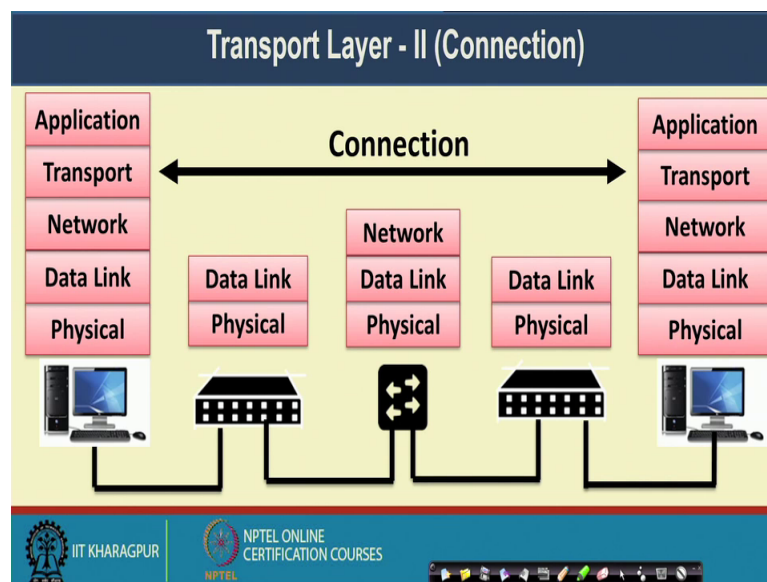


**Computer Networks and Internet Protocol**  
**Prof. Sandip Chakraborty**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 13**  
**Transport Layer – II (Connection) (Contd.)**

Welcome, back to the course on Computer Network and Internet Protocol. So, we are looking into the transport layer and the connection establishment mechanism in the transport layer. So, in the last class we have seen that well whenever you are setting up a logical connection between 2 N host of a network the challenge is the delayed duplicate packets.

(Refer Slide Time: 00:52)

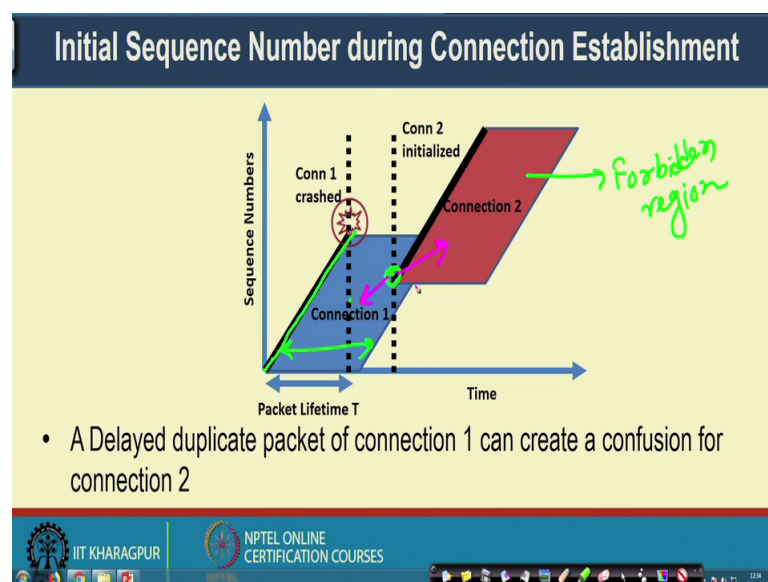


So, the delayed duplicate packets may create a confusion like during the connection establishment phase that whether the packet that you are being received see if you have received the delayed duplicate message whether that delayed duplicate message is or in other words if I say it more clearly like whenever you receive up duplicate message you do not be sure or duplicate connection establishment request you cannot be sure whether that duplicate message is a delayed duplicate of the earlier one. Or the client has crashed and it has reinitiated the connection with the server and that connection request message is coming from that one.

So, what we have looked till now that will the way we can mitigate this problem is by utilizing a virtual clocking mechanism with the help of sequence number where we are utilizing the concept of bite sequence number that every bite in the network will have a unique sequence number. And during the connection establishment phase we need to generate the sequence number in such a way so, that the other end like here in the example a server can correctly identify from a duplicate message that whether this duplicate connection request is the delayed duplicate of the old connection request or it is a new connection request after the client has crashed.

So, to do that it can look into the sequence number field and it can find it out, but we need to ensure here that this connection request sequence number, the initial sequence number that has been utilized for connection request it is not going to re-use within a time duration  $T$  which is the maximum packet life time in the network. So, you need to ensure that with this time of duration  $T$  every instances of a segment or here we are using bite sequence number. So, every instances of a bite with the sequence number is only one such instances outstanding in the network. There are not more than one instances of the same bite in the network same bite means the bite with the same sequence number in the network which can create confusion for the other end for the receiver.

(Refer Slide Time: 03:09)



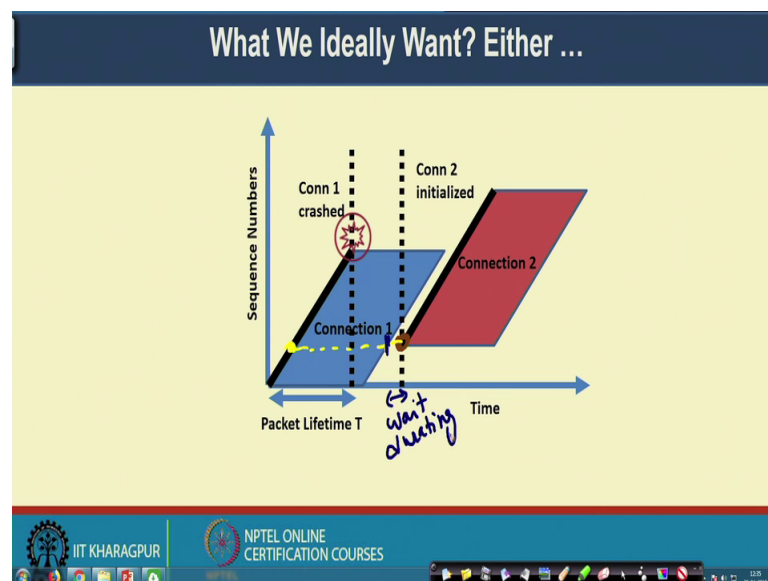
So, we are looking into this problem from the context of this forbidden region. So, what we see that whenever connection one connection selects one sequence number. So, this

particular dark black line indicates. So, this line indicates the sequence number that a particular connection is going to use with the respect of time. Now, if it is using this set of sequence number so, every bite with that sequence number they have a life time here this life time is  $T$ . So, within that life time the packet or the bite can be outstanding in the network.

Now, in a scenario when this connection is being crashed and you want to initialize another connection during that time if this region of the old connection and the forbidden region; so, this region we call as the forbidden region. So, the forbidden region of the old connection and the forbidden region of the new connection if that gets overlapped then there is a they are may be a problem of confusion. Why? Because at this time instance you have you may have two different instances of the same sequence number in the network.

So, one sequence number is from the old connection another sequence number is from the new connection. So, you can have one sequence number from this old connection another sequence number from this new connection and we want to avoid that. Now, to avoid that the solution mechanism that we can employ is something like this, like either you make it separate with respect to time.

(Refer Slide Time: 04:48)

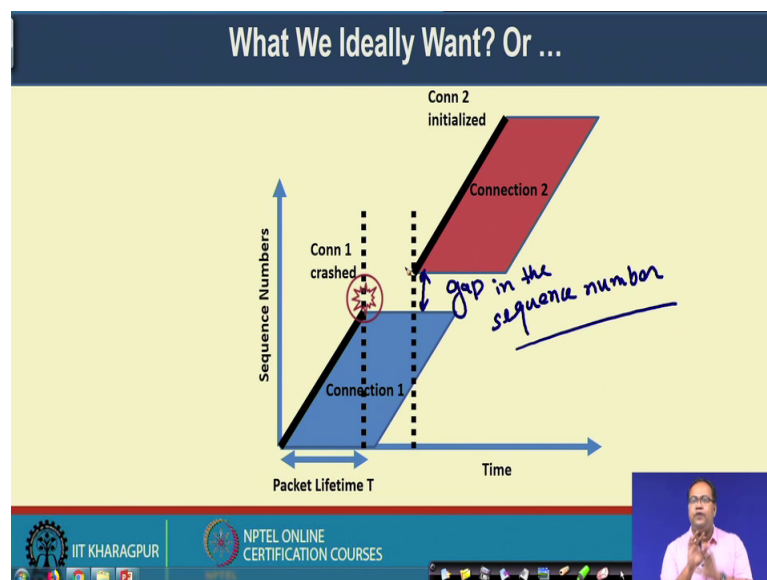


That means, you wait for some amount of time before initializing the new connection such that you become ensured that this particular sequence number which was there say

this sequence number it was completely gone out of the network. So, it is like that this the same sequence number which was there for this connection 1, that sequence number had a lifetime up to here. So, that sequence number is out from the network and that cannot create a confusion invoke.

So, you wait for certain duration. So, here this is the wait duration and then you initiate with say this is the wait duration. So, you wait for this wait duration and then initiate this new connection such that this forbidden region does not overlap with each other. So, if this forbidden region do not overlap with each other, you will be stored that, there is always a single instances of a particular sequence number outstanding in the network. that is one way by shifting the connection in the time skill.

(Refer Slide Time: 06:18)



Another way is to shift the connection another way is to shift the connection in the in the sequence number skill. So, you use the sequence number which is high enough of the sequence number that was utilized for connection 1. So, that you became sure that well the sequence number fill that you are going to use for connection 2 that does not have a overlap with connection 1.

So, here we are utilizing this piece. So, we are we are making gap in the sequence number, such that we become sure that whatever sequence number that we are going to use that particular sequence number has not been used by connection 1. So, this are two feasible way of setting the initial sequence number in the network.

(Refer Slide Time: 07:11)

### Connection Establishment – Handling Delayed Duplicates

- Receiver receives two segments having the same sequence number within a duration  $T$ 
  - One packet must be the duplicate
  - The receiver discards the duplicate packets.

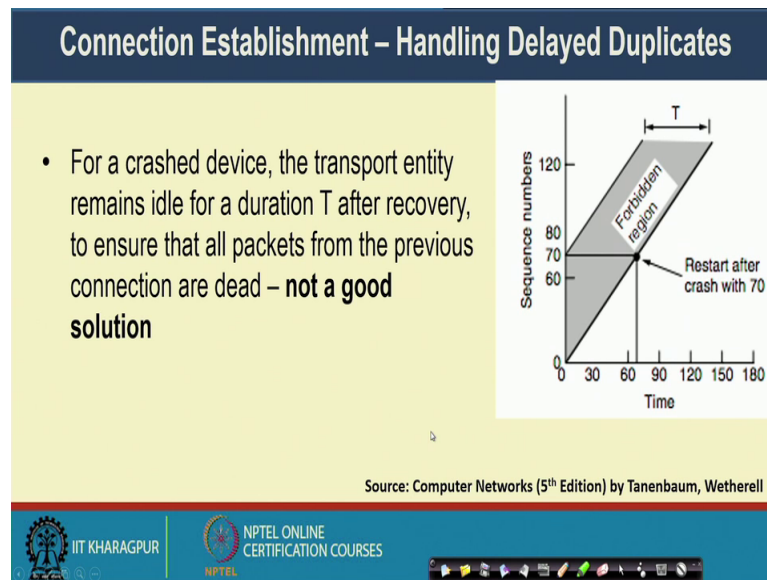
Source: Computer Networks (5<sup>th</sup> Edition) by T

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

Now, let us see that. How you can handle the delayed duplicates during the connection establishment by mitigating this two problem. So, if we ensure this kind of things now when a receiver receive two segment having the same sequence number within a time duration  $T$  the receiver knows that one packet must be the duplicate. Say it happens that well the sequence so, you are always ensuring that within that time duration  $T$  the receiver cannot receive a packet cannot receives a two different packets with the same sequence number or two different bites with the same sequence number.

Now, if the receiver is receiving two different bites with the same sequence number with the duration  $T$  then the receiver can correctly decode that the second one is the delayed duplicate of the first one or the visa versa anything can happen and, but you can you in that case the receiver can accept one and can describe the second one that it has received.

(Refer Slide Time: 08:23)



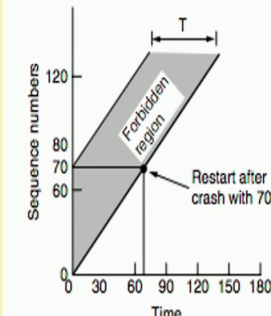
Now, for a crashed device, then the transport entity that remains idle for a duration  $T$  if you are just utilizing this time skill at time period based sequence numbering to ensure that all the packets from the previous connection are dead. So, here actually we are utilizing the first solution that I was mentioning that you wait in the time skill and ensure that by that time all the instances of the previous sequence number is dead from the network.

So, whenever you are going to use a sequence number there is no possibility that there are two bytes with the same sequence numbers are outstanding in the network, but this is not a good solution because you have to wait for certain amount of duration. So, for that you can also choose the sequence number in such a way we will see that in the constant of the TCP that you can choose the sequence number in such a way so, that you are significantly high above the forbidden region of the previous one and you can be sure that the sequence number has not been utilized by the previous connection, for connection establishment.

(Refer Slide Time: 09:34)

### Connection Establishment – Handling Delayed Duplicates

- **Adjust the initial sequence numbers properly** - A host does not restart with a sequence number in the forbidden region, based on the sequence number it used before crash and the time duration  $T$ .



Source: Computer Networks (5<sup>th</sup> Edition) by Tanenbaum, Wetherell

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

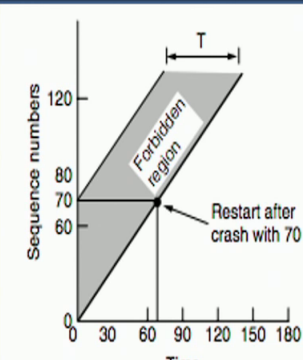
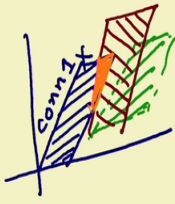
Now, the solution here is you adjust a initial sequence number properly; that means, a host does not restart with the sequence number in the forbidden region based on the sequence number it used before crash and at the time duration  $T$ . So, it is just like that if the if you have if the system has crashed then whenever whatever sequence number was there the new sequence number that you are going to generate you generate in such a way, so that is above the previous sequence number. So, we will see that how we can generate this particular sequence number.

(Refer Slide Time: 10:12)

### Packet Sequence Numbers are Out of the Forbidden Region

Two possible source of problems

1. A host sends too much data too fast on a newly opened connection



Source: Computer Networks (5<sup>th</sup> Edition) by Tanenbaum, Wetherell

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

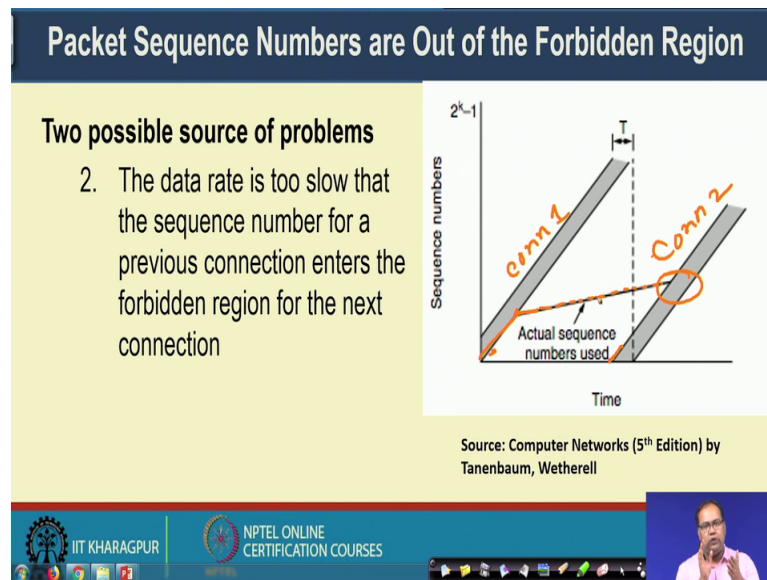
Now, there can be two different source of problems whenever there are two connections like whenever one connection has crashed and another connection is going to use a initial sequence number field. So, let me just give you one example here. So, it is just like that one connection it has used this sequence number and then it got crashed and this is the forbidden region for this old connection, say I name it as connection 1.

Now, say there is second connection. The second connection is start from here with the initial sequence number. Now, if the second connection starts from here with the initial sequence number and follow this line the sequence number space then this would be the forbidden region, there is no overlap my life is happy, but if this particular new connection starts sending data at a very fast rate. So, if it follows this line rather than the dotted line that I have drawn earlier then, you see that these becomes the forbidden region and here for some packets you have a overlap. So, you have certain overlap here in this region.

So, in this region there can be still the confusion about the sequence number whenever you will receive a packet whether the packet belongs to connection 1 or this new connection, connection 2 so, if you increase the sequence number space too fast then that can become a problem. So, that is why the sequence number need to be increased at a constant rate or at a bounded rate. So, that the sequence number space of this increase of the sequence number for the new connection does not overshoot the sequence number space of the previous connection. So, that they do not overlap with each other.



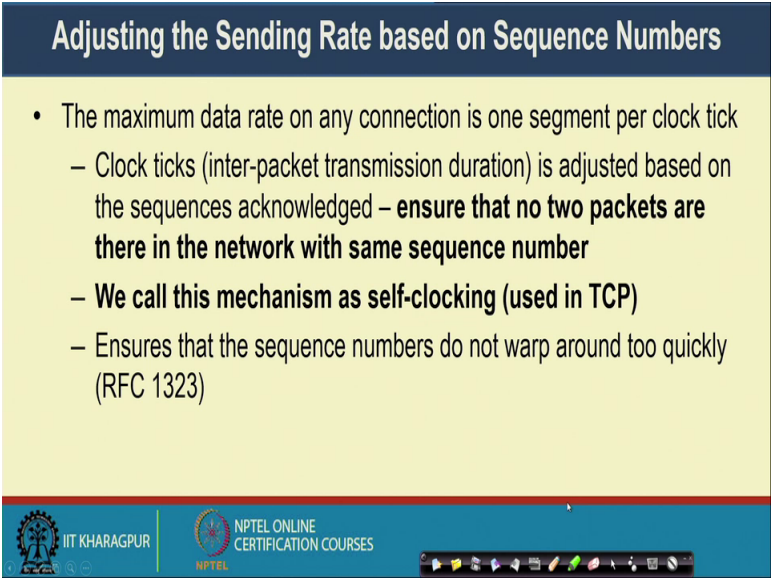
(Refer Slide Time: 12:33)



Another problem is there for selecting this initial sequence number that the data rate is too slow. If the data rate is too slow like in this example say the initial sequence number was used like this then the data rate was too slow. So, it started generating the initial sequence number at a very slow rate and after that it crashed and the new system it just start using this initial rate that say if I name it as connection 1, that connection 1 used and this one as connection 2 that connection was 1 used then again there is a possibility of having a overlap here.

So, both are fast connection and a slow connection can create a problem. So, we need to ensure that the sequence numbers are always generated at a bounded rate. So, that this kind of overlapping of sequence numbers between two connection does not happen.

(Refer Slide Time: 13:30)



**Adjusting the Sending Rate based on Sequence Numbers**

- The maximum data rate on any connection is one segment per clock tick
  - Clock ticks (inter-packet transmission duration) is adjusted based on the sequences acknowledged – **ensure that no two packets are there in the network with same sequence number**
  - **We call this mechanism as self-clocking (used in TCP)**
  - Ensures that the sequence numbers do not warp around too quickly (RFC 1323)

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, how will you do that? So, you can do that that you can bound the maximum data rate that you can sale over a transport protocol. So, the maximum data rate on any connection we bound it as one segment per clock tick. So, here we used the hardware clock, but only the hardware clock of my machine. So, here we will see that we do not require the synchronization of the hardware clock across multiple machine. So, the hardware clock of a single machine will serve my purpose.

So, with every hardware clock tick so, the clock tick is the inter packet transmission duration the clock ticks is adjusted based on the sequences that is acknowledged. So, TCP uses this concept of self clocking or a virtual mechanism that whenever you received an acknowledgement. So, this is something like a mix of the connection establishment under flow control mechanism for handling the sequence number space that whenever you are receiving an acknowledgement during that time you make a tick that you generate new packet or new segment with a new sequence number.

So, you ensure that the no two packets are there in the network with the same sequence number. So, this also ensures that the sequence number space that do not wrap around too quickly. So, you have a finite sequence number space in case of TCP kind of protocol, you have 32 bits sequence number. Now, if you have a 32 bits sequence number then you can used to 2 to the power 32 different sequence numbers. So, you need to ensure that this entire sequence number speed does not wrap around too quickly.


So; that means, if you are sending data at a very high rate it may happen that you are generating the data in such a rate that you have within that time duration  $T$  or even before the time duration  $T$  you have finished this entire 32 bit of sequence number space. So, that may create a confusion, because that may create a confusion. So, you want to prevent that and to prevent that you want to regulate the sender flow as well apart from the receiver flow normally with the flow control algorithm we coordinate between these a sender flow and receiver flow but, whenever we are generating the sender packets you also want to ensure that the packets which are being generated from the sender they follow certain kind of I will not say it is a constant rate rather they are in within a bounded rate.


So, that is why the application is generating data as it is own rate and the data is being buffered at the transport layer, buffer and the transport layer picks up the packets from there, picks up the bytes from the there and generate the segments it predefined bounded rate. So, that the sequence number space does not get overlapped with each other. So, we will look all these mechanism in details for the time being it may be little unclear or you may have a little bit doubt that the things will be much cleared to you whenever we will look into the flow control algorithms in details, where we look into that how flow control actually helps in adjusting this entire sequence number space.

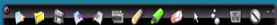
(Refer Slide Time: 16:40)

### Adjusting the Sending Rate based on Sequence Numbers

- **We do not remember sequence number at the receiver:** Use a **three way handshake** to ensure that the connection request is not a repetition of an old connection request
  - The individual peers validate their own sequence number by looking at the acknowledgement (ACK)
  - **Positive synchronization among the sender and the receiver**

 IIT KHARAGPUR

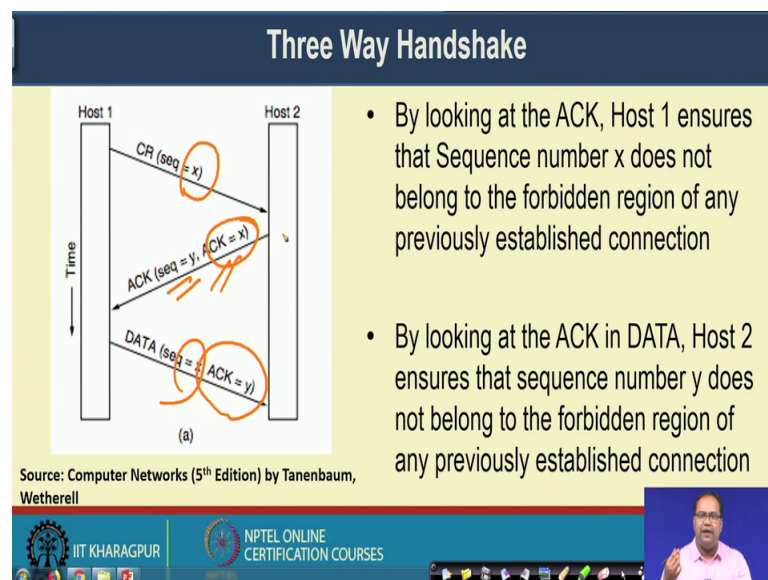
 NPTEL ONLINE  
CERTIFICATION COURSES



Now, the second thing is that, if you remember that the trampling sense from the trampling sense proposal that our first requirement was to have a way to selecting the initial sequence number. So, here what do you want that we do not want to remember the sequence number at the receiver side. So, the receiver does not want to remember the old sequence number rather the sender will manage its own sequence number.

So, for that we use a three way handshake mechanism, to ensure that the connection request is not a repetition of the old connection request. Now, the individual peers they validate their own sequence number by looking into the acknowledgement. So, this provides the positive synchronization among the sender and the receiver.

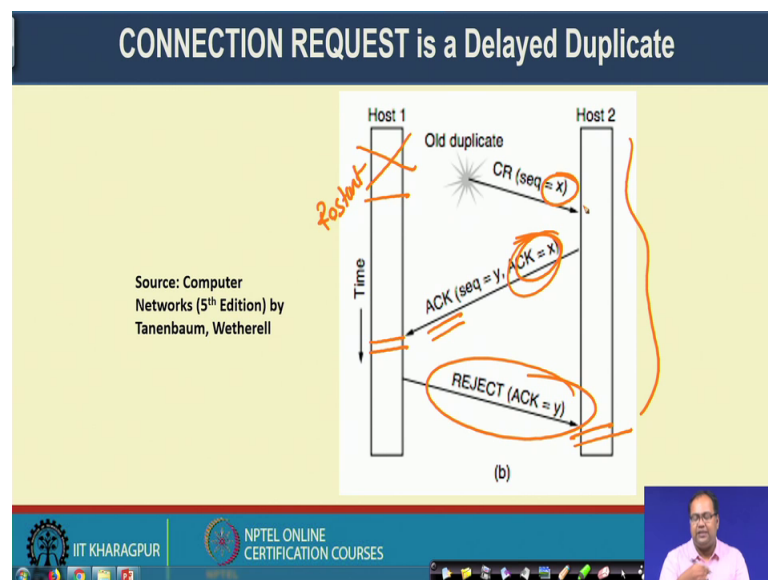
(Refer Slide Time: 17:28)



So, let us look into how this three way handshaking works. So, whenever host 1 is sending a connection request message. Host 1 sends the sequence number value  $x$ , so, here host 1 sends the sequence number value  $x$  and host 2, it sends back with an acknowledgement. With the acknowledgement it also put that sequence number  $x$  and it also. So, in case of transport layer normally our connections are bi directional because the connections are bi directional with this particular acknowledgement the host 2 also sends another acknowledgement, sends another sequence number it for reverse connection from host 2 to host 1 that is the sequence number  $y$ .

Now, whenever the host 1 receives an acknowledgement with sequence number  $x$  host 1 can verify whether this sequence number which was there in the acknowledgement it is the original sequence number or not it is the sequence number of the connection request message that is sent it is corresponding to that or not. If it is then it can send the data with that sequence number  $x$  and at the same time it can also send acknowledgement for the sequence number space or the sequence number that was proposed by host 2 for host 2 to host 1 data transfer. So, this three way handshaking ensures that their all the delayed duplicates are correctly identified by both host 1 and host 2. So, let us see that how with this mechanism they can correctly identify delayed duplicate of the sequence numbers.

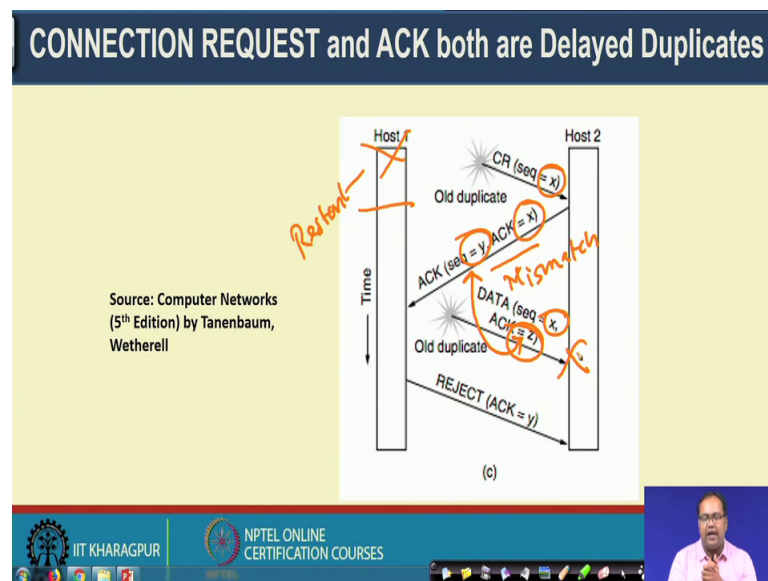
(Refer Slide Time: 19:12)



So, let us see a case when the connection request is a delayed duplicate. Now, if the connection request is a delayed duplicate so, the host 2 has received the delayed duplicate connection request with the sequence number  $x$ . So, it sends back with an acknowledgment of that  $x$ . Now, host 1 can verify that this particular acknowledgement that it has received it is not for a connection request that it has sent. So, that connection request was a old duplicate that it has sent long back and now host 1 do not want to use that connection anymore, maybe host 1 has crashed here and then it got and restarted here; then it got restarted here. So, it do not want to use that old connection request that it has requested earlier.

So, host 1 can find it out and if it finds it out this acknowledgement is a delayed duplicate then it can send a reject message by looking into the reject message host 2 can identify that the connection request that was sent that it was received it is not going to be accepted by host 1 anymore. So, there is no point in establishing the connection.

(Refer Slide Time: 20:27)



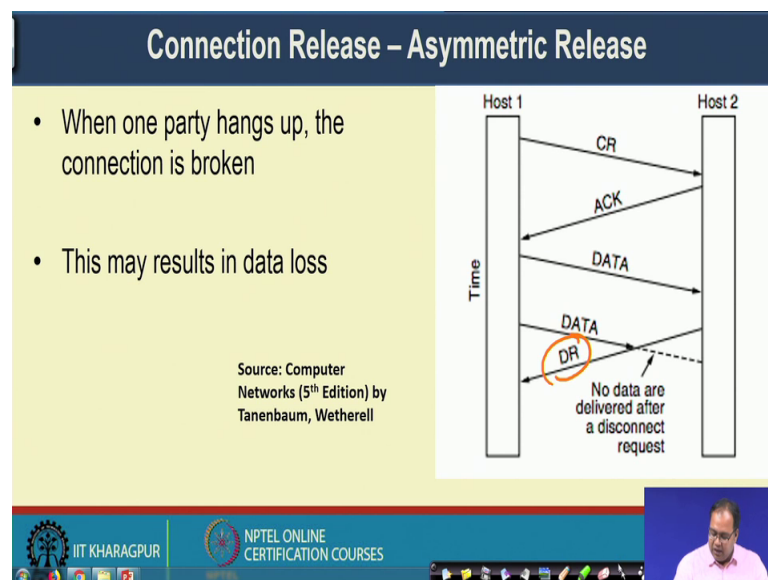
Now, let us look into another case when both the connection request and the acknowledgement that delayed duplicates. Now, when the connection request is a delayed duplicate and host 2 sends back an acknowledgement here with this sequence number x which it was received as a part of the connection request message and it proposed with a new sequence number y during that time it gets a reject message. But for the old duplicate that it has sent say it has received one acknowledgement here, this acknowledgement says that it is a sequence number for x, but the acknowledgement number says that will the acknowledgement number is z.

Now, if you look into this three way handshaking mechanism these acknowledgement numbers should corresponds to the acknowledgement number should corresponds to the sequence number that was proposed by host 2 in it is acknowledgement. Now, these two are not matching. So, host 2 will reject this duplicate acknowledgement. At the same time host 1 whenever it has received this acknowledgement message and it finds out that

it do not want to use this acknowledgement it is anymore because it has crashed here and say restarted here, then it can sends the reject message.

So, with this way you can identify that both the connection request was a delayed duplicate that was identified by host 1 and it sends a reject message and host 2 whenever it looks a different acknowledgement number, acknowledgement gate acknowledgement number gate compare to the one which it has sent with its own acknowledgement whenever it finds out that there is a mismatch here; there is a mismatch here it simply discard this acknowledgement. So, that way you can properly differentiate between with the help of this sequence number between the normal connection request messages and a delayed duplicate connection request messages.

(Refer Slide Time: 22:44)



Well; so, that was the all about the connection establishment. So, what we have broadly seen here just to give you a summary of the entire procedure, the connection requesting. So, what we have seen that because packets can get dropped in a packet switching network, there can be arbitrarily delayed transferring the packet, there can be lose because of this reason, there is always a possibility of having a delayed duplicate.

Now, what we have learned till now that my major problem is to select the initial sequence number for connection establishment. Once this initial sequence number is established then the flow control algorithm takes care of maintaining the sequence number for the data pack is that will look later on that how flow control algorithm

actually helps you to set up the sequence number for the data packets or the data segments in the context of the transport layer.

But, the challenge here is to select a initial connection request requesting such a way, so that you can ensure that the forbidden region of a new connection does not get overlap from with the forbidden region of an older connection. Where both the new connection and older connection are initiated on the same application at the same source destination pair. So, they are likely to use the same port. So, in that case our objective is to separate out a normal connection request from a delayed duplicate connection request and in that case we take the help of a sequence number.

Now, we have looked into that how to chose a initial sequence number, but whenever you are choosing the initial sequence number you have to ensure that well your initial sequence the sequence number field does not increase too fast or too slow such that it gets overlapped with another connection. In that particular context this rate of control of sequence number that is taken care of by the flow control algorithm and we ensure that well the packets are generated or the bytes are not, I will not say generated that the transport layer transmits the byte transfers the byte in such a rate, so that it is not too slow or not too fast all the connections follow almost a bounded rate.

But, whenever you are selecting the initial sequence number you can use this three way handshake mechanism for selecting the initial sequence number such that if any or connection request is the delayed duplicate or the connection request or the corresponding acknowledgement is the delayed duplicate you will be able to differentiate between the old connection and the new connection by ensuring that the sequence numbers are not generated from the forbidden range of the previous connection. And you are making sure that no two bytes in the networks are having same sequence number between the same source destination pair coming from the same application at the same time instance.

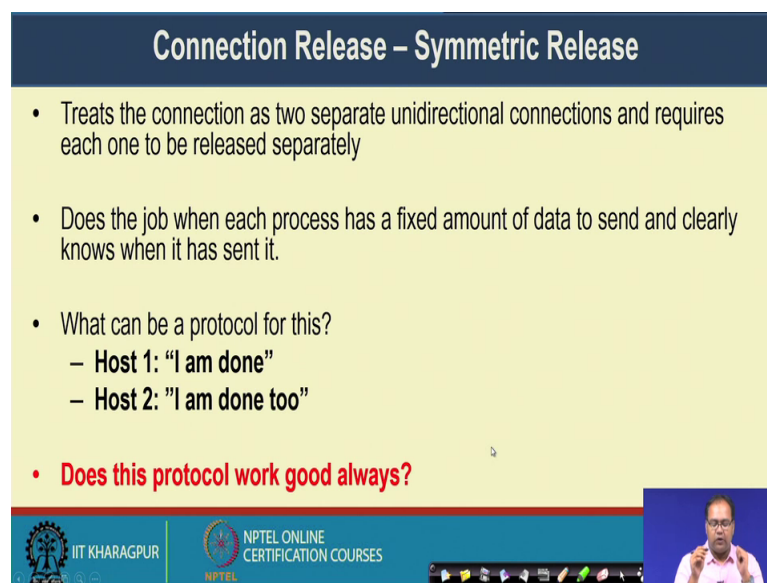
Now, let us look into the connection release. Connection release is little bit easier compared to connection establishment because we do not have the problem of sequence number here, but here we have different problem. So, there can be two type connection release; so one we called as a asymmetric release. So, the asymmetric release says that when one party hands up the connection is broken. Now, it is just like that whenever host



1 is ready to or whenever host 1 is done transferring the data host 1 breaks the connection say. It may happen that host 2 now wants to close the connection; host 2 simply sends the connection released message. So, here it is DR data release message host 2 sends the data release message and host 2 goes to sleep.

Now, even if host 1 has some data to send to host 2 that particular data if any host 1 sent it host 2 may not be able to receive that data. So, there would be a possibility of data loss with this concept of asymmetric release.

(Refer Slide Time: 27:13)



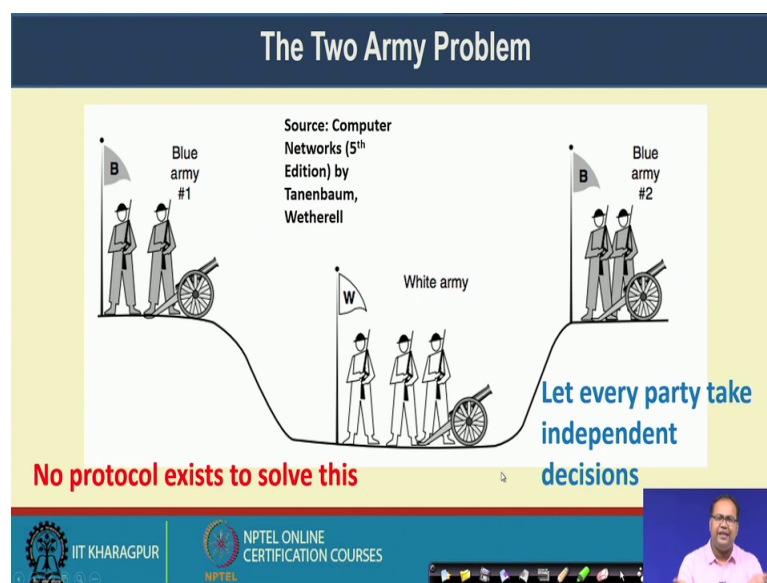
The slide is titled "Connection Release – Symmetric Release" in a dark blue header. The main content area is yellow and contains four bullet points. The first three are black, and the fourth is red. A small video inset in the bottom right corner shows a man speaking. The footer is blue and contains the IIT Kharagpur and NPTEL logos.

- Treats the connection as two separate unidirectional connections and requires each one to be released separately
- Does the job when each process has a fixed amount of data to send and clearly knows when it has sent it.
- What can be a protocol for this?
  - Host 1: "I am done"
  - Host 2: "I am done too"
- **Does this protocol work good always?**

Now, we can have another variant of these connection release which we call as the symmetric release. Now, in case of a symmetric release you treat the connection as two separate unidirectional connection every individual connection is treated as two separate unidirectional connection and it requires that each one to be released separately.

So, when both one is released the connection then the final connection will get released. Now, this is good when this particular symmetric release is good when each process has a fixed amount of data to send and it clearly knows that when it has sent it. So, it knows it has an idea that when it has send a particular data, but the question comes that can we design a protocol for the symmetric release? So, let us look into a very simple protocol that host 1 will say that, I am done; host 2 will say that I am done too. So, when both are saying that I am done and this one is saying I am done too they will release the connection. Let us see that whether this protocol works good always.

(Refer Slide Time: 28:23)



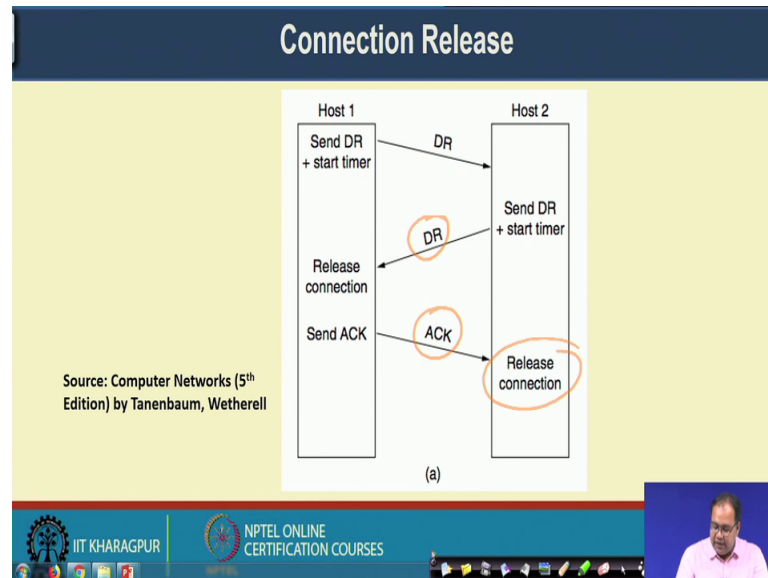
So, we map this protocol in the context of a problem called two army problem. So, we have a white army which is there in a valley and the blue army which was there in the hill now you see that the total fighters in the blue army it is more than the white army, but they are separated now, they are in the two part of the hill. So, they need to communicate with each other to make sure that both of them attack simultaneously; both of them are able to attack simultaneously then they only they will be able to defeat the white army otherwise they will be not able to defeat.

Now, the problem here is that if the blue army wants to send message called attack this blue army 1 wants to send that message to blue army 2 they have to go back the valley which is the vulnerable position. So, it may always happen that one soldier of white army is able to see that the soldier of the blue army and kill that person and the message is not delivered in the other way. So, the environment is unreliable.

Now, whenever the environment is unreliable you can see that you will never be able to make protocol, correct protocol to solve this particular problem that both the blue army will come into consciences and they will be able to attack the white army simultaneously because whenever you are you are sending one soldier of blue army where this valley blue army 2 is may not get that particular soldier message from that particular soldier and they will not be able to sure that whether to make an attack or what is the current

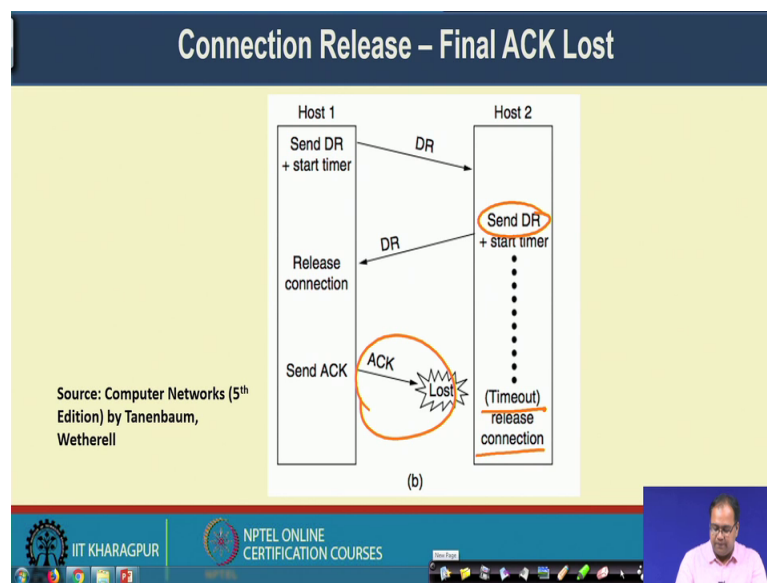
condition. So, we cannot have one protocol to solve this particular problem. So, in this case the best way you can do is that let every party take their own independent decisions.

(Refer Slide Time: 30:22)



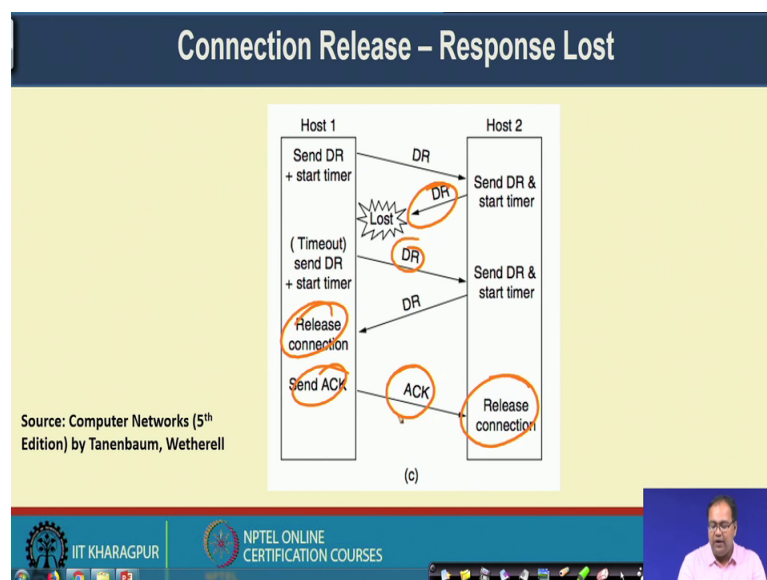
So, that is the protocol we implement here that every individual host so, host 1 it will send the data release message and then it will start the timer. Similarly, host 2 it will send the data release message and it will start the own timer, whenever it is receiving the message from host 2 within this time out value it will release the connection and it will send the acknowledgement, and if host 2 is getting this acknowledgement message within this timeout value it will release the connection.

(Refer Slide Time: 30:59)



Now, let us see that how this protocol works when the acknowledgement is lost. If this final acknowledgement is lost say this final acknowledgement is lost; if this final acknowledgement is lost then this host 2 it has started its timer after sending the data release message it will not get the acknowledgement. So, it will wait for the time out value whenever the timeout will occur it will release the connection.

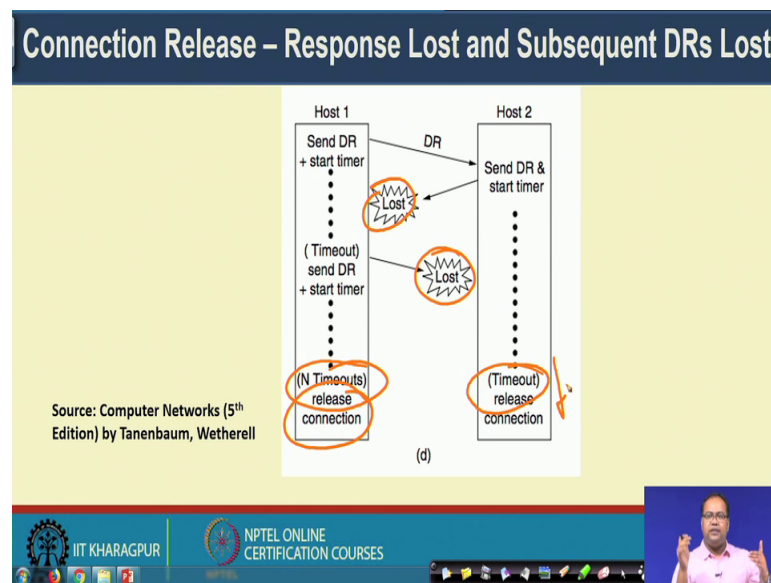
(Refer Slide Time: 31:28)



Then say this particular data release message from host 2 got lost. So, if this data release message from host 2 got lost. So, here host 1 has sent a data release message with host 2

received and host 2 has sent another data release message. So, host 1 will get a timeout, after it will get a timeout it will again send the data release message. So, once it sending a data release message host 2 will receive that message it will again start the timer send the data release message. So, once it is getting these data release message it will release the connection send the acknowledgement and once host 2 will get that acknowledgement it will release the connection.

(Refer Slide Time: 32:10)



Now, let us see another scenario when both the data release and the acknowledgement are lost. So, here this data release is lost and as well as the acknowledgement is lost. In this case so, if both this messages are getting lost then both the node will wait for a timeout value. So, here you can specify that host 1 it will try for N different timeouts and once this N T different timeout occurs it will release the connection and host 2 it will wait for similarly for the timeout value once the timeout occurs it will release the connection.

So, here we are we are basically making the protocol from a asymmetric view that we will wait for certain timeouts and if you are not able to solve the problem with that timeout value then you independently release the connection, but as we have seen earlier in case of asymmetric connection there is always a possibility of having a data loss. So, this particular lecture it has given you the idea about the fast services fast of the services which is being supported by the transport layer of the protocol stack, where we need to

establish the connection between two remote host which is a kind of logical pipe to establish the hello messages between two end and because of this reliability problem in the network we see that ensuring the connection establishment is a challenge. You can you can argue that well I have the reliability protocol then why should I bother about all this difficulties during this connection establishment.

But, you remember that the reliability only comes that when you have set up this initial sequence number. So, then you can apply your flow control and the reliability protocol which will look later on we call them as the automated request protocol ARQ protocol. So, this ARQ protocol they can take care of the loss by retransmitting the packet because they have a reference frame through which you can it can utilize the sequence number field.

But, whenever you are setting up the initial connection during that time you do not have any initial difference frame like from where you will start the sequence number. If every connection starts from sequence number 0 then that can be a problem because of that forbidden region concept that we have looked at here.

Now, to solve this particular problem we have seen that well, by utilizing this self clocking mechanism through the hardware clock or by clocking from the acknowledgement you can generate the sequence number fields ensuring that the sequence number of a connection for the same source destination pairs with the same port they do not overlap with each other. And at the same time you are ensuring the sequence number in such a way it should be higher enough from the previous sequence number, so that they do not get overlap and finally, once this sequence number is established to this three way handshaking mechanism and, during that time you have seen that if there is a loss or a delayed duplicate of the messages the other ends will be able to correctly decode that.

In the context of connection release you have seen that well symmetric release is a good option asymmetric release is a good option, but you cannot design a protocol for asymmetric release in a in a unreliable channel. So, that is why we go for a symmetric release with the timeout value. So, you will try to make a symmetric release if you are not able to do that within certain number of timeout then you forcefully close the connection. There is always a possibility of data loss, but as I have mentioned earlier

there is always a tradeoff between the performance and the correctness. So, here we are not going for a protocol which will be completely correct. There can be certain amount of data loss always due to this symmetric release mechanism. But, our target is to minimize it as much as possible by utilizing that timeout value. The timeout ensures that whatever packet that has been sent by the other end that will reach at the destination within that timeout value.

So, that is all about the connection establishment and connection release. In the next class we will look into another service in the transport layer. So, thank you all for attending the course.