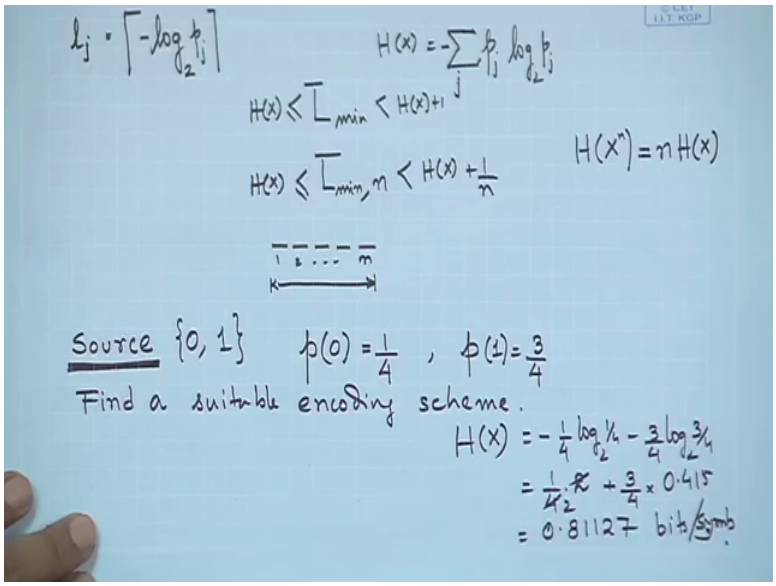
Modern Digital Communication Techniques
Prof. Suvra Sekhar Das
G. S. Sanyal School of Telecommunication
Indian Institute of Technology, Kharagpur

Lecture - 13
Source Coding (Contd.)

Welcome to the lectures on Modern Digital Communication Techniques, we have been discussing Source Coding. And what we have covered till now is a unique decidability, prefix free codes, Kraft's inequality, and finally we have come to understand the mechanism by which one could encode a discrete source. Of course, we are talking about discrete sources.

So, while doing it we have come across several ways of doing the compression in some way, or we finding out that what is the average number of bits per source symbol that we need to send. So, when we did it we wanted to also find out that this average number of bits per source symbol that we are getting what is the minimum value of it. And when we did it we use the Kraft inequality because we have to set the prefix free condition. And the reason for doing so is we wanted to have the unique decodability criteria. So, when doing it we use the Lagrangian method and we relaxed the integer length constraint.

(Refer Slide Time: 01:28)



And when we did it we found that you could choose lengths l for a particular symbol j to be equal to the ceiling function of log base 2 minus sign of p j; where p j is the probability of occurrence of the j-th symbol.

So, with this we have shown that if you select symbols in this way then the number of bits that you get is indeed very close to the entropy, where we define entropy as the measure of the minimum number of bits that is required to encrypt the source. So, if this runs over g then p of j log base 2 p of j the minus sign in front of it. And this gives a measure of what is the average on an average minimum number of bits required. And when we did this particular method we found that these are integer numbers.

So, only when this p j was equal to sum integer power of 2 we realized that this could be the actual minimum, and when it is not then it is more than the entropy, but it is at most 1 bit more than the entropy. And these bounds on entropy we had established in the previous discussion. So, we said that L bar min is less than H x plus 1 and is greater than or equal to H x. So, this is also something which we are established.

And we took one particular example in quick passing that this may not the method of finding or the using this particular expression is not always going to give the best result. And we will look at the counter example where we said let the probability of 0 b to the power of sorry 2 to the power of minus 10; and we will see another example today.

So, further we said that if these you what you need to look at is rather encode n-tuples. We had seen this thing for fixed length code, and we had also shown that if you have variable length code then also you can come arbitrary close to H x. So, at that point we said that L min bar when taken n-tuples at a time is very close so you get tighter bound 1 upon n and H x. That means, by choosing n very large you can make it very close to H x this is all that it says.

So, if you are going to take n-tuples at a time. So, when we did n-tuples what we meant is that if you take a few symbols 1, 2 up to n symbols and encode then in that case on an average you are going to get this particular situation. So, this is a fixed length code, because from fixed length words you are mapping to variable length and then only your establishing this, because we used for establishing this we used H of X n. That means, the entropy of a discrete memory less source for n-tuples is n times H x. So, this is what we had used and we arrived at this result.

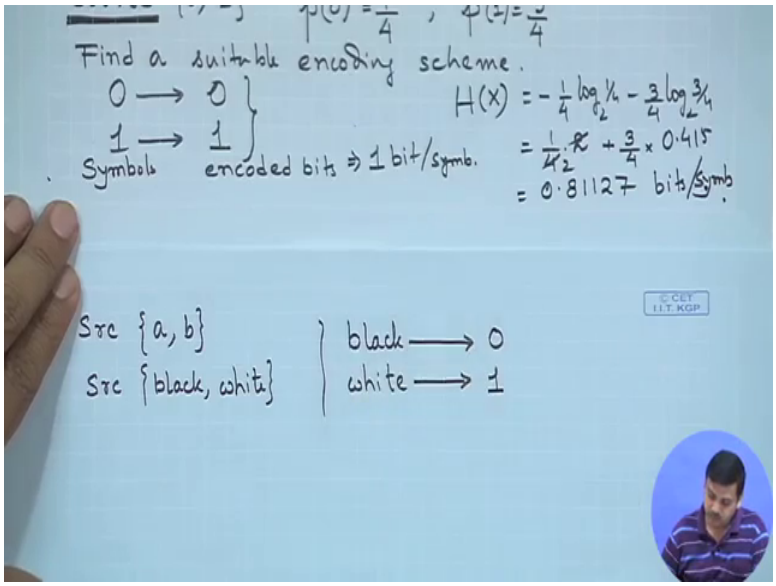
So, we move on further with this basic background and let us see that what can we get out of this. So, with these let us start off by considering a source which generates a 0 or 1. So, this particular source is a binary source. If it generates 0 and 1 then we need to characterize the source, and as mentioned before that the source characteristics can be described by means of the probability of occurrence of one of the symbols. And we say for our particular example let it be 1 upon 4.

Since there are only two symbols definitely this p of the second one is 1 minus the probability of that. So, p of 1 the probability of getting a one is 3 upon 4. So, for this particular source what we would like to find, is find a suitable encoding scheme. Now this might seem to be a very simple task. So, before we proceed to find a suitable encoding scheme what would like to is try to find out what is the entropy. So, if you are interested in finding the entropy you would calculate H x, going by this expression.

So, that will be minus 1 upon 4 log base 2 1 upon 4, so it goes up minus 3 upon 4 log base 2 3 upon 4. Of course, this minus comes in 4 by 3 minus 4, so this goes 4. So, this part is very easy; it is 1 by 4 times minus, so 4 is goes on top log base 2 of 4. So, that is 2 to the power of 2; 2 comes outside log base 2 of 2 is 1. So, you have 2 minus sorry there is a plus let us say you cnnot help it here you have to look into the log table. So, we are going to get 0.415 and this is actually half. So, if we evaluate this is almost 50 percent of 0.34.

So, what you get is nearly0.81127; as an answer that what you would get. And this is definitely bits per symbol. So, this is what we can start off with.

(Refer Slide Time: 07:58)



So, if we take this the next simple thing that we have, we do not have much option in this. We have a 0, we can encode 0 with the 0 or 1. And if you have 1 you have no other option, but to encode it as 1 or 0. So, whether this is 0 or 1 and this is 1 or 0 it is a same story, so we have this.

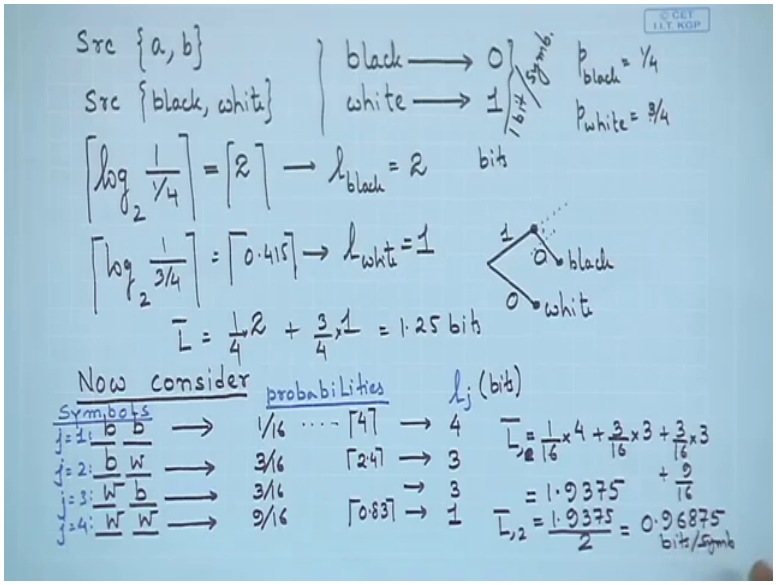
So, if you take this source coding that is your basically doing nothing you are allowing the source to pass through almost as it is or is one of the simplest possible source coding 0 becomes 0 and 1 becomes a 1. So, there is no change. So, these are the symbols and these are the encoded bits. So, what we have one symbol mapping to 1 bit, one symbol mapping to 1 bit. Therefore, we can say that you have 1 bit per symbol.

So, if this example is a bit confusing because you have 0 mapping to 0, we have a simple thing let us talk about a source where there are only two possibilities a and b.Another sample source could be a black and white. So, we can take off screen or a page which has only black and white dots. And a black would be represented by let us say 0 and a white could be represented by 1. If that is the case then all these things become very clear- that we have a black which could be map to 0. So, this would be equivalent to this 0 and this 1 would be equivalent to this 1.

So, in this case you can similarly say that- the black gets mapped to 0 and the white get mapped to 1; that is it. So, what we are asking is what kind of coding you can do. So, this is the coding which you did. And you can get 1 bit per source symbol. Let us proceed a

bit further with this example. And if we have to take this solution that we have mentioned earlier, if we have to take the solution then let us see what do we get.

(Refer Slide Time: 10:28)



So, if we do log base 2 1 by 1 upon 4, because you have a minus hear that means log base 2 of 1 by p j using which we can write. So, we are using that simply that expression.

So, this is very very clear. So, this is equal to 2, and if we have to use the ceiling function of 2 is basically 2. That means, l for let us say black is basically 2. So, this means 2 bits it says this says you use 2 bits which represents black. And if I have to represent white which comes with the probability of 3 by 4; that is the same example that we have. So, of course, what we said is p of black is 1 by 4 and probability of white is 3 by 4. So, log base 2 of 1 upon 3 by 4 and you take this ceiling function that is the ceiling of 0.415. Now since I cannot take 0.415 bits I have to take the nearest integer, so length of white should be equal to 1. As for the solution of this season the method to find the minimum number of bits.

So, going by this if we have to construct a code tree we can constructor a code tree one branch and of course the second one here. So, this one I could label as white, this one I could label as black, this is 0 1 0. Now, going by or earlier discussion; since you can clearly see that one branch can be added it is not a full tree and also this branch could be reduced and brought over here. Still it does not violate the prefix tree condition. That means there is although this appears to be an earlier intermediately node, but we can

shorten the tree and then the code becomes 1 for black, 0 for white or vice versa. We could have drawn the tree otherwise. So, this is superfluous we do not need this we can actually do better by simplistic into our original encoding scheme this one or simply this encoding scheme.

So, well this really does not help us at this point. And in fact if we try to calculate the average; so if try to use this method let us say what we are going to get is 2 bits coming with the probability of 1 by 4 plus 1 bit coming with the probability of 3 by 4 and this will be 1.25 bits, which is definitely worse that this which is 1 bit per symbol.

Now, we clearly see that L bar, this L bar this is not a good solution that we have arrived at. So, instead what we can do is now we can consider; let us use a different technique and we say that if you remember what you studied earlier that you can actually take n-tuples. So, if we take n very very large that of course we can reduce this excess. So, what we do as a first step: we can take two symbols. So, this black we can take.

Basically if I take two symbols the first place can be filled in two different ways black and white, second place can be filled in two different ways black and white, so we basically end up with four possible options. So, we could have a black and a black, a black and white, a white and a black, and a white and a white. So, this forms my new symbol super symbol black black, or black white, or white black, or white white.

See if we take these four symbols then we can encode this new source. So as if this source generates a symbol which is black black, it generates another symbol could be white black, or black white, or could be white and white. So, these are the four distinct symbols that it generates.

So, since this source is a discrete memory low less source that we have taken; that means, the symbol is independent of the first symbol, so the joint probability of this would be product of the individual probabilities. So, this probably of occurrence of this symbol is simply the probability of occurrence of black times the probability of occurrence of black. So, this is 1 by 4 times 1 by 4, so this is 1 upon 16. So, we could write these as symbols and we could write this as probabilities. So, by doing so we can calculate the probability of getting a black followed by white which is 1 by 4 of black times 3 by 4 of white. So, this is 3 upon 16 by the same logic this one is also 3 upon 16, and getting white and white is three times 16 times sorry 3 upon 4 that is 9 upon 16.

So, these are the probabilities of each of the symbols. And if we have to calculate the lengths that are l j; so j is equal to 1 in this case, j is equal to 2, j is equal to 3, j is equal to 4. So, we have these four symbols that are there with us. So, l j's following this: it will be 4 bits. And this particular case, what this result is straight forward if you evaluate this you going to get 4 a ceiling function of 4 leads to 4. If it is 3 by 16, so basically log base 2 of 3 by 16 or log base 2 of 16 by 3 this is ceiling function of 2.4. This will be landing up definitely with 3 bits, because 2.4 bits you cannot. So, same over here 3 bits 9 upon 16, this turns out to be 0.83, this results in 1 bit.

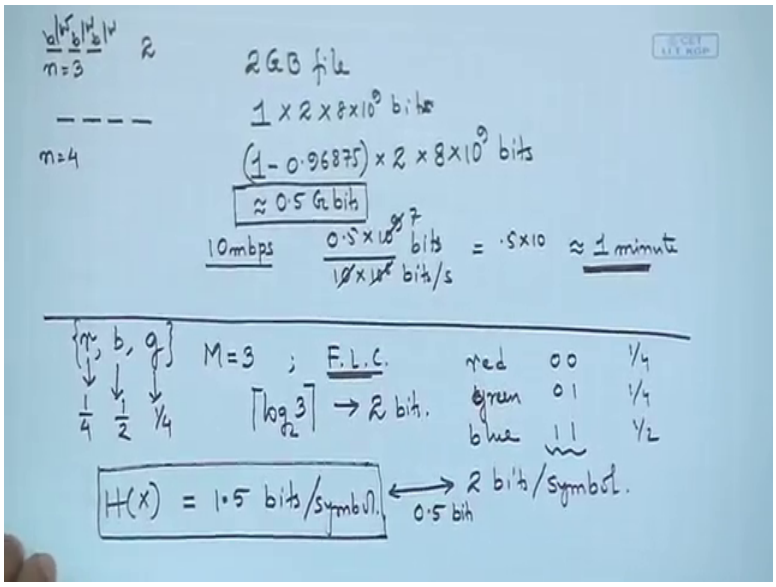
So, what we have in this case we are trying to encode this new symbol set with these lengths. So, if you are going to do it, of course we can find out whether it satisfies the Kraft inequality. But what we have seen earlier that if we follow this method, it is definitely going to follow Kraft inequality. That means, we can ensure prefix tree quotes. So, that is why we looked at the theorem and therefore we do not need to do it again, but for your own sake you can easily calculate.

So, we can do this, and therefore we can construct tree and we can make codes, but what is our interest as of now is to find what is L bar. So, L bar is the average number of bits per source symbol. So, the way we calculate is we are going to get 4 bits whenever we becomes, but b b comes with the probability of 1 upon 16 and every time it comes we going to get 4 bit. 3 bits are going to occur with the probability of 3 by 16, because this is going to come with the probability of 3 by 16 every time we get 3 bits, we get it 2 times. And this one is going to occur with the probability of 9 by 16. So, we going to add up only ones.

So, when you add this you will get the answer of 1.9375. Now if you look at L bar, L bar is with 2 bits. Please remember this with 2 bits. So, what is L bar per symbol? So, L bar per symbol would be L bar of this. So, I would rather say L bar 2, I would not put a two over L bar 2 would be this divided by 2. And this turns out to be 0.96875 So, what you can see is earlier we were having 1.25 bits if we followed this rho expression by taking one symbol at a time, and which is worse than the case if I would directly encoded which is 1 bit per symbol which is also represented here. But now, if I have clubbed two symbols that means I have formed an n-tuple where n-tuple is 2 I could reduce the average number of bits per symbol I could reduce that average number of bits per source symbol 2.96875; and this is definitely less than 1.

So, as we move further of course we just need to remind you that we had found the entropy of the system to the 0.81127. That means, we have been come a little bit closer to this number.

(Refer Slide Time: 21:35)



So, moving ahead further with this you could ask what would be the case if I take three symbols as a tuple. That means, if I let n is equal to 3. So if n is equal to 3, since each position can take two values; that mean each position can take a black or a white: two possibilities. So, definitely you are going to end up with eight possibilities and then you can do a similar calculations.

And what we can do now instead of going for 3 let us take the case if I am going to take n is equal to 4. If I take n is equal to 4 of course I have 2 2 2 2. So, 2 to the power of 4 possibilities and that would be. So, if I have there would be two possibilities here two possibilities, two possibilities, two possibility. So, I am going to get a much a smaller number in this case. So, as we move ahead further you can possibly get this numbers which are smaller than this.

So, with this if you go ahead and considered a particular example where we say that suppose I have a 2 gigabyte file to transfer. Suppose I have a 2 gigabyte file to transfer and I use this particular method of encoding as we have just disused. So, what is there any benefit? So, what you can calculate is that if you do not do this you would send 1 bit per source symbol multiplied by 2 multiplied by 8 multiplied by 10 to the power of 9 bits

all together. Whereas, if you are using this encoding scheme as we have just discussed here, if you are going to use this encoding scheme then we would be saving 1 minus 0.96875 bits. So, this much of savings is going to happen in this overall file size.

So, if you do this, if you work out these calculations you will almost get 0.5 gigabits. Please note I used gigabytes so over here but I converting to gigabits. And if we do the calculation that say- we have 10 mbps line; suppose I have a 10 mbps line so I would require 0.5 into 10 to the power of 9 bits divided by 10 into 10 to the power of 6 bits per second lesser time if I use this particular method. So, this would result in 0.5 into 10 to the power of. So, this, and this, and this, would cancel out to make it 100.

So, I am going to get nearly 50 seconds or roughly 1 minute less time to download this particular file. And which is significant. And if you look at this particular number then you can guess the number of bits that get saved. So, if you are thinking in terms of the subscribers, so one generally has to pay in terms of number of bits; that means finally you going for a package of like let us say 500 megabyte or let us say 2 gigabyte for so much of a sum of money. Let us say 100 rupees for 1 gigabytes something like that.

So, in this case you are saving quite a bit of data if you are doing some data compression. And if you are thinking in terms of speed or download time if this kind of data compression is done then again you can save quite a lot in terms of a bandwidth. That means, you can free up the channel for at least 1 minute in this particular example that we have taken.

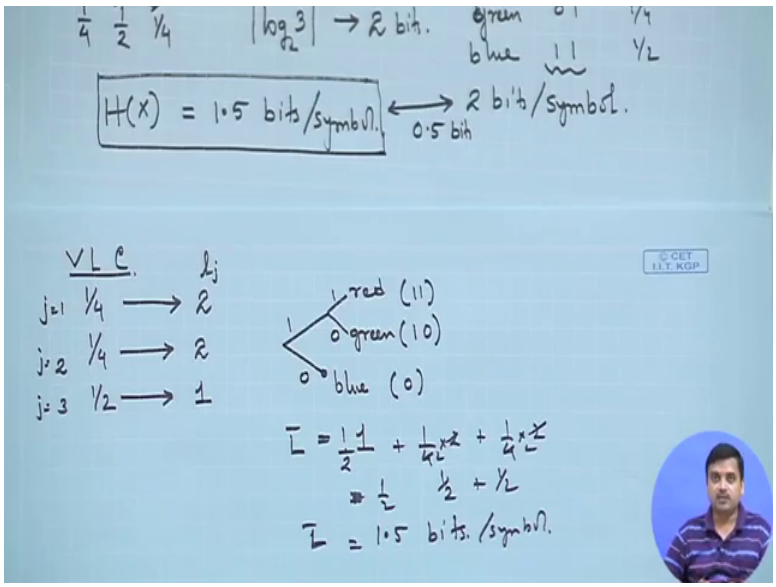
So, I hope this helps you to realize that if you do this kind of an encoding and of course many other advanced encoding schemes then you could have a huge potential in utilizing the spectrum or the bandwidth much better than if you do not do any kind of source coding. So, let us move on further and try to look at another situation.

So, where let us say we have the problem that we have three colors: red blue and green. And we say that red comes with the probability of 1 by 4, blue comes with the probability of half, and green comes with the probability of 1 by 4. That means, let us say I have a screen at or I have an image where every pixel in that can take one of these three colors and only one of these three colors and I would like to encode this picture in this pixel and send it out to the other side.

So, clearly there are three symbols. So, going by our previous notation m is equal to 3. And if you have to use some fixed length code then you have to take log base 2 of 3 in a ceiling function of that which would result 2 bits. So, clearly you can say that I would encode read with is 0 0, a blue or a green let us say green with the 0 1, and a blue with 1 1. Red comes with probability 1 by 4, green comes with probability 1 by 4, and blue comes with probability half. In this case we clearly see that you have 2 bits for every symbol: 2 bits for blue, 2 bits for green, and 2 bits for red- so 2 bits per symbol.

For this particular case if you would calculate H x this I leave it as a home exercise you are going to get the number as 1.5 bits per symbol. So, this clearly leaves us with a gap of 0.5 bits. Now if it is 0.5 bits per source symbol seen clearly estimate that the amount of saving that you can get if there is a 2 gigabyte file that requires to be transmitted. Now H x please note this gives us an indication that what is the best possible source coding that you can achieve with this.

(Refer Slide Time: 29:00)



So, well if you use fixed length you are in this situation and what you could do that you could go for variable length coding; if you go for variable length coding since now you have this probabilities. So, this would result in the code word length of 2 bits, this would result in a good word length of 2 bits, this would results in the code word length of 1 bit. So, I do not need to define what is j any further. And if you would like to construct the code tree you can easily construct a code tree like this where this will be blue, this will

be green, and this will be red; so 1 1 0 0. So, blue will have the code of 0, green is going to have the code of 1 0, red is going to have the code of 1 1.

So, if you do this then you will find that; if you calculate L bar for this what you going to get is blue comes with a probability of half. That means, this code word length comes to the property of half. So, are going to get 1 occurring 50 percent of the time plus a code word length of 1 occurring one-fourth of the time, so that is half plus half plus half. So, that is 1.5 bits. So, what you can see is that by using this mechanism you can achieve 1.5 bits per source symbol and one can easily go back and calculate the number of bits that one would save if one had to transmit a file which is as big as let us say 1 gigabit or 1 gigabyte or 2 gigabyte.

So, through these examples we have tried to establish how you can take advantage of the source coding schemes in order to reduce the number of bits that required to transmitted. So, this helps you in saving the number of bits and also saving bandwidth. And beyond that you can also imagine that extend your imagination or calculations in real terms of the amount of power that it saves which we should be able to calculate towards the end of the course. Basically their use the terminology number of or the amount of joules that is expanded oh that is pinned in sending 1 bit. So, if you can find that number let us say p joules are send is required to send 1 bit and I have to send I have saved let us say nearly 1 gigabit. So, we can easily calculate the amount of energy that can be saved.

So, it is multi fold implication if we can do source coding appropriately. We stop this lecture at this particular point, and we continue with these examples in the next lecture.