

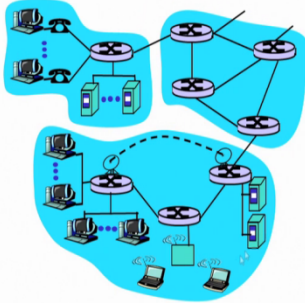
**Information Security 3**  
**Sri M J Shankar Raman,**  
**Consultant Department of Computer Science and Engineering,**  
**Indian Institute of Technology Madras**  
**Module 46**  
**Structure of A Network**

In this module we will actually try to talk about and discuss how a typical network is actually structured so how would a structure of a network will look like.


(Refer Slide Time: 00:31)

**A closer look at network structure:**

- ❑ **network edge:**  
applications and hosts
- ❑ **network core:**
  - ❖ routers
  - ❖ network of networks
- ❑ **access networks,**  
**physical media:**  
communication links



The diagram illustrates a network structure with three main components: a network edge (top left) containing applications and hosts, a network core (top right) consisting of a network of routers, and access networks (bottom) connecting the edge and core. Physical media and communication links are also shown connecting the various parts of the network.

 (C): James F. Kurose and Keith W. Ross

So we look at the network structure we need to identify that there are typically different parts of a network structure, so I have something called as a network edge, so the network edge will be the the systems, the network edge will be comprise the system that is actually running my applications and where the applications going to run, the applications are going to be actually running on my host systems.

So the host or a end system as we were referring to in one of the previous modules are the systems on which I will typically be running my network applications, so be it a browser application or web server application, be it a file server or be it a a email client on a email server, whatever it is. Now the next part of it is basically the network core.

So the network core is basically going to be consisting of typically this part of my entire network and here I will typically have routers and switcher switches devices configured and running, so as we were discussing in the previous module, the router devices is going to be the device that is actually responsible for forwarding a packets from a given source to a given destination.

So if I have to have my packets reaching to a network which is actually 10 hops away right, the router device is a device that is responsible for that, so in networking world when we say hop, we typically refer to one router device to another router device, so when my packet is going to go from first router to the second router, technically it is refer to as one hop that the packet is actually taking in its path towards the final destination.

So the network core part of it is going to be consisting of router devices, which which has is responsibility of forwarding the packets from the source to reach the final destination after taking many hops, right. So how many hops it takes will be typically dependent on what is the source and where is the destination and how many devices it has to cross to reach the final destination, right?

So between the network edge and the network core you have typically the access networks of the physical media, so which could which are which are the communication links at the physical level that is going to be actually transferring the data in bits, right. So that is where we really talked about the transmission rate or the bandwidth and the parameter of the transmission rate is basically what is going to differentiate between the different communication links and what communication link we will typically try to make use of will be dependent on what is the type of data that I am trying to transfer and which my network application is dealing with.

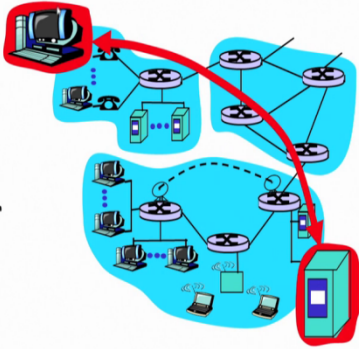
So if I for example are using is dealing, I am I am dealing with a network application that is dealing with a voice kind of traffic then I wouldn't want to much of a latency associated with the transfer of the data from the source towards the destination and because of this reason I would going for a very low latency less fault available less fault inducing a link as compare to lets say I am I am actually dealing with an application where I am transporting the normal data across from one mission to another mission, where I wouldn't be bothered too much about a small amount of latency being there and likewise based on that type of data that my application is going to be transferring from one system to another system and my own business requirements,

the organizations will typically depend, will design on what kind of communication link bandwidth that they would typically want to make use of.

(Refer Slide Time: 04:47)

## The network edge:

- ❑ **end systems (hosts):**
  - ❖ run application programs
  - ❖ e.g. Web, email
  - ❖ at "edge of network"
- ❑ **client/server model**
  - ❖ client host requests, receives service from always-on server
  - ❖ e.g. Web browser/server; email client/server
- ❑ **peer-peer model:**
  - ❖ minimal (or no) use of dedicated servers
  - ❖ e.g. Skype, BitTorrent, KaZaA



(C): James F. Kurose and Keith W. Ross

So the end systems which is basically comprising my network edge is going to run the application programs, so I was just explaining the application program here could be a web program or an email program and they will be typically the edge of the network, so why would we call this is the edge of the network because if I consider this entire thing as my network topology, you will typically find the host the end systems actually running only at my edge of my entire network topology, so whether it being a client or whether it being a server, you will always find that these are typically at the edge of my network and not right at the middle of my network, right.

So that is reason why this this part of my network is actually referred to as a network edge, so a network edge is typically consist of host or end systems in which I I run my network applications so whether it be a client application or whether it be a a server application. So in the client server model, how do we differentiate between a client and a server. Now a client end system is typically end system that is actually will initiate the request and to whom will it initiate the request, it will initiate the request to a server that is actually running on some part of my network

and the server is always expected to be on because the unless and until the server is on the client is not able to be a successful in communicating to the server all alone.

So the server is expected to be first brought up, it should be available, it should be waiting for acceptance, waiting for connection request, coming in from any kind of a client mission on the network and when it comes in, it basically accepts the connection request and then sorts of does this service whatever the client wants it to do and then sends a packet response packet back all the way to the client, right. So that is basically what we refer to as a client server model so it could typically be a browser, a web browser like your firefox or your chrome or your internet explorer browser talking to a web server like an apache server or an IAS server or a tomcat whatever it is and trying to get the data from there and then displaying it on your browser window or another example of an application could be an email client application that I am running here and that client application could be trying to talk to the email server application that is running on my network and trying to get the data back, right.

So likewise there are so many different combinations of the client and servers that are possible and also which we are using on a day today basis in today's highly network world, on the other hand I could also had something called, right. So likewise there are so many different combinations of the client and servers that are possible and also which we are using on a day today basis in today's highly network world, on the other hand I could also had something called as a peer to peer model where I don't really have any specific server classified as a a server and another mission classified typically as a client and it always be will be dependent on the the need basis where any system on the network could actually start acting as a a server and another system can be acting as a client, right. So some examples are something like Skype.

(Refer Slide Time: 08:06)

## Network edge: connection-oriented service

<p><i>Goal:</i> data transfer between end systems</p> <ul style="list-style-type: none"><li>□ <i>handshaking:</i> setup (prepare for) data transfer ahead of time<ul style="list-style-type: none"><li>❖ Hello, hello back human protocol</li><li>❖ <i>set up "state"</i> in two communicating hosts</li></ul></li><li>□ TCP - Transmission Control Protocol<ul style="list-style-type: none"><li>❖ Internet's connection-oriented service</li></ul></li></ul>	<p><i>TCP service</i> [RFC 793]</p> <ul style="list-style-type: none"><li>□ <i>reliable, in-order</i> byte-stream data transfer<ul style="list-style-type: none"><li>❖ loss: acknowledgements and retransmissions</li></ul></li><li>□ <i>flow control:</i><ul style="list-style-type: none"><li>❖ sender won't overwhelm receiver</li></ul></li><li>□ <i>congestion control:</i><ul style="list-style-type: none"><li>❖ senders "slow down sending rate" when network congested</li></ul></li></ul>
--	---

(C): James F. Kurose and Keith W. Ross

So when we talk of a connection oriented service in network edge point of view, what we exactly mean by connection oriented service is that, the basic purpose is that the data transfer between the two end systems should be done after there is something called as an handshaking that is done, so what exactly is handshaking so the handshaking is basically a set of steps that will be executed for doing the initial set up between the two end systems that wants to communicate, right. So it's like a sort of the sequence that we typically have whenever we try to make a phone call to our friend or to whoever it is wherein initially there is an exchange of an hello hello on either side, right so if you retrospectively why we use the word hello is to ensure that we are able to hear what other person is speaking and similarly the other person is able to hear what we are speaking.

Likewise in the computer networking world when I am basically trying to setup connection oriented service there is a initial set of steps that is actually done which is technically refer to as handshaking between the two end systems wherein both sides sort of agree upon the other parties capabilities and limitations so that the data transfer when it happens at the end of this handshaking phase could be very effective and or also it could be very reliable, right. So it sorts

of sets up the state in the two communicating host that it is basically the two end systems on either side of the network that is now trying to set up the communication path, right?

So what is a protocol that is actually used for this that is nothing but TCP , TCP stands for the transmission control protocol and this is actually one of the very popular protocols on the internet today wherever there is a requirement that the data requires to be reliably transferred across to the other side the application developers will always use TCP as the underline transport mechanism for transferring the data across. So what exactly where the different services that the TCP provides, the TCP protocol provides for the end systems, it provides a reliable in order file stream data transfer, so what do I mean by reliable here is that if I am basically transferring 10 bytes from my client to the server, all the 10 bytes has to be delivered to the server, right that is basically what we are meaning by reliable, now what do we mean by in order byte stream. If I am transferring the data let say as NPTEL, right?

The in order delivery essentially means that the server application should be receiving the data in the same order so it should be receiving it as NPTEL so NPTEL it should be receiving it in same order and it should not be in a jumbled up order, right?

So that is basically what we mean by reliable, in order, byte stream, data transfer, so how do we actually do this reliable in order byte stream data transfer in TCP is that we actually try to make use of an acknowledgement and a re-transmission to detect if there is a loss and if there is a loss, the sender will sort of re transmit the data till it gets acknowledgement from the other side yes that I have received this particular data, so the second service is basically what is called as a flow control, now what we mean by flow control is that let's take an analogy of a normal human conversation between two persons, now when we are actually having these two people talk face to face, there are always indications given by one party to the other party or whenever there is an over flow or whenever there is an a too much of information that is being provided the other person can say that please give me some time to assimilate whatever you have actually just now told me, right?

So what this other person is communicating thereby is that I need some time to digest the information that you've given me and therefore please stop communicating henceforth till I tell you to restart again right, now similarly in the networking world when two end systems are

trying to talk to each other you need to have a mechanism by which the other party the other end system could communicate to the the the the other party that he has only limited buffer and he doesn't want to overflow that buffer if the data continues to keep coming in the same manner, right?

So that is basically what we are referring to as a flow control here essentially the sender does not want to overwhelm the receiver by giving too much of data to it and thereby having the data overflowed on the receiver side because it doesn't have enough memory buffer space to to take take charge of the data and then deliver it to the higher level application, so how does the the two end systems that is the client and the server typical, right?

The typical network you have a client and you have a server, how the two end systems communicate with each other saying that I have only this much of space available free in my memory buffer for receiving more data from you this entire process is basically what is refer to as a flow control, so in the flow control we are essentially trying to ensure that one side is not overwhelmed the other side and sort out over flow the data which is received on the receiver side right, and so that the the the the flow of data between the two end systems is controlled in this manner as part of the the reliable service that the TCP protocol is actually offering to the application.

Now the next servers is basically the congestion control, now the flow control that we talked of recently is sort of trying to deduce the capacity of the other end system at any point in time so as to ensure that the buffers set are there at other end systems are not over flowing. Now this takes care of the the limitations or the capacity that I have on the two end systems are that point in time but I also have a requirement that the data that is getting transferred from the source to the destination which is actually going to flow over a network across multiple hops as we just saw is also the network is also capable of sort of holding the data without the data getting loss in between some in some of the devices that the data is going to cross, so like as I was giving an example before if my data is actually going to flow from one device in Chennai to another device in Mumbai it has to possibly cross let say like 6 different devices for example right, I have to ensure that on all the 6 devices there will be sufficient memory buffer space available so that my

data that is actually passing on all those 6 devices could be using the memory buffer till the time that it is able to get successfully transmitted out of that device into the next device on the path.

Now how does this get ensured is by the mechanism called as congestion control, so when I say congestion control what we essentially try to find out is we use some intelligent mechanisms to detect whether the network is it getting congested and whenever there is a detection then the network is getting congested, I am going to sort of slow down the amount of traffic that I am injecting into the network as a client system or as a server system so that over a period of time after I slow down my injection the whole network starts of getting free and once I detect the network is getting free I am going to start increasing the rate of traffic that I am going to inject into the system right, so likewise I basically have a mechanism by which I detect the state of the network at any point in time and do a sort of a self adjustment depending on the state system either I try to induce more traffic or I try to reduce the amount of traffic which I am injecting into the system right, so that is basically what is refer to as a congestion control.

So the TCP server is basically offers a reliable in order byte stream, it ensures flow control so that the data is not lost because of lack of sufficient memory buffers on my end systems it does congestion control to ensure that the data is not lost because of lack of memory buffers on all the intermediate network devices like my router devices not having enough memory buffer space at that instant of time.



(Refer Slide Time: 17:24)

## Network edge: connectionless service

Goal: data transfer between end systems

❖ same as before!

❑ UDP - User Datagram Protocol [RFC 768]:

❖ connectionless

❖ unreliable data transfer

❖ no flow control

❖ no congestion control

App's using TCP:

❑ HTTP (Web), FTP (file transfer), Telnet (remote login), SMTP (email)

App's using UDP:

❑ streaming media, teleconferencing, DNS, Internet telephony

(C): James F. Kurose and Keith W. Ross

So with respect to the connection less service the data transfer again the objective is basically do the data transfer between end systems but here I don't have an explicit connection set up like I have in a connection oriented service and the most commonly used protocol in the transport layer for providing connection less service is basically what is refer to as UDP protocol user datagram protocol. So it is basically a connection less service there is no express connection that is created before I start doing the data transfer it's an unreliable data transfer there is no flow control or no congestion control that is actually done in UDP, now some of the applications that usually used in TCP is http that is typically a web based protocol, so http https and so on. Ftp that is your file transfer protocol, telnet your ssh or smtp all are different applications which will typically require to run over the network, these kind of an applications will typically try to use TCP, the applications which actually use UDP are applications like my streaming media, so DNS my network management that is snmp related applications, any kind of voice related applications. All these applications make use of UDP as my underline transport protocol.

Thank you.