

Storage Systems
Dr. K. Gopinath
Department of Computer Science and Engineering
Indian Institute of Science, Bangalore

Communication Protocols for Networked Storage Systems

Lecture - 08


**FC Vs. 10GB Ethernet and alternatives, iSCSI Protocol -
introduction/configurations/operation semantics/working/flow control &
throughput, TCP Protocol - background on throughput & congestion control, ways
of achieving higher throughput in TCP, practically achieving better throughput &
congestion control in TCP with tcp reno/scalable tcp/htcp/bic cubic. Achieving Zero
Copy Architecture in iSCSI with RDMA over TCP, Infiniband Protocol, Summary
of Storage protocols**

Welcome again to the NPTEL course on Storage Systems. In the previous class we took a look at fiber channel and we looked at some other issues regarding fiber channel especially bandwidth aspects, how much throughput it can be (Refer Time: 00:43). In today's class we will continue, but we are start looking at instead of the fiber channel interconnect.

(Refer Slide Time: 00:53)

FC vs 10GEth

- Port cost of FC historically higher than Ethernet
- Many expect 10GEth to become dominant
 - Much lower costs
- Encapsulate FC frames in ethernet pkts
 - FCoE
 - FC based software does not change much
 - Increasing usage
- Or, drop FC completely and move to TCP as transport for SCSI



We start looking at Ethernet can interconnect and then we will go into something called SCSI, another protocol which is layered over Ethernet, SCSI protocol layered over Ethernet.

Now, if you look a fiber channel historically it has been very expensive because it is a you might call it a niche market for storage only and typically the port cost of fiber channel has been much higher than Ethernet almost a factor of about I think about 10. So, invariably Ethernet has vanquished other protocols. So, all the other protocols are usually become extinct. Where Ethernet has been becoming stronger over speed of time because of the volumes and because of its simplicity of the let us say the basic protocol.

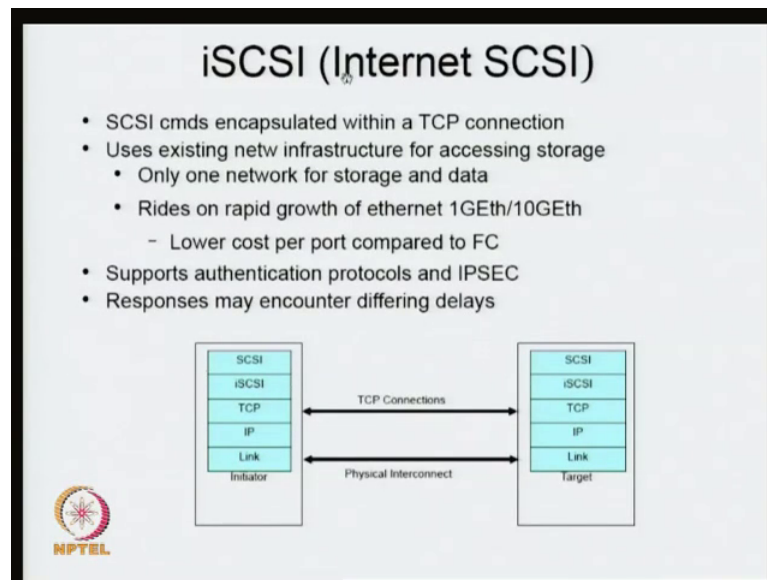
So, generally people believe that from 1 gigabit Ethernet to 10 gigabit Ethernet is when it happens it will become very low cost. Therefore, everybody is thinking where is there is a way to lower the cost of storage you might have to move to something like 10 gigabit Ethernet. Of course, 10 gigabit Ethernet is already is quite strong in the core part of the network is internet, but in a department of setting it is still not that common.

So, if and when this happens when the transition happens then you might be able to easily encapsulate fiber channel frames, Ethernet packets. This is what is called FCoE. The advantage of this particular model is that since lot of the enterprise storage is based on fiber channel. Basically, this software does not have to change all you have to do is to change the some of the switches and some of that is adopters.

So, lots of people believe that this might actually happen. So, you can see lot of increasing usage, but it is too early to say what is going to happen. Other option is, if you really want to exploit what is generally believed the 10 gigabit connect will actually become very predominant. Then you can probably drop the storage protocol specifically crafted for storage the fiber channel protocol and move to some other protocol which can be also used to transport SCSI. One option is to go to TCP IP ok.

So, when idea is to use TCP IP was the transfer protocol.

(Refer Slide Time: 03:43)



So, the iSCSI actually stands for internet SCSI; that means, that my device can be anywhere on internet and it should speak this SCSI protocol. So, there is a I have a SCSI device I should able to access it on the internet. So, the idea here is you take this SCSI commands encapsulated within a TCP connection. What is the advantage? You can use existing network infrastructure. This only one network for storage in data and as I mentioned before, it can ride on the rapid growth of Ethernet infrastructure; nowadays, here or something called metro Ethernet that means Ethernet that is available in a city ok.

And if you have metro Ethernet in the city you can negotiate with the providers for 300 megabit per second or 400 megabit per second etcetera. Those kinds of bandwidths you can negotiate and all of all of this of course, as I mentioned in the previous class goes on fiber. So, the basic idea is that iSCSI could be lower cost per port compared to fiber channel. One advantage of going to internet is that it supports authentication protocols and specifically IPSEC ok.

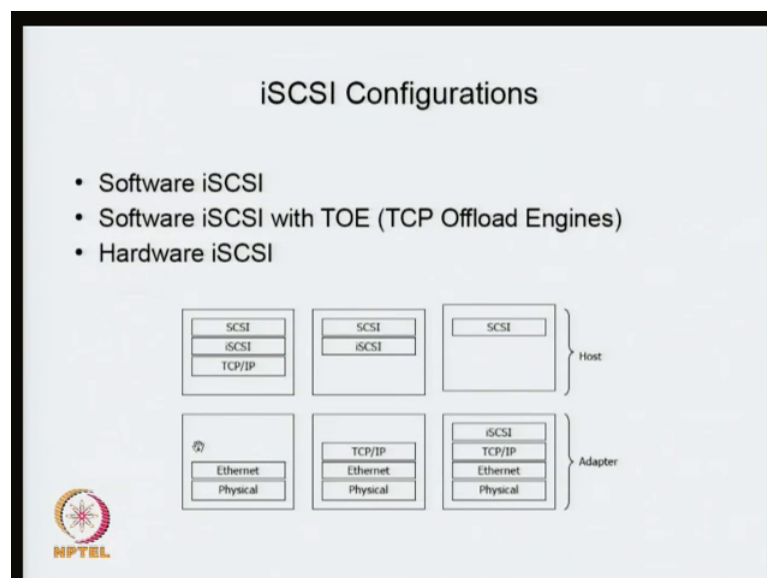
Now, if you look about fiber channel it was designed without security in mind because general assumption was that fiber channel kind of devices will sit in a data center and that has got physical security. Whereas, if you look at internet is always been in the beginning it was designed for let us say regular use without thinking what security, but later the security aspect has been added on to it and now you have something like IPSEC.

Even though there are problems in IPSEC, it is still a good protocol. It is may not be the best, but it is a good enough protocol and lot of people are using it ok.

So, the thing is if I go to iSCSI one advantage I have is that I can just directly use the security mechanism that is available for IP and a good thing about going to IP is that it is a wide lings protocol. So, whatever advantages you get because of the widespread option of IP all of it naturally comes to you to or you are SCSI also. That is one thing which is very strong with respect to going to iSCSI the only difficulty could be that internet being assemblage of widely different types of devices and interconnects you may not be able to guarantee certain let us say delays for example, you might have different patch or delays this is a big problem. If there is some way to handle this problem iSCSI could be a real good contender. So the basic issue is that internet is highly heterogeneous.

So, how do you ensure that you have reasonable guarantees with respect to delays? Diagrammatically you can look at iSCSI as follows you will have a physical interconnect between two links at the link layer of course, your IP and then you have a TCP connection actually multiple TCP connections between at the TCP level. On top of it is iSCSI which is basic in the protocol which essential encapsulates SCSI on to using TCP and so, essentially what we are talking about is SCSIs application. iSCSI is some kind of presentation layer and TCP is the protocol. Let us see what has happened.

(Refer Slide Time: 07:47)

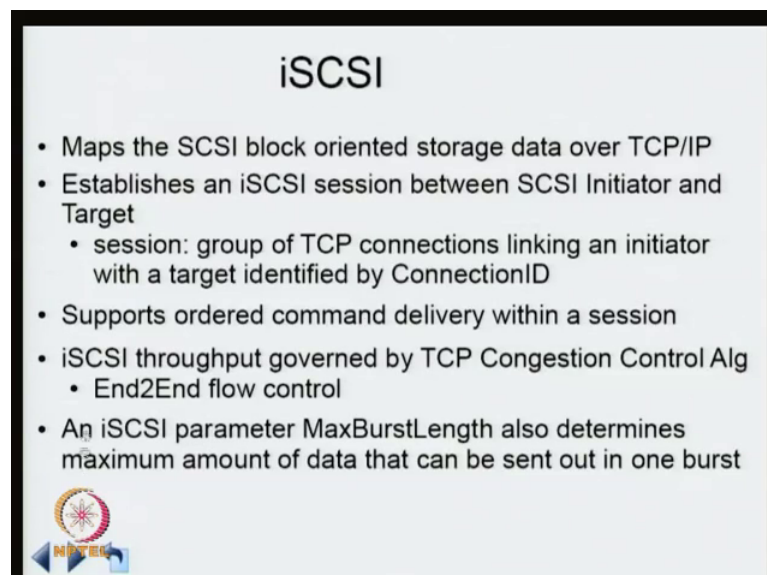


Again there are varieties of iSCSI possible. You can have a purely software iSCSI. That means that their host does all the processing with respect to both TCP IP processing as well as iSCSI. In the sense, this is completely done in software whereas, the adapter is a regular Ethernet card and whatever what you might call vanilla adapter card you can directly use or you can attempt to move some parts of the functionality that is in software into hardware. Firstly, because this part of it can be especially the TCP IP part of it is very expensive to be done in software.

So, that is basic idea about what is called TCP offload engines. A TCP offload engines basically take care of the TCP processing in hardware and then the software only has to handles SCSI and iSCSI part. So, this is one other type of adapters. Now for the final thing would be that the host only speaks SCSI protocol and all the adapter which is connected to the CPU. It speaks SCSI protocol directly, but using iSCSI as the medium as the way to send commands in the C commands.


So, this also is possible. So, there is a lot of possible configurations. This being lowest cost in terms of lowest and slow. This could be where in the more expensive one and probably much faster.

(Refer Slide Time: 09:32)



iSCSI

- Maps the SCSI block oriented storage data over TCP/IP
- Establishes an iSCSI session between SCSI Initiator and Target
 - session: group of TCP connections linking an initiator with a target identified by ConnectionID
- Supports ordered command delivery within a session
- iSCSI throughput governed by TCP Congestion Control Alg
 - End2End flow control
- An iSCSI parameter MaxBurstLength also determines maximum amount of data that can be sent out in one burst



So, again just to reiterate iSCSI what we do is we map the SCSI block oriented storage data over a TCP IP. Now notice that TCP IP is it does not have any; it is what is called a

string protocol and you have to essentially now layer on top of it a something which has got a structured way of structured way of sending the data ok.

So, there is some small conflict between TCP IPs model and what SCSI is doing. Of course, you will see this problem lot of places. If you look at SSL there also you will see that its what is called its got some structure that has to be send. So, it is not really a string or protocol that we are going to use. So, here what we do is in iSCSI we create a session between the SCSI initiator and the target SCSI target. What is the session? It is a group of TCP connections linking an initiator with a target and this particular session is identified by a connection ID ok.

So, basically its a group TCP connections linking an initiator with a target and this group is identified by connection ID. Now because SCSI requires ordered command delivery iSCSI has been specifically designed with some sequence numbers so, that you can support ordered command delivery within a session. Now, since iSCSI uses TCP IP the throughput of iSCSI is going to be determined by what TCP can provide and typically there is something called TCP congestion control algorithm. That we shall discuss soon that basically governs how much throughput you get in iSCSI. If you look at fiber channel, we had various classes and the class that is widely used is the one which has got buffer to buffer flow control. It does not have even though there is a class one which has got end to end flow control, the one which is used in fiber channel mostly is a class three which is got on the buffer to buffer.

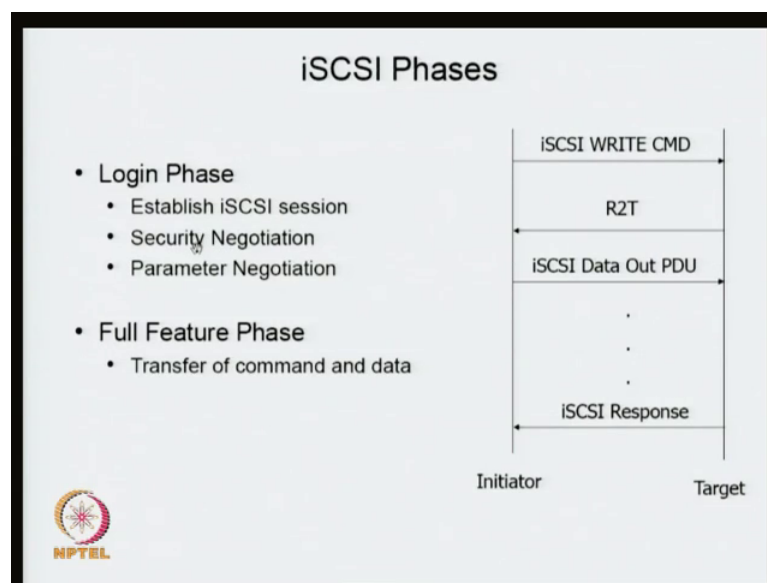
Basically because if you want to do end to end control in fiber channel, they require the switches and all the fabric or fiber channel to incorporate aspects like what TCP is doing; congestion and congestion control whatever some mechanisms. So, that there is a way to get as much throughput as possible through the fabric and so there has to be some fairly complicated or substance infrastructure that has to be there in fabric. So, that you get a throughput there is possible in the fabric.

Now, in fiber channel as far as I know, this particular exercise has not been done seriously. So, all you have is basically buffer to buffer flow control and because of this the kinds of throughput that you can get TCP mean with TCP are still not possible with its fiber channel. There is a basic issue basically TCP has got end to end flow control from the beginning. Whereas, fiber channel in spite of its design of class one, it is

generally now using only class three which is basically buffer to buffer flow control and so, it does not have an idea about the end to end aspect even today in spite of its class one in practice.

Therefore, iSCSI throughput can be better than possibly fiber channel. Especially on wide area networks the other parameters in iSCSI for example, max burst length, that also determines how much of data can be sent out in one burst. There are other controls in the system.

(Refer Slide Time: 14:01)



We just briefly look at the iSCSI. How it works? Again you can see that is basically encapsulating the SCSI commands in TCP. So, you basically standardize how it is encapsulated with various fields etcetera. For example, the SCSI write command is going to be there is going to be a structure that is going to take each other fields of the SCSI write command and then actually specified the standard specifies how it has to be encoded.

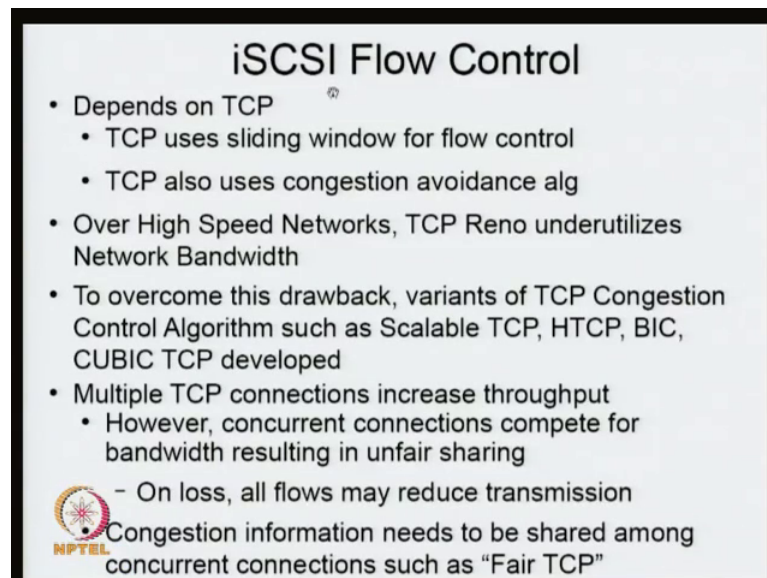
So, you will write the SCSI write command you send it from the initiative target. I can just as usual in SCSI the target has to allocate the buffers and be ready for getting the information from the iSCSI initiator. So, it has to send a ready to transfer. Again this ready to transfers SCSI command has to be encapsulated into a particular TCP on TCP. So, that the particular structure and that is basically the R2T. Again same issue with respect to when the initiator sends to the data, this has also has to be encapsulated in a

protocol data unit. This is basically iSCSI Data Out PDU unless response those comes back. As I mentioned earlier, you also need to other control management aspects. For example, you have to establish iSCSI session which has to means you have to create one or more TCP sessions ok.

And you might also if I am using IPSEC you might also have to use a security negotiation. Again IPSEC has a particular phase in which negotiate what kind of protocols are using for security. For example, Diffie Hellman or whatever the various methods, there are actually 6 or I think 6 methods for types of security. Let us say mechanisms present and also some of the parameter negotiation like I mentioned before you may want to negotiate, what is the max bus length ok.


So, all those things happen as part of iSCSI.

(Refer Slide Time: 16:09)



iSCSI Flow Control

- Depends on TCP[®]
 - TCP uses sliding window for flow control
 - TCP also uses congestion avoidance alg
- Over High Speed Networks, TCP Reno underutilizes Network Bandwidth
- To overcome this drawback, variants of TCP Congestion Control Algorithm such as Scalable TCP, HTCP, BIC, CUBIC TCP developed
- Multiple TCP connections increase throughput
 - However, concurrent connections compete for bandwidth resulting in unfair sharing
 - On loss, all flows may reduce transmission

 Congestion information needs to be shared among concurrent connections such as "Fair TCP"

Now, let us look with have a look at fiber channel have a it is flow control basically credit based. We will look at iSCSI flow control. First of all, as I mentioned earlier iSCSI is layered on top of TCP therefore, depends on TCP. Now, TCP uses sliding window for flow control, but also uses what is called congestion avoidance algorithm. The issue about TCP is that TCP assumes that the end devices are intelligent, but the network is dumber because the network is dumb. It is understood that it does not have any mechanisms to help you realize the throughput that it possible; that means, it has to use some way of probing the system to figure out how much bandwidth is there and

therefore, it has to use some slightly sophisticated control theoretic model and that control theoretic model we will discuss soon and it is difficultly called as a congestion avoidance algorithm.

Now, on high speed networks the typical TCP that we use TCP Reno. For example, Linux supports, TCP Reno and also something called CUBIC right now. It had used it had supported other models before but right now the most common one is TCP Reno. It turns out the TCP Reno underutilizes network bandwidth. I will again explain what this means. I will show exactly some I will give further details. So, because it, if I am using iSCSI and you want to get as much bandwidth as possible because typically storage is requires high bandwidth. If we were doing backup any of those kind of things its going to require huge amounts of bandwidth.

So, the idea in your especially is special important where has to use the network bandwidth well. So, since TCP Reno underutilizes network bandwidth we need to have different models are TCP. So, there are various other types of models that have been proposed we will look at few of them and some briefly. For example, there is something called Scalable TCP, HTCP, BIC, CUBIC TCP etcetera. As I mentioned, CUBIC TCP is the one that is currently there in Linux kernel recent versions some time back. There is still something called BIC which is no longer available in this in the standard Linux kernel.

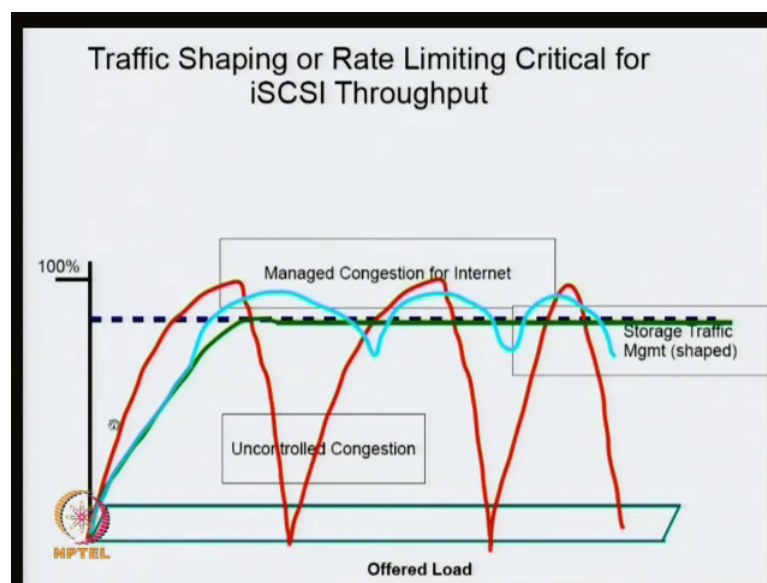
Now, another issue about iSCSI is that you can, as I mentioned you can have one or more TCP connections. With more one or more TCP connections it is possible to increase throughput, but there are some problems. If you have concurrent connections they can compete for bandwidth resulting unfair sharing and TCP by definition does not have any such models or how to make sure there is fairness in it and also there are some problematic aspects of TCP what are called synchronization aspects. For example, if there is a loss, if this is a packet has been send between two nodes and there is a loss then, the kernel implementation the TCP is common to all the flows.

So, because of a loss on one, it will decrease the amount of stuff each flow actually is going to send out because each flow is now say decreasing it, then the cumulative amount of stuff that is being send also decreases across all the flows. So, suddenly you can see a dramatic drop in all the flows, say in a sense it gets synchronized all the losses

gets synchronize across all the flows and therefore, it can be some more problematic. Basically I the congestion information needs to be shared among all the concurrent connections. I need to come up with some newer models. Like some people are call it fair TCP.

Again I go into some are the details in some detail and just give me the higher level ideas what exactly have the kinds of problems. When you take something like TCP which was not designed for a storage purposes used in storage context.

(Refer Slide Time: 20:10)



This gives you a diagrammatic way of understanding what could be going on. It turns out as I mentioned earlier the end devices are intelligent. The network is considered to be not so intelligent; that means that if you do not have any, let us say, take care of what the kind of situation that could develop, you might get into what is called uncontrollable congestion. For example, the red line like this. Why this is happening? The reason why is happening is because TCP nobody tells TCP how much bandwidth is there because I can suddenly connect my laptop here and expect to talk to some other party somewhere else right.

And no information is given to the laptop say this amount of bandwidth available. That means this particular laptop has to probe the network find out how much bandwidth is there and has to do a real time. Now this is somewhat different from if you are talking about fiber channel etcetera. In fiber channel it is a more structured environment and you

essentially tell each of those fiber channel entities. Some number you give it some number of credits and so, in a sense it is straightly more structure environment. Here, it is basically totally unstructured.

So, what does the TCP IP do it starts pumping packets to see how much packets how many packets can be sent before some kind of loss occurs to the flow. So, suppose is 100 percent is the maximum capacity in the system for that particular connection between my laptop and something else. I will keep on increasing my some number of packets are sent up to this point and then suddenly at this, experience a loss. Now if once I experience loss then I have to back off. I cannot be sending at the same rate because I noticed that I cannot between that. There are various models; the first simple TCP model was to come back to ground zero I start again. Again go down to zero and come back again.

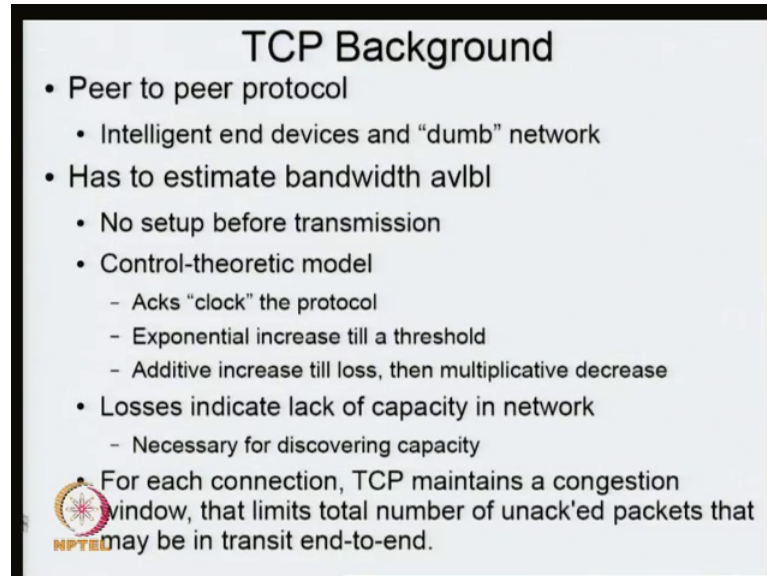
But basically we can see that with this kind of model you can essentially get if you integrate the area under this red curve that is essentially the amount of bandwidth ok you average it, essentially you are getting some closer to possibly half or close to that amount of bandwidth algorithm. Now, if you do some congestion avoidance probably you are going to be in this blue category. You increase slightly more gradually you do not overshoot as much as this possibly did, but you have do it very gently and then you suffer some losses.

You slightly contain it. You do not go to all the way to all most ground zero goes steady lower and somehow try to figure out a way of doing it slightly better. This is an example of managed congestion for internet. Ideally, what I want for storage is something like this. I want to discover the let us say, the capacity and hopefully stay constant like this. If I can do it then it is a best for me this is what I would prefer because this is what we normally do in a data center kind of situation. You can essentially get this kind of triples consistent performance.

And because for storage, backup is important aspect so, for those kind of things is really important to be able to say how long it is going to take because they take a long time in the first place. So, I need to have some kind of a idea about how long it takes. So, ideally I would like to be in this category even though it is not 100 percent, but I want to be consistent performance is what I am looking for. So, idea is, how does iSCSI or any other

model that writes on top of TCP, how can we make this happen? That means, we have to understand the TCP models.

(Refer Slide Time: 24:09)



TCP Background

- Peer to peer protocol
 - Intelligent end devices and “dumb” network
- Has to estimate bandwidth available
 - No setup before transmission
 - Control-theoretic model
 - Acks “clock” the protocol
 - Exponential increase till a threshold
 - Additive increase till loss, then multiplicative decrease
 - Losses indicate lack of capacity in network
 - Necessary for discovering capacity
- For each connection, TCP maintains a congestion window, that limits total number of unack'ed packets that may be in transit end-to-end.

NPTED

Again just to summarize some other things what we talked about, TCP is basically a peer to peer protocol. It has got intelligent end devices and “dumb” network. You have to estimate bandwidth available and unlike fiber channel or a telephone network there is no setup before transmission. Therefore, it has a controlled theoretic model and basically, the acknowledgments that you get essentially “clock” the protocol that essentially tells you how to go forward.

So, the standard method is in TCP it is to you start from ground zero, you increase exponentially. That is, what you do is, you send one packet, see if it comes back. If it comes back then you increase you know that there is capacity for two packets because one came back and you sent one and came back in one round trip. So, there are two packets already in boring flight. So, now we can increase it to you sent two packets and see if both the two packets come back the packets come back for it.

So, it becomes four then you again increase it by factor of two. So, it keep on increasing it. That is basically exponential increase till a threshold. After threshold you and this is determined by some other parts of system. You basically increase it additive because if you just go exponentially you are going to definitely hit a loss very quickly and you may want you may it can essentially. What it can do is? It can the network can get overloaded

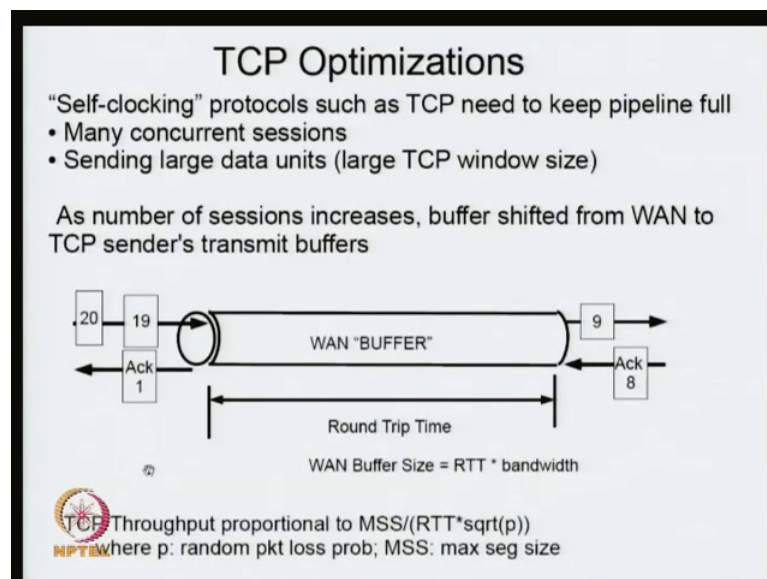
too fast. So, idea this for is not to go all the way till the packet loss happens the idea is to increase it in a more gradual manner then you actually hit the loss; so without overloading the network.

Now, once you hit the loss the idea is to cut back you sort of back off from the network and that is where what is called multiplicative decrease happens. One important thing about TCP is that the losses indicate lack of capacity in network. It is a very critical function. It is not some optional thing without losses you cannot figure out how much bandwidth is there. So, it is part of the network protocol. It is not something that can be dispensed with whereas, if you look at fiber channel etcetera. Losses are of see this problem the its not expected that you lose things ok.

So, because there is no serious attempted proving the network in the case of fiber channel, it is mostly set up. It is set up beforehand. Whereas, see you have to prove the network you have to suffer losses then you figure out what is going on. So, that is a big difference in the way prospecting in the first place. So, for each connection TCP also maintains a congestion window which limits in total number of unacknowledged packets that may be in transit end to end ok.

So, this is what is called the congestion window.

(Refer Slide Time: 27:07)



So, if you want to really do good in TCP you have to keep the if you are doing on a wide area network, you have to be sending packets so that the various routers etcetera they buffer your packets. So, essentially that pipeline between the various routers they are kept full. One way to do it is have been multiple concurrent sessions and sending large data units. That means, you need to have large TCP window size again because it is the flow control that window size essentially tells you how much you can send out.

So, the TCP throughput is proportional to $MSS \sqrt{p}$ where p are the random packet loss probability and MSS is the maximum segment size. So, the way to increase throughput is to increase MSS or decrease your probability of losing packets or make sure RTT is small. This is only way to increase throughput. Now in the case of fiber channel the throughput is given by number of credits because in fiber channel you can increase credits, but somebody has to do end to end control which is not currently there in most fiber channel use as the right now ok.

So, of course, fiber channel also has RTT as an issue. So, that also is inversely proportional to RTT . The one major difference between fiber channel TCP is that, your TCP performance depends on the loss. Whereas, fiber channel as far as possible does not want any this is want to see any loss. It is a basically a storage protocol because it is a it is designed for a different reason.

(Refer Slide Time: 29:01)

Achieving High TCP Performance

Large window sizes

Jumbo Frames

- Matches Ethernet frame size with TCP seg size


SACK: Selective Acknowledgements

- Useful when medium is dropping many packets

Traffic shaping or rate limiting:

- Critical to avoid packet drops
 - Takes a long time to recover in high RTT env
- TCP retransmit mechanisms should not be relied upon for **PERSISTENT** packet drops

TCP Window Scaling and Seq wrapping: RFC 1323



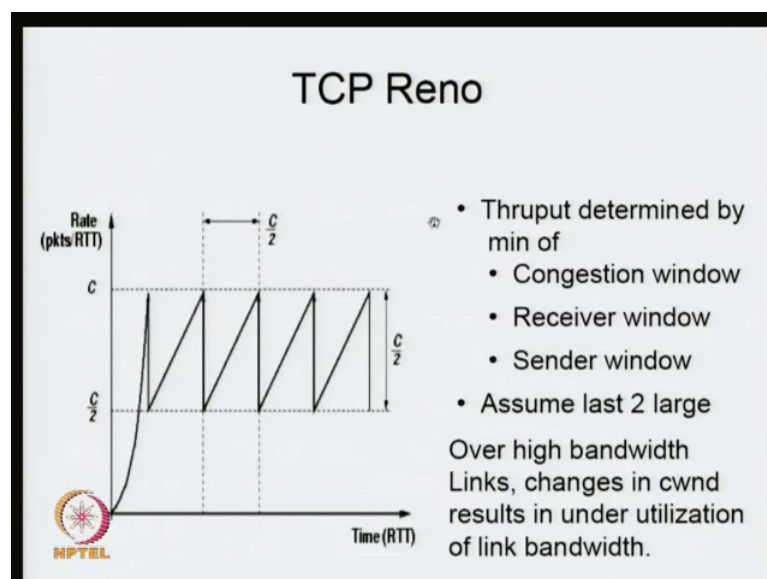
So, again how do you achieve high TCP performance large window sizes you can also use what is called jumbo frames.

So, the TCP segment size can be matched closer to the Ethernet jumbo size frame size it is about 9000 bytes. Typically Ethernet is 1500 bytes whereas, jumbo frames are 9000 bytes. You can use also what is called selective acknowledgements. Normally TCP is the something called cumulative acknowledgement. Here in selective acknowledgments you can send a bitmap which tells you which of them actually succeeded in going, which are them did not succeed. You do not have to hit last minute those portions that could be done in with cumulative acknowledgements ok.

So, these useful in when the medium is dropping many packets and this is sensible because iSCSI is going to do it across the WAN this may be appropriate. Again as I mentioned before you need to traffic shaping or rate limiting. We already discuss this part and most important thing is you have to avoid packet drops if possible. Why is that the case? Because it turns out if you avoid if you are going to have packet drops then the TCP retransmit mechanism actually takes a long time. I will come to the soon.

Basically because you are going to be in the additive phase in the TCP protocol. So, the ramp up is going to be quite slow and because it is going to be slow you are going to have serious problems with respect to achieving a full bandwidth.

(Refer Slide Time: 30:38)

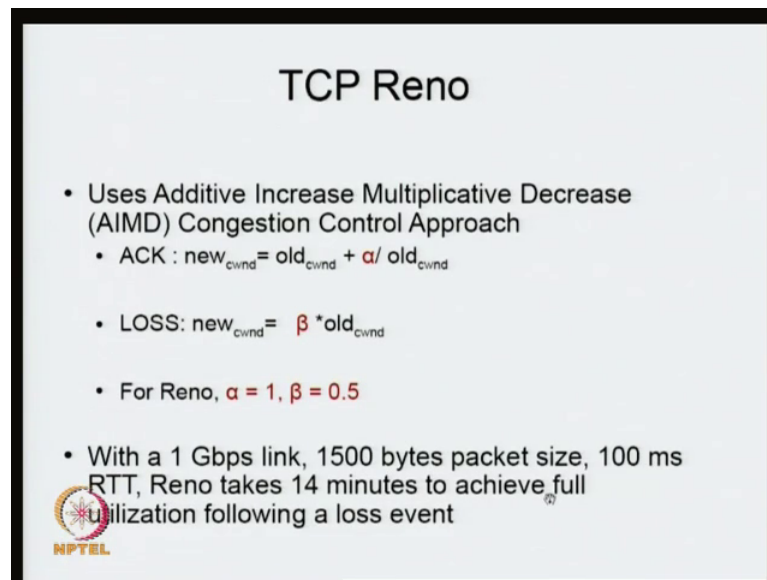


So, to let us just quickly look at some aspects of how this TCP performance; so this is what is called the exponent the rapid increase, the exponential what is exponential phase. Then it suffer a loss it comes down then you have a linear increase. Again it suffer a loss it come down. So, basically this is the threshold what I have talking about. So, the there are multiple types of TCP. So, in the beginning this one basically ramps up all the way to the beginning sorry to the maximum capacity till it hits the loss then it cuts on to half the capacity and again slow ramps up.

So, this is going to be a sort to pattern like this. Now, this is a round trip basically you can see start from here sorry I made a mistake. So, basically the throughput of this is determined by three parameters one is called the congestion window. That is number of packets that are out in the flight. The receiver window how much the recipient party can accept. The sender window basically how much it has buffered from the application. Normally, we can assume that a receiver and sender windows are very large. So, the only thing that limits a performance is congestion window ok.


Now, as we note as we saw just now this change in the congestion window its like this right. Essentially means that you are not able to hit the capacity. So, you essentially not going to see the full capacity now you can people have used this kind of the sort kind of patterns and if you integrate over this area. You will essential get the amount of bandwidth that is available in the system and that is how you get that $\frac{1}{\sqrt{p}}$ and $\frac{1}{RTT}$ a factors in the bandwidth I showed you before ok.

(Refer Slide Time: 32:40)



TCP Reno

- Uses Additive Increase Multiplicative Decrease (AIMD) Congestion Control Approach
 - ACK : $\text{new}_{\text{cwnd}} = \text{old}_{\text{cwnd}} + \alpha / \text{old}_{\text{cwnd}}$
 - LOSS: $\text{new}_{\text{cwnd}} = \beta * \text{old}_{\text{cwnd}}$
 - For Reno, $\alpha = 1, \beta = 0.5$
- With a 1 Gbps link, 1500 bytes packet size, 100 ms RTT, Reno takes 14 minutes to achieve full utilization following a loss event

 NPTEL

So, let us just look briefly at the various models that TCP has got. TCP Reno basically what he does is, if it uses as I mentioned it uses additive increase example here this additive increase part and these are multiplicative decrease. So, you are you going slowly one by one like this and then once a loss is that it dropped dramatically. So, this is basically new congestion window is basically old congestion window plus alpha times old congestion. Basically if you what it means is that this is the proportional increase.

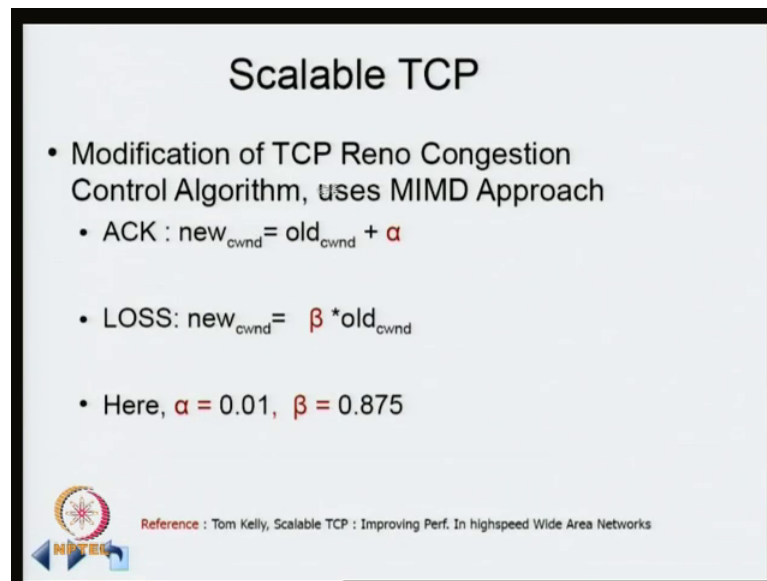
So, if you multiply by old congestion and its basically the you are increasing it by alpha the packet the new number of new packets are to additional packets constituting the alpha. The loss is going to be given by as a proportion of the old congestion window by factor of beta. So, that in Reno you know basically what happens is that if you your in the additive phase, if you get a ramp back you can send one more extra packet. That is your suppose you are sending 100 packets, we can send next number around 101 packets ok.

So, our basic problem is that this ramp up phase can be very slow because you can see right you are sending one extra packet every round trip. So, it will take C by 2 round trips before I can go from here to here. Now if I am able to send out 1000 packets. Let us say my capacity is 1000 packets; that mean, I have to wait 500 round trips before I can get back to this, which is very slow. For example, you have 1 gigabit per second link. Now-a-days this are available in a wide area networks, 1 gigabit per second is definitely

possible. 1500 bytes packet size, 100 meter, 100 millisecond RTT. Then Reno will take full 14 minutes to achieve full utilization following a loss event. That means, that in the previous example suppose I am here sorry I am here I suffer a loss event I come down here. It will take 14 minutes to get to this; it is excruciating this law ok.

So, this additive increases a bit too it is a bit too slow. So, people wanted to see if it is possible to something better.

(Refer Slide Time: 35:11)



The slide is titled "Scalable TCP" and lists the following points:

- Modification of TCP Reno Congestion Control Algorithm, uses MIMD Approach
 - ACK : $\text{new}_{\text{cwnd}} = \text{old}_{\text{cwnd}} + \alpha$
 - LOSS: $\text{new}_{\text{cwnd}} = \beta * \text{old}_{\text{cwnd}}$
 - Here, $\alpha = 0.01$, $\beta = 0.875$

At the bottom left is the Intel logo. At the bottom right is the reference: "Reference : Tom Kelly, Scalable TCP : Improving Perf. In highspeed Wide Area Networks".

So, there is scalable TCP basically does that instead of in the previous one you had you had increased proportionality with respect to congestion window. Here we use directly increase it by alpha and its 0.01. Loss is exactly same as before this is slightly faster way of ramping up in the additive phase.

(Refer Slide Time: 35:36)

HTCP

- Modification of TCP Reno
- Employs different alg to incr congestion window
 - Multiplicative Decrease Factor β adaptive
 - Also, Additive Increment α :
 $\alpha = \alpha^L$ if $\Delta \leq \Delta^L$
 $\alpha = \alpha^H(\Delta)$ if $\Delta > \Delta^L$ where
 α^L is additive increment in low speed mode
 α^H is additive increment in high speed mode
 Δ is the time elapsed since last congestion event

Δ^L is the Threshold

NPTEL

Reference : D. Leith, HTCP: TCP for High speed and Long distr

The other models also I am not going to go too much into it. There are things like example you can both play around with the alpha and beta and make it adaptive. So, for example, I just briefly indicate what this does the HTCP does. There is as a mentioned it is adaptive. So, there are two modes- low speed mode and high speed mode.

So, the alpha is some value in low speed mode, in high speed mode it is slightly bigger and it is decided upon depending on the time elapsed since the last congestion event and this alpha H delta is basically some function of delta and delta L. Some people have tried using this also .


(Refer Slide Time: 36:22)

Binary Increase Congestion (BIC) TCP

Modification of TCP Reno (in Linux kernel 2.6.8 -.18)

- Consists of two parts
 - Binary Search Increase
 - Additive Increase
- Given the minimum and maximum window sizes, set the target window to midway between the two
- If no losses are detected, the current window size becomes the new Minimum and a new target is calculated
- If losses occur, then current window size is the new Maximum and reduced window size is the new minimum

More aggressive initially, gets less aggressive as the window size approaches the target.

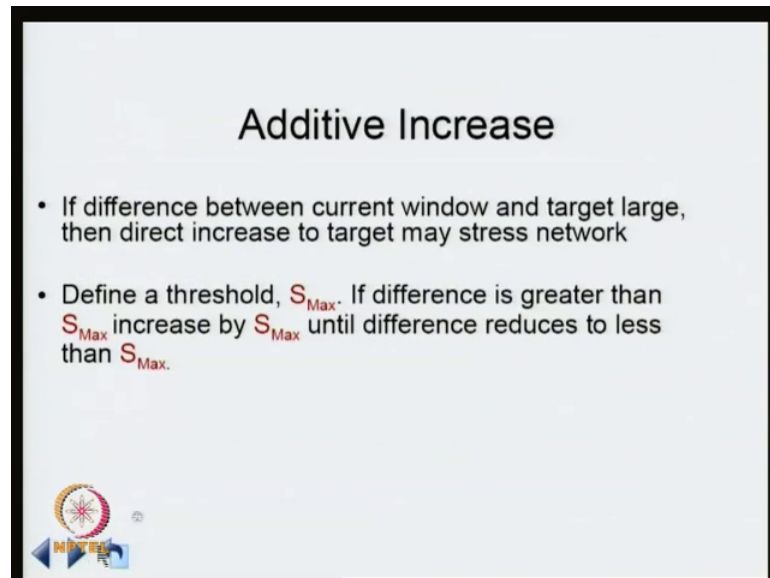
Reference : I Rhee, Binary Increase Congestion Control for fast long distance networks

Now, there was another thing that was attempted BIC and this was actually present in Linux kernel for quite some time 2.6.8 to 2.6.18 this called the binary increase congestion. Basically, as you can see the idea here is to see if there is a way instead of going gradually from here to here, to see if you can go from here go to some in between position etcetera. See if you can do binary search, the idea simple idea is to do binary search. Why go linearly one by one do some kind of binary search. You suddenly jump to see jump here see if it is ok. It is not keep increasing it by some suitable factor.

So, it consists of two parts binary search increase and additive increase. So, given the minimum and maximum window sizes, set the target window to midway between the two. Just like in binary search you want to look at in between first same thing, you target window to midway between the two. You find that there are no losses at this point then current window becomes the new minimum and the new target is calculated ok.

So, what is a lower value becomes the current window becomes the lower the minimum and again a new target in between values calculate again. If there is a loss then the current window becomes the max and then you reduced the window size the you reduced the window size and that becomes to minimum. Essentially, what is happening is that you are aggressive initially, but you gets less aggressive as the window size approaches the target.

(Refer Slide Time: 37:58)



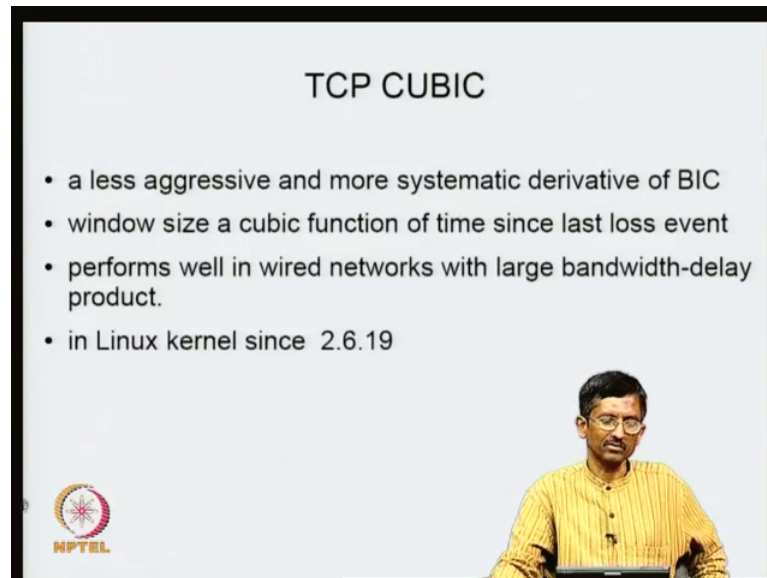
Additive Increase

- If difference between current window and target large, then direct increase to target may stress network
- Define a threshold, S_{Max} . If difference is greater than S_{Max} increase by S_{Max} until difference reduces to less than S_{Max} .

Now, it turns out that this also can get into some problems basically because if the difference between current window and target is very large, then directly increasing it might stress the network. So, to avoid this kind of stressing you can define some thresholds. So, basically you come up with the difference. If it is bigger than that threshold you actually increase it only by the threshold. Till finally, you come to the difference less than the threshold ok.

So, various people have tinkered with this kind of algorithms and various improvements have been made.

(Refer Slide Time: 38:36)



TCP CUBIC

- a less aggressive and more systematic derivative of BIC
- window size a cubic function of time since last loss event
- performs well in wired networks with large bandwidth-delay product.
- in Linux kernel since 2.6.19

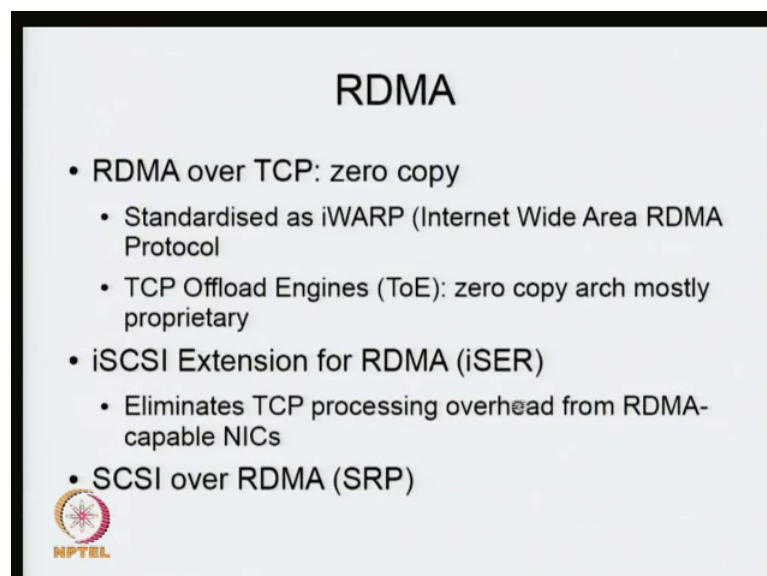
NPTEL

The slide features a black border and a light blue background. In the bottom right corner, there is a small inset image of a man with glasses wearing a yellow shirt. The NPTEL logo is located in the bottom left corner.

Currently, actually Linux kernel uses a modification of BIC it is called TCP CUBIC; where the window size is a cubic function of last times since last loss event. So, this has been experimentally shown that it works well in wide networks with large bandwidth delay product and essentially it is a less aggressive and more systematic derivative of BIC (Refer Time: 39:04) details one should look at the papers ok.

So, this has been there in Linux kernel and it is currently there standard let comes in Linux kernel.

(Refer Slide Time: 39:15)



RDMA

- RDMA over TCP: zero copy
 - Standardised as iWARP (Internet Wide Area RDMA Protocol)
 - TCP Offload Engines (ToE): zero copy arch mostly proprietary
- iSCSI Extension for RDMA (iSER)
 - Eliminates TCP processing overhead from RDMA-capable NICs
- SCSI over RDMA (SRP)

NPTEL

The slide features a black border and a light blue background. The NPTEL logo is located in the bottom left corner.

Now, given that we have talk a bit about TCP the question is for a iSCSI what do we. These are various possibilities I mentioned last time that one important thing about a storage protocol is that it should be zero copy. If it is not zero copy then you are going to suffer great difficulties especially from the CPU utilization point of view ok.

Now, for that reason there has been some standards called RDMA over TCP and which essentially guarantee zero copy and there has been standardized as a particular protocol internet wide area RDMA protocol. Now, as I mentioned before TCP offload engines also attempt to do the same thing, but unfortunately TCP offload engines were done by various vendors in a proprietary manner. So, even if they have zero copy architecture their ability to incorporate has been somewhat poor. Therefore, these standard like iWARP are better for achieving zero copy across various types of devices.


So, basically what we what we have right now is I want to have zero copies. So, one I can do is I can RDMA over TCP, but I am doing iSCSI over TCP, but means that I can directly use this RDMA over TCP. That means, I need to find a way of extending iSCSI protocol that was cleared over TCP. So, it actually layered over RDMA because for me what the problem is that my original definition iSCSI was over TCP, but if I want to use zero copy I need to have RDMA on top of TCP itself.

So, I need to essentially change iSCSI. So, that it is layered of RDMA which in turn uses TCP, that also has been defined is called iSCSI extension for RDMA [iSER] and basically this combination of iSCSI, RDMA in TCP. It eliminates TCP processing overhead from RDMA capable NICs. There has been another effort called SCSI over RDMA, where SRP which I am not very sure, but I think it is not that very popular and so, that also has been attempted ok.

So, it is basically directly instead of going through iSCSI this is the original protocol that came in the earlier 2000. It directly tries to layer SCSI over RDMA. So, what we are talking right now is iSCSI over RDMA over TCP. This seems to be the one which is used currently.

(Refer Slide Time: 41:56)

Infiniband

- Proposed as system to system interconnect
- Widely used in HPC due to high speed
 - 40Gbps common
- Has multiple lanes in each connection with QoS
 - Also, failover
- Used as storage interconnect
 - Can use RDMA over Infiniband
- No std API in specification: [®]
 only a set of "verbs": functions that must exist.

Now, there is another in interconnect called Infiniband. Now, initially in the early two thousand this was proposed as a system to system interconnect. So, basic idea is that when CPUs have to talk to each other you have a multi processor system and you need high speed interconnect. Then Infiniband was proposed as a system interconnect and this also became has been sorry it has become widely available and it is widely used in high performance computing, for high speed because of it high speed. For example, it is very easy to get 40 GB per second kind of connections. It is not that difficult ok.

So, and actually if you look at the Infiniband roadmap, people talk about 120 gigabyte per second, 200 gigabyte and 300 gigabyte per second. So, high speed is certainly possible and so, if you look at all the top 500 hundred machines worldwide. You will find that a good number probably more than I do not know close to half or so, I am not sure exactly the amount they use Infiniband as a system to system interconnect.

So, because of the wide availability of this interconnect the idea would be why not is it first storage also. That is basically the original plan somehow it been take off that easily in the beginning. But now I think because of the widespread is of Infiniband in high speed systems. I think it is going to likely to come back again. One thing about Infiniband is that it has some quality of service models as part of its specification. It also handles some failover aspects and so, it is a slightly more complex a specification compared to Ethernet and it is not clear how much of all the specification. For example,

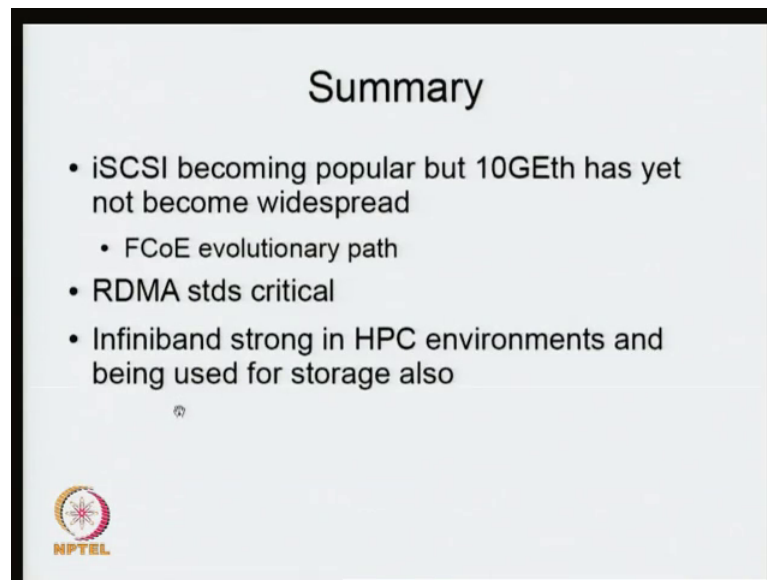
failover or quality of service is widely available in all the products, but the specification is has got quite a few substantial support for quality of service of failover ok.

So, if such a thing is available the natural thing would be to use it for storage interconnect and then again here also we need to make sure that somehow you do not have extra copies because extra copies going to kill your performance. So, there is a RDMA over Infiniband band also. So, this also has been defined and I think many are displeased like 40 gigabit per second etcetera. Without RDMA you cannot really survive. So, for CPU to CPU in large scale transfers RDMA over infiniband is necessary and I think as far as it is widely used.

Now, one specular thing about infiniband is that compared to SCSI or other a TCP IP etcetera. Many things have been left slightly under specified. So, for example, it has a notion of what is called “verbs” that means, these are certain types of functionality that has to exists. For example, I should be able to send something I should receive something. But most of the parameters are not really spelt out. The reason why this is the case is because it has to interact if it is basically based on something called the via specification and that tries to ensure that as little extra copies are made as possible and therefore, it turns out that some of the things interact heavily with other aspects of the system especially operating systems or kernels. And so, they have left it sort of open and so, its slightly it is very well specified with respect to some areas like example quality of service or failover etcetera. But in terms of the API itself it seems to be under specified and so, this is one issue with respect to infiniband.

But there are attempts at the various types of let us say standards that have come up in addition to the main standard. So, that there is a possibility of interoperability. So, given that it is highly used in HPC area so; that means that in spite of this under specification is still not a problem.

(Refer Slide Time: 46:55)



So, let me summarize what we done so far. First of all we looked at some in the previous class we looked at iSCSI sorry we looked at SCSI and then we saw how fiber channel can be used as a way to transport SCSI protocol.

Now, fiber channel is very well suited for as a storage is very well suited as a storage protocol. But storage is somewhat of a niche market because of that the cost of fiber channel has been typically very high and so, the trend towards is being to figure out how to reduce the cost of this fiber channel kind of based models interconnects. So, the idea would be to see if there is some way to move towards so, say a different kind of model and the model that is most attractive they somehow go to gigabit Ethernet or 10 gigabit Ethernet. And so, iSCSI is one model there is a evolutionary path what I mentioned as fCoE - fiber channel over Ethernet, which is an evolutionary path.

Now, it turns out that this is becoming available also except that it also has certain complications. I am not really gone into it, but it turns out that if you want to use fCoE then there are certain subtle aspects to the protocol in fiber channel. For example, when a particular switch fails in fiber channel, there is some way to recover from it. Now that specific way in which you recover from it also has to be made appropriately available even if you use Ethernet. That means that you are the same kind of semantics that was there in a pure fiber channel network also has to be made available and Ethernet a

network; that means that there has to be some more extra infrastructure that takes care of those issues.

So, even though it is an evolutionary path it also entails certain additional complexities and so, but I think it is being attempted its being done. Of course, you are losing something also when you go through this particular path; basically, because you are not able to guarantee certain latencies because the Ethernet protocol has a different model. So, iSCSI is becoming popular, but as far as I can see 10 gigabit Ethernet has not yet become widespread. Therefore, its use has not become that dramatic within a department of setting it is still not happened. I think probably in about next five years or so, if 10 gigabit Ethernet becomes very common then iSCSI could be used as a protocol, but it is still not the norm right now ok.

Now, for all these things it turns out RDMA standard are very critical, without RDMA standards there is no possibility of zero copy and so this basically there has to be lots of standardization around RDMA and lot of effort is going here and it is just taking a lot of time to stabilize this particular issue and if you look at infiniband they are very strong in the HPC environments and because this attractiveness about the kind of installed base of high speed 40 gigabit per second or 20 gigabit per second kind of ports. One can many have been attempting use it for storage and I think it is also picking up ok.

So, currently it is a bit confusing situation there is no one strong interconnects that is available. The only thing that you can say is that iSCSI on the top is everywhere and fiber at the bottom is everywhere. In between there is a lot of churning going on. So, we will not know exactly what is going to happen, but it is very likely that gigabit Ethernet actually will finally takeover but it is going take some time.

With this I am going to conclude this particular lecture.

Thank you.