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

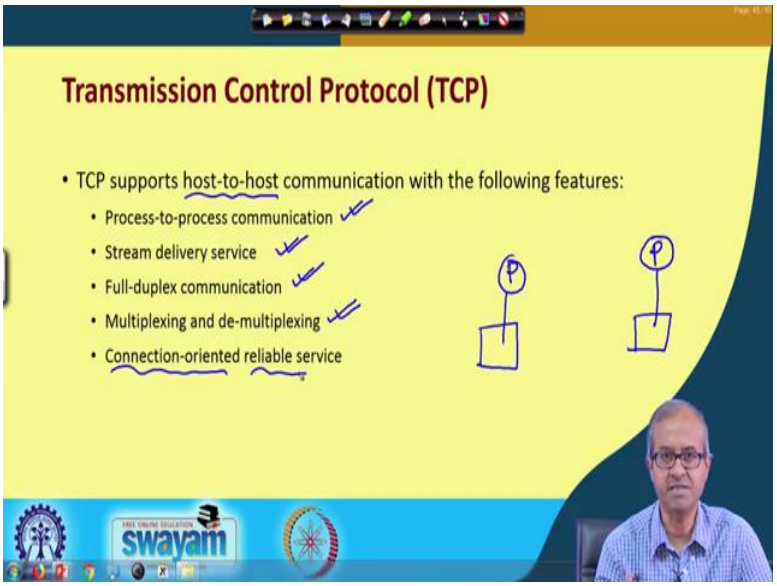
Lecture - 09
TCP and UDP (Part II)

We continue with our discussion on the TCP and UDP protocols in this lecture as well.

(Refer Slide Time: 00:25)

Here in the second part of this lecture, we shall be looking at the actual header fields of both the TCP and UDP protocols. And, in particular we shall be looking at the process of TCP connection establishment and also connection termination. Because, these are a few very interesting things which can lend itself to some kind of network based attacks that we shall be talking about ok.

(Refer Slide Time: 00:54)

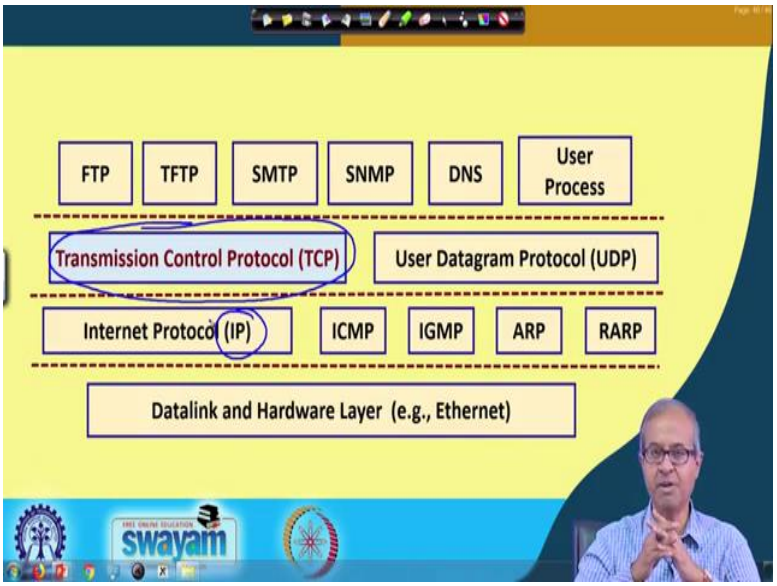


So, let us start with TCP. TCP we have already mentioned that at the transport layer level, we have a host to host connection facility. So, TCP supports host to host communication facility and it supports a set of features. The features are like these 2 processes can communicate among themselves, a process on a local machine and a process on a remote machine.

There can be a process here, there can be a process here, they can communicate. Stream delivery service, the data that have been transmitted, these are called segments, they are considered to be as a stream of bytes, a stream of bytes are sent, a stream of bytes are received at the other end, this is said to be a stream oriented service.

Full duplex means both way communication, two way communication. Multiplexing and demultiplexing means multiple active connections may be present between the two ends and multiple packets for multiple connections can flow through the same links and same logical links, these are called multiplexing and de-multiplexing. And TCP ensures connection oriented and reliable service, this briefly we have mentioned earlier, that there are a few features using error control retransmission. So, it tries to ensure this kind of features.

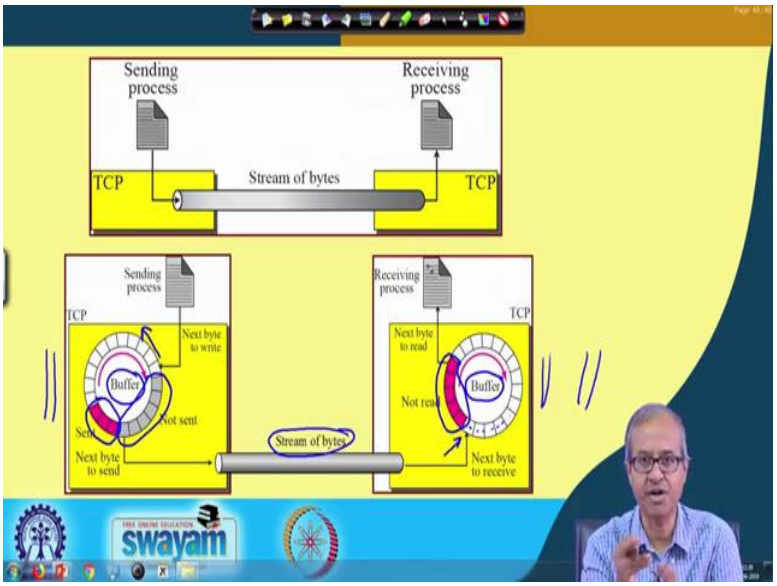
(Refer Slide Time: 02:53)



Now, you look at the whole TCP/IP protocol family and you can see that TCP sits here, TCP is here. Below TCP, there are several protocols at the network layer level, where IP is the dominant protocol. And, there are a few sister protocols which are there, which have very specific purposes. So, the point to note, I am just repeating, so, although TCP provides a connection oriented reliable service to the applications at the higher layer, but all data transmission has to take place via IP and IP is a unreliable datagram service.

So, TCP has to do a lot of bookkeeping management, buffering, so that all this error correction and streaming and these reliable facilities can be provided to the application.

(Refer Slide Time: 04:04)



Now, let us look at these diagrams. This first diagram actually shows you the conceptual view of TCP. This is a sending process; this is a receiving process, on two different computers. This is the TCP layer on this and there is the TCP layer on this. So, TCP provides a host to host logical connection, this is not a physical connection, a conceptual connection. So, you have this pipe that has been shown here, this represents the logical connection. And, over this logical connection, a stream of bytes are flowing, this is how a TCP communication system looks like.

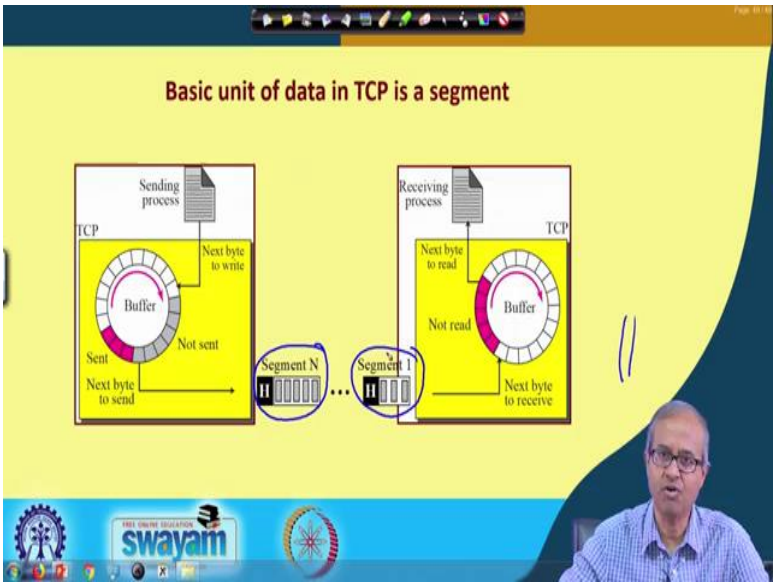
Now, let us look into some more detail inside TCP which is shown in the next diagram, you see both the sender and the receiver they maintain some kind of a buffer. Buffer at the sender side will contain the message that it is trying to send. It is a stream of bytes, the receiver also will maintain the buffer whatever comes it will put in the buffer. And, both the sides are keeping track of how many bytes have been sent, how many bytes have been received and so on and so forth.

Like you see in the sender process, there are some bytes marked in pink, which indicates the bytes which have been sent, already sent. The bytes marked in gray indicates which have not yet been sent and the process when it is generating some data to be sent, it will be storing them into the buffer from the next available location onwards. In this way the buffer at the sending side will get filled up. And, on the receiving side similarly some data

is being received, this pink one says that this data have been received, but not yet read by the receiving process and whatever comes over the network, they will get stored out here.

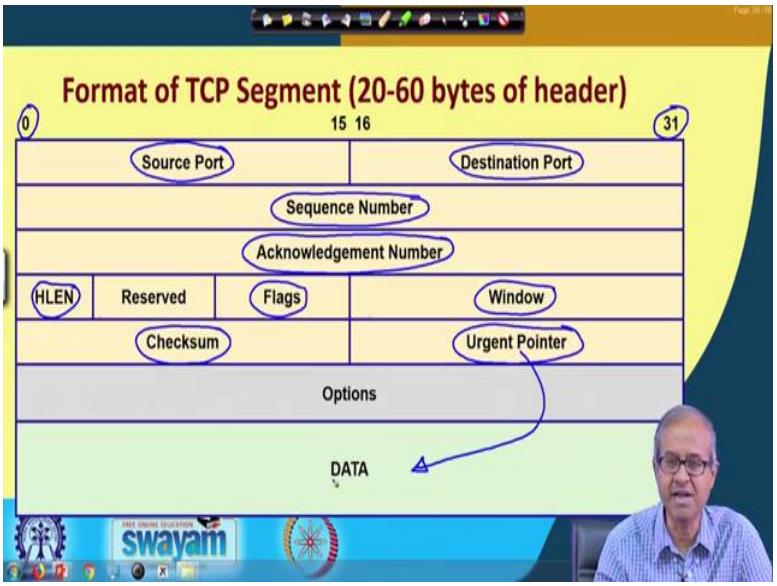
Starting from the next available location onwards and the receiving process will be reading the bytes one by one from this pink part right. Now, TCP maintains this buffer automatically, you see due to some error in between it may see that somewhere in between there is a vacancy, all bytes have not yet been received. So, there will be an acknowledgment mechanism, the receiver will be requesting the sender to send back the missing bytes again, so that the receiving process can be forwarded the bytes in the correct order ok.

(Refer Slide Time: 07:27)



And, the other point we note is that which is just depicted in this diagram, a similar diagram. It shows some basic data transfer unit called segments. You see at the network layer level, we talk about packets are being transferred. At the data link layer level, we talk about frames; frames are being transferred, but at the transport layer level, at the TCP level we talk about segments, data segments, which are nothing, but stream of bytes, multiple segments are being transmitted one after the other, this is the concept.

(Refer Slide Time: 08:13)



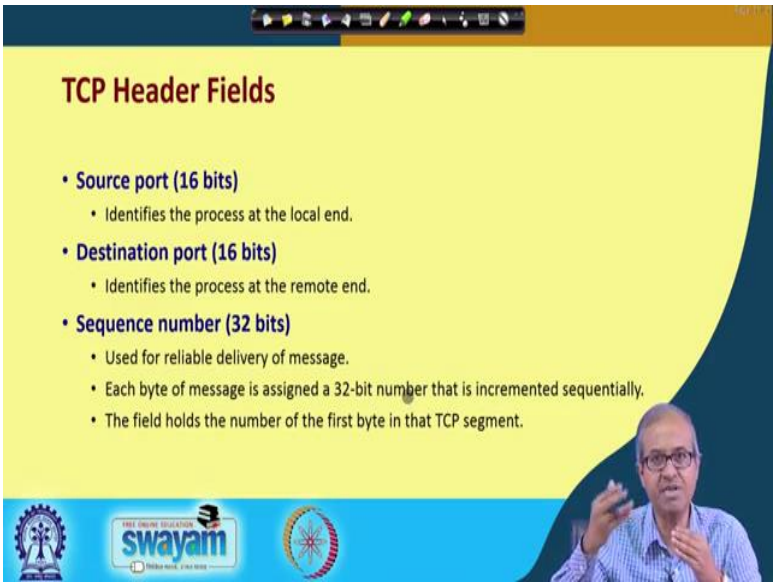
Now, let us look at the header format of a of the TCP protocol. Just like IP, I am showing this bit numbers on top starting from 0 up to 31, which indicates that each row of this indicates a 32 bit quantity, 4 bytes. So, first you have source port number, destination port number 16 bits each. Then, we have a sequence number, well sequence number means I told you that data are being sent in units of bytes, stream of bytes, sequence number is like a byte number which byte number is being sent. This is a 32 bit quantity.

So, which means maximum segment size can be 2^{32} quite large. Similarly, acknowledgement number is also 32 bits, acknowledgement number is coming from the other side, it tells the receiver, tells the sender that well I have received up to byte number so, and so, that is the acknowledgement number. So, that the sender will know that well up to this byte number is already being received so, I need not keep it in my buffer anymore, let me remove it from my own buffer ok.

Then, there are some other fields like, there is a header length, just like TCP, there some reserved field, there are some flags, we shall talk about, there is a window, well window is like that buffer I told you, you know some data are being send, some data are being received. So, there is some windows that are maintained by both sender and receiver, this is called a sliding window protocol, but I am not going into detail of this. So, this window specifies this so called sliding window and is used for flow control.

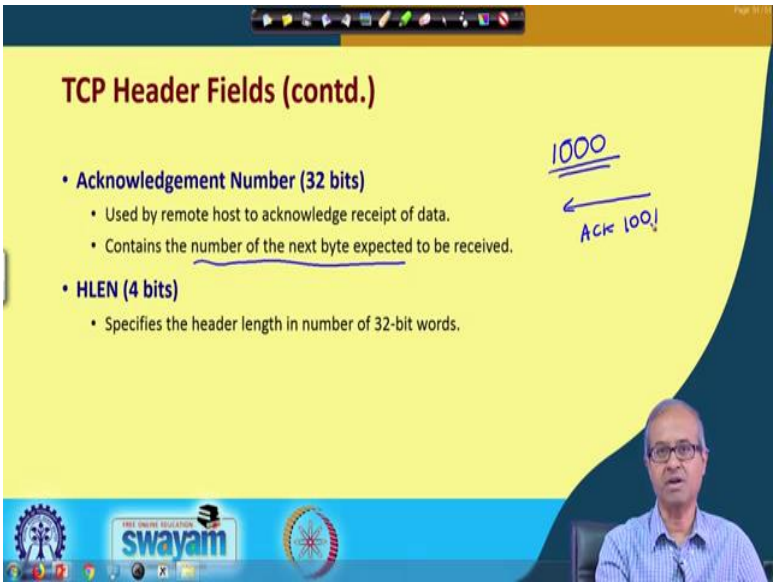
Flow control means the maximum speed of data transmission you can control by changing the size of this window ok. And of course, there is a checksum for correcting errors, detecting errors and there is an urgent pointer, sometimes as part of the TCP packet, you can send some urgent message, an urgent pointer actually points to some part of the data where that urgent message is located and some option field is also there. So, the TCP header is typically minimum 20 bytes and can go up to 60 bytes.

(Refer Slide Time: 11:14)



So, a brief explanation, source port already I have mentioned, destination port, this identifies the process at the local end and at the remote end. Sequence number, I am just repeating, this is used for keeping track of the bytes being transmitted for the purpose of reliability. Each byte is assigned a 32-bit number and as the bytes are being transmitted that number is incremented by 1, 1, 1, like that fine.

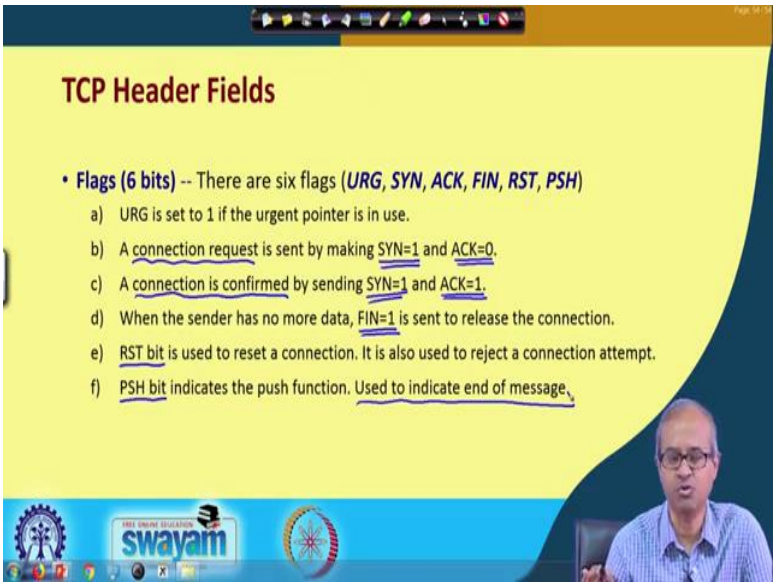
(Refer Slide Time: 11:49)



Similarly, from the other side acknowledgement number is coming, the remote host will acknowledge receipt of the data; it will contain the number of the next byte expected. Like say, suppose the receiver has correctly received up to byte number 1000, let us say. So, it will be sending back an acknowledgment saying acknowledgment 1001.

So, it will be sending back an acknowledgment packet with the acknowledgement number field as 1001 ok. Now, header length is a 4 bit field, it just similar to the IP and the IP header also has a header length field, it specifies the header length in multiple of 32 bit words ok. So, minimum value is 5, because minimum header size can be 32, can be 20 bytes. So, it can be 5 minimum.

(Refer Slide Time: 12:59)



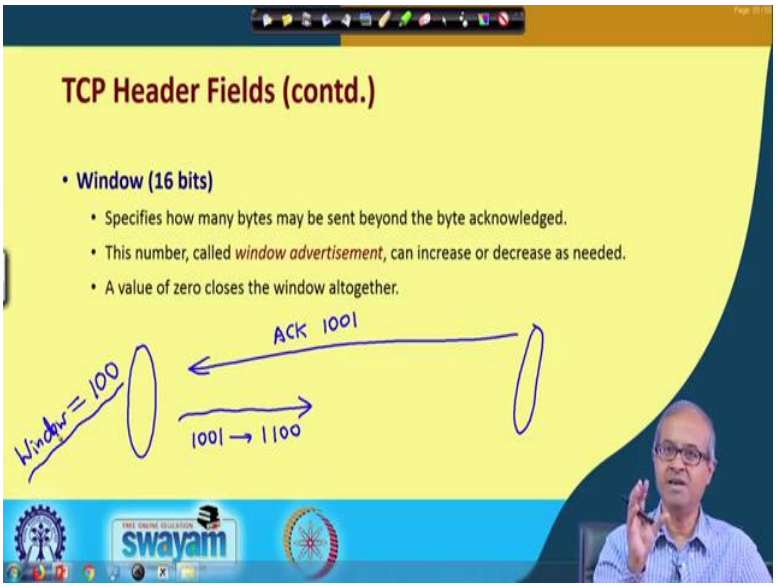
Well, these are interesting, there are several flags, in the flag field, in TCP there are in fact, 6 flags URG, SYN, ACK, FIN for finish, RST and PSH. They are used in combination for various reasons. The 6 of the most important reasons are mentioned here, well if there is any urgent data, that is carried by the TCP, urgent data may be some kind of interrupt processing, some important thing need to be handled.

So, then the urgent flag will be set to one and I mentioned the urgent pointer, there is an urgent pointer field, it will be pointing to some place in the data part that urgent data will be stored there. So, if the URG pointer is set to 1, it means that the urgent pointer is now active. In TCP whenever you are doing a connection establishment, you are requesting for a connection then you set this SYN flag to 1 and ACK flag to 0, this this combination means connection request. Similarly, connection confirmation I will show the details later.

So, when a connection is confirmed then the SYN and ACK both are set to 1 right and during sending if the sender has no more data, it has finished sending the data, then the finish flag will be set to 1, so that the connection can be released. So, when the connection has to be terminated, this FIN flag is set to 1. And, sometimes you may need to reset a connection due to some problem, then RST bit is used for this purpose. Some connection attempt, you do not want to accept that connection, then also you can send the RST bit, the connection will be rejected.

And, there is the last bit, PSH bit this indicates a push function, this can also be used to indicate end of message. So, when a message is finished, you can set the PSH flag to 1, it will tell the receiver that there is no more bits in the message ok.

(Refer Slide Time: 15:35)



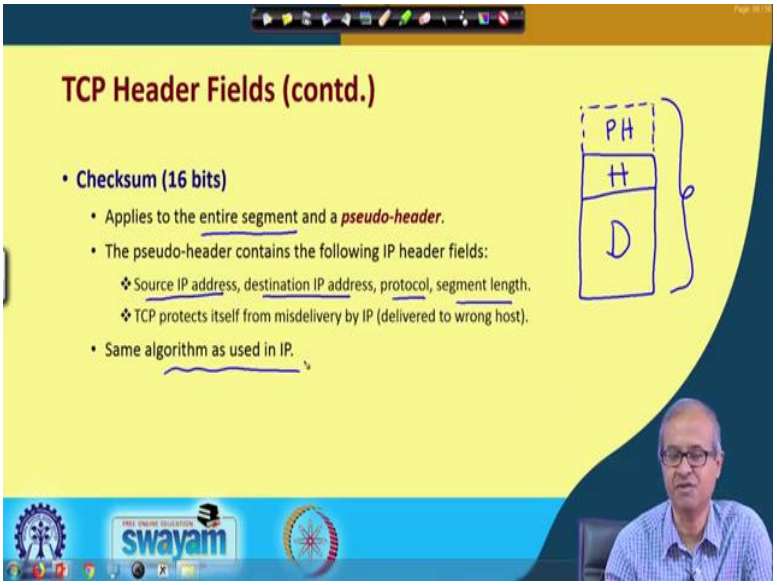
So, these are the flags and the ways they can be used. Window is a 16 bit field; now this field actually tells you how many bytes you can send, beyond the bytes acknowledged like let me given example. This is the sender, this is the receiver, sender is sending bytes one by one, suppose the receiver has sent back an acknowledgement. It has said that well I have received all bytes up to 1000, next byte I am expecting is 1001 ok.

Now, for the sender let us say the window field is 100, let us take an example. This means that the sender can now start sending byte numbers starting from 1001 up to 1100 without waiting for an acknowledgment. This window size actually tells how many bytes the sender can send without getting the acknowledgment. Once the window is full, finished, all 100 bytes have been sent, then the sender will be waiting an idle.

It will be now waiting for the acknowledgement to come back. So, as you can see if the window size is larger, you can send more data and wait less and then wait for the acknowledgement to come, but if the link is slow then possibly window size will be made smaller. So, by adjusting the size of the window, the rate of transmission of data can be controlled. So, you can use something called flow control, using this window, and the

window size is referred to as window advertisement. As, I mentioned it can be increased or decreased.

(Refer Slide Time: 17:46)

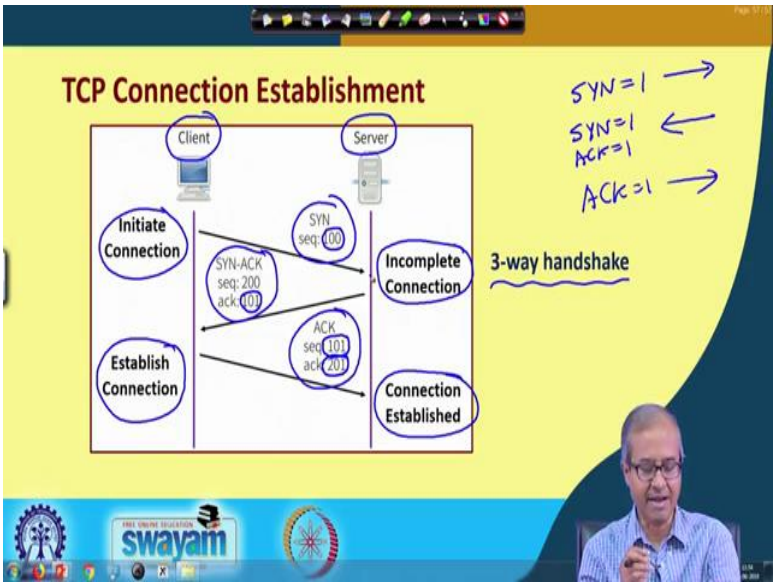


And finally, there is a checksum; checksum is used for error correction and this checksum is applied not only for the header, but for the entire data segment. Like in the TCP protocol, you have the header, you have the data and there is also something called a pseudo header. This checksum is applied to the entire part of it.

What does the pseudo header contains? Pseudo header borrows some of the fields from the underlying IP, that is not strictly part of the TCP header, but it borrows something from IP like, source IP address, destination IP address, which protocol, it is using TCP or UDP and what is the length size? right.

Because, this information are also important so that TCP can verify that whether the packet that has come, the message that has come, is actually coming to the right IP address or not. So, just for an additional level of checking, this pseudo header is maintained and for computing checksum, that same 1's compliment just add up. And, then take 1's complement of the result, same algorithm as is used for IP checksum is used here also.

(Refer Slide Time: 19:24)



Now, let us look very quickly at TCP connection establishment process, this is interesting, you see here there is a client and client is trying to establish a TCP connection with a server. There is something called a 3-way handshake protocol which is followed here.

The first step is to initiate the connection what it does? The client will send a packet with the SYN flag set; first this SYN flag is set to 1 and a packet is sent. Well here some sequence number will be carried with the packet, let us say 100 as an example. So, when this server receives this, initiate connection request, the connection is still not completed, this is incomplete connection.

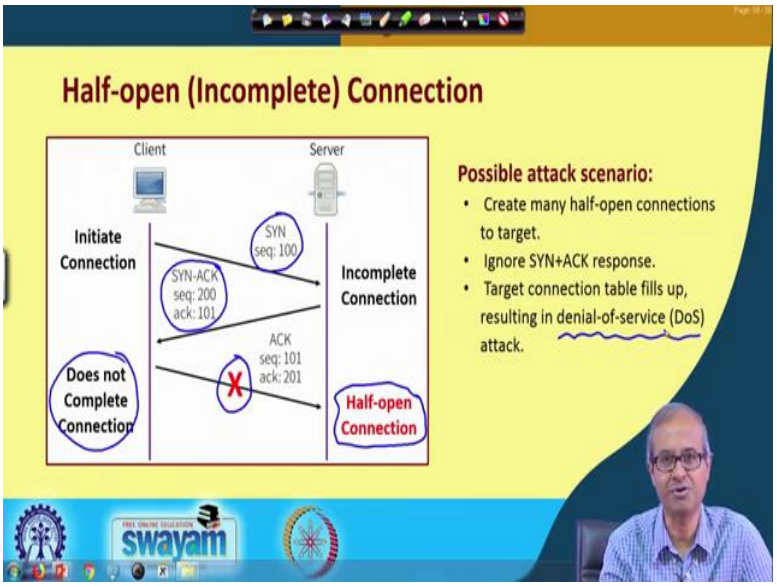
Server responds by sending a SYN-ACK packet. So, this is going in this direction SYN-ACK means SYN is also 1, ACK is also 1; both these flags are 1, this will be going in this direction. And, the sequence number will be from the server to client side, let us say this is 200 and the SEQ was 100. So, now, this acknowledgement number will be 101, next one. So, after receiving this client will now know that well yes now the connection can be established and what is done finally, is the client sends back the last packet, it is an ACK packet.

So, the final acknowledgement ACK equal to 1, here this sequence numbers was 200 was received. So, now, ACK will be 201 and previous packet was carrying sequence number 100, now this will be sequence number next 101, let us say. So, when server receives this, connection is finally established. So, the idea is that when the server receives the first

packet, it gets ready for the connection that well there is the connection request, let me keep aside some buffer, some entry in the table, so that this connection can be taken care of, once my 3-way handshaking is done.

So, server sends back and client finally sends back again, then only, after this 3 way handshake the connection is said to have been established ok. This is how TCP connection is established?

(Refer Slide Time: 22:16)



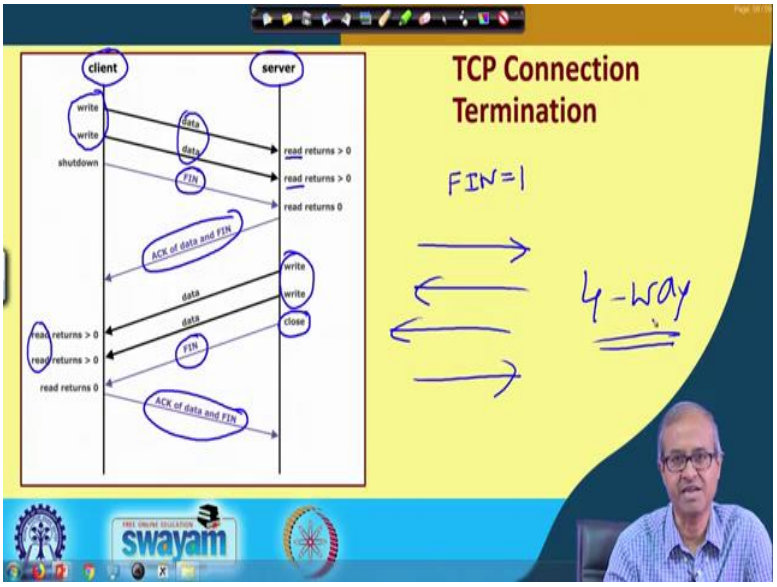
But, look at a scenario like this which we refer to as half open or incomplete connection. Well that same diagram I am showing, the client is sending a request, server gets that request. So, it gets ready for the connection, it sends back a SYN-ACK, but what the client does, well either deliberately or otherwise accidentally it does not complete the connection, means it either does not send back the final acknowledgment or it had sent, but due to some problem, it did not reach the server. So, what will happen, this server is waiting indefinitely for the connection to be completed.

This is something which we refer to as half open connection right. So, now, server is in a state where it was expecting the connection request to be completed, but it is waiting, waiting, waiting, well of course there can be a timeout mechanism; that means, it can wait for some maximum amount of time. Then it can assume that due to some error this connection is not going through, let me discard this, but the problem is, this feature of TCP, I means actually has led people to mount some kinds of attacks on systems.

So, this is a simple attack scenario I am just mentioning, suppose some hackers decide that I have to target that particular computer. So, many of these hackers, they come together and they try to send a large number of TCP requests to that particular server, particular computer. And, for each of the requests what they do, they do not complete the final ACK, they do not send the last ack. So, the server will be receiving a large number of requests, but none of those requests will be completed. So, the size of the table, the table will slowly get filled up.

That means connection requests has come, not yet fulfilled. So, finally, that table will be full, there will be a finite size of the table. So, all future connections will be denied because there is no space to keep records in the table. So, even the legitimate connections will get denied, this is something which is called denial-of-service attack. So, this was one of the simpler ways to mount denial-of-service attacks based on the TCP protocol right ok.

(Refer Slide Time: 25:28)

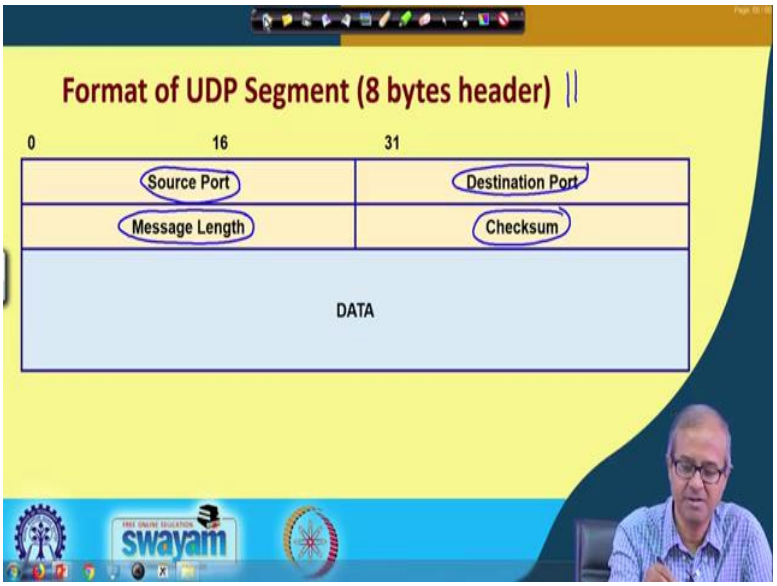


And, lastly talking about the TCP connection termination, well I am just illustrating it here. This is the client, this is the server, well I am showing it in the context of an ongoing communication. Suppose the client was transmitting some data right, some data was coming and was coming to the server. So, server was doing some read and read was successful, it was returning some number greater than 0 means read was successful. Now, when the client wants to shut down the connection, terminate the connection, it will send a packet with the FIN flag set to 1.

What the server will do? I means it will send back an acknowledgment of both the data that it has received and also this FIN it has received right. Then in response to this data it is received maybe the server is doing some final write, some data it writes back to the server and server finally, reads. And, then the server is also done with the connection, server also wishes to close. So, it sends back another FIN to the client and client finally, acknowledges data and FIN and now the connection actually closes.

So, it is like a 4-way handshake, 4 packets are going one in this direction, second in this direction, third in this direction, fourth in this direction. So, this is some kind of a 4-way handshaking that is going on for TCP connection termination, this is how it works.

(Refer Slide Time: 27:32)

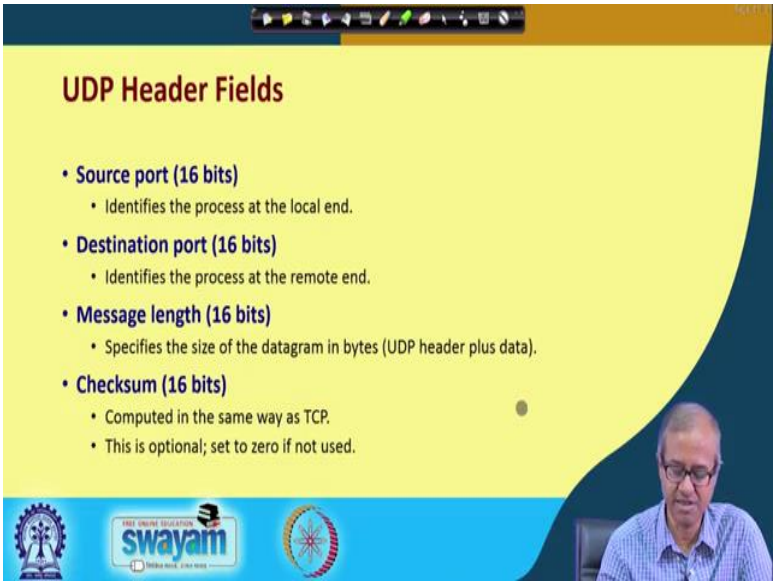


Now, lastly for the UDP packets, the header format is very simple. You see the IP layer provides a datagram service, this UDP is also datagram, but it contains a few other fields. Like, because it is at the transport layer level, it contains the port numbers, it contains a checksum, it also keeps track of the total message length, these are the additional things that are there in the UDP.

And, here the total header size is 8 bytes, 4 plus 4. You see for UDP, because the header size is small the overhead is less. So, for many applications where reliability is not that important, you need fast data transmission, no connection establishment required, you see for TCP you need 3-way handshaking for connection establishment. For UDP there is no question of connection establishment, you just send the data packet, it will be routed and

go. UDP is faster in that sense. So, there are many application where you prefer speed to reliability.

(Refer Slide Time: 28:53)



So, these are the 4 UDP header fields, I have already mentioned right. So, with this we come to the end of this discussion on TCP and UDP. In the next lecture we shall be talking about something like IP subnets, how subnets can be created within IP networks and what are the advantages therein.

Thank you.