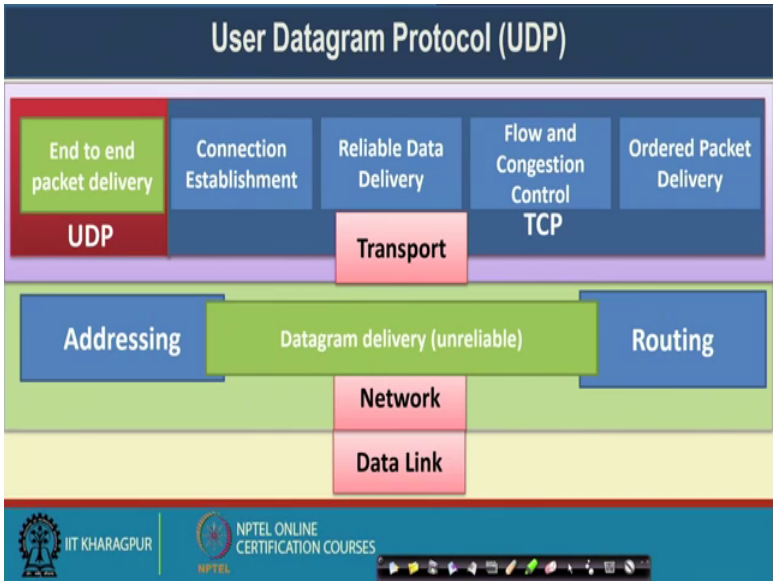
Computer Networks and Internet Protocol
Prof. Sandip Chakraborty
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 23
User Datagram Protocol

Welcome back to the course on Computer Network and Internet Protocols. So, in the last class, we have discussed about the transmission control protocol at a TCP and the several features which are there in TCP.

(Refer Slide Time: 00:29)



So, in this class we will discuss about another transport layer protocol which is not that much widely used as TCP, but still many of the applications use a start will call it as user datagram protocol or UDP. Apart from UDP we will discuss another protocol which is recently developed by Google and gradually getting popularity over the internet which we call as the QUIC and that QUIC it uses this UDP as standard line transport layer protocol.

So, the basic difference between the TCP and UDP is that: a UDP actually is a very simple protocol and it does not support the functionalities which are provided by TCP. So, as we have seen that the TCP protocol it supports the connection establishment, reliable data delivery, flow control congestion control or the packet delivery all these different features, but to implement this kind of features in TCP certainly the protocol

has a significant amount of overhead or the signaling over head what we call it in the networking term.

So, this signaling over head is something like this to make the protocol correct or to make the operation of the protocol correct. You are sending some additional data to the internet. So, for example, for the connection establishment and the connection release, you need to send or you need to have this three way handshaking mechanism which is time consuming.

So, for every individual data packet if you think about multiple sort flows which are flowing in the network just like whenever you are doing web browsing during that time if you are sending such multiple short flows that means, HTTP request response messages. And if every request response message is embedded in TCP connection then that is a problematic. Because for this request response message you require three messages for connection establishment and three messages for connection closer, so that is one of the over head.

The second over head which comes from the flow control and the congestion control algorithm. So, if the protocol detects that there is a packet loss you need to retransmit it. And retransmit always a block the flow of the existing data packet or the new data packets. On the other hand, if you look into the congestion control algorithm, the congestion control algorithm to make the congestion control algorithm distributed and to support maximum fairness. We make the protocol in such a way that you have to start from the slow start phase. That means, you need to start from a very low value of the congestion window and gradually increase that value to reach the capacity because you do not know that capacity.

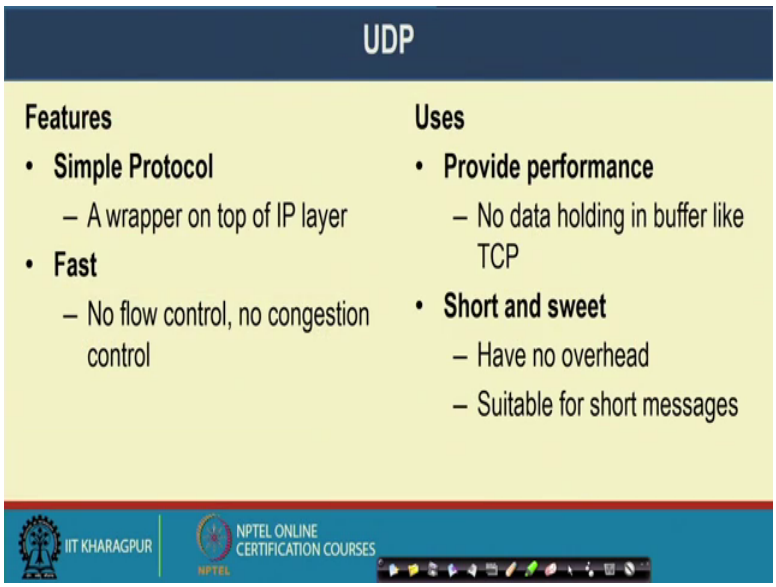
Now, if you think about a high speed link in that high speed link itself you are starting from a very low value of the congestion window and then applying the slow start to reach at the operating point. But with this, if you just think about your flows are very short like just like a HTTP request response message, by the time the TCP flow will reach to the operating point the connection will get closed because you have transferred to your request and response message for HTTP. And you will again start a new TCP with this new set of congestion control algorithm or the new set of slow start mechanism.

So, because of this protocol the TCP, it has this significant amount of overhead which makes the protocol very slow for practical deployment although it supports a good amount of reliability and what is perfectly or works correctly over a real network. But for many of the application, we do not tolerate or we are not able to tolerate or some time we do not require this kind of additional services. What is more required is to send a data somehow at the other end, and then just parse the data.

So, one application is just like your DNS protocol or the domain name system protocol. So the DNS protocol for sending a DNS request and the DNS response, you do not require a TCP connection establishment, because that is going to be a significant over head rather you just send a DNS simply the DNS message. And if the DNS message is not responds, you will certainly have a timeout at the sender side you can again request for the domain name with the next message.

So, for that we support this user datagram protocol or UDP protocol in the internet. So, this UDP just provide you the end-to-end packet delivery or in UDP we term it as the datagram; so end-to-end in datagram delivery. So, it does not provide any of the services like this connection establishment, reliable data delivery, flow control and congestion control, ordered packet delivery all these things in the network.

(Refer Slide Time: 05:26)

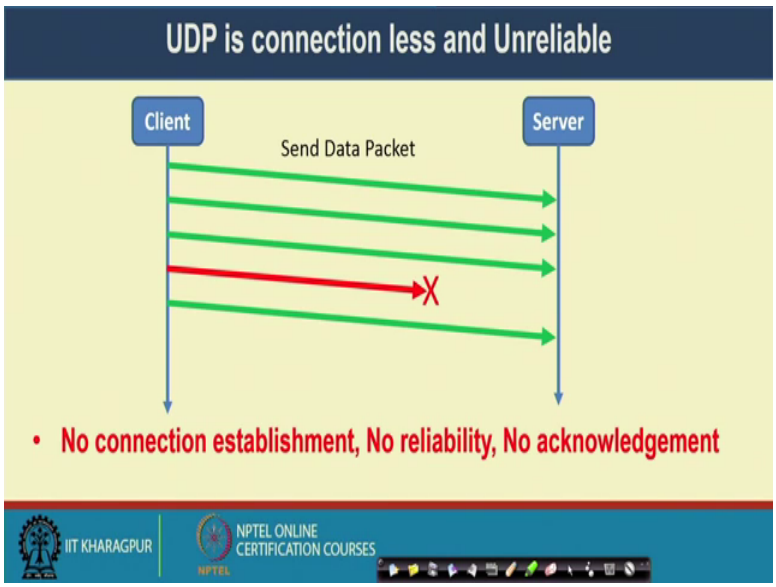


So, these are the broad features and the uses of TCP. So, the feature is that first of all it is a very simple protocol. It is or sometime we say that it is not at all a transport layer

protocol. It is just like a wrapper on top of the IP layer. So, whatever services is being provided by the IP layer, the same set of service is just forwarded to the application by bypassing the basic transport layer functionality. So, UDP works like a wrapper on top of the IP layer, the protocol is very fast, because you do not need to wait for the slow start phase. You do not need to wait for the connection establishment and the connection closer, and you do not have any flow control and the congestion control algorithm. The protocol itself is very fast and it worked nicely when the network has a low loss probability.

So, the use cases of UDP is to provide performance, you do not have any buffer like TCP. So, you can give or support a protocol with faster delivery whenever your link is good, or you do not bother about the packet loss. And it is the kind of short and sweet protocol. So, it does not have any overhead. And it is suitable for short message transfer just like the domain name system.

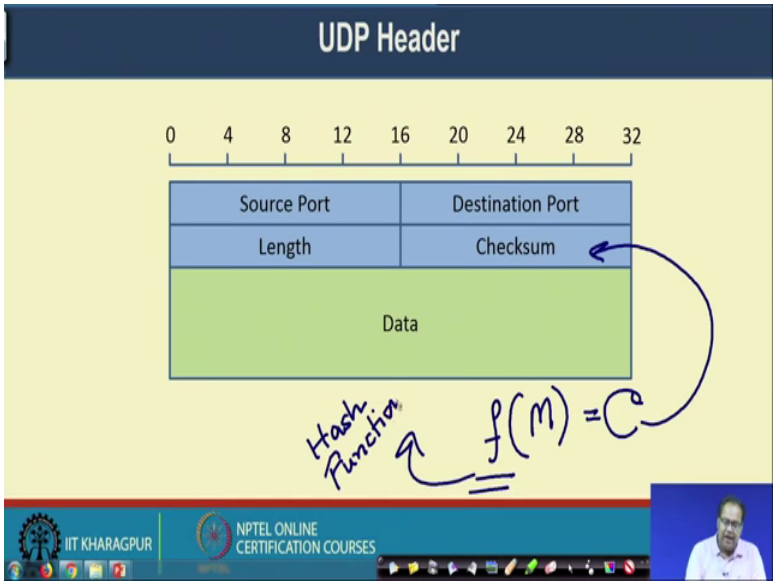
(Refer Slide Time: 06:49)



So, as I have mentioned that UDP is connection less and unreliable. So, you just send data packets or datagrams one after another. If a datagram is being lost the server does not take care of that or the packet get lost. So, you do not care about that. So, if the application cares about packet lost, the application will apply its own acknowledgement mechanism or its own procedure for handling or recovering from the loss. So, you just create a datagram and send it one after another. So, you do not have any connection

establishment. You do not bother about whether the server is running or not. You simply send a packet you do not have any reliability and no acknowledgement mechanism.

(Refer Slide Time: 07:33)



So, this is the structure of the UDP header. So, earlier we have looked into the structure of the TCP header. So, compared to the TCP header this UDP header is fairly simple you only have four different fields the source port the destination port the length and the checksum that is all and then you have that data. So, the source port and the destination port at the server port, and the client port, the length of the packet which is required to find out that how much data is there in your UDP datagram and a checksum field to check the correctness of the packet.

So, although reliability is not implemented, but at the server side or the destination side, you want to find out whether the packet or the datagram that you have received whether that is a correct datagram or something got some suppose happened or something got flipped. So, you want to find out that, so that is when we put the checksum field.

Now, checksum calculation in TCP and UDP, so I have not mentioned that during the discussion of TCP just kept it for the discussion here. So, this checksum calculation in TCP and UDP is a nice feature. So, how checksum is calculated or what is checksum, the details of checksum. We will discuss later on while we will look into various error correction codes in the context of data link layer, but just to give you a brief idea that

checksum is nothing but a function. So, you can think of checksum as a function if. So, inside that function you are providing the message and then you are getting certain value.

So, whatever value you are getting, so this C this is the checksum that C you are putting here. Now, because the checksum is of fixed length, you can think of this function as a hash function. So, any hash function can be used as a checksum, but ideally these IP checksum or internet a checksum which we apply for network data transfer. It is not a complicated hash function like our traditional pictographic hash function, because we do not require that one way property that much rather we are just concerned about to get a fixed size message digest out of the message. So that is why this internet checksum computation is fairly simple.

And if you are again apply a pictographic hash or a complicated hash function here. It will take a significant amount of time to compute that corresponding checksum, so that we do not what. So, we simply use a simple method for doing the checksum. So, a simple method as the name suggests in internet the checksum is just like you divide the entire message into fixed size block, and then make ones complement addition to compute the checksum, so that is the basic idea of internet checksum that we apply here.

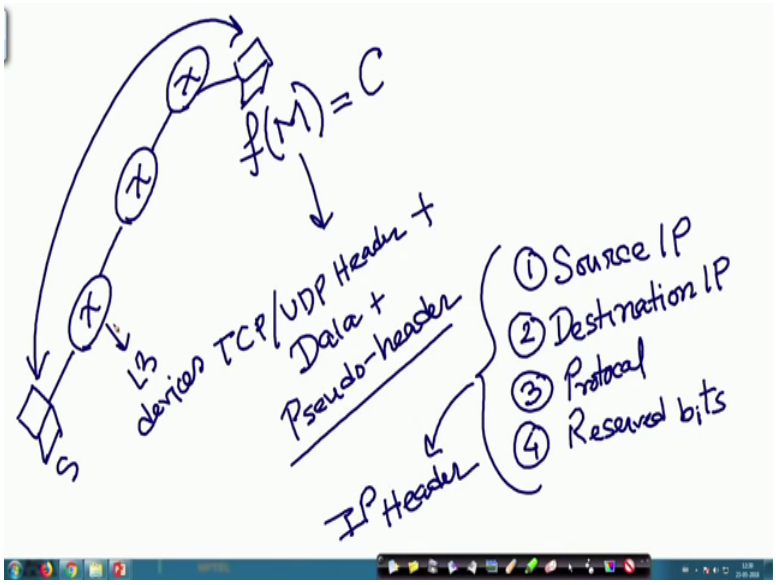
Now, in internet checksum, so the detailed procedure of internet checksum and example will discuss later while we discuss about the error correcting code as I mentioned just before. But ultimately this checksum is giving you a fixed size code that is the C. Now, whenever you are receiving this message at the receiver end what you do whatever you are receiving with this received data, again you apply the same function to compute the checksum. And find out that what is the checksum that has been transferred with the message and the checksum that has been computed at the receiver side whether they are getting matched or not. If they are getting matched, that means, whatever value has been transferred from the sender side, you have received that for a particular value.

Now, again remember that checksum is not to ensure packet integrity from the security attacks or the external attacks. It is just to ensure packet integrity from the network fault or the system faults. So, whenever you are transferring the data because many of the time you are doing a digital to analog conversion or analog to digital conversion, then encoding decoding a packet, you are applying the modulation technique at the physical layer. So, it may happen because of the sampling error or the quantization error some bit

ideally or it was one but that got flipped to 0 during this analog digital conversion or the modulation and coding states or some zero bits got flipped to 1.

So, just to detect those kinds of things we applied the checksum. It is not to not to ensure that the packet is free or the packet has saved from some security attack based under cryptography or network security point of view. So, because of that so whatever checksum you are getting at the receiver side you compute the checksum value that has been transferred with the packet if they matche; that means, the packet integrity got fizz up, there was no such error that has been introduced during signal processing. And you have received the correct datagram that was sent by the sender. Now, while compute checksum in TCP and UDP; TCP and UDP takes certain things into consideration during checksum computation.

(Refer Slide Time: 13:02)



So, as we have said that checksum is nothing but a function in where you are taking a message a mess input and you are computing the checksum value. Now, in this message, so TCP or UDP they put TCP or UDP header plus the data that you are sending plus a pseudo header. So, this pseudo header is actually not transmitted with the packet, it is just used for the computation of checksum. And once the checksum is computed that pseudo header is getting dropped or that is that gets dropped.

So, what is the content of the pseudo header, the content of the pseudo header is the source IP, the destination IP, then the protocol field from the IP header. So, it actually

takes certain fields from the IP header. So, all these frame fields are coming from the IP header. And the fourth is the reserved bits. So, there are 8 reserved bits in IP. So, those reserved bits from the IP header. So, all these fields comes from the IP header so that way we consider the pseudo header in the computation of checksum. But remember that this pseudo header is not transmitted with the packet rather once you have computed the checksum; they do just put the checksum and discard the pseudo header. At the receiver end, receiver will again construct the pseudo header and compute the checksum make a match with the receive checks and drop that pseudo header.

Now, the thing is that; what is the purpose of including the pseudo header in the checksum computation; so this pseudo header is included in the checksum computation just to do a double validation of the source IP, destination IP and a protocol field, and a reserved value field. So, these fields are very important from the perspective of transmission of a packet because these fields actually help you to identify the correct source at the correct destination. So, although IP header includes its own checksum field, but this IP header changes the checksum at every individual layer, because if you look into the network diagram you have this source followed by multiple hops routers and then finally, is your destination.

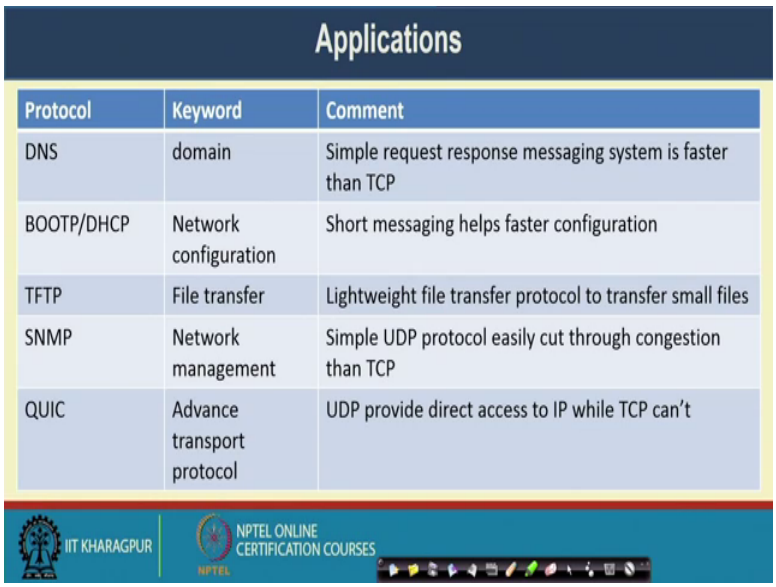
Now, every router they apply the routing mechanism, they look into the IP header, they may make IP header and then they again compute the checksum the IP header checksum and put it at the part of the IP header. So that is why the checksum which is there in the IP header may get changed whenever you are going from one layer tree hop to another layer tree hop. So, all these are the layer three devices. So, whenever you are going from only a tree hop to and a tree hop then the things may get changed. But at the same time we do a end-to-end invalidation with this end-to-end protocols.

So, this transport layer protocols are basically the end-to-end protocol. So, the UDP header or the TCP header never gets changed at the lower layer of the protocol stack at the internet layer of the protocol stack so that UDP header or the TCP header will never get changed at the individual routers. So that is why we make a double check in the TCP header or the UDP header by putting this pseudo header at the source IP, destination IP. All these fields that this intermediate router somehow there this fields has not got changed.

Because, in the router if you later on will discuss the entire processing of the routers you will see that whenever it receives an IP packet it takes the IP header out, applies the routing mechanism then again adopt the IP header and send it to the outgoing link, because that IP layer processing is done at the router level. So, if there is certain inconsistency or certain faults inside the router that may introduce an error to the source IP or the destination IP field. So, we want to make sure through this integrity change at the UDP header that no such error has been occurred during the transmission of the transport layer segment or in TCP or the transport layer datagram at UDP. So, that is why you put the pseudo header as a part of the checksum computation.

But as I have mentioned earlier again repeatedly I am mentioning that this pseudo header is just used only for the computation of the checksum. And this pseudo header is not transmitted during the transfer of the data, and that is just to check this end-to-end integrity of the data transmission ok.

(Refer Slide Time: 18:39)



Now, there are multiple application that uses UDP as you have looked earlier the DNS protocol the domain name system protocol. It is simple request response message. So, we require it faster than TCP. So, we apply UDP here. Then this BOOTP or DHCP, they are the network configuration protocol. Again they are short messaging protocol which helps faster configuration of the device devices. TFTP, TBL, file transfer protocol it is a simple lightweight file transfer protocol to transfer small files. SNMP, the network management

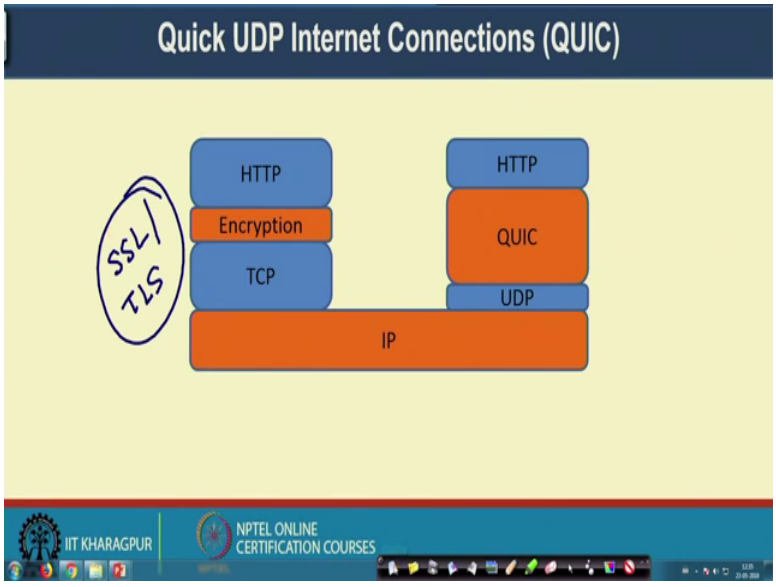
protocol or the simple network management protocol it is again a simple UDP protocol which is easily cut through congestion than TCP.

So, in TCP, if there is a congestion, then TCP deduce the rate, but in case of UDP as it does not take care of the congestion, if the packet comes to the buffer. And if it is not dropped from that intermediate buffer, eventually it will get transmitted so that is why we use this UDP in case of SNMP.

Then the interesting protocol comes which is QUIC, QUIC, UDP internet connection that was developed by Google a couple of years back which is a advance of the transport protocol. So, idea behind QUIC is to overcome many of the shortcomings which were there in TCP, because of the slow start phase, the connection establishment for every individual flow. And in QUIC UDP provide a direct access to IP.

So, with the help of UDP, what we does that it directly send a packet via IP, but whatever additional facilities like flow control, congestion control, liability. All these things are there they are implemented as a part of application with a secure binding.

(Refer Slide Time: 20:31)



So, let us look briefly about how QUIC works. So, the full form of QUIC is QUICK UDP internet connection. So, the first paper of QUIC or the detailed paper of research paper of QUIC that came from Google in 2017 see comes. So, I suggest all of you to go

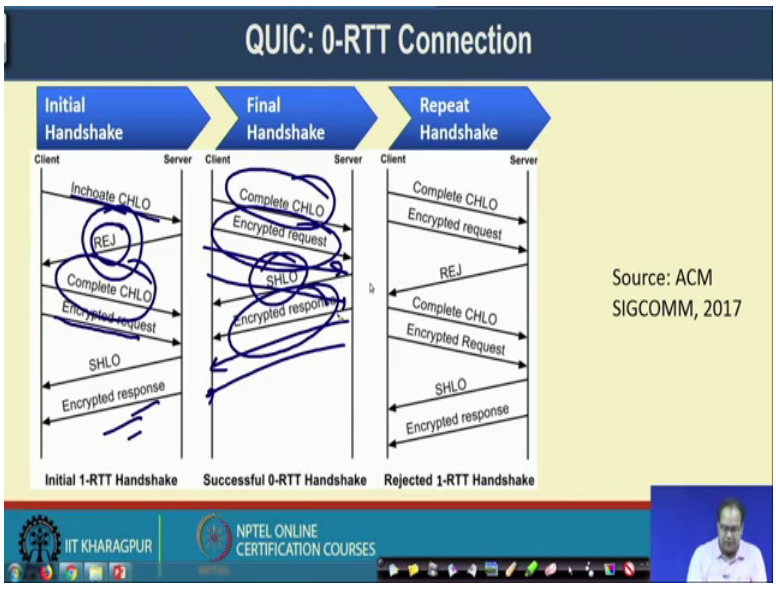
through that paper to know more details about QUIC, also you can see the internet draft of QUIC to look into the specific protocol.

So, if you look into the difference between the application over TCP and application over QUIC, so in bottom we have the IP layer whenever you are applying HTTP or say HTTP is the secure version of the HTTP. So, under HTTP you apply encryption layer through transport layer security or secure or secure socket layer and then you use TCP to transfer the data at IP.

Now, what QUIC does, it makes a direct interaction with HTTP. So, HTTP runs on top of this QUIC; and QUIC runs on top of the UDP, and it then access IP. Now, this security features the encryption part it is embedded inside that QUIC protocol. So, you do not require any other secure layer. So, every QUIC packet is end-to-end encrypted and it provides end-to-end security. So, you do not require another security layer that we require in case of TCP with the help of SSL or secure socket layer or transport layer security protocols.

So, this SSL and TLS; this kind of encryption protocols are not required in case of QUIC because we inherently applies the security features ok.

(Refer Slide Time: 22:11)



So, one of the important aspect of QUIC is to combine multiple short flows together. So, if you remember the problem which is there in TCP the problem that I was explaining

earlier that if you want to transfer 1 HTTP packet in HTTP 1.1, you had one TCP connection for every request response message. And even with HTTP 1 or 2 HTTP 1.1 what you had that you can combine multiple request responds together, but based on the web browsing nature normally we browse a web then we wait for certain time, and then we move to another page. And during that time although it co it is combining multiple such requests responds together, but that mostly will mostly related or mostly limited to a single session. Whenever you are moving to a different page, you need to create a different TCP connection.

Now, for every TCP connection you require three way handshaking. So, just to send a few HTTP request response message, you require three connection at three way handshaking at the sender side and a three way handshaking during the connection termination as well. Now, QUIC actually solves this problem in this way. So, in case of QUIC, during the initial handshaking, whenever you are connecting to the server for the first time during that time you have to do a detailed handshaking. But after that, you do not need to do that detail handshaking rather you can directly use the previous handshaking part the connection that has already been established to send further data.

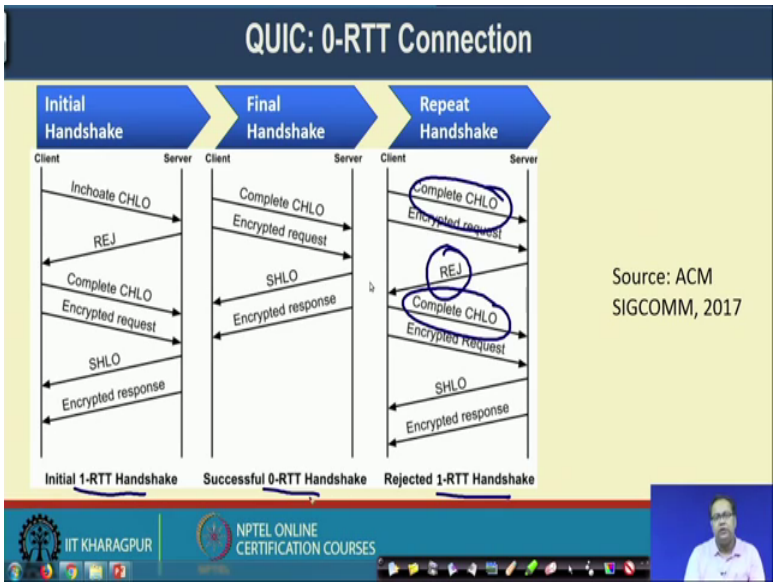
So, this works in this way. So, as I have mentioned that QUIC is an end-to-end encryption protocol because of that you require certain credential from the server. Now, initially that client does not have those kinds of credentials. So, the client said say sends a equate CHLO, client CHLO. When the client says this equate client CHLO which is received by the server. And the server finds it out that the client CHLO does not has the required security certificate, it sends a reject message. And with this reject message the server sends the security credential to the client.

Now, the client has the security credential. With this security credential, it sends a complete client CHLO. Now, here the interesting thing is that because the client has already received this reject message from the server, the client actually knows that the server is running, and the server is ready to accept packet. So, the client can start sending encrypted request. Now, if the server wants to send the message from the server side to the client side, that means the responses, the server sends a server CHLO initially. After the server CHLO is sent, because the server has already received the client credential from this client; CHLO, it starts sending the encrypted responses from the server side. So, you actually required 1-RTT here.

Now, once this connection has been established then for the next ongoing connections between the same set of client server, you do not require this 1-RTT handshake rather you require a 0-RTT handshake. That means, you already have received the server credentials, because you already have received the server credentials. You can start with this complete client CHLO. And you know that the server is running because it has already received certain packets. So, you can start sending the encrypted request. And if the server wants to send data to the client, server start with a server CHLO and then that encrypted responses. This is not only one encrypted response, it can send multiple responses or (Refer Time: 26:06) multiple requests simultaneously.

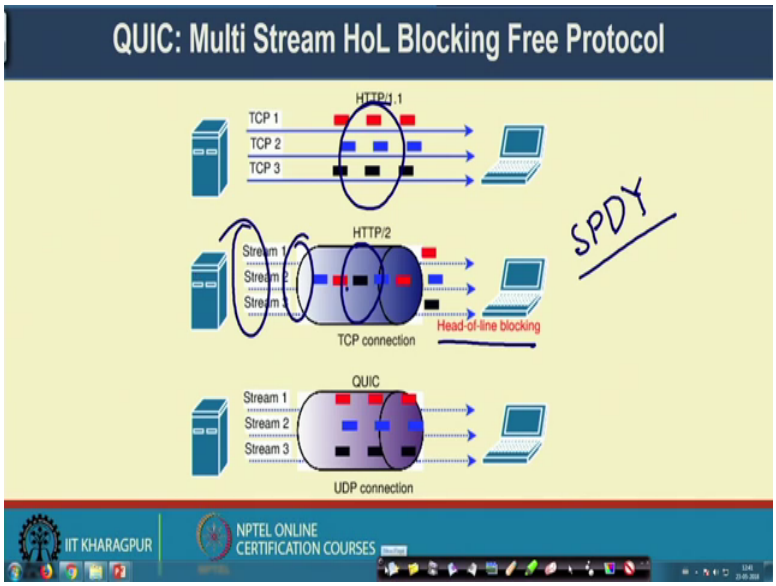
Now, sometime it may happen that well during this time it may happen that the server credential has been changed.

(Refer Slide Time: 26:15)



If the server credential has been changed, when the server received is complete client CHLO, the server sends reject message with the updated server credential. Now, with this updated server credential, the client can reinitiate the connection and start sending the request. So, whenever the server credential gets it you require a 1-RTT handshake. And at the initial time you require a 1-RTT handshake; after in between you can always use the 0-RTT handshake to transfer the packets ok.

(Refer Slide Time: 26:49)



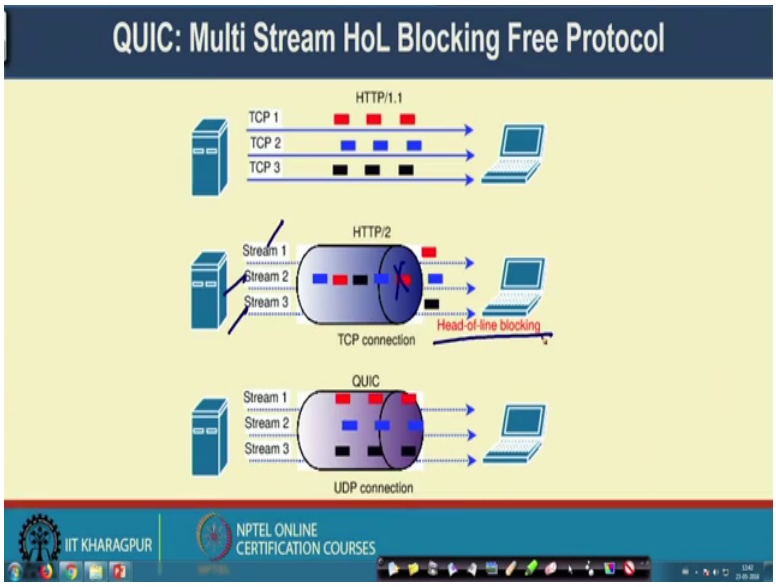
Another interesting feature of QUIC is to support multi streaming. And it supports something called head of line blocking free protocol. I will come to that point of what is mean by head of line ball blocking in a couple of minutes. So, what happens in HTTP 1.1 that you can have multiple TCP streams one after another and to with this multiple TCP streams you can send multiple with every TCP stream you can send multiple request response messages in between the client and the server.

But, because you are having these multiple TCP streams in parallel for every individual stream, you should have this connection establishment and every TCP stream will go through the slow start phase. So that is overhead for HTTP 1.1; so that is so in HTTP 2 or sometime that was a earlier proposal from Google, they call that particular protocol as SPDY SPDY. So, in that particular protocol, you can multiplex multiple streams together.

So, here you are combining all these streams together and have a single TCP connection between the server and a client. So, here all these streams are getting multiplexed to a single stream. And this multiple stream is sent to the client. So, in this case you have a problem called head of line blocking. What is that say the nature of the TCP is that if it receives a single out of order packet, then it put that out of other packet in the buffer and start sending duplicate acknowledgements. But if it does not receives in order packet then it will not send the packet to the application.

Now, here whenever you are combining multiple streams together what happens even if a single packet get lost, because of that single packet lost this entire TCP connection will get blocked and which will in turn block. All the streams even if certain through packets from certain streams are receiving and a buffer.

(Refer Slide Time: 29:01)



So, because you have a single connection say assume at red packet got lost. And this is the stream corresponds to the red packet. These are the streams corresponds to the blue packets and the black packets. Even if you are deceiving blue packets and a black packets, because you have a single TCP connection that TCP connection will not send those packets to the corresponding stream.

So, those streams will also get blocked. So, this is called the head of line blocking. Now, QUIC solves this head of line blocking problem by using UDP connection. So, UDP connection does not have this problem of blocking due to reordering. So, UDP simply passes the packet to the streams. And then the streams take cares of the QUIC protocol itself takes care of sending the packet to the individual streams. And it maintains the stream wise flow control and a congestion control algorithm.

So, I am not going to flow control and the congestion control algorithm of QUIC in detail. If you are interested you can look into the specific draft or the (Refer Time: 29:53) 2017 paper. Another interesting feature in QUIC is that TCP uses this duplicate acknowledgment, but QUIC does not use the duplicate acknowledgement. Even for a

retransmission, it assigns a new sequence number to the packet. So, because the packets are basically transmitted over UDP, QUIC is not a stream oriented protocol; it does not use the byte sequence number rather it uses the packet sequence number for simplicity. And for every packet that means the original packet as well as the retransmitted packet. It puts a new sequence number, so that is why you do not have this problem of duplicate sequence number and blocking due to this duplicate acknowledgements.

So, these are few of the features of QUIC and this protocol QUIC is gradually getting popularity in the internet, many of the services like YouTube or Google drive which is coming from Google. Google has already started a deployment of QUIC, and the current version of chromium based browsers, they have the implementation of QUIC. So, many of the recent protocols have started using quick recent applications are mostly all the applications from Google they have started using QUIC. So, possibly QUIC is the future protocol which is going to replace the standard TCP based data delivery. And during that time UDP is actually going to be more important compared to TCP. So that is possibly the future of internet, so that is all about the transport layer protocol a different kind of transport layer protocol.

In the next couple of classes, we will do a practical thing. We look into this concept of socket programming and network programming. We will see that with the help of the socket programming, how you can access a specific transport layer protocol and you can start transmission of data. And you can write your own application by utilizing this different type of transport layer protocol. We will see certain demo of that. And then we will move to the next layer of the protocol stack that is the star internet layer or the network layer of the protocol stack.

Thank you all for attending this class. See you again.