RGPVNOTES.IN

Program : **B.Tech**

Subject Name: **Computer Networks**

Subject Code: **CS-602**

Semester: **6th**

Department of Computer Science and Engineering
CS602 Computer Networks
Subject Notes: UNIT-V

Syllabus: **Transport Layer: Design Issues, UDP: Header Format, Per-Segment Checksum, Carrying Unicast/Multicast Real-Time Traffic, TCP: Connection Management, Reliability of Data Transfers, TCP Flow Control, TCP Congestion Control, TCP Header Format, TCP Timer Management. Application Layer: WWW and HTTP, FTP, SSH, Email (SMTP, MIME, IMAP), DNS, Network Management (SNMP).**

Transport Layer: Design Issues

- Accepting data from Session layer split it into segments and send to the network layer.
- Ensure correct delivery of data with efficiency.
- Isolate upper layers from the technological changes.
- Error control and flow control.

UDP: Header Format

The User Datagram Protocol (UDP) is simplest Transport Layer communication protocol available of the TCP/IP protocol suite. It involves minimum amount of communication mechanism. It is said to be an unreliable transport protocol but it uses IP services which provides best effort delivery mechanism.

In this, the receiver does not generate an acknowledgement of packet received and in turn, the sender does not wait for any acknowledgement of packet sent. This shortcoming makes this protocol unreliable as well as easier on processing.

Features

- UDP is used when acknowledgement of data does not hold any significance.
- UDP is good protocol for data flowing in one direction.
- UDP is simple and suitable for query based communications.
- UDP is not connection oriented.
- UDP does not provide congestion control mechanism.
- UDP does not guarantee ordered delivery of data.
- UDP is stateless.
- UDP is suitable protocol for streaming applications such as VoIP, multimedia streaming.

0	15 16	31
16-Bit Source Port	16-Bit Destination Port	
16- Bit UDP Length	16- Bit UDP Checksum	
Data (If Any)		

Fig 5.1 UDP Header Format

The UDP header consists of four fields each of 2 bytes in length:

Source Port (UDP packets from a client use this as a service access point (SAP) to indicate the session on the local client that originated the packet. UDP packets from a server carry the server SAP in this field)

Destination Port (UDP packets from a client use this as a service access point (SAP) to indicate the service required from the remote server. UDP packets from a server carry the client SAP in this field)

UDP length (The number of bytes comprising the combined UDP header information and payload data)

UDP Checksum (A checksum to verify that the end to end data has not been corrupted by routers or bridges in the network or by the processing in an end system. The algorithm to compute the checksum is the Standard Internet Checksum algorithm. This allows the receiver to verify that it was the intended destination of the packet, because it covers the IP addresses, port numbers and protocol number, and it verifies that the packet is not truncated or padded, because it covers the size field. Therefore, this protects an application against receiving corrupted payload data in place of, or in addition to, the data that was sent. In the cases where this check is not required, the value of 0x0000 is placed in this field, in which case the data is not checked by the receiver.

Pre-Segment Checksum

Checksum is a simple error detection mechanism to determine the integrity of the data transmitted over a network. Communication protocols like TCP/IP/UDP implement this scheme in order to determine whether the received data is corrupted along the network. The sender of an IPv4 datagram would compute the checksum value based on the data and embed it in the frame. The receiver would also compute the checksum locally and based on the result ascertain the data integrity. Similarly the TCP/UDP data which forms the payload for the IP datagram would have its checksum computed and embedded as a part of the TCP/UDP frame.

In an attempt to improve performance and to assist drivers in ensuring data integrity, checksum computation is increasingly being done in hardware. The checksum offload feature can be implemented as a combination of hardware and software functions - the hardware assists the driver in completing the checksum computation.

This functionality can be enabled in Asics and disabled in the existing drivers (TCP/IP protocol stack) easily.

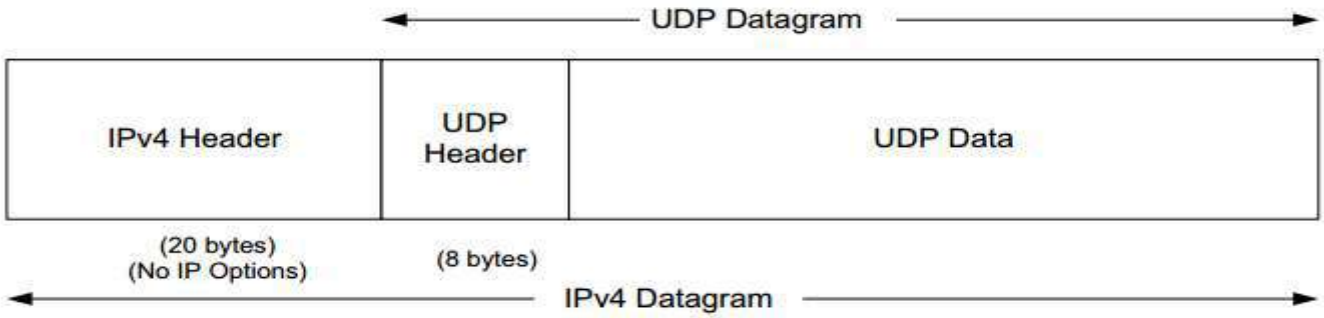


Fig 5.2 UDP Per-Segment Checksum

Unicast/Multicast Real-Time Traffic

Data is transported over a network by three simple methods i.e. Unicast, Broadcast, and Multicast.

- **Unicast**: from one source to one destination i.e. One-to-One
- **Broadcast**: from one source to all possible destinations i.e. One-to-All
- **Multicast**: from one source to multiple destinations stating an interest in receiving the traffic i.e. One-to-Many

Note: There is no separate classification for Many-to-Many applications, for example, video conferencing or online gaming, where multiple sources for the same receiver and where receivers often are double as sources. This service model works on the basis of one-to-many multicast and for that reason requires no unique protocol. The original multicast design i.e. RFC 1112, supports both the ASM (any-source-multicast) based on many-to-many service model and the SSM (source specific multicast) based on a one-to-many model.

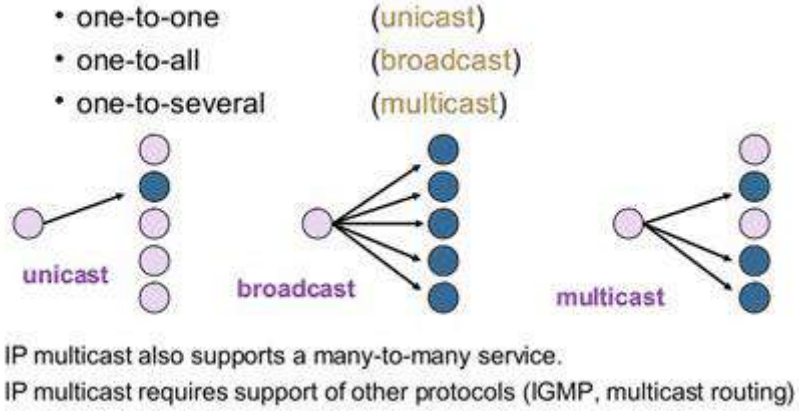


Fig 5.3 IP Services

Unicast

1. Traffic is sent from one host to another. A replica of each packet in the data stream goes to every host that requests it.

2. The implementation of unicast applications is a bit easy as they use well-established IP protocols; however, they are particularly incompetent when there is a need for many-to-many communications. In the meantime, all packets in the data stream must be sent to every host requesting access to the data stream. However, this type of transmission is ineffective in terms of both network and server resource as it equally presents obvious scalability issues.
3. This is a one-to-one connection between the client and the server. Unicast uses IP provision techniques such as TCP (transmission control protocol) and UDP (user datagram protocol), which are session-based protocols. Once a Windows media player client connects via unicast to a Windows media server that client gets a straight connection to the server. Every unicast client that connects to the server takes up extra bandwidth. For instance, if you have 10 clients all performing 100 Kbps (kilobits per second) streams, it means those clients taking up 1,000 Kbps. But you a have single client using the 100 Kbps stream, only 100 Kbps is being used.

Multicast

Multicast lets server's direct single copies of data streams that are then simulated and routed to hosts that request it.

Hence, rather than sending thousands of copies of a streaming event, the server instead streams a single flow that is then directed by routers on the network to the hosts that have specified that they need to get the stream. This removes the requirement to send redundant traffic over the network and also be likely to reduce CPU load on systems, which are not using the multicast system, yielding important enhancement to efficiency for both server and network.

Multicast is true broadcast?

The multicast source depends on multicast-enabled routers to forward the packets to all clients' subnets that have clients listening. However, there is no direct affiliation between clients and Windows media server. The Windows media server creates an ".nsc" (NetShow channel) file when the multicast station is first formed. Usually, the .nsc file is sent to the client from a web server. This file holds data that the Windows media player requires to listen for the multicast. This is quite same to fine-tuning a station on a radio. Every client which eavesdrops to the multicast includes no extra overhead on the server. In fact, the server sends out only single stream per multicast station. The equal load is experienced on the server whether only a single client or multiple clients are listening.

Important note

Multicast on the Internet is usually not a concrete solution because only small sections of the Internet are enabled with Multicast. On the other hand, in corporate environments where all routers are multicast-enabled can save quite a bit of bandwidth.

So what is the difference between Multicast and Unicast?

There are two central methods that Windows Media servers use to send data to Windows Media Player clients i.e. Unicast and Multicast…

Multicast or Unicast both can be used for broadcasting live video or audio.

TCP

TCP is a unicast connection-oriented protocol. Before either end can send data to the other, a connection must be established between them. TCP detects and repairs essentially all the data transfer problems that may be introduced by packet loss, duplication, or errors at the IP layer (or below).

Because of its management of connection state (information about the connection kept by both endpoints), TCP is a considerably more complicated protocol than UDP.

A TCP connection is defined to be a 4-tuple consisting of two IP addresses and two port numbers. It is a pair of endpoints or sockets where each endpoint is identified by an (IP address, port number) pair.

A connection typically goes through three phases:

- Setup
- Data transfer (called established)
- Teardown (closing).

Some of the difficulty in creating a robust TCP implementation is handling all of the transitions between and among these phases correctly.

Connection Management:

A connection typically goes through three phases:

- Setup
- Data transfer (called established)
- Teardown (closing).

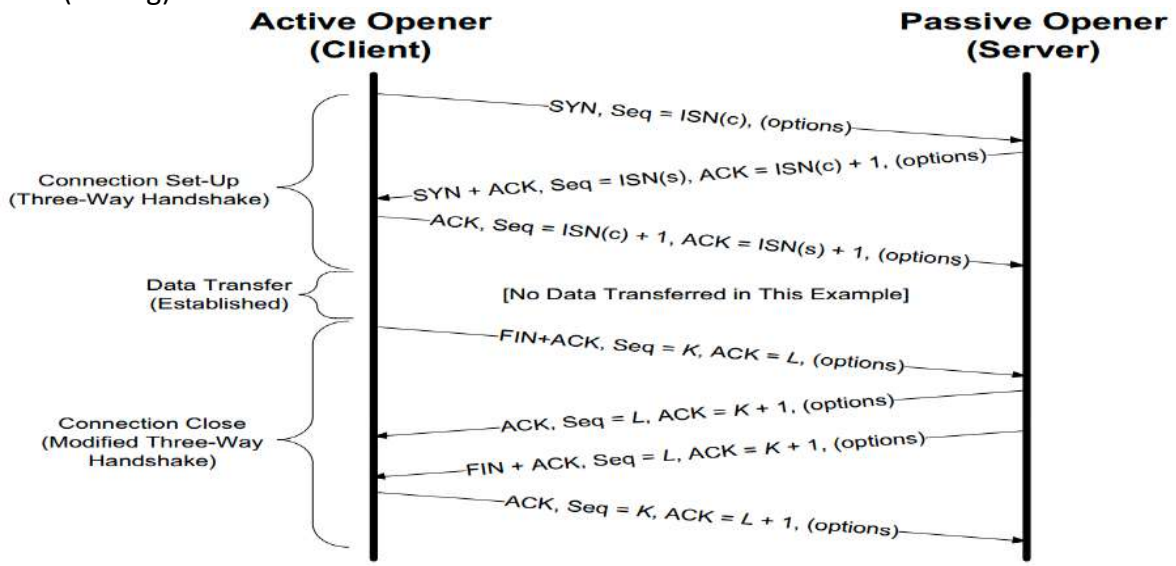


Fig 5.4 TCP- Connection Management

Reliability of Data Transfers

Let us consider the problem of reliable data transfer in a general context. This is appropriate since the problem of implementing reliable data transfer occurs not only at the transport layer, but also at the link layer and the application layer as well. The general problem is thus of central importance to networking. Indeed, if one had to identify a ``top-10'' list of fundamentally important problems in all of networking, this would be a top candidate to lead that list.

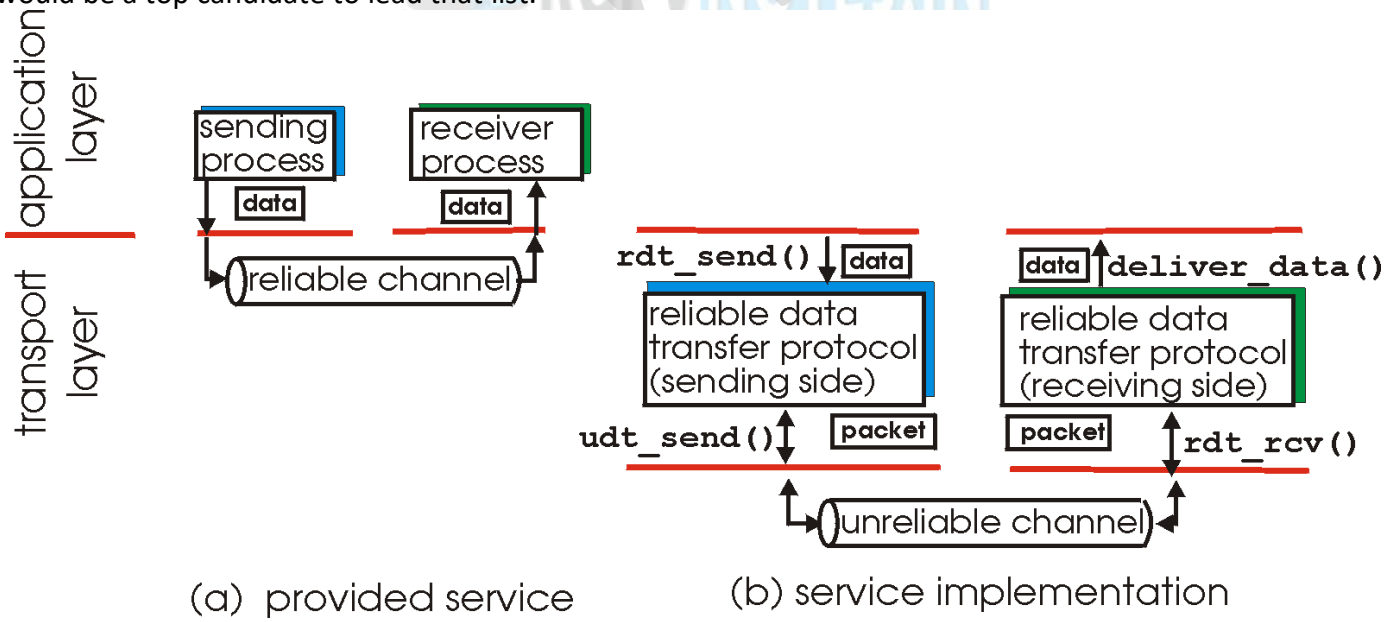


Fig 5.5 Reliable data transfer: service model and service implementation.

It is the responsibility of a **reliable data transfer protocol** to implement this service abstraction. This task is made difficult by the fact that layer below the reliable data transfer protocol may be unreliable. For example, TCP is a reliable data transfer protocol that is implemented on top of an unreliable (IP) end-end network layer. More generally, the layer beneath the two reliably-communicating endpoints might consist of a single physical link (e.g., as in the case of a link-level data transfer protocol) or a global internetwork (e.g., as in the case of a transport-level protocol). For our purposes, however, we can view this lower layer simply as an unreliable point-to-point channel.

The internet network layer provides only best effort service with no guarantee that packets arrive at their destination. Also, since each packet is routed individually it is possible that packets are received out of order. For connection-oriented service provided by TCP, it is necessary to have a reliable data transfer (RDT) protocol to ensure delivery of all packets and to enable the receiver to deliver the packets in order to its application layer.

A simple alternating bit RDT protocol can be designed using some basic tools. This protocol is also known as a stop-and-wait protocol: after sending each packet the sender stops and waits for feedback from the receiver indicating that the packet has been received.

Stop-and-wait RDT protocols have poor performance in a long-distance connection. At best, the sender can only transmit one packet per round-trip time. For a 1000 mile connection this amounts to approximately 1 packet (about 1500 bytes) every 20 ms that results in a pathetic 75 KB per second rate.

To improve transmission rates, a realistic RDT protocol must use pipelining. This allows the sender to have a large number of packets "in the pipeline". This phrase refers to packets that have been sent but whose receipt has not yet verified by the receiver.

TCP Flow Control

TCP is the protocol that guarantees we can have a reliable communication channel over an unreliable network. When we send data from a node to another, packets can be lost, they can arrive out of order, the network can be congested or the receiver node can be overloaded. When we are writing an application, though, we usually don't need to deal with this complexity, we just write some data to a socket and TCP makes sure the packets are delivered correctly to the receiver node. Another important service that TCP provides is what is called *Flow Control*. Let's talk about what that means and how TCP does its magic.

What is Flow Control (and what it's not)

Flow Control basically means that TCP will ensure that a sender is not overwhelming a receiver by sending packets faster than it can consume. It's pretty similar to what's normally called *Back pressure* in the Distributed Systems literature. The idea is that a node receiving data will send some kind of feedback to the node sending the data to let it know about its current condition.

It's important to understand that this is **not** the same as *Congestion Control*. Although there's some overlap between the mechanisms TCP uses to provide both services, they are distinct features. Congestion control is about preventing a node from overwhelming the network (i.e. the links between two nodes), while Flow Control is about the end-node.

How it works

When we need to send data over a network, this is normally what happens.

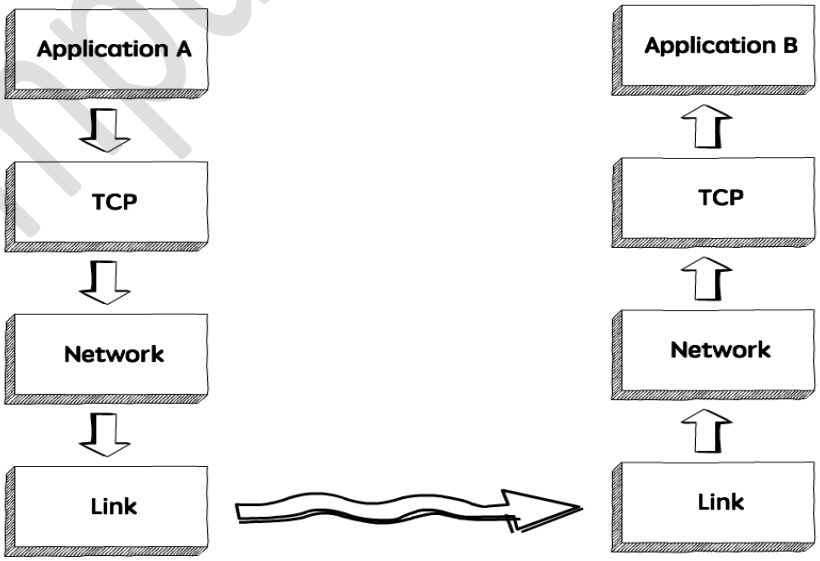


Fig 5.6 TCP Flow Control- works.

The sender application writes data to a socket, the transport layer (in our case, TCP) will wrap this data in a segment and hand it to the network layer (e.g. IP), that will somehow route this packet to the receiving node. On the other side of this communication, the network layer will deliver this piece of data to TCP, that will make it available to the receiver application as an exact copy of the data sent, meaning it will not

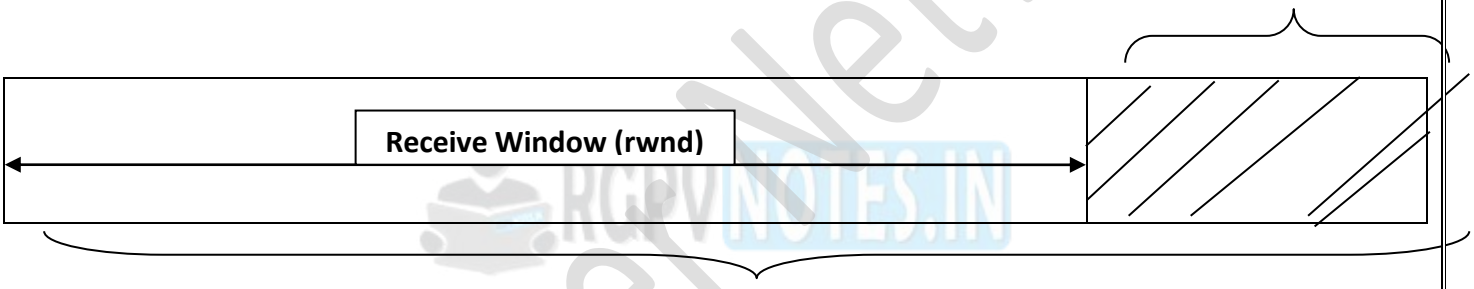
deliver packets out of order, and will wait for a retransmission in case it notices a gap in the byte stream. If we zoom in, we will see something like this.

Fig 5.7 TCP Flow Control.

TCP stores the data it needs to send in the send buffer, and the data it receives in the receive buffer. When the application is ready, it will then read data from the receive buffer.

Flow Control is all about making sure we don't send more packets when the receive buffer is already full, as the receiver wouldn't be able to handle them and would need to drop these packets.

To control the amount of data that TCP can send, the receiver will advertise its Receive Window (rwnd), that is, the spare room in the receive buffer. **Data in the Buffer**



TCP Receive Buffer

Fig 5.8 TCP Flow Control - Receive Window

Every time TCP receives a packet, it needs to send an ack message to the sender, acknowledging it received that packet correctly, and with this ack message it sends the value of the current receive window, so the sender knows if it can keep sending data.

TCP Congestion Control

TCP uses a congestion window and a congestion policy that avoid congestion. Previously, we assumed that only receiver can dictate the sender's window size. We ignored another entity here, the network. If the network cannot deliver the data as fast as it is created by the sender, it must tell the sender to slow down. In other words, in addition to the receiver, the network is a second entity that determines the size of the sender's window.

Congestion policy in TCP –

1. Slow Start Phase: starts slowly increment is exponential to threshold
2. Congestion Avoidance Phase: After reaching the threshold increment is by 1
3. Congestion Detection Phase: Sender goes back to Slow start phase or Congestion avoidance phase.

Slow Start Phase: exponential increment – In this phase after every RTT the congestion window size increments exponentially.

Initially cwnd = 1

After 1 RTT, cwnd = 2^ (1) = 2

2 RTT, cwnd = 2^ (2) = 4

3 RTT, cwnd = 2^ (3) = 8

Congestion Avoidance Phase: additive increment – This phase starts after the threshold value also denoted as ssthresh. The size of cwnd(congestion window) increases additive. After each RTT cwnd = cwnd + 1.

Initially cwnd = i
After 1 RTT, cwnd = i+1
2 RTT, cwnd = i+2
3 RTT, cwnd = i+3

Congestion Detection Phase: multiplicative decrement – If congestion occurs, the congestion window size is decreased. The only way a sender can guess that congestion has occurred is the need to retransmit a segment. Retransmission is needed to recover a missing packet which is assumed to have been dropped by a router due to congestion. Retransmission can occur in one of two cases: when the RTO timer times out or when three duplicate ACKs are received.

- **Case 1: Retransmission due to Timeout –** In this case congestion possibility is high.
  (a) ssthresh is reduced to half of the current window size.
  (b) Set cwnd = 1
  (c) Start with slow start phase again.
- **Case 2: Retransmission due to 3 Acknowledgement duplicates –** In this case congestion possibility is less.
  (a) ssthresh value reduces to half of the current window size.
  (b) Set cwnd= ssthresh.
  (c) Start with congestion avoidance phase.

  Example – Assume a TCP protocol experiencing the behaviour of slow start. At 5th transmission round with a threshold (ssthresh) value of 32 goes into congestion avoidance phase and continues till 10th transmission. At 10th transmission round, 3 duplicate ACKs are received by the receiver and enter into additive increase mode. Timeout occurs at 16th transmission round. Plot the transmission round (time) vs congestion window size of TCP segments.

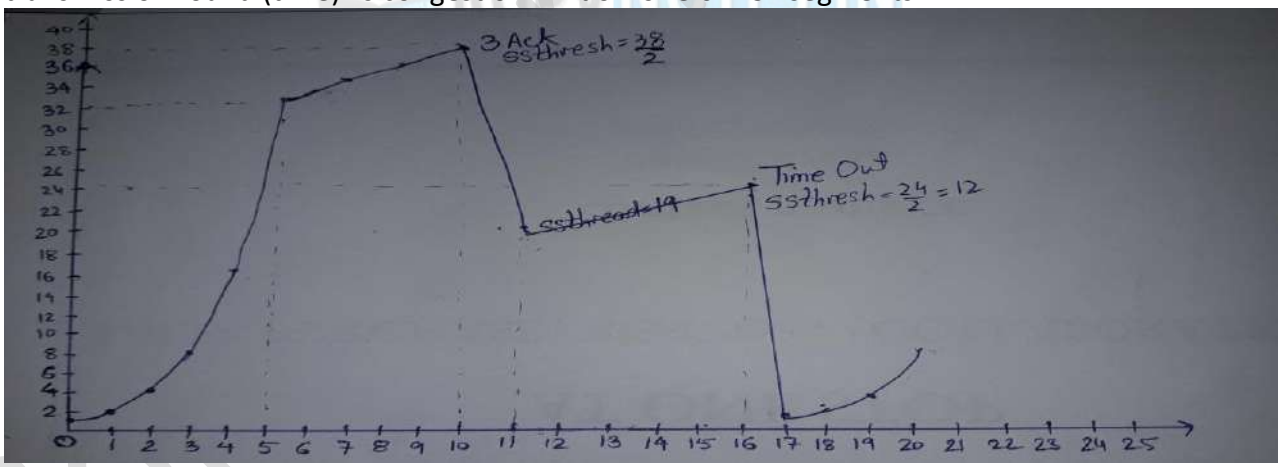


Fig 5.9 Example

TCP Header Format

Each TCP header has ten required fields totalling 20 bytes (160 bits) in size.
They can also optionally include an additional data section up to 40 bytes in size.
This is the layout of TCP headers:

1. Source TCP port number (2 bytes)
2. Destination TCP port number (2 bytes)
3. Sequence number (4 bytes)
4. Acknowledgment number (4 bytes)
5. TCP data offset (4 bytes)
6. Reserved data (3 bits)
7. Control flags (up to 9 bits)
8. Window size (2 bytes)
9. TCP checksum (2 bytes)

10. Urgent pointer (2 bytes)
11. TCP optional data (0-40 bytes)

TCP inserts header fields into the message stream in the order listed above.

- *Source and destination TCP port numbers* are the communication endpoints for sending and receiving devices.
- Message senders use *sequence numbers* to mark the ordering of a group of messages. Both senders and receivers use the acknowledgment *numbers* field to communicate the sequence numbers of messages that are either recently received or expected to be sent.
- The *data offset field* stores the total size of a TCP header in multiples of four bytes. A header not using the optional TCP field has a data offset of 5 (representing 20 bytes), while a header using the maximum-sized optional field has a data offset of 15 (representing 60 bytes).
- *Reserved data* in TCP headers always has a value of zero. This field serves the purpose of aligning the total header size as a multiple of four bytes (important for the efficiency of computer data processing).
- TCP uses a set of six standard and three extended *control flags* (each an individual bit representing *on* or *off*) to manage data flow in specific situations. One bit flag, for example, initiates TCP connection reset logic. The detailed operation of these fields goes beyond the scope of this article.
- TCP senders use a number called *window size* to regulate how much data they send to a receiver before requiring an acknowledgment in return. If the window size becomes too small, network data transfer will be unnecessarily slow, while if the window size becomes too large, the network link can become saturated (unusable for any other applications) or the receiver may not be able to process incoming data quickly enough (also resulting in slow performance). Windowing algorithms built into the protocol dynamically calculate size values and use this field of TCP headers to coordinate changes between senders and receivers.
- The *checksum* value inside a TCP header is generated by the protocol sender as a mathematical technique to help the receiver detect messages that are corrupted or tampered with.
- The urgent pointer field is often set to zero and ignored, but in conjunction with one of the control flags, it can be used as a data offset to mark a subset of a message as requiring priority processing.
- Usages of optional TCP data go beyond the scope of this article but include support for special acknowledgment and window scaling algorithms.

<table>
<tr><td colspan="3">Source Port Number
(2 Bytes)</td><td>Destination Port Number
(2 Bytes)</td></tr>
<tr><td colspan="4">Sequence Number
(4 Bytes)</td></tr>
<tr><td colspan="4">Acknowledgement Number
(4 Bytes)</td></tr>
<tr><td>Data Offset
(4 Bits)</td><td>Reserved
(3 Bits)</td><td>Control Flags
(9 Bits)</td><td>Window Size
(2 Bytes)</td></tr>
<tr><td colspan="3">Checksum
(2 Bytes)</td><td>Urgent Pointer
(2 Bytes)</td></tr>
<tr><td colspan="4">Optional Data</td></tr>
</table>

Fig 5.10 TCP Header Format (20-60 Bytes)

YOUTUBE LINK: https://www.youtube.com/watch?v=wlTtOqeklJI

TCP Timer Management

- TCP uses multiple timers (at least conceptually) to do its work. The most important of these is the RTO (Retransmission Time Out). When a segment is sent, a retransmission timer is started. If the segment is acknowledged before the timer expires, the timer is stopped. If, on the other hand, the timer goes off before the acknowledgement comes in, the segment is retransmitted (and the timer OS started again). It is difficult to predict how long should the timeout be.
- This problem is much more difficult in the transport layer than in data link protocols such as 802.11. In the latter case, the expected delay is measured in microseconds and is highly predictable (i.e.,

has a low variance), so the timer can be set to go off just slightly after the acknowledgement is expected, as shown in Fig below. Since acknowledgements are rarely delayed in the data link layer (due to lack of congestion), the absence of an acknowledgement at the expected time generally means either the frame or the acknowledgement has been lost.

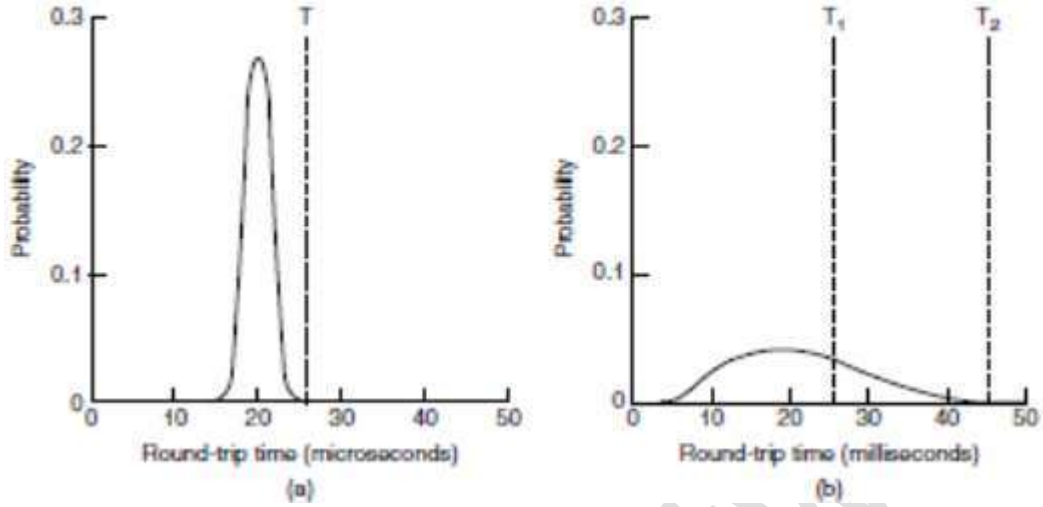


Fig. 5.11 (a) Probability density of acknowledgement arrival times in the data link layer. (b) Probability density of acknowledgement arrival times for TCP.

- TCP is faced with a radically different environment. The probability density function for the time it takes for a TCP acknowledgement to come back looks more like Fig. (b) than Fig. (a) as shown above. It is larger and more variable. Determining the round-trip time to the destination is tricky. Even when it is known, deciding on the timeout interval is also difficult. If the timeout is set too short, say, T1 in Fig. (b), unnecessary retransmissions will occur, clogging the Internet with useless packets. If it is set too long (e.g., T2), performance will suffer due to the long retransmission delay whenever a packet is lost. Furthermore, the mean and variance of the acknowledgement arrival distribution can change rapidly within a few seconds as congestion builds up or is resolved.
- The solution is to use a dynamic algorithm that constantly adapts the timeout interval, based on continuous measurements of network performance. The algorithm generally used by TCP is due to Jacobson (1988) and works as follows. For each connection, TCP maintains a variable, SRTT (Smoothed Round-Trip Time) that is the best current estimate of the round-trip time to the destination in question.
- When a segment is sent, a timer is started, both to see how long the acknowledgement takes and also to trigger a retransmission if it takes too long. If the acknowledgement gets back before the timer expires, TCP measures how long the acknowledgement took, say, R. It then updates SRTT according to the formula.

$SRTT = \alpha SRTT + (1-\alpha) R$

Where α is a smoothing factor that determines how quickly the old values are forgotten. Typically, α = 7/8. This kind of formula is a EWMA (Exponentially Weighted Moving Average) or low-pass filter that discards noise in the samples.

Session layer:

This layer is primarily concerned with coordinating

- Applications as they interact on different hosts.
- Support the dialog between cooperating application programs
- The session layer offers provisions for efficient data transfer.
- The session layer decides when to turn communication on and off between two computer.
- Provides duplex, half-duplex or simplex communications between devices.

Authentication: Before establishing a session with some network peer, it is important for one of the computers to know that another peer it is communicating to is a legitimate one.
Authorization: Authorization is more like "Are you authorized to do so?"
Example:
If someone knows my email address and password, he can easily authenticate himself as 'Amar Shekhar' and he can log in as well. However, since he is not the right person to access my personal email account, so he is not an authorized person to do so.
So the basic difference between the two is: authentication is the process of verifying that "you are who you say you are", authorization is the process of verifying that "you are permitted to do what you are trying to do". Authorization thus presupposes authentication.

Session layer protocol
PAP: Password Authentication Protocol (PAP) provides a simple method for the peer to establish its identity. This is done only upon initial link establishment.
SCP: The Session Control Protocol (SCP) manages logical links for DECnet (DECnet is a suite of network protocols created by Digital Equipment Corporation) connections.
H.245: Call Control Protocol for Multimedia Communication
It is a protocol for the transmission of call management and control signals in packet-based networks using H.323 equipment. The H.245 specification is used in audio, video, and data transmissions, as well as in voice over IP (VoIP). H.245 messages are sent over special channels called H.245 control channels.

Presentation Layer
It responds to service requests from the application layer and issues service requests to the session layer concerned with syntax and semantics of the information exchanged between two systems.

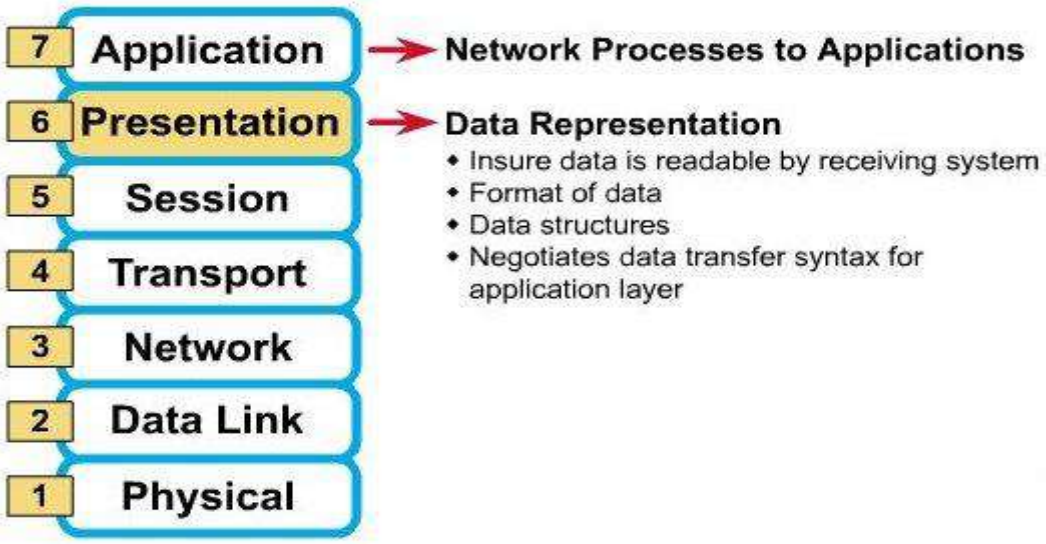


Fig. 5.12 Presentation Layer

Data Conversion- it is responsible for the conversion of computer data from one format to another. Different computers encode data in different ways on the basis of certain standards. On top of that, each computer program handles data in a different manner. Data conversion comes in handy in those situations when the representation of data is needed on different platforms.
Character Code Translation- Before being transmitted, the data remains in the form of characters and numbers. This data has to be changed to bit streams before transmission. The presentation layer is responsible for interoperability between encoding methods as different computers use different encoding methods. It translates data between the formats that a network requires and the format a computer needs.
Data Compression- Data compression is related closely to data representation. One way to transmit a 32-bit integer is to simply encode it as four bytes and send it on its way. However, if it is known that 95% of the numbers sent are between 0 and 250 then it may be better to transmit these integers as a single

unsigned byte, and to use the code 255 to indicate that the following data is a true 32-bit integer. In this case while it is true that 5 bytes will be needed instead of 4, the gain from being able to use one byte most of the time certainly offsets any losses.

Data compression can be approached in three general ways.

1. **The finiteness of the set of symbols: -** A typical book has about 20 characters in its title. Expressed in ASCII such a book title requires 140 bits. By simply giving each book a sequence number, it is possible to reduce the numbers of bits needed from 140 to 26 or fewer. However the receiver must have the numbered book list.
2. **The relative frequencies with which the symbols are used: -** If we are transmitting a DNA code and note that the relative frequencies of the four constituents of DNA are 0.7, 0.2, 0.07, and 0.03, then we can encode the transmission as 0, 10, 110, and 111 respectively. This leads to a tree encoding with 0 for left and 1 for right. (Related to Huffman coding)
3. **The context in which a symbol appears: -** If we are transmitting a DNA code and note that the signalling. Say we wish to transmit the following:
   0001000001000000100000000000000100000010001
   Run lengths of zeros. ie:
   3,5,6,14,6,3
   or
   011 101 110 111 111 110 011

Encryption and Decryption

Encryption is the process of translating plain text data (plaintext) into something that appears to be random and meaningless (ciphertext). Decryption is the process of converting ciphertext back to plaintext.

To encrypt more than a small amount of data, symmetric encryption is used. A symmetric key is used during both the encryption and decryption processes. To decrypt a particular piece of ciphertext, the key that was used to encrypt the data must be used.

The goal of every encryption algorithm is to make it as difficult as possible to decrypt the generated ciphertext without using the key. If a really good encryption algorithm is used, there is no technique significantly better than methodically trying every possible key. For such an algorithm, the longer the key, the more difficult it is to decrypt a piece of ciphertext without possessing the key. It is difficult to determine the quality of an encryption algorithm. Algorithms that look promising sometimes turn out to be very easy to break, given the proper attack. When selecting an encryption algorithm, it is a good idea to choose one that has been in use for several years and has successfully resisted all attacks.

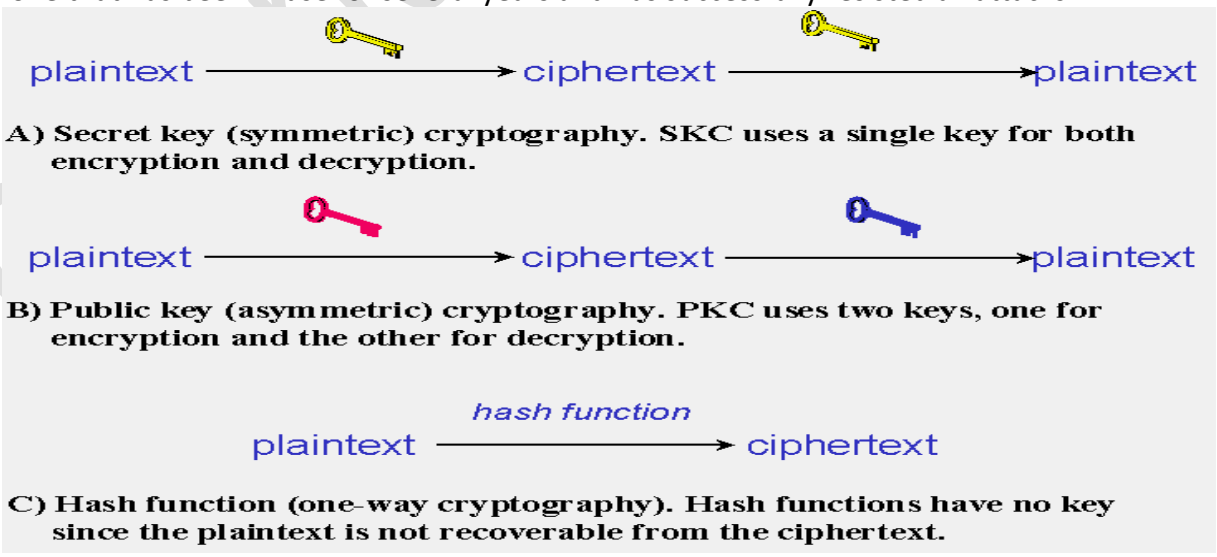


Fig. 5.13 Encryption and Decryption

Presentation layer protocol

LLP

The full form of LPP is Lightweight Presentation Protocol. It is presentation layer protocol Used to provide ISO presentation services on top of TCP/IP based protocol stacks. It is defined in RFC 1085. This protocol refers to an approach used for providing stream-lined support of open source interface (OSI) application services on top of Transmission Control Protocol/Internet Protocol-based (TCP/IP) network for some constrained environments . It is designed for particular class of OSI applications, namely those entities whose application context contains only an ACSE (Association Control Service Element) and ROSE (Remote Operations Service Element). Their is one more thing, Lightweight Presentation Protocol is not applicable to entities whose application context is more extensive. Used to provide ISO presentation services on top of TCP/IP based protocol stacks. It is defined in RFC 1085.

TELNET

A terminal emulation that enables a user to connect to a remote host or device using a telnet client, usually over port 23. For example, typing telnet hostname would connect a user to a host named hostname. Telnet enables a user to manage an account or device remotely. For example, a user may telnet into a computer that hosts their website to manage his or her files remotely.

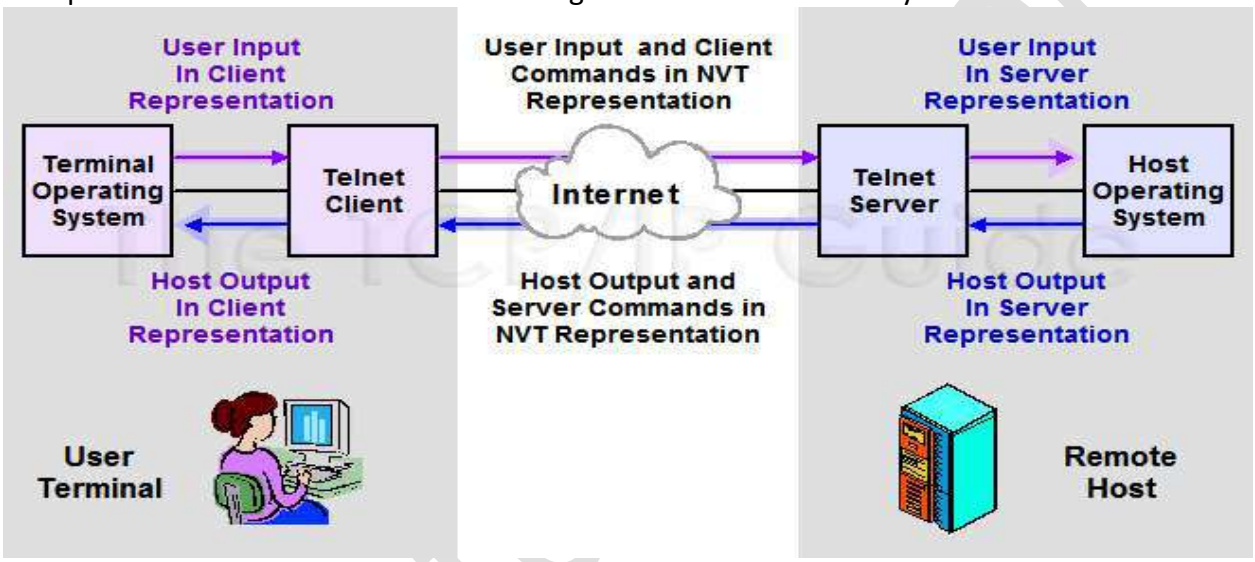


Fig. 5.14 TELNET

X.25 Packet Assembler Disassembler (PAD)

X.25 is a standard suite of protocols used for packet-switched communications over a wide area network—a WAN. A protocol is an agreed-upon set of procedures and rules. Two devices that follow the same protocols can understand each other and exchange data.

X-25 offered three basic layers of protocols:

- Physical layer
- Data link layer
- Packet layer

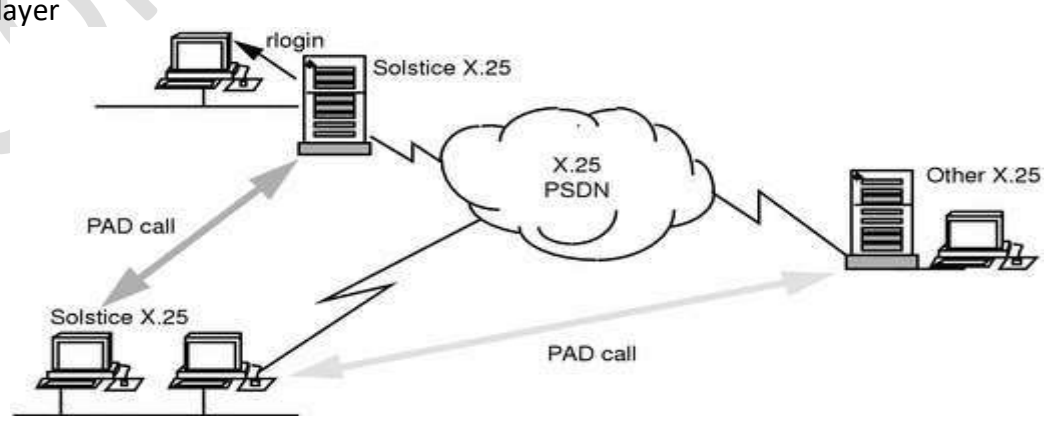


Fig. 5.15 X.25 Packet Assembler Disassembler (PAD)

Application Layer

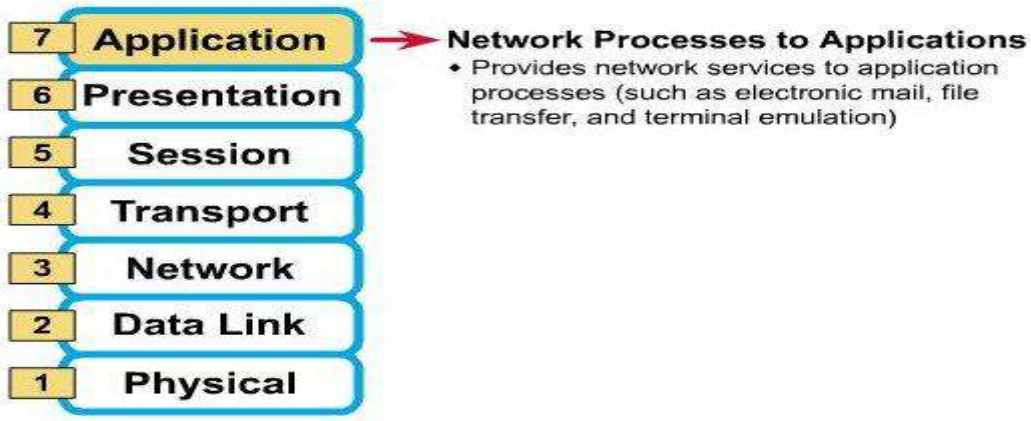


Fig. 5.16 Application Layer

HTTP : Short for HyperText Transfer Protocol, HTTP is a set of standards that allow users of the World Wide Web to exchange information found on web pages. When accessing any web page entering http:// in front of the address tells the browser to communicate over HTTP. For example, the URL for Computer Hope is https://www.computerhope.com. Today's browsers no longer require HTTP in front of the URL since it is the default method of communication. However, it is kept in browsers because of the need to separate protocols such as FTP.

FTP- Stands for "File Transfer Protocol." FTP is a protocol designed for transferring files over the Internet. Files stored on an FTP server can be accessed using an FTP client, such as a web browser, FTP software program, or a command line interface.

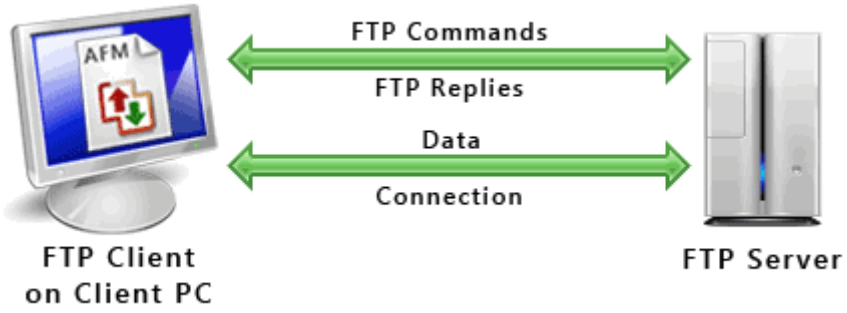


Fig. 5.17 FTP

SSH- The Secure Shell (SSH) protocol performs public-key encryption using a host key and a server key. SSH improves security by providing a means for the storage system to authenticate the client and by generating a session key that encrypts data sent between the client and storage system.

The SSH server version running on Data ONTAP is Data ONTAP SSH version 1.0. For information about the Common Vulnerabilities and Exposures (CVE) fixes implemented in Data ONTAP, see the Suspected Security Vulnerabilities page on the NetApp Support Site.

Data ONTAP supports the SSH 1.x protocol and the SSH 2.0 protocol.

Data ONTAP supports the following SSH clients:

- OpenSSH client version 4.4p1 on UNIX platforms
- SSH Communications Security client (SSH Tectia client) version 6.0.0 on Windows platforms
- Vandyke SecureCRT version 6.0.1 on Windows platforms
- PuTTY version 0.6.0 on Windows platforms
- F-Secure SSH client version 7.0.0 on UNIX platforms

SSH uses three keys to improve security:

- Host key

SSH uses the host key to encrypt and decrypt the session key. You determine the size of the host key, and Data ONTAP generates the host key when you configure SecureAdmin.

- Server key

SSH uses the server key to encrypt and decrypt the session key. You determine the size of the server key when you configure SecureAdmin. If SSH is enabled, Data ONTAP generates the server key when any of the following events occur:

- o You start SecureAdmin
- o An hour elapses
- o The storage system reboots

- Session key

SSH uses the session key to encrypt data sent between the client and storage system. The session key is created by the client. To use the session key, the client encrypts the session key using the host and server keys and sends the encrypted session key to the storage system, where it is decrypted using the host and server keys. After the session key is decrypted, the client and storage system can exchange encrypted data.

EMAIL: E-mail Protocols are set of rules that help the client to properly transmit the information to or from the mail server. Here in this tutorial, we will discuss various protocols such as SMTP, POP, and IMAP.

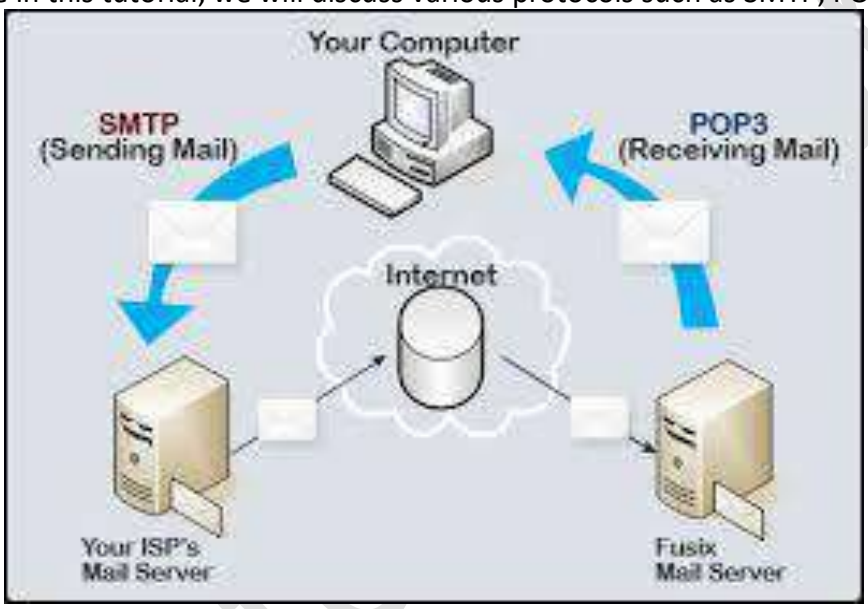


Fig. 5.18 e-mail Protocol

SMTP : SMTP stands for Simple Mail Transfer Protocol. It was first proposed in 1982. It is a standard protocol used for sending e-mail efficiently and reliably over the internet.

- SMTP is application level protocol.
- SMTP is connection oriented protocol.
- SMTP is text based protocol.
- It handles exchange of messages between e-mail servers over TCP/IP network.
- Apart from transferring e-mail, SMPT also provides notification regarding incoming mail.
- When you send e-mail, your e-mail client sends it to your e-mail server which further contacts the recipient mail server using SMTP client.
- These SMTP commands specify the sender's and receiver's e-mail address, along with the message to be send.
- The exchange of commands between servers is carried out without intervention of any user.
- In case, message cannot be delivered, an error report is sent to the sender which makes SMTP a reliable protocol.

IMAP : IMAP stands for Internet Mail Access Protocol. It was first proposed in 1986. There exist five versions of IMAP as follows:

1. Original IMAP
2. IMAP2
3. IMAP3
4. IMAP2bis
5. IMAP4

- IMAP allows the client program to manipulate the e-mail message on the server without downloading them on the local computer.
- The e-mail is hold and maintained by the remote server.
- It enables us to take any action such as downloading, delete the mail without reading the mail.It enables us to create, manipulate and delete remote message folders called mail boxes.
- IMAP enables the users to search the e-mails.
- It allows concurrent access to multiple mailboxes on multiple mail servers.

MIME:

Multipurpose Internet Mail Extension (MIME) is a standard which was proposed by Bell Communications in 1991 in order to expand limited capabilities of email.
MIME is a kind of add on or a supplementary protocol which allows non-ASCII data to be sent through SMTP. It allows the users to exchange different kinds of data files on the Internet: audio, video, images, application programs as well.
Limitations of Simple Mail Transfer Protocol (SMTP):

1. SMTP has a very simple structure
2. It's simplicity however comes with a price as it only send messages in NVT 7-bit ASCII format.
3. It cannot be used for languages that do not support 7-bit ASCII format such as- French, German, Russian, Chinese and Japanese, etc. so it cannot be transmitted using SMTP. So, in order to make SMTP more broad we use MIME.
4. It cannot be used to send binary files or video or audio data.

Features of MIME –

1. It is able to send multiple attachments with a single message.
2. Unlimited message length.
3. Binary attachments (executables, images, audio, or video files) which may be divided if needed.
4. MIME provided support for varying content types and multi-part messages.

MIME with SMTP and POP –SMTP transfers the mail being a message transfer agent from senders side to the mailbox of receiver side and stores it and MIME header is added to the original header and provides additional information. while POP being the message access agent organizes the mails from the mail server to the receivers computer. POP allows user agent to connect with the message transfer agent.

SNMP: Simple Network Management Protocol (SNMP) is an application-layer protocol used to manage and monitor network devices and their functions. SNMP provides a common language for network devices to relay management information within single- and multivendor environments in a local area network (LAN) or wide area network (WAN). The most recent iteration of SNMP, version 3, includes security enhancements that authenticate and encrypt SNMP messages as well as protect packets during transit.
One of the most widely used protocols; SNMP is supported on an extensive range of hardware -- from conventional network equipment like routers, switches and wireless access points to endpoints like

printers, scanners and internet of things (IoT) devices. In addition to hardware, SNMP can be used to monitor services such as Dynamic Host Configuration Protocol (DHCP). Software agents on these devices and services communicate with a network management system (NMS) also referred to as an SNMP manager, via SNMP to relay status information and configuration changes.

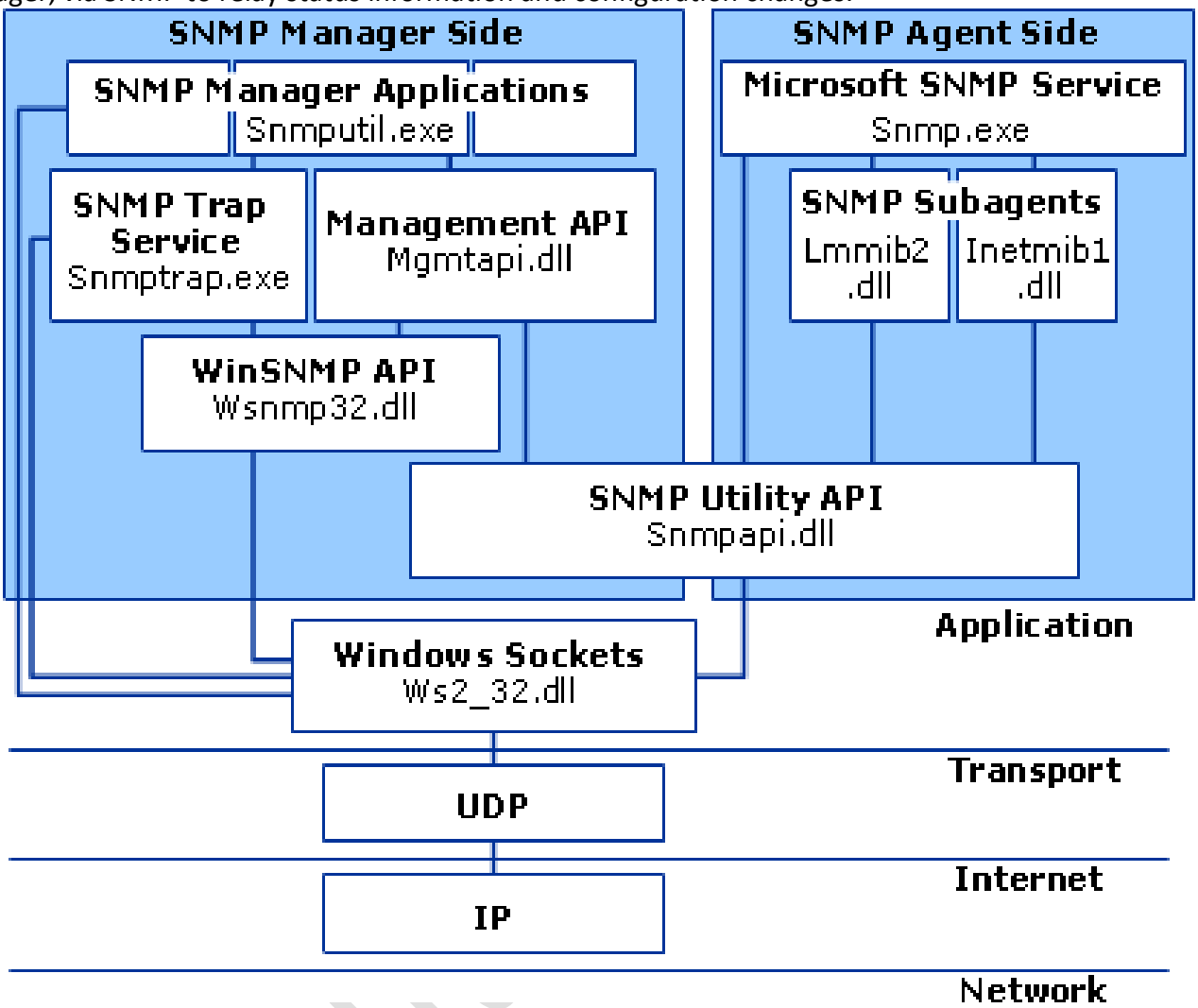


Fig. 5.19 Simple Network Management Protocol

DNS : The domain name system (DNS) is the way that internet domain names are located and translated into internet protocol (IP) addresses. The domain name system maps the name people use to locate a website to the IP addresses that a computer uses to locate a website. For example, if someone types TechTarget.com into a web browser, a server behind the scenes will map that name to the IP address 206.19.49.149.

Web browsing and most other internet activity rely on DNS to quickly provide the information necessary to connect users to remote hosts. DNS mapping is distributed throughout the internet in a hierarchy of authority. Access providers and enterprises, as well as governments, universities and other organizations, typically have their own assigned ranges of IP addresses and an assigned domain name; they also typically run DNS servers to manage the mapping of those names to those addresses. Most URLs are built around the domain name of the web server that takes client requests.

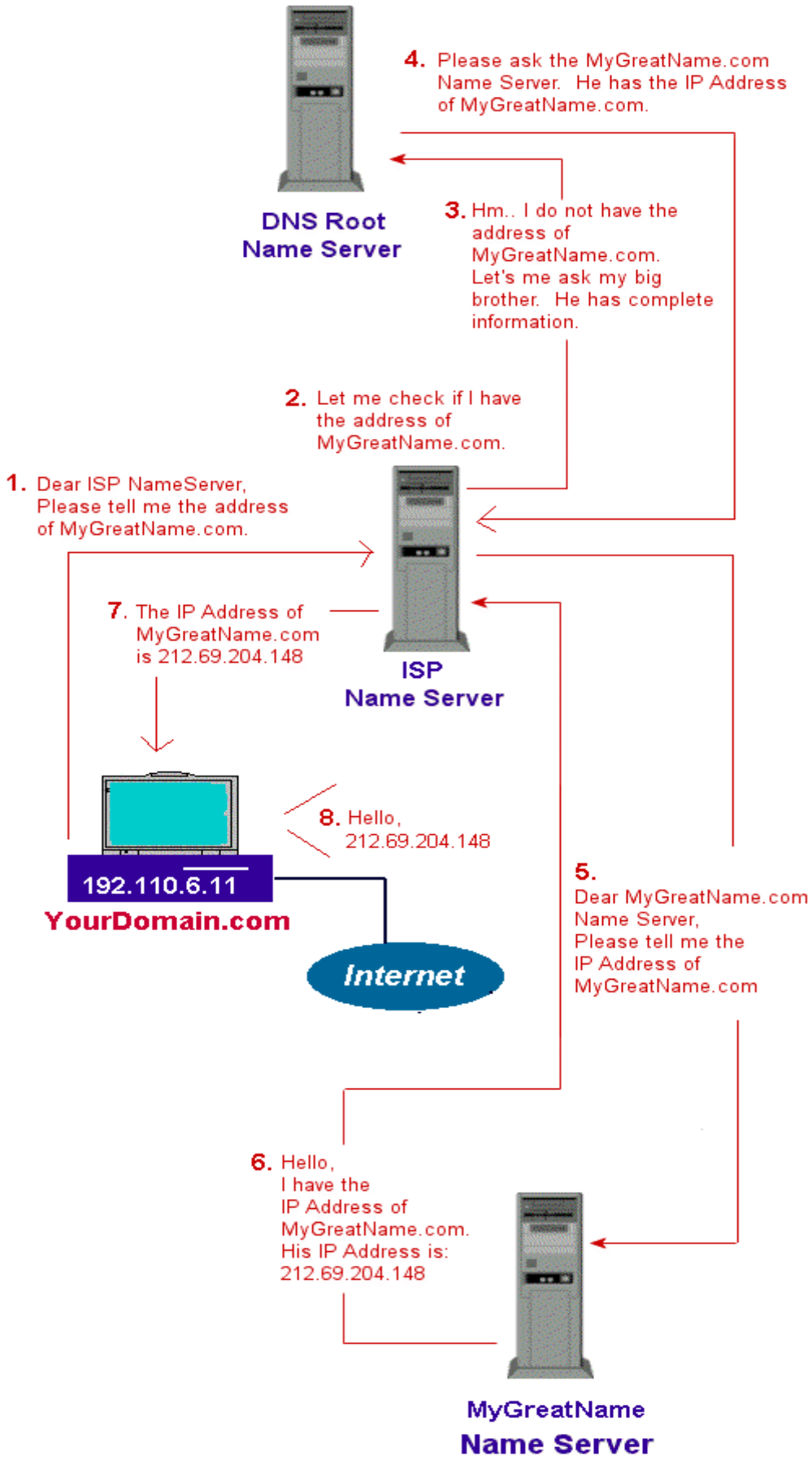


Fig. 5.20 Domain Name Server

RGPVNOTES.IN

We hope you find these notes useful.

You can get previous year question papers at https://qp.rgpvnotes.in .

If you have any queries or you want to submit your study notes please write us at rgpvnotes.in@gmail.com

LIKE & FOLLOW US ON FACEBOOK
facebook.com/rgpvnotes.in