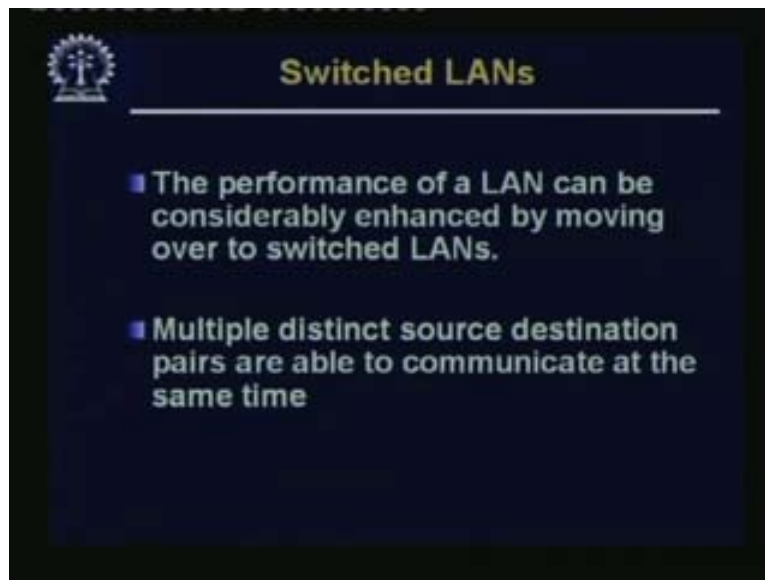


Computer networks
Prof. Sujoy Ghosh
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture Name - 8
Packet switches

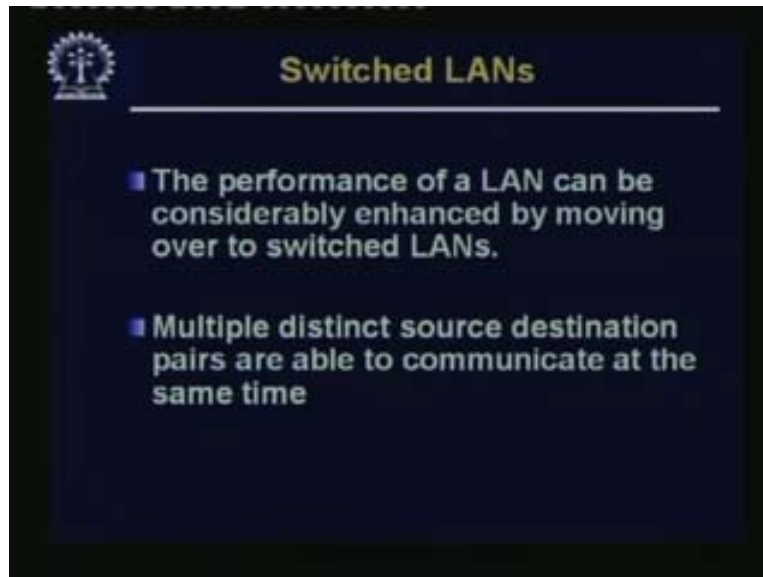
Good day. In this lecture, we will discuss about packet switches; packet switches are switches, which are specifically designed for handling packets. We have seen the telecom switches like space division, time division switches, etc. We have also discussed something about multistage switches, which are used in packet switches. In this lecture, we will specifically discuss about packet switches, which are at the heart of almost all your computer networks.

(Refer Slide Time: 01:22)



First, to give you some motivation, why do we require switches? The performance of a LAN can be considerably enhanced by moving over to switched LANs. If you remember, some of the earliest kinds of LANs that we have just seen, a brief glance of some of them used shared memory like hub or a bus. If you are using a shared medium, what is going to happen is that only 2 nodes connected to this LAN can communicate with each other at a particular point of time, whereas if you have a switch, multiple pairs of users or multiple pairs of nodes can communicate with each other at the same time. Naturally, you get a very good boost in performance of the overall LAN; that is one reason we require switches. We require switches in other place also.

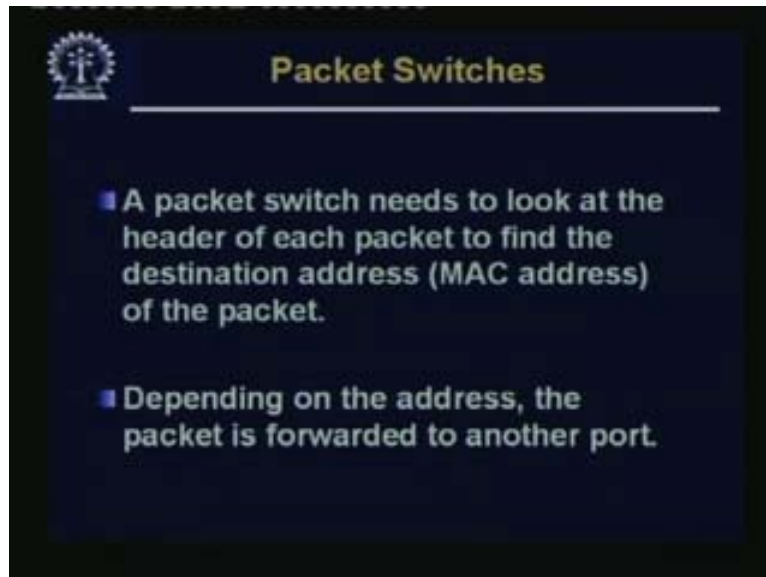
(Refer Slide Time: 02:28)



Multiple distinct source destination pairs are able to communicate at the same time through a switch. That is why we require a switch in packet networks. A packet switch needs to look at the header of each packet to find the destination address of the packet. If you remember, when we discussed about connection-oriented and connectionless systems, telephone networks are predominantly connection-oriented, although connectionless technology is also getting there. Data networks are predominantly connectionless although some connection-oriented network technologies are also used in some parts of the network.

Today when we are talking about packet switches, we are specifically talking about connectionless systems. In a connectionless system, since there is no fixed connection between the source and destination, which is held during a session etc., such concepts are not there. Therefore, each packet is on its own and that packet is perishable and contains data from the source node, which is meant for the destination node. It will contain lots of other things like headers, etc., which we talked about earlier but now we are specifically interested in the packets – that means, the data stream has been broken up into packets. So each packet must contain the destination address of where the packet will finally reach because since there is no space connection, each packet must carry the destination address.

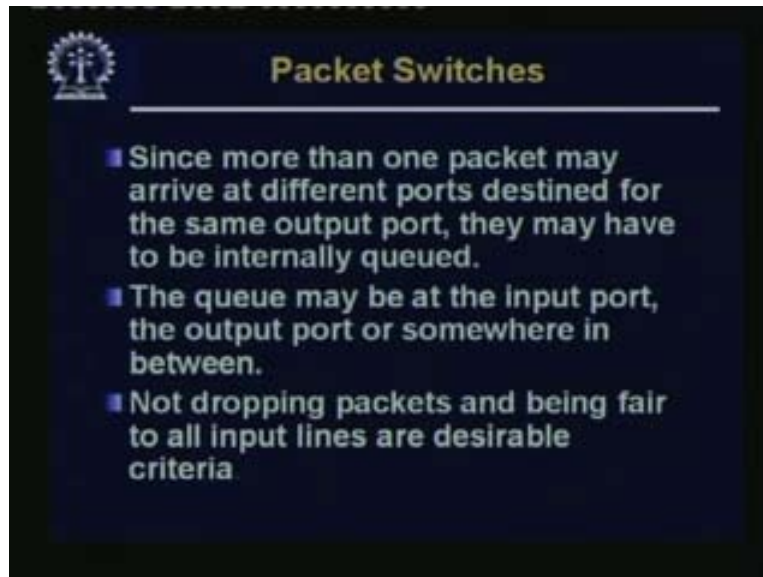
(Refer Slide Time: 04:12)



And this destination address is sometimes also called the MAC address; MAC is for media access control. There are various types of MAC address. We will come to that later on. Depending on the address, the packet is forwarded to another port. What happens in a packet switch is that through one port, some data has come in; this data will contain a MAC address, this MAC address is the address of the node, the address of the destination where it is supposed to reach, the destination node.

This destination node – the switch has to know that this destination is connected to this particular outgoing link. There is a question of mapping from the MAC address to a particular output line and if that output line has a tag as we had seen in the earlier lecture on some of the switches that they use tags. If they have a tag, depending on this MAC address, this tag could be sort of maybe looked up from table or something and added to the packet so that packet can make its way to the switch and reach the final, correct output line, and from then on to the destination station.

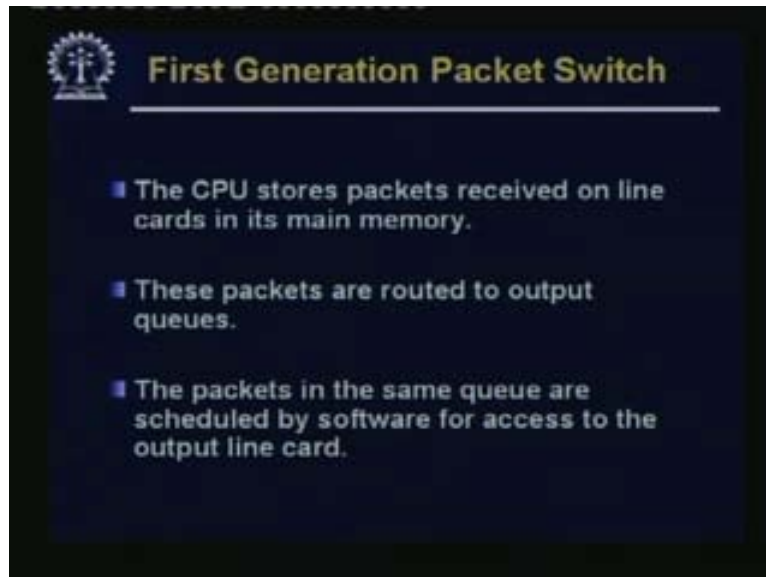
(Refer Slide Time: 05:42)



Depending on the address, the packet is forwarded to another port. Since more than one packet may arrive at different ports destined for the same output port, they may have to be internally queued. This was also discussed in the last lecture, that if we have more than one packet, which is sort of destined for the same output port, obviously we cannot take both of them together. One of them has to wait in some memory or in some buffer or maybe it will be queued because there may be other packets coming up for the same port so they will be all queued and this queue could be serviced by the output line 1/1, that means 1by 1 will be popped out of the queue.

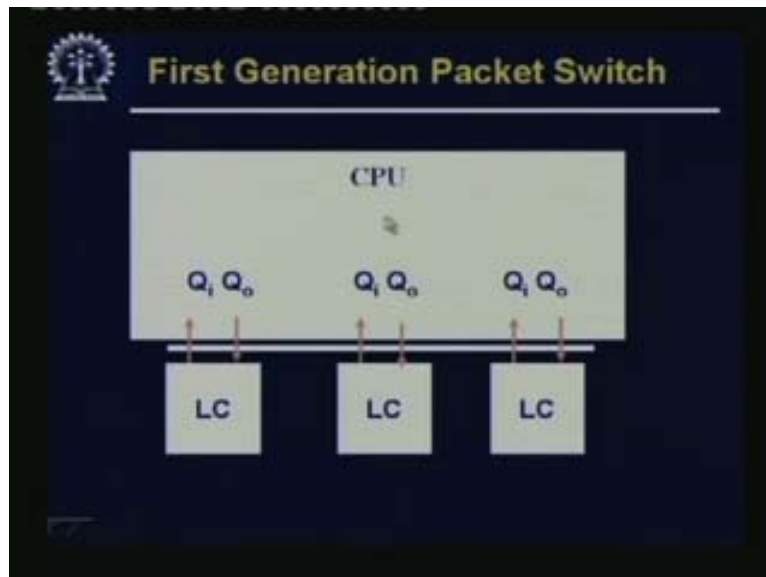
You may queue it right at the input, if you see that the output port is busy, you may queue it at output port or somewhere in-between. Not dropping packets and being fair to all input lines are desirable criteria for packet switch design. That means we do not like to drop packets; we want to buffer them as far as possible. Of course, if a switch is of small capacity, which is swamped by a huge number of incoming streams, etc., maybe for the same output port or something the queue, the switch may be forced to drop some packets. But usually we would not like to do that. We would not like to drop any packet and at the same time, we do not want to starve any particular input line. In general, we like to serve all of them in a fair fashion. We use these criteria for evaluating a design for a particular packet switch.

(Refer Slide Time: 07:37)



The first-generation packet switch went through several generations; we are in the 4th generation or something. Anyway, the 1st generation packet switch is here, it simply uses a CPU, stores packets received online cards in its main memory. These packets are routed to output queues; the packets in the same queue are scheduled by software for access to the output line card. Let us look at this, the simple kind of a situation.

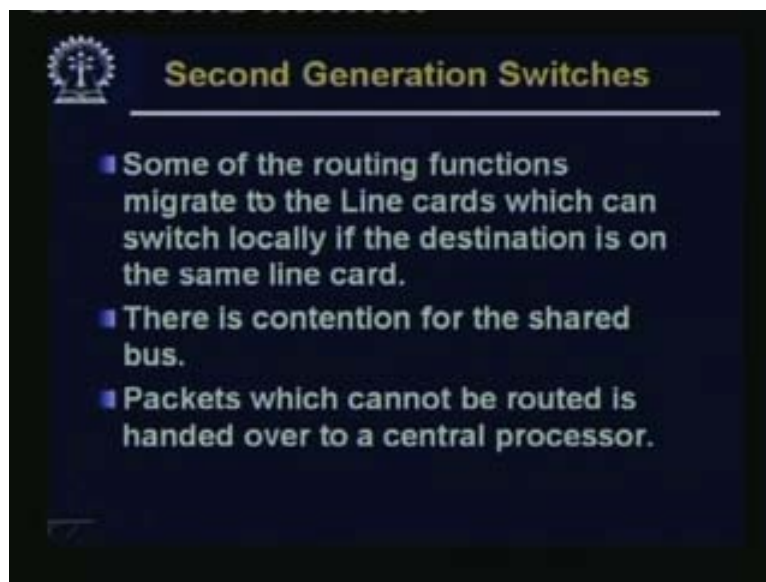
(Refer Slide Time: 07:56)



We have a CPU with some memory. There are particular input and output queues; these are the line cards. What are the line cards? This is a 6-port switch as you see; there are 6 lines, and there are maybe so many ports.

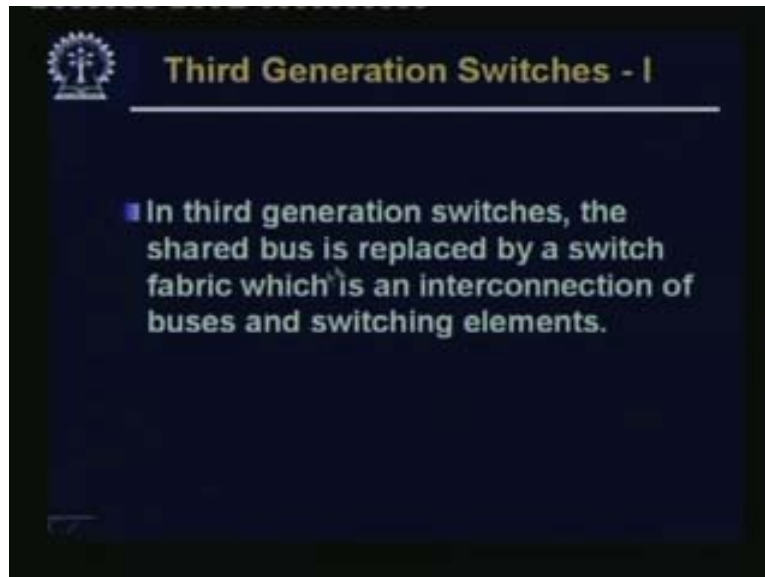
These ports, some of the functionalities are collected together and they are implemented in a small dotter card kind of thing, which is called a line card. A line card will handle just a few of the lines, and there will be several line cards on the same switch. If you go back these packets are routed through the CPU, which does the lookup – it looks at the MAC address, the particular output port to which it must go. CPU will have all that information. The CPU looks up and then it directs that packet to the particular output port. There may be a queue and this queue is scheduled by software for access to the output line card. This was the first-generation packet switch, which is fine – excepting that because of this CPU and memory storage, looking up, tables, etc., it takes time. These switches were comparatively slow.

(Refer Slide Time: 09:27)



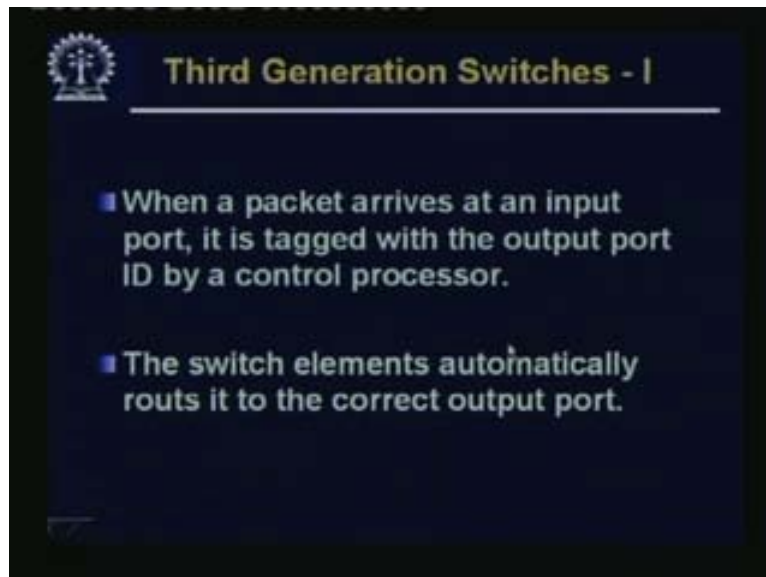
Next we came to the second-generation switches. Here some of the routing functions are involved. Do you remember what the routing function is? The routing function in general means that we look at the destination address and decide which output line this packet must go to. And this routing function, if you are doing centrally, naturally it will be slower. You can distribute it somehow, like distributing some of the routing functions to the line cards themselves. Line cards which can switch locally if the destination is on the same line card – there is another point you can do that – and if a particular line cards sees that the output destination will go through the same line card through a particular line, then the line card need not send it to the CPU at all. This can switch locally. So that was one improvement over the first-generation switch. Secondly, there is contention for the shared bus; that is of course a problem: a contention for the shared bus, and packet switch cannot be routed and the same is handed over to a central processor. This cannot be the local line card, which does not have enough intelligence and knowledge to know how to route it. In that case, that is sent to the central processor.

(Refer Slide Time: 10:56)



In third-generation switches; the shared bus is replaced by a switch fabric, which is an interconnection of buses and switching elements. The shared bus is replaced. If you remember, we can use a bus for broadcast, etc., like we did in knockout switch. That may become a problem because of the contention for this shared bus. So we replace that by the switch fabric, which is the interconnection of buses and switching elements.

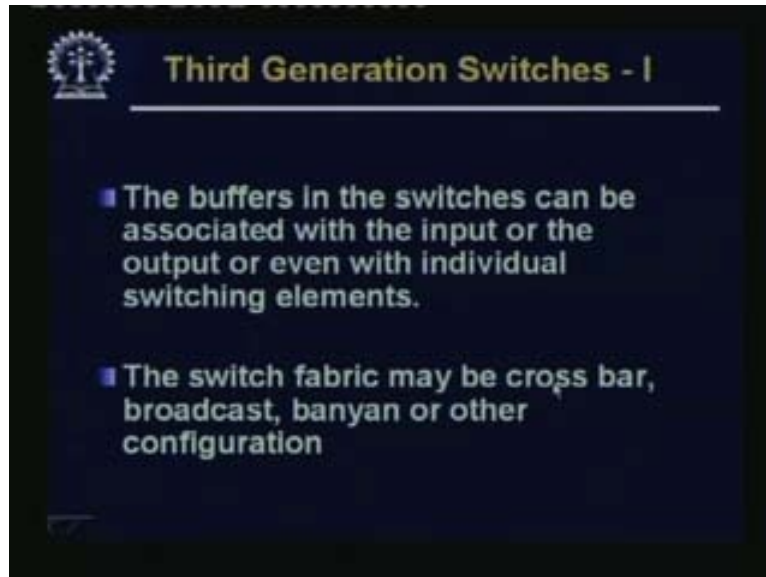
(Refer Slide Time: 11:32)



When a packet arrives in this third-generation switch, which we will discuss in detail, when a packet arrives at an input port it is tagged with the output port ID by a control processor.

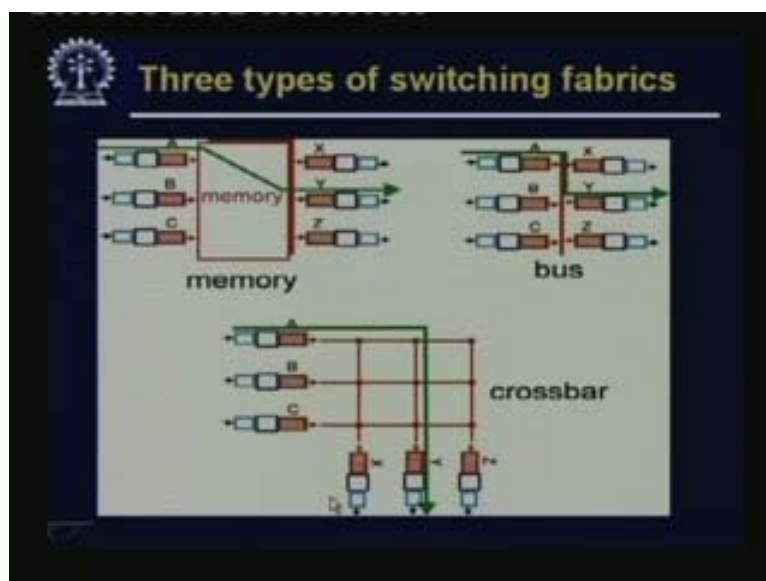
The switch elements automatically routes it to the correct output port, as we have seen, this routing takes place in a MIN. The buffers in the switches can be associated with the input or the output or with individual switching elements.

(Refer Slide Time: 11:52)



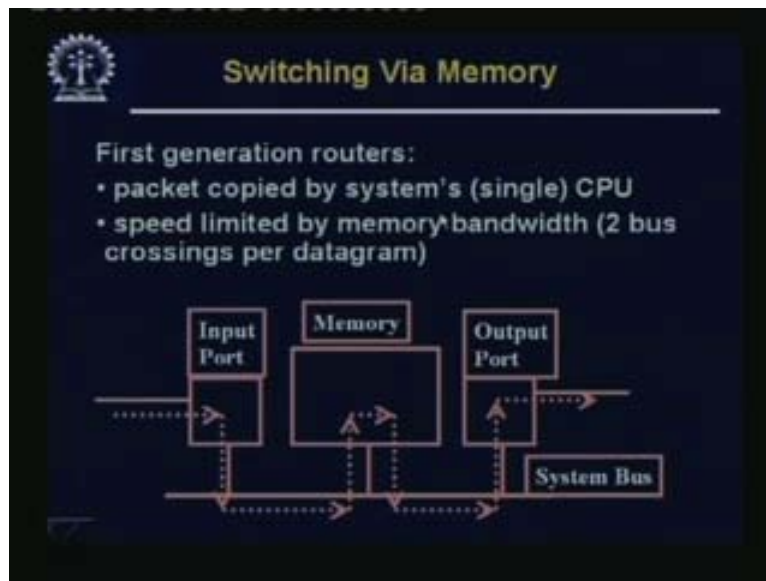
The switch fabric may be cross bar, broadcast, Banyan or other configuration. We have seen some of these configurations; remember the cross bar switches? That means just a simple connection of switches; then we have some broadcast like the knockout switch, maybe a Banyan like a multistage switch or other kinds of configuration.

(Refer Slide Time: 12:23)



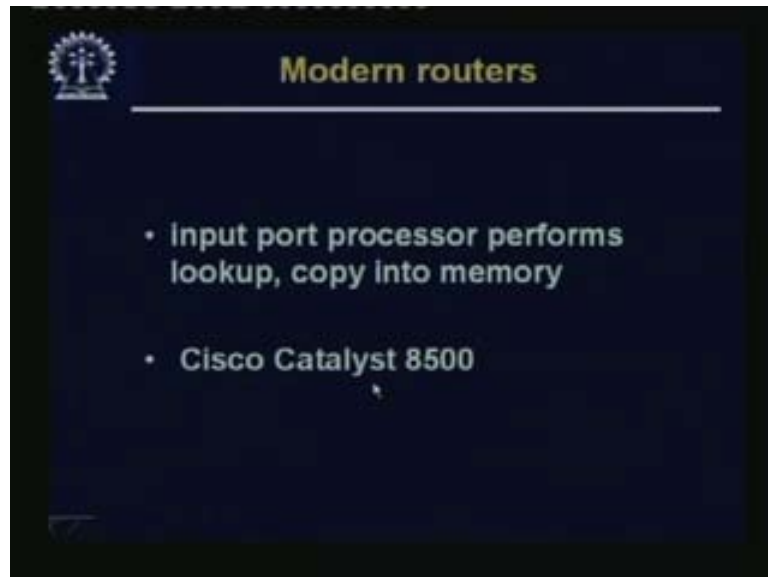
There are three types of switching fabrics, one is that we are using through the memory, other is one which you switch through a shared bus or you have a fabric of switching elements like cross bar – it is the simplest kind of fabric. Of course you can have more complex fabric over here as we have seen, like multistage interconnect network. These are the three different types of switching fabrics that we have.

(Refer Slide Time: 12:54)



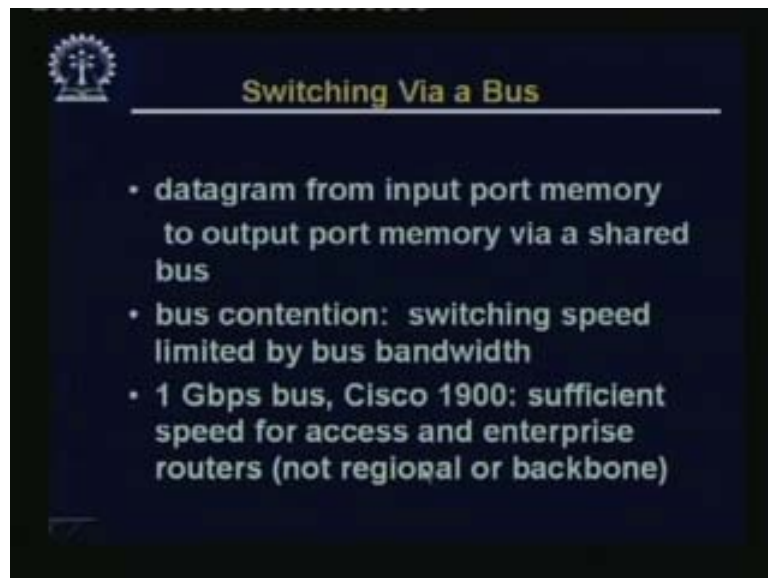
Switching via memory: this was the first-generation routers as we have discussed. Packets are copied by a system, a single CPU. Now, speed is limited by memory bandwidth at two bus crossings per datagram, what is happening is that from the input port, some packet is coming through the system; the bus sent to the memory where the CPU is looking into it and then decides where it must go, then sends it back to the data bus in that output port, which catches it and throws it out. This is the sort of part which gives a bottleneck to this entire scheme because this bus part tends to be slow.

(Refer Slide Time: 13:48)



In modern routers, as I mentioned input port processor performs lookup, copy into memory, etc. An example is – this is short of commercial system – like Cisco Catalyst 8,500, which would take this kind of approach.

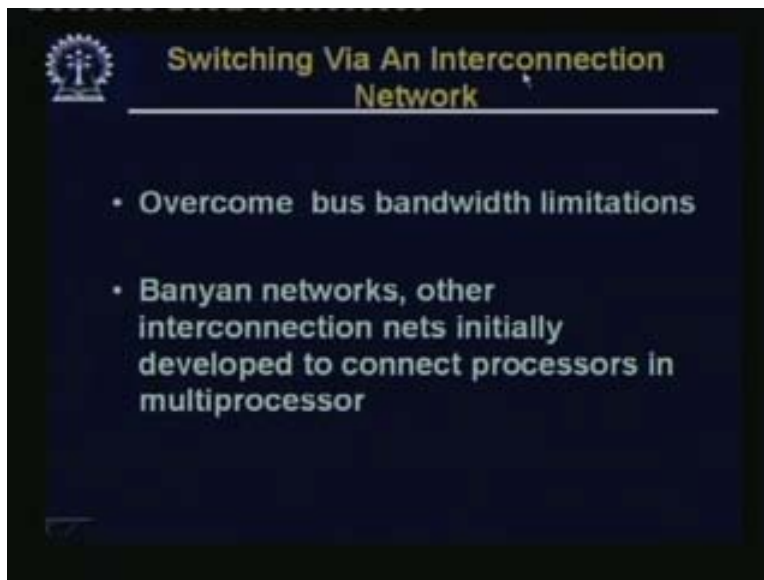
(Refer Slide Time: 14:04)



Switching via bus: datagram from input port memory to output port memory is via a shared bus. Bus contention: switching speed is limited by bus bandwidth. You have very fast buses like 1 Gbps for Cisco 1900, which is low and Cisco switch, which has now been discontinued. It has sufficient speed for access and enterprise routers, not regional or backbone. But this bus structure does not really do well for higher level switches.

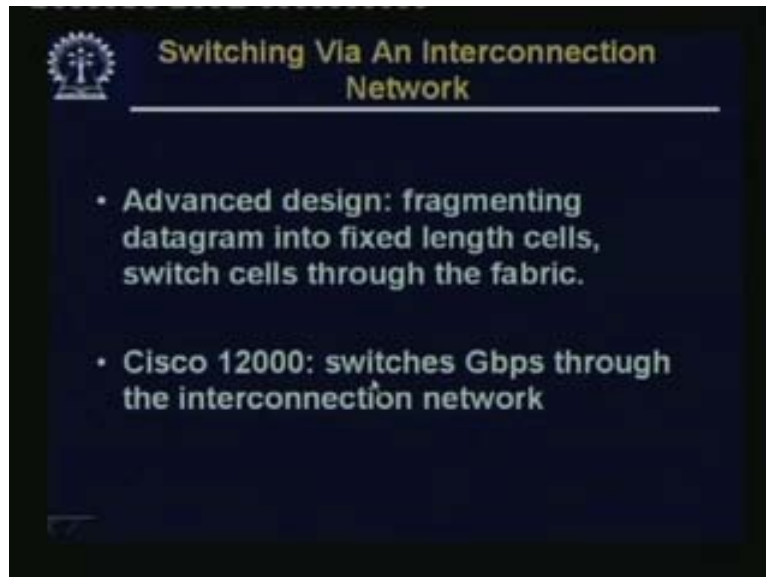
And one example of bus structure is a knockout switch. We have already discussed, just to recapitulate, we have n bus interface units, one for each outlet and each takes input from all the broadcast buses; each delivers output to its own outlet. This is, if you remember the diagram of the knockout switch, you have these buses on which the data is broadcast from the input side. In the output side, particularly bus interface unit for which it is destined, it catches the particular packet; others ignore that packet and so it is queued up over here – like q_1, q_2, q_3, q_4 , etc., and it is pumped out as per this queue: on a first come first serve basis; this basis could be something else. Also if you want to have some other pair, also in particular packets, etc., it's possible. But in the simplest case, this is a first come first serve kind of queue.

(Refer Slide Time: 15:47)



Switching via an interconnection network are just as in the other kind of switches we have seen. It overcame bus bandwidth limitations, that is the basic implication. As we want to go faster and faster, we have to migrate from buses to an interconnection network. We have already seen Banyan networks. Other interconnection was initially developed to connect processors and multiprocessors, etc. It became useful in this class of packet switches.

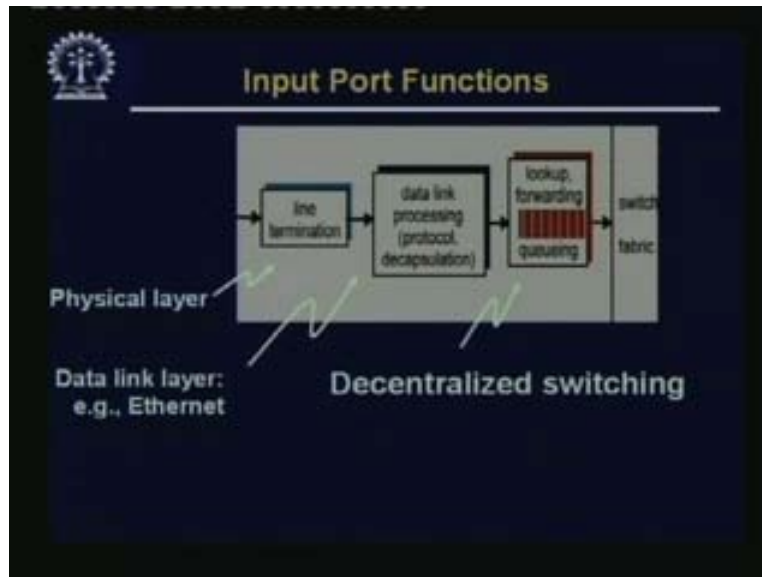
(Refer Slide Time: 16:17)



Advanced design may involve fragmenting datagram into fixed length cells, switch cells through the fabric – this was the idea. For example, this is done in, let us say ATM switches. In ATM switches, what they do is that whatever the packets may come in the form of packets, it may come in the form of streams originally; it is finally broken into fixed length cells. The good thing about a fixed length cell is that you can, if you know length in advance, then you can design hardware so that it can be handled in faster a possible way.

Although the ATM cell has 53 bytes inside the switch, it is something like 64 bytes, these 64-byte cells are fixed length cells and they are pumped as fast as possible. It is for designing very fast interconnection network, in which we fragment them into fixed length cells. Cisco 12,000 switches gbps through the interconnection network: when you go to this gigabit kind of switches, you require an interconnection network or maybe something more as we will see.

(Refer Slide Time: 17:38)



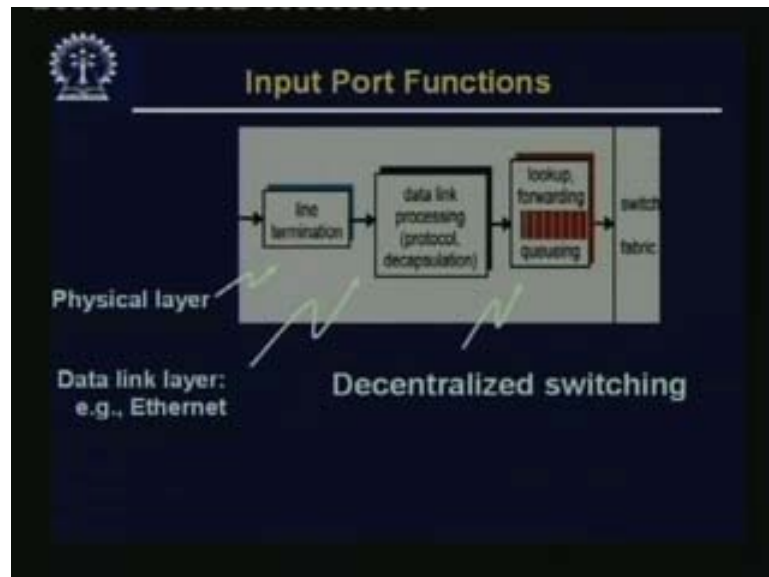
As was discussed earlier, we have three kinds of approaches to handling, contention and buffering, etc. One is on the input side, another is on the output side; the other one is somewhere in-between. We will discuss these one by one; first let us come to the input side but before we go forward, I will digress a little bit, I will just explain a little bit about what is involved in the input side, the input part of the switch. If you remember in our discussion about the several OSI models, in the bottom-most layer, we had the physical layer. Above that was the data link layer, above that we had the network work layer, above that we had transport and other layers, which do not occupy us at the moment.

We will look at these later but we are having the physical layer, the data link layer and then network layer or the routing layer, these three are sort of very closely linked to switches operation. First is the physical layer, that means, physically the link is coming in some fashion. That means there will be certain kind of connectors, certain number of lines are coming per input line when we are saying one input line, but the one input line may consist of a number of wires when you are connecting. For example, if you are connecting through a cat -5 cable, there are a number of wires which will connect to that particular port; that is, a single port. Similarly, in other cases, you have a physical layer, you have an optical connection coming in, and you must have an optical detector over there, etc.

This also means physical characteristics of the media have to be handled by the input port. If you look at this picture again, we have just the input part shown over here: this side is the switch fabric, that means, getting inside the switch so here it is sort of the line card itself. First there is a line termination, which takes its specification, etc., from the particular physical layer that is being used. We have something like a data link layer functionality. An example of a data link layer kind of protocol is the Ethernet. We will, of course, look at details of Ethernet later on. What does the data link layer contain?

The data link layer contains the address called the MAC address, which we have talked about in the last lecture – the MAC address, the media access control address – of the next machine to which it is destined. Please remember that these addresses are addresses of the machine, because the source node knows the destination, let us say. It may not know it also directly. Anyway let us forget about the large network, let us now focus on a particular switch and some nodes connected to this switch. This node A knows that I want to communicate with the node B. Of course node A, B, etc., do not have any meaning at this level. So at this level, some particular address is given called the hardware address, which actually is the address of the networking card. If it is a computer that you have as a node, then the networking card of the computer, the network interface card, the NIC card, it will have some address built into it in the hardware. Similarly, the destination will also have another hardware address. This hardware address is put in this data link layer part and that means the data link layer puts this hardware destination hardware address into the packet.

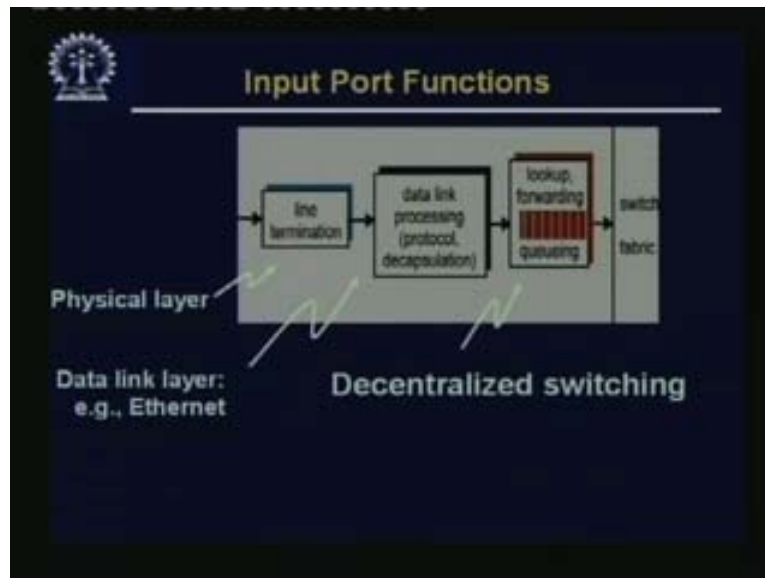
(Refer Slide Time: 21:53)



This address has to be read and then somehow you have to make a mapping from this address. This particular address is connected to this particular line. For that you need some lookup, forwarding, and queuing – if you are doing input queuing, you will do the input queuing also. What kind of lookup? This lookup will tell you that for this particular MAC address that is the output line card. Nowadays, what the switches do is that they keep on sort of getting this information in a dynamic fashion. When a switch is connected to a number of computers, the switch will listen to the network traffic and find out all the hardware addresses, which are connected to these different lines. It will sort of slowly gather this information and store it inside; it will know that whenever some particular node is sending, it will give it the source address, also called the source MAC address. It will now get the source or MAC address and this particular MAC address must be connected to this line.

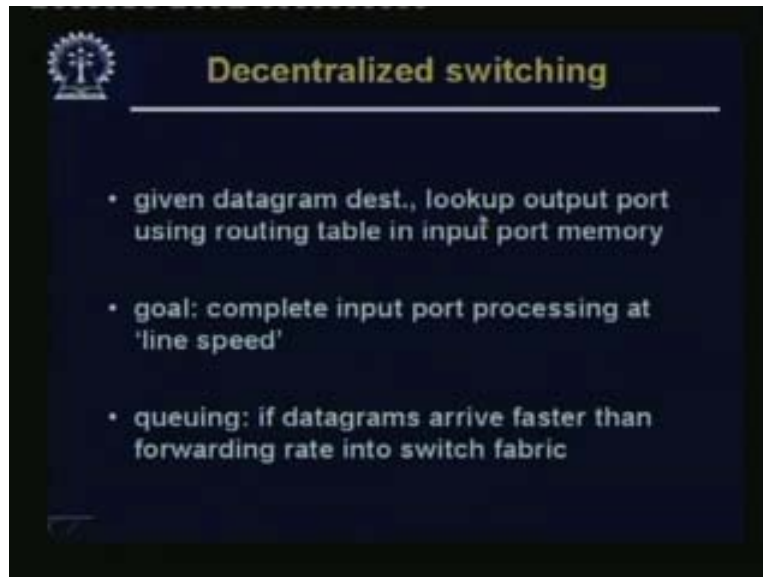
This switch keeps on gathering kind of information, and storing inside; and part of it may be stored locally in the line card. Some part of it may be stored centrally and there are thousands – nowadays, a switch can remember thousands of such MAC addresses, which are directly connected to the switch. There is another level of problem, which is finding about the addresses of remote machines, machines that are not connected to this switch at all. Machines, which are connected through others switches, routes, etc., in a different country, maybe. So that is a different problem; that is a routing problem, but in the data link problem we are concerned with all the MAC addresses of all the nodes which are connected to this particular network which is local.

(Refer Slide Time: 24:07)



So that is stored in this switch; you will now look up the table and decide that that is the final destination output line, which is the forwarding line and then you do the queuing. Then you send it to the switching fabric for its final destination.

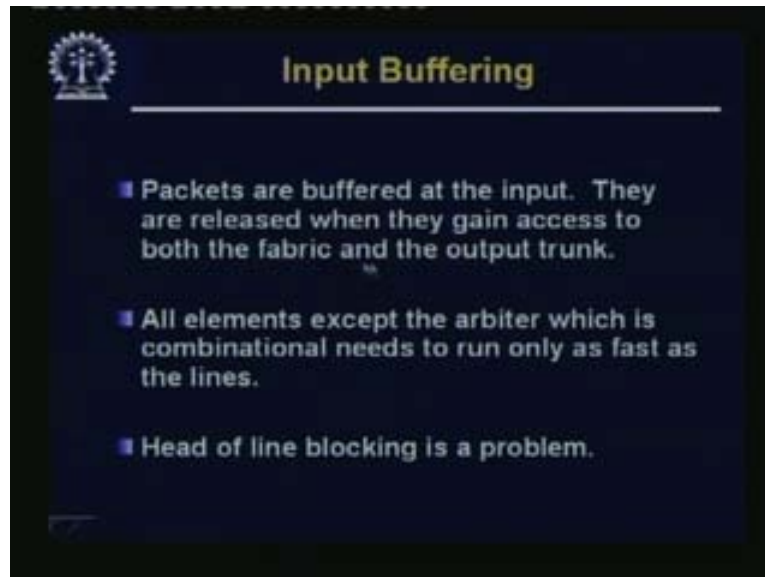
(Refer Slide Time: 24:22)



We will now talk about decentralized switching: given datagram destination, there is lookup of output port using routing table in input port memory, the goal is complete input port processing at “line speed;” so this is what I want to do. We do not want to hold up this particular packet; you get the maximum throughput at the maximum speed. We do not want to hold up any packet in any buffer – ideally that is what we would like to do. So we want to look it up locally at the input port itself, if that is possible. But please remember that even if you can do all this lookup business at the line speed at which the packet comes, even then you may be forced to store the packet for some time.

Because either the output port where this particular packet is headed is already occupied because it is sending some other packet through it at the moment to some intermediate switching element, there may be a contention for intermediate switching elements, etc. For all this contention, etc., we may be forced to put this packet on hold even if we do the lookup at line speed. So we have to queue this in the datagram and the other possibility is that if the datagram arrives faster, then forwarding it into switching fabric and forwarding maybe constrained by all this contention, etc. Then you have to put it in a queue. So what you can do is that you can put it in the input Side; you can buffer it at input called input buffering.

(Refer Slide Time: 26:05)

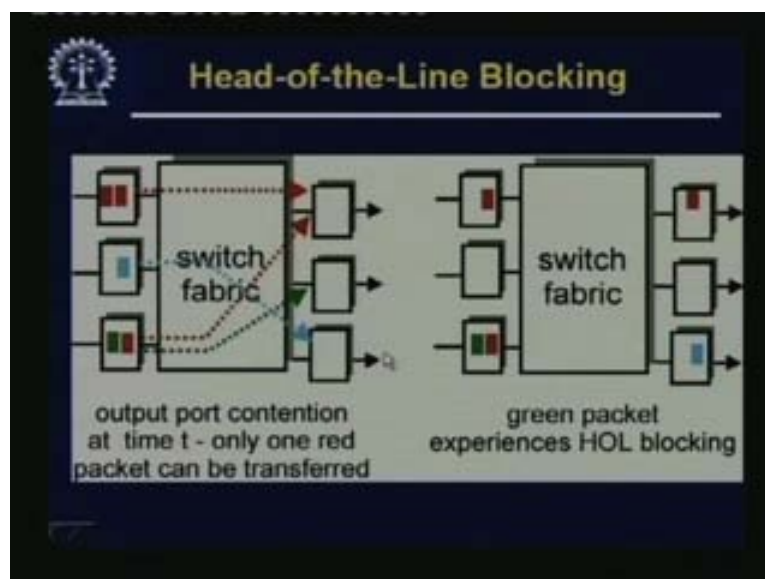


Input Buffering

- Packets are buffered at the input. They are released when they gain access to both the fabric and the output trunk.
- All elements except the arbiter which is combinational needs to run only as fast as the lines.
- Head of line blocking is a problem.

Packets are buffered at the input. They are released when they gain access to both the fabric and the output trunk. That means both the internal switching elements as well as the output port must be free and then this packet maybe released from the buffer, which is held at the input. All elements except the arbiter, which is combinational, need to run only as fast as the lines. So this is a good point about input buffering that elements do not need to be faster than the rate at which the lines operate. In some other cases, we will see that at faster speed, this is a big problem, this input buffering, which is known as the head of line blocking. Let us look at head of line blocking.

(Refer Slide Time: 27:03)

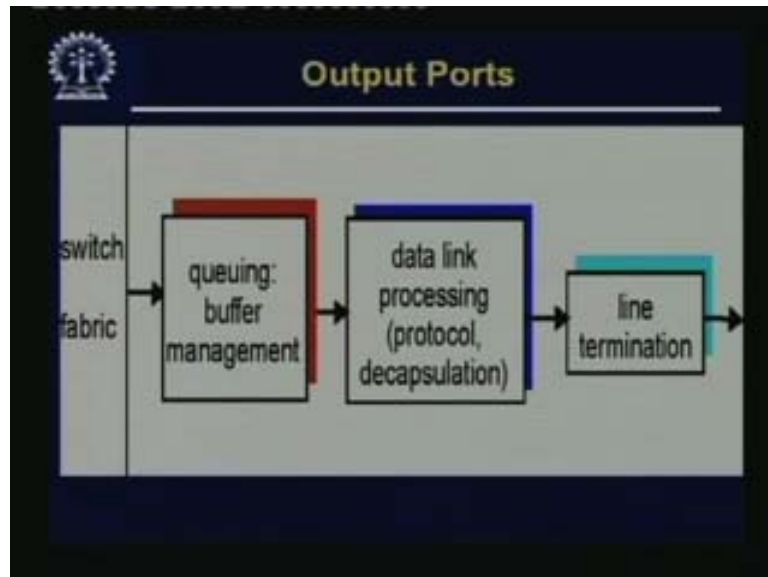


Look at this head of the line blocking – you see what is happening is, here you see there are three input ports, and this is the switch fabric. There are three input ports; the destination is shown by the color that means the following: red means it is destined for this particular port; blue one is destined for this particular port; green one is destined for this particular port. See what is happening, the first packet, red, comes here and goes to the output port; this one is also red, which means it is destined for the same output port although there is no contention in the switch fabric; it has to wait. That is accepted. The trouble is, this green one, which was sort of destined for something in the middle, and which could have gone through this fabric at the same time as this red packet, is flowing at the top.

This has to wait because the queue is blocked by this particular red packet. This is called head of line blocking. At the head of the queue, there is somebody who is blocked and all the people who are behind, all the packets that are behind, have to wait, this is called head of line output port contention time t , when only one red packet can be transferred. So the green packet experiences head of line blocking. This is the head of line blocking, which is a problem with input buffering. Other reason you might require input port queuing is that the fabric is slower than input ports combined. Input ports combined means it is a special case when all the lines are trying to pump data at the same time. What you require is a sum total of all the speed that you get over there, which is not always possible, because then surely there will be contention inside for the switching element.

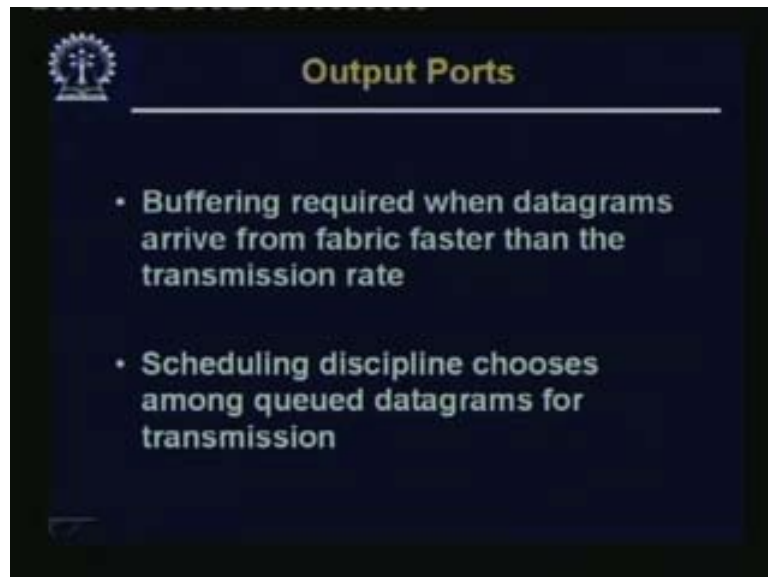
So you will be forced to queue some of it at the input head of line blocking. We have already discussed that queue datagram in front of a queue prevents others in queue from moving forward. So there is queuing, delay and loss due to input buffer overflow. This is the other thing, which might happen; what might happen is that this queue is given for each input port. At each input port since this buffer is not shared, this one is only for that particular input port. Naturally, the size of the queue is limited; if it so happens that data is coming in at a fast rate and the fabric is very congested, it is not getting a chance to go through; this queue at the input port may overflow. If they overflow, which means that some packets have been lost, it is a very undesirable situation; but this may happen. This is the head of line blocking.

(Refer Slide Time: 30:33)



So the other option is to buffer them at the output; that means, you have a queuing buffer management, this part, then the output part is being fed from the switching fabric; queuing buffer management; data link processing, that means, protocol decapsulation for the next jump, etc., and finally the line termination. Once again, you have to go to the physical layer over here. This is the output port queuing.

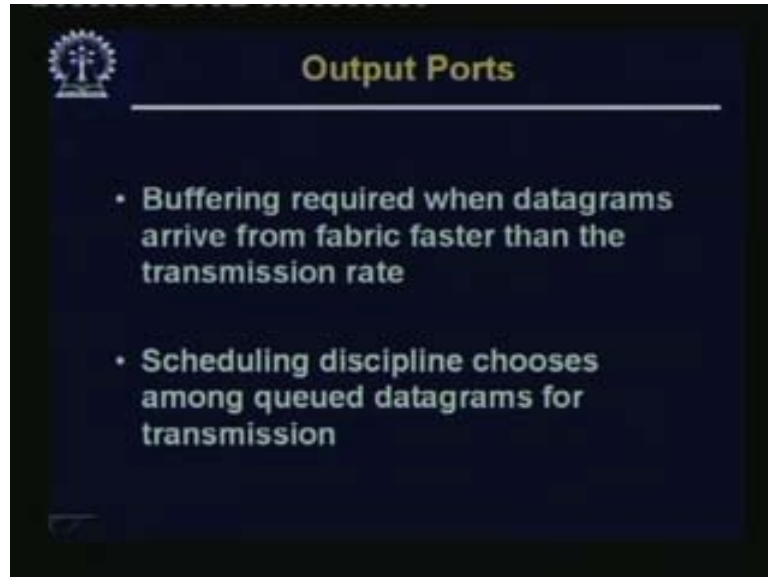
(Refer Slide Time: 31:00)



What are the implications of output port – queuing buffering required. When datagram's arrive from fabric faster than the transmission rate, the fabric is pushing a datagram into a particular output line at a faster rate than the output line can pump it out.

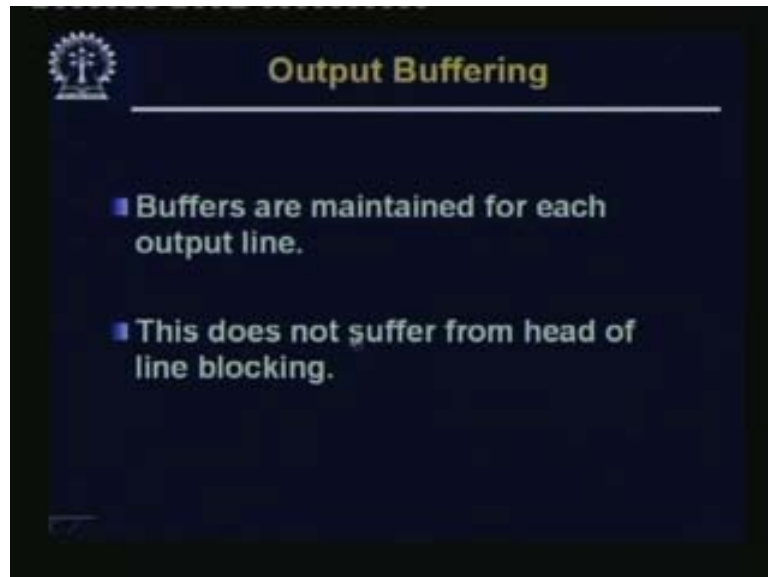
Naturally, packets are getting accumulated near the output port, so packets are getting queued over there. So you develop a queue.

(Refer Slide Time: 3:30)



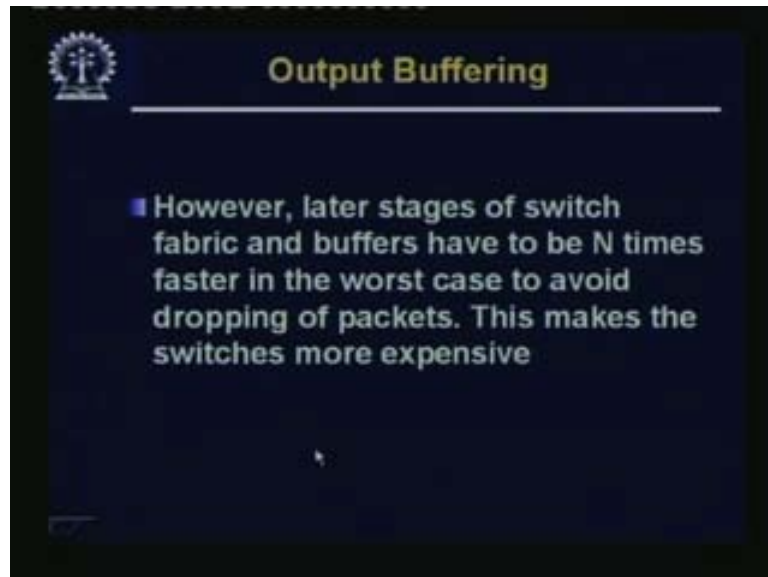
There is a scheduling discipline, which we might choose among the queued datagrams for transmission – which is the one that gets the higher priority, if you are putting some higher priority etc., and even if you are trying to give the same priority to everyone, it might so have happened that we might have kept one particular packet quite long in other places; we may have sort of fed it back; I did something with it as it was already late. We might like to give higher priority than other packet, etc. So there may be a scheduling which is going on in this. Queued buffers are maintained for each output line.

(Refer Slide Time: 32:11)



This is a great advantage, because it does not suffer from head of line blocking. Naturally, because it is been buffering as soon as possible, but there are problems.

(Refer Slide Time: 32:26)



Later stages of switch fabric and buffers have to be N times faster in the worst case to avoid dropping of packets within this space. This rule makes the switches more expensive. Another thing is that suppose, in the worst case, all the N inputs, say $N - 1$ input lines, are trying to feed data into the same output port. What is going to happen is that these packets, which are coming at the line speed through $N - 1$ lines, they sort of start converging to the N th port. How can they converge?

Obviously, if everything is operating at the same speed, then there is going to be contention in these switching elements or one way to avoid this contention is to make switching elements faster. That means suppose two of them have to go to the same switching element but suppose that the switching element operates at double the speed – I am just giving a simple example – at double the speed but at the same rate at which it is coming, it can push it forward. So as you go more towards output port, the speed requirement becomes higher and higher.

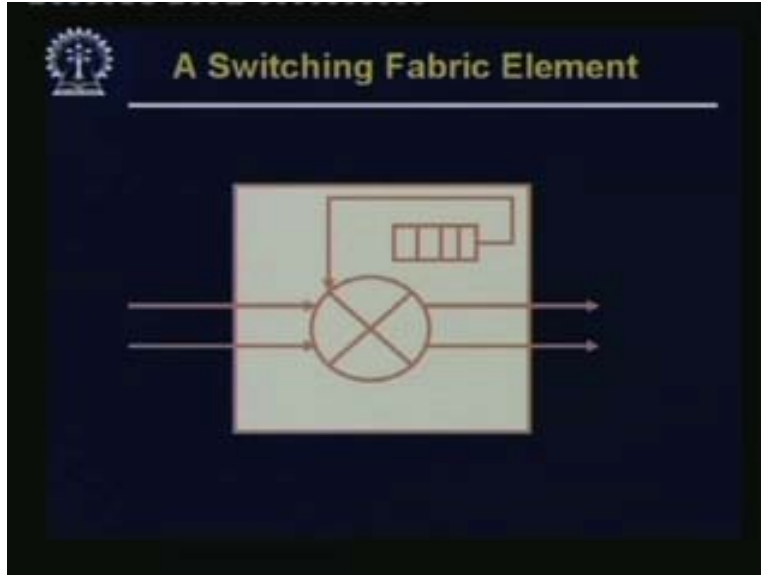
Finally, near the output, at the very point of the output port, it has to be N times faster approximately, which is not done like N times faster. That is not possible but the point is this is a difficulty which we have with output buffering; although we do not have head of line blocking, they tend to be more expensive if you have a buffer fabric. I mean there is always a sort of trade-off between how expensive a particular switch is and what its performance is. Naturally, if you want to go for higher and higher performance where it is very important, then you need to sort of get switches that are more expensive.

(Refer Slide Time: 34:47)



We have the buffer at the input and buffer at the output. The buffer may be distributed in-between all over the fabric, the buffer is associated with each switching element, which is used when the next stage is not available. It needs a lot of memory and quality of connection is difficult to control. The point is that if, throughout switching fabric, if you have distributed our buffer, it is very difficult to have any kind of policy about priorities, etc. The switches will handle it any way they can, so we cannot have a policy which holds over all the buffer because we do not have processing capability over there; it is just a buffer and a switch.

(Refer Slide Time: 35:32)



This is an example of a switching fabric element. You look at this switch in the center. We have the switch and then with two input lines coming in, you have a buffer over here. If the next switching element is busy, then that particular thing can be put into the buffer, from the buffer it will come back to the switch and then get switched when the next element is available.

(Refer Slide Time: 36:10)



Buffered Fabric

- Buffers need to be only as fast as an element's fan-in ratio and during overload, the resulting queue is distributed.

Buffers need to be only as fast as an element's fan-in ratio, and during overload, the resulting queue is distributed.

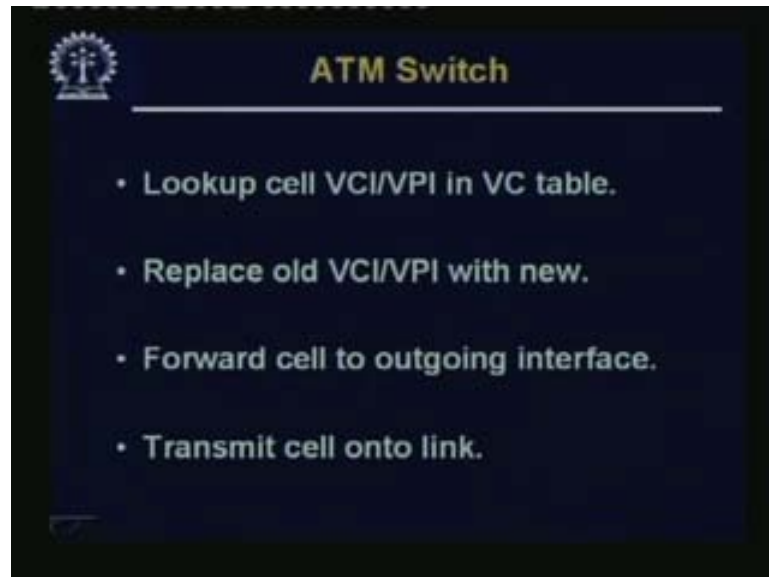
There is one good thing about distributing the buffer – which is that instead of putting all the buffer either on input side or the output side as the output side gets more and more congested in one particular area, what will happen is that the previous switching element will also get congested. They will buffer something and then the buffer will be pushed back. What is happening is that there is a distribution of this buffer all over the fabric. These later stages need not be all the way up to N times faster than the initial stages, which is not always possible. So that is one good thing about buffer fabric.

(Refer Slide Time: 37:04)



We will look at some examples of packet switches; we will talk about three switches. That is, Ethernet switches, ATM switches and IP routers – these are all packet switches. All of them hold very important positions in the world of networking; we will just look at these one by one.

(Refer Slide Time: 37:35)



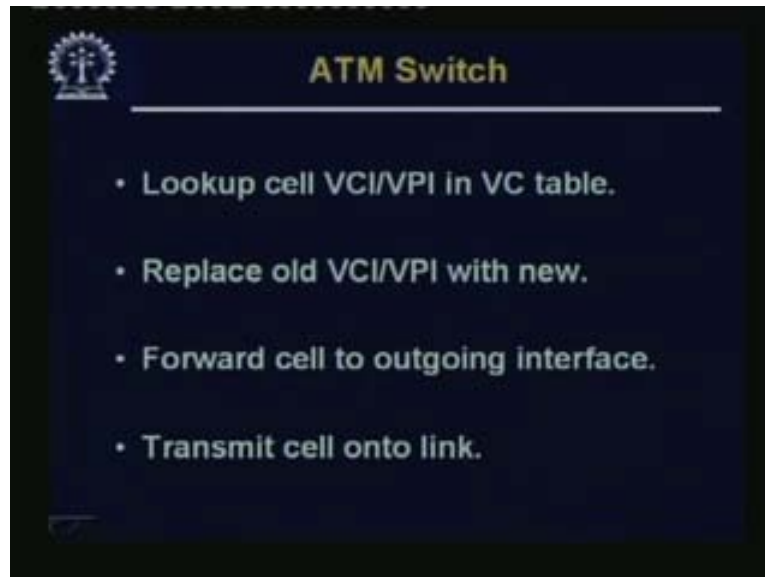
Let us take up an ATM switch first. If you remember, in the ATM system, we have fixed sized cells – in the ATM world, there called cells, then in the Ethernet world, etc., they are called cells packets, datagrams, etc. Anyway, they are called cells; these are of fixed size, like 53 bytes etc. Each cell will contain a so-called destination address and the destination address is in two parts: one is called virtual path identifier and the other is virtual connection identifier – VCI and VPI in the virtual connection table. What happens is that ATM is a set of virtual connections; what is a virtual connection? A virtual connection is somewhere in-between a physical connection on one side and completely connectionless, let's say Ethernet packets, on other side.

In a completely connectionless Ethernet packet, that packet contains the full destination and at each node in-between, this destination is interpreted, looked up into some table, etc., some algorithm is done, and then you decide what is the next step to go to. In a connection-orientated system, physical connection through the switches is set up from the source to the destination. In a virtual connection, virtual connection is also set up. So there is initial set-up time just as there is for connection-oriented, purely physical connection-oriented systems.

Similarly, for virtual connection also, we need a set-up phase and then you can start communicating. In the set-up phase, all the nodes in-between know that we are going to have a session, where the destination is going to be; what would be its next address, and so on – it is computed, let's say, as we do in Ethernet, etc. But once it is computed, it is coded in a very small VCI/VPI. The port one has to go to for this particular session is identified by the VCI and VPI. So this computation is done, recomputed at each of the intermediate nodes, and then the virtual connection is set up; that is all. There is no physical connection.

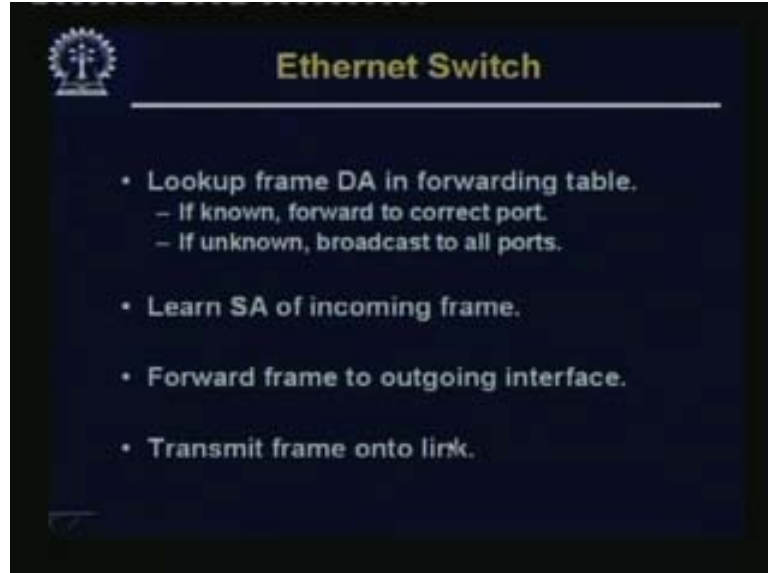
So as soon as there is some cell, which is for this virtual connection, an intermediate node will immediately know that this is the next step and that is where it will push it. So you do not have to do a full-scale look-up, you have a very big table so we look up a very small table of VCI and VPI in the VC table.

(Refer Slide Time: 40:29)



But, of course, just as the virtual connections are set up, they may be taken down. So we replace the old VCI/VPI with new ones; forward cell to outgoing interface transmit cell onto link. This is how an ATM switch works. You will see that the basic concept of the cell remains the same; that means, we have to find out the output port from the address. Once the output, the addresses are set up, it is simply a small table of VCI/VPI, which make this look-up process much faster. With this, together with the fact that the cells are of same size, you can develop your hardware in such a way, so the ATM switch will work very fast. Otherwise, the rest of the switch and switching principle remains the same as we have discussed already.

(Refer Slide Time: 41:40)



Next comes an Ethernet switch, which is perhaps the most common type of switch that you see almost everywhere. As a matter of fact, the number of Ethernet switches which are being used almost regularly, is so high that the cost of switches has also come down. It is because if you produce something in large numbers, you can amount the cost of its development, etc., over a large number of switches, so the cost of switches comes down. Ethernet switches have become very cheap these days (switches which used to cost maybe 1 lakh or so 10, 15 years back, cost only few thousands of rupees today).

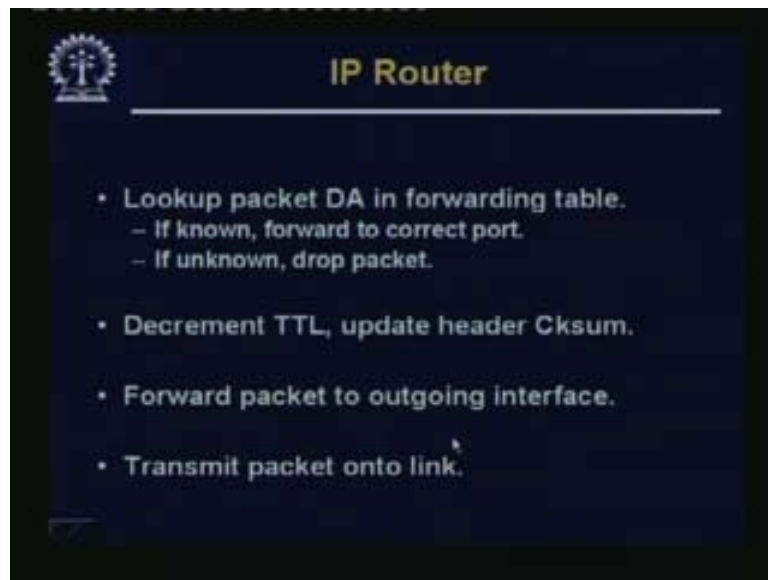
This is the most common type of switches, which we see everywhere. If it is an Ethernet packet, then it has to handle the Ethernet addresses, which are in the packet. The Ethernet address is actually a 6-byte address, so we will have a 48-bit address. Usually we do not get to see it as users, because this is 6 byte; it is very difficult to remember such a long address, and these addresses come in a very random fashion. It is very difficult to remember Ethernet addresses, but human beings need not remember Ethernet addresses.

Ethernet addresses are learnt by the system; when a system is put in a network where the system maybe a computer or a node or the system maybe a switch or a router, they learn about the Ethernet addresses, local Ethernet addresses, which are there. We will get into the details of Ethernet, etc., later on. But anyway, this lookup is necessary. The first step in Ethernet switch is look up frame destination address in forwarding table; if known, forward to correct port; if unknown broadcast to all ports. So this is what an Ethernet switch does. Suppose a packet comes with a particular destination address, and if the destination address is known, the node having this particular address is connected to this particular port. Then the output port, for this packet's output port, is forwarded to the correct port. If unknown, you broadcast it to all ports. Why is this?

If you remember as I mentioned earlier, these switches automatically learn from whatever is going on; they automatically learn the MAC addresses of the local machines. But suppose it is put in a network, in the beginning it does not know anything. What does it do? What it does is that, as soon as it comes from some port, it broadcasts it to all the other ports. It will listen and find out which is the one that really was successful and there are various protocols for finding these things out; it will slowly learn. It may so happen that this switch may get a packet for which the destination address is not known to the switch. In that case, it broadcasts to all the ports.

Then we have, learn source address of incoming frame – one thing of course is learned from this frame, although the destination address is unknown, it knows the source address; the source address is the particular incoming link it came in through. This particular address must be the node having a particular address, must be connected to this port so that later on if this particular node comes as a destination, it knows where to send it to. Forward frame to outgoing interface – that is the other thing, transmit frame onto a link. These are the various functions of an Ethernet switch.

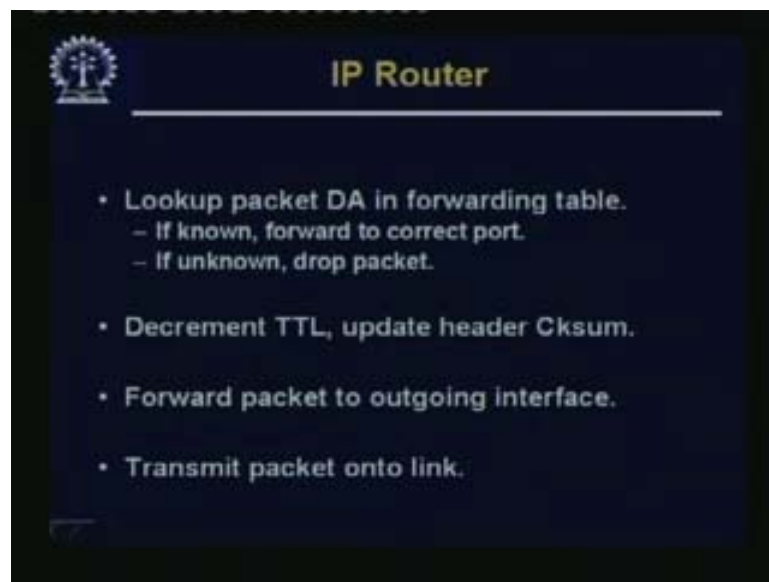
(Refer Slide Time: 46:07)



We have something called an IP router; router as you know works at the network layer; it is not in the data link layer. That means a router has a different set of address called IP address; so IP address is a global kind of addressing scheme. There is some scheme for this IP address although not as good as the telephone numbering scheme, but anyway, there is some central control. The job of the IP router is to find out this global address that I have got. Maybe this particular node is in some other country; it has to understand that for going to that, reaching that final destination, I have to take this link locally. Then it will send it to the next level, that is, the data link level, for hopping that particular local layer. Once again, we have the same problem that we have got an address, some kind of destination address, which happens to be an IP address. We have to do that, look up, we have to decide this is the port it must go to and then send it to the port.

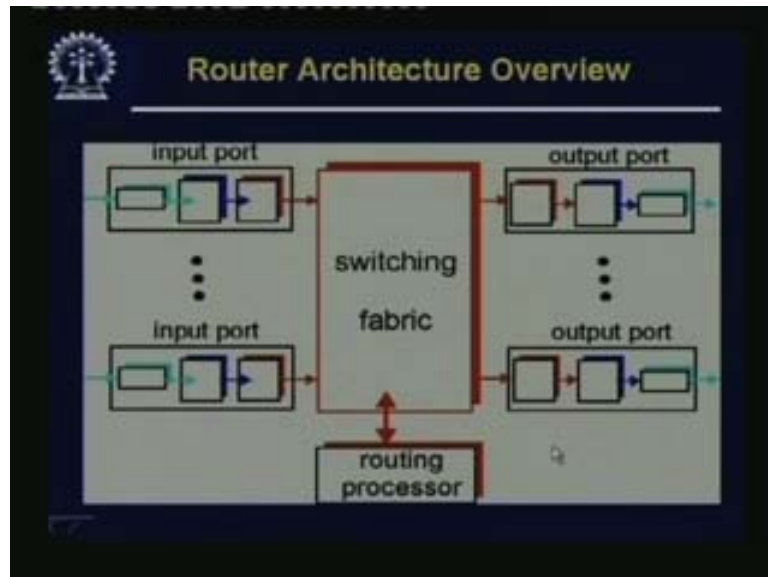
So the same switch is used with some added processing capability for doing the process router – usually it comes with all kind of other functionalities and also it needs some processor of its own and it needs some memory naturally for running programs, etc. There is some kind of an operating system, which is there in a router, so a router is more complex, but the hardware part, the major part of an hardware, is still the switch and the input and output line ports. Usually a router can handle different types of ports; for example, usually an Ethernet switch will handle only Ethernet port. So the physical characteristics or the physical layer properties of the ports are constant for all the ports, but a router can cut different types of cards, which can handle different types of ports: maybe serial port, high-speed serial port, and all kinds of things.

(Refer Slide Time: 48:27)



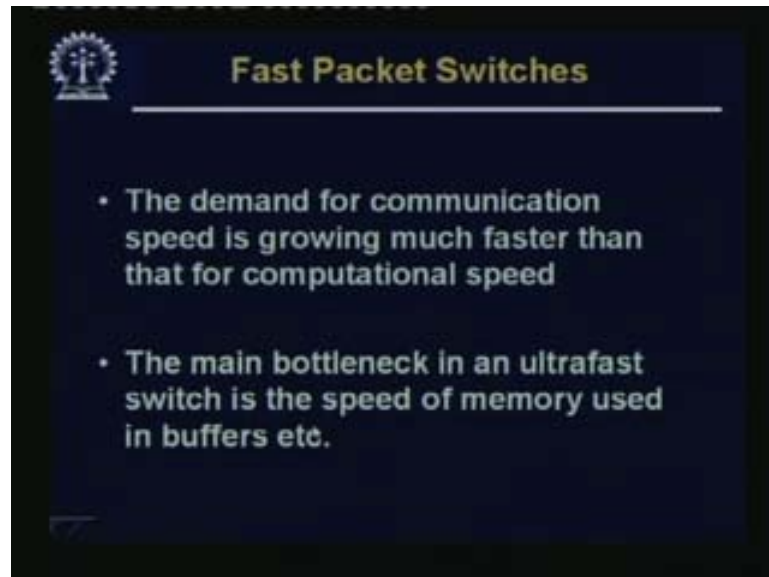
But the basic thing remains the same look-up, the packet destination address in forwarding table – if known, forward to correct port; if unknown, well, the router cannot broadcast it because if all routers start broadcasting then there will be broadcast all over the world through all the routers, which is not really acceptable at all. You can broadcast something only locally by an Ethernet switch; so if the destination is unknown, and it has no way of finding where to send it, it simply drops the packet. When it sends, it decrements the time to leave, update header, checksum – so these are some of the side-effects which it has to go through – let me not go into the details of this – and finally forward the packet to outgoing interface, transmit packet on the link as usual.

(Refer Slide Time: 49:14)



This is a router – once again, we have a switching fabric, we have input port and we have output port, and we have a routing processor.

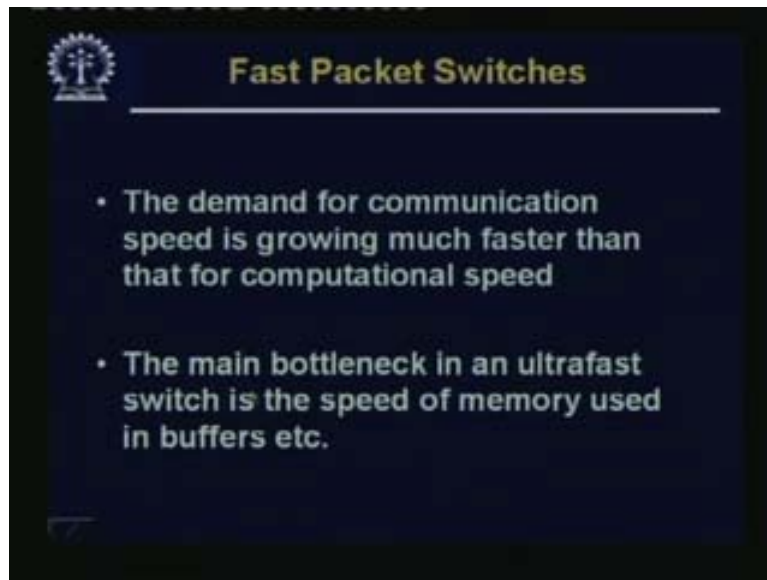
(Refer Slide Time: 49:27)



Let me quickly look at some of the issues in developing fast packet switches. First thing you must know, understand, is that the demand for communication speed is growing much faster than that for computational speed. You know that in computers there is this Moore's law, which tells that every one-and-half years maybe, the speed doubles; the memory capacity maybe doubled; etc.

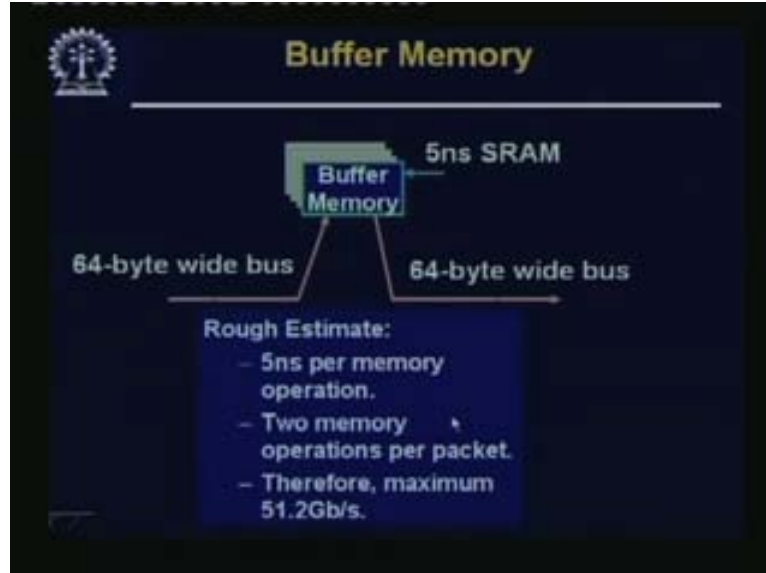
But if you look at the way growth has taken place in the communication field, especially with this speed of communication which is required, this is much faster than the similar developments in the computational area. There are two reasons for this: one is, of course, people are trying to communicate more. People previously were just satisfied with voice; now they want to send data, they want to upload or download large files, they want to send movies, etc. The individual demand for bandwidth is growing, that is one thing; and the other thing is that more and more people are getting connected to the network. As more and more people get connected to the network, the number of possible connections goes up as N^2 ; if there are N people, as N goes up, it goes up in the squares so all this is going through the central backbone, etc. At the backbone level, the speed requirement is very high, so we require very fast packet switches and we will look at just some issues over here.

(Refer Slide Time: 50:57)



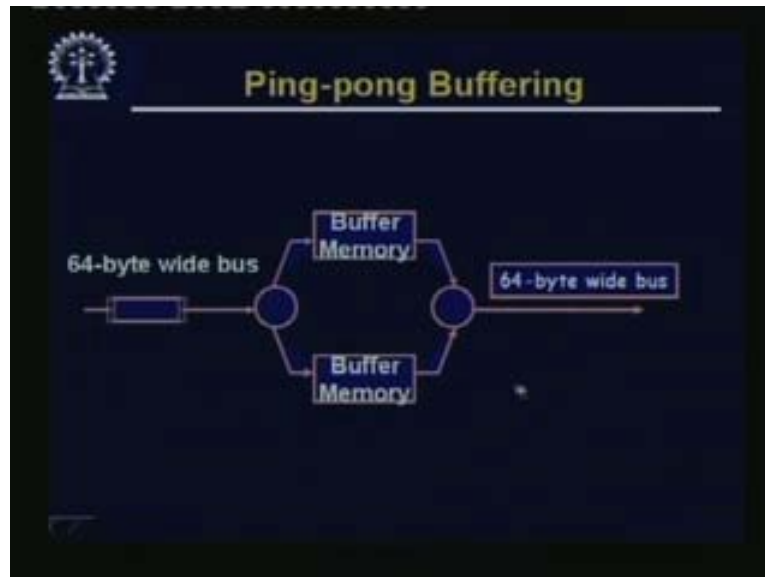
The main bottleneck in an ultra fast switch is the speed of memory used in buffers. This is the main bottleneck.

(Refer Slide Time: 51:09)



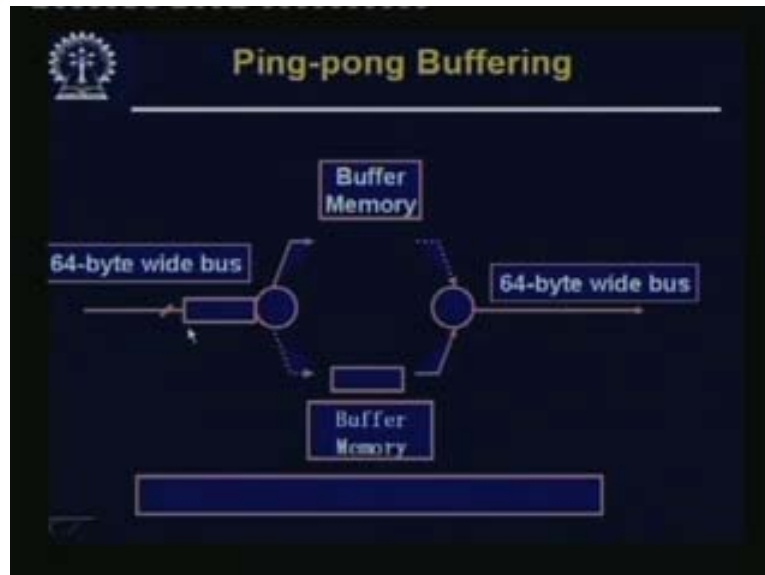
For example, if you have a 5-nanosecond SRAM, that is, static RAM – it is the faster kind of RAM rather than DRAM and you have 64-byte wide bus –you have 5 ns per memory operation: that is your memory speed, two memory operations per packet. You have to write it there and you have to read it, so that makes it 10 ns and you have 64 bytes, which means 64 into 8, that is, 512 bits – so you are handling 512 bits in 10 ns, which comes to 51.2 Gb by s. This looks like a very high figure but now, we get switches, which have speeds of maybe 100 or even 200 GB by s. They are very high speed, so the buffer memory is the maximum. This is not as simple as that; ultimately it comes to about 40 Gb by s or so. We use the fastest possible RAM but even that is not enough. So we can take some possibility: One is the ping-pong buffering, which means the packet comes over here.

(Refer Slide Time: 52:18)



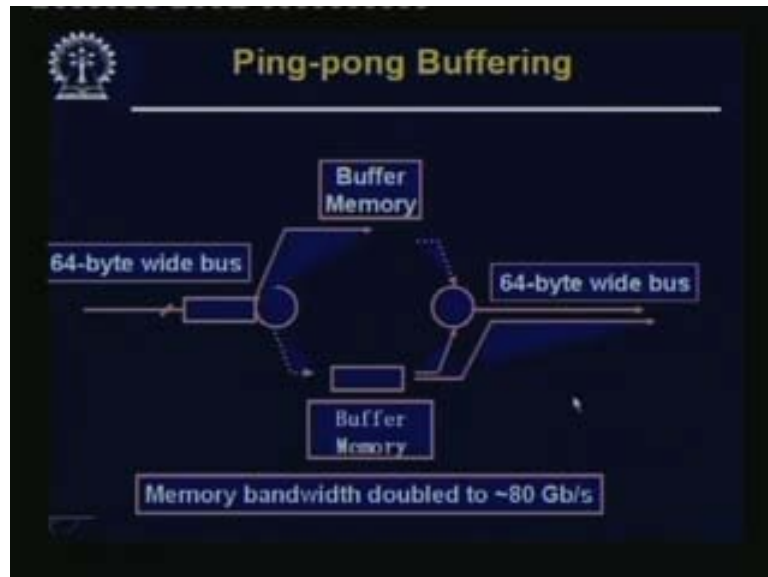
There are two buffer memories: one is that either you send it or you put it in a buffer like this.

(Refer Slide Time: 52:34)

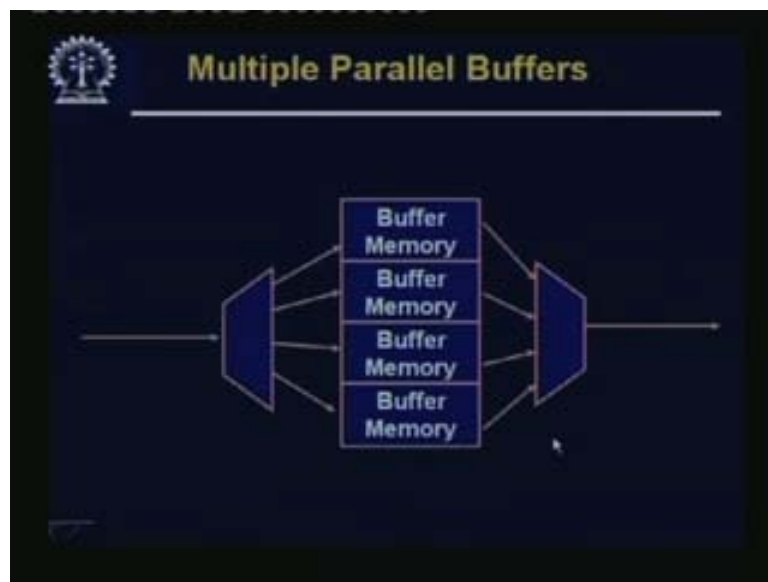


For example, when it comes, maybe one of them is being sent while the other is being buffered. Then this one will be sent and the next one that comes, will be buffered. So you are trying to double the speed at which it is operating. You require two buffer memories for it. That is one approach.

(Refer Slide Time: 53:00)

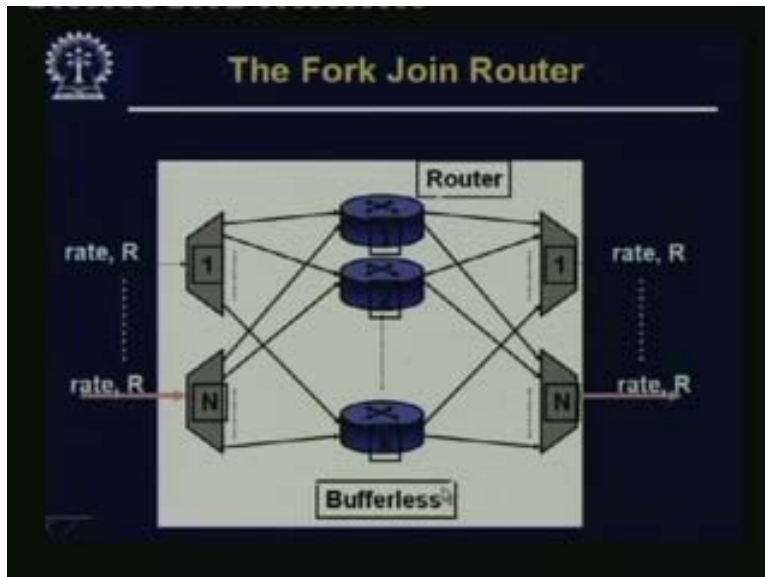


(Refer Slide Time: 53:02)



The other approach is to use multiple parallel buffers; instead of two, you have many of them and you put them in buffer in parallel, so you can go about it maybe in a round robin fashion, put in an multiple buffer. And the other bottleneck is that, especially in a router, if that your processor, if that is centralized, then that becomes the bottleneck for speed.

(Refer Slide Time: 53:27)

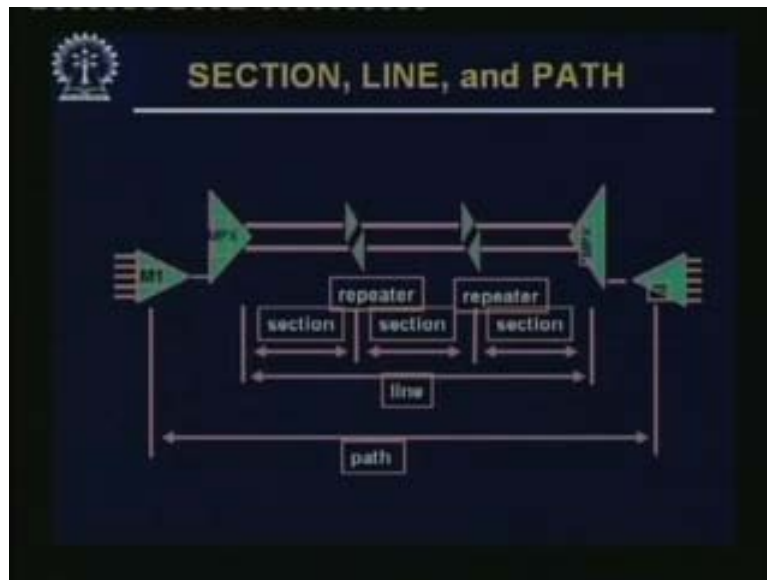


We have the fork, the so-called fork join router, where maybe you have put a number of routers in parallel. These lines are coming at a particular rate; they are distributed over all the routers so the routers individually do the look up, etc., at whatever speed they operate on. Then they keep on pumping this side as well as this side – these are combinational so they can operate very fast – the look up, etc., is paralyzed. These do not have any buffers in them, this part has the buffer. So this way, by going in for parallel operation, we try to make as fast switches and routers as possible. This speed is going on increasing as they have to increase in order to cater to the growing demand of packet switches, speed of packet switches, and capacity of packet switches, all over the world. Thank you.

Preview of next lecture
Computer Networks
Prof. Sujoy Ghosh
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture No - 9
SONET/SDH

This lecture, we will discuss about SONET. The word SONET stands for Synchronous Optical NET work sonnet in the USA, Canada and Japan, synchronous digital hierarchy. Elsewhere for example in India, We will be calling SDH; this is a time division multiplexing system that transmits a constant stream of information

(Refer Slide Time: 55:14)

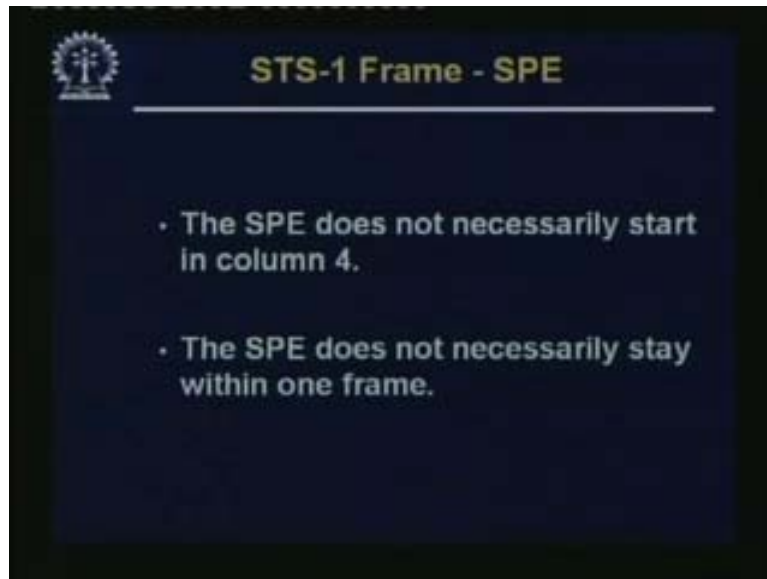


Please look at this figure so we have some multiplexers. As the figure shows, we have a multiplexer in this side another which is an output fits to another multiplexer, this multiplexer is going in this direction and then after some time, the signal becomes weak. We want a repeater that means what is a repeater. A repeater is something, which boosts the signal strength, so there is a repeater, then it travels some more distance and then there is a repeater again and then it travels some more distance and the other side. We have the corresponding demultiplexer and then it fits into the other demultiplexer. From repeater to repeater, we call it as section so all from repeater to multiplexer this is also a section.

Multiplexer to repeater, repeater to repeater, these are called sections and then from multiplexer to multiplexer we call it a line. At the repeater nothing happens except the signal is cleaned up, signal maybe boosted or there may be other cleaning operation synchronizing operation etc that maybe done at the repeater but as such the signals which are traveling here are the same set of signals are traveling here.

At a multiplexer some of the signals may go off in another direction, some signals may go in some other direction etc. At the multiplexer there maybe convergence or divergence depending on which way the signals is flowing so that may happen at a multiplexer so from multiplexer to multiplexer, we call it a line and then from the end user part point to end user point, we called it a path.

(Refer Slide Time: 56:57)



The SPE does not necessarily start in column 4, which means that the SPE that does not necessarily stay within 1 frame, these are 2 very important points in. On it, the point is that all through, you have 87 columns but actually, the data may start being transmitted as some arbitrary point inside that 87 columns. What is the idea? Why do want to leave something and they only start from the middle? The point is that if there are some kinds of mismatches because of miss matches of rate etc, if everything in the world were synchronous, all activities an all equipments etc then would have started from the beginning but that is not the case.

This is where we absorbed this kind of variation and this gives a great flexibility to solve it, which was not there earlier. Other interesting thing is that SPE does not necessarily stay within 1 frame which means, that the SPE start in 1 frame and then 1 in another, we will just look at a diagram of this. Let us have a diagram of this, you see the SPE, this light green color that really starts from somewhere. After leavening some of the root, starts here in the path overhear is somewhere here and there are 2 frame is here so the SPE is really spent in both the frames.