

Computational Complexity
Prof. Subrahmanyam Kalyanasundaram
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Lecture - 54
Introduction to Communication Complexity: Part 1

(Refer Slide Time: 00:15)

The image shows a screenshot of a presentation slide. The slide title is "Lecture 54 - Introduction to Communication Complexity". The word "Communication" is highlighted in yellow. Handwritten notes in blue ink state: "This model of computation was introduced by Andrew Yao (1979). There are two parties who wish to compute a joint function. The individual parties have unbounded computation power. But we want to charge a large cost for communication." Below this, it says "Alice and Bob have to compute" followed by a table:

Alice	Bob
001	101

At the bottom right, there is a small video inset showing a man speaking.

Hello and welcome to lecture 54 of the course computational complexity. This is also the beginning of week 11. In this week we will see the topic of communication complexity. So, I have highlighted, it is the key aspect here is communication. So, far we have seen models of computation where the complexity classes are based on a certain resource, how much of a certain resource is used it could be time, it could be space, it could be size of a certain circuit.

It could be randomness in the case of a randomized algorithm and so on. In this model the resource that is under consideration is communication, how much communication happens. So, what do I mean by communication? Well, in the setting of communication complexity there is another fundamental difference from what we have seen so far. So, in the models we have seen so far, all the input is provided to the Turing machine or the circuit as the case may be.

And it is up to the Turing machine or the circuit to decide or to output accept reject or to compute a certain function. However, in this model we assume there are two parties or two or more parties actually who are sitting in different places or it could be geographically separate

and they have to compute and they have both have their own inputs. So, the in the two-party setting these two parties are called Alice and Bob usually.

And Alice has a certain input x Bob has a certain input y and together they have to compute a joint function of x and y . So, it could be it could be the product of x and y it could be the sum it could be the difference or it could be certain other things. So, the function could be anything that depends on and it could also be just output Alice's last bit. It is also technically a function of x and y .

And the one way in which this differs from whatever we have seen so far is that there is no bound on the computation power, so both Alice and Bob are considered all powerful. They can do any computation that is decidable or that is doable or even perhaps undecidable things. The problem is that we are charging them on the communication. So, since the function they have to compute is a joint function to actually execute the computation they will have to send some bits across.

So, perhaps I will just send some to Bob perhaps Bob send some back to Alice perhaps there is a back and forth and what we are interested in is a total amount of communication that has happened so this is the model of communication complexity.

(Refer Slide Time: 03:25)

wish to compute a joint function. The parties have unbounded computation power. But we want to charge a large cost for communication.

Alice and Bob have to compute $f: X \times Y \rightarrow Z$. Alice has $x \in X$ and Bob has $y \in Y$.

Protocol π
 Each communication $a_i(x)$ depends on x and all the prior communication.

Alice	Bob
$x \in X$	$y \in Y$
$a_1(x)$	\leftarrow
\leftarrow	$b_1(y)$
$a_2(x)$	\leftarrow



So, as you can check the see the red part here. So, Alice has the input x from it from a domain capital X and bob has an input y from the domain capital Y and they have to compute a joint function f which is $f: X \times Y \rightarrow Z$ is a joint function of x and y and the joint function belongs to the set

capital Z. And in most of the I think at least in this we will in this set of lectures we will restrict that to be binary or Boolean, Z will be just 0 1 function.

So, the function that they will have to compute will be a 0 1 function at least for our purposes in the in the in the few set of lectures that we will have in our course. So, how can Alice and Bob jointly compute the function? So, well they could have some kind of an arrangement so depending on so they know what function they have to compute they just do not know the inputs.

So, rather Alice knows Alice's input but she does not know Bob's input, Bob knows Bob's input but he does not know Alice's input. So, well one way is so they could do some communication so one simple way is Alice sends everything to Bob, Alice sends her entire input to Bob and then Bob can compute the function and then Bob sends the function the computed function value back to Alice, this is one way.

So, but then sometimes there are better ways of doing this, whatever I just said is available regardless of whatever function it is but there may be better ways for certain functions. So, in general what could happen is Alice sends something to bob, let us say a_1 is what she sends and then bob upon seeing a_1 he gets some information, so a_1 could be 1 bit, 2 bits whatever depending on the protocol that they have agreed upon.

So, depending on what a_1 is bob sends back b_1 , so b_1 could depend on a_1 it need not depend on a_1 . So, b_1 is a function of both what Bob has and also what Alice has sent and then maybe Alice send something back a_2 , a_2 will be a function of what she has combined with what function Bob sent so what input bob sent. So, like this back and forth could go on for many rounds.

But what we care about is not how many rounds of back and forth happen rather how many total bits have been communicated. So, it does not matter if they come communicated 10 bits over 10 rounds or 10 bits over one round, what matters is how much of communication has happened.

(Refer Slide Time: 06:18)



Each communication $a_i(x)$ depends on x and all the prior communication. Similarly for $b_j(y)$. At the end of all the communication, the function value $f(x, y)$ must be clear from the transcript.

Def Comm. Complexity $\} D(f) = \min_{\Pi} \left(\max_{\substack{x \in X \\ y \in Y}} \text{no. of bits used by } \Pi \text{ to compute } f(x, y) \right)$

Applications in other sub-areas of complexity.



So, again the same thing I have written down in pros each communication a_i depends on so a_i is the in input sent by Alice to bob, it depends on x and all the prior communication she has received. Similarly, b_j depends on y and all the prior communication Bob has received. At the end just by looking what has been transmitted, what Alice sent, what Bob sent and what Alice sent back and etcetera.

Now one should be able to compute the function just by looking at the message transcript. So, message transcript is the set of all the messages that have been communicated. Of course, if you must know the protocol without the protocol you cannot make sense of the transcript. So, the message transcript is the entire set of messages that have been sent back and forth by looking at the transcript one should be able to determine what the function is.

So, obviously Alice and Bob both of them know the entire transcript because Alice either all the a_i 's were sent by Alice so Alice knows a_i 's, all the b_j 's were received by Alice all and it also knows b_j s, similarly Bob also. So, at the end the function value should be clear from the transcript meaning there should not be any confusion on the function value, it should be one should be able to determine the function value from the transcript, this is the requirement.

So, what is the deterministic communication complexity? So, in these few lecture that we will see we will only see deterministic communication complexity. We will briefly describe something called randomized communication complexity later. But we will see mainly deterministic communication complexity. So, there is a corresponding randomized notion as well. It is denoted by the symbol $D(f)$ capital D of f .

It is so basically given a certain protocol, protocol is a certain protocol an agreement between Alice and Bob to communicate in a certain way to compute the function to jointly compute the function. So, let us say π is a protocol π so given a certain protocol so we want to minimize over all the possible protocols. So, there could be protocol that does a lot of communication but we do not want that we want the protocol that does the smallest communication.

So, what do I mean by communication done by a protocol so what so consider all the possibilities what input Alice can have, what input Bob can have? So, what is the worst case over all pairs of inputs that they have. So, it is possible that in some inputs some x some pairs x, y without much communication they are able to compute the function. So, what we are interested in is what is the pair x, y that causes them to communicate the most.

And that is what we will consider so \max of all \max over all the pairs x, y the number of bits communicated by the protocol π to compute the function. And we want to choose π such that π minimizes the maximum communication for all pairs x, y . So, this may seem confusing if this seems confusing, think of any algorithm. How do we measure algorithms running time? We let us say we want to say sorting can be completed in order $n \log n$ in time.

Or the complexity of sorting is $n \log n$, what do I mean by that what do you mean by that is that you consider so consider any algorithm for sorting and we want to consider we will say the complexity of sorting is $n \log n$ because that is the fastest or most efficient algorithm that can come that can come perform the sorting. So, that is like the minimum overall π the minimum over all procedures that do sorting.

And then within that once what do we want, for a certain algorithm we want to maximize we want to consider the worst case input. So, that is what is happening here as well. So, certain given once the protocol π is fixed, we want to consider the worst case input pair x, y . So, it is exactly in parallel to what we know about computational problems and algorithms how we say that a certain problem can be done in a certain time.

(Refer Slide Time: 11:01)



As clear from the transcript:

$$\left. \begin{array}{l} \text{Def Comm.} \\ \text{Complexity} \end{array} \right\} = D(f) = \min_{\Pi} \left(\max_{\substack{x \in X \\ y \in Y}} \text{no. of bits used by } \Pi \text{ to compute } f(x, y) \right)$$

Applications in other sub-areas of complexity:

- Streaming
- Limit lower Bounds (we'll see one).

Examples:



So, this may seem so things will become clear with some examples that we will see very soon. And so, one may wonder why is this why are we studying this? So, the other models that we have seen so far had some motivation like they were real models of computation or they look like what we think are real models but what is this like why would like why would Alice and Bob be so powerful but then they have to pay so much for communication that is the model.

We do not charge anything for their local computation but we charge further only for the number of bits that are being sent. So, there are situations like distributed computing where the major choke point becomes the communication rather than the local computation and interestingly even without appealing to distributed computing etcetera. There are other areas of complexity other sub areas of complexity where communication complexity finds applications.

So, for instance streaming, streaming algorithm is a class of algorithms where you have such a large amount of input that you cannot possibly store it in your memory. So, you can make passes through the input but cannot store the entire input. So, in these passes you have to compute certain function of the input, so you cannot store the input in the memory. So, this may seem surprising but you can use lower bounds from communication complexity to arrive at lower bounds for streaming.

So, if you want you can say things like because function f takes this much communication complexity or requires at least this many bits of communication that means that a certain

function let us say g to compute in the streaming setting requires at least order this much space or these many passes. So, if once you learn about it will be it will be fairly natural but the idea is that in the streaming you have a limited memory.

So, you could think of as communicating between the current memory and the future memories. So, you could think of or rather you could think of the memory beings used as a vehicle to communicate between the present and the future or present and the past. So, that is a very high level picture of why communication complexity is applicable in streaming. Another surprising application is we saw the model of computation called circuits.

So, we will actually see one circuit lower bound using communication complexity. So, it will be a very specific setting and for the certain class monotone functions but still it is interesting to see how what we have already seen such as circuits has a lower bound coming from something communication complexity. So, you one may think that communication has nothing to do with circuits but then there is a way to use this to get an application as in circuits. So, in fact this part we will actually see in the upcoming lectures.

(Refer Slide Time: 14:37)

① EQUALITY: $EQ: \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}$
 $EQ(x,y) = \begin{cases} 1 & \text{if } x=y \\ 0 & \text{o.w.} \end{cases}$
 Alice's input
 Protocol: Send x to Bob. Bob sends 0/1 back.
 $D(EQ_n) \leq n+1$. In fact $D(P) \leq n+1$
 for any $f: \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}$

② PARITY $(x,y) = \bigoplus_{i=1}^n (x_i \oplus y_i) \rightarrow 1$ iff an odd no. of 1's in x and y combined.
 There is a trivial $n+1$ bit protocol.
 But Alice can send $\bigoplus_{i=1}^n x_i$ and Bob



So, let us see some examples, so one simple function is the equality function. So, Alice has x bob has y and they have to compute the equality function. What is the equality function? Equality function is simply 1 if and only if Alice's input and Bob's input are the same. So, only if and only if $x = y$ the output is 1 otherwise it is 0. So, if x is not equal to y is 0. So, one very simple protocol is for Alice to send x her entire input to Bob.

So, this is the entire input Alice is sending to Bob and now that Bob has x and he already had y , he can just check whether x and y are equal and once that happens, she Bob sends a 0 or 1 back he sends 0 if they are not equal, he sends 1 if they are equal. And what is the complexity of this protocol. So, this is a specific protocol so Alice sends her entire input, Bob sends the function, so the entire input so again the let me just be a bit more give a bit more details.

So, both Alice's input and Bob's input are considered to be n bit binary vectors and they have to compute the 0 1 function. So, Alice sends her entire n bits so that is n bits of communication, Bob sends one bit back so that is $n + 1$ bits of communication and regardless of whatever pair $x y$ in happens in this case they use $n + 1$ bits of communication. So, $n + 1$ is the number of bits of communication for this protocol.

But there could be better there could be other protocols that do even better. So, what we know is that the deterministic communication complexity of equality so denoted by EQ subscript n equality of n bit numbers is at most $n + 1$. So, this protocol gives you a complexity of $n + 1$ but specific maybe there are better protocols. So, we know that deterministic communication complexity of equality is at most $n + 1$.

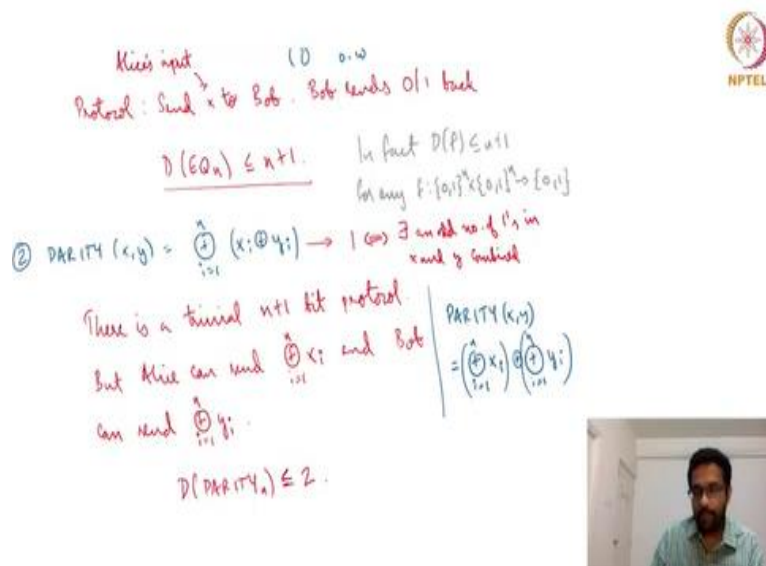
And some of you who are paying close attention may already be thinking that this protocol had nothing to do with equality. We did not use any special properties of the function equality. Alice sends her entire input Bob computes a function and sends it back. So, with any function whose output is 0 1. We can implement this protocol so this means any Boolean function that when they jointly compute Alice can send her entire input and bob can send 1 bit output.

So, this is true for any f . In fact, maybe I just write it over here, in fact $b f$ is at most $n + 1$ for any function let us say 0 1, 0 1 to 0 1. So, we did not use any specific properties of equality. So, now let us see another function called parity. So, parity of $x y$ is the total number of bits so it is $x_i y_i$ parity of that and then parity of the entire thing. So, in other words this is simply one if and only the total number of bits in x and y the total number of 1s in x and y is odd.

So, maybe I just write here 1 if and only if there is an odd number of 1s in x and y combined. There is an odd number of ones in x and y combined this is a parity. So, of course we could

do whatever we did with equality we could just send have Alice send everything to Bob and then Bob could send it back so this this would take $n + 1$ bits.

(Refer Slide Time: 19:23)



Alice's input (x, y)
 Protocol: Send x to Bob. Bob sends 0/1 back.
 $D(EQ_n) \leq n+1$. In fact $D(P) \leq n+1$
 for any $f: \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}$
 ② $PARITY(x, y) = \bigoplus_{i=1}^n (x_i \oplus y_i) \rightarrow 1 \Leftrightarrow \exists$ an odd no. of 1's in x and y combined.
 There is a trivial $n+1$ bit protocol.
 but Alice can send $\bigoplus_{i=1}^n x_i$ and Bob can send $\bigoplus_{i=1}^n y_i$.
 $D(PARITY_n) \leq 2$.
 $PARITY(x, y) = \left(\bigoplus_{i=1}^n x_i \right) \oplus \left(\bigoplus_{i=1}^n y_i \right)$

But however, in this case there is a better algorithm better protocol. So, notice that parity of x y is actually you can also write it like this you could compute the individual parity of x and compute the individual parity of y and then take parity of these two things. So, it is because the parity is distribute so you could take the individual parity of x and individual parity of y and you could take the parity of both of them together.

So, one thing that you can do is Alice can send the parity of her entire input and Bob can send the parity of his entire input and now both of them can compute the function and so can anybody who is just looking at the transcript. Alice just needs the parity of Bob's input and Bob just needs the parity of Alice's input. What is the complexity here the communication complexity here?

It is simply two because Alice just sends one bit and Bob just sends one bit back. So, we have shown a protocol where the maximum communication required is 2 bits and so this is an upper bound. So, just coming back to equality in fact we will later see that the communication complexity of equality is actually equal to $n + 1$ even though here we showed an upper bound.

(Refer Slide Time: 21:00)



can read $\{0,1\}^n$

$D(\text{PARITY}_n) \leq 2$

③ AVG. Alice gets $S \subseteq \{1, 2, 3, \dots, n\}$
 Bob gets $T \subseteq \{1, 2, 3, \dots, n\}$

$S = \{1, 3, 5, \dots\}$
 $S = \{1, 2\}$
 $1010101\dots$
 $1100\dots0$

AVG(S,T) = Average of multiset
 S ∪ T

If $S = \{1, 2, 5\}$, $T = \{1, 2, 4, 6\}$
 $\{1, 2, 5, 1, 2, 4, 6\}$

AVG(S,T) = $\frac{1+2+5+1+2+4+6}{7} = \frac{21}{7} = 3$



Another function say average, so let us say Alice gets a set S of 1, 2, 3 up to n a subset of 1, 2, 3 up to n t gets a set, Bob gets us at t which is subset of 1 to 3 up to n. So, and they have to compute the average of the multi set between them so just let me give an example. Let us say Alice gets the set S which is 1, 2, 5 and Bob gets a set of t which is 1, 2, 4, 6 then the S union t the multiset union is actually if a element is common to both then it will be counted twice.

So, it will be the s the multiset will be the multiset union will be actually 1, 2, 5, 1, 2, 4, 6. So, we retain the multiple copies so 1, 2, 1 and 2 appear two times so we retain the multiple copies and then the average of this set. So, it is simply 1 + 2 + 5 + 1 + 2 + 4 + 6, so 1 + 2 + 5 is 8 1 + 2 + 4 is 7 15 + 6 is 21 divided by 7 is 3. So, the average of the multiset s union t is 3, so how can Alice and bob come up with a protocol.

So, this is something I am just I am not going to explain how but I would like you to think about it and one more point that I want to mention now. So, till now we talked about Alice and bob getting a binary string but here we are talking about Alice and Bob getting sets or subsets but it is essentially the same thing so for instance. If Alice's set is just let us say the set of all odd numbers let us say S is 1 3 5 and so on this corresponds to the binary vector 1 0 1 0 etcetera.

Basically, the first bit indicates whether 1 is that 1 is there second bit being 0 indicates 2 is not there third bit b one indicates that 3 is there fourth bit being 0 indicates four is not there and so on. And another example is let us say this is just a set 1, 2 two element set, this

corresponds to the binary string 110000. So, the first two bits are 1 and followed by a bunch of 0s.

So, you can see how there is a correspondence between a subset of 1 2 3 up to n and an n bit vector. The all ones vector will correspond to the set 1, 2, 3 up to n. All 0s vector will correspond to the empty set. So, again here also the input is a 0 1 vector of length n but the output is actually a number.

(Refer Slide Time: 24:18)

EXERCISE 1: Come up with a protocol for AVG.

MEDIAN: $S, T \subseteq \{1, 2, \dots, n\}$.

Binary search: Alice & Bob have an interval $[i, j]$ within which the median should lie.

Alice: Sends no. of elements above $\frac{i+j}{2}$ and no. of elements below $\frac{i+j}{2}$.

Bob: 0 if median is above $\frac{i+j}{2}$.

So, I would like you to think of how to come up with a protocol for this. So, each one of them can compute their average and send but that may not work out. How will they communicate the average all these things need to be thought about? So, just try to work it out and try to compute exactly how much of communication is required. Another problem is of median. So, again Alice and Bob have sets S and T denoted the way.

I said about like here there is another example set 1 3 4 is 1 0 1 1 0 let us say over 5 bits. And how do they compute the median how can they compute the median of the multiset union?

(Refer Slide Time: 25:20)



④ MEDIAN: $S_1 \cup S_2 \cup \dots \cup S_k$

Binary search: Alice & Bob have an interval $[i, j]$ within which the median should lie.

Alice: Sends no. of elements above $\frac{i+j}{2}$ and no. of elements below $\frac{i+j}{2}$

Bob: 0 if median is above $\frac{i+j}{2}$
1 if median is below.

Now, repeat with $\frac{i+j}{2}$ replacing a boundary.
i.e. $T.C. = O(\log n)$



Again, multiset union like we said before. So, one protocol is based on binary search, let us explain the protocol. So, here Alice and bob will always retain or always maintain; an interval within which the median should lie an interval i, j . So, to begin with they will say that the median will lie anywhere between 1 to n because those are the elements under consideration. And then what Alice does is Alice sends the number of elements above the midpoint.

And number of elements below the midpoint, so Alice sends both. Because there is no requirement that Alice has a contiguous set or some anything. So, Alice sends the number of elements above as well as below the midpoint in her own set. Now Bob knows the midpoint Bob also can see how many elements are above the midpoint and below the midpoint in his own set and depending on that he can see where the larger number of elements.

So, what I am saying is that if this is a range let us say i and this line j and this is the midpoint $i + j$ by 2. Alice can send how many elements are there in her set in this range and Bob can also do this and depending on where and let me repeat. Alice send how many elements are there in the range above $i + j$ by 2 not in this range alone above $i + j$ by 2 and below $i + j$ by 2 and obviously Bob also knows how many elements that he has in his range.

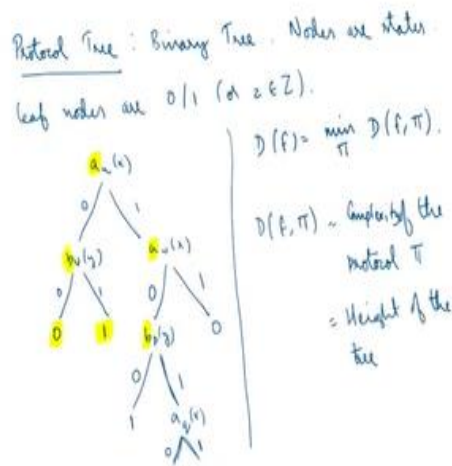
He does not need to send it but he knows. So, now after this he knows the total number of elements and he also knows how many so which side it is above $i + j$ by 2 or if it is below $i + j$ by 2 or the median is equal to $i + j$ by 2. So, depending on that he will inform Alice that the

median is above or below, so now maybe let us say he says it is above. Now Alice can refine the range let us say he says it is above.

So, now you set i to be this and i to be the new i to be $i + j$ by 2 and then repeat. So, every time the range of where the median is being narrowed and how many times do, we will we have to narrow? Because every time we are at least making the range half of the original size so, we can we do it at most $\log n$ number of times because initially the range is of width n and at each time, we are sending $i + j$ by 2.

We are sending the count; the count of numbers will be some the number of elements that we know there are at most n elements so the count will be at most of a number of size order $\log n$. So, order $\log n$ many bits and order $\log n$ exactly $\log n$ many rounds.

(Refer Slide Time: 29:13)



So, it is order $\log n$ multiplied by $\log n$ the complexity is order $\log n$ squared or order \log square. So, this is another example of a protocol. So, now let me explain another important notion say in the case of communication complexity. It is called a protocol tree, so this is just another way of representing the same protocol that we already explained. So, the nodes are the states it is a binary tree the nodes are the states and the leaf nodes are all marked 0 1.

So, assuming that the function is a 0 1 function. So, let us say Alice is the one to speak so whenever Alice is the one to speak then the function will be a something and whenever Bob is the one to speak the function will be b something. So, suppose Alice says a u x which is

some function on her own input and she may compute the function and transmit it to Bob. Let us say the function is 0 or the function is 1.

Bob, so now what I am saying is that if the function is 0 then Bob may respond and if the function is one perhaps. So, it is not always necessary that the protocol has Alice saying something followed by Bob saying something even in the same protocol that I have drawn here. When Alice says 0 Bob responds and when Alice is 1 Alice continues to transmit. So, it may not be every branch need not have the same kind of communication interchange.

So, when Alice goes to the address says 0 initially Bob speaks and when Alice is 1 then it means that Bob knows to wait that and allow Alice to speak as to speak. So, you can at the bottom you have these leaves which are marked terminals mark 0 or 1 and when you have another function here like a q here which means there is another sub tree beneath a q . Again, some a q will be 0, something will happen a q 1 something else will happen and so on.

So, the leaves mean that let us say when I have it when I reach a let us say a u is 0, b is 0 and then you reach this left most leaf. What it means is that? At this stage the function is determined and the function is 0. So, Alice and Bob can stop communicating because at this stage it is there is only one way to or there is this me if they have both send 0 initially this means the function has to be 0 and both of them know that and then they can stop communicating.

This is what it means and it is easy to see that the total bits communicated it is actually the height of the tree. So, the what is the longest route to leaf path for whichever leaf it is, that is the height of the tree and that is also the depth of the protocol maybe not really depth complexity of the protocol. So, the complexity of the protocol is not denoted D_f by this is the complexity of the protocol not of the function.

The complexity of the function will be D_f is actually minimum over all protocols $D_{f, \pi}$ and $D_{f, \pi}$ is a complexity of the protocol. So, which is the maximum so the height of the tree means it is the maximum length of the path. So, here the in this picture that tree is of length at least 1, 2, 3, 4, we do not know how deep this tree goes. But even though there is a root leaf path of length 2 that is not the maximum.

happens so suppose x_2 and y_2 are the same x_2 and y_2 are both 0, what does it mean? This means that x is 0 0 and y is 0 0. Suppose x_2 and y_2 are both 1 in this case we know that x is 1 1 and y is 1 1 and there are two other cases where x_2 is 0, y_2 is 1 so which is 0 0 and 0 1.

Or another case when x_2 is 1 y_2 is 0 so it is 0 1 0 0. So, in this case so you say the answer is yes and no as per the input combination. So, when they are equal, we need to say s when they are not equal, we say no. And similarly, we have a sub tree marked in the red as a red triangle in the right side as well. So, this is the protocol tree for Alice and Bob when they compute equality or this is one protocol tree for a certain protocol, so this is the protocol tree.

So, you can see that in some cases it is possible for one node of the protocol tree one leaf of the protocol tree to capture multiple input combinations. So, the circled leaf here at the height at the depth 2 is capturing four combinations x is 0 0 0 1 and y is 1 0 1 1 and this one also this second no is also capturing four combinations. But the leaves at the depth of four are only capturing one combination each.

So, sometimes the leaves capture more sometimes the leaves do not capture more and so can we say something about what do like what are the inputs that reach each node of the protocol tree and what can we infer about the complexity of the protocol from that. So, that; and some inferences is from that we will see in the next lecture. So, what we have seen so far or maybe I will just say one more thing is that what we are mostly interested in is bounds on the complexity.

So, here we saw equality and this particular equality protocol the tree has depth 4, even the this this red part is that is shaded is not 8 filled but that will be symmetric to the one in the right in the left so this tree has depth four so this is a protocol for equality that uses four bits. So, is this the best protocol? No, because we already saw that there is a protocol where Alice sends her entire input to Bob and Bob just computes the function so that is just 3 bits $n + 1$ bits.

But this is just to illustrate what can be done with a protocol or how to represent using a protocol tree and we will be seeing how to bound the communication complexity of a certain function using various techniques. So, but perhaps I should do it in the next lecture so just to

summarize what we have seen. We have seen the communication complexity model; we have seen some examples and we have seen what is a protocol tree.

How we can be used to represent a certain protocol for it to compute a certain function with the example of equality. Next lecture we will see some bounds arising out of this. Thank you.