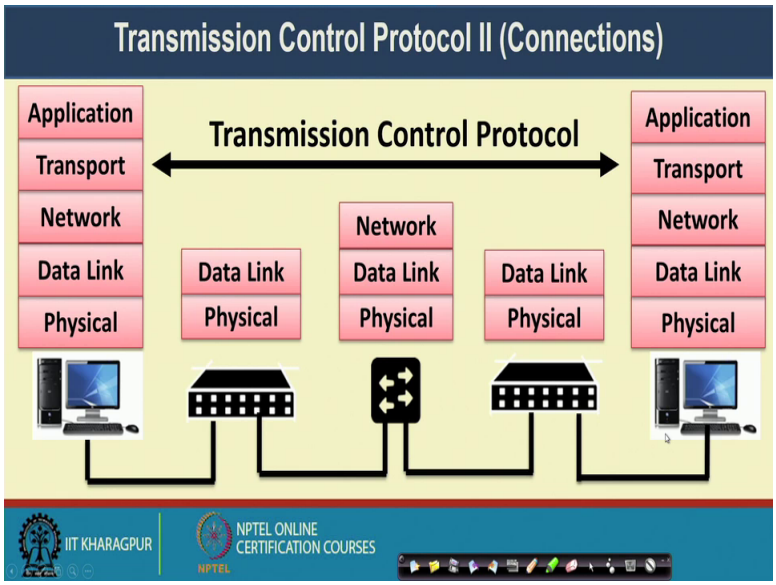
Computer Networks and Internet Protocol
Prof. Sandip Chakraborthy
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 20
Transmission Control Protocol - II (Connections)

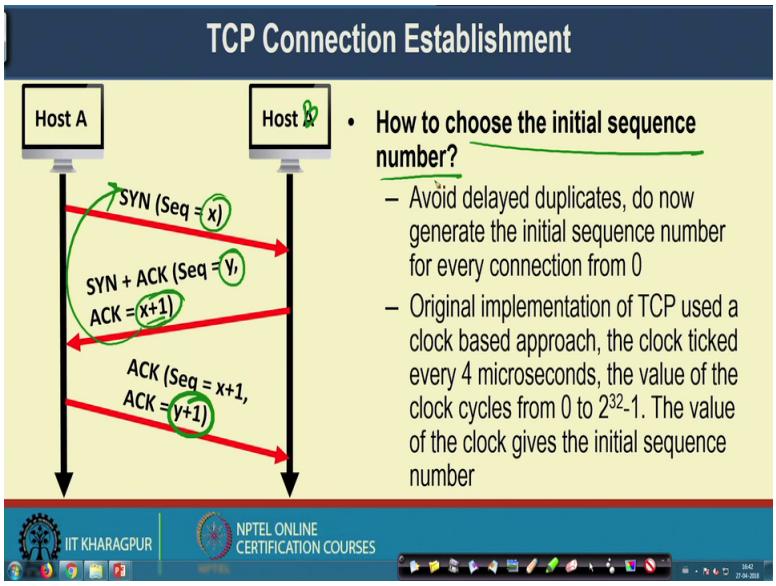
Welcome back to the course on Computer Network and Internet Protocols. So, we are looking into the details of Transmission Control Protocol or TCP.

(Refer Slide Time: 00:22)



So, in this lecture we will look into the details of TCP connection establishment and how TCP chooses the initial sequence number, based on the concept that we discussed earlier and then in the subsequent lecture. So, we will go to the flow control mechanism in TCP in detail.

(Refer Slide Time: 00:43)



So, TCP connection establishment, it is a three way handshaking based mechanism it is utilizes a special connection request message called SYN a short form for synchronization we call it as TCP SYN message. So, the connection establishment using 3 way handshaking mechanism that is something like this like Host A and Host B wants to communicate with each other. So, Host A and Host B wants to communicate to it each other Host A initiates the connection establishment.

So, Host A sends a SYN message with certain initial sequence numbers. So, in a moment we will discuss that how TCP chooses this initial sequence number. So, it sends a SYN message with the initial sequence number as x then Host B sends an acknowledgment message along with also a SYN message, so this SYN message from Host B to Host A it is used to ensure the bidirectional connection in TCP. So, if you remember in the last class we have talked about that TCP connection is bidirectional. So Host A can communicate with Host B at the same time Host B can also communicate with Host A and because of this reason Host B also sends a same packet with an initial sequence numbers. So, here in this example Host B sends this SYN message while sending back the ACK.

So, we are basically piggy backing SYN and ACK together piggy backing means we are combining 2 message together in terms of TCP header, you need to set bit 1 for both the SYN flag and for the ACK flag. So, this SYN plus ACK message it is sending a new

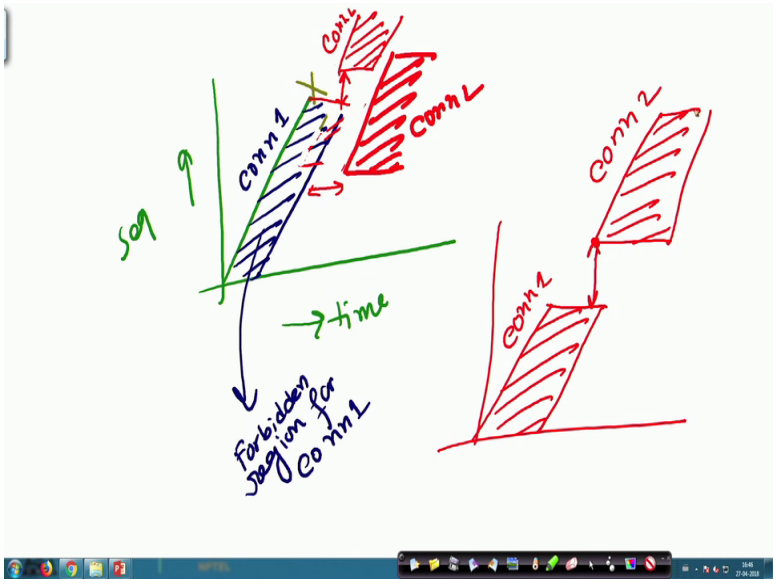
initial sequence number, so this sequence number y it will be used for B to a data transfer and earlier proposed sequence number from A to B that is x will be used for A to B data transfer and in acknowledge with this x, so it sends a acknowledgement number of x plus 1.

Now, if you remember the connection establishment procedure 3 way handshaking mechanism that we have discussed earlier in the case of general transport layer service model. Host A can see this message host a can find out that the acknowledgment number it corresponds to the SYN message that it has transmitted and if it corresponds to the SYN message that is transmitted it takes this SYN plus ACK as a feasible one or a valid one. And then it sends a acknowledgement message finally to B and in that acknowledgement message it sends a sequence number of x plus 1 incrementing the previous sequence that it has initiated and acknowledges this acknowledgement number y plus 1.

So, with this 3 way handshaking mechanism Host A and Host B initiates the connection. Now the question is that how will you choose the initial sequence number. So, choosing the initial sequence number is an important aspect that we have looked into a generic discussion of a transport layer service models. So, while choosing the initial sequence number the objective is to avoid the delayed duplicate. So that you can identify a message by looking into your sequence number whether that message is a delayed duplicate or it is just like the application has crashed and the application has initiated another connection at the same port with the difference sequence number.

So, to do that what you need to ensure that well, the initial sequence number that you are choosing that initial sequence number should not fall within the forbidden region of the previous sequence number. Now how will you ensure that to ensure that earlier we have seen that well, whenever you are choosing the initial sequence number you have 2 ways to choose the initial sequence number. So, just try to remember the concepts that we discussed earlier, so just briefly explaining it again for your convenience.

(Refer Slide Time: 04:42)



So, whenever you are choosing this initial sequence number. So this is the time axis and this is the sequence number axis. So, this was the earlier connection if this was the earlier connection then this was say the forbidden region for this particular connection. So, this is the connection 1 and this is the forbidden region for connection 1.

Now whenever you are initiating a new connection say at this point connection 1 got crashed here, so once connection 1 got crashed you want to initiate a new connection. And whenever you are initiating a new connection you need to ensure that you are not starting the new connection say; this is your new connection, connection 2 you are not starting this connection 2 at a point such that the forbidden region for connection 2 overlaps with connection 1 so this we do not want.

So, to prevent that what we do that to prevent that we want to initialize connection 2, such that these 2 forbidden region does not overlap with each other. Now to do that you have 2 options the first option is the first option is just, so the first option is you make shift at the time domain and the second option is that to make a shift at the connection establishment domain. That means, the sequence number domain. So, the first step is that you start it after giving a gap so this is connection 2. So, you start it after giving a gap, so that these gap will ensure that the sequence number space do not overlap.

So, you wait for certain amount of time to ensure that all the packets for connection 1 which was transmitted in the network they have bide off and no traces of that those of

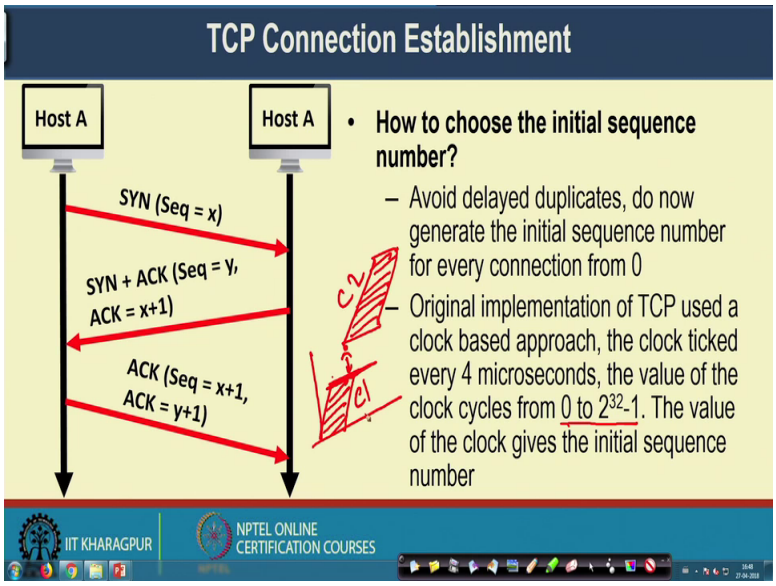
they are in the network and then only you try a new connection, otherwise the option is that you choose the initial sequence number in such a way which will be high enough. So, there would be difference here from the last sequence number which is used by connection 1 and a new sequence number that you are using from connection 2. So, there is a difference here such that you will be able to ensure that this sequence number space which was been used by connection 1, you are not going to use that sequence number space for the connection 2 for the data of connection 2.

Now TCP uses the second principle, so TCP ensures that whenever a connection say connection 1 crashes, so this was connection 1 whenever connection 1 crashes whenever you are starting connection 2 you choose the connection 2, the initial sequence number of connection 2 in such a way that there is a gap in between so this is for connection 2; there is a gap in between and there is no overlap between the sequence number which is being used by connection 1 and which is being used by connection 2.

So, for that TCP uses a clocking mechanism, so TCP generates the sequence number based on a clock. So, that was the first implementation of TCP or the earlier version of the TCP, it used the sequence it used to generate the sequence number based on a clock mechanism. So, the methodology was something like this, so this original implementation of TCP it used a clock based approach. So, the clock ticked every 4 microseconds, so whenever the clock is ticking you are generating a new sequence number if you have a byte 2 set and the value of the clock it cycles from 0 to 2 to the power 32 to minus 1.

So, you remember that TCP uses a 32 sequence number, so your entire sequence number space is 0 to 2 to the power 30 to minus 1; so that means, at every 4 microseconds you are generating a new sequence number and whenever a connection crashes and get restarted then you will use the sequence number which is being generated by the clock. So, that is used for generating the initial sequence number, then the sequence number will incremented based on the bytes that you are receiving and you are transmitting based on your flow control and the congestion control algorithm. So, this value of the clock it gives the initial sequence number which will be being used.

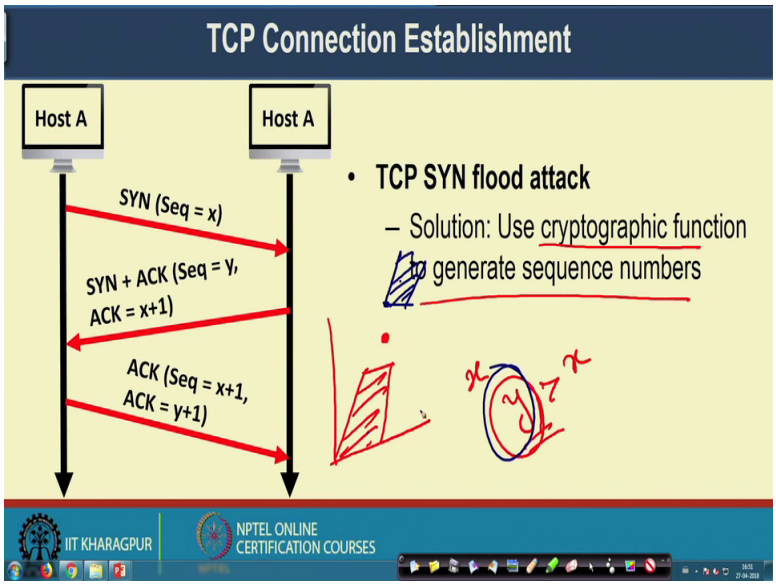
(Refer Slide Time: 09:29)



So, with this clock based mechanism what you are ensuring; that means, whenever a previous connection say get crashed here the connection get crashed here and you are restarting the connection by the time the clock value will increase. And because the clock value is increasing you will obviously get a initial sequence number here, which has certain gap from the sequence number filled which was used by the previous connection.

So, you will start from here and you will be able to ensure that the forbidden region of the sequence number which is been used by connection 2, so this is connection 2 that is not overlap with the forbidden region of connection 1. Now, with this particular approach we have a problem like this sequence number generation becomes little bit deterministic.

(Refer Slide Time: 10:17)



So, if you know that well the clock is ticking at every 4 microsecond and at every 4 microsecond you are generating a new sequence number; that means, an attacker will be able to understand by looking into the previous sequence numbers that, what is the clock recreate current clock recreate. And when the previous connection got crashed how much time has been passed in between divided by the 4 microsecond that should be the initial sequence number of the next connection. If that is the possible that is the case then there is the possibility of SYN flood attack which can happen in case of TCP.

So, in case of TCP the SYN flood attack is that you are continuously sending this kind of spurious SYN connection to a node and that particular node will accept those connection at a genuine connection and they will get a blocked here, because, they will think of that that particular initial sequence number which is been generated it is it is indeed a correct initial sequence number based on my clock input, so it will accept those SYN connection. And if you are sending multiples such SYN connections from multiple computers that translates to a denial of service attacks. So, the computer and a server will only become busy to response to the SYN packets, it will not be able to say in the any data packets in further.

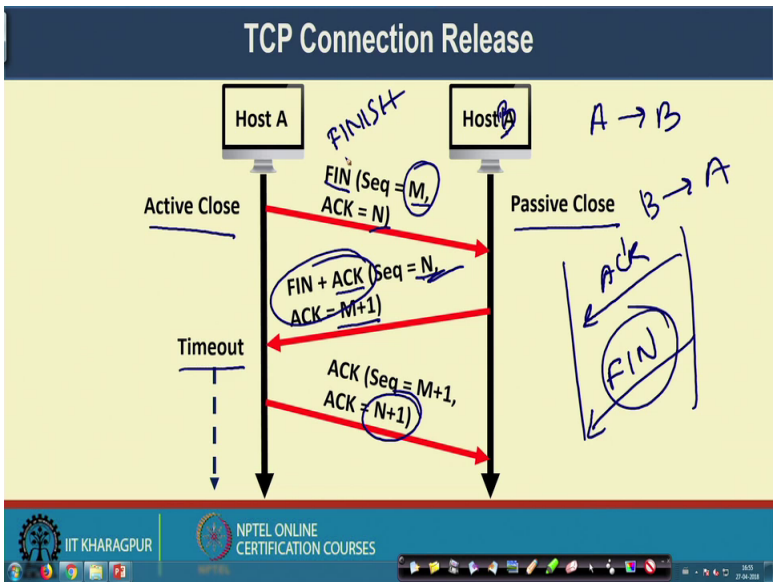
So, that is a possibility of a SYN flooding attack to launch a denial of service over TCP. So, that is why the later function of the TCP or indeed the current version of the TCP what it does, that it uses the cryptographic function to generate the initial sequence

numbers. So, it is like that your clock value will give 1 one function 1 value. So, say the clock value is saying that your initial sequence number should be x, if your initial sequence number is x then you apply a cryptographic function to generate a initial sequence number such that your initial sequence number y it is more than x and because this is generated from a cryptographically secured function, so the attacker will not be able to predict the value of y.

So, that way you are ensuring that well in case of a previous connection, when the connection got crashed here and you are trying to generate a new sequence number, your clock value says that you should generate the new sequence number from this point. But, then whenever you are restarting a connection you should generate it from this point, but then the cryptographic has function generates another value which is more than this particular point say for yet here. And you are starting your new connection from that point.

So, that way it will ensure because you are going higher of that, it will ensure that there is no overlap between the forbidden region of this new connection and the forbidden region of this old connection, and at the same time because this value was cryptographically generated the attacker will be not be able to guess that. So, that way you will be able to safeguard thus in flood attack in a TCP ok.

(Refer Slide Time: 13:28)



Now, TCP connection release it again uses the 3 way handshaking mechanism. So, we have 2 Host, Host A and Host B now Host A want to close the connection when Host A wants to close the connection at it initiates this connection closer we call it as an active close. So, in case of active close Host A will send a FIN message FIN is the full form of finish.

So, you want to close the connection send a FIN message with a current sequence number and a current acknowledgement number, then Host B wants it receives the FIN message it again go to the close connection closer it this passive close. So, in the passive close it sends a FIN message, it sends an acknowledgement message to this fin, so that Host A can close the connection in that acknowledgement it acknowledges this FIN message with n plus 1. And at the same time it Host B once to close the connection itself, so this FIN message from Host A to Host B it is closing the connection from A to B.

Now, if B wants to close the connection as well, so we have a bidirectional connection B also B to A now if B wants to close this connection B sends this FIN message. If B does not want to close it immediately then what B can do that B can only sends the acknowledgement message and later on when it wants to close the connection it can sends the it is own FIN message that is also possible. Now, once Host A receive these acknowledgement message it starts a timeout, this timeout is to prevent these data loss due to the symmetric nature of the closer. So, if you remember we have looked into earlier that asymmetric closer insight and unreliable network is not possible so we want to implement a symmetric closer with a timeout value.

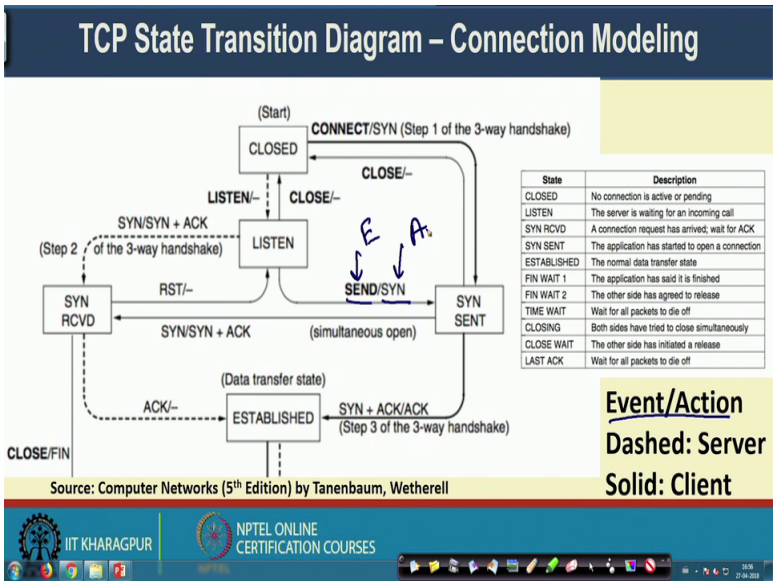
So, this timeout value ensures that well if you are still receiving some packets from B, then you can continue receiving that packet once this timeout occurs you completely close that connection you will not accept any packet after that. Even if you get any certain packets after that, but those packets will get lost you cannot do anything with those packets and 1. Similarly Host A sending the acknowledgement message against a FIN message or FIN plus acknowledgement message send by Host B and it updates the acknowledgement number accordingly against these sequence number, and sends back the acknowledgment to Host B, so Host B again closes the connection and do not send any data.

So, you can see that the timeout is here in case of the active close, but for passive close we do not require any timeout because, that is the last entity which is going to close. We require this timeout for active close because it may happen that when Host A is initiated this closer, Host A after getting the acknowledgement Host A can still receive some data from Host B because, Host B has not sent any FIN message with it or even if it has sent a FIN message it may happen that because of these reordering of the packet you may receive certain packets after that.

So, we apply this timeout mechanism at the active close side, but at the passive close side we do not require the timeout because, in the passive close side whenever you are getting an acknowledgement from Host A. You know that Host A has already closed it is connection, Host A has send a FIN message itself. So, it is just like that your friend has closed the door and your friend has not do not want you to enter his room. So, you do not want to wait any more.

So, so that is the reason here that Host A has already initiated that finish message. So, Host A will not send any more data Host B knows that, so for the passive close case you do not need to wait for the timeout value, whereas for the active close case I am forcefully trying to close the connection. So I am giving an opportunity for the other in to send few more data to me if it wants, so that is why we have this timeout value here. Now as you have looked earlier for that hypothetical transport layer protocol that these entire transport layer protocol follows a state transition diagram. So we also have a state transition diagram for TCP.

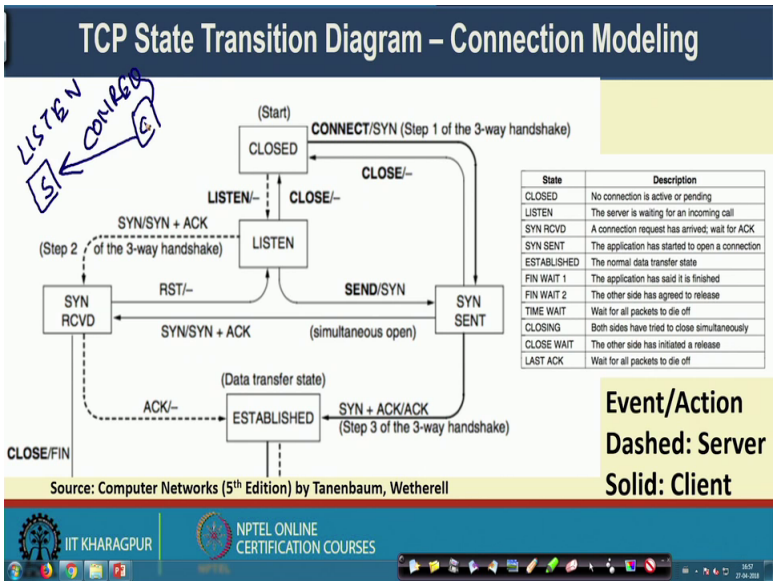
(Refer Slide Time: 17:48)



So, let us look into the state transition diagram of TCP in little details because, that is the important concept for TCP. So, this entire state transition procedure start from this close state; that means, the server and the client both of them are closed, so they are they have not started any TCP socket yet.

So, this are the notation that you see that everything is written by 1 message slash another message, so this is the event action appear. So, that first 1 is the event and second 1 is the action then, this dashed line is for the server. So, this dashed line which is being followed that is for the server and solid line is for the TCP client.

(Refer Slide Time: 18:43)



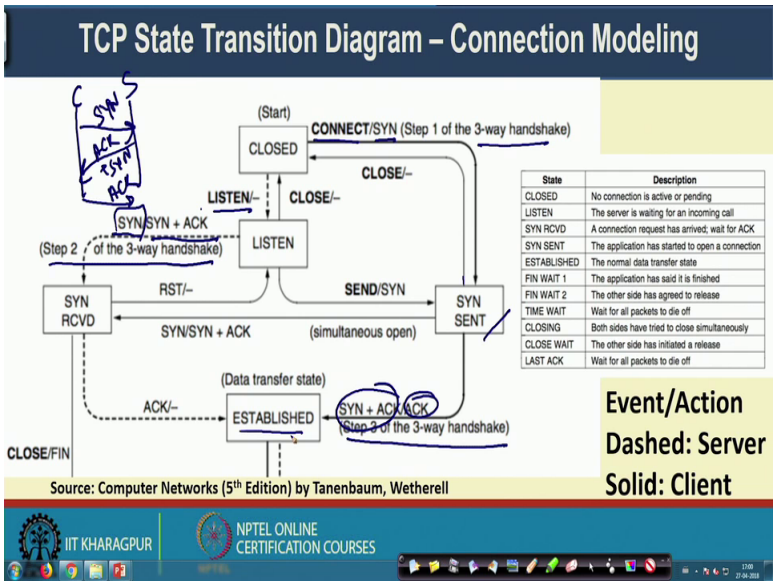
So, the client as you know that in a client server OSI model, the server remains in the listen state for getting a connection getting a connection request from a client. So, the client initiates the connection request and once connects client receives sends a connection request and a server receives it starts processing with that connection.

So, let us see that how this entire team moves using this state transition diagram using TCP state transition diagram. So, from this close state let us first look into the client side. So, the client initiate the connect system call and sends the SYN message. So, that is the first step of the 3 way handshaking procedure and then the client moves to the SYN sent state.

So, at this state the client has sent a SYN message and it is waiting for the acknowledgement from the server. Now from this SYN state sent state client can decide after sending the SYN that I do not want to send any more data want to immediately close the connection, so it may use a close message to close the connection immediately and move to the close state.

So, whenever it is in the close state even if the server receives the SYN message and send back to it an acknowledgement it will not accept that acknowledgement, it will simply drop the acknowledgement. And, will not send any more data because it is in the close state and the server will wait for some amount of time get a time out and again move to the close state. So, that is for SYN sent state.

(Refer Slide Time: 20:29)



Now, after you have send a SYN then you can in that 3 way handshaking mechanism from the client to server first you have to send the SYN message, then you will receive an ACK from the server along with the SYN from the server as well and finally you will send the ACK message.

So, here in the second stage you have received SYN plus ACK message from the server. So, once you have received these SYN plus acknowledgment message from the server, then you send an acknowledgement message and move to the established state. Similarly the server from the close state it first makes this listen system call and moves to the listen state. So, at the listen state it is ready to receive any connection establishment message. So, once it receives a SYN message it sends back with a SYN plus acknowledgment message.

So, this is the second step of the 3 way handshaking mechanism. So, the server has received the SYN message and then sending a SYN plus acknowledgement message and this is the third step of the 3 way handshaking where the client is receiving the SYN plus ACK from server and sending back with the final acknowledgement and once the client has done that client is moving to the established state and it is ready for data transfer.

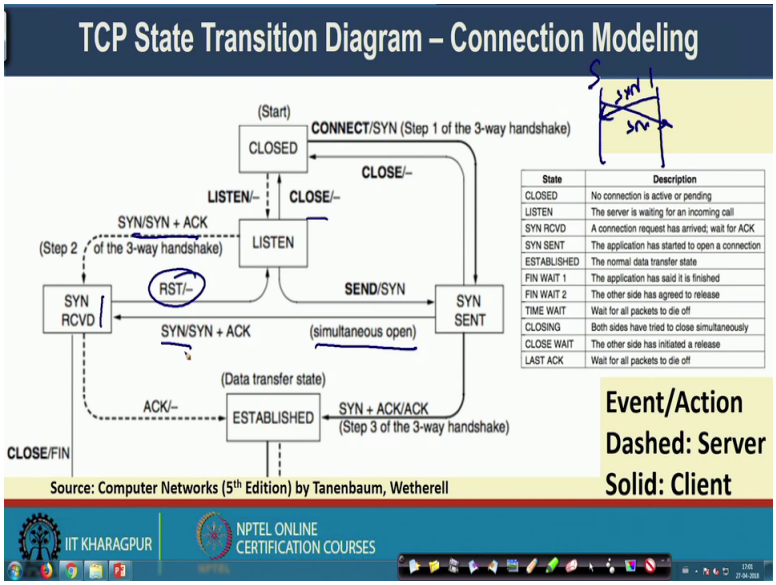
Now, from this listen state again the server can execute a close and close the connection immediately, when the server has received a send SYN message and send back a SYN

plus acknowledgement message, server moves to the SYN receive state. So, from the SYN receive state it can make a reset call and reset the connection to the listen state.

So, this reset call is that server somehow decides that it do not want to continue the connection any more, that is sometime required to prevent the attack whenever you are receiving multiple SYN messages from the same client like a SYN flooding thing to prevent that you can have a reset call or maybe for some application requirement or based on the application programming or certain exception in the application side you want to reset the existing connection.

So, from the SYN received you can call a reset call and again move to the listen state and ignore these SYN you have already received. Now, there is there can be 1 case where both the server and the client are initiating the connection together, so in that case that is we call as a simultaneous open.

(Refer Slide Time: 23:07)

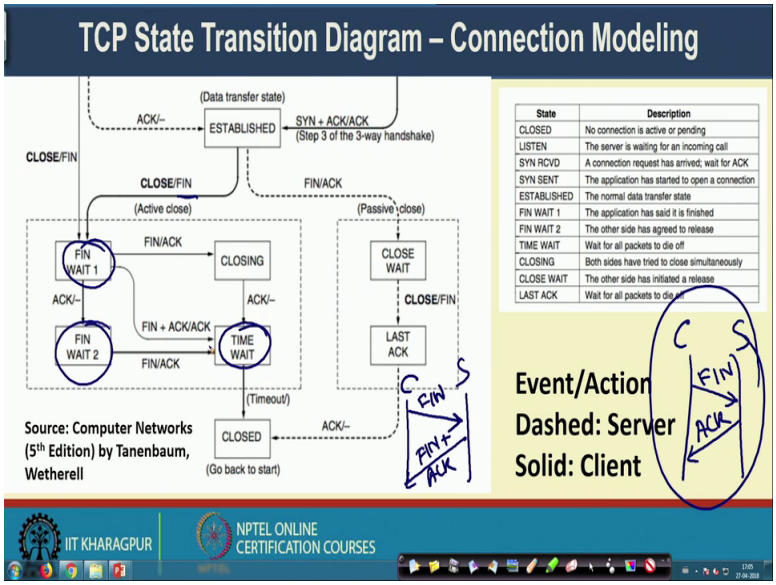


So, it is just like that from the server and client, the client has send a SYN and at the same time the server has also sent a SYN. So, if that is the case like you are getting a SYN message from the server, the client is getting a SYN message from the server because ideally the client should sent a SYN and after that get the client should get a SYN plus acknowledgement. But if it is just getting the SYN message from the server, it sends a SYN plus acknowledgement message and the client can also move to the SYN receive state.

So, it is just like that you have sent a SYN message to the server, but rather than getting a SYN plus acknowledgement the acknowledgement to the SYN that you have sent you are getting a SYN message and not the acknowledgement message, so you are the client is moving to the SYN receive state. At this stage whenever you are getting an acknowledgement message you are moving towards the established state. So, the server is moving server is getting this acknowledgment final acknowledgement message for the 3 way handshaking and it is moving to the establish state.

So, that way through this procedure everyone is moving towards the established state and from this establishment state data transfer can get initiated. So, this is for the connection establishment of TCP that it moves to this multiple states, and finally reaches to the establish state when you can initiate data transfer. Now the data transfer can goes on based on the principle that we have shown you earlier that if established then sent then send data or if established and receive data. And after this connection established state then after this data transfer is over say you want to move to the connection closer state you want to close that connection.

(Refer Slide Time: 24:54)



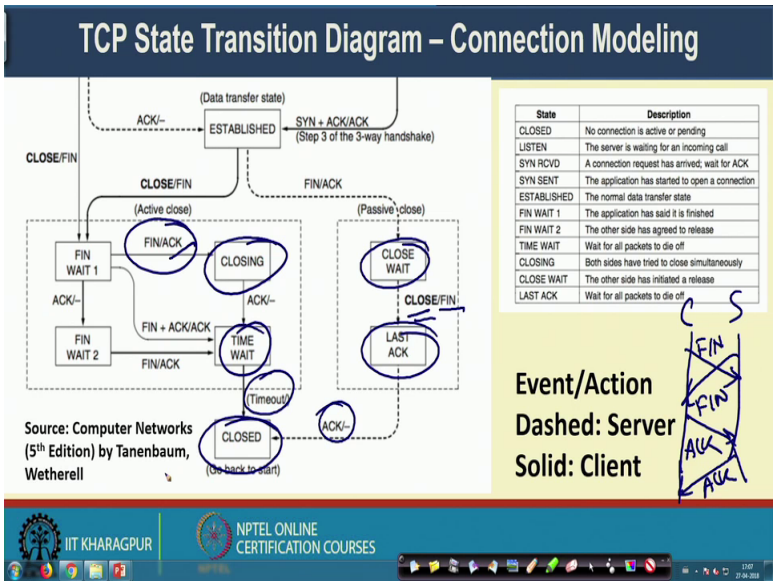
Now, whenever you are wanting to close that connection the client can initiate the connection that particular connection we call it as an active close, because the connection closer is initiated by the client and for the server who is receiving that finish message FIN message we call that as a passive close. So, we have seen that earlier.

Now, in case of the active close, the client send an client execute the close primitive and send a FIN message. So, whenever it has SYN send a FIN message it moves to this FIN wait 1 state, then after sending this FIN message you think of the connection released phase from the client to the server; you have sent a FIN message after sending a FIN message there are 2 possibilities, 1 possibility is that you get a FIN plus ACK. And the second possibility is that the client and the server the client has send a FIN message and it is not getting an ACK it is it is not getting a FIN it is just getting the ACK. So, if it is this case that you are not getting the finish from the server. That means the server is believing that it has more data to send you just receive an acknowledgement message and you move to the FIN wait 2 state from FIN wait 1 state, because you have not received a FIN plus ACK.

Now, if you are receiving the FIN plus ACK message after receiving this FIN plus ACK message you go to this time wait state. So, you remember that for the active connection active closer we have this time out value, where after receiving this FIN plus ACK you wait for a timeout value once the time out happens then you clear the connection, so it moves to this time wait state.

Similarly this FIN wait state it has received the acknowledgement, but it was waiting for getting the FIN from the server, once it get this FIN from the server it sends that ACK and moves to the time wait state. Now there can be another case like it has sent a FIN message to the server, but without getting an ACK it has received another FIN message from the server itself.

(Refer Slide Time: 27:22)



So, it is a case of so this case is simultaneous closer case, where the client has sent a FIN message to the server and server has sent another FIN message to the client without sending the ACK. So that means, the server is believing that it has more data to receive, so in that case the client moves to the closing state by sending a ack. So, you get the finish message because you are anyway ready to finish so you send that acknowledgement message and move to this closing state. In this closing state you are basically waiting for the acknowledgement from the server for this finish message that you have sent.

Now, after that if the server sends this acknowledgement message to you, then you move to the time wait state and after the time out occurs you move to the close state. In case of passive close things are pretty simple that you are in the close wait state because, you have received the FIN message and you have send back with an acknowledgement message and in the passive wait state you finally make a close call, so the server here is making a close call here you sent your own FIN message, server is sending it is own FIN message and waiting for the last acknowledgement message.

So, once it server gets the last acknowledgement message it close the connection and again it goes back to the initial state that is the starting of the connection. So, this is the entire state transition diagram of TCP. The important aspect here is this time wait state that means, after getting the final acknowledgement, in case of the active close the node

which is initiating the closer it will wait for certain timeout duration and once the timeout occurs then only it will close the connection and another interesting state is this closing state where you have sent a FIN message to the server, but rather than getting an acknowledgement you have received the FIN message from the server.

So, it is just like that you want to close the connection the server also wants to close the connection, but server is not immediately acknowledging because, it has a belief that it may receive some more data or it is waiting for some other processing. So, this are the 2 interesting steps here in case of connection closer and this connection closer is interesting because, here our objective is to prevent the data loss as much as possible. Because of a result of this impossibility principle that we talked earlier with the example of these 2 army problem that, if you have a unreliable channel then over that unreliable channel you will be never be succeed to have a protocol of simultaneously closing the connection or getting a consensus over this unreliable channel if the system is asynchronous.

So, we need to go for this synchronous closer and in case of synchronous closer to prevent the data loss as much as possible, TCP has taken this steps and in this particular steps the interesting part is this timeout for active closer that once all the things is over the node which is initiating for the closer it waits for certain amount of time. Once the timeout occurs they close it, but for the passive close you do not require the timeout because, for the passive close the other end has already closed the connection, so you know that it is not going to send anymore data to me.

So, this is the entire connection modeling part of TCP protocol and in the next class we look into the flow control algorithm. So, this connection establishment it has helped you to set the initial sequence number. So once it is initial sequence number has been established then you can use that initial sequence number. So, you are at the established state and with that established state and the initial sequence number that has been agreed upon during the connection establishment phase; you can use that for further data transfer using you flow control algorithm.

So, in the next class we will look into that flow control algorithm in details.

Thank you all for attending this class.