BACHELOR OF COMPUTER APPLICATIONS

SEMESTER 4

DCA2201

COMPUTER NETWORKING

Unit 8

Transport Layer Protocols

Table of Contents

1. INTRODUCTION

In the previous unit, we discussed network layer routing and different routing algorithms. The main functions of the network layer are to accept data from transport layer, convert it into packets and routing packets from source to destination in this unit. We will discuss different transport layer protocols. Transport layer has the critical role of providing communication services directly to the application processes running on different hosts. The main aim of transport layer is to provide data transport from a process on a source machine to a process on a destination machine.

In this unit, we will start with process-to-process delivery in the transport layer. In the next section, we will discuss connectionless transport protocol known as UDP (User Datagram Protocol). Then, we will discuss the connection-oriented transport protocol known as TCP (Transmission Control Protocol). In the last section, we will discuss transport for Real-Time Application (RTP).

1.1 Objectives:

After studying this unit, you should be able to:

- *Describe process-to-process delivery*
- *Explain User Datagram Protocol*
- *Describe Transmission Control Protocol*
- *Explain transport for Real-Time Applications (RTP)*

2. PROCESS-TO-PROCESS DELIVERY

A process (a program in execution is known as a process) is assigned a process identifier number (process ID), which is likely to be different each time the process is started. Process IDs differ between operating system platforms; thus they are not uniform. A server process can have multiple connections with multiple clients at a time, thus simple connection identifiers are not unique. The concept of ports and sockets provides a way to uniformly and uniquely identify connections. It also allows to identify the programs and hosts that are engaged in them, irrespective of specific process IDs.

Ports

Each process that needs to communicate with another process identifies itself to the TCP/IP protocol suite by one or more ports. A port is a logical connection place. A port is always associated with an IP address of a host and the protocol type of the communication. A port is identified for each address and protocol by a 16-bit number, commonly known as the ***port number***. A port is used by the host-to-host protocol to identify to which higher-level protocol or application program (process) it must deliver incoming messages. There are two types of ports: *well-known and ephemeral*

Well-known: Well-known port numbers belong to standard servers, for example, Telnet uses port 23. Well-known port numbers range between 1 and 1023 (prior to 1992, the range between 256 and 1023 was used for UNIX-specific servers). Well-known port numbers are typically odd, because early systems using the port concept required an odd/even pair of ports for duplex operations. Most servers require only a single port. Exceptions are the BOOTP server, which uses two ports: 67 and 68 and the FTP server, which uses two ports, 20 and 21.

The well-known ports are controlled and assigned by the Internet Assigned Number Authority (IANA) and on most systems can only be used by system processes or by programs executed by privileged users. Well-known ports allow clients to find servers without configuration information. The well-known port numbers are defined in STD 2 – Assigned Internet Numbers.

Ephemeral: Some clients do not need well-known port numbers because they initiate communication with servers, and the port number they are using is contained in the UDP/TCP datagrams sent to the server. Each client process is allocated a port number, for as long as it needs, by the host on which it is running. Ephemeral port numbers have values greater than 1023, normally in the range of 1024 to 65535.

Ephemeral ports are not controlled by IANA and can be used by ordinary user-developed programs on most systems. Confusion created due to two different applications trying to use the same port numbers on one host, is avoided by writing those applications to request an available port from TCP/IP. Because this port number is dynamically assigned, it can differ from one invocation of an application to the next.

Sockets

The socket interface is one of several application programming interfaces to the communication protocols. Designed to be a generic communication programming interface, socket APIs were first introduced by 4.2 Berkeley Software Distribution (BSD). Although it has not been standardized, Berkeley socket API has become a de facto industry standard abstraction for network TCP/IP socket implementation.

Consider the following terminologies:

A *socket* is a special type of *file handle*, which is used by a process to request network services from the operating system.

A *socket address* is the triple: *<protocol, local-address, local port>*

For example, in the TCP/IP (version 4) suite: <tcp, 192.168.14.234, 8080>.

A *conversation* is the communication link between two processes. Two processes communicate through TCP sockets. The socket model provides a process with a full-duplex byte stream connection to another process. The application need not concern itself with the management of this stream; these facilities are provided by TCP.

TCP uses the same port principle as UDP to provide multiplexing. Like UDP, TCP uses well-known and ephemeral ports. Each side of a TCP connection has a socket that can be identified by the triple: <TCP, IP address, port number>. If two processes are communicating over TCP,

they have a logical connection that is uniquely identifiable by the two sockets involved, that is, by the combination <TCP, local IP address, local port, remote IP address, remote port>. Server processes are able to manage multiple conversations through a single port.

SELF-ASSESSMENT QUESTIONS – 1

1. An application process is assigned a Process identifier number (Process ID), which is likely to be different each time that the process is started. (True/False)
   (a) True (b) False
2. There are two types of ports, known as _____and _____.
3. Well-known port numbers range between _____and _____.
4. The well-known ports are controlled and assigned by _____.
5. Port numbers have values in the range of 1024 to 65535, which is known as _____ .
6. A _____is a special type of file handle, which is used by a process to request network services from the operating system.
7. A *socket address* is the triple represented as _____.

3. CONNECTIONLESS TRANSPORT: UDP

UDP (User Datagram Protocol) is a connectionless transport protocol. UDP provides a way for applications to send encapsulated IP datagrams without having to establish a connection. UDP is described by RFC 768.

UDP provides a mechanism for one application to send a datagram to another. The UDP layer can be regarded as being extremely thin and is, consequently, very efficient, but it requires the application to take responsibility for error recovery and so on.

Applications sending datagrams to a host need to identify a target that is more specific than the IP address, because datagrams are normally directed to certain processes and not to the system as a whole. UDP provides this by using ports.

UDP datagram format

Each UDP datagram is sent within a single IP datagram. Although the IP datagram might be fragmented during transmission, the receiving IP implementation will reassemble it before presenting it to the UDP layer. The UDP datagram has an 8-byte header followed by the payload as shown in figure 8.1:

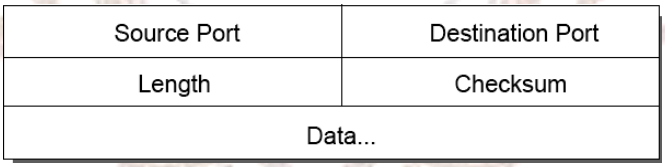

Fig 8.1: UDP: Datagram format

Source Port indicates the port of the sending process. It is the port to which replies are addressed. Source port is primarily needed when a reply must be sent back to the source. The *UDP length* field includes the 8-byte header and the data. The minimum length is 8 bytes, to cover the header. The maximum length is 65,515 bytes, which is lower than the largest number that will fit in 16 bits because of the size limit on IP packets. *Destination Port* specifies the port of the destination process on the destination host.

An optional *checksum* is also provided for extra reliability. Its checksums the header, the data, and a conceptual IP pseudo header. When performing this computation, the *Checksum* field is set to zero and the data field is padded out with an additional zero byte if its length is an odd number. The checksum algorithm is simply to add up all the 16-bit words in one's

complement and to take the one's complement of the sum. As a consequence, when the receiver performs the calculation on the entire segment, including the *Checksum* field, the result should be 0.

In Figure 8.2, we see a pseudo-IP header. It contains the source and destination IP addresses, the protocol (protocol number for UDP, is 7), and the UDP length (byte count for the UDP segment).

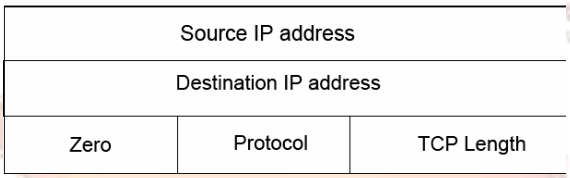

Fig 8.2: UDP: Pseudo-IP header

The pseudo-IP header in the UDP checksum computation helps detect mis-delivered packets.

UDP application programming interface

The application interface offered by UDP is described in RFC 768. It does not do flow control, congestion control, or retransmission upon receipt of a bad segment. What it does do is provide an interface to the IP protocol with the added feature of demultiplexing multiple processes using the ports and optional end-to-end error detection. One area where it is especially useful is in client-server situations. Often, the client sends a short request to the server and expects a short reply back. If either the request or the reply is lost, the client can just time out and try again.

Standard applications using UDP include:

- Trivial File Transfer Protocol (TFTP)
- Domain Name System (DNS) name server
- Remote Procedure Call (RPC), used by the Network File System
- Simple Network Management Protocol (SNMP)
- Lightweight Directory Access Protocol (LDAP)

4. CONNECTION-ORIENTED TRANSPORT: TCP

TCP (Transmission Control Protocol) was designed to provide a reliable end-to-end byte stream over an unreliable internetwork. TCP is described by RFC 793. TCP provides considerably more facilities for applications than UDP. Specifically, this includes error recovery, flow control, and reliability. TCP is a *connection-oriented protocol,* unlike UDP, which is *connectionless.* Most of the user application protocols, such as Telnet and FTP, use TCP.

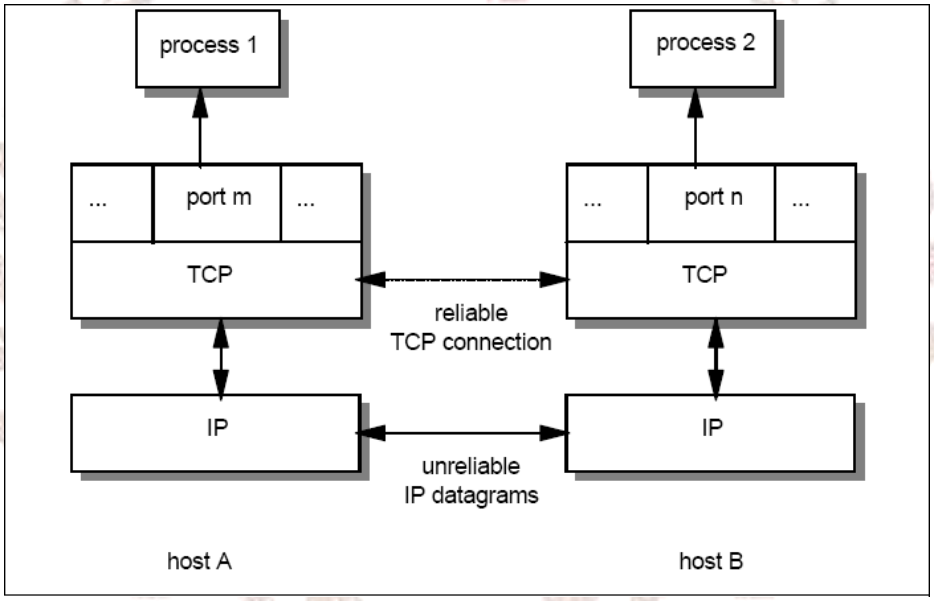

Fig 8.3: TCP: Connection between processes

The two processes communicate with each other over a TCP connection (Inter-process Communication, or IPC), as shown in Figure 8.3. In figure 8.3, processes 1 and 2 communicate over a TCP connection carried by IP datagrams.

As noted earlier, the primary purpose of TCP is to provide a reliable logical circuit or connection service between pairs of processes. TCP can be characterized by the following facilities it provides for the applications using it:

- *Stream data transfer:* From the application's viewpoint, TCP transfers a continuous stream of bytes through the network. The application does not have to bother with breaking the data into basic blocks or datagrams. TCP does this by grouping the bytes into TCP segments, which are passed to the IP layer for transmission to the destination.

Also, TCP itself decides how to segment the data, and it can forward the data at its own convenience.

- *Reliability:* TCP assigns a sequence number to each byte transmitted, and expects a positive acknowledgment (ACK) from the receiving TCP layer. If the ACK is not received within a timeout interval, then the data is retransmitted. The data is transmitted in blocks (TCP segments), only the sequence number of the first data byte in the segment is sent to the destination host. The receiving TCP uses the sequence numbers to rearrange the segments when they arrive out of order, and to eliminate duplicate segments.
- *Flow control:* The receiving TCP, when sending an ACK back to the sender, also indicates to the sender the number of bytes it can receive without causing overflow in its internal buffers. This is sent in the ACK in the form of the highest sequence number it can receive without problems. This mechanism is also referred to as a window-mechanism.
- *Multiplexing:* Achieved through the use of ports.
- *Logical connections:* The reliability and flow control mechanisms described here require that TCP initializes and maintains certain status information for each data stream. The combination of this status, including sockets, sequence numbers, and window sizes, is called a logical connection. Each connection is uniquely identified by the pair of sockets used by the sending and receiving processes.
- *Full duplex:* TCP provides transmission for concurrent data streams in both directions.

A simple transport protocol might use the following principle: send a packet and then wait for an acknowledgment from the receiver before sending the next packet. If the ACK is not received within a certain amount of time, retransmit the packet. This principle is known as the window principle. See Figure 8.4 for more details. Although this mechanism ensures reliability, it only uses a part of the available network bandwidth.

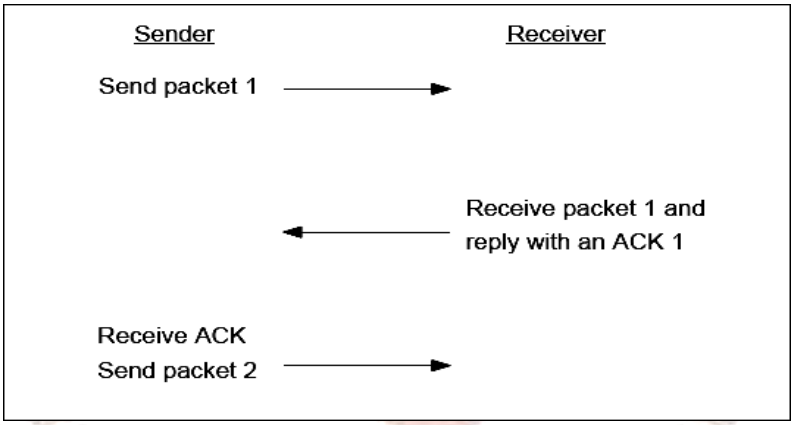

Fig 8.4: TCP: The window principle

Now, consider a protocol where the sender groups its packets to be transmitted, as in Figure 8.5, and uses the following rules:

- The sender can send all packets within the window without receiving an ACK, but must start a timeout timer for each of them.
- The receiver must acknowledge each packet received, indicating the sequence number of the last well-received packet.
- The sender slides the window on each ACK received.

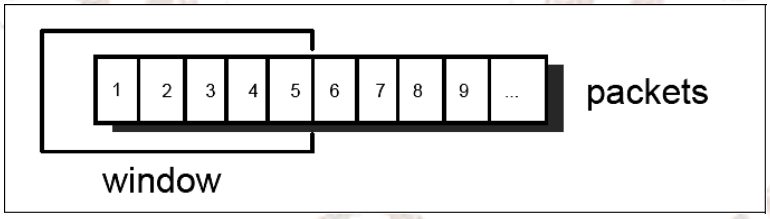

Fig 8.5: TCP Message packets

As shown in Figure 8.6, the sender can transmit packets 1 to 5 without waiting for any acknowledgment.

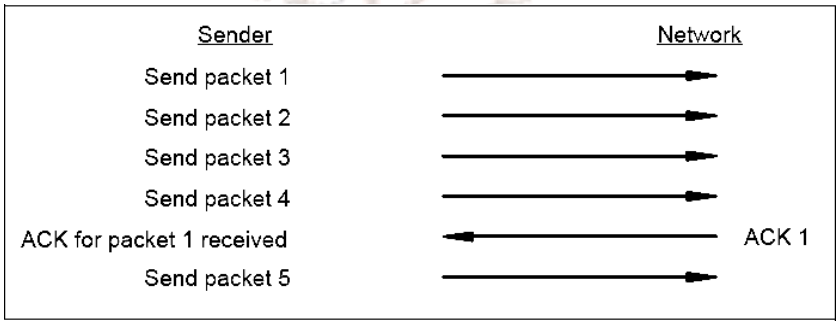

Fig 8.6: TCP: Packet transfer

At the moment the sender receives ACK 1 (acknowledgment for packet 1), it can slide its window one packet to the right. At this point, the sender can also transmit packet 6.

This window mechanism ensures reliable transmission, better use of the network bandwidth (better throughput), flow-control, because the receiver can delay replying to a packet with an acknowledgment, knowing its free buffers are available and the window size of the communication.

In TCP, the window principle is used at the byte level, that is, the segments sent and ACKs received will carry byte-sequence numbers and the window size is expressed as a number of bytes, rather than a number of packets. The window size is determined by the receiver when the connection is established and is variable during the data transfer. Each ACK message will include the window size that the receiver is ready to deal with at that particular time.

TCP segment format

Figure 8.7 shows the layout of a TCP segment. Every segment begins with a fixed-format, 20-byte header. The fixed header may be followed by header options. Segments without any data are legal and are commonly used for acknowledgements and control messages.

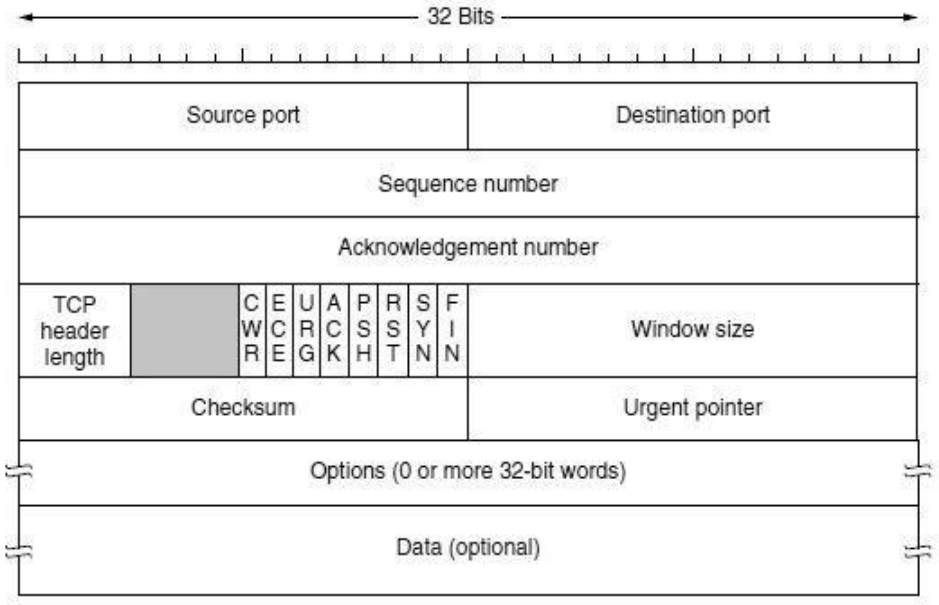

Fig 8.7: TCP Header

Table 8.1 gives a description of the various fields in a TCP segment.

Table 8.1: Description of various fields of TCP Header

Field Name	Description
Source Port	The 16-bit source port number, used by the receiver to reply.
Destination Port	The 16-bit destination port number.
Sequence Number	The sequence number of the first data byte in this segment. If the SYN control bit is set, the sequence number is the initial sequence number (n) and the first data byte is n+1.
Acknowledgment Number	If the ACK control bit is set, this field contains the value of the next sequence number that the receiver is expecting to receive.
Data Offset	The number of 32-bit words in the TCP header. It indicates where the data begins
Reserved	Six bits reserved for future use; must be zero.
URG	Indicates that the urgent pointer field is significant in this segment.
ACK	Indicates that the acknowledgment field is significant in this segment.
PSH	Push function
RST	Resets the connection
SYN	Synchronizes the sequence numbers
FIN	No more data from sender
Window	Used in ACK segments. It specifies the number of data bytes, beginning with the one indicated in the acknowledgment number field that the receiver (the sender of this segment) is willing to accept.
Checksum	The 16-bit one's complement of the one's complement sum of all 16-bit words in a pseudo-header, the TCP header, and the TCP data. While computing the checksum, the checksum field itself is considered zero.

The pseudo-header is the same as that used by UDP for calculating the checksum. It is a pseudo-IP-header, only used for the checksum calculation, with the format shown in figure 8.8.

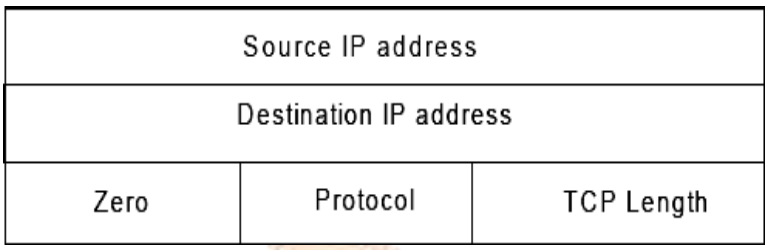

Fig 8.8: TCP Pseudo-IP header

TCP connection establishment and connection release

Before any data can be transferred, a connection has to be established between the two processes. One of the processes (usually the server) issues a passive *OPEN* call, the other an active *OPEN* call. The passive OPEN call remains dormant until another process tries to connect to it by an active OPEN.

As shown in Figure 8.9, in the network, three TCP segments are exchanged.

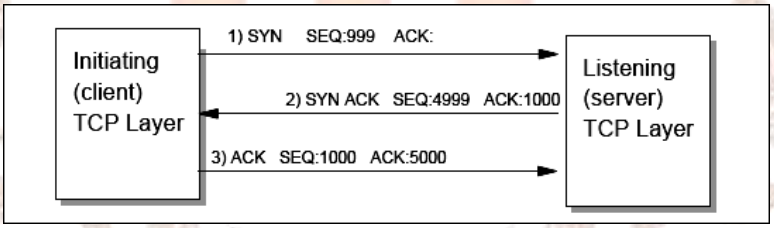

Fig 8.9: TCP Connection establishment

This whole process is known as a three-way handshake. Note that the exchanged TCP segments include the initial sequence numbers from both sides, to be used on subsequent data transfers.

The following table 8.2 shows the different states used in TCP connection management.

Table 8.2: Different states used in TCP connection management

State	Description
LISTEN	Awaiting a connection request from another TCP layer.
SYN-SENT	A SYN has been sent, and TCP is awaiting the response SYN.
SYN-RECEIVED	A SYN has been received, a SYN has been sent, and TCP is awaiting an ACK.
ESTABLISHED	The three-way handshake has been completed.
FIN-WAIT-1	The local application has issued a CLOSE. TCP has sent a FIN, and is awaiting an ACK or a FIN.

FIN-WAIT-2	A FIN has been sent, and an ACK received. TCP is awaiting a FIN from the remote TCP layer.
CLOSE-WAIT	TCP has received a FIN, and has sent an ACK. It is awaiting a close request from the local application before sending a FIN.
CLOSING	A FIN has been sent, a FIN has been received, and an ACK has been sent. TCP is awaiting an ACK for the FIN that was sent.
LAST-ACK	A FIN has been received, and an ACK and a FIN have been sent. TCP is awaiting an ACK.
TIME-WAIT	FINs have been received and ACK'd, and TCP is waiting for two MSLs to remove the connection from the table.
CLOSED	Imaginarily, this indicates that a connection has been removed from the connection table.

Closing the connection is done implicitly by sending a TCP segment with the *FIN* bit (no more data) set. Because the connection is full-duplex (that is, there are two independent data streams, one in each direction), the FIN segment only closes the data transfer in one direction. The other process will now send the remaining data it still has to transmit and also ends with a TCP segment where the FIN bit is set. The connection is deleted (status information on both sides) after the data stream is closed in both directions.

SELF-ASSESSMENT QUESTIONS – 2

8. _____ is a connectionless transport protocol.
9. Which protocol was designed to provide a reliable end-to-end byte stream over an unreliable internetwork?
   a) TCP
   b) UDP
   c) IP
   d) HTTP
10. Every TCP segment begins with a fixed-format byte header.
11. The number of 32-bit words in the TCP header which indicates where the data begins is known as _____.
12. Closing the connection is done implicitly by sending a TCP segment with the bit set.

5. TRANSPORT FOR REAL-TIME APPLICATIONS (RTP)

UDP is widely used in client server RPC (remote procedure call). Another area in which it is widely used is for real-time multimedia applications. Because of the need for a generic real-time transport protocol for multiple applications, RTP (Real-time Transport Protocol) was developed. It is described in RFC 3550 and is now in widespread use for multimedia applications. We will describe two aspects of real-time transport. The first is the RTP protocol for transporting audio and video data in packets. The second is the processing that takes place, mostly at the receiver, to play out the audio and video at the right time. These functions fit into the protocol stack as shown in figure 8.10.

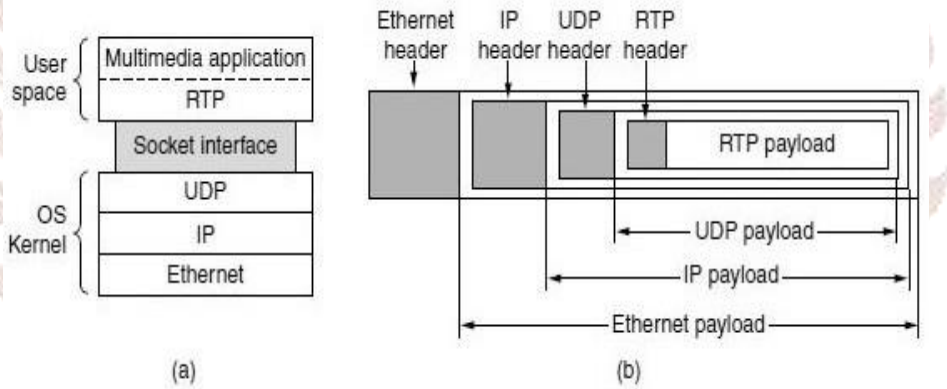

Fig 8.10: (a) The position of RTP in the protocol stack. (b) Packet nesting.

RTP normally runs in user space over UDP. The multimedia application consists of multiple audios, video, text, and possibly other streams. These are fed into the RTP library, which is in the user space along with the application. This library multiplexes the streams and encodes them in RTP packets, which it stuffs into a socket. On the operating system side of the socket, UDP packets are generated to wrap the RTP packets and handed to IP for transmission over a link such as Ethernet. The reverse process happens at the receiver. The protocol stack for this situation is shown in figure 8.10 (a). The packet nesting is shown in figure 8.10 (b).

The RTP format contains several features to help receivers work with multimedia information. Each packet sent in an RTP stream is given a number one higher than its predecessor. This numbering allows the destination to determine if any packets are missing. In audio or video data, retransmission is not a practical option since the retransmitted packet

would probably arrive too late to be useful. As a consequence, RTP has no acknowledgements, and no mechanism to request retransmissions.

The RTP header is illustrated in figure 8.11. It consists of three 32-bit words and potentially some extensions. The first word contains the Version field, which is already at 2.

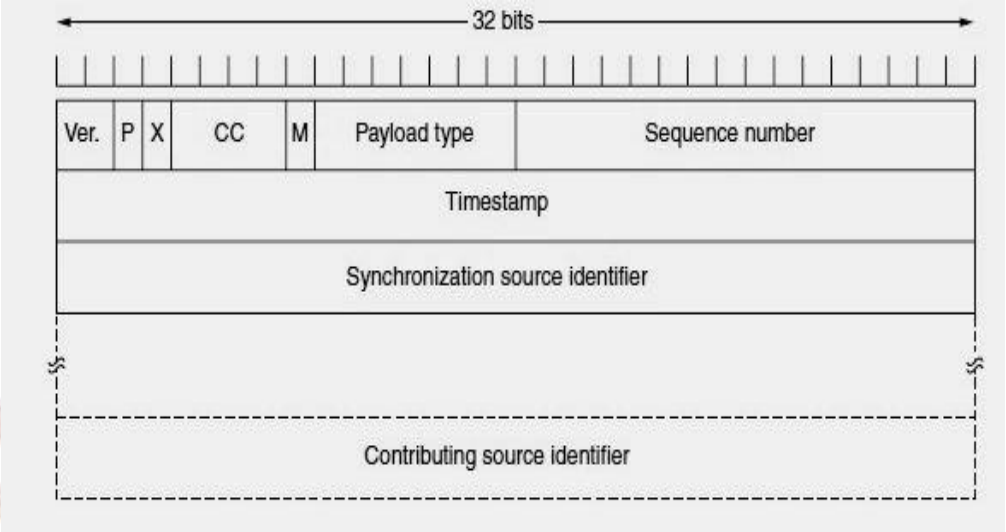

Fig 8.11: The RTP Header

The P bit indicates that the packet has been padded to a multiple of 4 bytes.

The last padding byte tells how many bytes were added. The X bit indicates that an extension header is present. The format and meaning of the extension header are not defined. The only thing that is defined is that the first word of the extension gives the length. This is an escape hatch for any unexpected requirements.

The CC field tells how many contributing sources are present, from 0 to 15. The M bit is an application-specific marker bit. It can be used to mark the start of a video frame, the start of a word in an audio channel, or something else that the application understands. The *Payload type* field tells which encoding algorithm has been used. Since every packet carries this field, the encoding can change during transmission. The *Sequence number* is just a counter that is incremented on each RTP packet sent. It is used to detect lost packets. The *Timestamp* is produced by the stream's source to note when the first sample in the packet was made. This value can help reduce timing variability called jitter at the receiver by decoupling the playback from the packet arrival time. The *Synchronization* source *identifier* tells which

stream the packet belongs to. It is the method used to multiplex and demultiplex multiple data streams onto a single stream of UDP packets. Finally, the *Contributing source identifiers,* if any, are used when mixers are present in the studio.

Real-time Transport Control Protocol (RTCP)

RTP has a sibling protocol called RTCP (Real time transport control protocol). It is defined along with RTP in RFC 3550 and handles feedback, synchronization, and the user interface. It does not transport any media samples.

The first function can be used to provide feedback on delays, variation in delay or jitter, bandwidth, congestion, and other network properties to the sources. This information can be used by the encoding process to increase the data rate when the network is functioning well and to cut back the data rate when there is trouble in the network. By providing continuous feedback, the encoding algorithms can be continuously adapted to provide the best quality possible under the current circumstances. The Payload type field is used to tell the destination what encoding algorithm is used for the current packet, making it possible to vary it on demand.

A trouble with providing feedback is that RTCP reports are sent to all participants. For a multicast application with a large group, the bandwidth used would be large. In order to prevent this, RTCP senders reduce the rate of their reports to collectively consume not more than 5% of the media bandwidth. Each participant can learn the media bandwidth from the sender and can estimate the number of participants by listening to other RTCP reports.

RTCP also handles interstream synchronization. The issue is that different streams may use different clocks, with different granularities and different drift rates. RTCP can be used to keep them in sync. Finally, RTCP provides a way for naming the various sources. This information can be displayed on the receiver's screen to indicate who is talking at the moment.

SELF-ASSESSMENT QUESTIONS – 3

13. RTP stands for_____.
14. RTP is in widespread use for _____applications.
15. RTP has a sibling protocol called_____.
16. Which protocol handles feedback, synchronization, and the user interface?
    (a) RTCP (b) TCP (c) UDP (d) HTTP

6. SUMMARY

Let us recapitulate the important concepts discussed in this unit:

- A port is a 16-bit number used by the host-to-host protocol to identify to which higher-level protocol or application program it must deliver incoming messages.
- Well-known ports belong to standard servers. Well-known port numbers range between 1 and 1023.
- The well-known ports are controlled and assigned by the Internet Assigned Number Authority (IANA) and on most systems can only be used by system processes or by programs executed by privileged users.
- Ephemeral port numbers have values greater than 1023, normally in the range of 1024 to 65535.
- Ephemeral ports are not controlled by IANA and can be used by ordinary user-developed programs on most systems.
- Socket APIs were first introduced by 4.2 Berkeley Software Distribution (BSD).
- UDP (User Datagram Protocol) is a connectionless transport protocol. UDP provides a way for applications to send encapsulated IP datagrams without having to establish a connection.
- TCP (Transmission Control Protocol) was designed to provide a reliable end-to-end byte stream over an unreliable internetwork.
- RTP is described in RFC 3550 and is now in widespread use for multimedia applications.
- RTP has a sibling protocol called RTCP (Real time transport control protocol).

- RTCP is defined along with RTP in RFC 3550 and handles feedback, synchronization, and the user interface. It does not transport any media samples.

7. TERMINAL QUESTIONS

1. Describe process to process delivery.
2. Explain user datagram protocol.
3. Describe TCP segment format.
4. Explain TCP connection establishment and connection release.
5. Describe Real-time Transport Protocol (RTP).

8. ANSWERS

Self-Assessment Questions

1. (a) True
2. Well-known, Ephemeral
3. 1, 1023
4. Internet Assigned Number Authority (IANA)
5. Ephemeral port numbers
6. Socket
7. <protocol, local-address, localport>
8. UDP
9. (a) TCP
10. 20
11. Data offset
12. FIN
13. Real-time Transport Protocol
14. Multimedia
15. RTCP (Real-time Transport Control Protocol)
16. (a) RTCP

Terminal Questions

1. An application process is assigned a process identifier number (process ID), which is likely to be different each time that the process is started. Process IDs differ between operating system platforms; Thus, they are not uniform. (Refer section 2 for more details).
2. UDP (User Datagram Protocol) is a connectionless transport protocol. UDP provides a way for applications to send encapsulated IP datagrams without having to establish a connection. UDP is described by RFC 768. (Refer section 3 for more details).
3. Every segment begins with a fixed-format, 20-byte header. The fixed header may be followed by header options. Segments without any data are legal and are commonly used for acknowledgements and control messages. (Refer section 4 for more details).
4. Before any data can be transferred, a connection has to be established between the two processes. One of the processes (usually the server) issues a passive OPEN call, the other an active *OPEN* call. The passive *OPEN* call remains dormant until another process tries to connect to it by an active OPEN. (Refer section 4 for more details).
5. Because of the need for a generic real-time transport protocol for multiple applications, RTP (Real-time Transport Protocol) was developed. It is described in RFC 3550 and is now in widespread use for multimedia applications. (Refer section 5 for more details).

References:

- Andrew S Tanenbaum, David J. Wetherall, *"Computer Networks,"* Fifth edition.
- Larry L. Peterson, Bruce S. Davie, "Computer *Networks – a Systems Approach*," Fifth edition.
- James F. Kurose, Keith W. Ross, "*Computer Networking – A top-down approach*," Sixth edition.
- Behrouz A. Forouzan, Sophia Chung Fegan, *"Data Communication and Networking,"* Fourth edition.
- William Stallings*, "Computer Networking with Internet Protocols and Technology,"* Third edition.