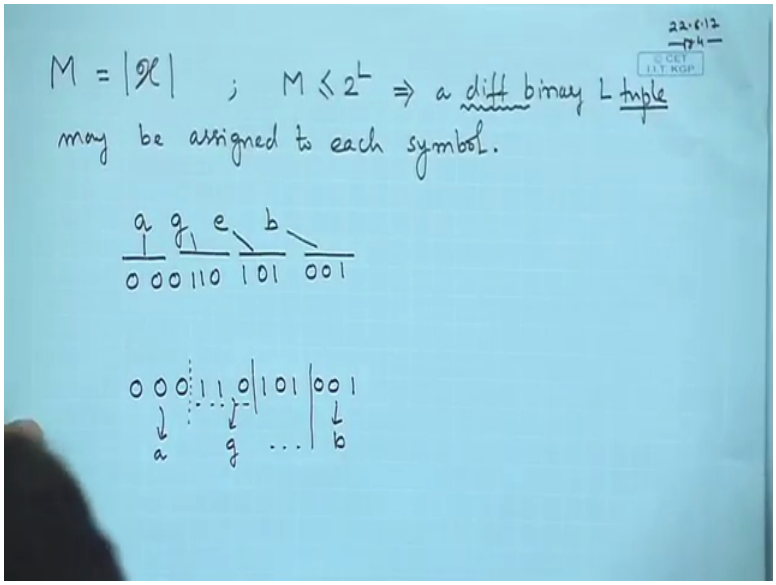
Modern Digital Communication Techniques
Prof. Suvra Sekhar Das
G. S. Sanyal School of Telecommunication
Indian Institute of Technology, Kharagpur

Lecture - 07
Source Coding (Contd.)

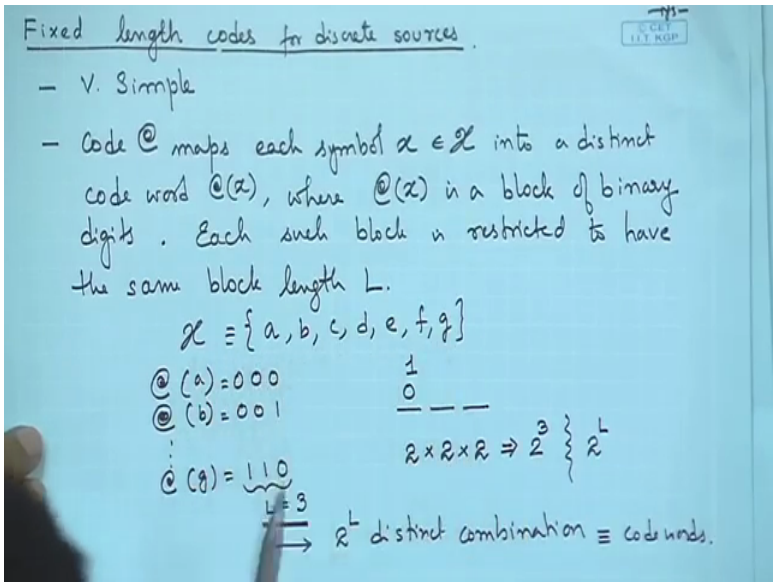
Welcome to the lectures on Modern Digital Communication Techniques. We are discussing source coding. In the previous lecture, we had introduced a fixed length code.

(Refer Slide Time: 00:35)



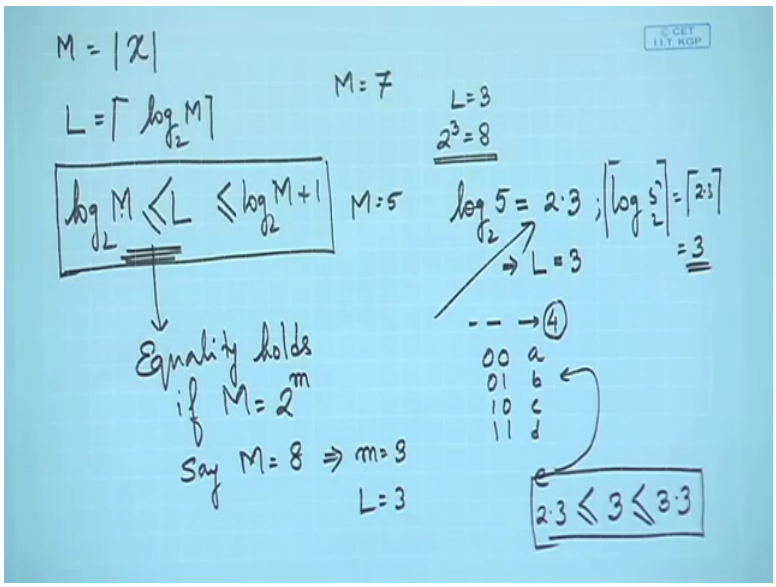
In the fixed length code, what we have said is that if you have M is the number of elements in a particular set, that means if x which is the set of all symbols in a discrete source which is given by an example like this.

(Refer Slide Time: 00:43)



So, x is the cardinality of the set. Then, if we choose L in such a manner that M is less than or equal to 2 to the power of L, then we can assign a different binary L tuple to each symbol. Now, this is very important and that is what we said in the previous lecture that this different means unique. Unique in the sense that every symbol is assigned a unique code. If every symbol is not assigned an unique code, then the decoder will not be able to decode. This is the fundamental thing for a source coding. Whatever kind of encoding we do, we would like the decoder do an ambiguously decode each of the symbols.

(Refer Slide Time: 01:35)



So, now we proceed with this particular description of things and we have already seen that if we have M, let us as discussed in the previous thing. M denote the size of the set which contains these symbols and we would assign L which is the length as ceiling function of log base 2 of M to indicate the number of bits required to represent a source symbol. So, by ceiling function what we mean is that if M is equal to let us say 7, if M is equal to 7, in that case what we are going to get over here is that L should be equal to 3 because 2 to the power of 3 is equal to 8. This we should keep in mind. So, if we can choose M L equals to 3, in that case we do not have any particular problem.

The other example could be if you have M is equal to 5, in that case log base 2 of 5 would be 2.3, approximately 2.3129 in that case would also lead to a value of L is equal to 3 because if you take the ceiling function of this, that means if I take log 5 base 2, that means I am taking the ceiling of 2.3 which is equal to 3. That means, if you have 3 bits, then you can uniquely decode if you choose 2. So, that clearly means I have 2 bits that clearly means there are four distinct possibilities 0 0 0 1 1 0 1 1. So, let us say a b c and d now if we have the 5th one e that would be reassigned to any of the code words and it will get confused with any of the previous code words.

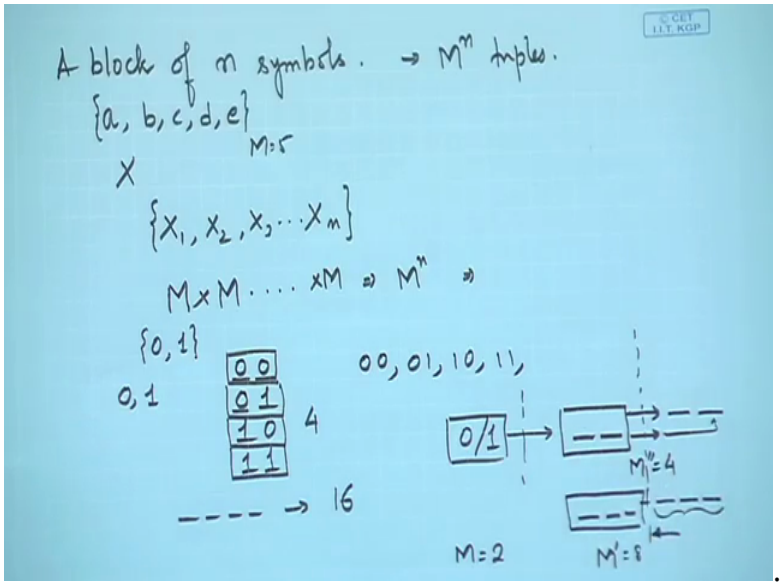
So, that is why we should use this particular restrained. Now, we can also say that L this number should be greater than or equal to log base 2 of M because this number is the largest possible integer or the smallest possible integer greater than m. This is basically the largest possible integer greater than m. So, what we have over here is greater than this particular number. So, this L because it is the smallest possible integer that we are talking about should be less than log base 2 of M plus 1 because this is a real number. If this is a real number, this may not turn out to be an integer. If it does not turn out to be an integer, then this is the smallest integer just greater than this and since this is the smallest integer just greater than this, it is less than this particular number.

So, this is also clear from this particular example if we take this example L is equal to 3. So, that means log of 5 base 2 is 2.3 and this is less than or equal to L is equal to 3 which is less than or equal to 3.3. So, that means 3 is the number which is between log of M base 2 and log of M base 2 plus 1. So, if this condition is satisfied, then you can make codes which are uniquely decodable. So, this equality would hold if M is equal to 2 to the power of M that is clear. That means, say M is equal to 8. That means, small M is equal to 3 and from this relationship we would find that L is equal to 3. In that case, this

equality holds because log base 2 of 8 is equal to 3 and L is equal to ceiling function of 3 is 3. So, that means they are all equal. So, this expression holds with equality if M is an integer power of 2. So, this is well said, but what we can see in this is that at most there can be one extra bit that would be required when M is not an integer power of 2.

Now, if you look at practical sequences or practical sources, you will not always find that the sources have 2 to the power of some integer number of discrete symbols. So, that is not always necessarily true. If that is not necessarily true, this extra margin that means here we have is around 0.7 bits. Roughly there is 0.7 bits which are extra per symbol. Now, if we are sending let us say 10 to the power of 7 such symbols, so that means 0.7 multiplied by 10 to the power of 7. So, many extra bits are required to communicate. So, we will see next that if this loss, if there is a possibility of improving things while using fixed length code.

(Refer Slide Time: 06:57)



So, this point what we will consider is let us say that we have a block of n symbols. So, that means there are M to the power of n tuples. So, what is meant that earlier we said that let a b c d e be the set. We have taken this particular example and we send one symbol at a time. Now, we say let there will be X1 X2 X3 up to Xn. Let this be a tuple, all aright. So, this is the tuple. That means, these n tuple because there are n symbols. So, this forms one block of symbols. So, if this forms one block of symbols, this is straight forward. The first one can take M different values, the second symbol would again chose

from this set. So, here M is equal to 5. In this particular example, this would take M different values and so on. So, there are M raise to the power of M different possible values for these combinations.

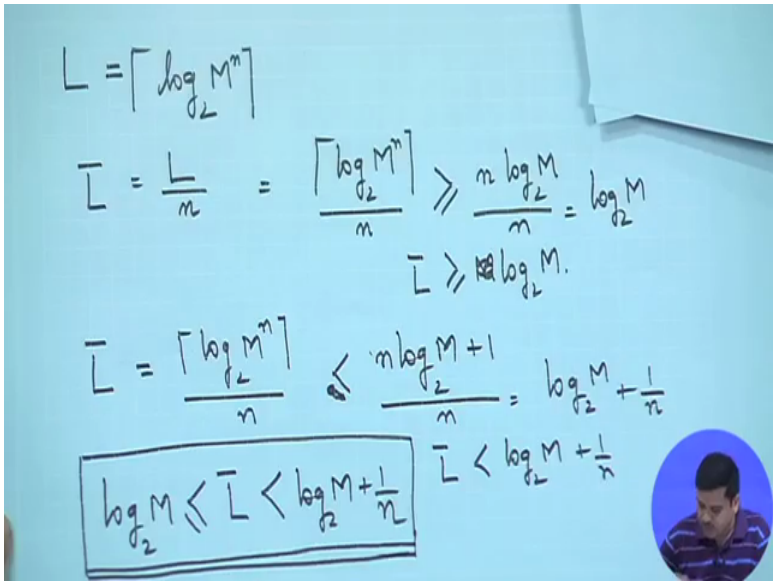
So, that means there M raise to the power of n different symbols again we can think of it like let us say I have 0 and 1 as my basic set. So, instead of this if I would group let us say 10, such or let us say 4 such or let us say even 2, the minimum is 2. So, that means I will have two positions. So, this can take a value of 0 or 1; this can take a value of 0 or 1. So, this can take a value of 0 or 1; this can take a value of 1 again like this. So, what we see that this forms a symbol, this forms another symbol, this forms another symbol and this forms another symbol. So, now previously this was one symbol that was coming out of it, this was another symbol that was coming out of it. Now, we are going to get symbols as this or as this or as this or as this or else any other sequence. So, that means, there are four possible symbols that have now been created. So, it is a super set that has been created from that same source.

So, what we are having is a bigger or a larger source which have been artificially created and we will see what the advantage of this is, you can go on doing look like this instead of taking 2 if I would take 4. So, that would clearly make 16 symbols. So, that means I have a new source which takes 16 symbols when easy way of looking at it is suppose there is a source which generates 0 and 1. I can have a buffer which places which has a storage unit of two. So, it takes the first one, it takes the second one and then, it gives out these two as the symbol, right. So, the first one is here and the second one goes there. So, every time if I look at this particular output, I am going to get 2 bits at a time. If I am looking at this output, I am going to get one bit at a time.

Similarly, if my buffer would have stored 3 bits and at every instant it would at a delay, of course there will be delay. It gives out 3 bits. The first one, second one and third one, I am going to group these and say that the source gives out 3 bits at a time. So, if we combine these three different bits, then what we have is 2 to the power of 3 and that is 8 possible symbols. So, at this point if I look at the source, I would say that this size of M I would call it M prime is equal to 8 whereas, here the size is 2. The size over here let us say mark it M double prime is equal to 4.

So, suppose we have this. Now, moving further in this case we would say that the number of bits that would be required to encode.

(Refer Slide Time: 11:07)



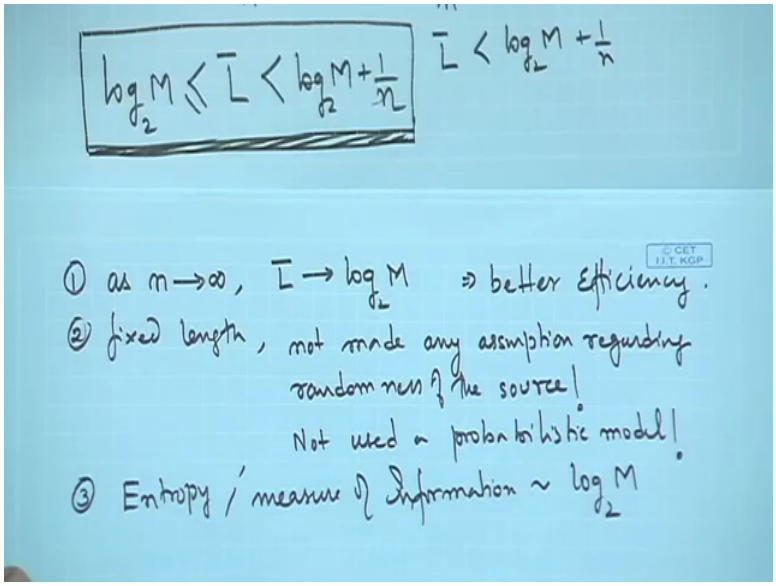
This would be the ceiling function of log base 2 M raise to the power of n. This is clear from our earlier setup. We have said L is equal to log base 2 of M. Now, we say that we have modified the source in such a way group them, so that our M has become M to the power of n M to the power of n. This is the total number of source symbols. So, if there are M to the power of n number of source symbols, in that case we are going to have L because log base 2 M to the power of n and the ceiling function of that so many bits required to encode each of this. For example, here I will require 2 bits here, I will require 3 bits as good as that. So, this is the number of bits that are required and L would indicate this as the number of bits per source symbol. That means, this is equal to L divided by n. Why I do this is because this has n symbols encoded in it. So, on an average every symbol will require L upon n number of bits per original source symbol.

So, if I look at this L bar, this is equal to log base 2 M to the power of n upon n this. Of course, the numerator is greater than n log base 2 of M because this whole number is greater than log base 2 of M to the power of n. So, that comes out in the front and then, you have the denominator and this particular term is equal to log base 2 of M. So, that means what we have established over here is L bar is greater than or equal to log base 2 of m. Similarly, we can also argue that L bar which is equal to log base 2 of M to the

power of n ceiling function upon n, now this number is definitely less than or equal to log base 2 of M plus 1 because log base 2 of M to the power of M ceiling function is less than log base 2 M to the power of n plus 1. So, M to the power of n comes here this upon n and this expression is equal to log base 2 of M n and n cancels out plus 1 upon n.

So, what we can establish here is that L bar is less than log base 2 of M plus 1 upon n. If we combine these two, then the expression that we can write is log base 2 of M from here is less than or equal to L bar and L bar is less than log base 2 of M plus 1 upon n. Now, what can we conclude from this particular expression, this is significant what it shows is that we look carefully into this set of expression.

(Refer Slide Time: 14:51)



What we would find? We can make a few conclusions. One that is as n becomes very large let us say n tends to infinity L bar tends to log base 2 of m, right. This is very interesting. So, if I make n very large. I can make this loss negligible. If we compare this to the earlier result, we look at this that L is less than equal to log base 2 of M plus 1, but here what we have is that this is less than log base 2 of M plus 1 over n.

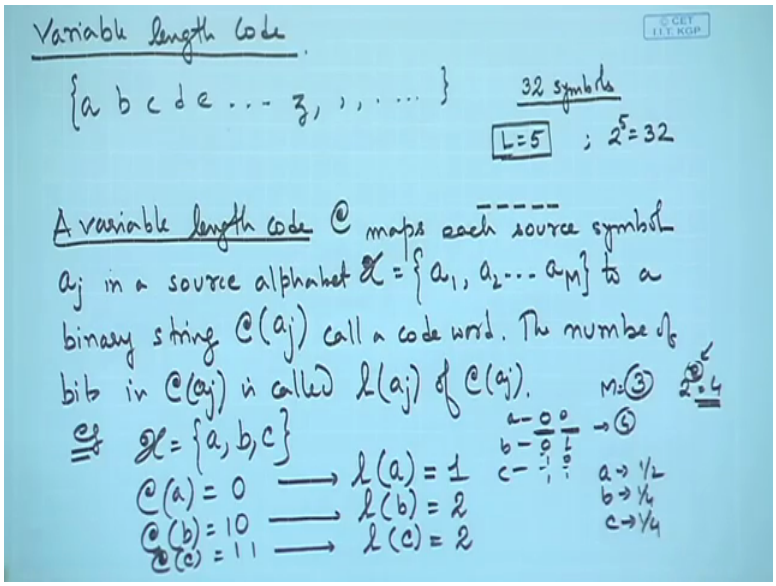
Now, this change has happened because we have taken a group of n symbols. You recall this. We have taken a block of n symbols because there are M to the power of n tuples that are possible. So, as there are M to the power of n tuples that are possible, this is the number of bits that would be required and we have showed here this L bar is greater than log base 2 of n and we have also showed is less than this particular number and

therefore, able to establish this particular relationship. So, with this particular relationship, we are able to establish that as n is made very large, our losses which were earlier can be reduced to significantly small number. So, that means we can have better efficiency.

The second thing that we can note over here is, it is still fixed length. This still fixed length and we have not made any assumption regarding the randomness of the source. This is important. So, we have not used a probabilistic model, right. So, even without using a probabilistic model, now with probabilistic model what is the advantage we will come to it very soon. Even without using probabilistic model, we are able to get a situation where we are able to reduce this particular kind of, we are able to reduce the overhead. The other important thing to note at this point is that this gives us a hint if you look at this or even the earlier expression, even this particular expression even this, but better this because even is very large, this tends log base 2 of M. You may have heard about the term entropy which we will again define. Why entropy or a measure of information is expressed as log base 2 of some number and that we are going to see at to certain point.

So, these are some of the observations which we can take and this is quite useful. We can refer back to this particular expression when we are doing random source or when we are taking probabilistic model and variable length source coding to hint at even fixed length codes can give us quite a lot of advantage if we take blocks which are pretty large.

(Refer Slide Time: 18:41)



So, with this we move to variable length code. Now, why we move to variable length code is simple because this source that we have said these letters that we have used let say even I take a b c d e, let us say up to z and a, a full stop and all these things together what is well known is that one of the largest used letters of such a source. There is the English alphabet source would be e, right compared to let say z. So, the next common letter that can be thought of is probably t. So, if you look at these particular statistics, what we find is if we draw histogram, we are going to find the most probable symbol is e, whereas the least probable symbol is may be z.

So, now, this can give us some advantage instead of assigning the same number of bits to all of this. So, suppose I have let us say 32 symbols for example. So, our previous exercise let us that we can choose L is equal to 5 because 2 to the power of 5 is equal to 32. So, that means I will be assigning the same number of bits to all the symbols whether it is e or t or h or q or z or p, whatever it is. However, these symbols do not come out with the equal probability. So, more often I am going to get e and less often I will get z. So, that gives us a hint that if we can assign lesser number of bits to e and more number of bits to z, then on an average we might have the number of bits going out of the source per source symbol which is less than if we had used this kind of a fixed length coding.

So, with that particular motivation we get into this variable length code. Now, variable length code you may have heard about mass code which was used in the telegram system

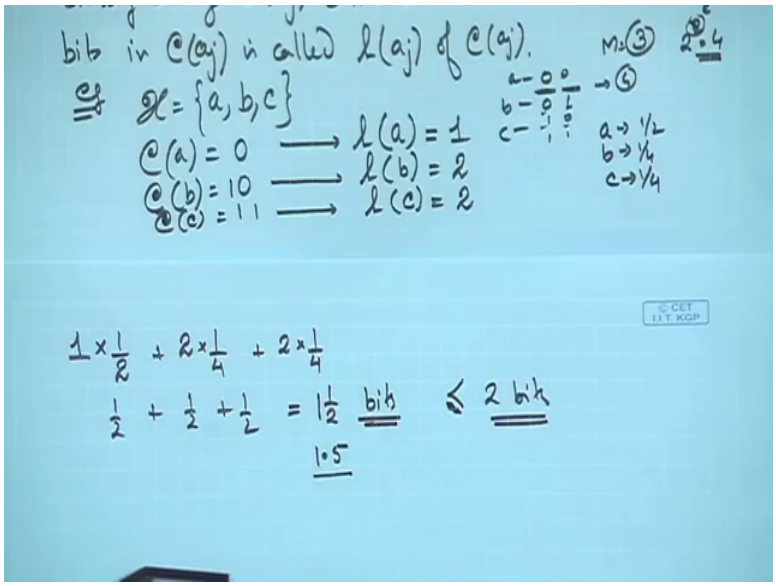
that inherently use this kind of property, where it use the more frequent of the letters or the symbols that are present. You are going to put less number of bits or dots and dashes to that particular symbol whereas, the less frequent symbol, you would make it longer stream of dots and dashes. So, same thing applies over here. So, we can think of a variable, a variable code, a variable length code let us say we put c which maps similar to the definition we have given for fixed length code maps each source symbol. Let us say we assign with the name a of sub j in a source alphabet x which contains a1 a2 up to am. Let us say we have this to a binary string c of a j called a code word or this is similar to the fixed length and the length or the number of bits in a j c of a j is called L of a j. This is the notation that we are going to follow of c of a j. That means, you are saying so many bits are assigned to this.

Now, this is an additional thing in case of fixed length code. We said it is the fixed number. In this case, every symbol will have a certain number of bits assigned to it, right. For example, if we take a particular example and let say our x contains a b and c, if we take a simple source like this and the lengths could be or the code let us look at the codes. The code of a if we say it is 0, then the code of b let it be 1 0 and the code of c let it be 1 1. In this case, we would point out length of symbol a is equal to 1. In this case, the length for symbol b is equal to 2 and in this case, the length of symbol c is equal to 2.

That means, we are using 1 bit to encode a 2 bits to encode b and 2 bits to encode c. So, in this case if it was a fixed length code, we would have chosen 2 bits because we have the number 3 M is equal to 3 in this case and if we have 2 to the power of 2 that is equal to 4. That is the smallest integer greater than log base to of 3. So, that means it is 2. So, that means in that case 2 is the number. So, we have two positions, 2 bits and there are four possible outcomes 0 0 0 1 1 0 1 1. So, we could assign this to a, this to b, this to c and this is not going to be used. So, every symbol is taking 2 bits, whereas here first one is taking 1 bit, this taking 2 bit, this taking 3 bit.

Now, if you could try to see that whether it gives an advantages or not, suppose I say that a comes with the probability of half, b comes with the probability of one-fourth and c comes with the probability of one-fourth.

(Refer Slide Time: 25:06)



If we take this particular example, then we can calculate that whenever a comes, we are going to get 1 bit. So, how often does a come half percentage of time? A comes plus, we are going to get 2 bits. Whenever b is going to come and b comes one-fourth of the times plus, we are going to get 2 multiplied by 1 by 4. So, this results in half plus half plus half which is equal to one and half bits on an average. This is less than 2 bits which were used in case of fixed length code. So, although you might be confused that what is the meaning of 1.5 bits, 1.5 bits what mean over here is on an average every symbol as you are seeing over here whenever the source generates a, it generates a 0. Whenever the source generates b, it generates 1 0. Whenever it generates c, it generates 1 1. So, that means that the source is generating distinct symbols and the encoder is generating distinct code words.

So, there is no question of generating half a bit. It is always generating integer number of bits, but if we are taking this source and observing it for a very long time, let us say we are looking at the source which generates 100 such symbols. Let us say we take that. So, what we will find that 50 percent of all those symbols will be 25 percent, of all the symbols will be b and 25 percent of all the symbols will be c. So, if I have used 0 to represent a, I am going to get 50 number of zeros. If I have used 1 0 to represent b, I am going to get 25 number of b s. That means, I am going to get 50 number of binary digits and c there are 25 number of c. That means, again I am going to get 50 number of total of ones and zeros. So, in total I am to get 150 bits if there are 100 such symbols whereas, if

you have a fixed length code, you would have used 2 bits per symbol. So, that means you would have had to encode these 100 symbols into 200 bits. So, this clearly shows you have saved 50 bits for this particular source when you are supposed to send 100 bits.

Now, just imagine when the world is going towards an exaltation communication systems and it is expected that the traffic would be like 10 to the power of 15 or 10 to the power of 17 kind of bits, then the huge amounts of saving can be done if we exploit the probabilistic nature of the source rather than use a fixed length coding, but again at tacit point I should like to remind you that required a few minutes ago we explained that if we use M tuples, then the loss that you would encounter in fixed length code could be reduced, but of course the penalty have to pay. There is a delay in it.

So, we would like to bring this particular discussion as I stop and we continue on with this in the next immediate lecture.