Digital Voice and Picture Communication

Prof. S. Sengupta

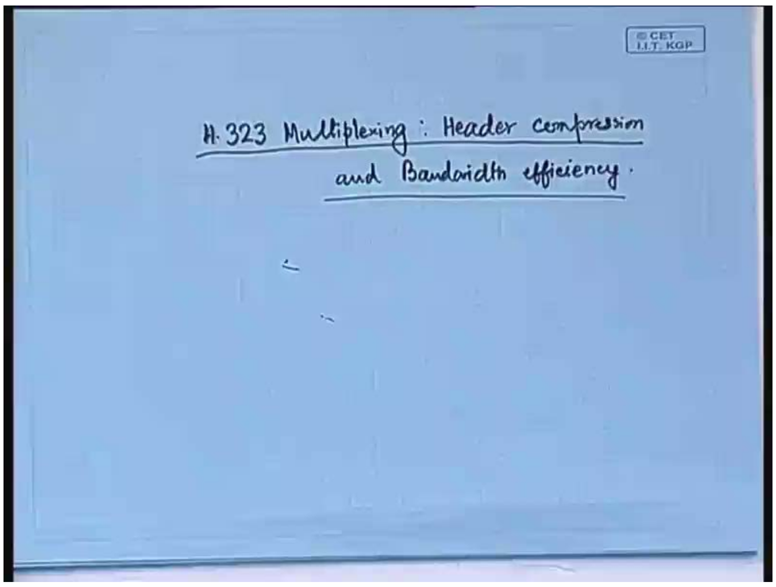
Department of Electronics and Communication Engineering

Indian Institute of Technology, Kharagpur

Lecture - 37

H.323: Multiplexing: Header Compression and BW Efficiency

(Refer Slide Time: 00:57)



H. 323 Multiplexing : Header compression and Bandwidth efficiency.

H dot 323 multiplexing:

Today we are going to talk about the scheme for the header compression which we had in fact introduced in the last class and subsequently as a result of the header compression what kind of bandwidth efficiency one can achieve by using the H dot 323 multiplexing. This is what we are going to discuss. And before we take up the topic, just a very quick brush up with what we were doing last time.

Now the essential idea behind the H dot 323 multiplexing was that there we will be using a multiplexer where several sources may be able to communicate with the destinations in a simultaneous manner and in a time multiplexed manner. The advantage that we are looking in this is a kind of a bandwidth efficiency in the sense that instead of using the individual headers which come from the different RTP streams which are associated with the individual source force.

We will be using the multiplexing where before we transmit the multiplexed bit-stream through the wide area network over again the UDP/RTP packet there what we have to do is to form the packet header in a manner that is efficient. That means to say that we would like to avoid the redundant information that goes with the packet headers. the redundant information comes in the sense that for some particular streams...........; when a session has been established between one channel with one of the destination nodes when a session has started then for a session we have to find that there are certain fields in the UDP/RTP header which remains the same and there are some fields which go through the constant increments like the time stamp, the sequence numbers those will go through some constant increments.

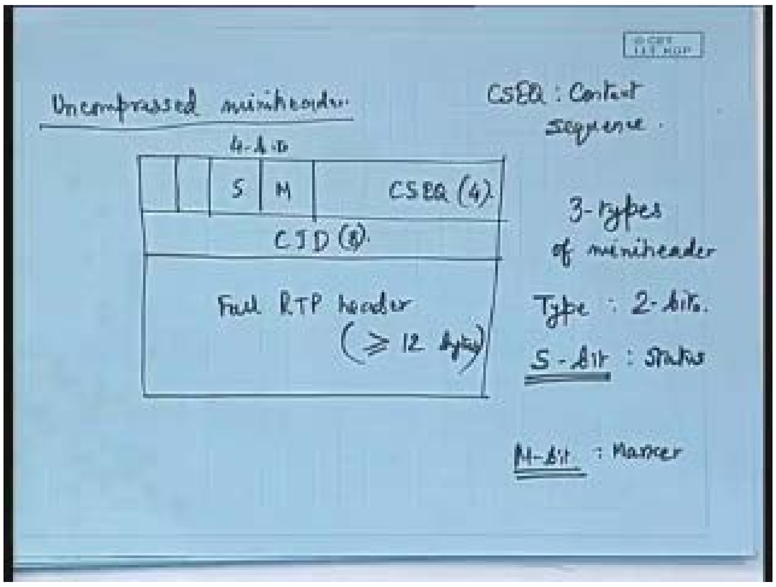
Now because there are some constant fields and for some of the fields there are constant increments that is the reason why one can go through some amount of header compression whereby when we are multiplexing the streams into one, there, the miniheaders that we form for every stream; for every source destination combination when we have a stream there we will be having the header information which is there in a compressed form.

What I have described is that there are three types of miniheaders which are possible: one is the uncompressed mode in which case we will be having the full RTP header (Refer Slide Time: 04:33). In fact somebody was asking me that what is the composition of these 4 bits. Basically there are three different types of miniheaders, three types of miniheaders meaning that what type of miniheader it is that is indicated by 2 bits; type is indicated by 2 bits and there are other 2 bits which are referred to as the S-bit and the M-bit.

S-bit stands for the status bit and in fact it informs the status of the bit rate of the codec means assuming that the codec goes through two different bit rates, so which bit rate it is that will be indicated by this S-bit and M-bit stands for the marker bit. The marker bit is one when the first when any sessions starts the first packet would contain the M-bit as 1 and the subsequent packets will contain the M-bit as 0. So the status and the marker bits.................. so 2 bits for the header information and two bits for the status and the marker so this will form a 4-bit field and then we will be having a 4-bit sequence number and 8-bit CID which is the context

ID and this is the context sequence number and in the uncompressed miniheader the full RTP header will go.

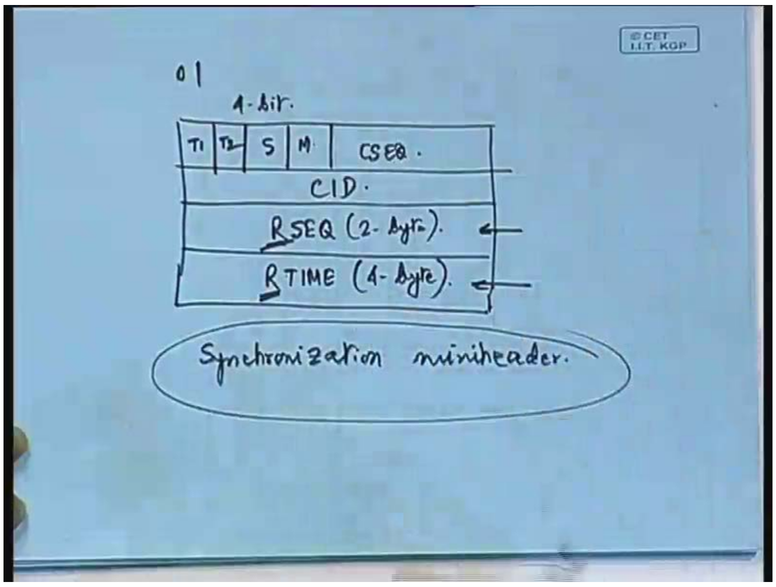
(Refer Slide Time: 6:20)



Therefore, as you understand that initially we have to send the full RTP header because then the demultiplexer will not be having any information about the fields which the multiplexer is using.

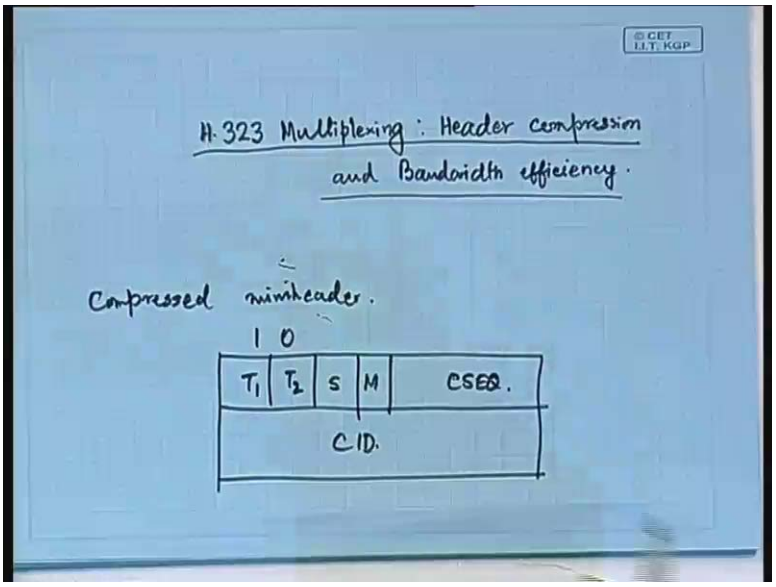
The second type of miniheader we talked about is what is called as the synchronization miniheader where the complete miniheader is not given; the complete RTP header is not given but other than this 4-bit field that means to say the type number and the S-bit and the M-bit so T1 and T2 (Refer Slide Time: 7:02) are the type number that what type of miniheader it is; so if you let us say put (0, 0) for the uncompressed you will be putting another one may be (0, 1) for the synchronization miniheader, the C sequence and CID remains as it is but you have to add the extra R sequence and R time. So, from time to time one has to send this synchronization miniheaders, this will not be avoidable.

(Refer Slide Time: 07:28)



But most of the times what we want to do is to go through a third type of miniheader which we will be calling as the compressed miniheader. In the case of compressed miniheader it is just a 2 byte miniheader; fully compressed miniheader only has got 2 bytes and the first byte that contains the usual 4-bit information like the type T 1 and T 2 where we can use say 1 and 0 combination and then the S-bit and the M-bit and then the C sequence which will be 4 bits and then the CID the context ID. Because normally when the receiver and when the multiplexer and the demultiplexer they are having a perfect understanding then the CID itself should be good enough in order to identify the packet. So CID should be CID and C sequence should be enough of information, we do not require the entire RTP header streams that is required. So that is the factor that really contributes to the header compression.

(Refer Slide Time: 07:42)



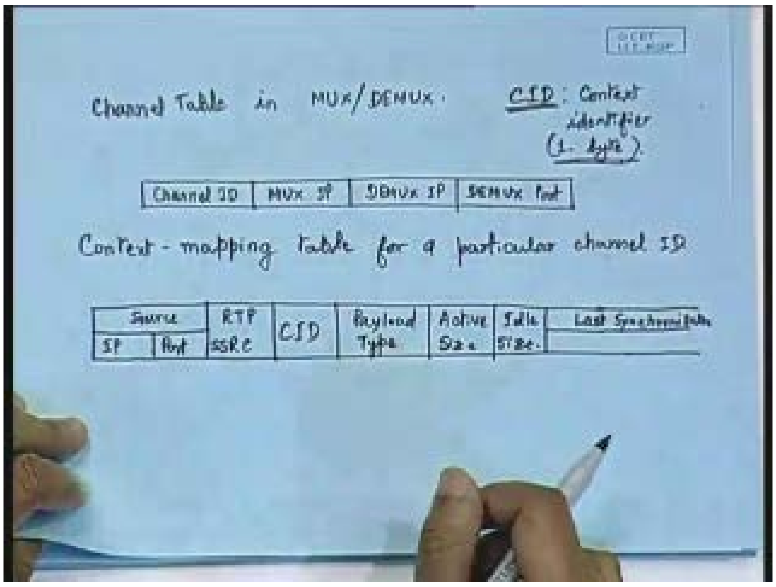
© CET I.I.T. KGP

H.323 Multiplexing : Header compression and Bandwidth efficiency.

Compressed miniheader.

And I was telling you that how with help of this C sequence and CID we can identify the channel.

(Refer Slide Time: 09:06)



© CET I.I.T. KGP

Channel Table in MUX/DEMUX.

CID : Context identifier (1- byte)

Channel ID	MUX IP	DEMUX IP	DEMUX Port

Context - mapping table for a particular channel ID

Source		RTP	CID	Payload	Active	Idle	Last Synchronization
IP	Port	SSRC		Type	Size	Size.	

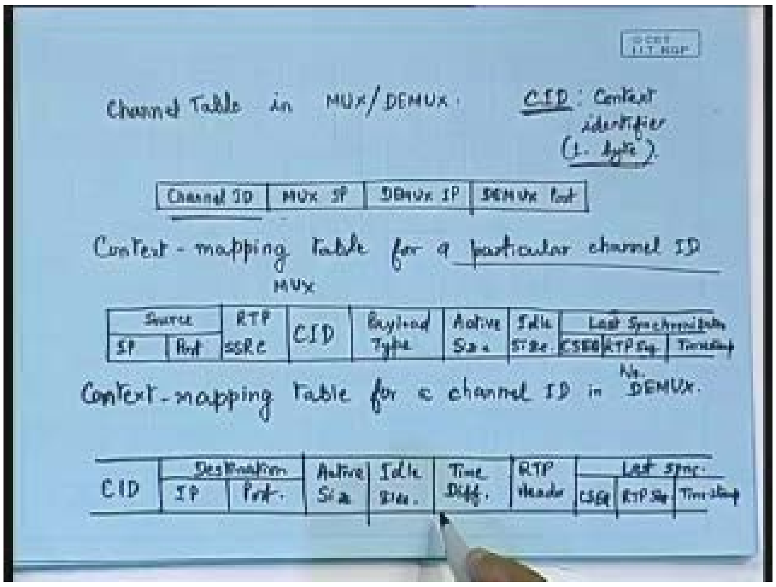
It is like this that it maintains; the multiplexer/demultiplexer system that maintains one channel table in the multiplexer/demultiplexer and the channel table is identified by this channel ID. So each multiplexer and demultiplexer combination; demultiplexer of course with the demultiplexer port number, so each multiplexed and demultiplxed combination would have a unique channel ID and for every channel ID there will be a context mapping table like this in the multiplexer. The multiplexer's context ID table for a particular channel ID will contain these things which i discussed last time the source IP and the port, then the RTP's own identification, then the context ID, the payload type etc. But what is important is that in the last synchronization means when the last synchronization miniheader was sent that time along with this whatever C sequence number was sent and whatever RTP synchronization number was sent RTP sequence number was sent and whatever was the time stamp all these information remain stored in the context mapping table for that particular channel ID. Now this is the context mapping table which gets stored in the multiplexer.

What the multiplexer does is that when the multiplexer receives at its input when the multiplexer receives a voice packet that time it identifies the source because it knows when the RTP packet comes from the source to the multiplexer the actual source to the multiplexer that time it identifies the source with its IP address and there will be for every session a unique CID context ID assigned. So, using the source number and the CID it is able to identify the stream and from that it will be able to know that for the last synchronization what was the C sequence number, RTP sequence number, time stamp etc because all that the multiplexer will do now after identifying the source and it also knows the destination because when it goes to this table using the channel ID it can find out that what is the demultiplexing IP and code so it knows that where it is to be sent and it knows and it can form the C sequence value. Because as it is, when it is receiving the packet, the voice packet from the actual source, it has got the RTP sequence number not the context sequence number. But using this table (Refer Slide Time: 12:10) it can form the context sequence and then the context sequence could be sent; context sequence and the context ID could then be dispatched to the demultiplexer to the proper port which it can find out from this channel table.

Now, once the demultiplexer receives it then the demultiplexer has to consult a very similar context mapping table which goes like this. So, context mapping table for DEMUX goes like this mapping table and this table is also for a particular channel ID for a channel ID in the DEMUX. So in DEMUX it has got these fields (Refer Slide Time: 13:03) we would not be concerned with all the fields actually; the CID this is the essential field which has to be there and then for the demultiplexer there has to be the destination IP and the port address, then the active size, then the idle size etc whatever are there in the multiplexer table that is there here also; there is a field called time difference.

Actually, because in the compressed miniheader the actual time stamp is not sent, so please note that this is one big change that whereas in the uncompressed miniheader and also in the synchronization miniheader it is essential to send the time stamp. But in the compressed miniheader we do not send the time stamp. That means to say that the demultiplexer has to derive the time stamp; how? It is by using this time difference information and the last synchronization information. So time difference, then the RTP header and then the last synchronization information and last synchronization information will contain as usual the sequence number, the context sequence, the RTP sequence and the last synchronization's time stamp. Therefore, using the last synchronization's time stamp and the time difference; ever since the last synchronization it should be able to compute the current time stamp. So, how it can do; it will be clear from a particular example. So as an example let us say that the context mapping table in the MUX contains the following information at a given time.

(Refer Slide Time: 14:46)



Channel Table in MUX/DEMUX:

CID: Context identifier (1- byte).

Channel ID	MUX IP	DEMUX IP	DEMUX Port

Context - mapping table for a particular channel ID MUX

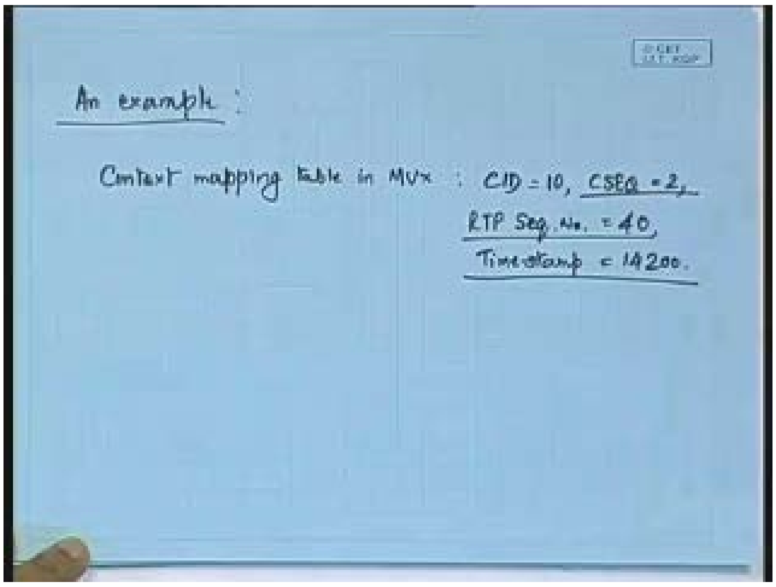
Source		RTP	CID	Payload Type	Active Size	Idle Size	Last Synchronization		
IP	Port	SSRC					CSEQ	RTP Seq. No.	Timestamp

Context - mapping table for a channel ID in DEMUX.

CID	Destination		Active Size	Idle Size	Time Diff.	RTP Header	Last Sync.		
	IP	Port.					CSEQ	RTP Seq	Time-stamp

Supposing as an example we take up a particular movement where the context mapping table in MUX contains the following information context mapping table in MUX; we are not bothered about all the fields but fields with which we are most concerned the CID this is equal to let us say 10, the context sequence number let us say it is 2, RTP sequence number let us say it is 40, time stamp..... this C sequence is 2 this C sequence mind you is the C sequence of the last synchronization, so C sequence, RTP synchronization and the time stamp, time stamp let us say is 14200 so this is also from the last synchronization.

(Refer Slide Time: 16:11)


An example:

Context mapping table in MUX : CID = 10, CSEQ = 2,
RTP Seq. No. = 40,
Timestamp = 14200.

So you see that we are referring to these information basically (Refer Slide Time: 16:16) last synchronization, C sequence, RTP sequence and time stamp its specific values are let us say 240 and 14200.

And we now look at the demultiplexer; context mapping in the demultiplexer and say that the context mapping table in DEMUX is like this; say it contains the CID value because CID has to be the same; since the session is the same the CID has to be the same, say the time difference is equal to 10 then the C sequence is equal to 2 and then the RTP sequence number this will remain the same that is equal to 40 time stamp is equal to 14200 so these are all the last sync information.

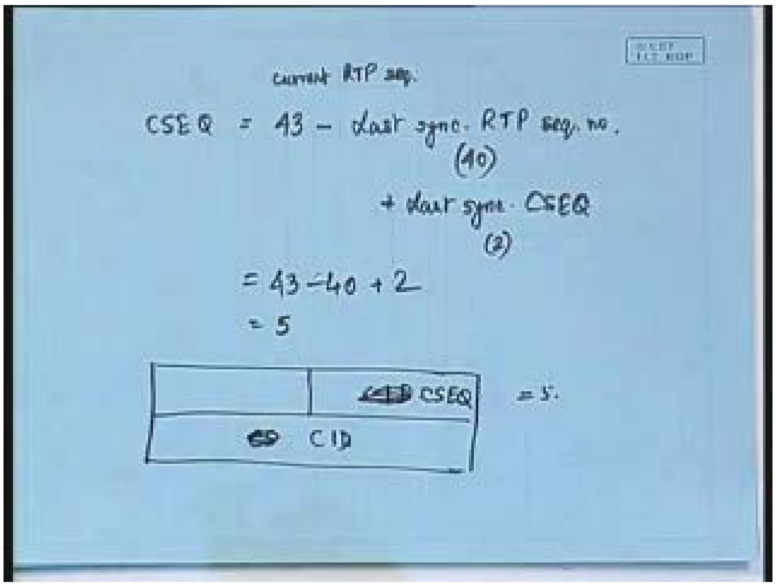
Now, supposing that at this that at this time the multiplexer receives an RTP packet; say the MUX receives an RTP packet an RTP packet now RTP packet has got an associated time stamp and the RTP sequence number. Mind you; it is receiving from the source it is receiving the RTP packet so it is containing the RTP sequence number and it is containing the time stamp RTP time stamp. But now the multiplexer has to when the multiplexer has to send this information to the demultiplexer that time it has to perform a header compression and in order to perform the header compression it has to derive that what the C sequence value is.

Therefore, in order to derive the C sequence value let us say that the particular RTP sequence number that the multiplexer has received is 43 and the time stamp it has obtained is 14230. now this RTP sequence number of 43 it has to now it cannot send this RTP sequence number but instead it has to send a C sequence to the stream. Now in the header, now in the compressed header, let us say that it is using a compressed header; the compressed header has to contain a C sequence value. Now what is the C sequence value that will be used over there?

so in order to Now the C sequence can be computed like this. the C sequence value can be computed as the RTP sequence number; the RTP sequence number in this case is 43 minus whatever RTP sequence number was there in the last sync, last sync RTP sequence number, this is the current RTP sequence, the current RTP sequence is 43 the last sync RTP sequence number which we know is how much 40 and plus the last sync's C sequence; last sync's C sequence was having a value of 2 that means to say that 2 corresponds, 2 in C sequence number corresponds to RTP sequence number 43 so 3 in sequence number contains RTP sequence number 41 like that so that when it receives 43 RTP sequence number it knows that the corresponding sequence number for this for corresponding C sequence number for this has to be 5 that is very clear; 43 minus the last sync RTP sequence number is 40 plus last sync C sequence is 2 so that makes it 5. So the C sequence number is 5. So now in the header information what it gives is this value of 5.

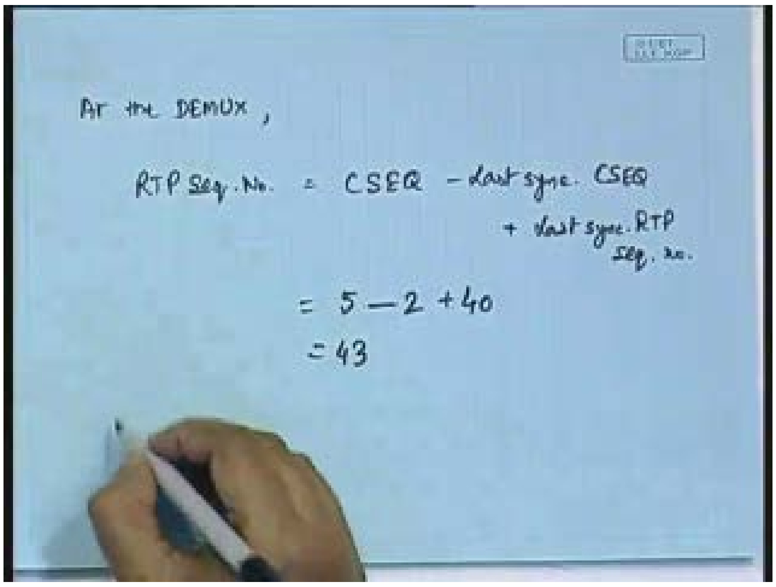
Hence now in the header.....................; it can now afford to give a 2 byte header; in the 2 byte header 4 bits will contain the CID and CID in this case is 10 and the C sequence no sorry sorry (Refer Slide Time: 21:08) C sequence value C sequence value is 4 bits so C sequence value is equal to 5 in this case and then the CID.

(Refer Slide Time: 21:18)



So now the demultiplexer receives this C sequence value, it does not know that what is the RTP header. So when demultiplexer receives a C sequence value equal to 5 it has to accordingly reconvert the RTP sequence number. So at the demultiplexer it can form the RTP sequence number easily, the RTP sequence number can now be obtained as the C sequence number that it has received minus the last sync's C sequence number which is already stored. Last sync's C sequence number is 2 plus the last sync’s RTP sequence number is also stored. So it is 5 received now, the last sync was 2 and the last sync RTP sequence was 40 so it gets back as the RTP sequence number 43 which was what the source was sending.

(Refer Slide Time: 22:26)



At the DEMUX,

$$\text{RTP Seq. No.} = \text{CSEQ} - \text{last sync. CSEQ} + \text{last sync. RTP Seq. no.}$$

$$= 5 - 2 + 40$$

$$= 43$$

Now about the time stamp; now it has to recover the actual time stamp. It is 14230 we know that is what the source had sent to the multiplexer but multiplexer did not disclose that information; multiplexer only says that your C sequence value is 5 but how the time stamp can be recovered? Well, you have the C sequence minus the last sync's C sequence last sync C sequence so the C sequence number's difference what is that; it is 5 now, it was 2 in the last sync, C sequence was 2 so there is a C sequence difference of 3 and now this needs to be multiplied by the corresponding time difference; the time difference in the sequence is 10. Now the time difference is 10 so it is 5 minus 2 into 3 that means to say 30 plus the last sync RTP sequence number is 14200 so it knows that the time stamp now should be 14230 so it recovers the time stamp.

Therefore, we are sending less information; the header compression definitely saves in lot of information but essentially the usage of this context tables in both the multiplexers and the demultiplexers essentially permits the multiplexer to derive the C sequence number and from the C sequence number it enables the demultiplexer to get back the original RTP information. So, that is the trick that is being done in this process of multiplexing.

Now one point which should be noted is that initially of course the demultiplexer will not be having these information. So for the initial so at the start of the session the first few packets they have to be sent in the fully uncompressed header type so the first few packets will have uncompressed miniheader. Now with that the demultiplexer will be able to find out that whether the fields in a particular session which are the constant fields and it will be able find out that which are the fields that are changing that are having constant implements like the time stamp and the sequence number. So accordingly then once those information are established from the next RTP packet onwards one can go through the one can then go through the RTP miniheader compression.

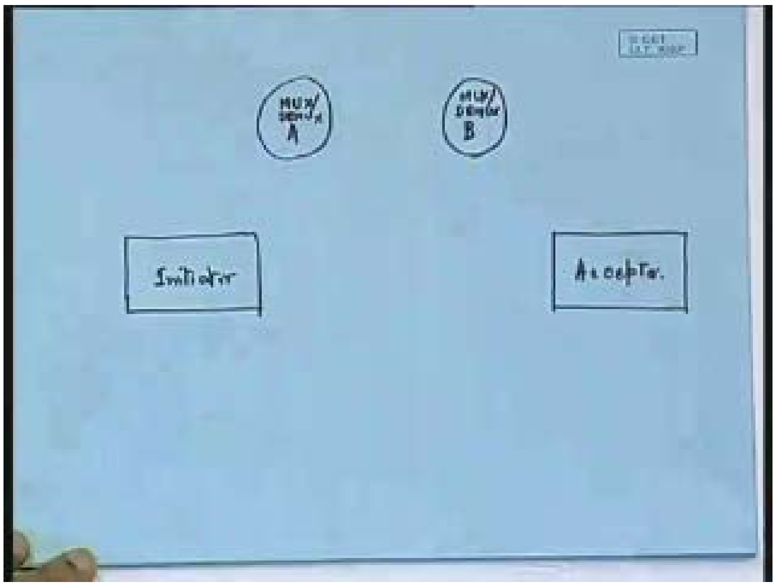
Miniheader compression you can do but periodically you have to transmit the synchronization miniheader; why, because you can see that C sequence number is only a 4-bit field that means to say that after 4 bits that means to say after 16 the sequence number again rolls over from 0 0 0 0 if you start up to 1 1 1 1 again it rolls back to 0 0 0 0. Since it is a 4-bit field that is why once in every 16 RTP packets there has to be the synchronization and they have to contain the synchronization miniheader.

And not only that, there are certain times there are like say for example when an audio channel starts from a silence that there is a silence followed by some voice chunk that is being sent, there the synchronization information tends to lose and that can be taken care by having the........... so that loss of synchronization can be corrected by giving forcibly the synchronization miniheader. So first few packets, uncompressed miniheader then the regular packets will contain the compressed miniheader and periodically or when needed the synchronization miniheaders have to be sent. So this is the overall scheme that one goes through for the header compression.

Naturally compressing these to 2 bytes; most of the times sending a 2 byte miniheader is really quite efficient and in fact the efficiency we can calculate when we come to the formal way of calculating the bandwidth efficiency in the multiplexed scheme so that we can indicate then. But before going to that let us see that what the call establishment protocol is. I mean, we know that what sort of call signaling takes place in H dot 323; already we are knowledgeable about that. But let us see that how it can happen in a typical multiplexer/demultiplexer type of a scenario.

So, for that, let us say that we have an initiator, we have the station which is the initiator and we have another station which acts as the acceptor and the initiator is connected to a multiplexer. Let us say that this is the multiplexer A (Refer Slide Time: 28:39). Now incidentally these multiplexers they also act as gatekeepers these are intelligent multiplexers so these are acting also as the gatekeeper. So, in the total connectivity we will be having the MUX A and we will be having the demultiplexer correspondingly which we call as B. thus, in fact both these things will be MUX/DEMUX; MUX/DEMUX both will be there because every such node will have the multiplexing and demultiplexing capability both. In this case of course since we are telling that this is the initiator and it is connected to A and this is the accepter and it is connected to B so in our case A will act as the multiplexer and B will act as the demultiplexer.

(Refer Slide Time: 29:32)

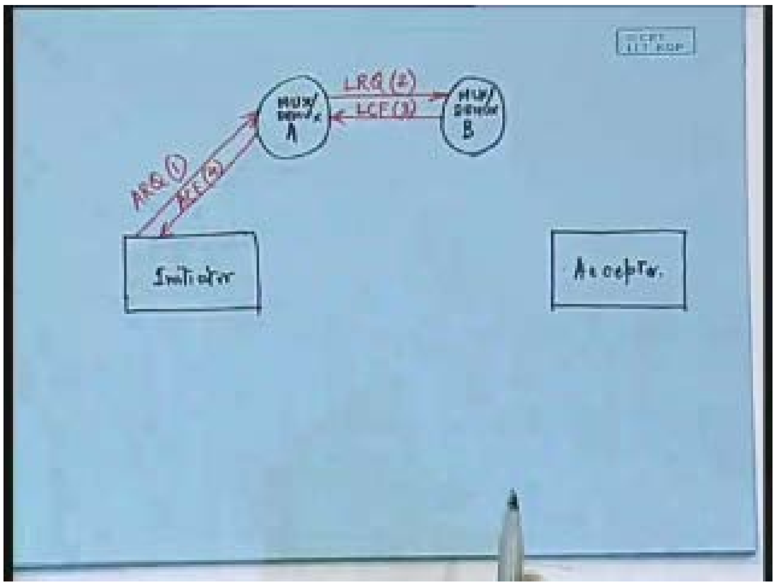


Now how the sequencing will be made; how the call setup will be made is like this. The initiator first sends an admission request admission request in short form we write as ARQ. This is the step number 1. So ARQ this is the first signal that will be sent from the initiator to the MUX A. Now the MUX A.............. I mean, the admission request is there but before having the admission confirmation the multiplexer has to find out or the multiplexer has to locate the accepter.

So what it does is that it broadcasts locates a request the locate accepter by sending a locate request LRQ signal and this will be the step 2. So the first step was initiator sending an admission request to the multiplexer and then multiplexer sending a location request to this demultiplexer. In fact it will be sent to all the demultiplexers to which it is connected. But the particular demultiplexer to which it is actually connected it will respond. Supposing in this case B is connected to the accepter so the B will respond by saying that there is a location confirmation so it sends a signal; B now sends a signal called as LCF which stands for Location Confirmation and this is the step number 3 so it comes from B to A.

Now the multiplexer knows that the accepter has been located. So once the accepter has been located and it has verified the authenticity of the initiator it sends an admission confirmation. So admission confirmation or ACF will be sent as the step number 4. So step number 4 the admission confirmation will be there.

(Refer Slide Time: 32:12)

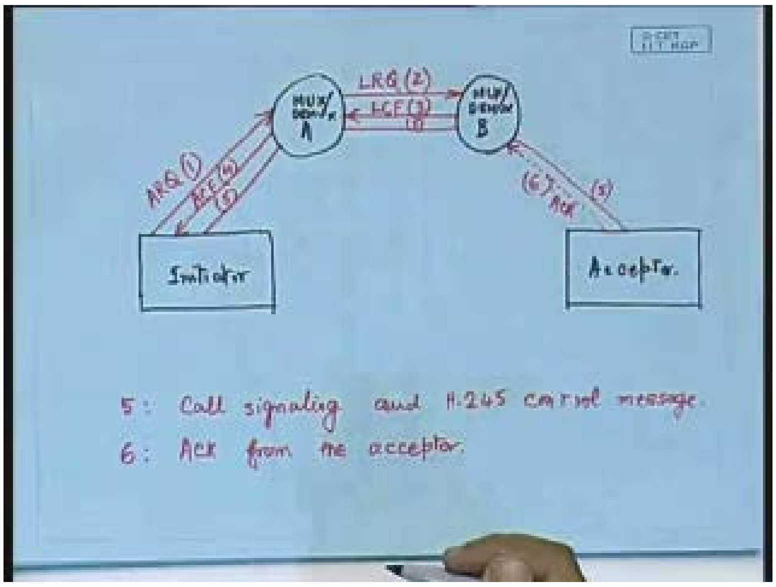


Once the admission confirmation is there then the endpoints can exchange the call signaling and the H dot 245 control messages; that is as per the H dot 323 we know that the endpoint to endpoints there has to be a call signaling and there has to be a control negotiation or the capability negotiation through the H dot 245 so that has to take place so that will be step number 5 so it is for the signaling and I am not giving any specific direction because actually it will be bidirectional because the signaling will be coming back and forth so this is step number 5 step number 5 where there will be............ so step number 5 is for endpoints exchanging, call signaling and H dot 245 control messages.

Once the accepter gets the call signaling accepter has to now respond and the accepter responds by saying by giving an acknowledgement signal. So this could be in another dedicated channel which goes from the accepter to the multiplexer. This is step number 6 (Refer Slide Time: 34:13) where it will be step number 6 will be the acknowledgement from the accepter. So step number 6 is ACK so it is acknowledgement from the accepter.

And once the acknowledgement comes from the accepter, now the multiplexer and demultiplexer will know that now a session has to start because; I mean, all the admission confirmation is there now the call signaling has been initiated and the accepter has also accepted that and acknowledged this so now it is ready. Now the voice packet has to begin and before the voice packet transmission begins what the multiplexer and the demultiplexer has to ensure that this session has got a unique CID.

(Refer Slide Time: 35:16)



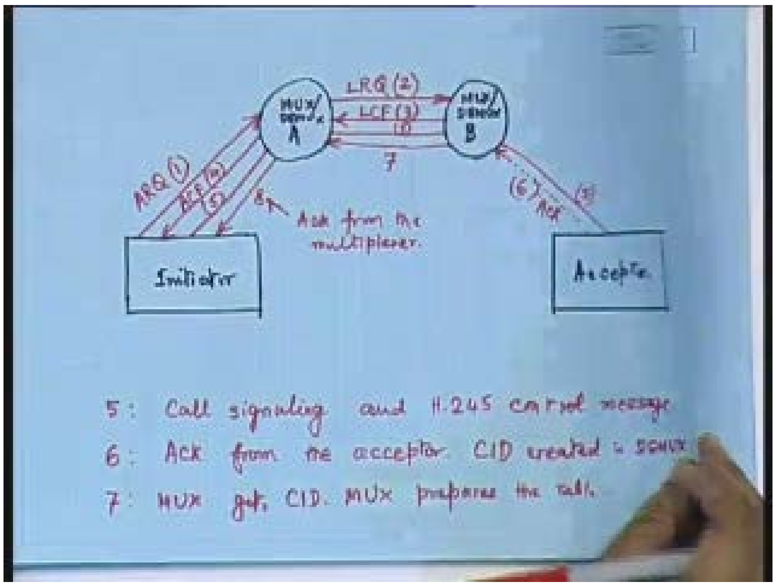
5: Call signaling and H.245 control message

6: Ack from the acceptor.

now when the In step number 6 it receives this acknowledgement then a unique CID the context ID will be assigned to this session that this demultiplexer will do and subsequently when this CID is assigned by the demultiplexer that CID has to be communicated to the multiplexer also. Therefore, now in step number 7 we will be having.......... the demultiplexer that the multiplexer now gets so the MUX gets the.......... so acknowledgement from the accepter, CID created in DEMUX CID created in DEMUX and not only the CID but it also creates the table the DEMUX table is prepared, now the MUX gets the CID value and MUX prepares the table context table.

Once that is done then the initiator receives an acknowledgement from the multiplexer. So, that is step number 8 so this is an acknowledgement from the multiplexer; the table has been created, CID has been assigned and now the actual voice packets can be sent. Then the actual voice packet transmission has to begin. This is the setup sequence that will be followed. Now the important question is that why we are doing this and exactly what kind of bandwidth efficiency are we going to achieve in this process. Hence, let us see some performance analysis with that. So we will now talk about the performance analysis.

(Refer Slide Time: 37:20)



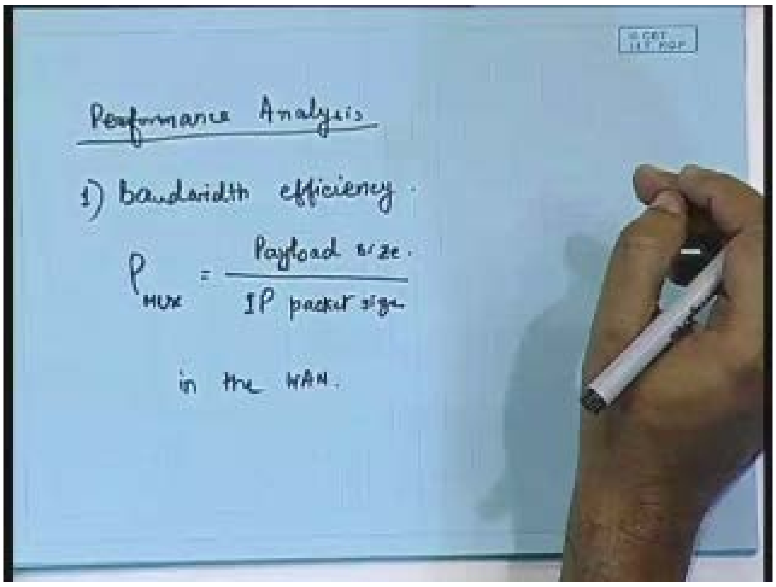
5: Call signaling and H.245 control message

6: Ack from the acceptor. CID created = DEMUX

7: MUX gets CID. MUX prepares the table.

Some quantitative measures are there for this and the first performance analysis is what we call as the bandwidth efficiency. To do that, we define the efficiency like this. We call this as rho (Refer Slide Time: 38:06) and we say rho with the multiplexer in the multiplexed scheme that what is the efficiency. The efficiency is measured as a ratio of the payload size. That means to say that in the WAN domain. In the WAN, means after the multiplexing is done because the multiplex stream will be sent in the WAN, so in the WAN the bandwidth efficiency or rho MUX will be measured as the payload size upon the actual IP packet size. IP packet size is again in the multiplex domain and payload size also in the multiplex domain.

(Refer Slide Time: 38:57)



Performance Analysis

1) bandwidth efficiency.

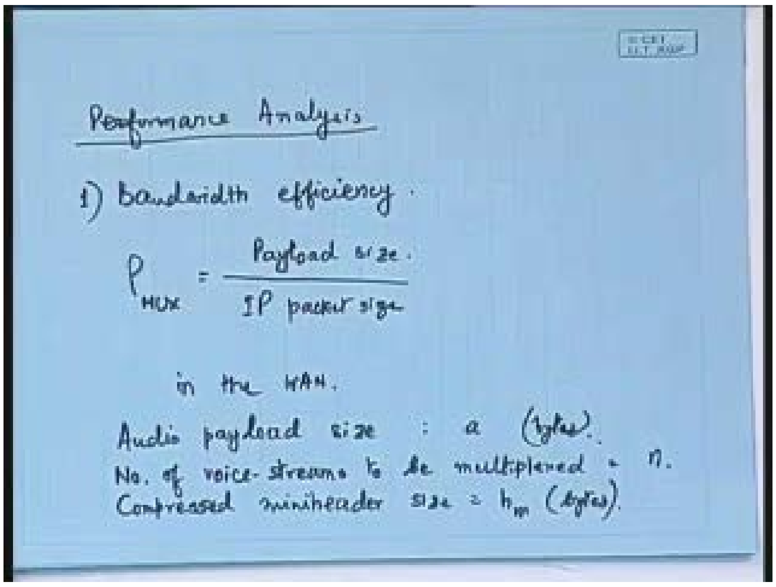
$$\rho_{MUX} = \frac{\text{Payload size.}}{\text{IP packet size}}$$

in the WAN.

Now, how to find out that? Because how to find out the payload size; so we need some more information for that. What is the information? That is to say that what is the audio payload size? Assuming that every channel or every MUX/DEMUX payer or every source/destination payer has a fixed audio payload size audio payload size and let us say that audio payload size is a so normally we will be expressing this a in the unit of bytes. Then say the number of voice streams that we want to multiplex number of voice streams to be multiplexed let us say that it is equal to n that is equal to n and now for every stream we have got a miniheader for every such n we have got a miniheader and the compressed miniheader size is let us say h m compressed miniheader size let us say that this is equal to h m in bytes.

Now you can argue that all the miniheaders will not be the compressed miniheaders but some of the miniheaders will be uncompressed. But that is at the beginning only and some of the miniheaders will be having the synchronization; that also we have to send periodically. But normally 99 percent of the miniheaders or 90 of the miniheaders will be of 2 bytes only so normally h m will be 2 bytes so that is the compressed miniheader size.

(Refer Slide Time: 41:09)



Performance Analysis

1) bandwidth efficiency.

$$\rho_{MUX} = \frac{\text{Payload size.}}{\text{IP packet size}}$$

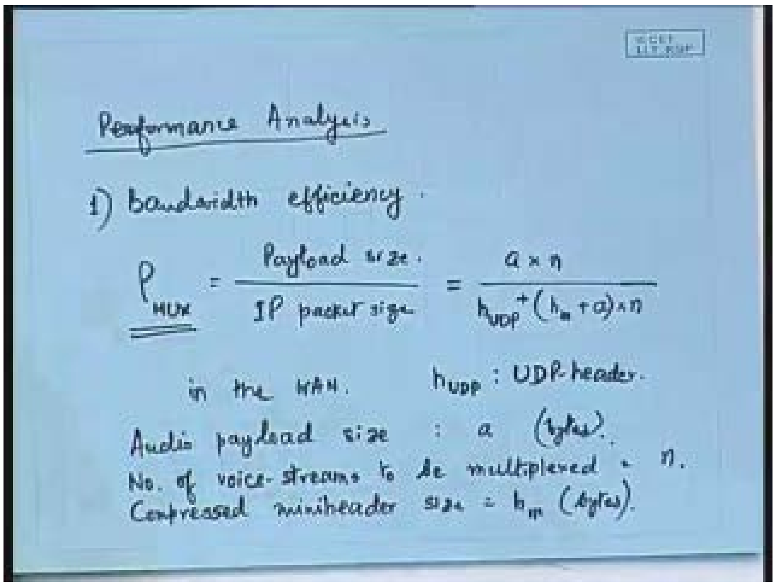
in the WAN.

Audio payload size : a (bytes).
No. of voice-streams to be multiplexed = n.
Compressed miniheader size = h_m (bytes).

Hence, now tell me that what is going to be the total payload size; in this rho mux expression what will be the total payload size given this information? a times n; because a is the audio payload size for every stream. For every stream the audio payload size is a and we have got n such multiplexed stream. So naturally this is a multiplied by n that is the payload size. And what is going to be the IP packet size?
h m; h m is the miniheader size, h m the miniheader has to be added........ miniheader will be there in every stream; h m plus a so h m plus a becomes the complete size of a particular stream and there will be n such streams. So h m plus a that multiplied by n will give us the miniheader plus the individual payload and n for n such channels.

But mind you we are forgetting one more thing that there is also an overall UDP header because this has to be this entire IP packet has to be sent over the UDP so there will be an overall UDP header which we call as the hUDP. So hUDP is the UDP header. So this will be our total bandwidth efficiency in the case of a multiplexed channel. Now this is with a multiplexer in H dot 323.

(Refer Slide Time: 43:08)



Performance Analysis

1) bandwidth efficiency.

$$\underline{\underline{\rho_{MUX}}} = \frac{\text{Payload size.}}{\text{IP packet size}} = \frac{a \times n}{h_{UDP} + (h_m + a) \times n}$$

in the WAN. h_{UDP} : UDP header.

Audio payload size : a (bytes).

No. of voice-streams to be multiplexed = n.

Compressed miniheader size = h_m (bytes).

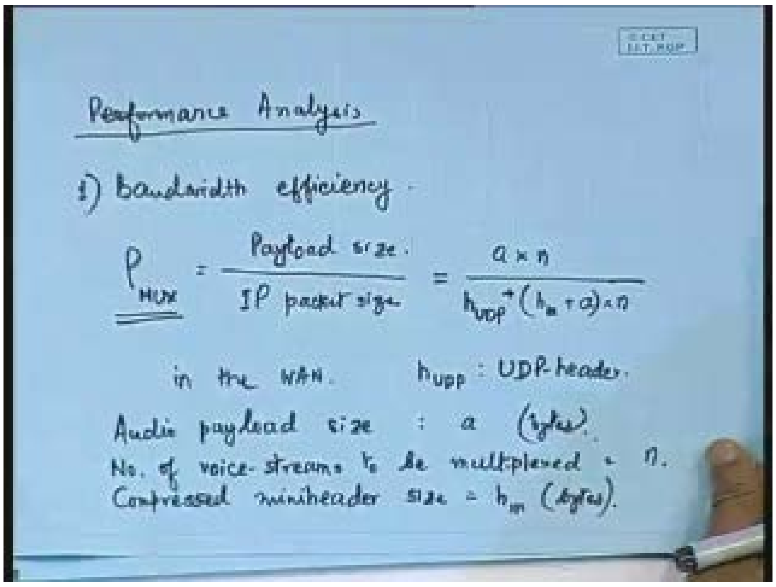
Now if we do not have any multiplexer that means to say that a non-multiplexed method or the old method if we use in that case the efficiency we can write as rho old. The rho old will be defined again as the payload size payload size divided by the IP packet size. When we are using the older methodology of non-multiplexing there the payload size is going to be a simply a because we are not having any channels and all the individual channels are sent individually so a, a is the payload size and what is the IP packet size; it will be the RTP header plus a so let us say that it is hRTP plus a where hRTP is the RTP/UDP/IP header size the RTP/UDP/IP header size. But mind you this is very significant. it is 40 bytes of information no jokes; it is 40 bytes of information which is there in this RTP/UDP/IP header.

Now let us see that what could be a?

If a is very large then no problem. But then let us say that we are using the state-of-the-art codec; the state-of-the-art speech codec is G dot 729 and G dot 729's bit rate is only 8 kilobits per second. This 8 kilobits per second then considering the delays that we require to form a packet the packetization delay will be very severe if we try to make our a very large. Therefore, given that........... actually the audio frame size in G dot 729 the audio frame size is equal to 10 bytes and the payload is made equal to one audio frame size. So a is equal to 10 bytes so that will really give us a very bad figure because with this figure the rho old is going to be only 10 a is 10 and the RTP header is 40 so 40 plus 10 so 10 by 50 that means to say 0.2

so only 0.2 efficiency 20 percent of efficiency is achieved if we are using the unmultiplexed thing.

(Refer Slide Time: 46:17)



Performance Analysis

1) bandwidth efficiency.

$$\rho_{MUX} = \frac{\text{Payload size.}}{\text{IP packet size}} = \frac{a \times n}{h_{UDP} + (h_m + a) \times n}$$

in the WAN. h_{UDP} : UDP header.

Audio payload size : a (bytes).

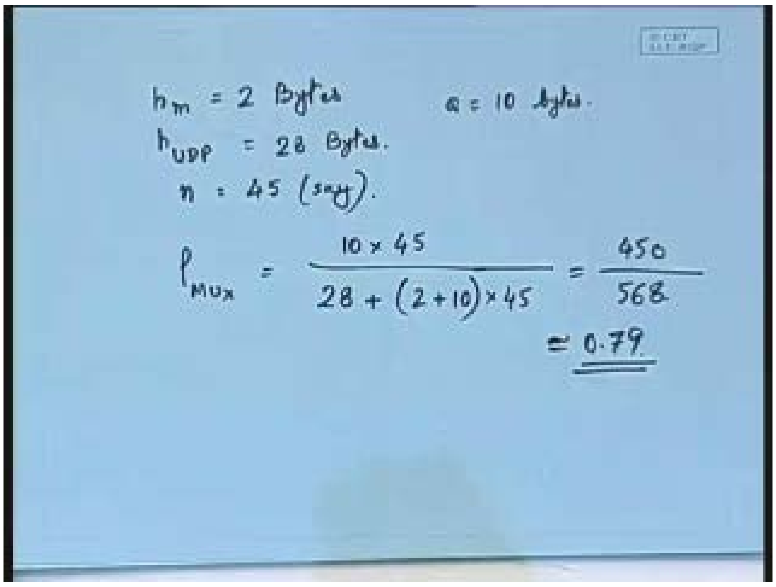
No. of voice-streams to be multiplexed = n.

Compressed miniheader size = h_m (bytes).

But let us say that if we use the multiplexed scheme the expression that we had formed here (Refer Slide Time: 46:18) let us put some typical figures for that. Let us say that h m h m in this case is 2 bytes; normally taking the compressed header size h m is 2 bytes the miniheader size, then hUDP hUDP is pretty large hUDP is 28 bytes 28 bytes and supposing we have n equal to 45 say we have 45 mutilplexed channels I think I specified everything else our a can be taken as 10 bytes as we have taken before so a is 10, n is 45, hUDP is 28, h m is 2 so now let us calculate what is the rho MUX.

The rho MUX will equal to a into n so 10 into 45 this divided by hUDP is 28 plus h m means 2 plus a is 10 and then we have got again n as 45. So it is 450 divided by 45 into 12 means it is 540 568 540 68. So, that makes it something like 0.7 or near about 0.8 roughly 0.79 yeah 0.79; so 0.79 is the efficiency the bandwidth efficiency. So what it was; it was only 0.2 in the unmultiplexed scheme and it goes up to 0.79 that is nearly 80 percent of the bandwidth is efficiently utilized in this process.

(Refer Slide Time: 48:33)



$h_m = 2$ Bytes $\quad a = 10$ bytes.

$h_{UDP} = 28$ Bytes.

$n = 45$ (say).

$$\rho_{MUX} = \frac{10 \times 45}{28 + (2+10) \times 45} = \frac{450}{568}$$

$$\simeq \underline{\underline{0.79}}$$

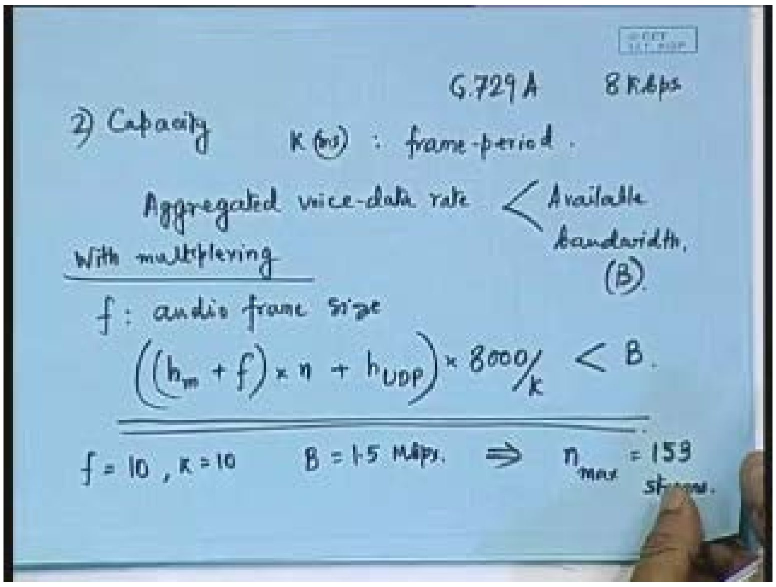
Now let us talk about the second performance analysis and that is the capacity. So the second performance analysis is the capacity. now in the capacity basically the number of concurrent schemes that number of concurrent streams that will be possible that will be given by this constraint is it not that the aggregated voice data rate the aggregated voice data rate that should be always less than the available bandwidth otherwise you will have packet loss. So aggregated voice data rate has to be less than the available bandwidth and let us say that the available bandwidth is B; the available WAN bandwidth we mean to say is B let us say, and how to compute the aggregated voice data rate.

Let us say that f is the audio frame size audio frame size and we have the miniheader as h m so h m plus f; f is same as what we were calling as a no problem therefore, h m plus f into n where n is the number of simultaneous streams plus the header of UDP. So this is the overall payload size that we will be having for n multiplexed streams and this is the payload size in bytes (Refer Slide Time: 50:28).

But we now have to convert this into the bandwidth. So bandwidth will be actually measured in some bits per second unit. In order to do that we have to multiply this by the data rate. So if we calculate so if we assume the G dot 729 A codec having the 8 kbps of rate then there we will be having 8000 that is this multiplied by 8000 and supposing K is the frame period so K milliseconds K in milliseconds is the frame period so now we can have this into 8000 by K is less than B. therefore, this is the constraint (Refer Slide Time: 51:25).

In fact, if you take a typical figure of f is equal to 10 10 bytes you take K the frame period to be equal to 10 and if you take the B the bandwidth of the WAN link to be 1.5 mbps then you can calculate and find out that the maximum number of n that we can achieve in this case is n maximum could be up to 153 streams whereas if we do not........................ so this is with multiplexing. So 153 streams we can have with multiplexing.

(Refer Slide Time: 52:25)



G.729 A 8 Kbps

2) Capacity K (ms) : frame-period.

Aggregated voice-data rate < Available bandwidth, (B).

With multiplexing

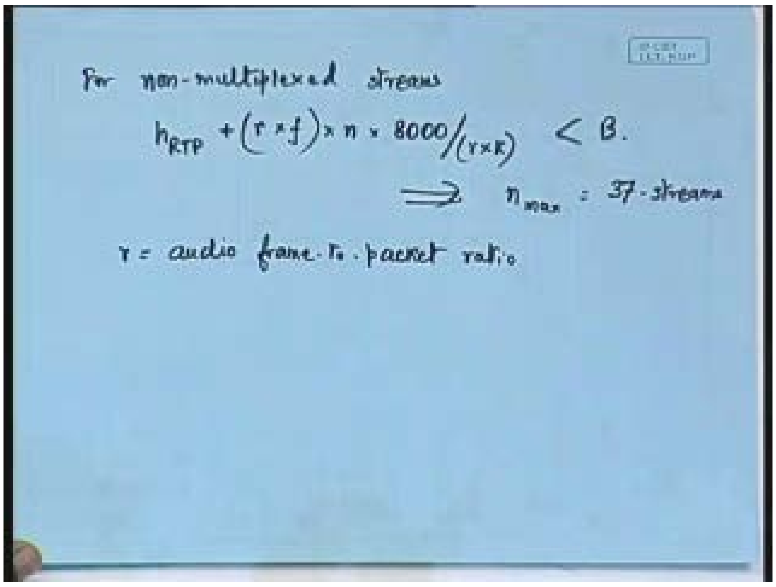
f : audio frame size

$$\left(\left(h_m + f\right) \times n + h_{UDP}\right) \times 8000/k < B.$$

$f = 10$, $K = 10$ $B = 1.5$ Mbps. $\Rightarrow$ $n_{max} = 159$ streams.

And if we do not go through the multiplexing scheme that means to say that in the old scheme, there, our rate would be slightly different. What it means to say is that the hRTP so there the RTP header size plus we have let us say that r is the audio frame to packet ratio r into f because in this case we cannot have the audio frame to packet ratio to be equal to unity so r is the audio frame to packet ratio because the packet has to be with smaller size, the packet cannot be one frame size; only in multiplexed scheme we can have the packet equal to one frame size but normally the packet is less than a frame size. So audio frame to packet ratio typically this figure will be less than unity so RTP hRTP the header plus r into f that times n number of channels and the bit rate 8 kbps this divided by again r into k this quantity will be less than B so this will for this will be for non-multiplexed streams and you can see that using this figure we can have n max to be equal to 37 streams only.

(Refer Slide Time: 54:12)



For non-multiplexed streams

$$h_{RTP} + (r \times f) \times n \times 8000/(r \times k) < B.$$

$$\Rightarrow n_{max} : 37\text{-streams}$$

r = audio frame-to-packet ratio

So, up to 37 streams can be accommodated in the non-multiplexed manner and using the multiplexer scheme 153 which is more than four times more than four times of the channels can be accommodated more than four times of the streams can be accommodated using the H dot 323 with the multiplexed scheme. This is what I had to say about the H dot 323.

Now H dot 323 is a standard; definitely it is a very useful standard that has been proposed by the ITUT. In fact, it is getting revised further and further and very recently even in June 2006 the last revision of H dot 323 standard was made and H dot 323 although I have described it in the context of the voice over IP but you will be surprise to know that H dot 323 can also support the video over the IP network.

In fact, in the initial days of the video conferencing, because of the very high bandwidth requirement of the video and also the questions of the connection-oriented protocols that should be used for the video there the ISDN video conferencing was the first generation of the video conferencing which was there. But more and more the ISDN video conferencing are getting replaced by video conferencing over IP where video/voice both has to be through the IP layer and there the H dot 323 becomes a very efficient and powerful standard.

Therefore, for video conferencing H dot 323 is there, the SIP is there where SIP is the Session Initiation Protocol plus ISDN video conferencing is also there. So in the next class onwards we will start with the video conferencing and we will see the ISDN video conferencing and the SIP protocol because we are already familiar with the H dot 323, thank you.