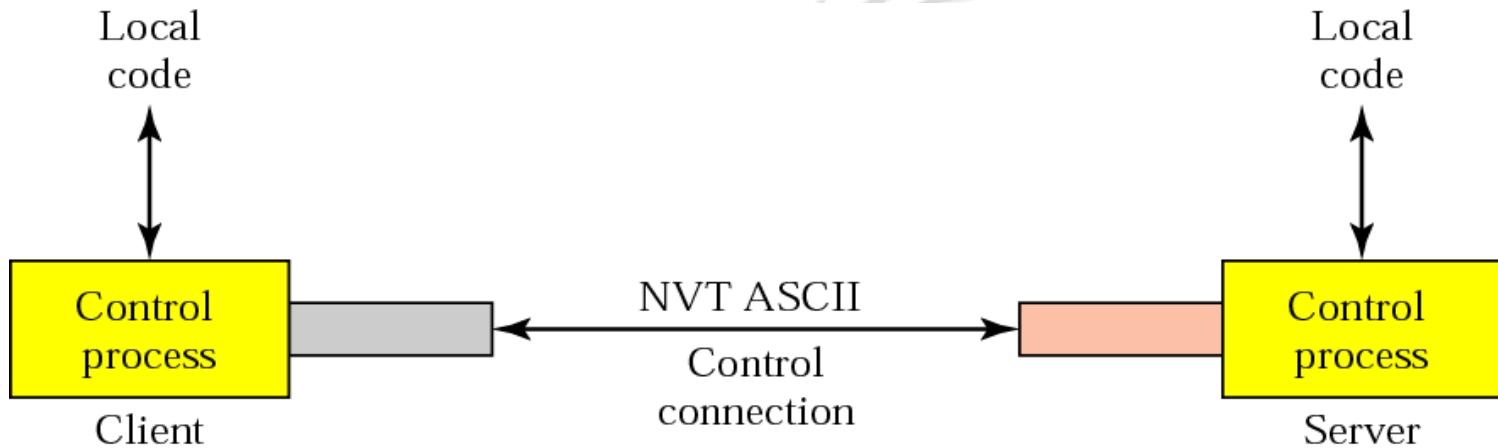


b. Sending ephemeral port number to server

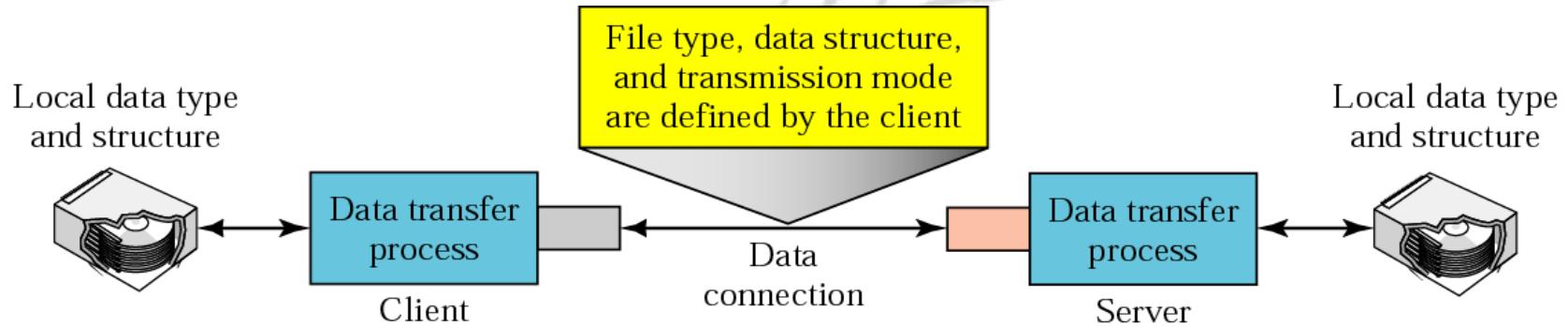


c. Active open by server

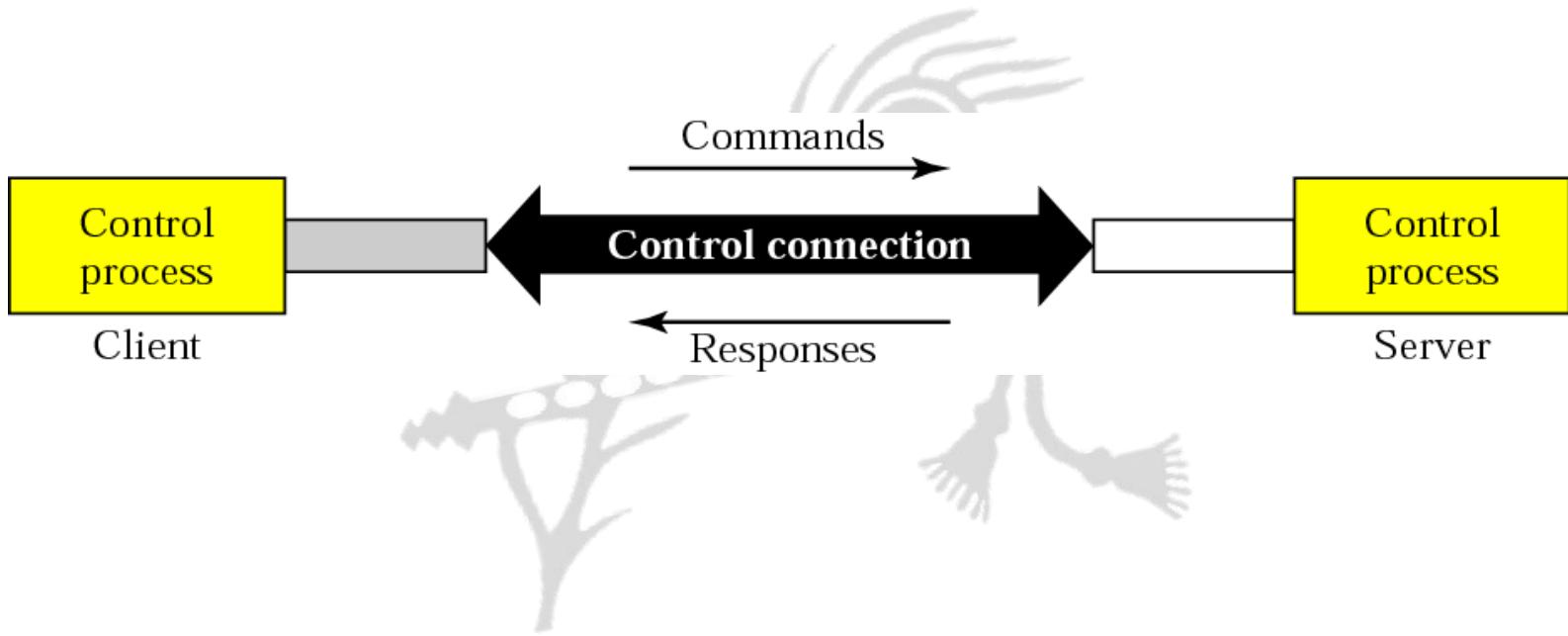
Using the control connection



Using the data connection



Command processing



Access commands

<i>Command</i>	<i>Argument(s)</i>	<i>Description</i>
USER	User id	User information
PASS	User password	Password
ACCT	Account to be charged	Account information
REIN		Reinitialize
QUIT		Log out of the system
ABOR		Abort the previous command



File management commands

<i>Command</i>	<i>Argument(s)</i>	<i>Description</i>
CWD	Directory name	Change to another directory
CDUP		Change to the parent directory
DELE	File name	Delete a file
LIST	Directory name	List subdirectories or files
NLIST	Directory name	List the names of subdirectories or files without other attributes
MKD	Directory name	Create a new directory
PWD		Display name of current directory
RMD	Directory name	Delete a directory
RNFR	File name (old file name)	Identify a file to be renamed
RNTO	File name (new file name)	Rename the file
SMNT	File system name	Mount a file system

Data formatting commands

<i>Command</i>	<i>Argument(s)</i>	<i>Description</i>
TYPE	A (ASCII), E (EBCDIC), I (Image), N (Nonprint), or T (TELNET)	Define the file type and if necessary the print format
STRU	F (File), R (Record), or P (Page)	Define the organization of the data
MODE	S (Stream), B (Block), or C (Compressed)	Define the transmission mode

Port defining commands

<i>Command</i>	<i>Argument(s)</i>	<i>Description</i>
PORT	6-digit identifier	Client chooses a port
PASV		Server chooses a port

File transfer commands

<i>Command</i>	<i>Argument(s)</i>	<i>Description</i>
RETR	File name(s)	Retrieve files; file(s) are transferred from server to the client
STOR	File name(s)	Store files; file(s) are transferred from the client to the server
APPE	File name(s)	Similar to STOR except if the file exists, data must be appended to it
STOU	File name(s)	Same as STOR except that the file name will be unique in the directory; however, the existing file should not be overwritten



<i>Command</i>	<i>Argument(s)</i>	<i>Description</i>
ALLO	File name(s)	Allocate storage space for the files at the server
REST	File name(s)	Position the file marker at a specified data point
STAT	File name(s)	Return the status of files

Miscellaneous commands

<i>Command</i>	<i>Argument(s)</i>	<i>Description</i>
HELP		Ask information about the server
NOOP		Check if server is alive
SITE	Commands	Specify the site-specific commands
SYST		Ask about operating system used by the server



Response commands

<i>Code</i>	<i>Description</i>
Positive Preliminary Reply	
120	Service will be ready shortly
125	Data connection open; data transfer will start shortly
150	File status is OK; data connection will be open shortly

<i>Code</i>	<i>Description</i>
Positive Intermediate Reply	
331	User name OK; password is needed
332	Need account for logging
350	The file action is pending; more information needed

<i>Code</i>	<i>Description</i>
Positive Completion Reply	
200	Command OK
211	System status or help reply
212	Directory status
213	File status
214	Help message
215	Naming the system type (operating system)
220	Service ready
221	Service closing
225	Data connection open
226	Closing data connection
227	Entering passive mode; server sends its IP address and port number
230	User login OK
250	Request file action OK

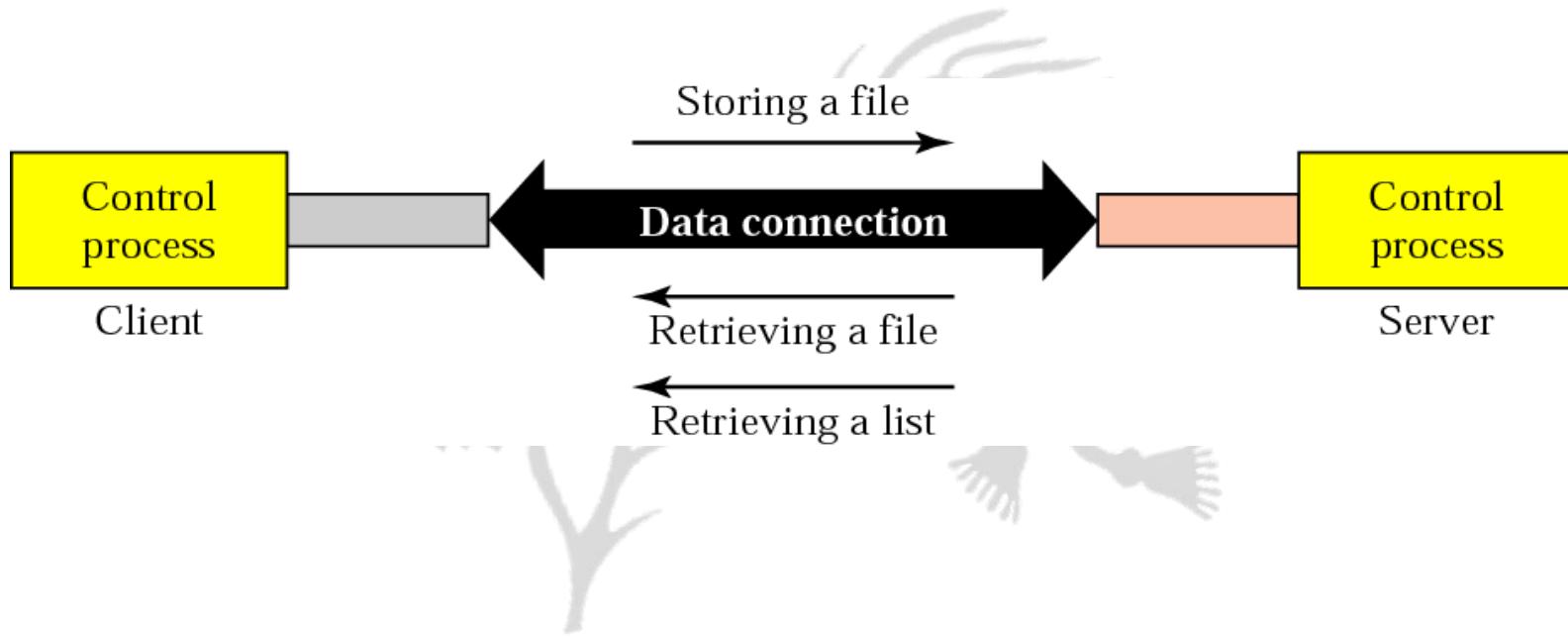
Response commands

<i>Code</i>	<i>Description</i>
Transient Negative Completion Reply	
425	Cannot open data connection
426	Connection closed; transfer aborted
450	File action not taken; file not available
451	Action aborted; local error
452	Action aborted; insufficient storage
Permanent Negative Completion Reply	
500	Syntax error; unrecognized command

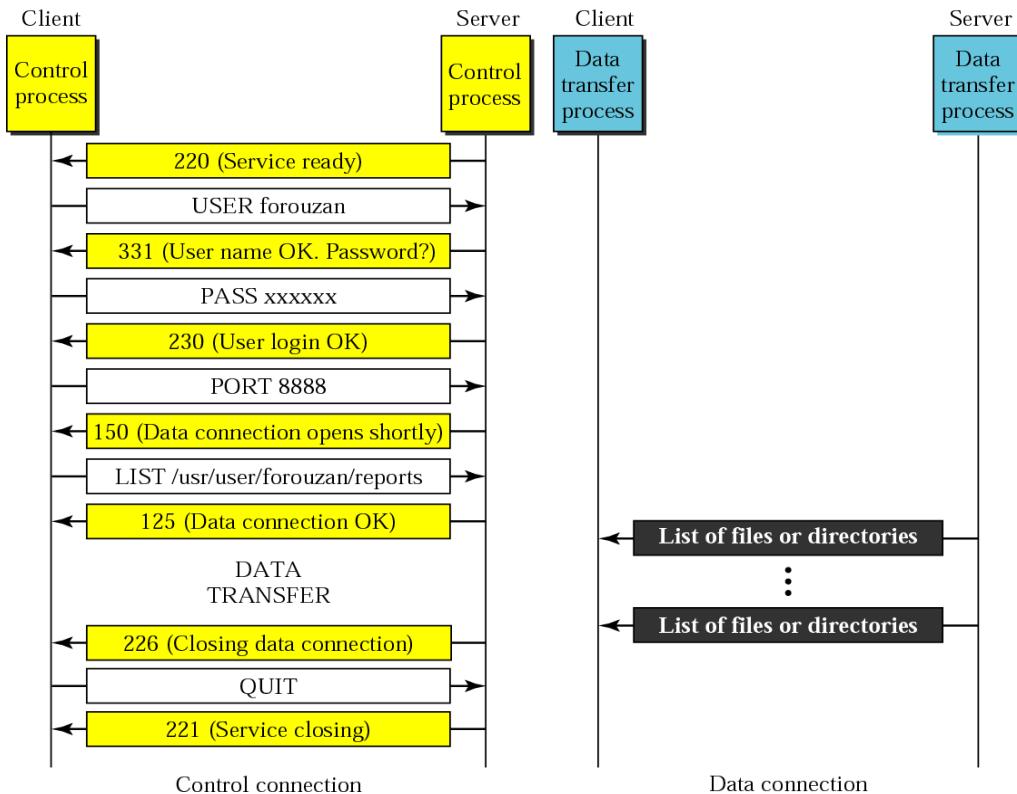


<i>Code</i>	<i>Description</i>
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command parameter not implemented
530	User not logged in
532	Need account for storing file
550	Action is not done; file unavailable
552	Requested action aborted; exceeded storage allocation
553	Requested action not taken; file name not allowed

File transfer

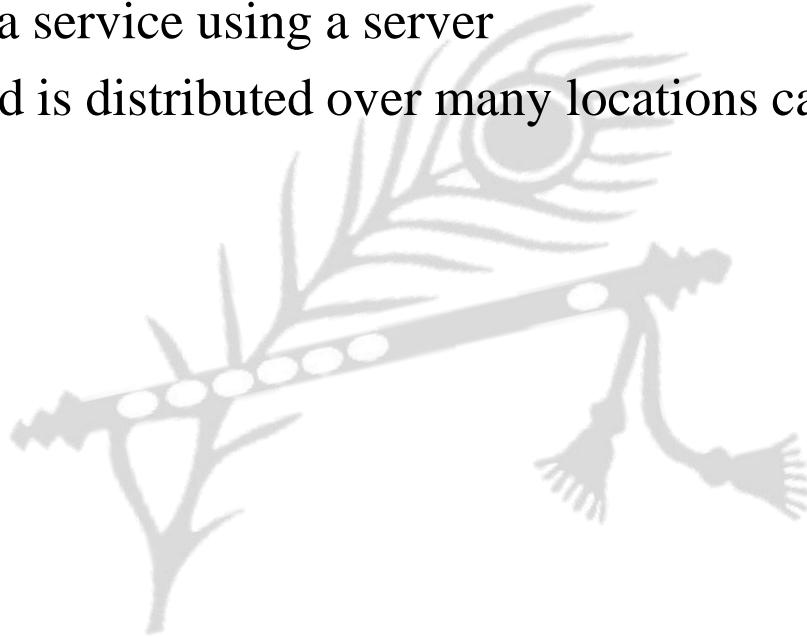


Example

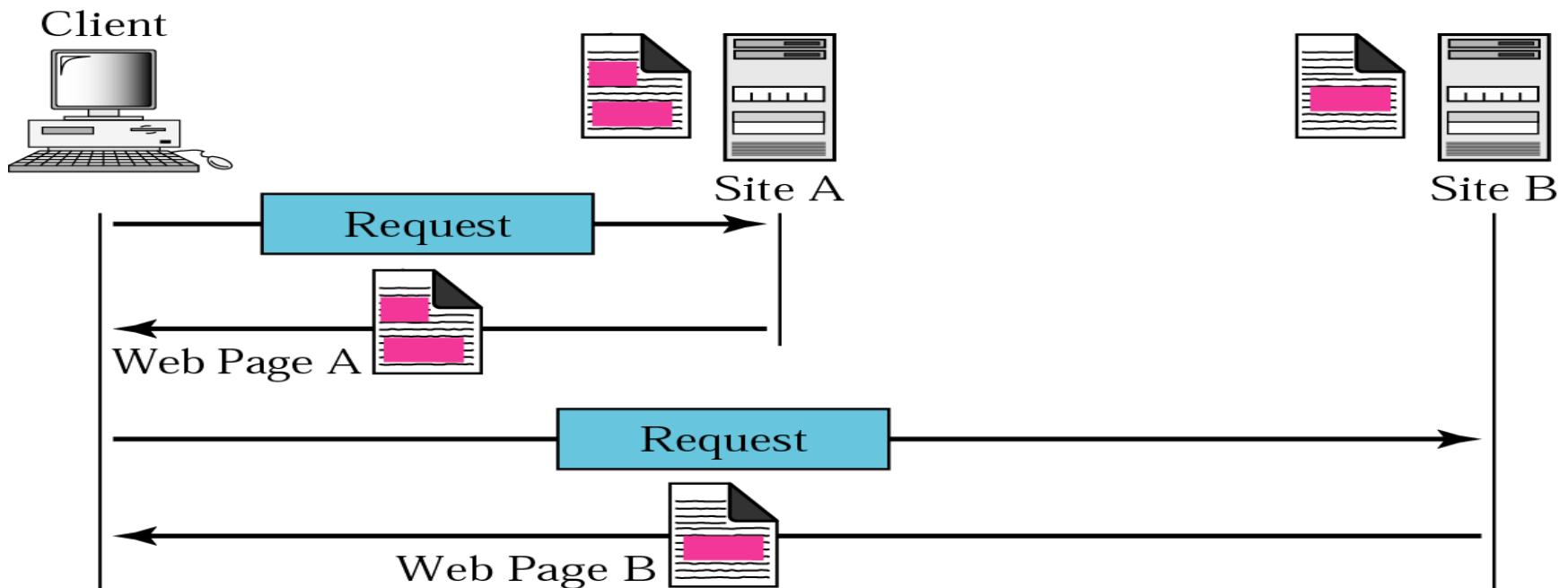


WWW

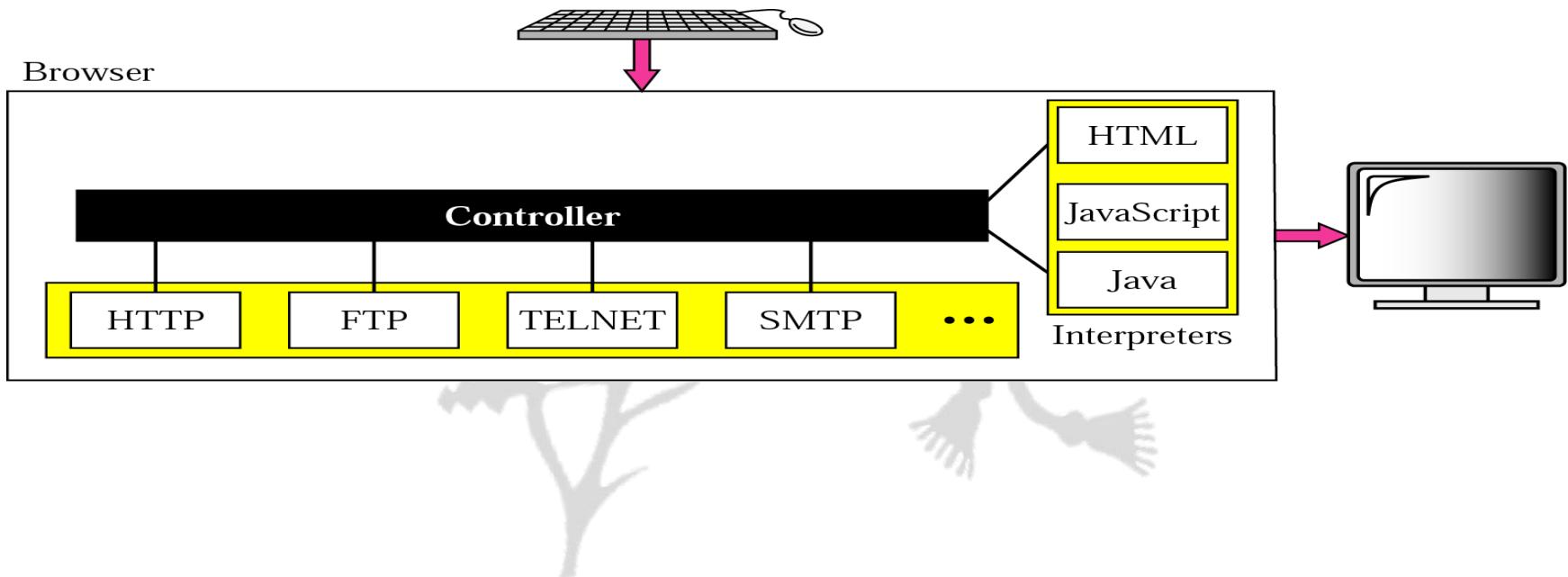
- The WWW is a distributed client-server service, in which a client using a browser can access a service using a server
- The service provided is distributed over many locations called sites



Architecture of WWW



Browser

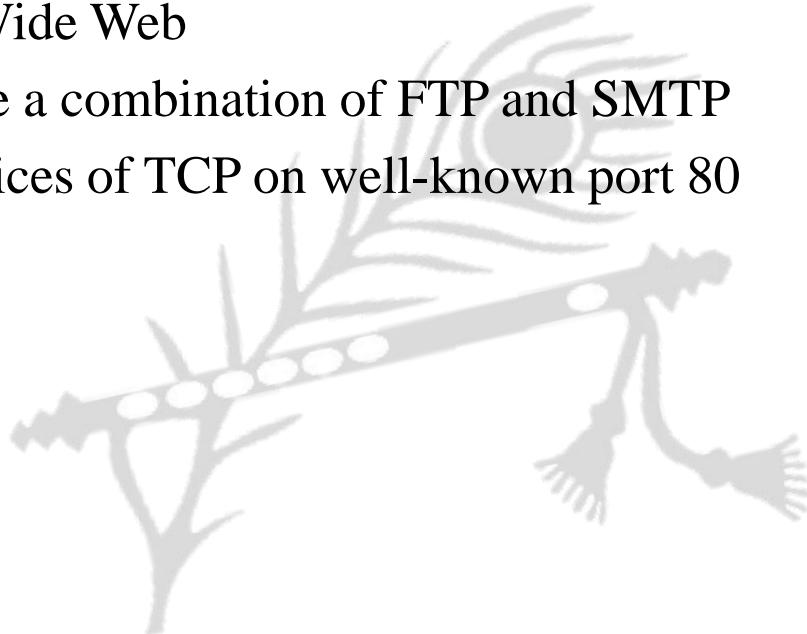


URL

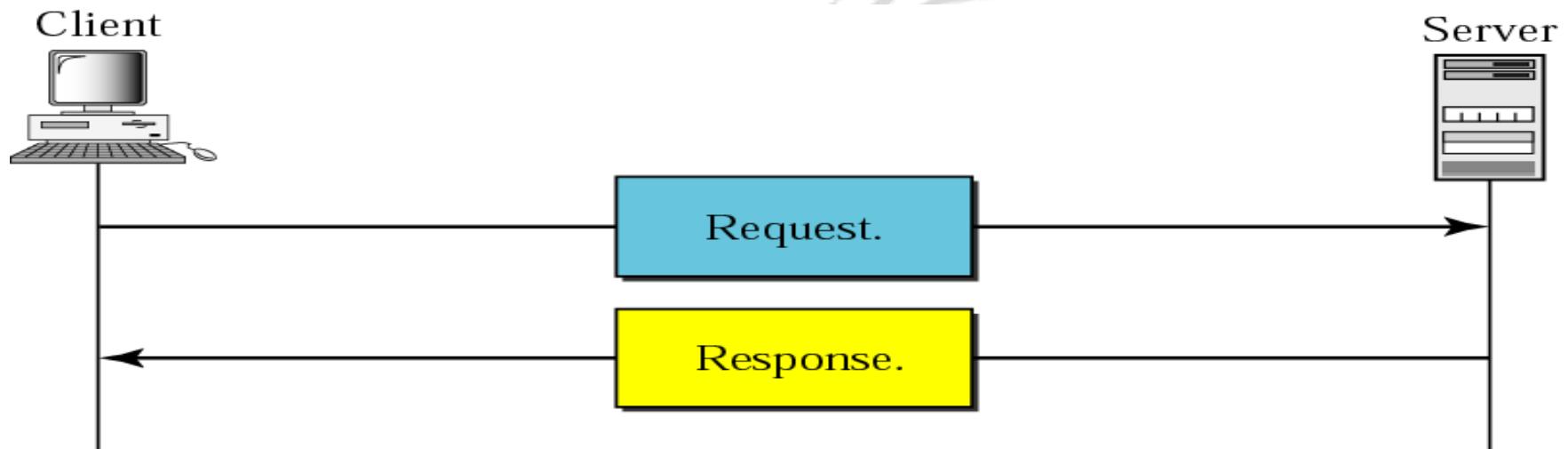
Protocol :// Host : Port / Path

HTTP

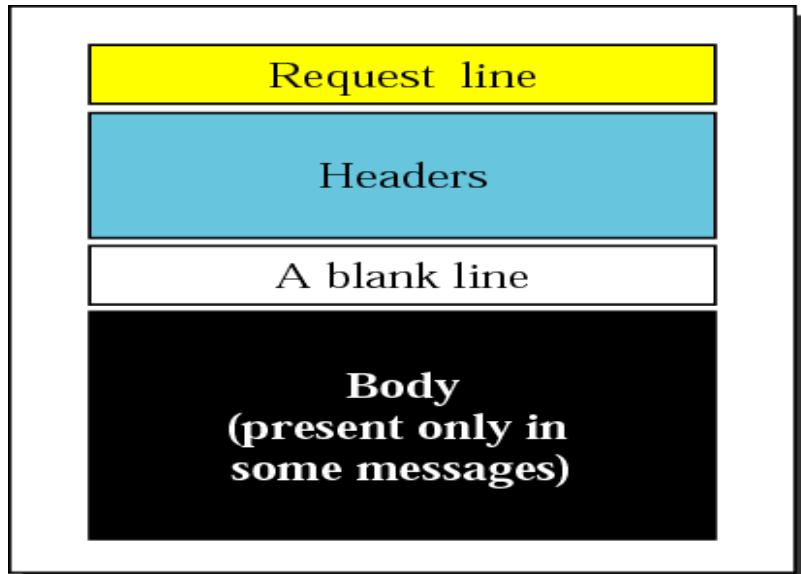
- The Hypertext Transfer Protocol (HTTP) is a protocol used mainly to access data on the World Wide Web
- HTTP functions like a combination of FTP and SMTP
- HTTP uses the services of TCP on well-known port 80



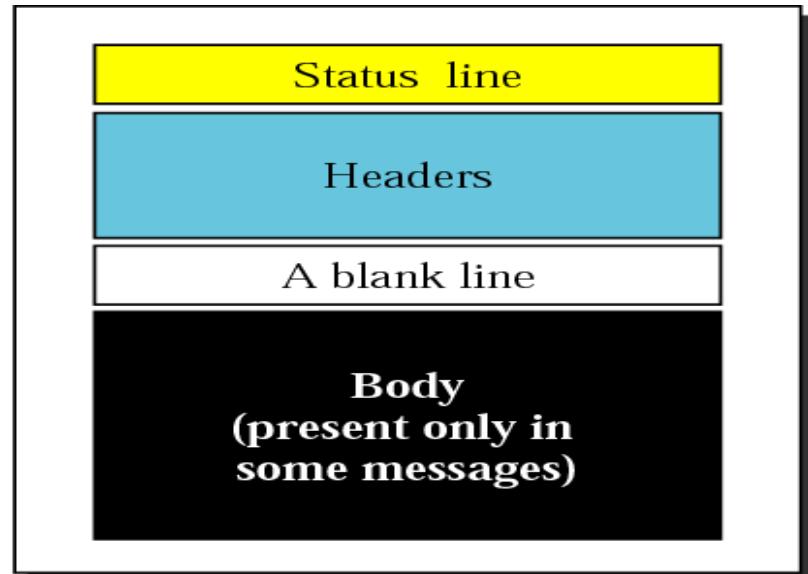
HTTP transaction



Request and response messages

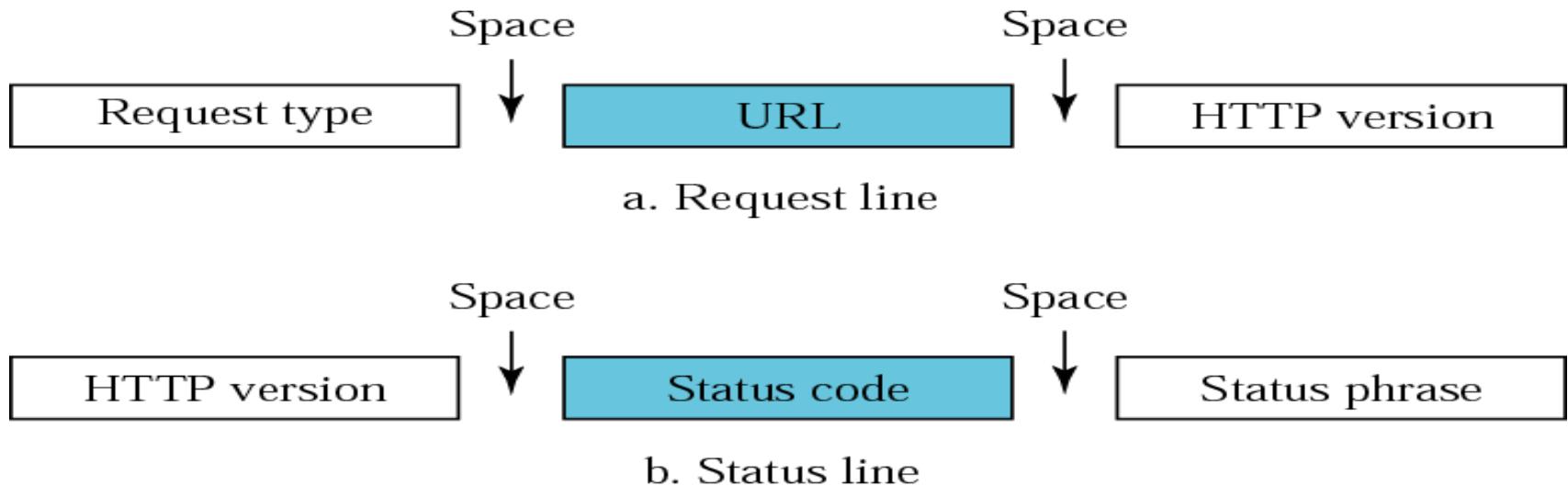


Request message



Response message

Request and status lines



Methods

<i>Method</i>	<i>Action</i>
GET	Requests a document from the server
HEAD	Requests information about a document but not the document itself
POST	Sends some information from the client to the server
PUT	Sends a document from the server to the client
TRACE	Echoes the incoming request
CONNECT	Reserved
OPTION	Enquires about available options

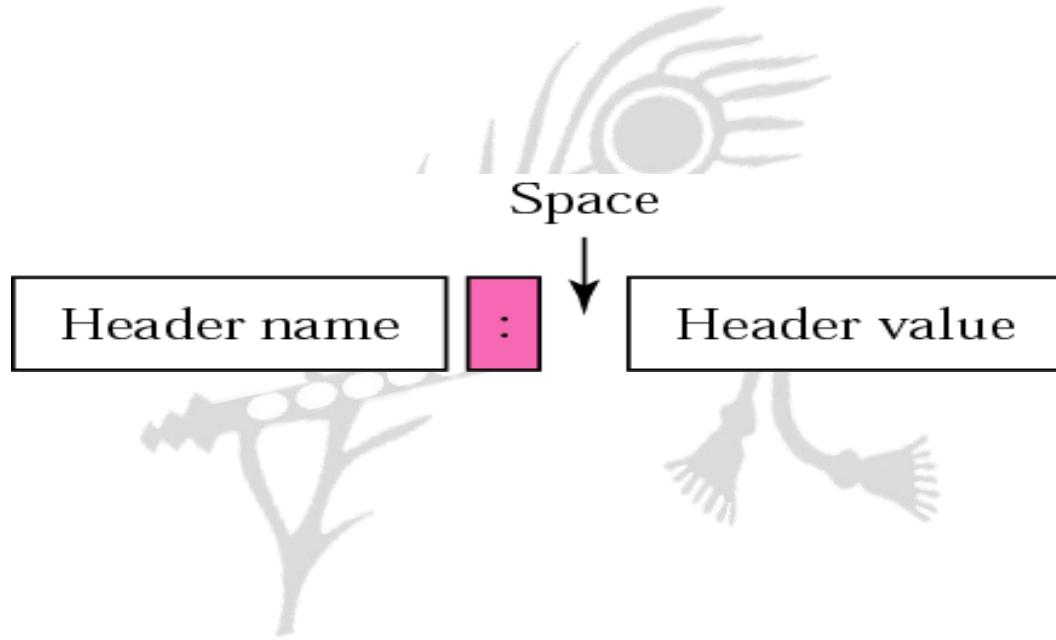
Status codes

<i>Code</i>	<i>Phrase</i>	<i>Description</i>
Informational		
100	Continue	The initial part of the request has been received and the client may continue with its request.
101	Switching	The server is complying with a client request to switch protocols defined in the upgrade header.
Success		
200	OK	The request is successful.
201	Created	A new URL is created.
202	Accepted	The request is accepted, but it is not immediately acted upon.
204	No content	There is no content in the body.

Status codes

<i>Code</i>	<i>Phrase</i>	<i>Description</i>
Redirection		
301	Multiple choices	The requested URL refers to more than one resource.
302	Moved permanently	The requested URL is no longer used by the server.
304	Moved temporarily	The requested URL has moved temporarily.
Client Error		
400	Bad request	There is a syntax error in the request.
401	Unauthorized	The request lacks proper authorization.
403	Forbidden	Service is denied.
404	Not found	The document is not found.
405	Method not allowed	The method is not supported in this URL.
406	Not acceptable	The format requested is not acceptable.
Server Error		
500	Internal server error	There is an error, such as a crash, at the server site.
501	Not implemented	The action requested cannot be performed.
503	Service unavailable	The service is temporarily unavailable, but may be requested in the future.

Header format



General headers

<i>Header</i>	<i>Description</i>
Cache-control	Specifies information about caching
Connection	Shows whether the connection should be closed or not
Date	Shows the current date
MIME-version	Shows the MIME version used
Upgrade	Specifies the preferred communication protocol

Request headers

<i>Header</i>	<i>Description</i>
Accept	Shows the media format the client can accept
Accept-charset	Shows the character set the client can handle
Accept-encoding	Shows the encoding scheme the client can handle
Accept-language	Shows the language the client can accept
Authorization	Shows what permissions the client has
From	Shows the e-mail address of the user
Host	Shows the host and port number of the client
If-modified-since	Send the document if newer than specified date
If-match	Send the document only if it matches given tag
If-non-match	Send the document only if it does not match given tag
If-range	Send only the portion of the document that is missing
If-unmodified-since	Send the document if not changed since specified date
Referrer	Specifies the URL of the linked document
User-agent	Identifies the client program

Response headers

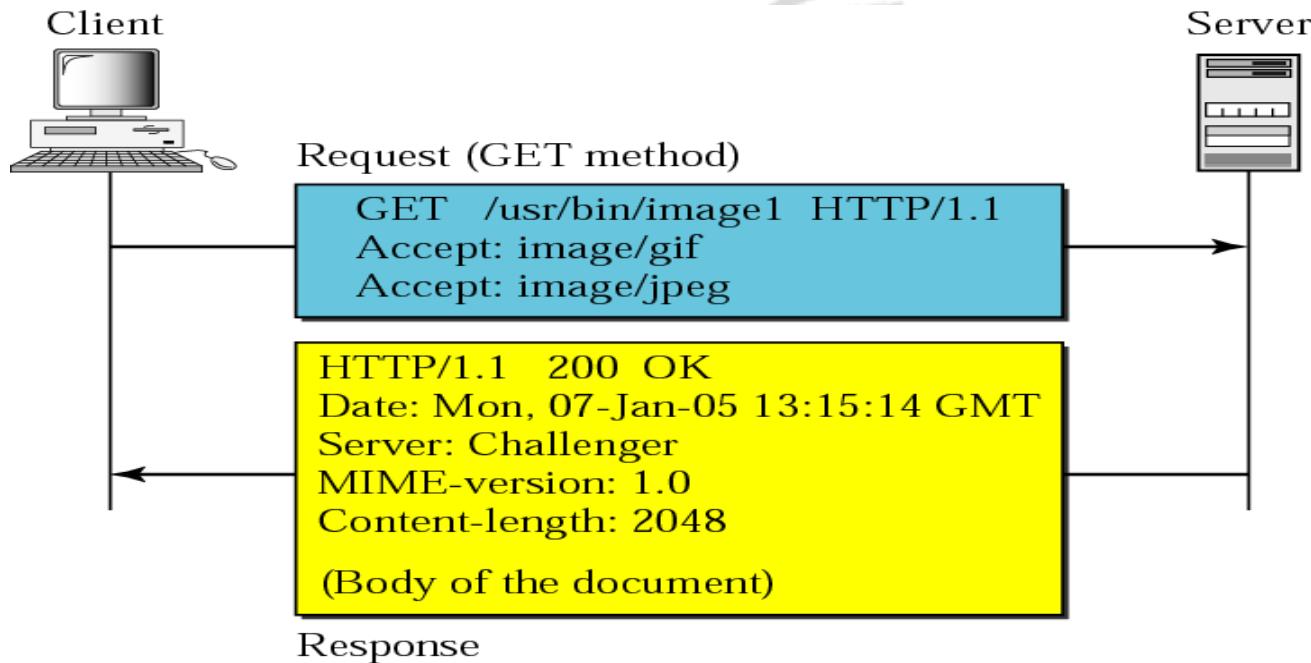
<i>Header</i>	<i>Description</i>
Accept-range	Shows if server accepts the range requested by client
Age	Shows the age of the document
Public	Shows the supported list of methods
Retry-after	Specifies the date after which the server is available
Server	Shows the server name and version number

Entity headers

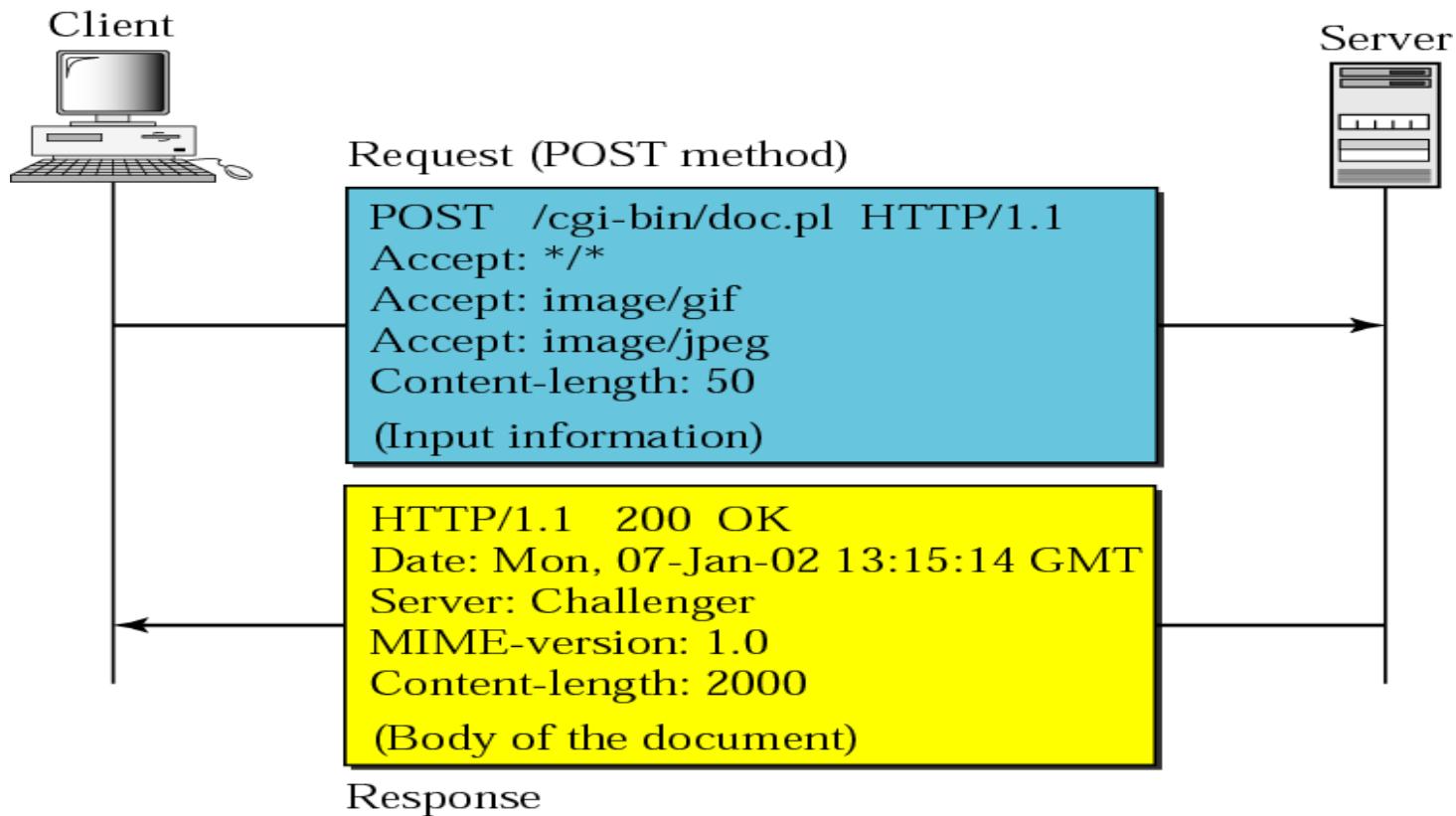
<i>Header</i>	<i>Description</i>
Allow	Lists valid methods that can be used with a URL
Content-encoding	Specifies the encoding scheme
Content-language	Specifies the language
Content-length	Shows the length of the document
Content-range	Specifies the range of the document
Content-type	Specifies the media type
Etag	Gives an entity tag
Expires	Gives the date and time when contents may change
Last-modified	Gives the date and time of the last change
Location	Specifies the location of the created or moved document

Example

- HTTP version 1.1 specifies a persistent connection by default

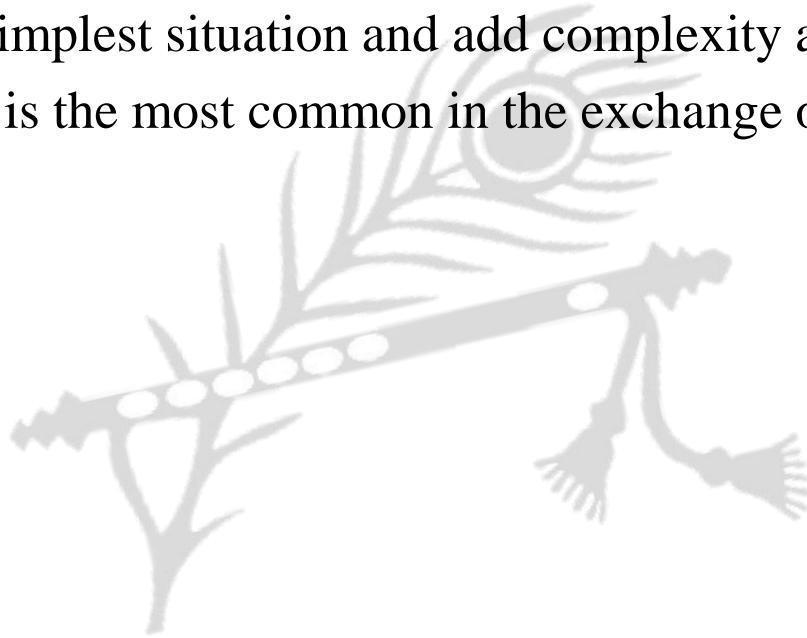


Example



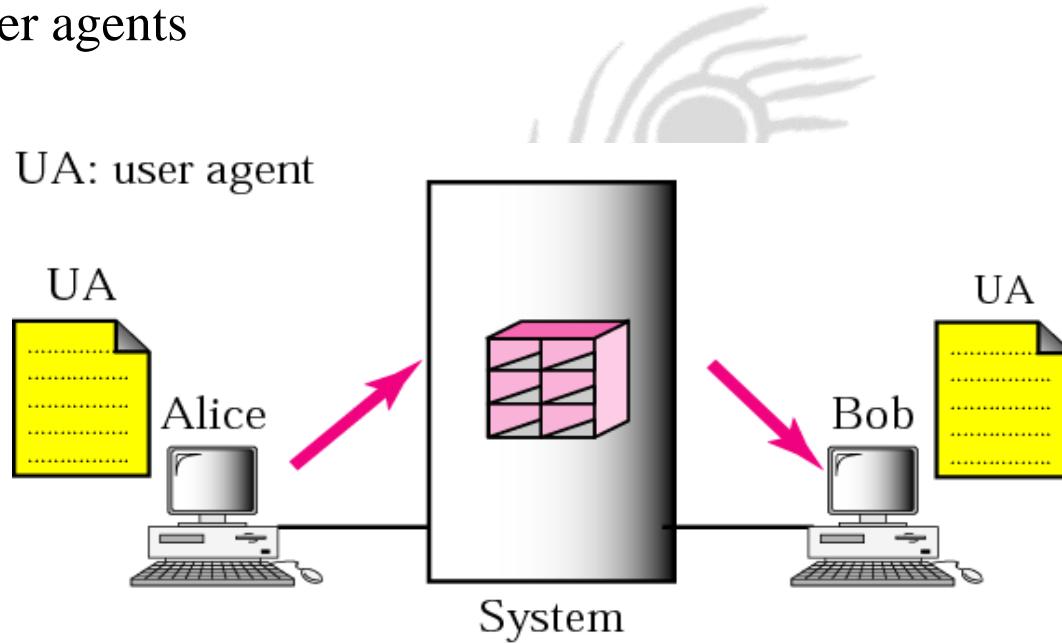
SMTP

- To explain the architecture of email, we give few scenarios
- We begin with the simplest situation and add complexity as we proceed
- The fourth scenario is the most common in the exchange of email



First scenario

- When the sender and the receiver of an email are on the same system, we need only two user agents

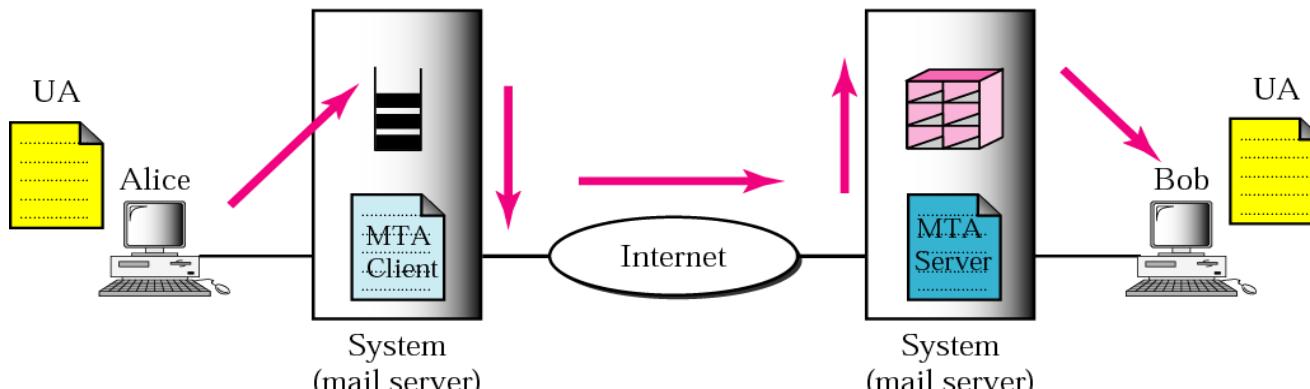


Second scenario

- When the sender and the receiver of an email are on different systems, we need two UAs and a pair of MTAs (client and server)

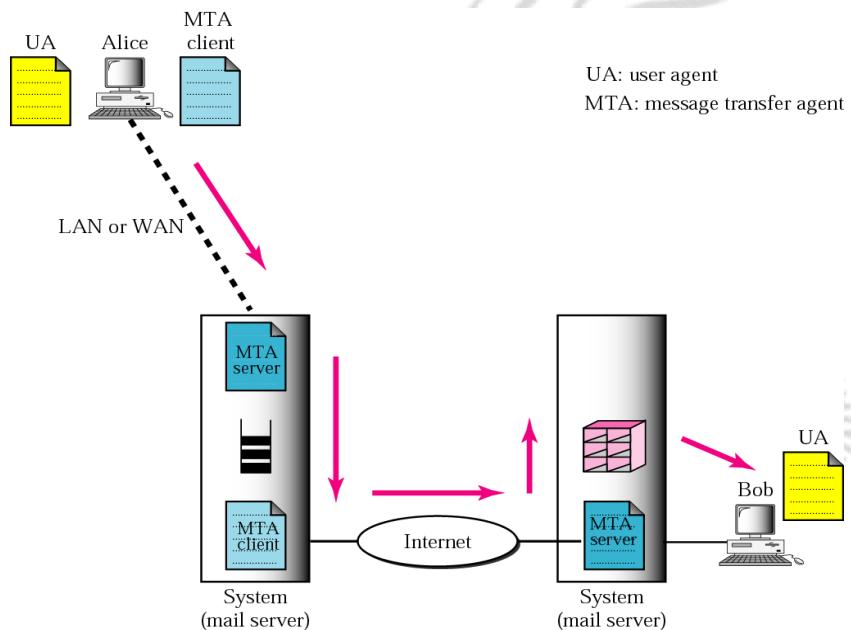
UA: user agent

MTA: message transfer agent



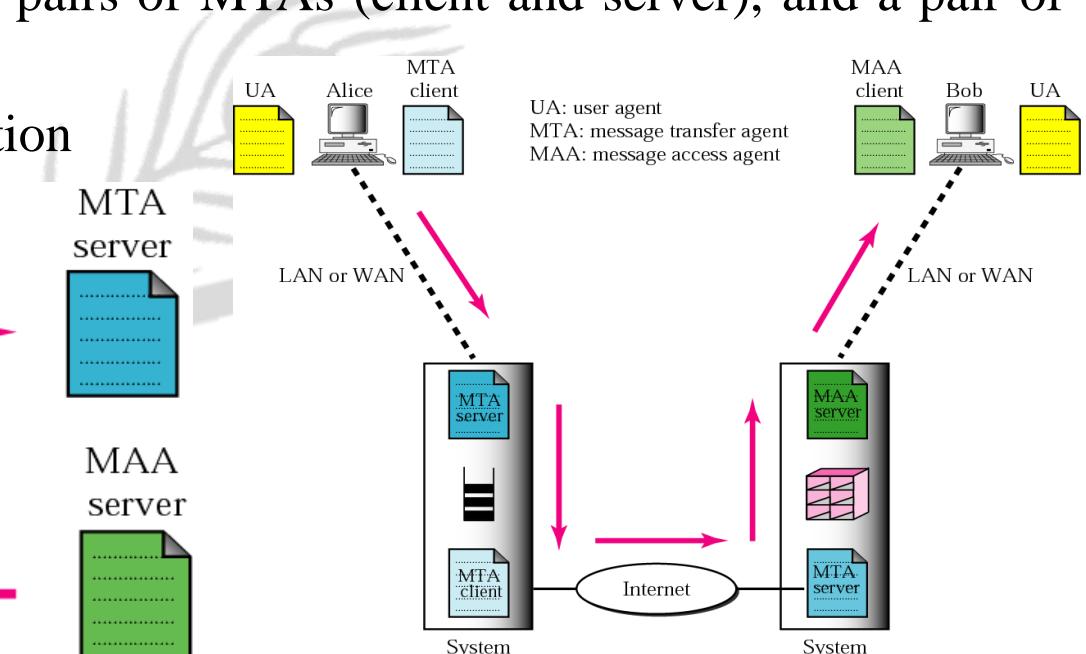
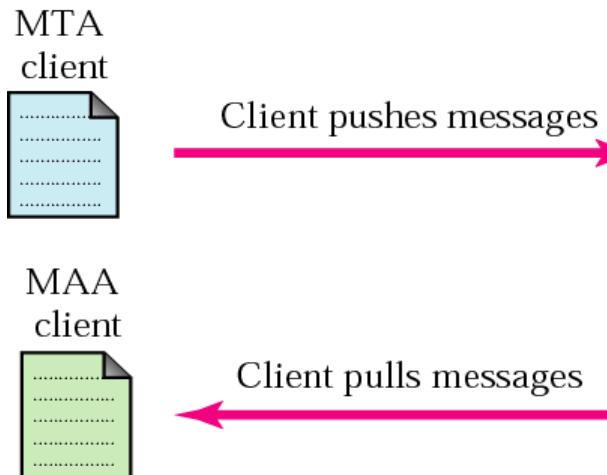
Third scenario

- When the sender is connected to the mail server via a LAN or a WAN, we need two UAs and two pairs of MTAs (client and server)



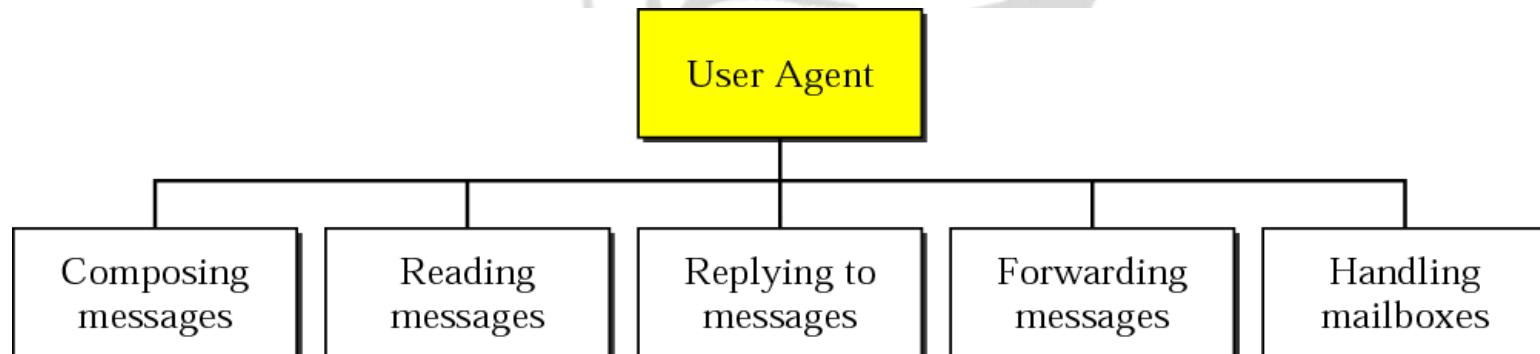
Fourth scenario

- When both sender and receiver are connected to the mail server via a LAN or a WAN, we need two UAs, two pairs of MTAs (client and server), and a pair of MAAs (client and server)
- This is the most common situation

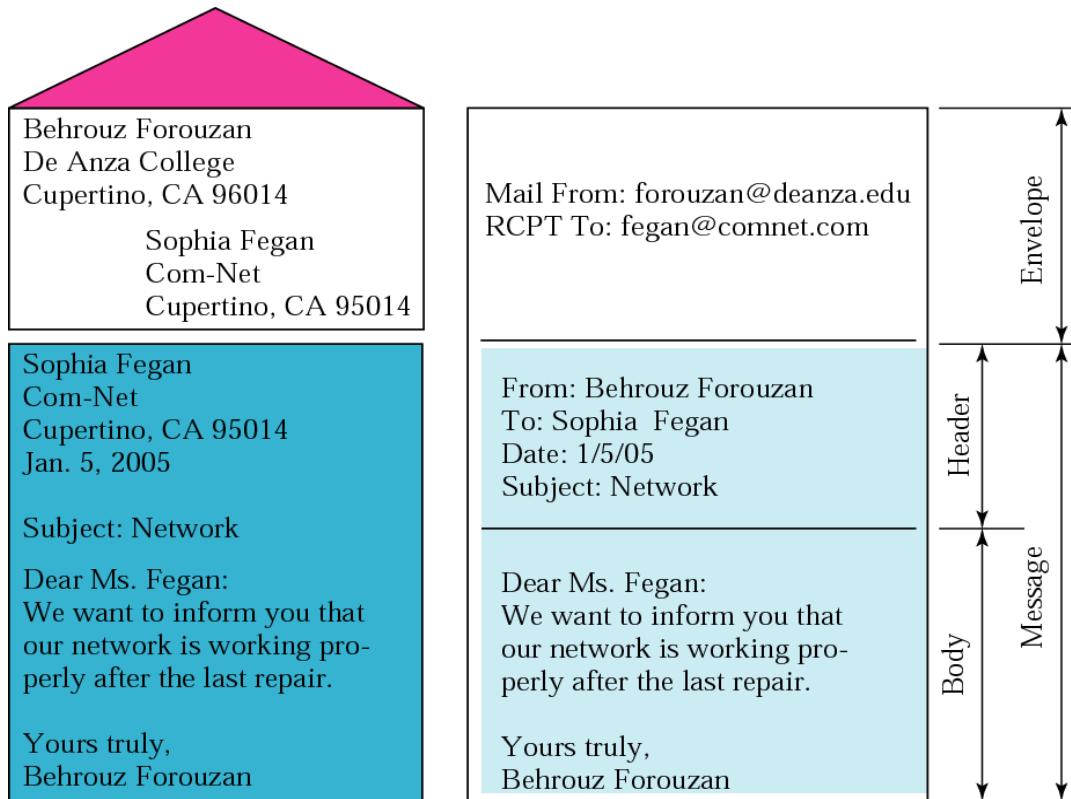


User Agent

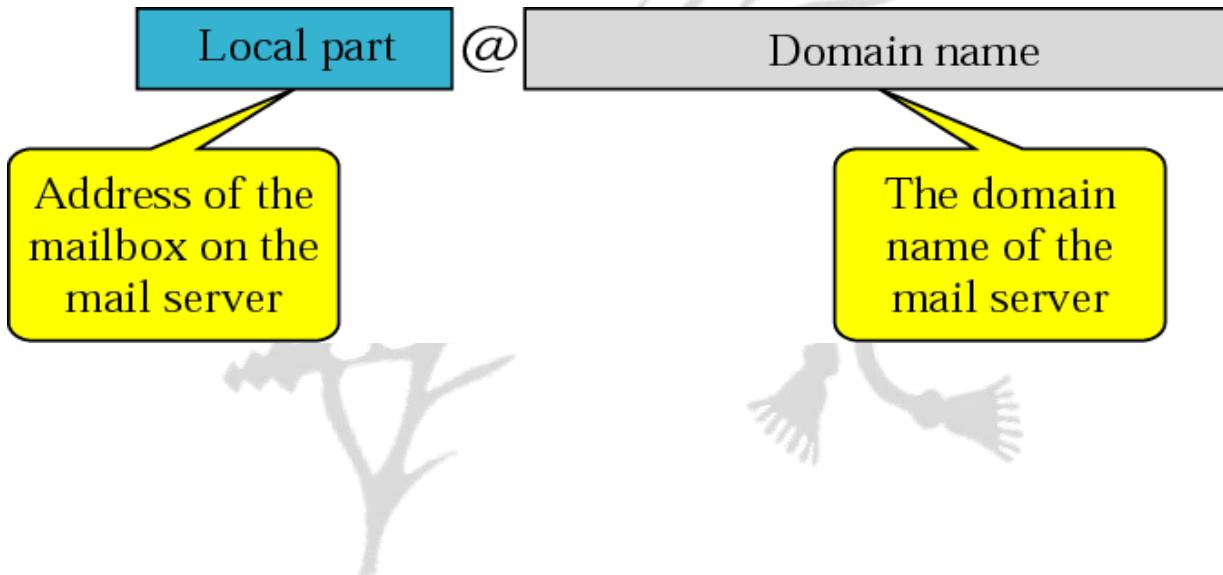
- The user agent (UA) provides service to the user to make the process of sending and receiving a message easier
- Some examples of command-driven user agents are mail, pine, and elm
- Some examples of GUI-based user agents are Eudora, Outlook, and Netscape



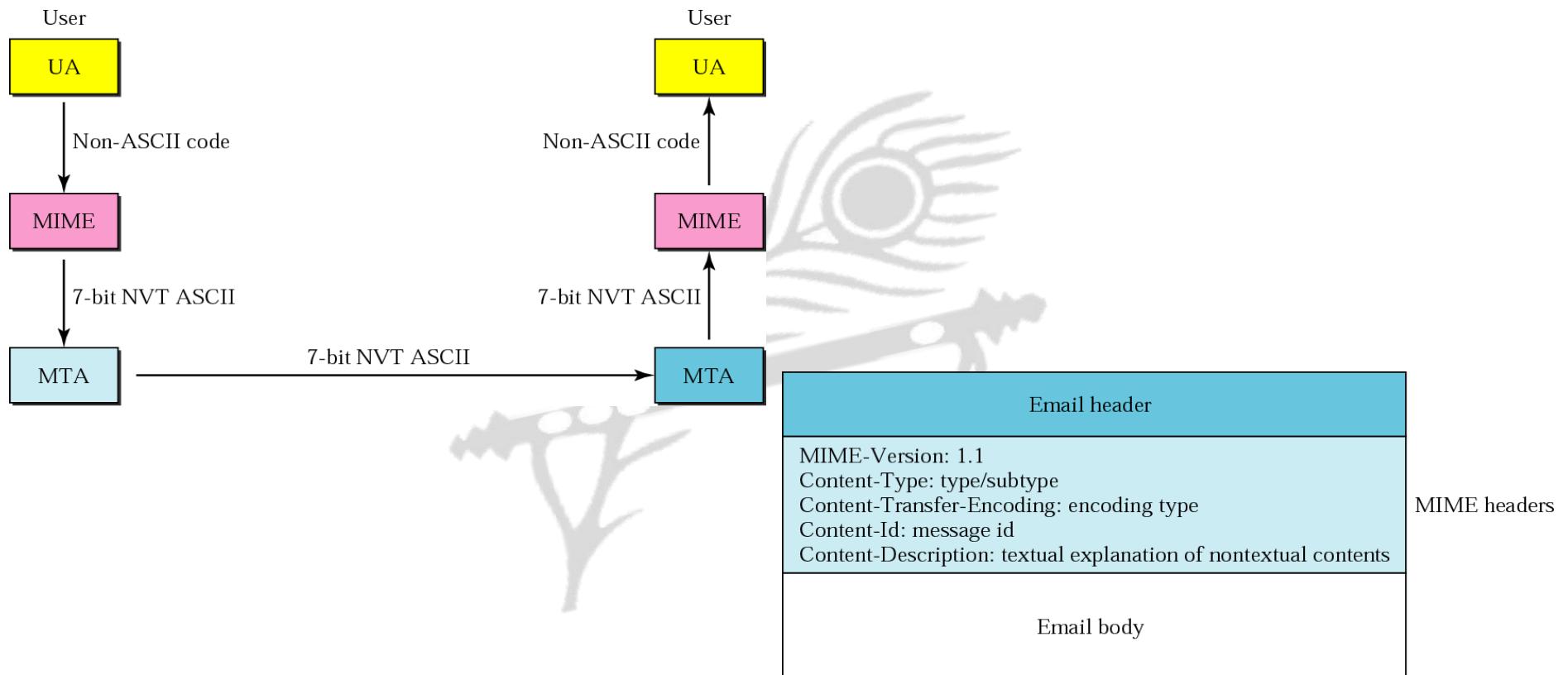
Format of an email



Email address



MIME



Data types in sub types in MIME

Type	Subtype	Description
Text	Plain	Unformatted
	HTML	HTML format (see Chapter 22)
Multipart	Mixed	Body contains ordered parts of different data types
	Parallel	Same as above, but no order
	Digest	Similar to Mixed, but the default is message/RFC822
	Alternative	Parts are different versions of the same message



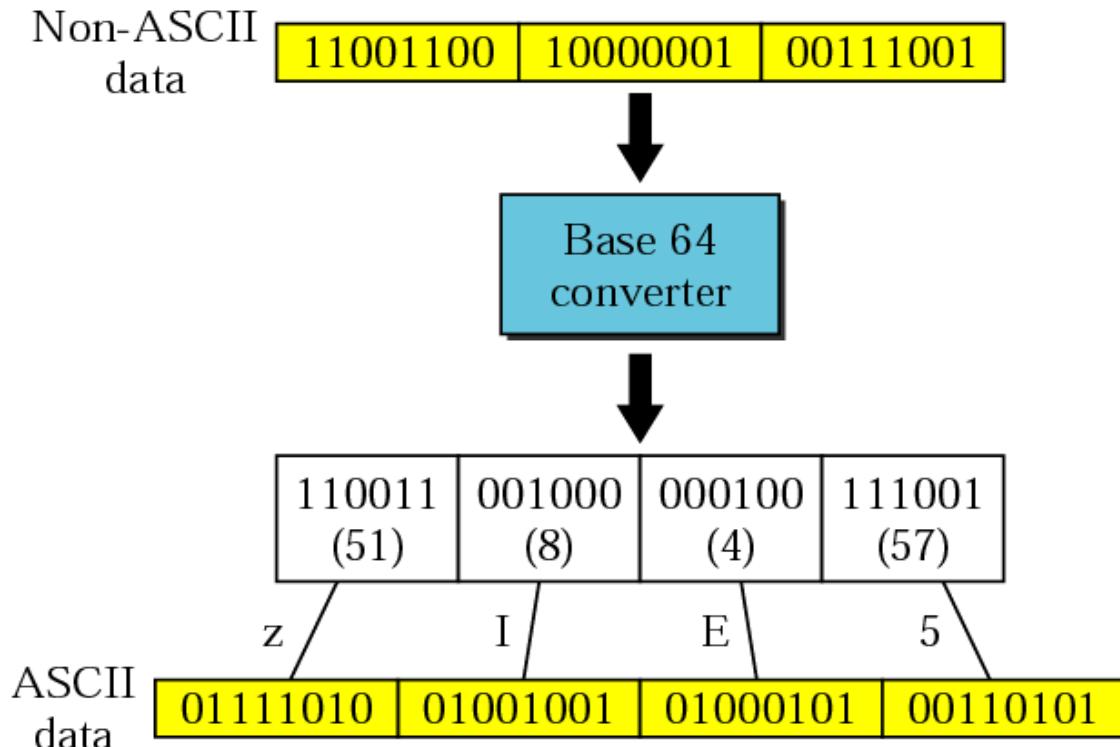
Type	Subtype	Description
Message	RFC822	Body is an encapsulated message
	Partial	Body is a fragment of a bigger message
	External-Body	Body is a reference to another message
Image	JPEG	Image is in JPEG format
	GIF	Image is in GIF format
Video	MPEG	Video is in MPEG format
Audio	Basic	Single channel encoding of voice at 8 KHz
Application	PostScript	Adobe PostScript
	Octet-stream	General binary data (eight-bit bytes)

Content transfer encoding

<i>Type</i>	<i>Description</i>
7bit	NVT ASCII characters and short lines
8bit	Non-ASCII characters and short lines
Binary	Non-ASCII characters with unlimited-length lines
Base64	6-bit blocks of data are encoded into 8-bit ASCII characters
Quoted-printable	Non-ASCII characters are encoded as an equal sign followed by an ASCII code



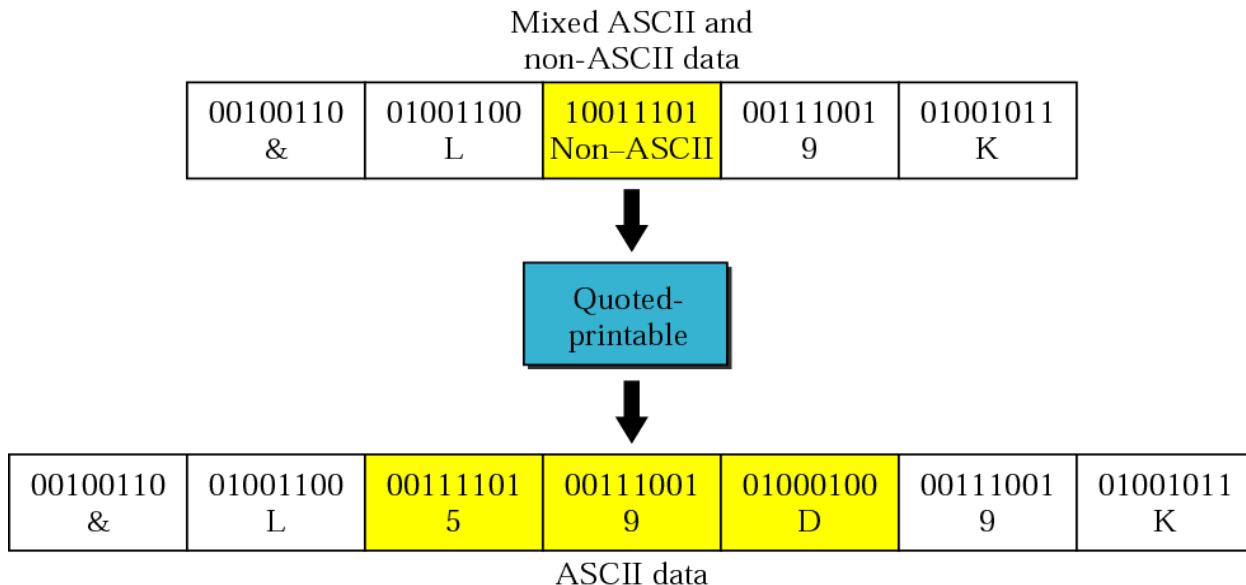
Base64



Base64 encoding table

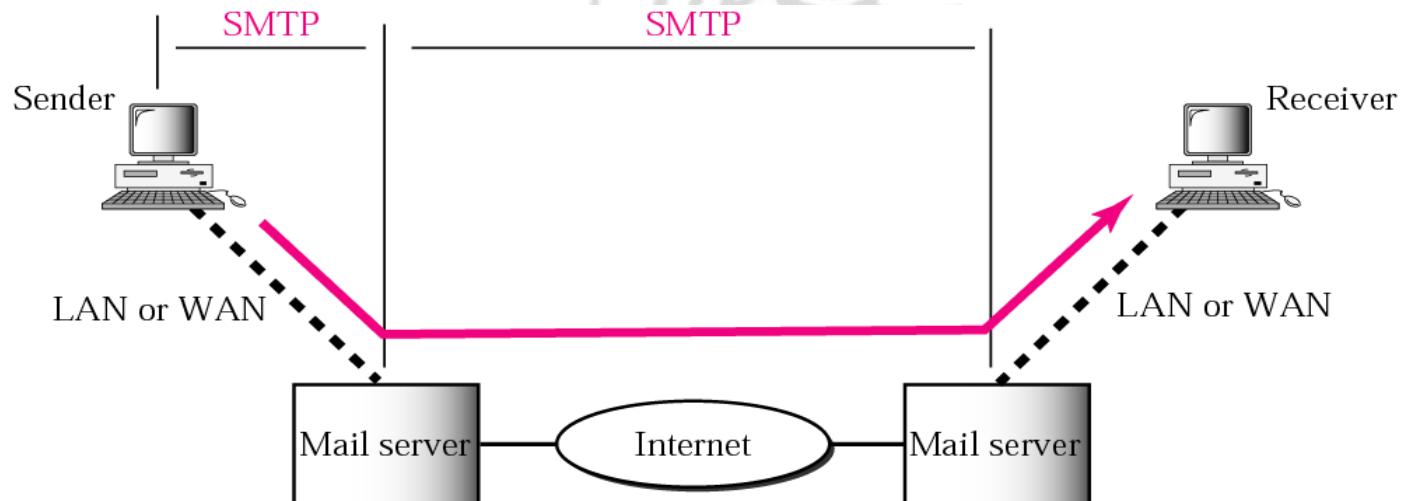
<i>Value</i>	<i>Code</i>										
0	A	11	L	22	W	33	h	44	s	55	3
1	B	12	M	23	X	34	i	45	t	56	4
2	C	13	N	24	Y	35	j	46	u	57	5
3	D	14	O	25	Z	36	k	47	v	58	6
4	E	15	P	26	a	37	l	48	w	59	7
5	F	16	Q	27	b	38	m	49	x	60	8
6	G	17	R	28	c	39	n	50	y	61	9
7	H	18	S	29	d	40	o	51	z	62	+
8	I	19	T	30	e	41	p	52	0	63	/
9	J	20	U	31	f	42	q	53	1		
10	K	21	V	32	g	43	r	54	2		

Quoted-printable

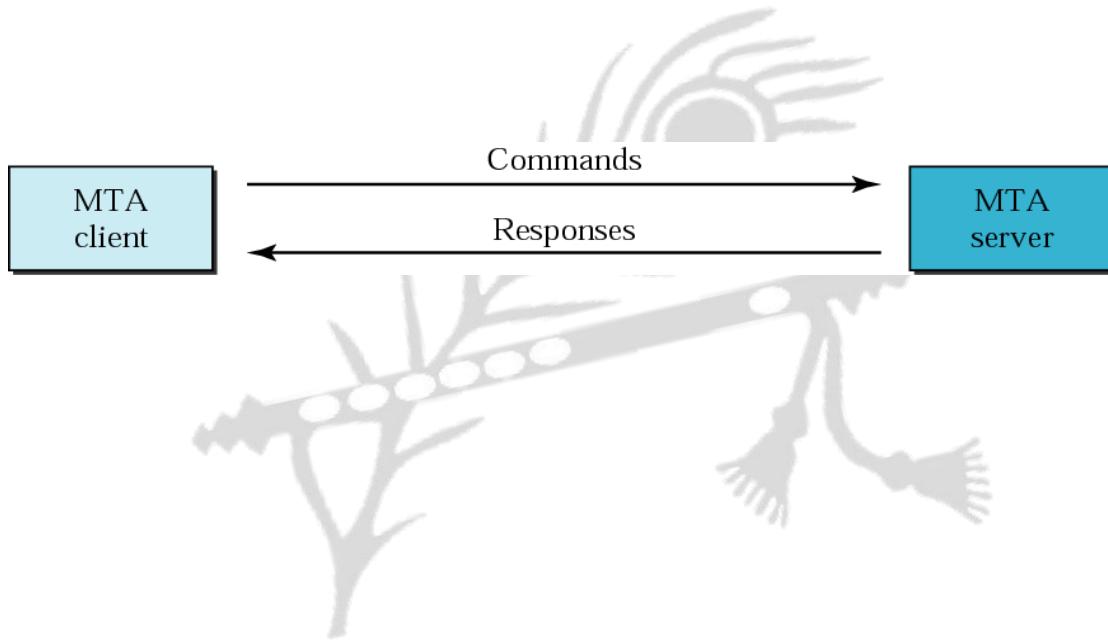


MTA

- The actual mail transfer requires message transfer agents (MTAs)
- The protocol that defines the MTA client and server in the Internet is called Simple Mail Transfer Protocol (SMTP)



Commands and responses



Command format

Keyword: argument(s)

Commands

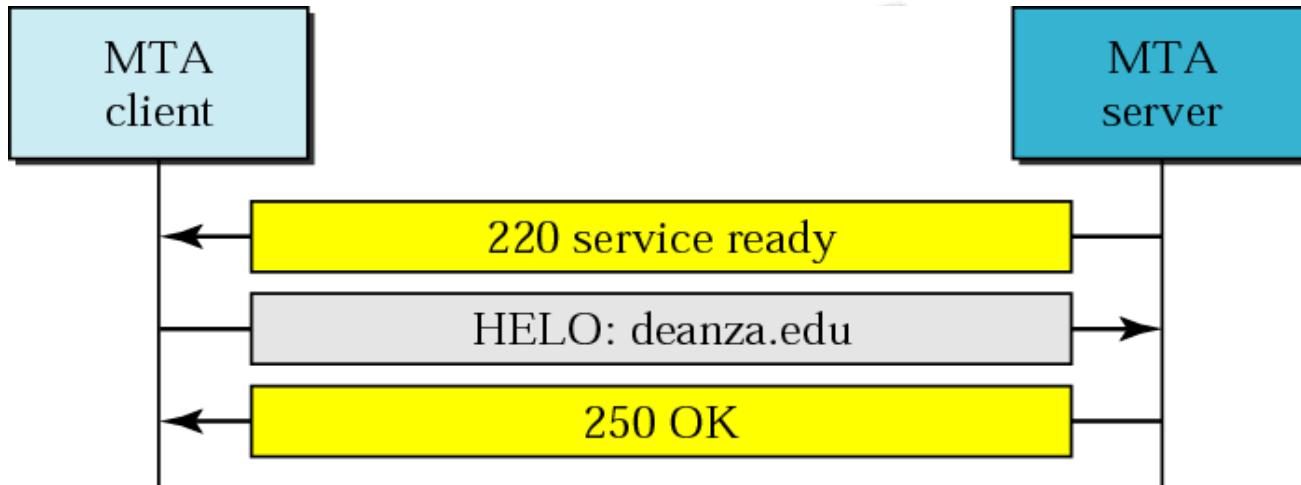
<i>Keyword</i>	<i>Argument(s)</i>
HELO	Sender's host name
MAIL FROM	Sender of the message
RCPT TO	Intended recipient of the message
DATA	Body of the mail
QUIT	
RSET	
VRFY	Name of recipient to be verified
NOOP	
TURN	
EXPN	Mailing list to be expanded
HELP	Command name
SEND FROM	Intended recipient of the message
SMOL FROM	Intended recipient of the message
SMAL FROM	Intended recipient of the message

Responses

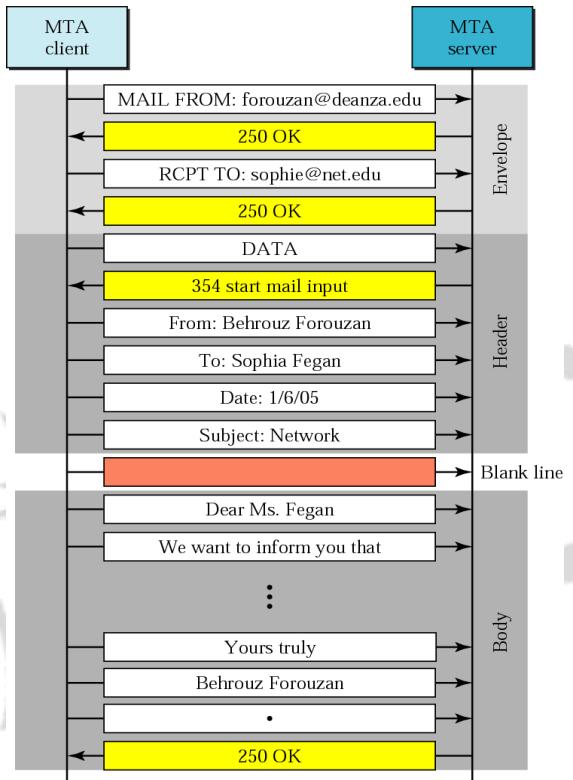
<i>Code</i>	<i>Description</i>
Positive Completion Reply	
211	System status or help reply
214	Help message
220	Service ready
221	Service closing transmission channel
250	Request command completed
251	User not local; the message will be forwarded
Positive Intermediate Reply	
354	Start mail input
Transient Negative Completion Reply	
421	Service not available
450	Mailbox not available
451	Command aborted: local error
452	Command aborted; insufficient storage
Permanent Negative Completion Reply	

Permanent Negative Completion Reply	
500	Syntax error; unrecognized command
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command temporarily not implemented
550	Command is not executed; mailbox unavailable
551	User not local
552	Requested action aborted; exceeded storage location
553	Requested action not taken; mailbox name not allowed
554	Transaction failed

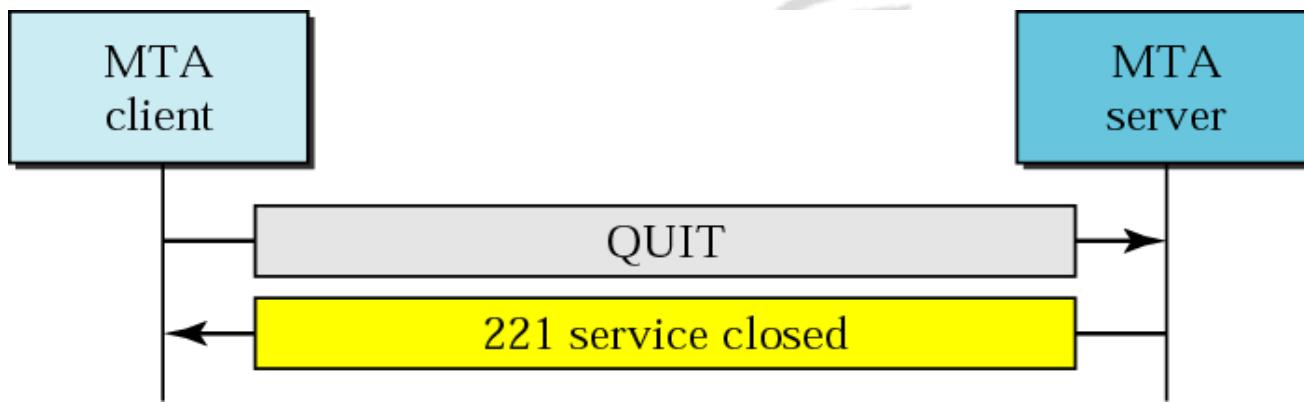
Connection establishment



Message transfer



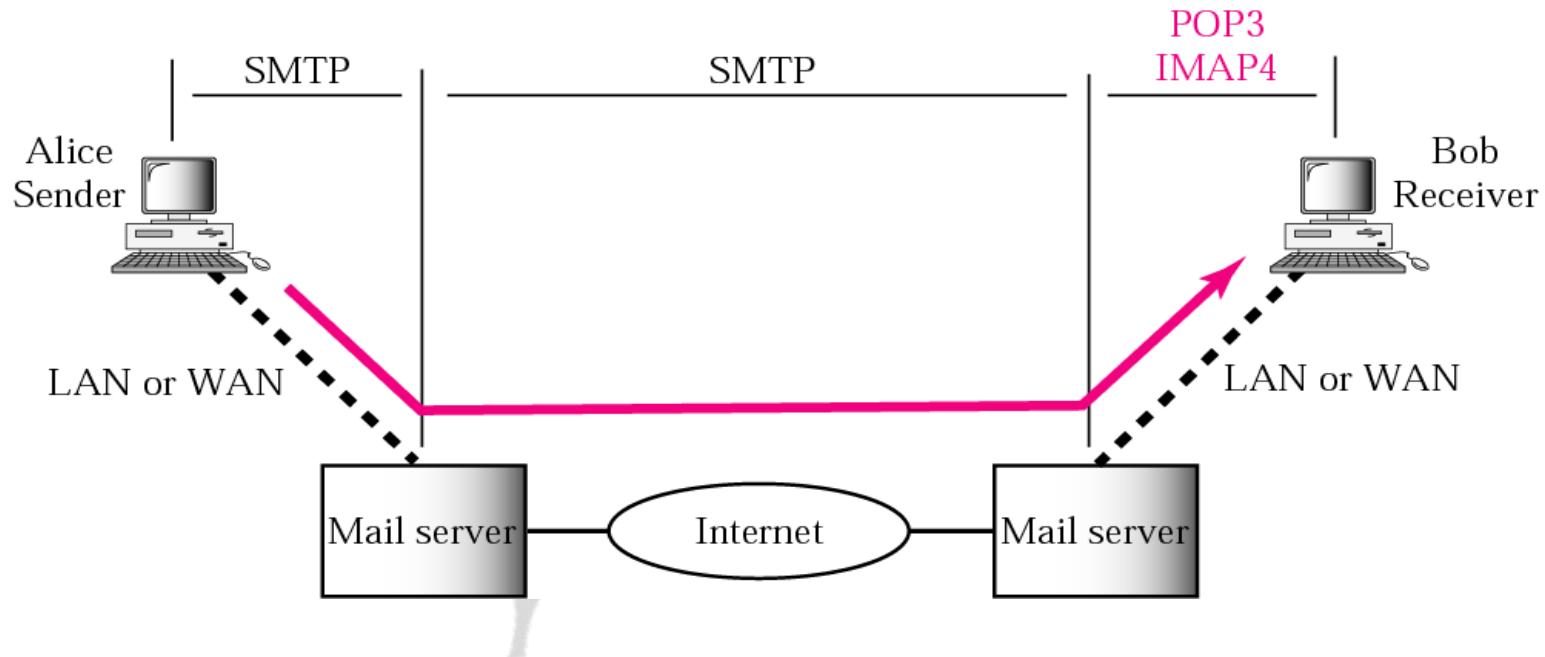
Connection termination



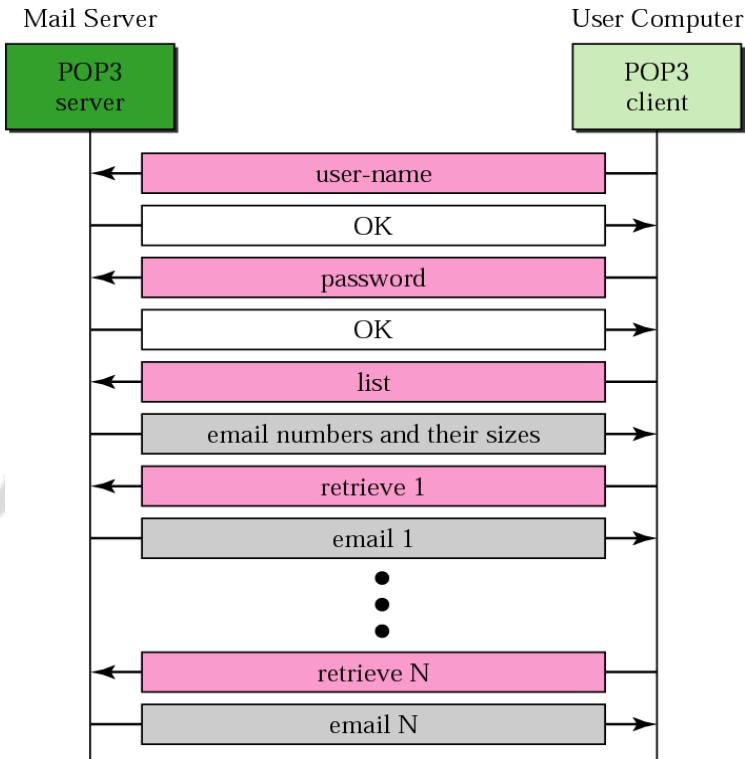
Message Access Agent: POP and IMAP

- The third stage of mail delivery uses a message access agent; the client must pull messages from the server
- Currently two message access protocols are available
 - Post Office Protocol, version 3 (POP3)
 - Internet Mail Access Protocol, version 4

POP3 and IMAP4

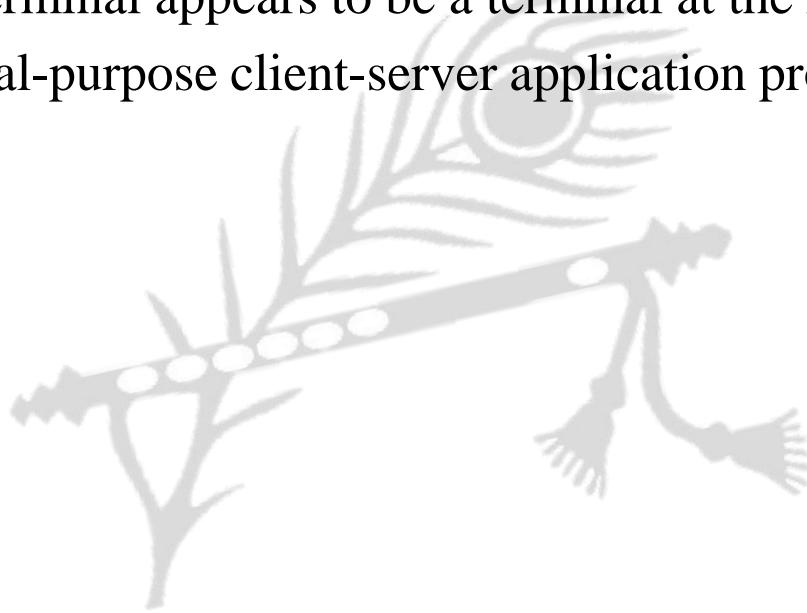


POP3

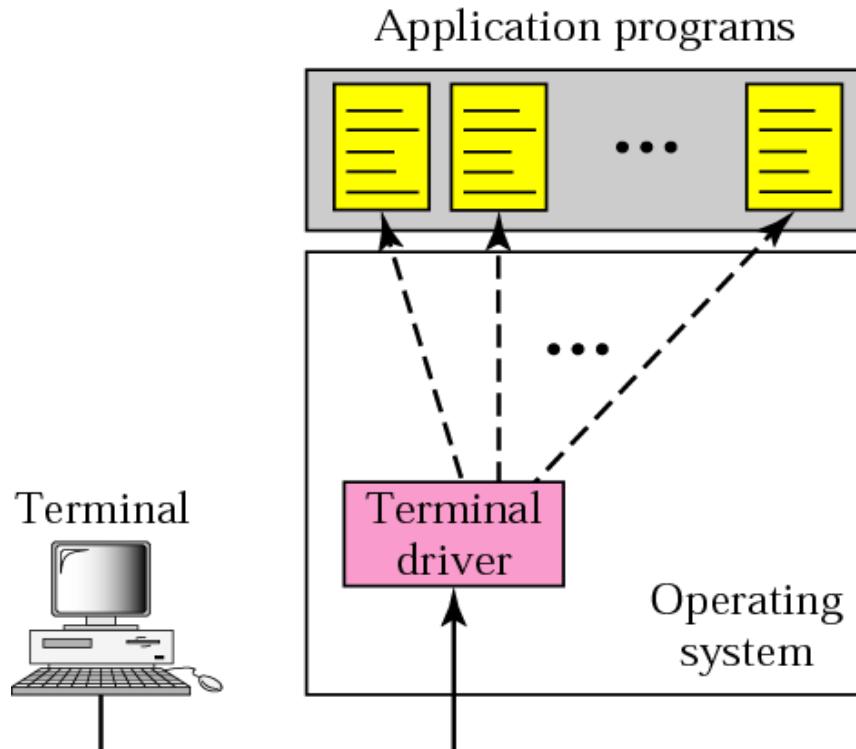


Telnet

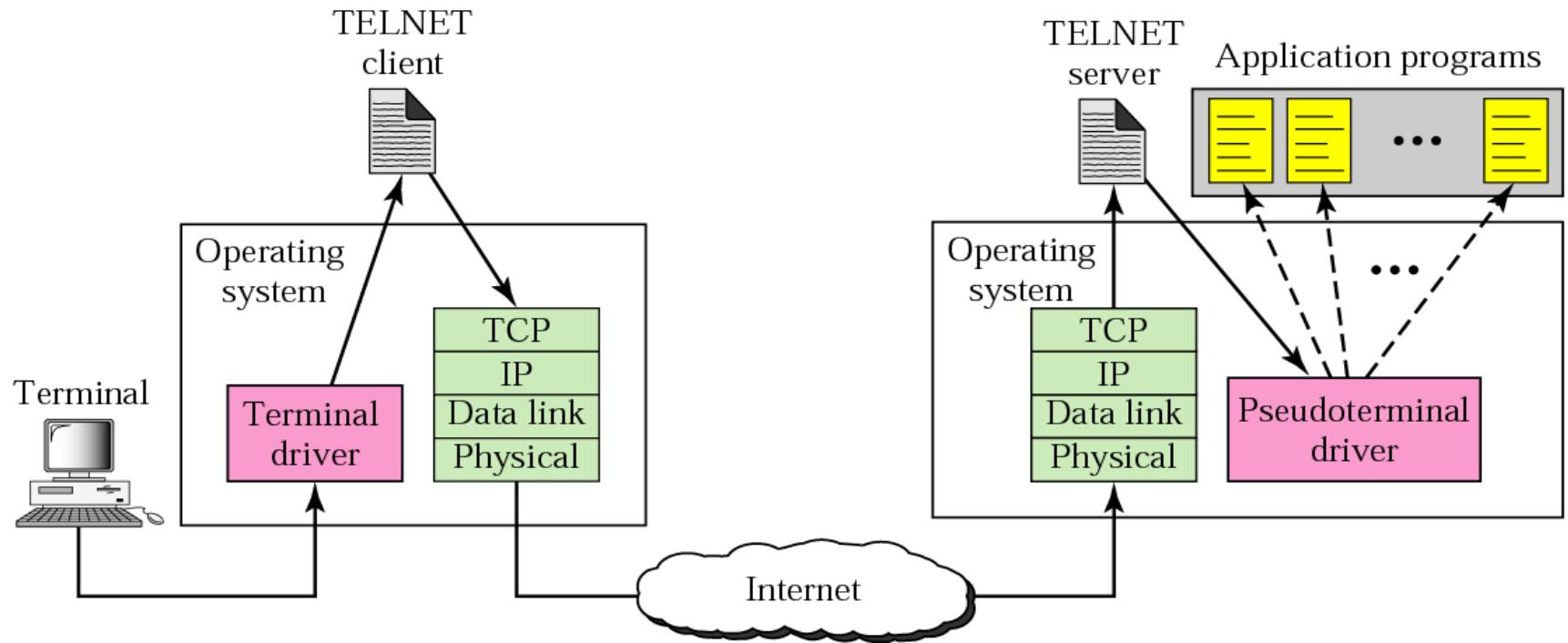
- TELNET enables the establishment of a connection to a remote system in such a way that the local terminal appears to be a terminal at the remote system
- TELNET is a general-purpose client-server application program



Local login



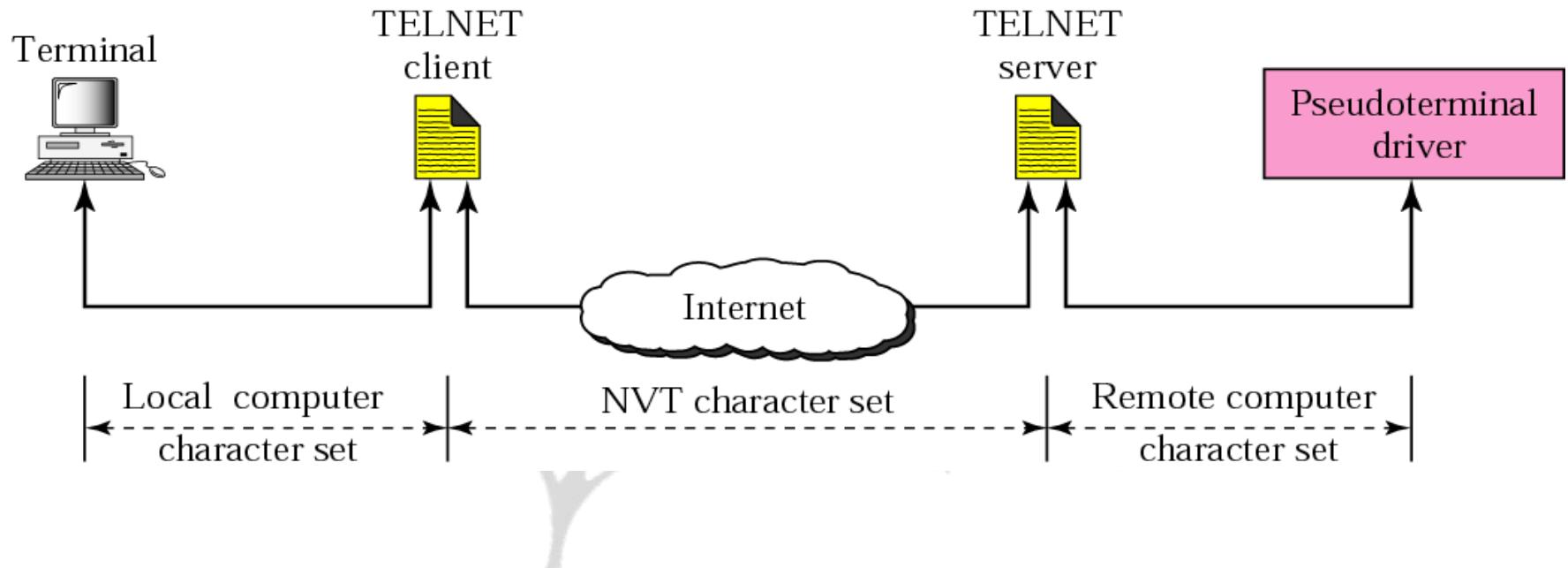
Remote login



Network Virtual Terminal (NVT)

- Via a universal interface called the Network Virtual Terminal (NVT) character set, the TELNET client translates characters (data or commands) that come from the local terminal into NVT form and delivers them to the network
- TELNET server translates data and commands from NVT form into the form acceptable by the remote computer.

Concept of NVT



NVT character set

- NVT uses two sets of characters, one for data and one for control. Both are 8-bit bytes

0							
---	--	--	--	--	--	--	--

Format of data characters

1							
---	--	--	--	--	--	--	--

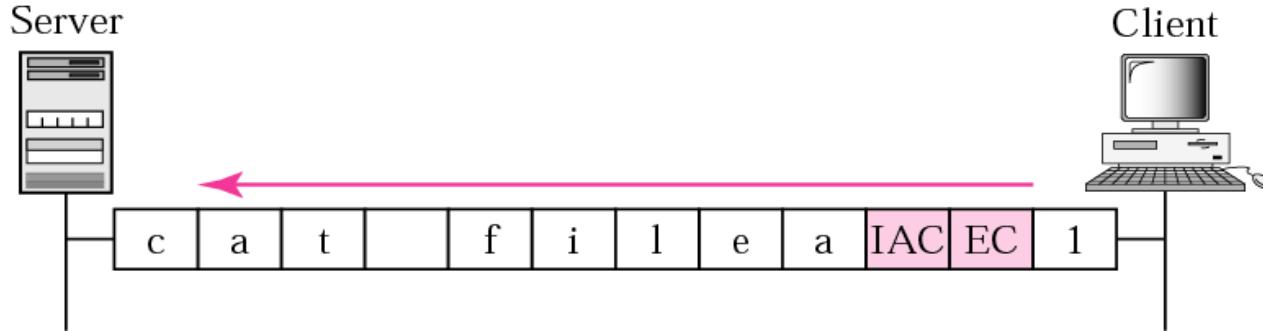
Format of control characters

NVT control characters

<i>Character</i>	<i>Decimal</i>	<i>Binary</i>	<i>Meaning</i>
EOF	236	11101100	End of file
EOR	239	11101111	End of record
SE	240	11110000	Suboption end
NOP	241	11110001	No operation
DM	242	11110010	Data mark
BRK	243	11110011	Break
IP	244	11110100	Interrupt process
AO	245	11110101	Abort output
AYT	246	11110110	Are you there?
EC	247	11110111	Erase character
EL	248	11111000	Erase line
GA	249	11111001	Go ahead
SB	250	11111010	Suboption begin
WILL	251	11111011	Agreement to enable option
WONT	252	11111100	Refusal to enable option
DO	253	11111101	Approval to option request
DONT	254	11111110	Denial of option request
IAC	255	11111111	Interpret (the next character) as control

Embedding

- The same connection is used by TELNET for sending both data and control characters
- TELNET accomplishes this by embedding the control characters in the data stream



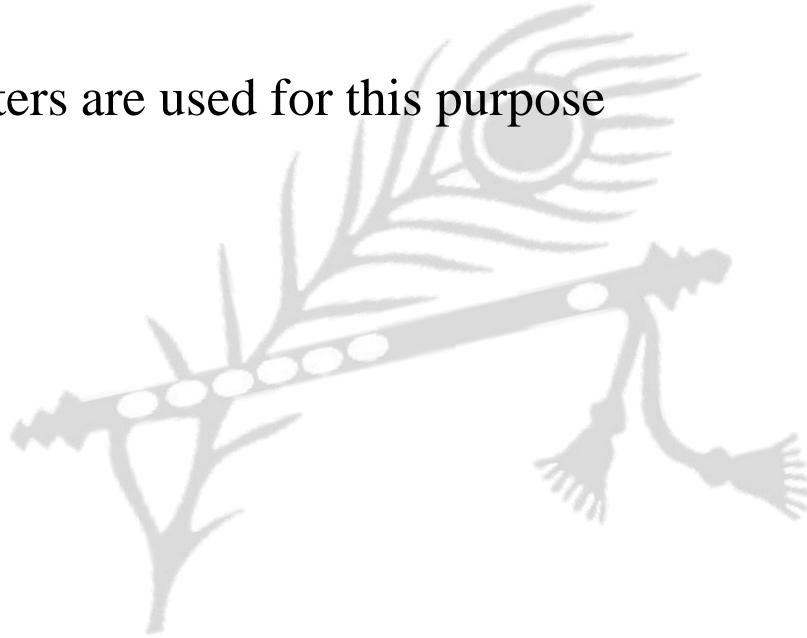
Options

- TELNET lets the client and server negotiate options before or during the use of the service
- Options are extra features available to a user with a more sophisticated terminal

<i>Code</i>	<i>Option</i>	<i>Meaning</i>
0	Binary	Interpret as 8-bit binary transmission
1	Echo	Echo the data received on one side to the other
3	Suppress go ahead	Suppress go-ahead signals after data
5	Status	Request the status of TELNET
6	Timing mark	Define the timing marks
24	Terminal type	Set the terminal type
32	Terminal speed	Set the terminal speed
34	Line mode	Change to line mode

Option negotiation

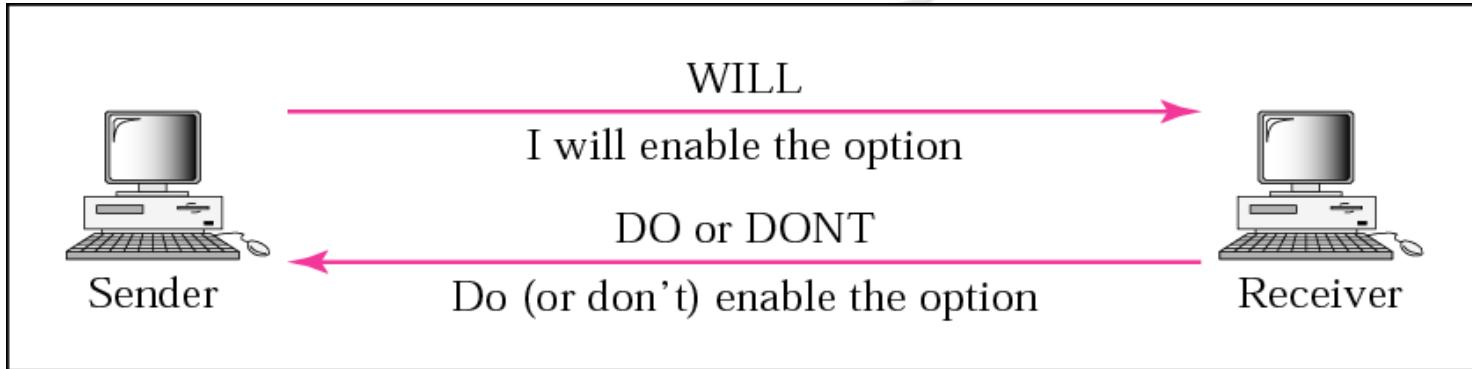
- To use any of the options first requires option negotiation between the client and the server
- Four control characters are used for this purpose



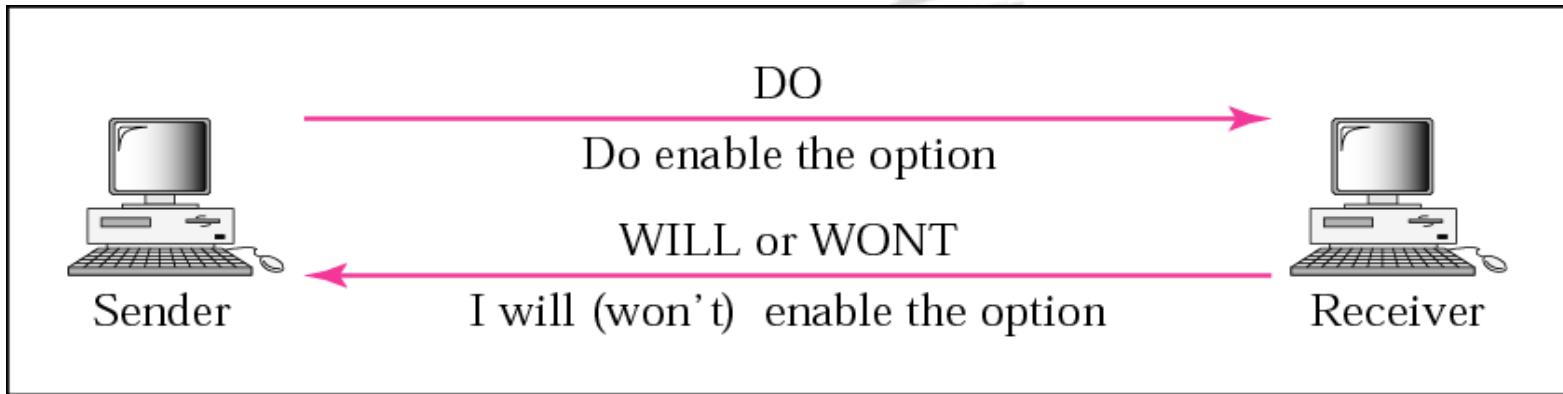
NVT character set for option negotiation

<i>Character</i>	<i>Decimal</i>	<i>Binary</i>	<i>Meaning</i>
WILL	251	11111011	1. Offering to enable 2. Accepting a request to enable
WONT	252	11111100	1. Rejecting a request to enable 2. Offering to disable 3. Accepting a request to disable
DO	253	11111101	1. Approving an offer to enable 2. Requesting to enable
DONT	254	11111110	1. Disapproving an offer to enable 2. Approving an offer to disable 3. Requesting to disable

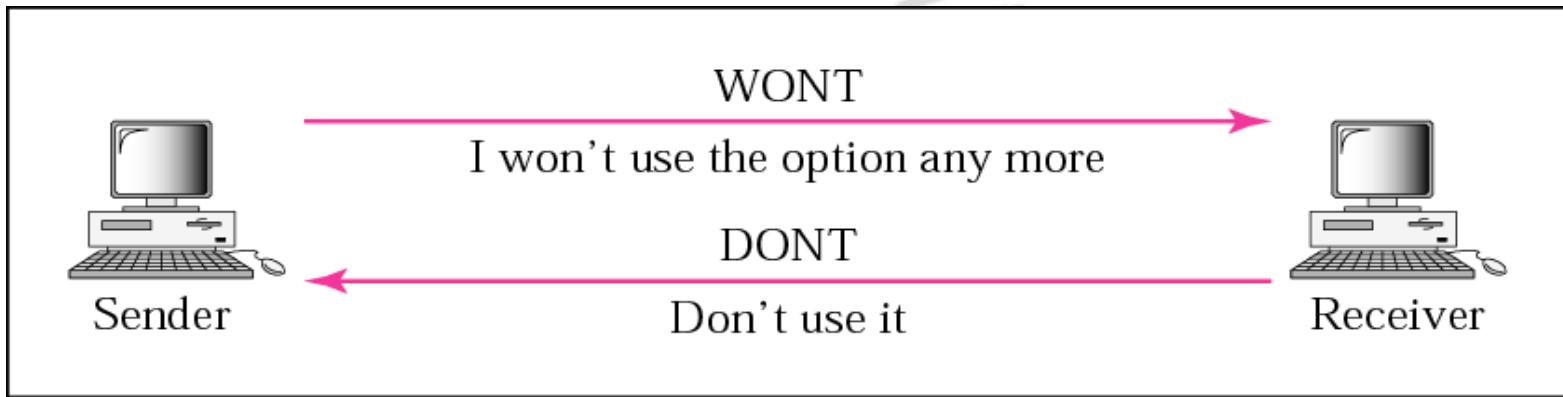
Offer to enable an option



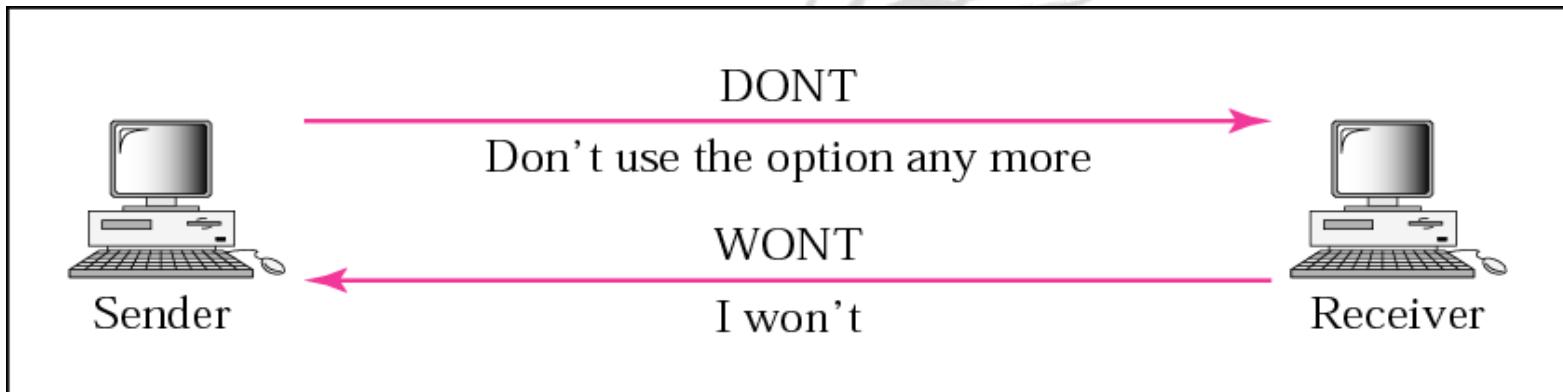
Request to enable an option



Offer to disable an option



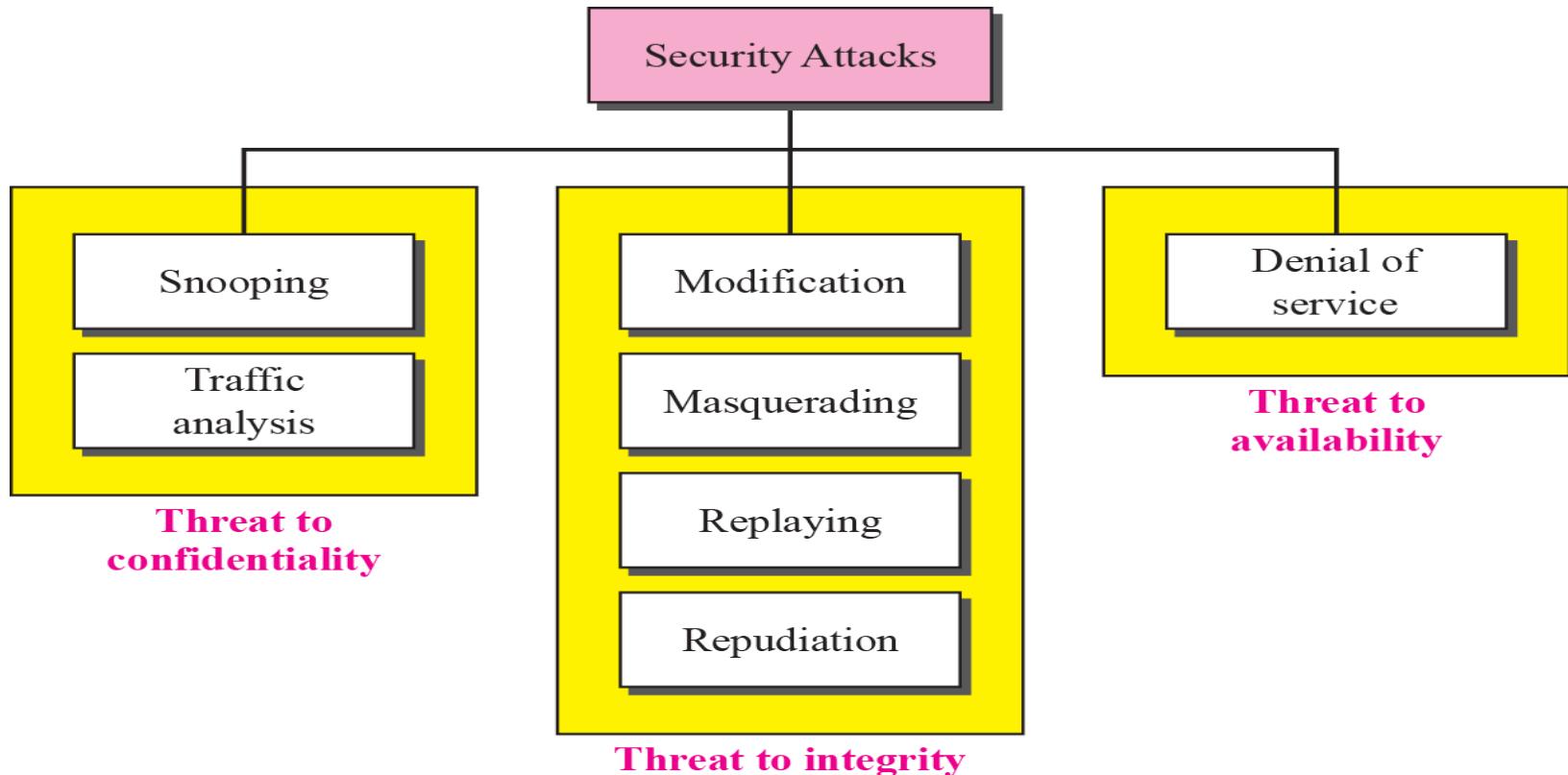
Request to disable an option



Cryptography

- We are living in the information age. We need to keep information about every aspect of our lives.
- In other words, information is an asset that has a value like any other asset.
- As an asset, information needs to be secured from attacks.
- To be secured, information needs to be hidden from
 - unauthorized access (confidentiality)
 - protected from unauthorized change (integrity)
 - available to an authorized entity when it is needed (availability)

Cryptography



Traditional Ciphers

- We now look at the first goal of security, confidentiality
- Confidentiality can be achieved using ciphers
- Traditional ciphers are called symmetric-key ciphers (or secret-key ciphers) because the same key is used for encryption and decryption and the key can be used for bidirectional communication

General idea of traditional cipher

- A substitution cipher replaces one symbol with another



Representation of characters in modulo 26

- In additive cipher, the plaintext, ciphertext, and key are integers in modulo 26

Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext →	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Value →	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Example

- Use the additive cipher with key = 15 to encrypt the message “hello”

Solution

- The cipher is mono-alphabetic because two instances of the same plaintext character (l's) are encrypted as the same character (A). The result is “WTAAD”

Plaintext: h → 07
 Plaintext: e → 04
 Plaintext: l → 11
 Plaintext: l → 11
 Plaintext: o → 14

Encryption: $(07 + 15) \text{ mod } 26$
 Encryption: $(04 + 15) \text{ mod } 26$
 Encryption: $(11 + 15) \text{ mod } 26$
 Encryption: $(11 + 15) \text{ mod } 26$
 Encryption: $(14 + 15) \text{ mod } 26$

Ciphertext: 22 → W
 Ciphertext: 19 → T
 Ciphertext: 00 → A
 Ciphertext: 00 → A
 Ciphertext: 03 → D

Example

- Use the additive cipher with key = 15 to decrypt the message “WTAAD”.

Solution

- We apply the decryption algorithm to the plaintext character by character. The result is “hello”. Note that the operation is in modulo 26, which means that we need to add 26 to a negative result (for example -15 becomes 11).

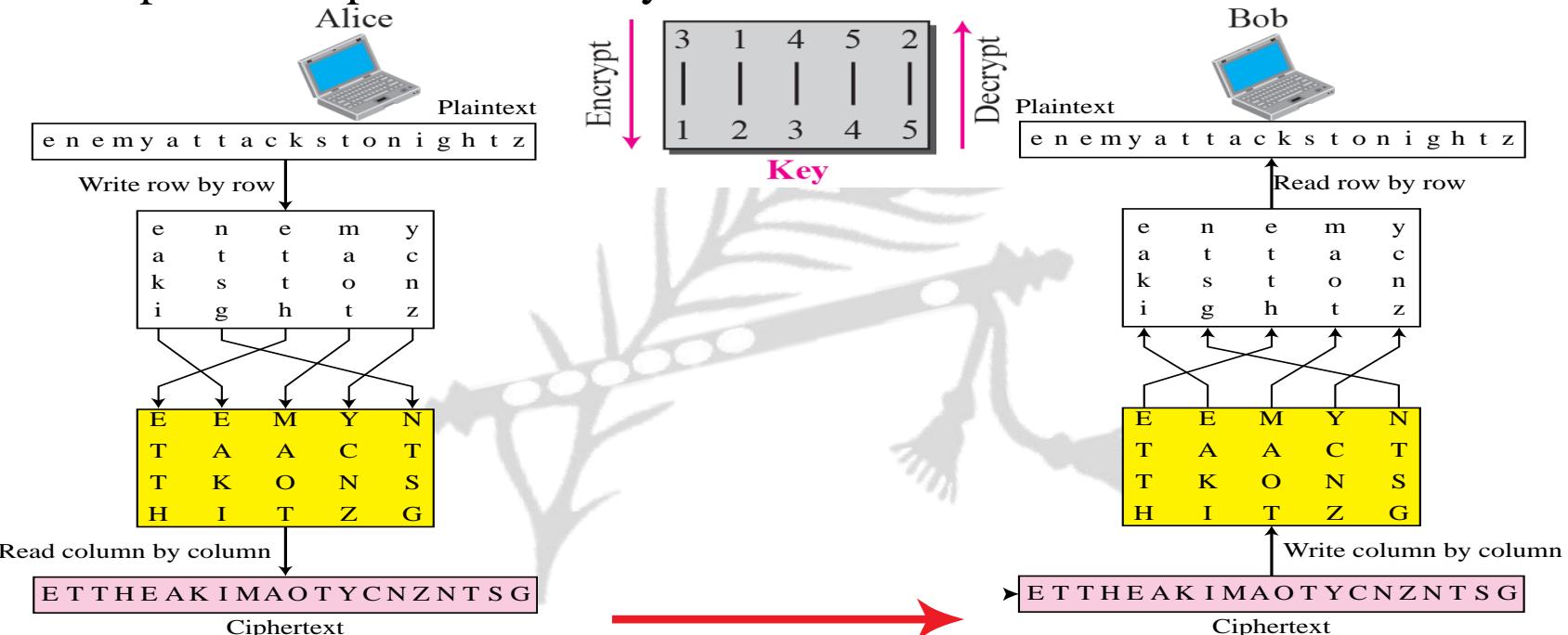
Ciphertext: W → 22
 Ciphertext: T → 19
 Ciphertext: A → 00
 Ciphertext: A → 00
 Ciphertext: D → 03

Decryption: $(22 - 15) \text{ mod } 26$
 Decryption: $(19 - 15) \text{ mod } 26$
 Decryption: $(00 - 15) \text{ mod } 26$
 Decryption: $(00 - 15) \text{ mod } 26$
 Decryption: $(03 - 15) \text{ mod } 26$

Plaintext: 07 → h
 Plaintext: 04 → e
 Plaintext: 11 → l
 Plaintext: 11 → l
 Plaintext: 14 → o

Transposition cipher

- A transposition cipher reorders symbols



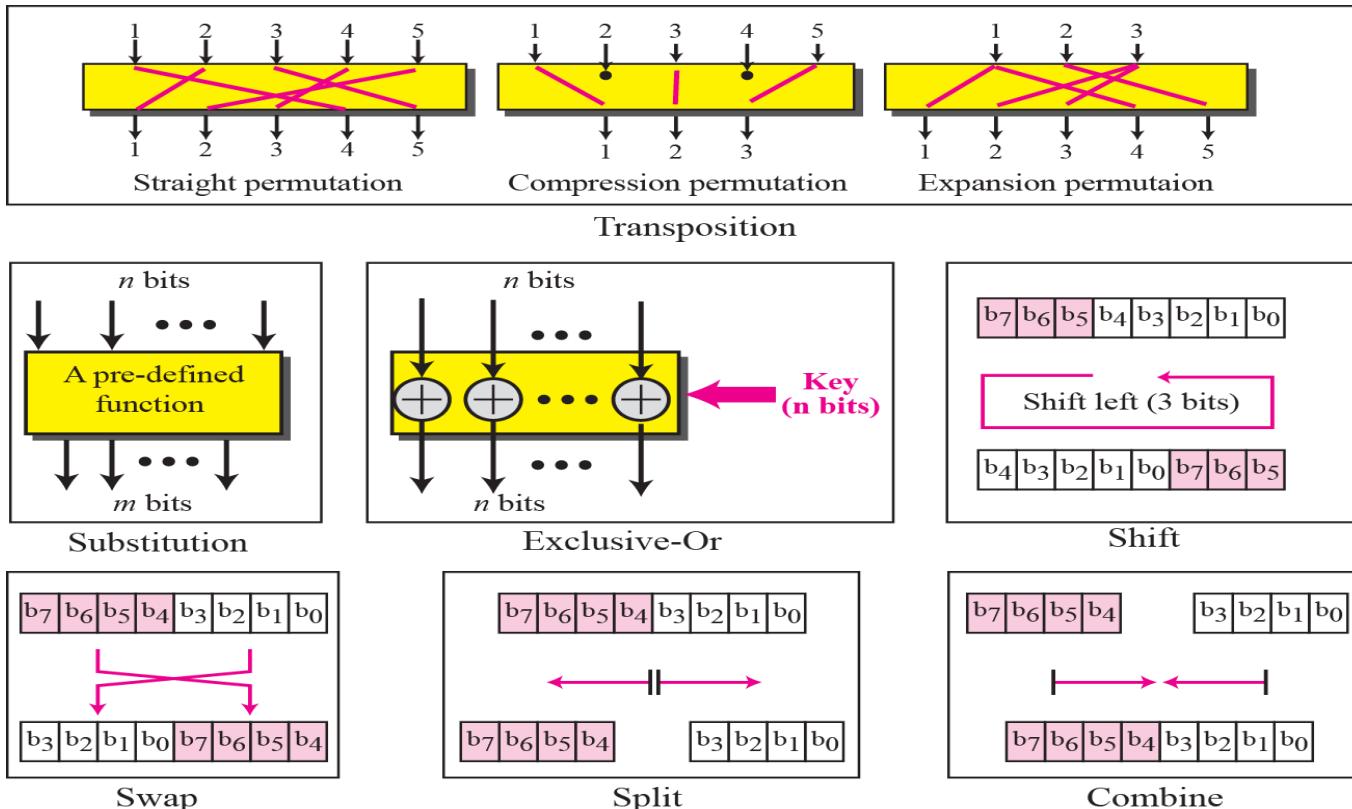
Modern ciphers

- The traditional symmetric-key ciphers that we have studied so far are character-oriented ciphers
- With the advent of the computer, we need bit-oriented ciphers
- This is because the information to be encrypted is not just text; it can also consist of numbers, graphics, audio, and video data
- It is convenient to convert these types of data into a stream of bits, to encrypt the stream, and then to send the encrypted stream
- A modern block cipher can be either a block cipher or a stream cipher

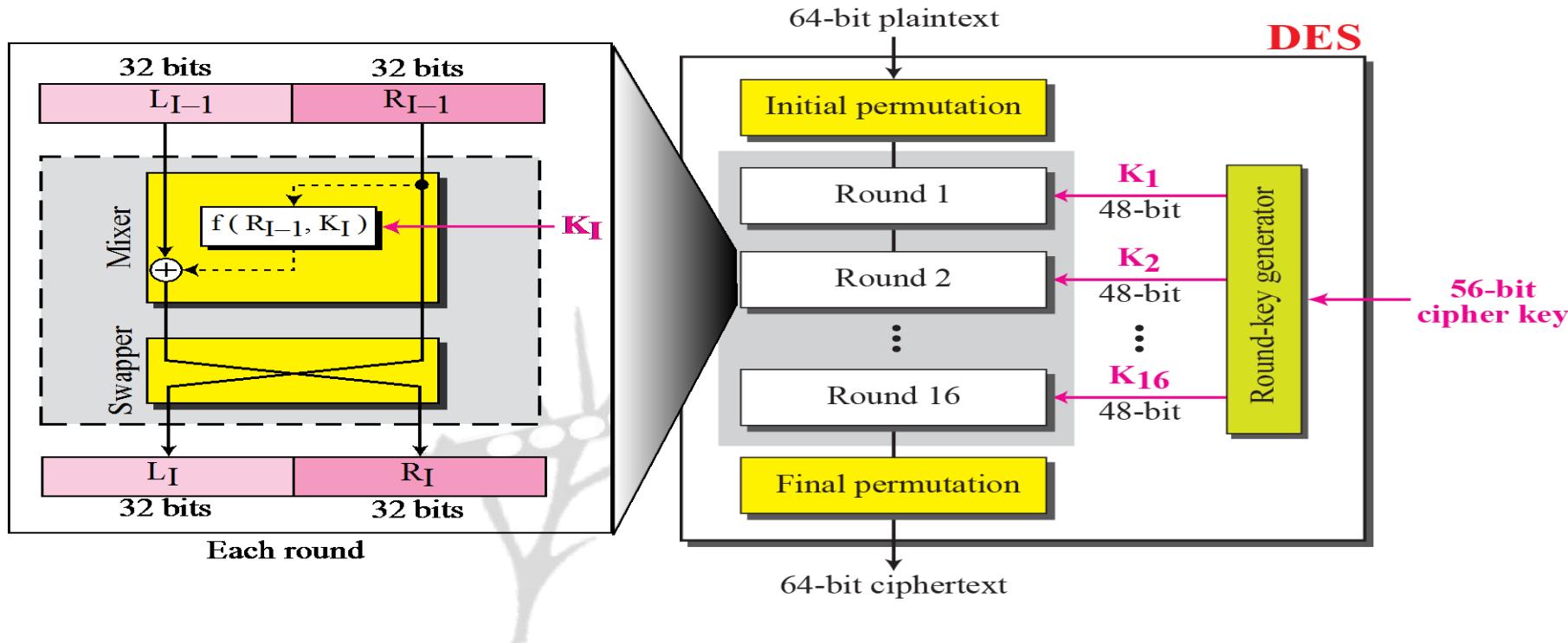
Modern ciphers



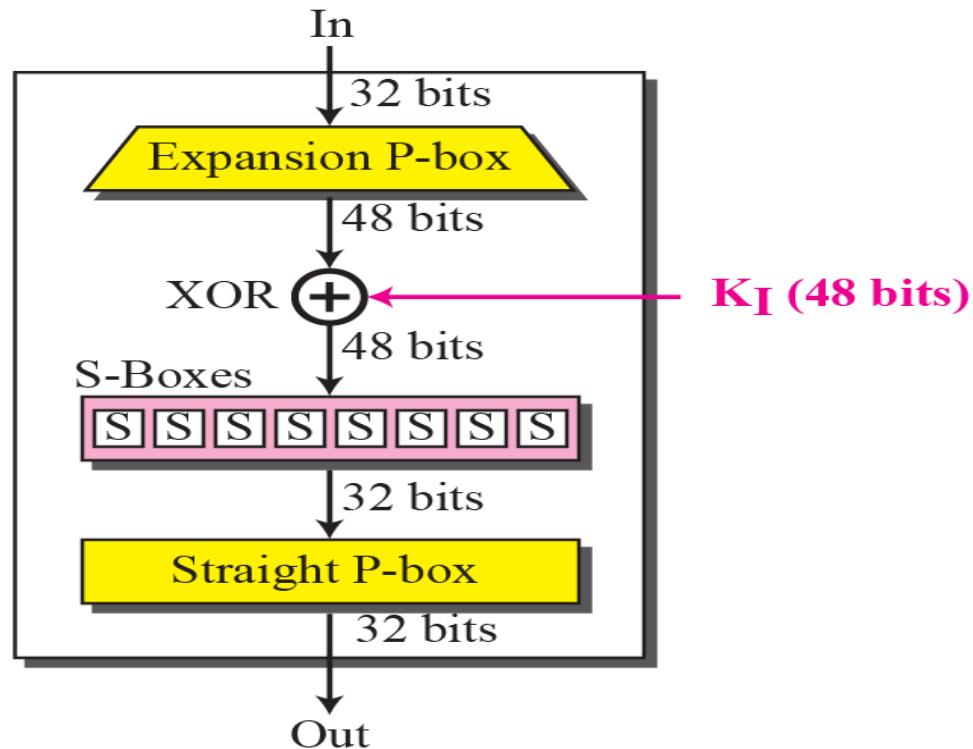
Components of Modern ciphers



General structure of DES



DES function

 $f(R_{I-1}, K_I)$


DES key generation



Example

- We choose a random plaintext block, a random key, and a computer program to determine what the ciphertext block would be (all in hexadecimal):

Plaintext:
123456ABCD132536

Key:
AABB09182736CCDD

CipherText:
C0B7A8D05F3A829C

Example

- To check the effectiveness of DES, when a single bit is changed in the input, let us use two different plaintexts with only one single bit difference. The two ciphertexts are completely different without even changing the key:

Plaintext :
0000000000000000

Key :
22234512987ABB23

Ciphertext :
4789FD476E82A5F1

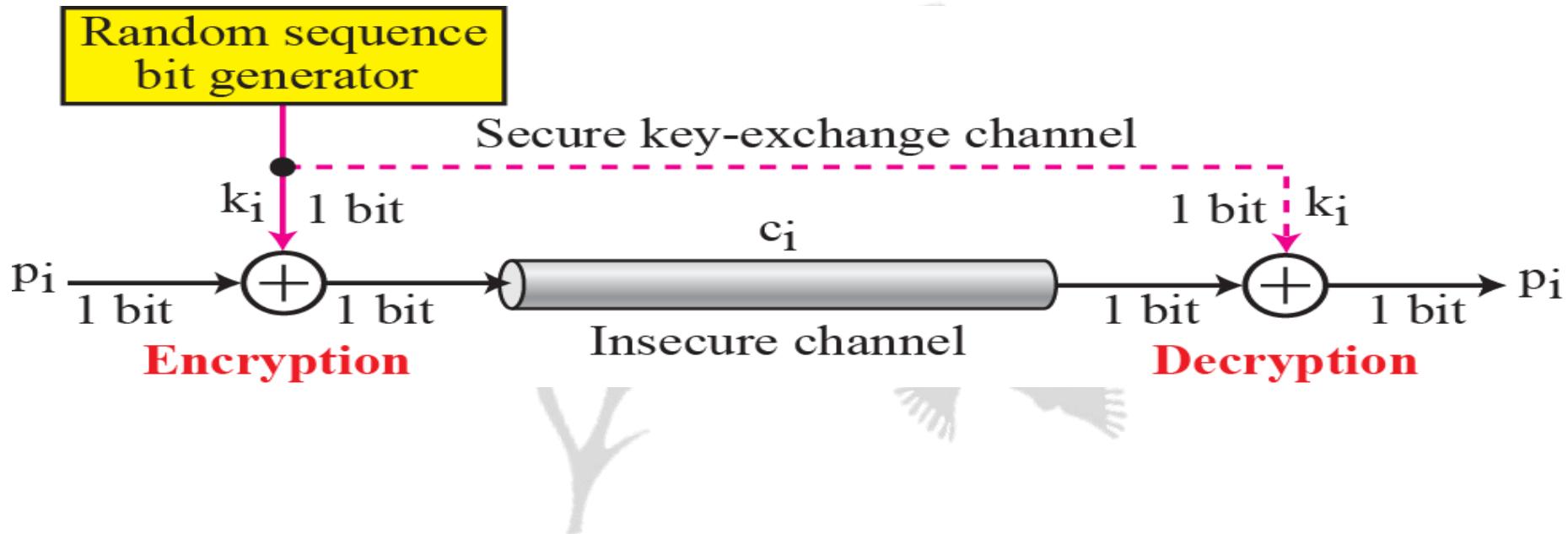
Plaintext :
0000000000000001

Key :
22234512987ABB23

Ciphertext :
0A4ED5C15A63FEA3

- Although the two plaintext blocks differ only in the rightmost bit, the ciphertext blocks differ in 29 bits

One time pad

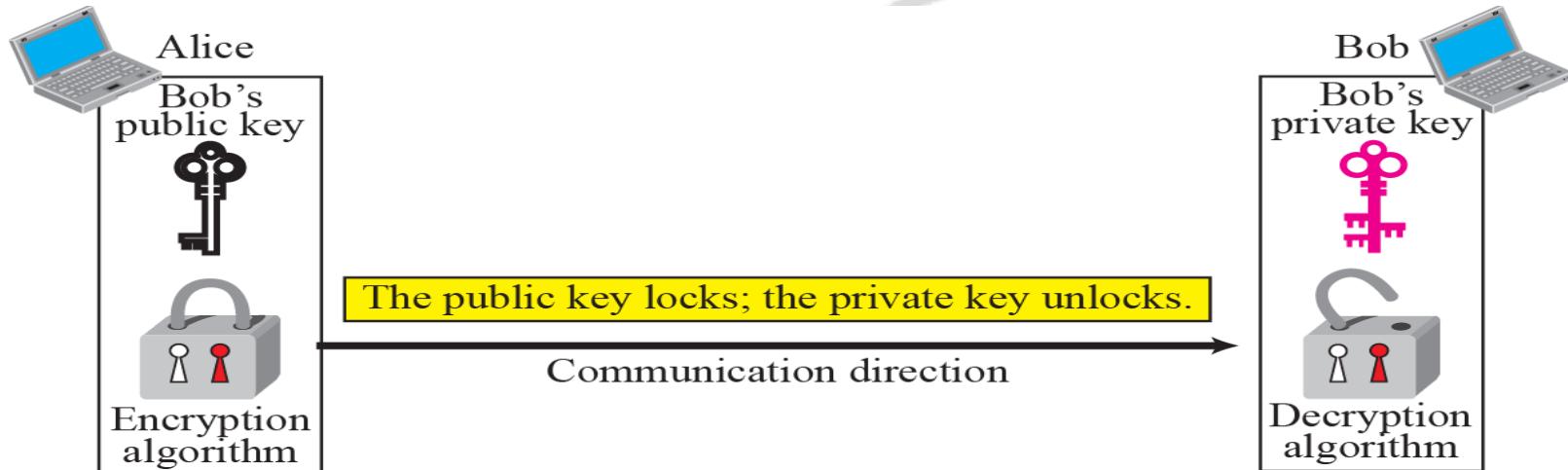


Asymmetric key ciphers

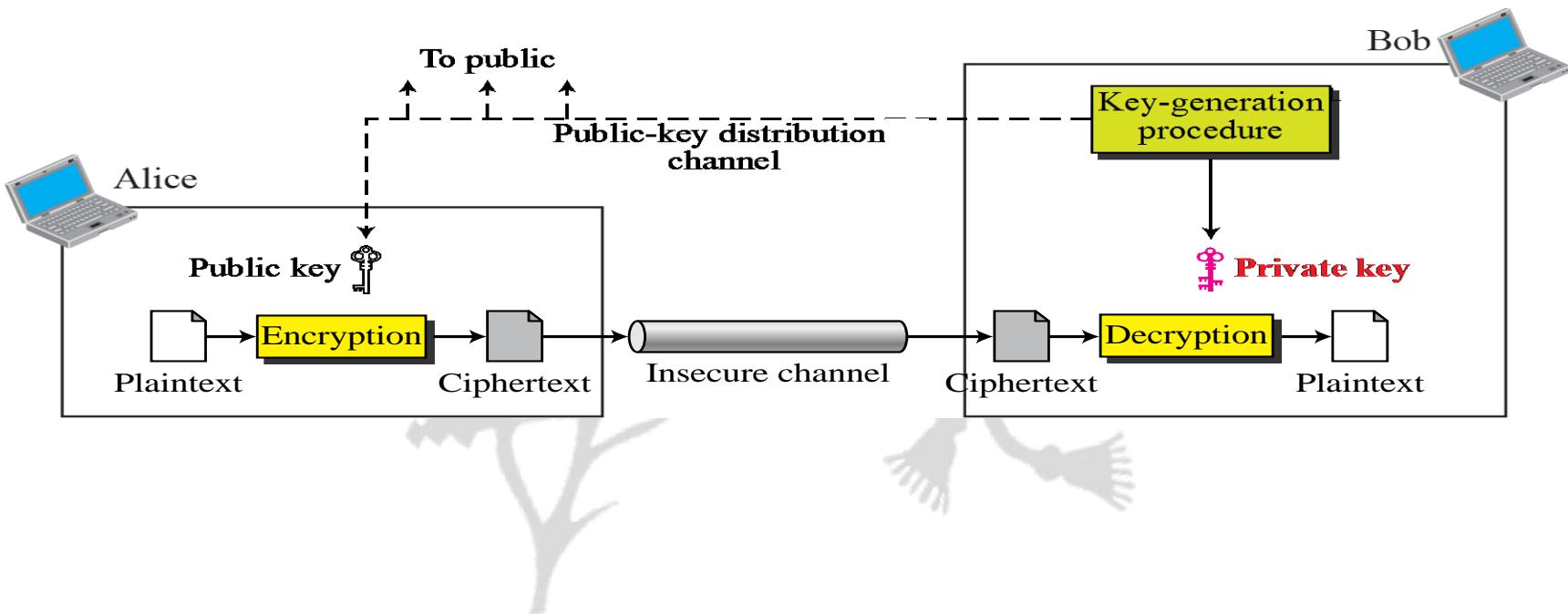
- Symmetric-key and asymmetric-key ciphers will exist in parallel and continue to serve the community
 - We actually believe that they are complements of each other
 - The advantages of one can compensate for the disadvantages of the other
-
- Symmetric-key cryptography is based on sharing secrecy
 - Asymmetric-key cryptography is based on personal secrecy
-
- In symmetric-key cryptography, symbols are permuted or substituted
 - in asymmetric-key cryptography, numbers are manipulated

Asymmetric key ciphers

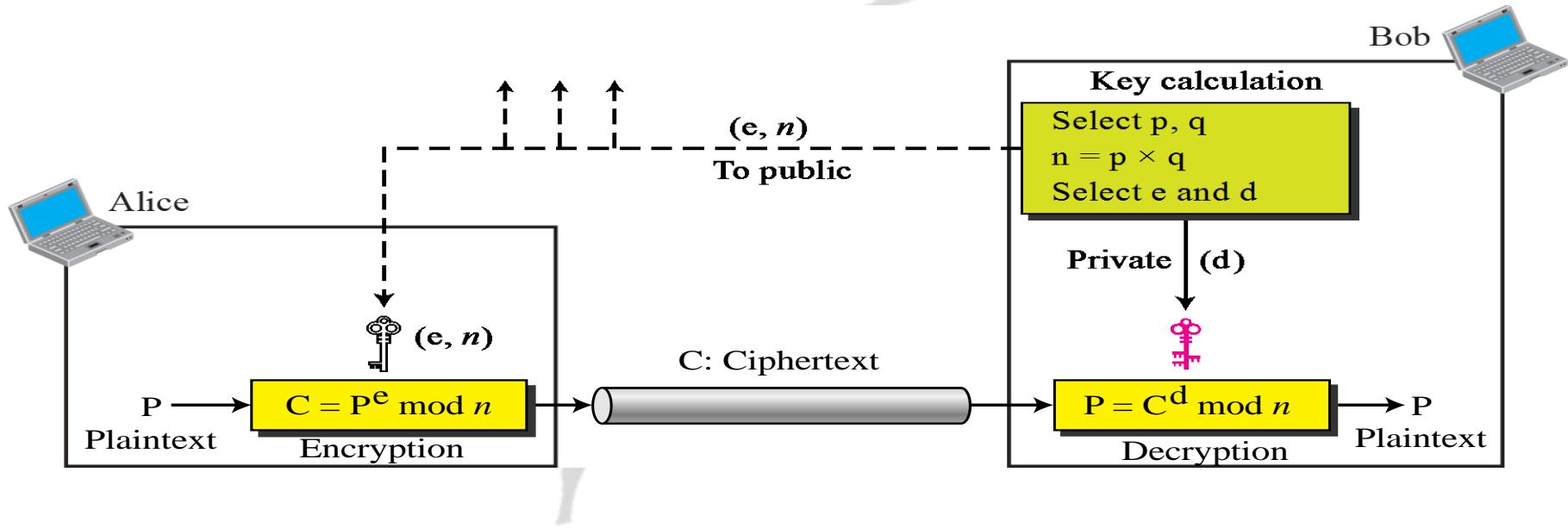
- Asymmetric-key ciphers are sometimes called public-key ciphers



General idea of asymmetric key ciphers



Encryption, decryption and key in RSA



Example

- For the sake of demonstration, let Bob choose 7 and 11 as p and q and calculate $n = 7 \times 11 = 77$
- The value of $\phi(n) = (7 - 1)(11 - 1)$, or 60. If he chooses e to be 13, then d is 37. Note that $e \times d \bmod 60 = 1$
- Now imagine that Alice wants to send the plaintext 5 to Bob. She uses the public exponent 13 to encrypt 5. This system is not safe because p and q are small

Plaintext: 5

$$C = 5^{13} = 26 \bmod 77$$

Ciphertext: 26

Ciphertext: 26

$$P = 26^{37} = 5 \bmod 77$$

Plaintext: 5