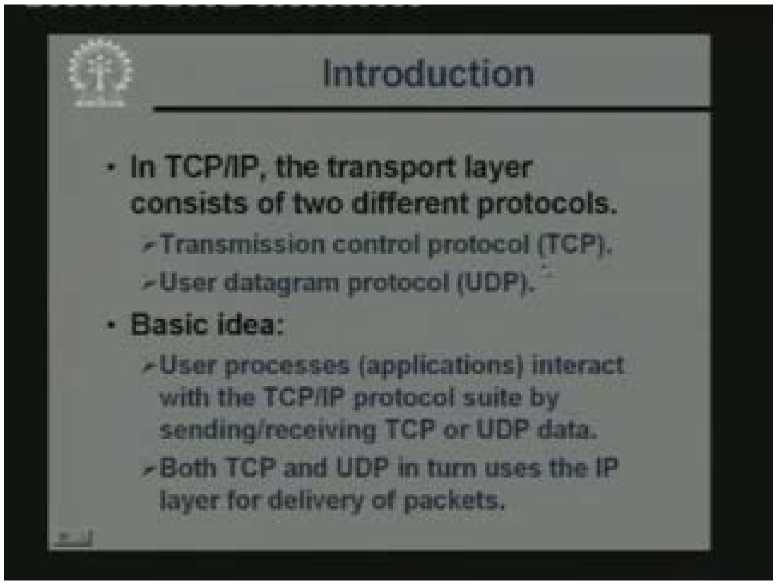
# Internet Technology

**Prof. Indranil Sengupta**
**Department of Computer Science and Engineering**
**Indian Institute of Technology, Kharagpur**

**Lecture No. # 05**
**TCP/IP – Part-III**

In our last lecture if you recall within our discussion on TCPIP. We had looked at some of the fields of the IP header which are responsible for the fragmentation and reassembly of packets. And finally we looked at well some issues regarding IP addressing; the address classes are in this lecture. We shall be discussing the transport level protocols that exist in TCP/IP.

(Refer Slide Time: 01:26)

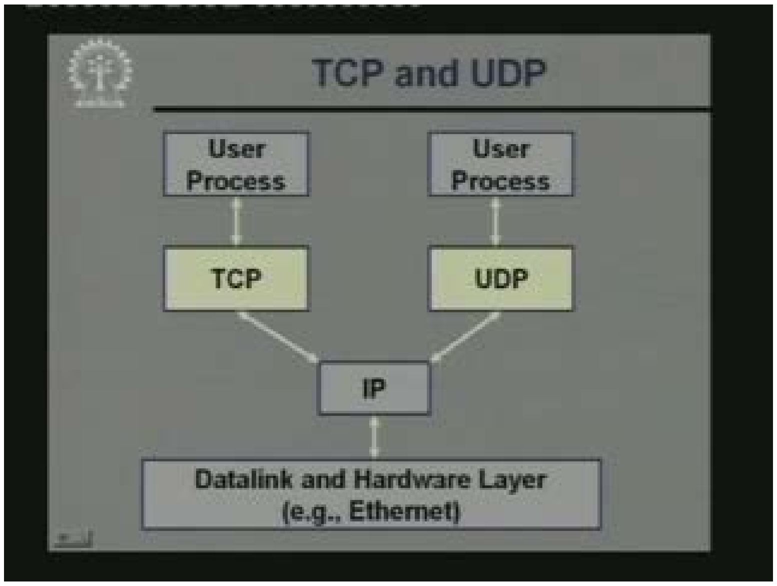


## Introduction

- In TCP/IP, the transport layer consists of two different protocols.
  - Transmission control protocol (TCP).
  - User datagram protocol (UDP).
- Basic idea:
  - User processes (applications) interact with the TCP/IP protocol suite by sending/receiving TCP or UDP data.
  - Both TCP and UDP in turn uses the IP layer for delivery of packets.

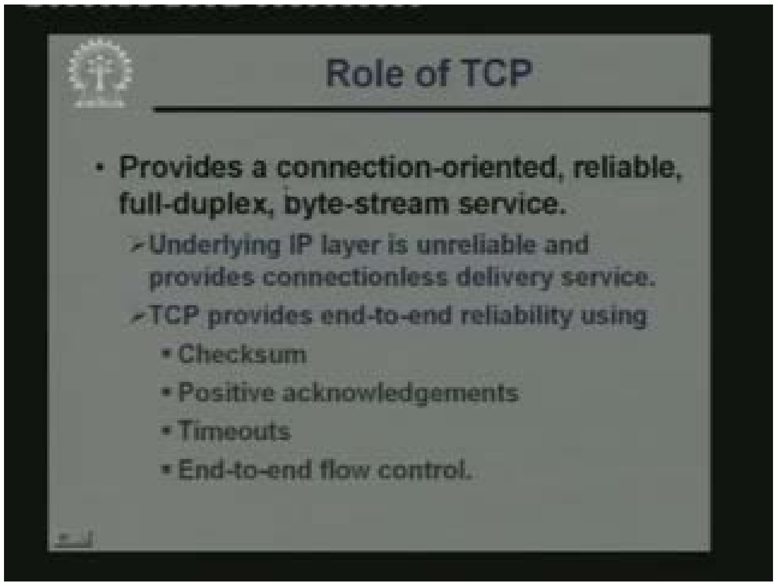
So we would be primarily looking at some of the features of the two transport level protocols in TCP/IP namely the Transmission Control Protocol TCP and the User Data Protocol UDP. Now these being transport layer protocols, the basic idea is that the applications which are running on the end hosts which you can treat as processes as programs and execution. They will be interacting with the TCP/IP protocol suite by sending receiving either TCP of UDP data depending on which of these protocols the application chooses to use. But one thing to notice that, both TCP and UDP they sit on top of the IP protocol. So both TCP and UDP will be generating the packets and forwarding them to IP for delivering them to the correct destination.

(Refer Slide Time: 02:36)



Pictorially the hierarchy looks like this. Let us have another look at this diagram. So here in today's lecture we would be concentrating on this part of the hierarchy. The transport protocols TCP and UDP. But the user processes will be sitting on top of them and the IP protocol would be sitting below them. So this is where TCP and UDP exist in the protocol hierarchy. Now first let us look at what are the basic purposes and roles of the TCP protocol, the Transmission Control Protocol. Now one thing to notice that, the main difference between TCP and UDP is primarily related to the kind of service that is provides to the applications which are invoking them.

(Refer Slide Time: 03:40)



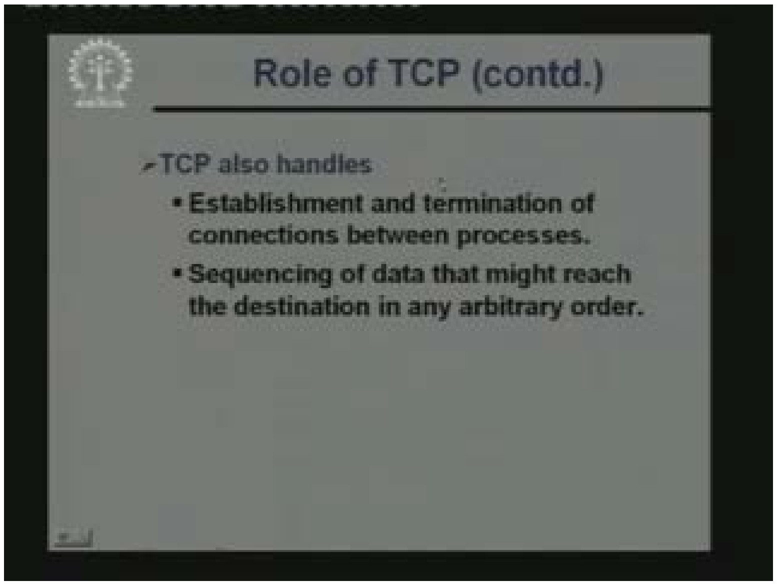
## Role of TCP

- Provides a connection-oriented, reliable, full-duplex, byte-stream service.
  - Underlying IP layer is unreliable and provides connectionless delivery service.
  - TCP provides end-to-end reliability using
    - Checksum
    - Positive acknowledgements
    - Timeouts
    - End-to-end flow control.

So TCP provides a so-called connection oriented reliable full duplex. Means bidirectional and byte stream service, this means that the two end hosts will have to establish a connection before data. Data communication can start communication is reliable reliable meaning even in case of some failure, some packet loss TCP will try to recover from that failure and regenerate the packet retransmit it if necessary. Full duplex means bidirectional and byte stream means to the applications the messages which are sent and which are received as if they are stream of bytes. But the point to notice that the underlying IP layer which TCP uses for the actual transport of the data is unreliable and also it is a datagram service which means it is connection less there is no concept of correction establishment in the IP layer.

So the question is that how then TCP ensures that the connection is reliable the communication reliable and there is a connection oriented service it does. So by explicitly maintaining a few things, first a checksum to check whether a packet or whether a TCP message is correct or not positive acknowledgements to send back information to the sender that will all the data messages have been received correctly or some have not been received. Time outs if acknowledgements do not come within a certain time. Then the data is again sent is retransmitted and flow control. Flow control means if the sender is sending data too fast the receiver can accept at that rate the receiver can send a request to the sender, that well you please slow down the speed of transfer, this is called flow control.

(Refer Slide Time: 05:54)

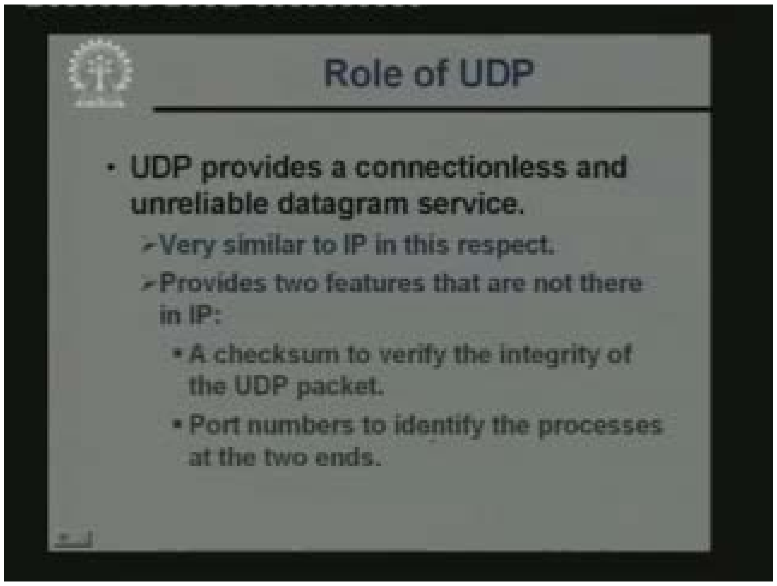


## Role of TCP (contd.)

- TCP also handles
  - Establishment and termination of connections between processes.
  - Sequencing of data that might reach the destination in any arbitrary order.

TCP in addition, since it is a connection oriented service it also handles establishment and termination of connections between applications of process and sequencing of data sequencing. Means at the level of TCP, say I am trying to send a message, it can be broken up into several packets and they may be sent by IP. Now IP does not guarantee any sequencing the packets may reach the destination in any arbitrary order. So what the TCP layer at the receiving side must ensure is that although small pieces that constitute the original message must be put together in the original order before they are forwarded to the application. So application will have a feeling that the messages coming in sequence in the order of bytes in byte order. They will not be forwarded to the application in an out of order fashion.

(Refer Slide Time: 06:57)

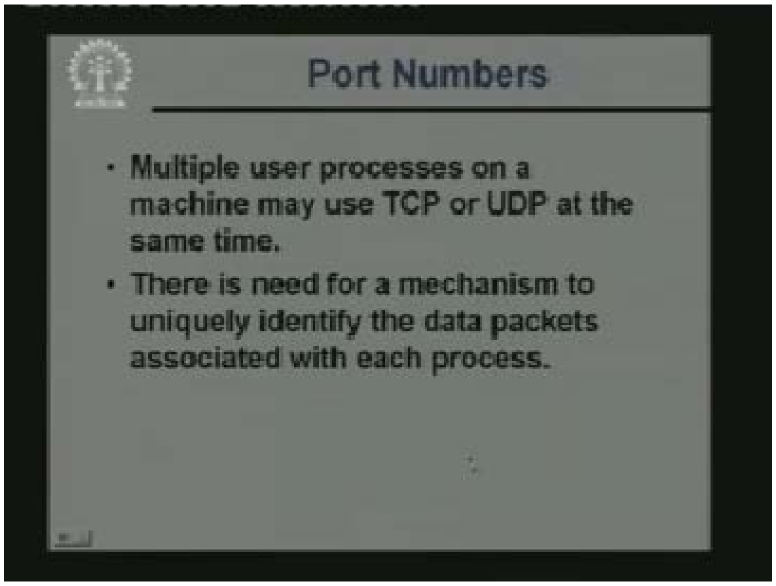


## Role of UDP

- UDP provides a connectionless and unreliable datagram service.
  - ➢Very similar to IP in this respect.
  - ➢Provides two features that are not there in IP:
    - A checksum to verify the integrity of the UDP packet.
    - Port numbers to identify the processes at the two ends.

Now in contrast the user data protocol or UDP it provides a connectionless and unreliable datagram service. So in this sense UDP is very similar to IP. Because IP is also connectionless IP is also a datagram base service. So now the question is that if they are so similar what extra does UDP have over IP? Well in fact UDP provides two additional things in addition to what is being provided by IP. The first feature is a checksum which can be used to verify the integrity of the UDP packet. Of course this checksum is an optional field. If it is not used the checksum field is set to all 0s. Secondly and more important is that it keeps track of something called port numbers. These port numbers will identify the processes at two ends because when UDP sends a packet to a destination it is targeted towards a particular process or an application which is running at the destination. So the port number in the packet will uniquely identify that which destination this packet must go to. So if a port number was not there then all packets which are targeted towards all applications running on a particular machine would reach that machine and would be no way to find out which jacket should be sent to which application. So port number is used specifically for that purpose.

(Refer Slide Time: 08:44)

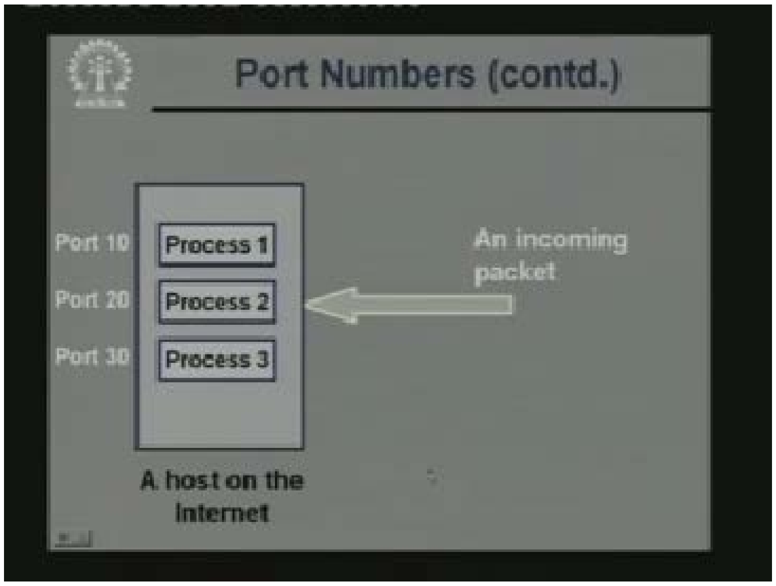


## Port Numbers

- Multiple user processes on a machine may use TCP or UDP at the same time.
- There is need for a mechanism to uniquely identify the data packets associated with each process.

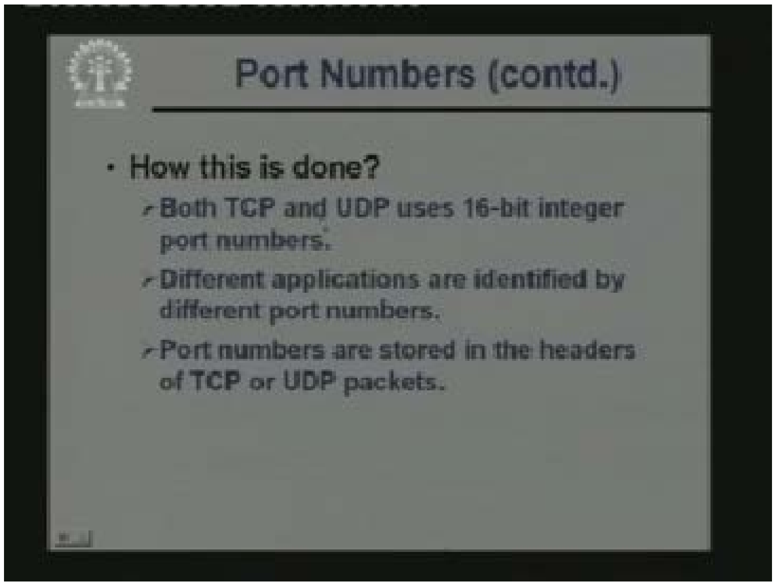
So this exactly what I just talked about multiple user processes on a machine may be using a transport layer protocols TCP or UDP at the same time. So when data packets are received, they are coming to the machine there is a need for a mechanism. So has to uniquely identify that which crosses the data packets need to be forwarded to.

(Refer Slide Time: 09:15)



Let us look at this diagram to illustrate. Suppose this box indicates a host on the internet this is a computer. And inside this host, there are three processes; Process1 and Process 2 and Process 3 which are executing or running. Suppose we assign port numbers 10, 20 and 30 respectively to these processes. So when an incoming packet is coming to this host it must also carry along with it the port number. So if it carries a port number of 20 that the packet will automatically be delivered to Process 2. If it carries port number 30, it will be delivered to Process 3 and so on. So this simple illustration will help you understand the importance of the need for having these port numbers. Port numbers will uniquely identify the applications which on the hosts. Now port numbers are 16 bit quantities. They are used to address an application and you recall IP addresses are a 32 bit quantity which identifies a host. In contrast the port number is the 16 bit quantity.

(Refer Slide Time: 10:41)

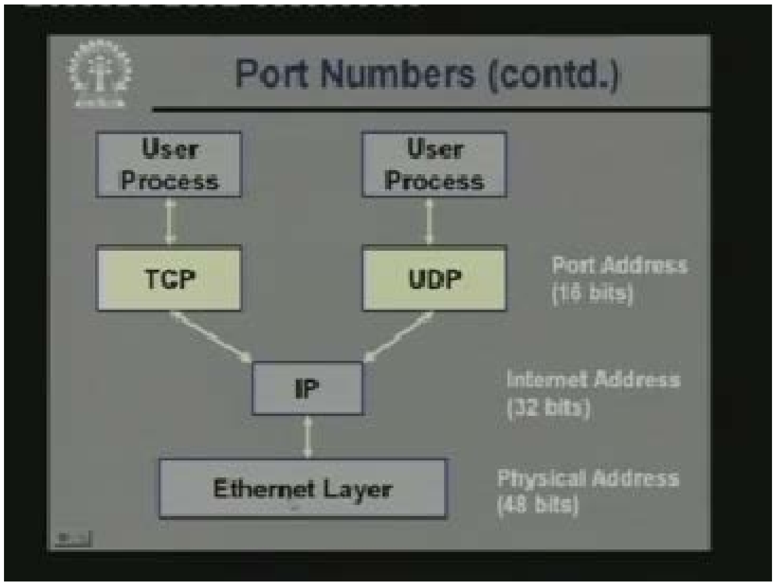


## Port Numbers (contd.)

- How this is done?
  - Both TCP and UDP uses 16-bit integer port numbers.
  - Different applications are identified by different port numbers.
  - Port numbers are stored in the headers of TCP or UDP packets.

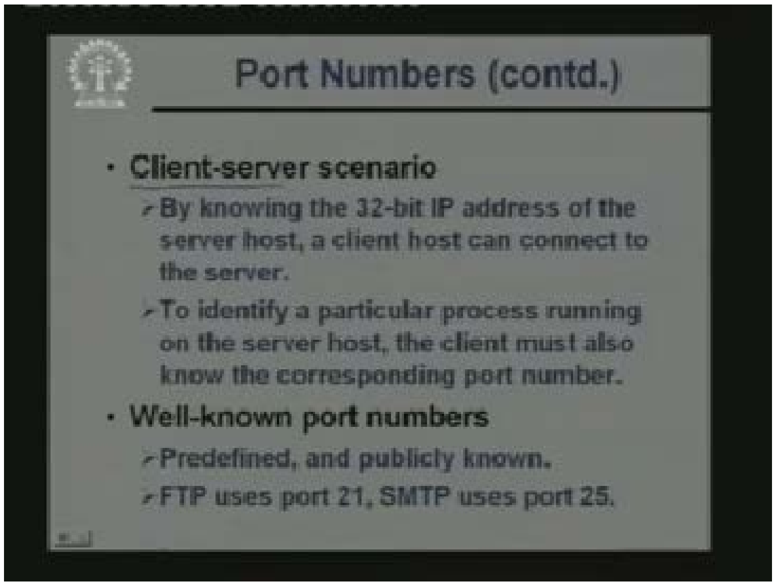
So both TCP and UDP use these 16 bit integers to represent port numbers and as I mentioned that applications are uniquely identified by different values of the port numbers. And the TCP or UDP packets will have the port numbers stored in their headers so that the packets can be delivered to the correct application.

(Refer Slide Time: 11:13)


## Port Numbers (contd.)

So just another look at this picture, in this hierarchy at the lowest level Ethernet here we address with respect to the Ethernet address is 48 bits at the level of IP. We use the internet address of the IP address 32 bits at the transport layer level. We use port numbers to identify the application and this is 16 bits. So at the different levels of the hierarchy we use different ways of addressing the entities at the lowest level who are identifying the physical interface. At the next higher level we are identifying a host at the next higher level we are identifying a particular application which is running on the host. Now in a typical client/server scenario let us see what happens. Because this client/server scenario is so common nowadays there is a server program which is running on some machine and is providing some service to other so called client programs and the client program running on another machine can send an explicit request to the server. The server program will be receiving the request will do the needful and will be sending back the response to the client of this is how the client server environment works.

(Refer Slide Time: 12:35)



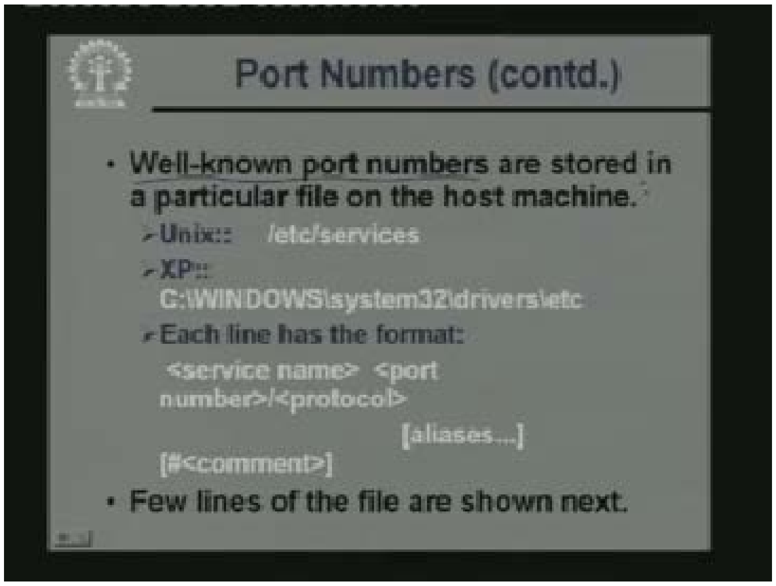
## Port Numbers (contd.)

- Client-server scenario
  - By knowing the 32-bit IP address of the server host, a client host can connect to the server.
  - To identify a particular process running on the server host, the client must also know the corresponding port number.
- Well-known port numbers
  - Predefined, and publicly known.
  - FTP uses port 21, SMTP uses port 25.

Now let us see in the client server scenario by knowing the 32 bit IP address of the server a client host can first establish a connection to the server. But not only this, in the client server scenario, the appropriate server program the particular process must also be identified. So the client must also know the corresponding port number. This means that whenever you are you are working on the client server scenario and a client wants to connect to a server you need to know the IP address of the server machine and also the port number which will identify the particular process which is running on the server machine. Say for example you just think of the World Wide Web the so-called HTTP protocol. Now whenever you are using a browser you want to visit a site you first type in either a site name or an IP address. There is a mechanism to translate a name into an address that we would be talking about later. But in some way it is getting translated into an IP address. But in addition in your browser setting somewhere it is set that if it is an HTTP request by default set the port number to 80.

Because 80 is a number that identifies the HTTP server which runs on the other machine. So in this way both the IP address and the port numbers are specified by the client of fine among the port numbers there are such port numbers which are predefined and are made publicly available or publicly known. For example port number 21 is used for the file transfer protocol for transferring files between machines. Port number 25 is used for the simple mail transfer protocol which is used in emails to these are just two examples there is a big such list which can tell you the list of all publicly available port numbers which everyone can use in order to contact the server or some particular machine. Now information regarding these port numbers and the servers are stored in computer systems in a particular directory. Because a computer must also know in some way that well port number 21 corresponds to the FTP protocol and port number 25 corresponds to the SMTP protocol. There is a particular file some where in the machine which contains all these details.

(Refer Slide Time: 15:32)



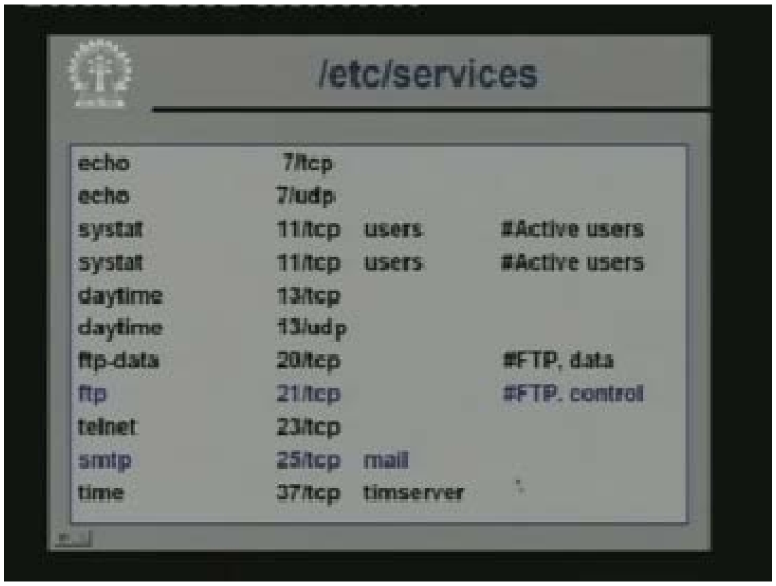
## Port Numbers (contd.)

- Well-known port numbers are stored in a particular file on the host machine.
  - Unix:: /etc/services
  - XP:: C:\WINDOWS\system32\drivers\etc
  - Each line has the format:

    <service name> <port number>/<protocol> [aliases...] [#<comment>]
- Few lines of the file are shown next.

These so called well known port numbers of stored in a file which on a UNIX machines. This is stored in a directory under root etc services. Another example under the Windows XP environment, it is stored in a directory like this under C Windows system 32 drivers etcetera. In this file there are a set of lines or records. Now each line or records has the format. So it starts with the service name, then the port number, then the protocol, then the aliases. That means I can have some other names to the service names followed by the comments. I will show an example in the next slide few lines of such a file we shall be having a look now.

(Refer Slide Time: 16:27)

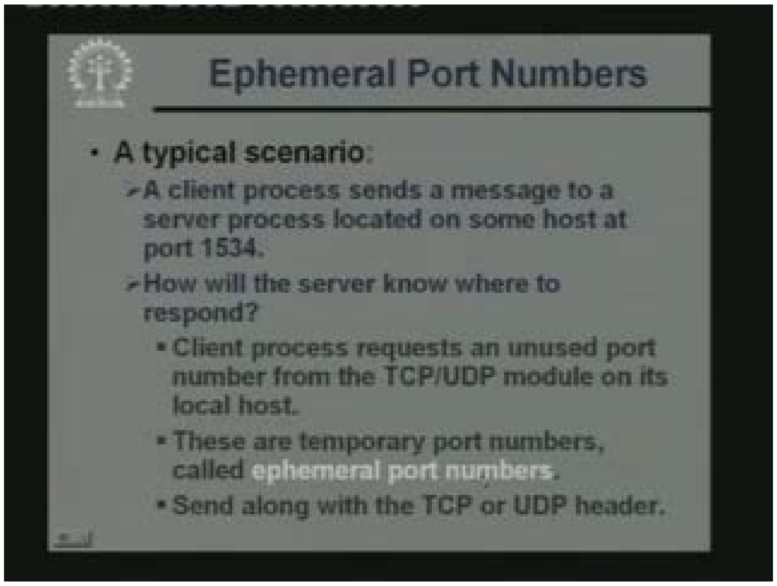


## /etc/services

```
echo          7/tcp
echo          7/udp
systat        11/tcp   users        #Active users
systat        11/tcp   users        #Active users
daytime       13/tcp
daytime       13/udp
ftp-data      20/tcp                #FTP, data
ftp           21/tcp                #FTP. control
telnet        23/tcp
smtp          25/tcp   mail
time          37/tcp   timserver
```

This is the content of the first few lines of the etcetera services files in an UNIX. The first column, these identify the names of the services or the servers. You see here you have some somewhere FTP, you have some SMTP and there are other also echo system etcetera. Now you look at the other fields FTP it says. Here 21 slash TCP, this means FTP protocol uses port number 21 and the TCP protocol this is a comment line. Just it says that it is a FTP protocol, this is a control connection take some other example say SMTP. First it says port number 25 TCP protocol there is an alternate name or alias. This is called mail so instead of SMTP you can also refer to this service or server as m a I l mail. Now you can see here some of the protocols also use the UDP protocol port number 7 UDP. So this is the kind of information that gets stored somewhere in each computer system. So that when a particular packet comes just by looking at that file the operating system can decide to which program the particular packet should be forwarded to. Because the information about port number and also the protocol TCP or UDP that is stored here fine.

(Refer Slide Time: 18:09)

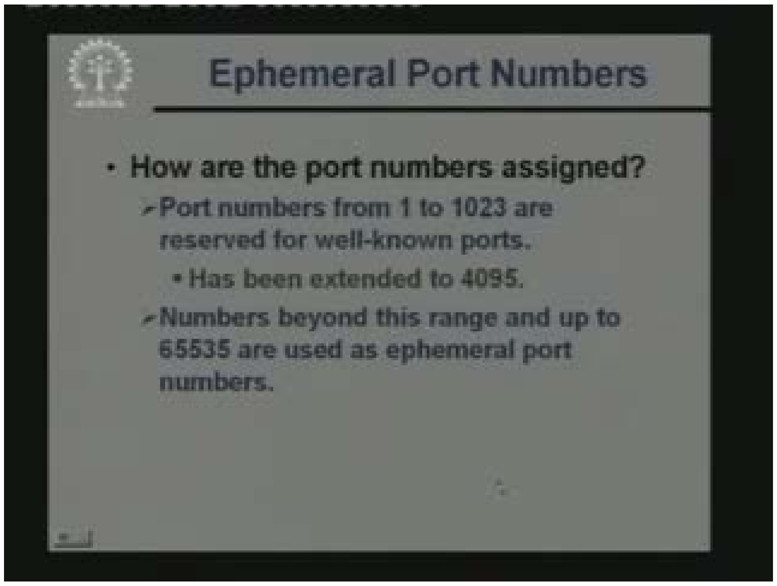


## Ephemeral Port Numbers

- A typical scenario:
  - A client process sends a message to a server process located on some host at port 1534.
  - How will the server know where to respond?
    - Client process requests an unused port number from the TCP/UDP module on its local host.
    - These are temporary port numbers, called ephemeral port numbers.
    - Send along with the TCP or UDP header.

Now there is something called ephemeral port numbers. Let us try to understand what this is? Let us look at a typical scenario. A client process sends a message to a server which is located on some host say at 1534. Now the server will have to respond back to the client. The question is how the server will know where to respond. Because the server does not know the port number of the client process and the server is well known. The client is not well known. So what the client process will do? The client process will request the local operating system for an unused port number. It gets generated these unused and temporary port numbers are called ephemeral port numbers. These are purely temporary. These temporary port numbers exist only for the duration of the current connection after the connection is terminated the port numbers are withdrawn. So these are purely temporary port numbers used on the client side. These are called ephemeral port numbers.

(Refer Slide Time: 19:27)

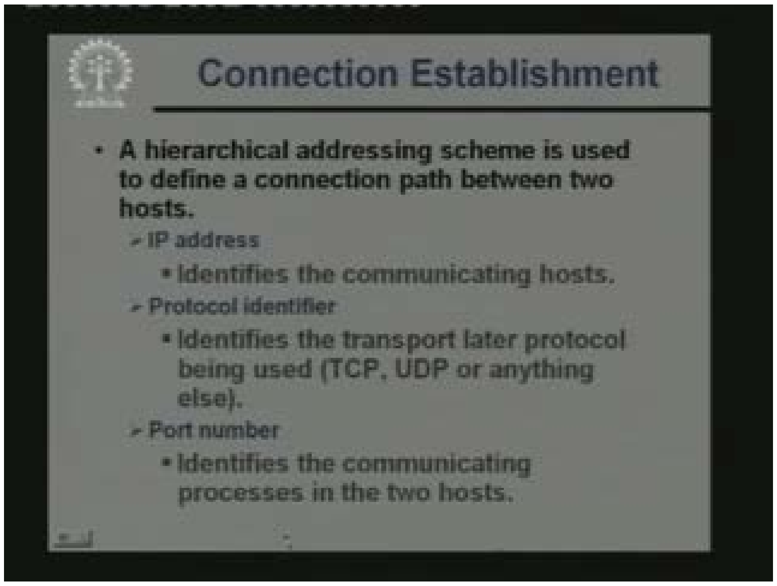


## Ephemeral Port Numbers

- How are the port numbers assigned?
  - Port numbers from 1 to 1023 are reserved for well-known ports.
    - Has been extended to 4095.
  - Numbers beyond this range and up to 65535 are used as ephemeral port numbers.

Now with respect to port number assignment, well typically the port numbers starting from 1 up to 1023, these are reserved for well-known port numbers. But some of the applications today also extend this number up to 4095. But anything beyond this up to a maximum of 65535. This can be used by a client program as ephemeral port numbers. So you have a pretty large range of port numbers available for use on the client side and these numbers are used as the ephemeral port numbers whenever the client wants to make a connection with the server.

(Refer Slide Time: 20:13)

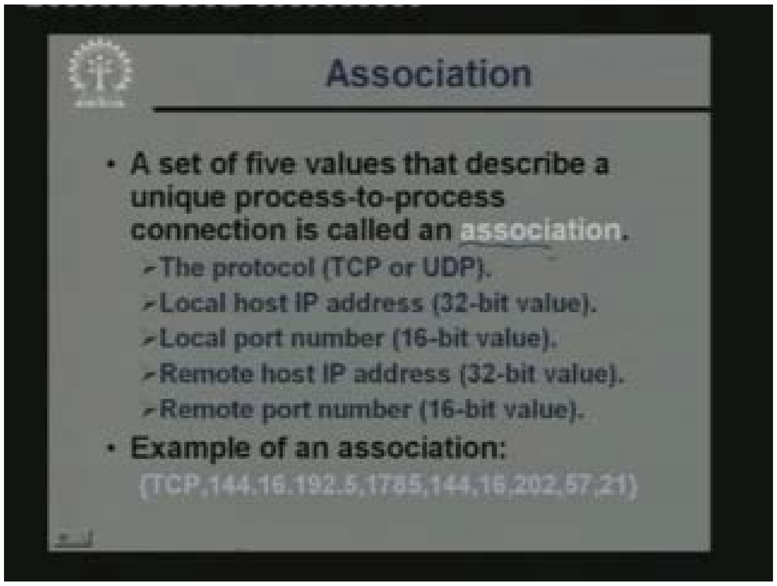


## Connection Establishment

- A hierarchical addressing scheme is used to define a connection path between two hosts.
  - IP address
    - Identifies the communicating hosts.
  - Protocol identifier
    - Identifies the transport later protocol being used (TCP, UDP or anything else).
  - Port number
    - Identifies the communicating processes in the two hosts.

Now talking about connection establishment there is a hierarchical addressing scheme. There are three things which need to be specified when you when you establish a connection. First is the IP address of the two ends the source and the destination, next is the protocol whether you are using TCP, UDP or any other protocol. The third is port number to identify the processes at the two ends. These are the three things you need to specify when establishing a connection.

(Refer Slide Time: 20:55)

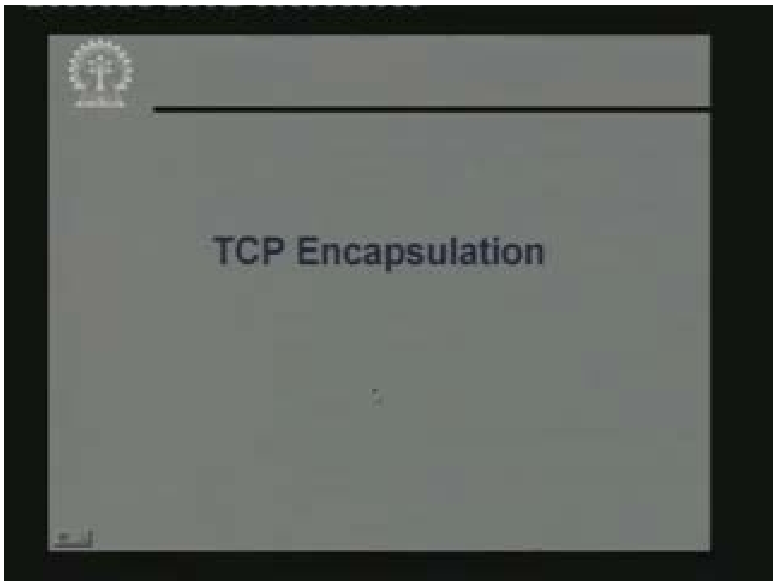


## Association

- A set of five values that describe a unique process-to-process connection is called an association.
  - The protocol (TCP or UDP).
  - Local host IP address (32-bit value).
  - Local port number (16-bit value).
  - Remote host IP address (32-bit value).
  - Remote port number (16-bit value).
- Example of an association:
  {TCP,144.16.192.5,1785,144,16,202,57,21}

Now when a connection is established we define something called an association. An association is actually a collection of these 5 things. The protocol TCP or UDP, local IP address, local port number, remote IP address, remote port number. Example of association is this. This is TCP address, local IP, local port, remote IP, remote port by dots. So this example this will these 5 information, this 5 components of this association will uniquely identify a connection which is getting established. So when you say that a connection has been established it means both the sides have come to know, all the five components of an association only then the packets can be send and received with all the relevant information.

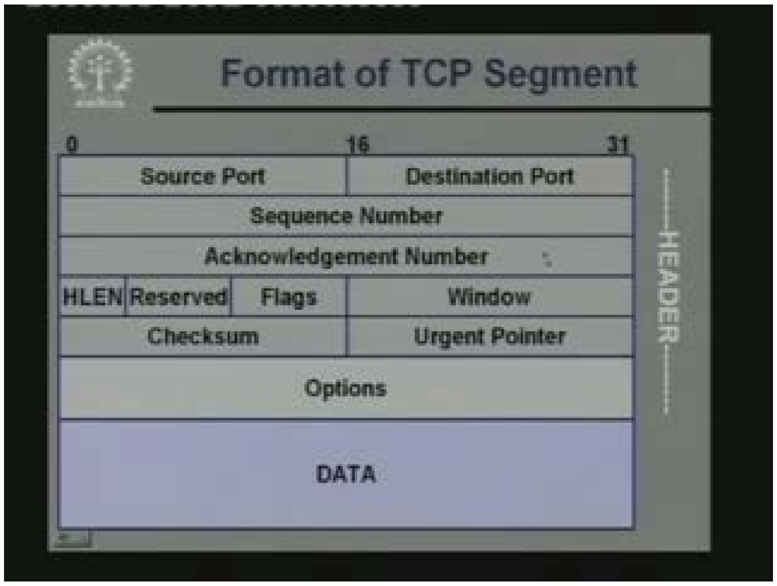
(Refer Slide Time: 21:57)



## TCP Encapsulation

So now let us have a quick look at the format of the TCP packet. The TCP header how TCP carries out encapsulation.

(Refer Slide Time: 22:07)

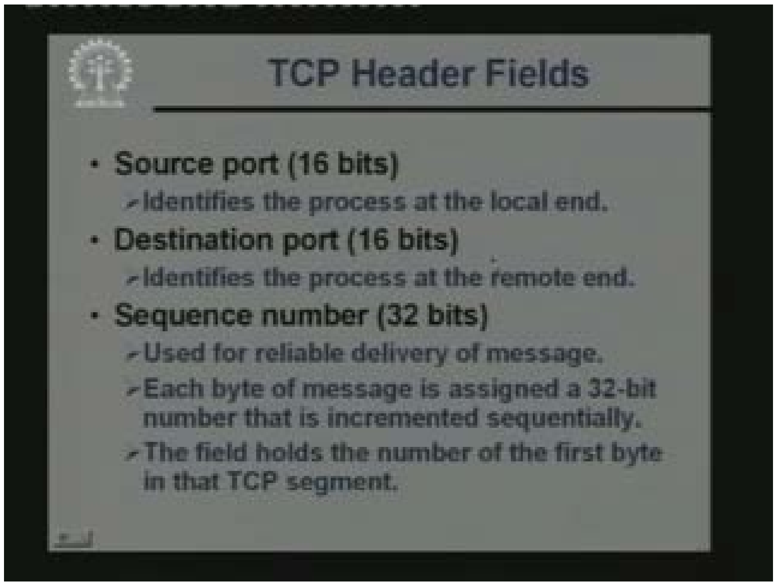


## Format of TCP Segment

| 0 | | | 16 | 31 |
|---|---|---|---|---|
| Source Port | | | Destination Port | |
| Sequence Number | | | | |
| Acknowledgement Number | | | | |
| HLEN | Reserved | Flags | Window | |
| Checksum | | | Urgent Pointer | |
| Options | | | | |
| DATA | | | | |

This is the diagram that shows the TCP header, a TCP segment which contains source port, destination port, sequence number, acknowledgement. There is header length; flags, there is some reserved fields; window, checksum, urgent pointer and again there are some optional fields here. Now let us very quickly see that what these fields are used for fine.

(Refer Slide Time: 22:09)

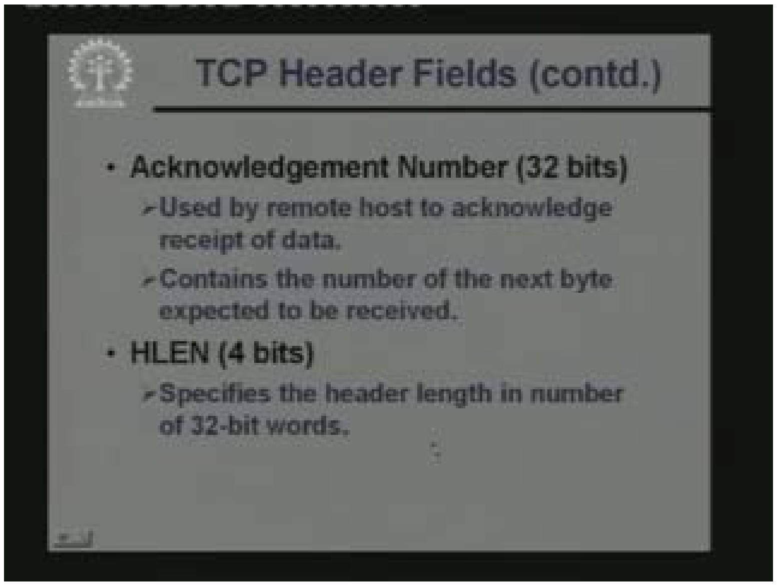


TCP Header Fields

- Source port (16 bits)
  - Identifies the process at the local end.
- Destination port (16 bits)
  - Identifies the process at the remote end.
- Sequence number (32 bits)
  - Used for reliable delivery of message.
  - Each byte of message is assigned a 32-bit number that is incremented sequentially.
  - The field holds the number of the first byte in that TCP segment.

Source port and destination port are self-explanatory. Sequence number is used for reliable communication. Now you are trying to transmit a message, now a message may not be transmitted in one go. It may have to be broken up into smaller pieces and several TCP packets may be generated corresponding to the same message. So what TCP will do? TCP will assign a sequence number to each piece of the message and that sequence number will correspond to the starting byte numbered in the original message from where that piece starts. So each bytes of message is assigned a 32 bit number, this is a byte level addressing that is incremented sequentially the field holds the number of the first byte of that particular TCP segment you are trying to send.

(Refer Slide Time: 23:40)

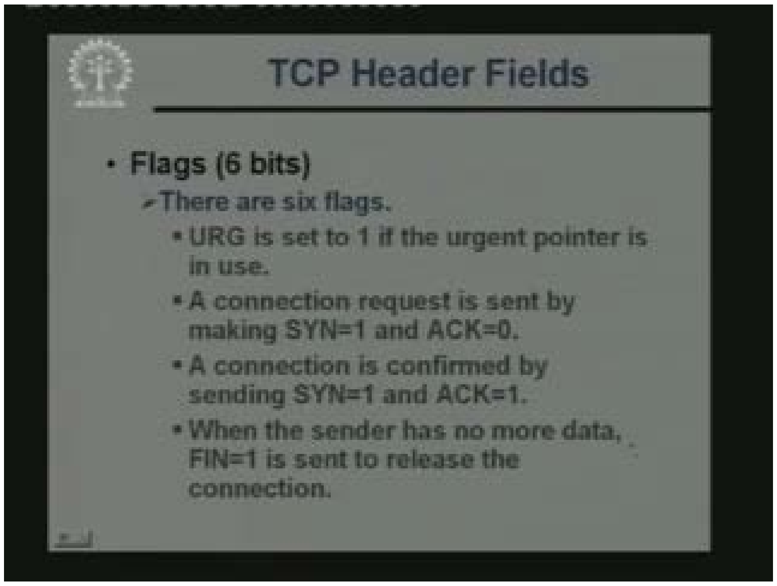


TCP Header Fields (contd.)

- Acknowledgement Number (32 bits)
  - Used by remote host to acknowledge receipt of data.
  - Contains the number of the next byte expected to be received.
- HLEN (4 bits)
  - Specifies the header length in number of 32-bit words.

Similarly from the other side you have an acknowledgement number 30 bit which is sent back the remote host is using this to acknowledge something. This contains the number of the next byte expected to be received. This means if the sender has is if the sender is sending the TCP segments one after the other, the receiver will receive the packets say up to byte number 1000. So receive will be sending back and acknowledgement with acknowledgement field value as 1001. So it says that 1001 is the next byte I am expecting. So these acknowledgements help in TCP in knowing or the sender to know that whether receiver has received all the segments correctly or not. And if not the segments which has not been received correctly is transmitted again. So even case of failure the failure is overcome through retransmission. Then just like the IP protocol we have header length, field 4 bits. This also specifies the header length in terms of 32 bit words.

(Refer Slide Time: 25:00)

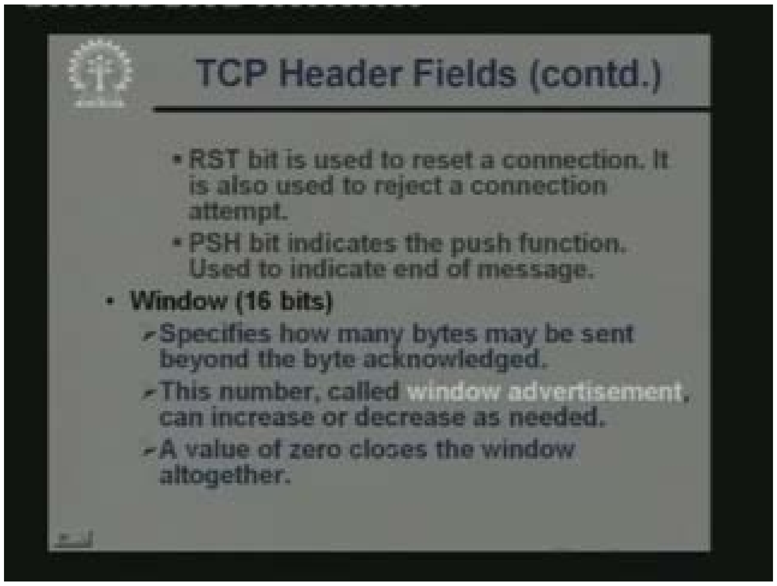


## TCP Header Fields

- **Flags (6 bits)**
  - There are six flags.
    - URG is set to 1 if the urgent pointer is in use.
    - A connection request is sent by making SYN=1 and ACK=0.
    - A connection is confirmed by sending SYN=1 and ACK=1.
    - When the sender has no more data, FIN=1 is sent to release the connection.

Flags there are 6 flags on their purposes are various. But I am not going the detail; I am just mentioning some of the some of the typical uses of this flags. There is a urgent flag URG, this is set to 1, if this so called urgent pointer is in use well urgent pointer actually points to some segment in the data which can contain some very important information for the receiver. So if the urgent pointer is set to 1, the receiver can immediate take out that urgent data and processes it immediately. And there is a SYN flag and an ACK flag if it is one SYN is 1 and ACK is 0, this means this sender is trying to send a connection request. If both are 1 which means the connection is confirmed there are, other combinations also and FIN is another flag finish. When the sender has no more data FIN equal to 1 is send to release the connection. That means now the connection can be terminated.

(Refer Slide Time: 26:08)

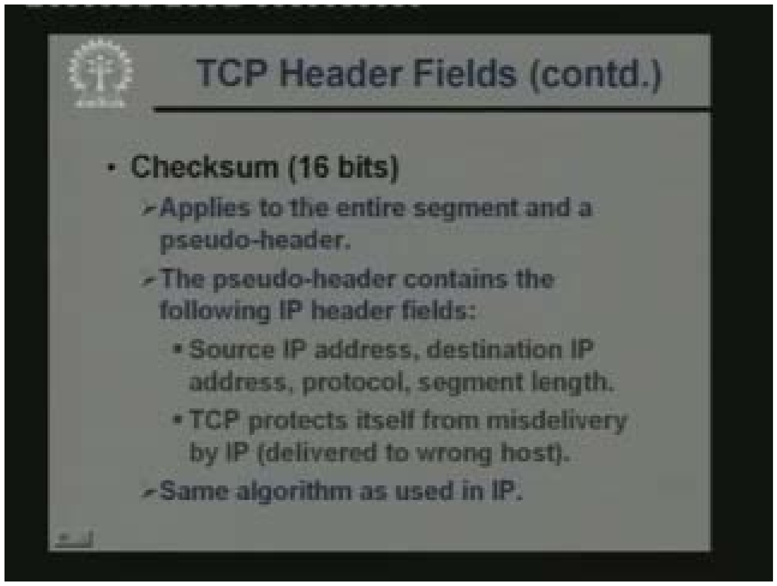

## TCP Header Fields (contd.)

- RST bit is used to reset a connection. It is also used to reject a connection attempt.
- PSH bit indicates the push function. Used to indicate end of message.
- **Window (16 bits)**
  - Specifies how many bytes may be sent beyond the byte acknowledged.
  - This number, called window advertisement, can increase or decrease as needed.
  - A value of zero closes the window altogether.

There is a reset bit RST this is used to reset or to reject a connection request there is a push flag this is used typically to indicate end of message. And there is another way interesting field window field is a 16 bit window field actually TCP uses a kind of sliding window protocol at the level of bytes and the window field indicates the current size of the window. Say if the window field contains a value 5000, this means that the sender can transmit maximum up to 5000 bytes without an acknowledgement coming back. So it specifies how many bytes may be sent beyond the byte acknowledgement, this number is called window advertisement. Now this number can be increased or decreased as per the requirements suppose the receiver finds that the sender is transmitting data too fast. It is not able to receive at that rate.

So now what the receive will do it will send back a TCP request to the sender requesting it to reduce the value of the window. So if the window value is reduced then the sender is prevented from sending data at that rate. So the rate can be increased or decreased by appropriately setting the value of the window field. So if the value of the window field is set to 0. This means the window will be closed altogether. That means the data transmission will stop. Now this window field is actually used for congestion control the network. If a segment of the network becomes congested, congested means the speed goes down the buffer space gets filled up then the window size may be reduced through a proper negotiation protocol. So that the congestion can be reduced, so this congestion and flow control these are two things which are served by this window field in TCP.

(Refer Slide Time: 28:35)

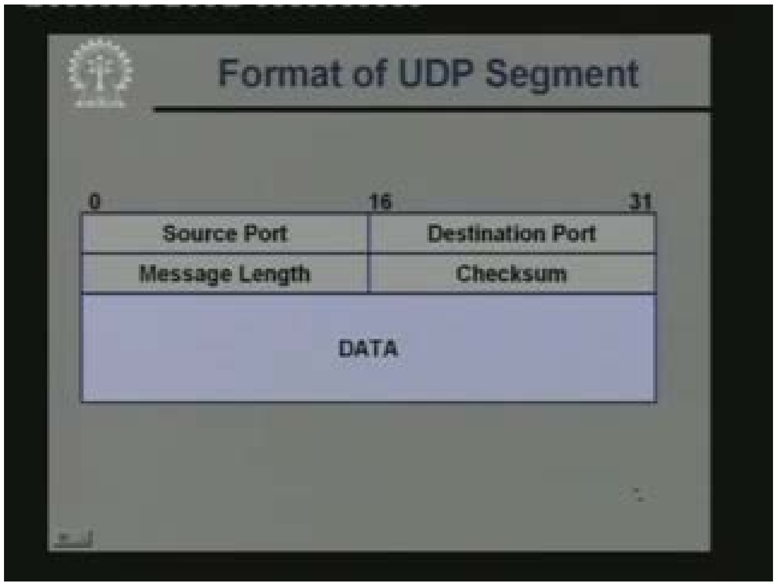


## TCP Header Fields (contd.)

- Checksum (16 bits)
  - Applies to the entire segment and a pseudo-header.
  - The pseudo-header contains the following IP header fields:
    - Source IP address, destination IP address, protocol, segment length.
    - TCP protects itself from misdelivery by IP (delivered to wrong host).
  - Same algorithm as used in IP.

And lastly there is checksum field which is 16 bits. Now the way it is computed is the same as IP that the algorithm used the same as IP. You take 16 bit segments of the numbers you take 1 complements sum and finally take the 1s complement of the result. But the data on which the algorithm is applied, there is something to observe. Here this check sum is applies to the entire TCP segment plus a pseudo header. Pseudo header contains from information which are barrowed from the IP headers. Pseudo header contains the following fields Source IP, address Destination IP, and address Protocol Segment length. Now the purpose of including this pseudo header in this check sum computation is that, suppose say IP is an under level service TCP knows that suppose by mistake IP forward the package to a wrong destination host. So there at the final destination when this TCP text sum is computed that error will be found out because the IP address are different. So this feature of pseudo header is mainly used for protecting TCP from miss delivery by IP. So some of the failure of delivering a packet to the correct destination by IP is taken care of by using this pseudo header.

(Refer Slide Time: 30:30)



## Format of UDP Segment

| 0 | 16 31 |
|---|---|
| Source Port | Destination Port |
| Message Length | Checksum |
| DATA | |

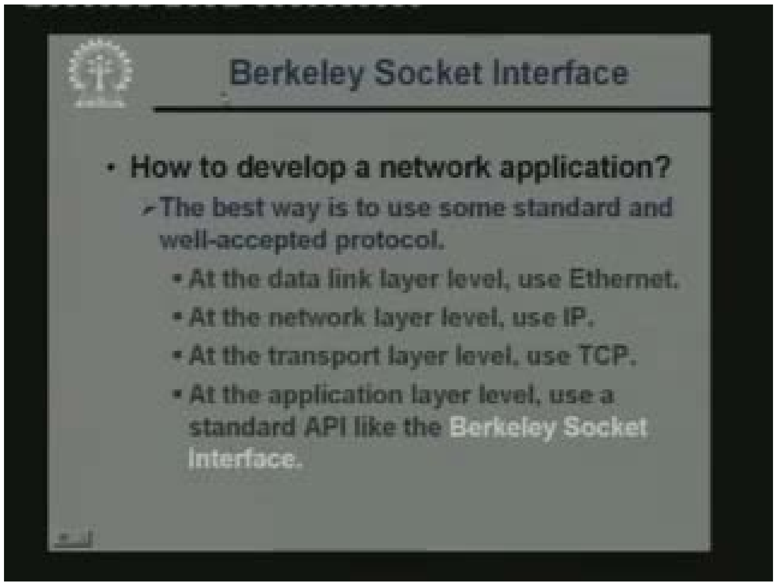
Now let us look at the UDP Protocol UDP Header format. As you can see from this diagram, the UDP Packet structure is extremely simple. There is no fancy headers, only 4 fields are there Source port, Destination port, Message length Checksum and of course the data. So in UDP the source port and destination port as usual meanings, Message length will specify the size of the datagram in bytes. Now when you say datagram it includes the UDP header plus data. Checksum is computed in the same way as TCP. Same way means here also include the pseudo header, include all the fields of the IP which are relevant for computing it and we use the same algorithm that taking the once complement and adding. But the point to note in UDP this checksum is not a mandatory field. You can also keep it optional this field will be set to 0 if it is not used. Now let us try to observe one thing.

Now we have two competing protocols at the transporting level in the TCP level protocols. Just to recall TCP provides you a connection oriented reliable service for the transport of messages. UDP provides you connection less unreliable datagram service. Now if you try to find out what kind of applications would be using which of these than naturally you can say at well if you have an application which does not like to so any rechecking out of its own it wants the under line network layer to provide all the reliability it requires. Then TCP will be the choice for the application. But here there are couples of things to know. See a particular applications if it uses TCP then it is true that you get very high reliability, you get a connection oriented service in the sense that before the communication starts, you have to explicitly establish the connection.

Now but the price of paying is that TCP is not a very efficient protocol, TCP is you can say relatively high overhead protocol it is relatively slow. But if you have some application, but the size of the messages are not very large they are small enough. So that they can fit into a particular package then you may choose to use datagram. But here the point to notice that the application should be such if a few packets get lost in transmit because UDP will never give you a guarantee of correct packet delivery. So the application should be such that even if there is an error in delivery some packets are getting lost either that loss can be tolerated or higher level protocol must explicitly keep track of that. I am giving you two examples, there is one protocol. As I mentioned earlier the Trivial File Transfer Protocol (TFTP) that uses UDP, but since the file transfer is an important operation.

Even a small portion of a file, if it is not transmitted, you cannot tolerate that. So TFTP exclusively keeps track of the portions of the file of the transmitted and which of the portions not been transmitted correctly. Take an another example say for example Simple Network Management Protocol, SNMTP is another protocol which is used to send some kind of network statistics information over the nets. So that in a centralized locations you can collect all the data and view the current health of the network statistically. In that application even one packet gets dropped it does not matter because the packet is sent ever few minutes or so it depends on the applications. So whether you really want to use TCP or UDP. Now the next question is TCP and UDP are transport layer protocols 5. Now we are saying that application layer protocols or applications are just sitting top of it they use either TCP or UDP. Now as a programmer I may ask the question that well how do I use TCP or UDP suppose I am writing a program in language like C, C plus plus, visual basic anything. So how do I really use TCP or UDP?

(Refer Slide Time: 36:32)



## Berkeley Socket Interface

- How to develop a network application?
  - The best way is to use some standard and well-accepted protocol.
    - At the data link layer level, use Ethernet.
    - At the network layer level, use IP.
    - At the transport layer level, use TCP.
    - At the application layer level, use a standard API like the Berkeley Socket Interface.

Now the answer is to have some kind of a standard interface. This is one very standard interface which people use. This is called Berkeley Socket Interface. Now the name Berkeley Socket Interface, this has come because this interface was first you can saw announced as part of the so called Berkeley socket distribution or BSD Unix. That was the very well-known version of UNIX which was developed at Berkeley and the socket interface was developed as a part of it. Now this Berkeley socket interface is a means for accessing the TCP or IP services over the network. So now let us see that how we can develop a network application. I am just giving you the outline. But it is the procedure to follow. Well of course some standard and well-accepted protocols are the most desirable. So in the modern day of computing at the data link layer level in most places we will find that the Ethernet protocol is in use.

But of course there are installations you can find somewhere who had used some other protocol like say FDDI or say ATM. Some other protocol which is not Ethernet. But Ethernet is most common at the level of the network use IP. See this IP is important because in today's internet the whole internet infrastructure is sitting on top of IP. So if it is not IP, then your packet cannot reach your intended destination where you wanted to reach. So it is mandatory to have this IP infrastructure in place if you want to be part of the internet and with respect to the standard applications at the transport layer level you can use TCP; of course UDP is another option. But most of the application use TCP because of the reliability it provides at the application layer level. We have used a standard application program interface like Berkeley socket interface. Now in Berkeley socket interface actually what happens is that here you as a programmer you are provided with a set of application calls or depends a library functions.

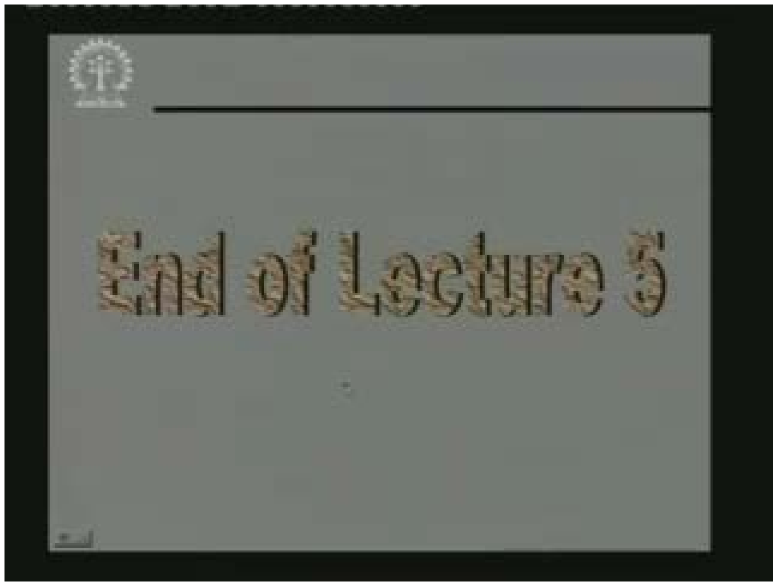
Now using this library functions you can do a host of things. For example you want to develop a connection oriented application. So you will have to first establish a connection between the two ends. Now establishment of the connection, there are such system called like socket bind accept and connect. So whenever a client wants to establish a connection with a server, these are the function calls which have to invoke at the two programs which are running on the two sides. So at the end of it the connection is established now. Now when I say a connection is established, it means that the all the 5 components of that association that I told you about the protocol local IP address, local port number, remote IP address, remote port number. So all this 5 things are known to both the ends of the communication. So ones this is so they can start the communication and whenever you open this kind of a connection. You have something called a socket.

Now a socket is very similar to file in terms of its usage. Now recall when you use a file say in C like program first we have to open that file; you get file descriptor and using the file descriptor you can do read and write from the file. Now here also a very similar things happens in the process of establishing a connection something called socket descriptor is returned. Now using the socket descriptor you can do a send or receive at the two ends if the client makes a sent there must be matching receive at the server end. So this send and receive go on as long as you need to it and at the end there will be another system called shutdown or close for closing or terminating the connection. But for connectionless application development, for example if you want to use UDP. There is no concept of a connection establishment there is nothing called socket. So there also you will be provided with certain system calls. But the system calls will be slightly different sent to or received from.

These are the names of the two system calls which are available now. Now in these functions you need to specify all the five components of the association as the parameters. So when you are sending a UDP packet to another process in another machine you have to specify all 5 components of the association as parameters. Now using those values you are passing as parameter the UDP packet is constructed and is sent. Similarly receive from will also get all the values from the other side and those values you can read and you can find out from where the packet as actually come. So using this Berkeley Socket Interface you can develop any kind of network application which is running on a client server environment over the network and if it is based on TCP and IP which Berkeley socket is so.

So this application can potentially run across any pair of hosts in the internet. For example my client may be located here in my room, the server may be located in some say means it may be located in US like when I browse the internet. I type in the name of the site. Now the corresponding server may be located anywhere in the world. But sitting here I am able to access directly this is because this application was developed on top of TCP and IP. This is a standard which was maintained and followed. Now we have talked about IP we have talked about TCP, we have talked about UDP. Now what would be talking in the next few lectures is that well understanding that today's network. Today's internet is based on IP. We want to have an better understanding that exactly how this IP packets are routed and forwarded through the internet. So this we shall be discussing in out next lecture. So here we come to the end of this lecture number 5.

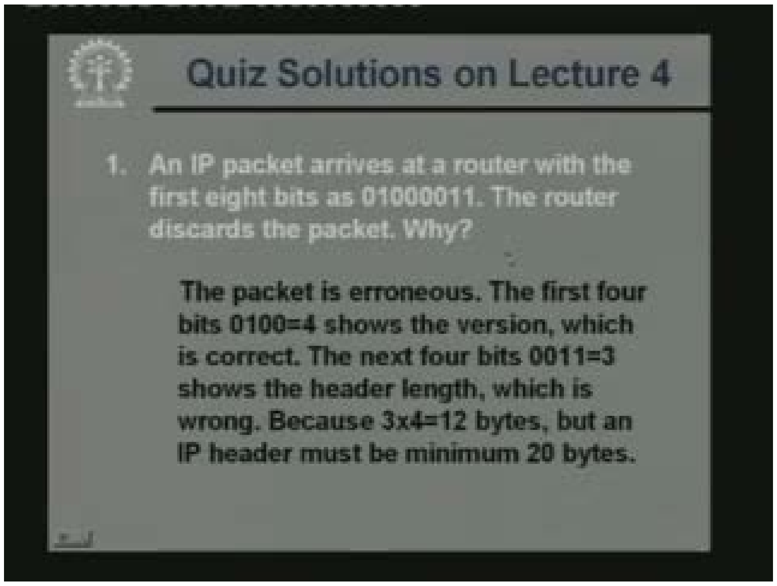
(Refer Slide Time: 44:51)



End of Lecture 5

But now let us have a quick look at the solutions to the question which we posed on our earlier lecture.

(Refer Slide Time: 45:05)



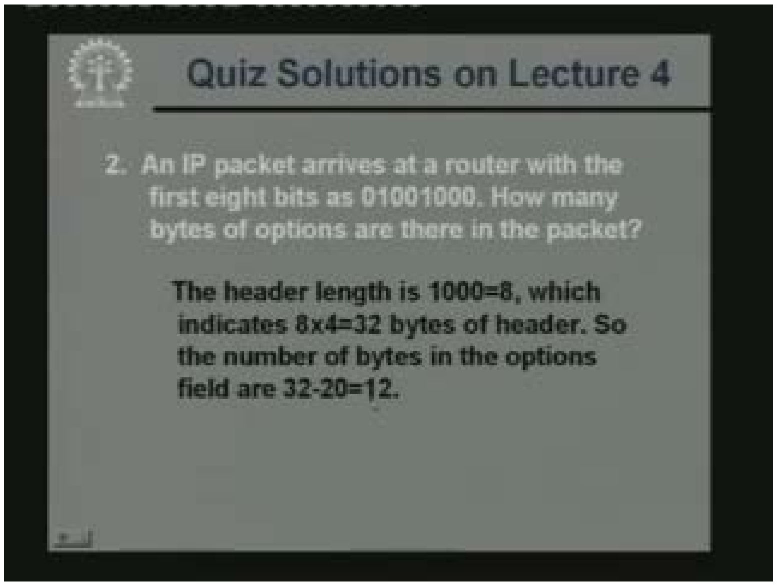
## Quiz Solutions on Lecture 4

1. An IP packet arrives at a router with the first eight bits as 01000011. The router discards the packet. Why?

   **The packet is erroneous. The first four bits 0100=4 shows the version, which is correct. The next four bits 0011=3 shows the header length, which is wrong. Because 3x4=12 bytes, but an IP header must be minimum 20 bytes.**

So the first question was an IP packet arrives at a router with the first 8 bits as 01000011 but the router discards the packet. Now you try to understand what the first 8 bits in a IP packet mean. Recall that the first field in a IP packet is the IP version the first 4 bits and the next 4 bits will represent the header length. So in this example the first 4 bits 0100 this will actually indicate the version which is version 4 and that is correct. The last 4 bits 0011, this indicates the number 3. Now 3 means this is the header length for the total header size will be 3 into 4 are 12 bytes. But this is an invalid number because you know that a IP header must be minimum 20 bytes which means the second field it cannot be 3. It must be at least five or more. So due to this invalid value of the header length field this IP packet well be discarded.

(Refer Slide Time: 46:30)



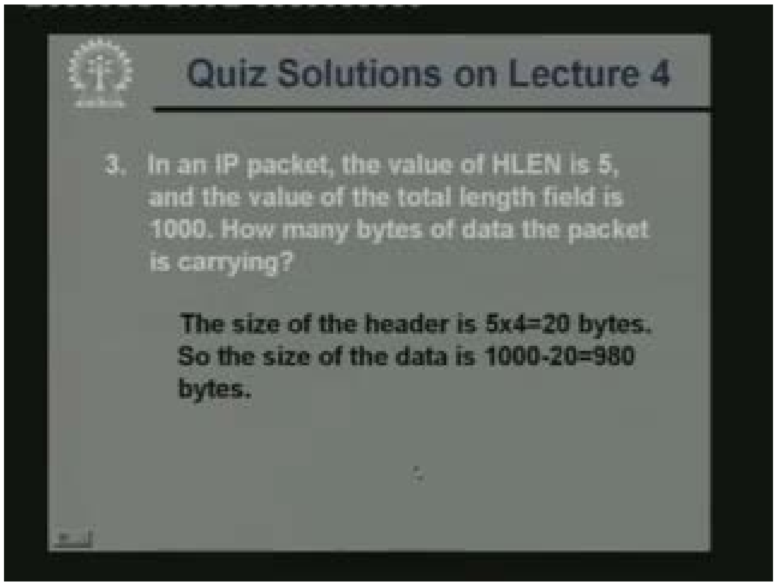
## Quiz Solutions on Lecture 4

2. An IP packet arrives at a router with the first eight bits as 01001000. How many bytes of options are there in the packet?

The header length is 1000=8, which indicates 8x4=32 bytes of header. So the number of bytes in the options field are 32-20=12.

Now the next question an IP packet arrives at a router again with the first 8 bits are given this combination. But the question is how many bytes of options field are there in the packet. Well here again the first 4 bits will indicate the version, the next 4 bit will indicate the header length. Now in the example the header length is 8. 8 mean this will indicate 32 bytes of header. But the basic IP protocol requires 20 bytes in the header and in this calculation. We are getting 32 so the difference between this 32 and 20 namely 12. This will be number of bytes in the optional options field in the IP header. So you calculate the actually number of bytes in the header subtract the minimum well of 20. You will get the size of the options field 12 bytes.

(Refer Slide Time: 47:37)



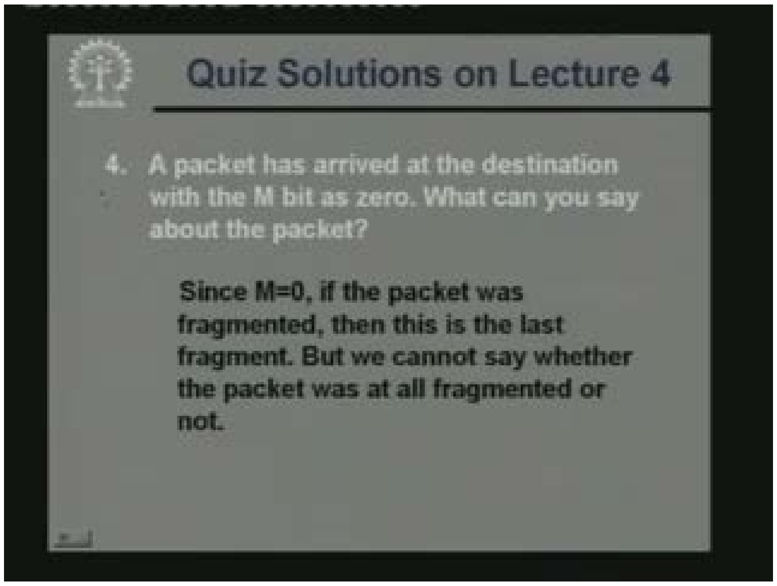
## Quiz Solutions on Lecture 4

3. In an IP packet, the value of HLEN is 5, and the value of the total length field is 1000. How many bytes of data the packet is carrying?

The size of the header is 5x4=20 bytes. So the size of the data is 1000-20=980 bytes.

In an IP packet the value of the header length is 5 and the value of the total length field is 1000 decimal. How many bytes of data the packet is carrying. Well since the header length value is 5, this means the size of the header will be 20 bytes. Now the total size of the IP packet is 1000 bytes. So out of this 1020 bytes are going for the header. So the remaining 980 bytes will be data fine.

(Refer Slide Time: 48:17)



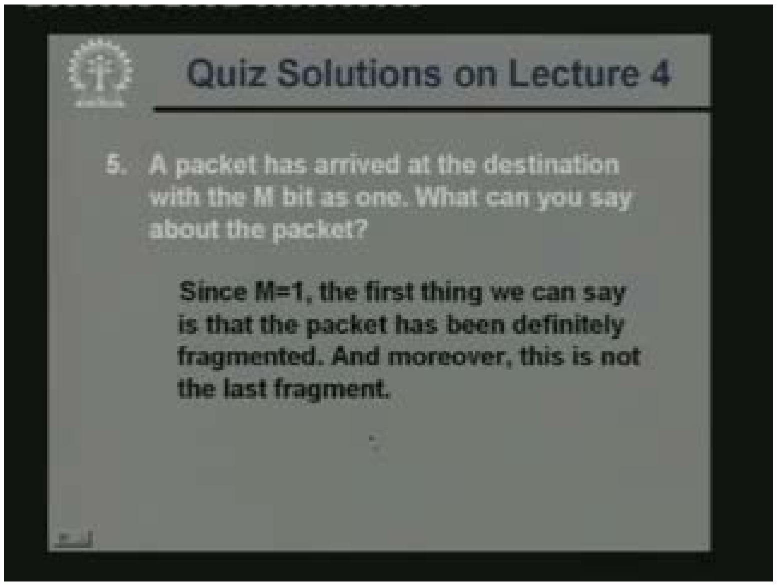
## Quiz Solutions on Lecture 4

4. A packet has arrived at the destination with the M bit as zero. What can you say about the packet?

Since M=0, if the packet was fragmented, then this is the last fragment. But we cannot say whether the packet was at all fragmented or not.

4. A packet has arrived at the destination with the M flag bit as 0. What can you say about the packet? Well here you recall that the n bit in the packet is used for the purpose of fragmentation reason which is a more fragmented if M is equal to 0. It means that this is not the last fragment of the original packet. There are more which follows, so if the M bit is 0 which means that there are no more fragments following this. So here there are 2 alternatives if the packet was fragmented. Then this is the last fragment because M is 0. But if the packet was not fragmented if it is the original packet in itself it will anyway have the M bit has 0. So if the M bit is equal to 0, we cannot say whether the packet was fragmented or not. What we can say is that if the packet was fragmented then this is the last piece of the fragment. But you cannot say whether it was fragmented or not.

(Refer Slide Time: 49:32)



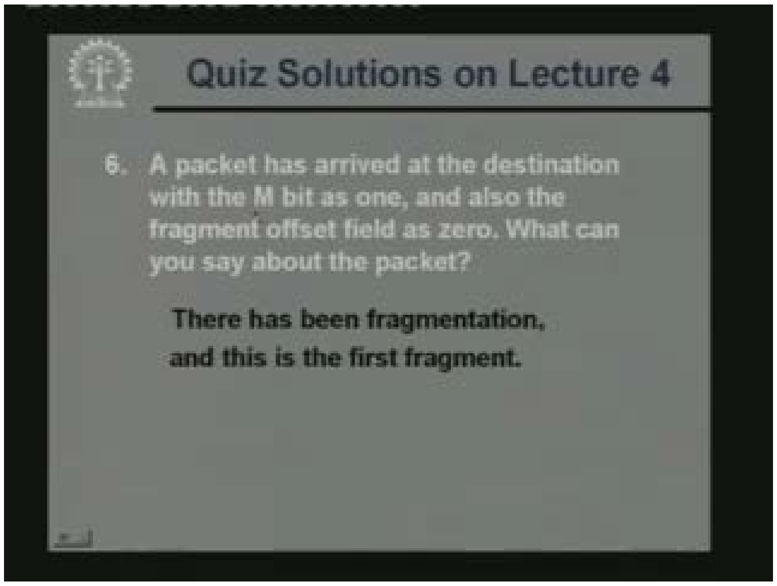
## Quiz Solutions on Lecture 4

5. A packet has arrived at the destination with the M bit as one. What can you say about the packet?

   Since M=1, the first thing we can say is that the packet has been definitely fragmented. And moreover, this is not the last fragment.

Now next question says that the M bit is 1, so here what can you say? Well M equal to 1 means this is a fragment because a normal packet can never have the M bit set to 1. M equal to 1 means there are more fragments following. So the packet must have been fragmented and this is not the last of the fragments. These are the two things we can say so the packet has been definitely fragmented and this is not the last fragment these are two things we can say.

(Refer Slide Time: 50:22)



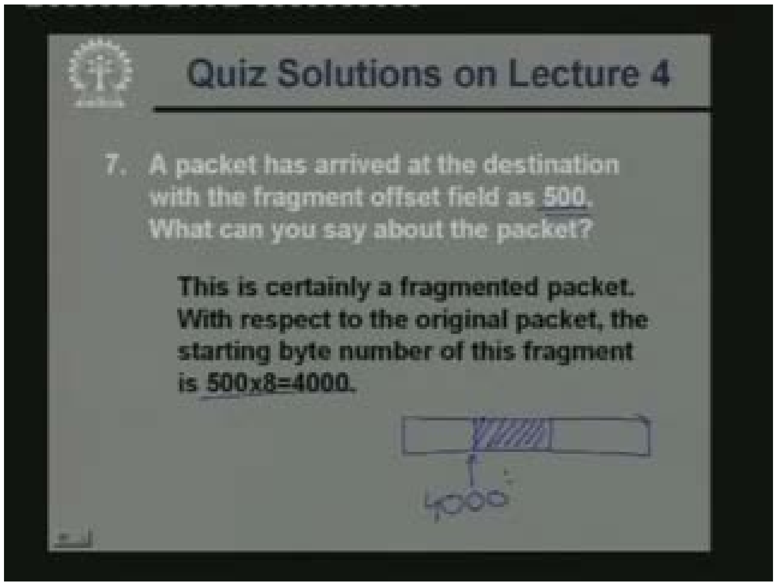
## Quiz Solutions on Lecture 4

6. A packet has arrived at the destination with the M bit as one, and also the fragment offset field as zero. What can you say about the packet?

   There has been fragmentation, and this is the first fragment.

So a packet is arrived with the M bit as one and also the fragment offset field as 0. What can you say well a fragment offset field as 0 means this is the first fragment M equal to 1 means that it has been fragmented definitely. So you can say that there has been fragmentation and this is the first fragment next one.

(Refer Slide Time: 50:51)



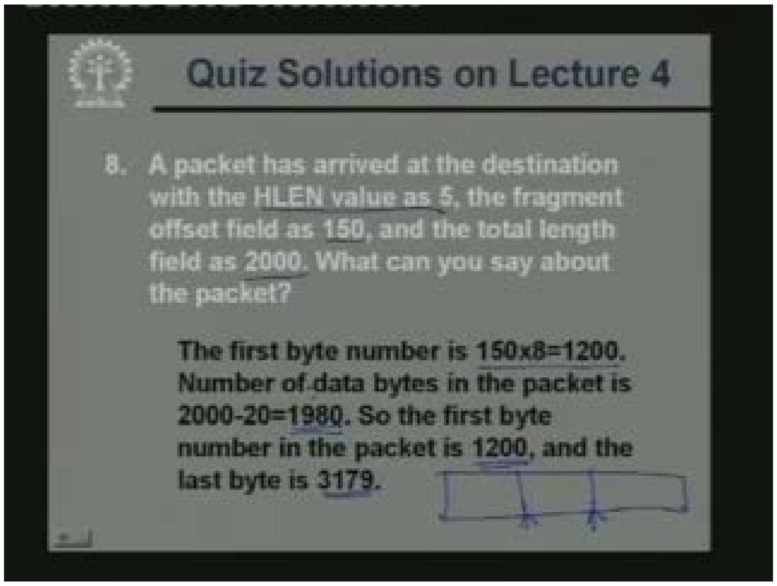
## Quiz Solutions on Lecture 4

7. A packet has arrived at the destination with the fragment offset field as 500. What can you say about the packet?

   **This is certainly a fragmented packet. With respect to the original packet, the starting byte number of this fragment is 500x8=4000.**

This says that the fragment offset field is 500 then what can you say? Well fragment offset is 500. Means that you recall the fragment offset field is specified in multiples of 8 bytes. So with respect to the original message if this is the original message and suppose the present fragment accords here in the original one, then this offset value will be 500 into 8 or 4000. So the starting byte number of the fragment with respect to the original packet will be 4000.

(Refer Slide Time: 51:28)



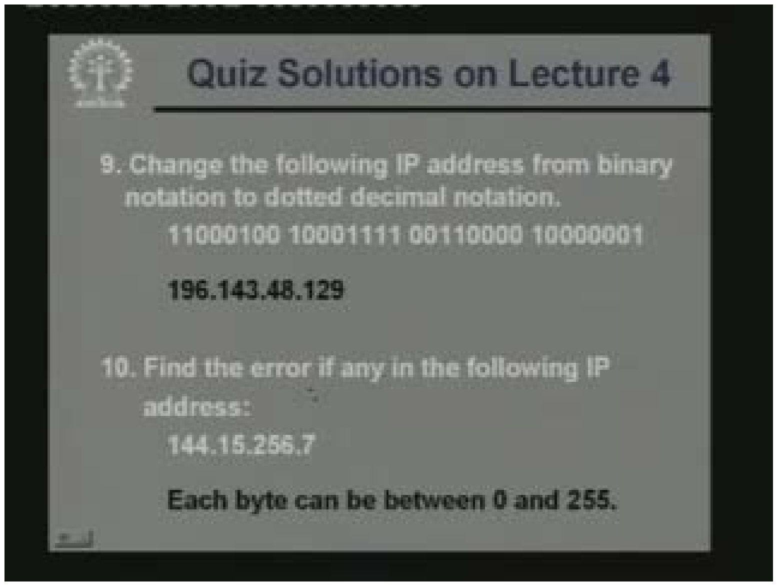
## Quiz Solutions on Lecture 4

8. A packet has arrived at the destination with the HLEN value as 5, the fragment offset field as 150, and the total length field as 2000. What can you say about the packet?

   **The first byte number is 150x8=1200. Number of data bytes in the packet is 2000-20=1980. So the first byte number in the packet is 1200, and the last byte is 3179.**

Now a packet has arrived at the destination with the header length as 5 fragment offset 150 and the total length field has 2000, then what you can say. Well since the fragment offset is 150 we can say that the first byte number is 150 into 8, 1200. The header length is five means 20 bytes of header, the total length is 2000. So you can say 1900 and 80 so many data bytes are there. Now since the first byte number is 1200, so the first byte in the packet corresponds to 1200, the last byte corresponds to 1200 plus 1979 which is 3179. So with respect to the original packet, the present fragments beginning byte position will be 1200, the last byte position will be 3179, this is what we can say.

(Refer Slide Time: 52:29)



## Quiz Solutions on Lecture 4

9. Change the following IP address from binary notation to dotted decimal notation.

   11000100 10001111 00110000 10000001

   **196.143.48.129**

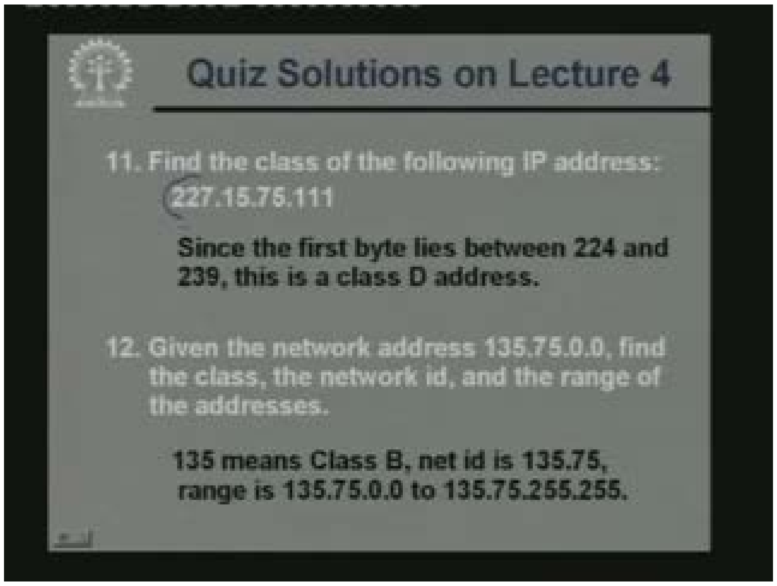
10. Find the error if any in the following IP address:

    144.15.256.7

    **Each byte can be between 0 and 255.**

Change the following IP address from binary to dotted decimal, this is easy you take each of these 8 bit chunks covert them to decimal and write the numbers separate by dots. Find the error in the following IP address. Well these decimal numbers correspond to 8 bit binary strings. Now an 8 bit binary number can have a value unsigned value from 0 to 55. Now this IP address notation has a number 256 which is invalid so this is not a valid IP address.

(Refer Slide Time: 53:10)



## Quiz Solutions on Lecture 4

11. Find the class of the following IP address:
    227.15.75.111

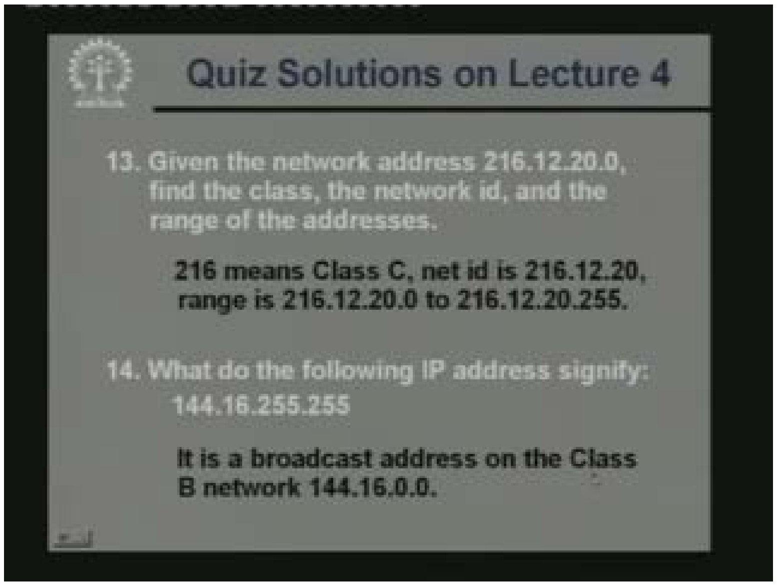
    **Since the first byte lies between 224 and 239, this is a class D address.**

12. Given the network address 135.75.0.0, find the class, the network id, and the range of the addresses.

    **135 means Class B, net id is 135.75, range is 135.75.0.0 to 135.75.255.255.**

Find the class of the following IP address. Well because it starts with 227, you can easily see it is class D address. Given the network address 135.0.0 find the class network id and the range. Since it starts with 135 it is class B. The network id for the class B are two first octets 135.75 and the range will be for the host part we apply all combinations starting from all zero upto all ones this is the range.

(Refer Slide Time: 53:50)



## Quiz Solutions on Lecture 4

13. Given the network address 216.12.20.0, find the class, the network id, and the range of the addresses.

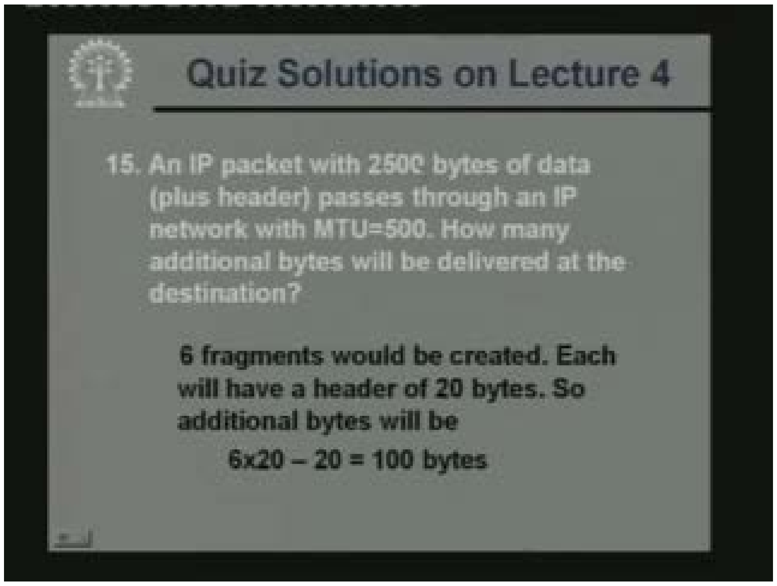
    **216 means Class C, net id is 216.12.20, range is 216.12.20.0 to 216.12.20.255.**

14. What do the following IP address signify:
    144.16.255.255

    **It is a broadcast address on the Class B network 144.16.0.0.**

Given the network address this find the class network id ranges same. Now 216 means class C. Now for the class C the first 3 octets will be the network id and the range the last one can be starting from all zero upto all ones this is the range. What do the following IP addresses signify? Well 144.16 means a class B address 255 in the host part means it is a broadcast address. That means this address means the broadcast all hosts on that particular class B network, this is a directed broadcasted address.

(Refer Slide Time: 54:34)



## Quiz Solutions on Lecture 4

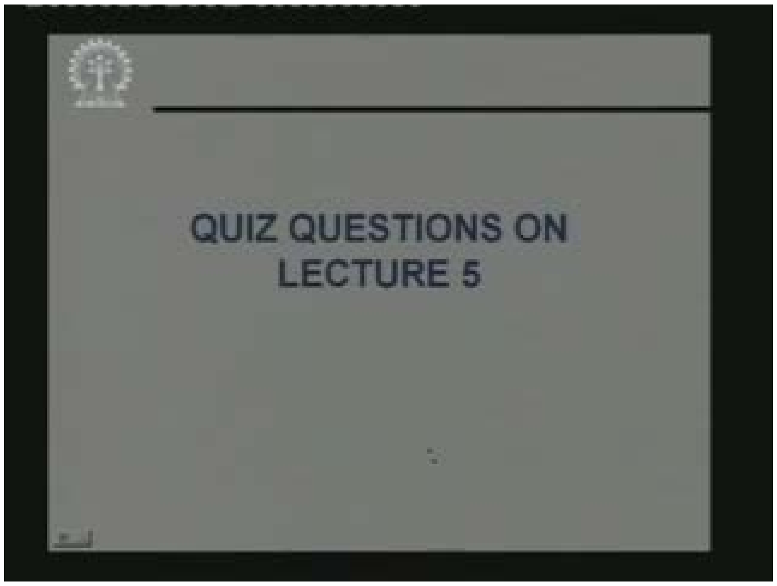
15. An IP packet with 2500 bytes of data (plus header) passes through an IP network with MTU=500. How many additional bytes will be delivered at the destination?

**6 fragments would be created. Each will have a header of 20 bytes. So additional bytes will be**

**6x20 – 20 = 100 bytes**

Well an IP packet with 2500 bytes of data plus header passes through an IP network with MTU of 500. How many additional bytes will be delivered now, 2500 header plus header will become 2520. Now it passes through an IP network MTU 500 you can easily see that a total of 6 fragments will get created. Now each of this IP fragments will be carrying a 20 bytes of header. So in the final destination 6 such IP packets will be delivered with a total of 120 bytes of header. But the original packet had only 20 bytes of header. So 100 bytes of extra header information will be delivered.

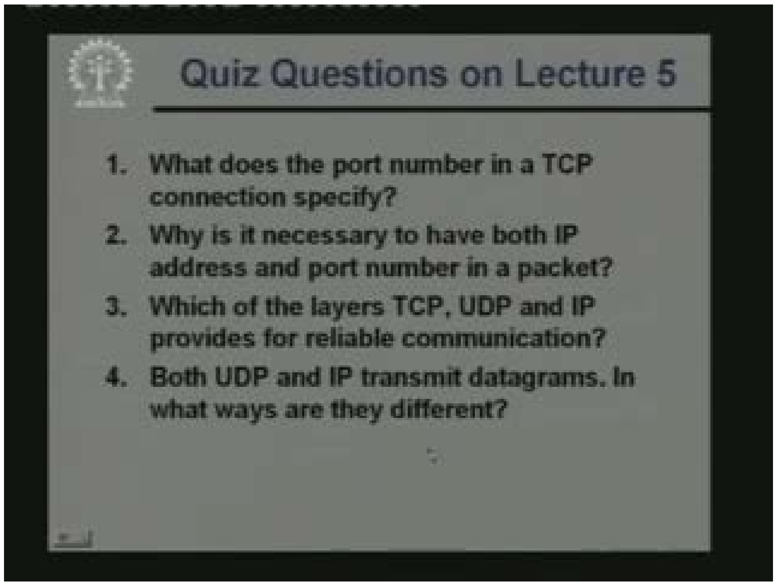
(Refer Slide Time: 55:19)



QUIZ QUESTIONS ON LECTURE 5

So now very quickly let us look at some of the questions from today’s lecture.

(Refer Slide Time: 55:26)

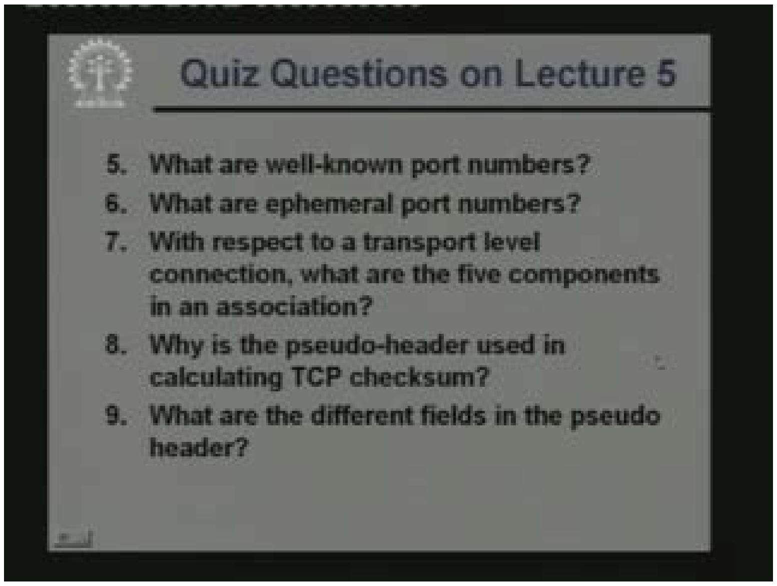


## Quiz Questions on Lecture 5

1. What does the port number in a TCP connection specify?
2. Why is it necessary to have both IP address and port number in a packet?
3. Which of the layers TCP, UDP and IP provides for reliable communication?
4. Both UDP and IP transmit datagrams. In what ways are they different?

What does the port number in a TCP connection specify?
Why it is necessary to have both IP address and port number in a packet?
Which of the layers TCP UDP IP provides for reliable communication both UDP and IP transmit datagrams? In what ways they are different?

(Refer Slide Time: 55:50)

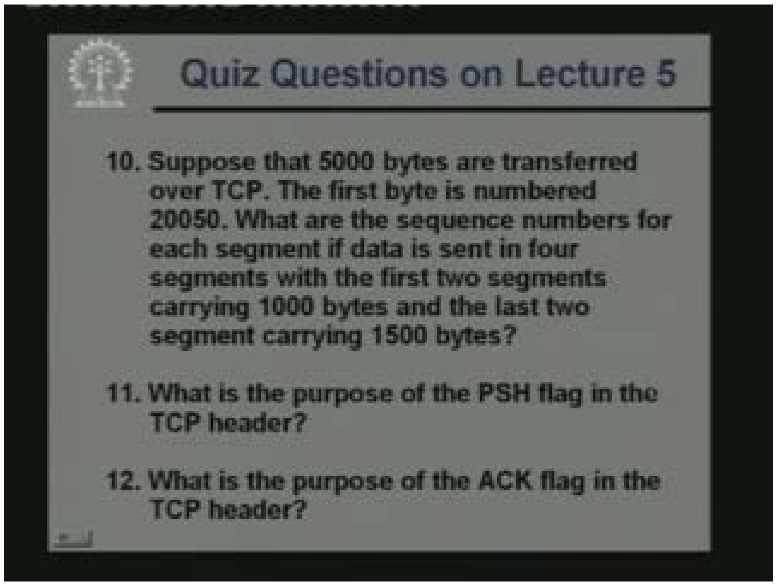


## Quiz Questions on Lecture 5

5. What are well-known port numbers?
6. What are ephemeral port numbers?
7. With respect to a transport level connection, what are the five components in an association?
8. Why is the pseudo-header used in calculating TCP checksum?
9. What are the different fields in the pseudo header?

What are well known port numbers?
What are ephemeral port numbers?
With respect to a transport level connection what are the five components in association? This we have discussed.
Why is the pseudo-header used in calculating TCP checksum? What are the different fields in the pseudo header?

(Refer Slide Time: 56:12)



## Quiz Questions on Lecture 5

10. Suppose that 5000 bytes are transferred over TCP. The first byte is numbered 20050. What are the sequence numbers for each segment if data is sent in four segments with the first two segments carrying 1000 bytes and the last two segment carrying 1500 bytes?

11. What is the purpose of the PSH flag in the TCP header?

12. What is the purpose of the ACK flag in the TCP header?

Well here is a question on TCP. Suppose that 5000 bytes are transferred over TCP, the first byte is numbered 20050, what are the sequence numbers for each segment if data is

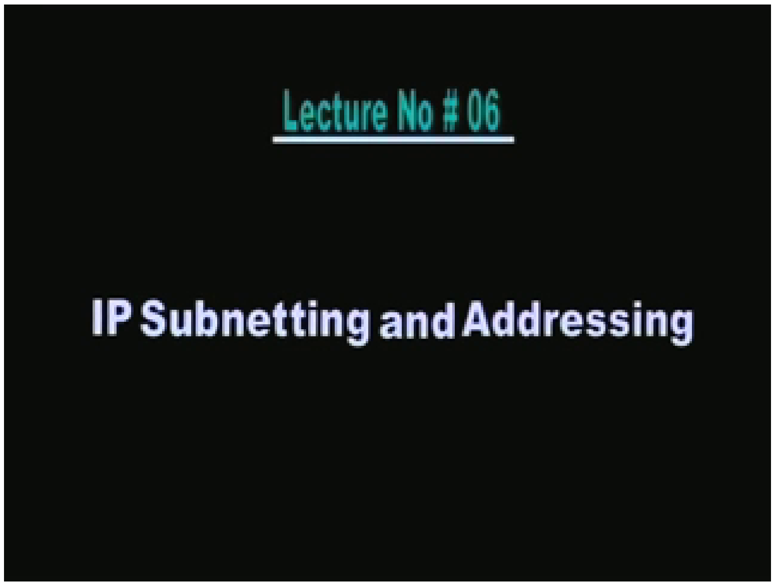
sent in four segments with the first two segments carrying 1000 bytes and the last two segments carrying 1500 bytes of the data?
Now the first byte number is 20050 means the numbering will start from there 20050, 20051, 20052. This all this 5000 bytes that have been sent they will be numbered accordingly. They will be split into 4 different TCP segments. So in this question we will have to say that in this four segments how will be the byte numbing taking place from the starting address the ending address this will have to find out.
What is the purpose of PUSH flag in the TCP header?
What is the purpose of the acknowledgement ACK flag in the TCP header?
If you are developing a network application on a reliable learn environment which of TCP or UDP would you prefer and why?
So these are questions from today's lecture. So as I had mentioned in our next lecture we shall be talking about something related to routing in IP networks. How the data packet actually flows through an IP network and finally finds their way to the ultimate destination. Thank you.

(Refer Slide Time: 57:59)

Preview of next lecture

Preview of next lecture.

(Refer Slide Time: 58:01)



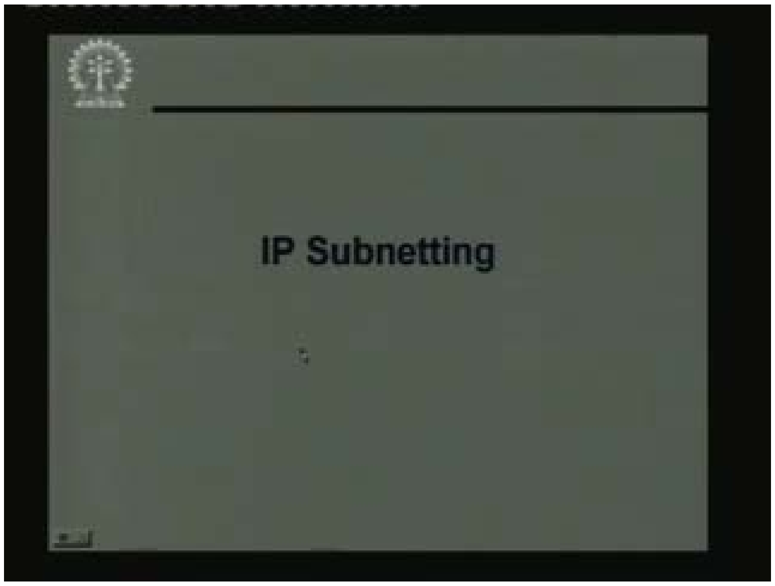
Lecture No # 06

IP Subnetting and Addressing

IP Subnetting and Addressing.

So we will start our discussion on IP addressing and routing. So if we recall what we have discussed in our last few lectures, we had looked at the TCP/IP protocol suite. We had looked at the basic functionalities of the IP, TCP and UDP protocols. Specifically we have mentioned that the IP protocol is responsible for the delivery of a packets from a source to the destination through a number of intermediate nodes which are typically routers. So we shall today look at some more details about how this addressing the level of IP is achieved. So IP Subnetting and Addressing is the topic of our discussion today.

(Refer Slide Time: 58:59)



IP Subnetting

So we start with something called IP Subnetting.