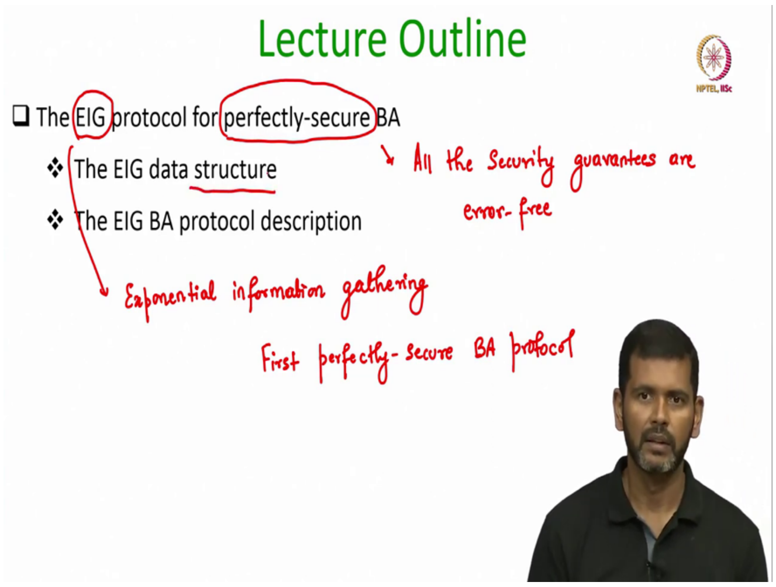
Secure Computation: Part II

Prof. Ashish Choudhury
Department of Computer Science and Engineering
Indian Institute of Information Technology, Bengaluru

Lecture - 03
EIG Protocol for Perfectly-Secure Byzantine Agreement

(Refer Slide Time: 00:36)



Lecture Outline

- The EIG protocol for perfectly-secure BA
  - The EIG data structure
  - The EIG BA protocol description

All the Security guarantees are error-free

Exponential information gathering

First perfectly-secure BA protocol

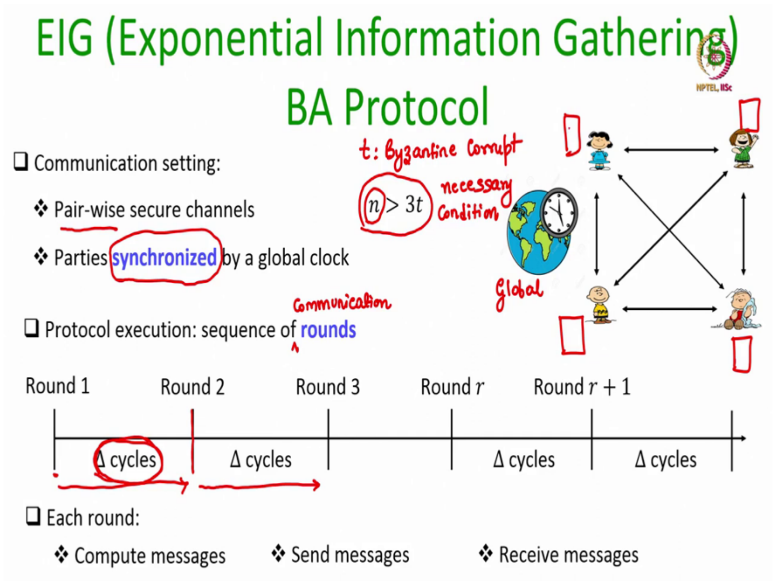
Hello everyone, welcome to this lecture. So, in this lecture we will discuss about EIG Protocol for Perfectly Secure Byzantine Agreement. Specifically, the lecture outline is as follows: we will see the EIG data structure and the protocol description. In the subsequent lectures we will do the analysis of the protocol.

So, this is a perfectly secure BA protocol Byzantine agreement protocol. That means, all the security guarantees are error free. Namely, the termination consistency and validity properties will be achieved without any error, even against a computationally unbounded adversary ok.

And it is called EIG because the full form of EIG here is exponential information gathering. So, as the name suggests, the running time, the computational complexity of the algorithm will be exponential in the number of participants. But still, we are going to discuss this BA protocol. Of course, we have polynomial running time BA protocols which we will discuss subsequently. But the reason for discussing this protocol is because of its historical significance. This is the first perfectly secure BA protocol ok.

And it is based on a very nice data structure and a very nice idea. So, that is why we are going to discuss this protocol ok.

(Refer Slide Time: 02:27)


EIG (Exponential Information Gathering) BA Protocol

- Communication setting:
  - Pair-wise secure channels
  - Parties synchronized by a global clock
- Protocol execution: sequence of rounds
- Each round:
  - Compute messages
  - Send messages
  - Receive messages

NPTEL, IISc
t: Byzantine Corrupt
$n > 3t$
necessary Condition
Global
Communication
Round 1
Round 2
Round 3
Round r
Round $r+1$
Δ cycles
Δ cycles
Δ cycles
Δ cycles

So, what is the communication setting which we are going to consider in the protocol? So, we have pair wise secure channels between every pair of participants. So, we assume we have n number of participants and t of them could be byzantine corrupted. It could be possible that there exists no corrupt parties during the protocol execution, but if at all any corruption occurs there could be at most t number of corruptions. The exact identity of the t corrupt parties will not be known beforehand ok.

And we assume that we have a dedicated secure channel between every pair of participants ok. We also require the condition n to be greater than 3t to hold for this protocol, otherwise the properties will not be achieved. Later on, as we proceed in the course we will prove that this condition n greater than 3t is actually a necessary condition for perfectly secure Byzantine agreement.

That means, if this condition is not there, then you can never design any kind of perfectly secure Byzantine agreement protocol. We are in the synchronized communication setting and in fact, the whole course will be considering only synchronous communication model. Where we assume that we have some kind of global clock, a common clock through which all the participants of the system are synchronized.

So, even if the n participants they are present across various corners of the globe, we assume that they will be synchronized through some clock common global clock. And as a result of that the protocol will operate as a sequence of communication rounds ok. So, whenever we assume a synchronous communication model the underlying protocols are assumed to be operated as a sequence of communication rounds.

So, you can imagine that we have several rounds one after the other and between every consecutive rounds, there exist a delay of a time gap of some Δ cycles. So, that cycle could be measured in terms of seconds or minutes or days or hours or weeks or so on. But there will be a fixed starting point and an end point for each round. So, for instance before the protocol starts the parties will have their own inputs etcetera and then the protocol starts, so we will be in round 1.

And everyone will know that every participant in the protocol will know that round 1 will be over after Δ clock cycles and once the Δ clock cycles have elapsed, round 1 gets over and round 2 starts.

Then round 2 will takes Δ clock cycles it will elapse for Δ clock cycles and so on. So, what happens in each round? So, in each round as per the protocol the parties will be doing the the following set of actions. So, remember when I say I have a BA protocol there is a piece of code, an algorithm which is supposed to be executed/run by every participant in the system.

And that algorithm will be executed as a sequence of rounds. So there will be steps mentioned in the algorithm as a sequence of rounds, where is in each round the parties will compute some messages as per the protocol description. And once the messages are computed they are communicated to the designated receivers, so that is a send action.

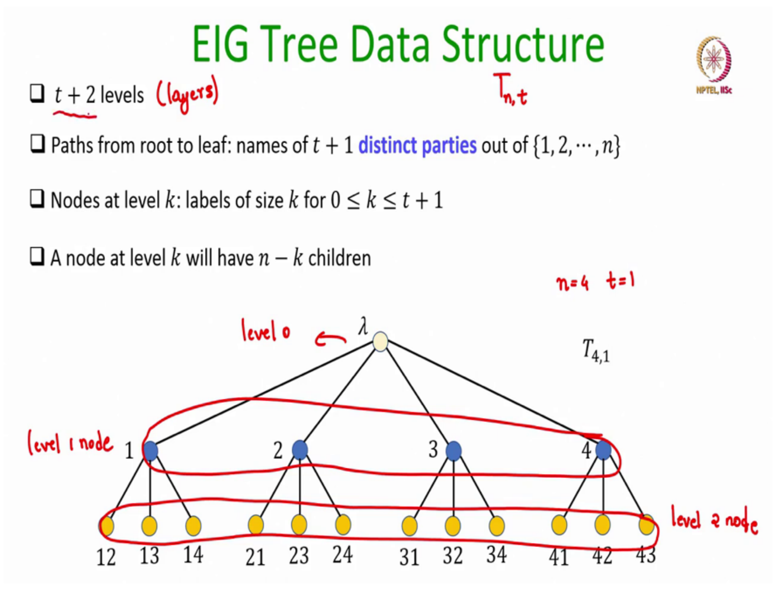
And whatever messages a party sends to other parties they will be received by the corresponding receivers in the same round. So, all these actions, compute, send and receive will be taking one full Δ cycle and that will constitute one round. Now, based on whatever messages a party receives at the end of the previous round it will process those messages as per the protocol code.

And then in the next clock cycle whatever messages that party is supposed to send to the other parties it will compute them, it will send them and receive all the messages that other parties would have sent to that party in that round. And again the same sequence of actions namely compute, send and receive gets repeated for the next clock cycle. And this keeps on happening for some specific number of rounds depending upon how many communication rounds are there in your underlying protocol right.

So, if my protocol has say 10 number of rounds, then this sequence of actions namely compute, send and receive will be performed 10 times. The honest participants namely the parties who are not under the control of the adversary will stick to the protocol code and they will compute the messages and they will send the messages as per the steps defined by the protocol.

But if there is a party who gets corrupted by the adversary then it may not follow the protocol steps. It may compute arbitrary messages, it may send arbitrary messages, it can behave in a completely arbitrary fashion, we have absolutely no control over their behavior. So, that is the setting in which we are going to design the EIG protocol.

(Refer Slide Time: 08:36)



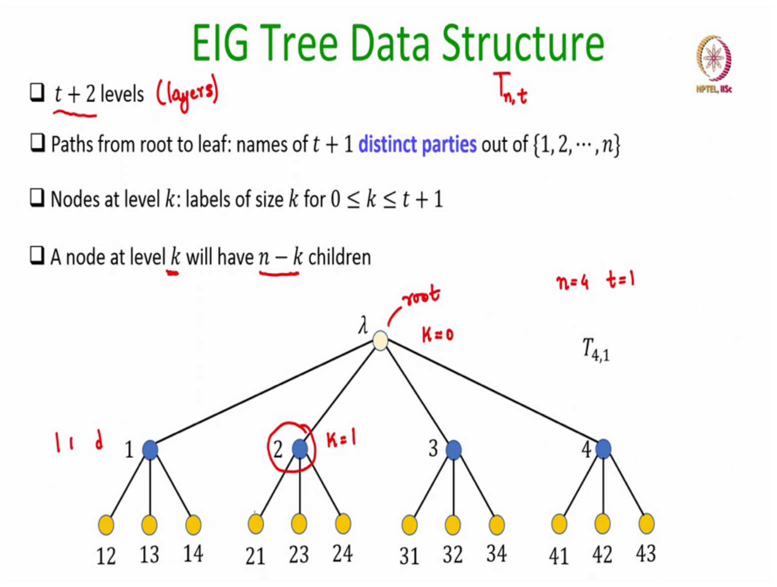
EIG Tree Data Structure

- $t+2$ levels (layers)
- Paths from root to leaf: names of $t+1$ distinct parties out of $\{1, 2, \cdots, n\}$
- Nodes at level k: labels of size k for $0 \leq k \leq t+1$
- A node at level k will have $n-k$ children

So, now let us see the EIG data structure ok. So, this will be denoted by the notation $T_{n,t}$, n is the number of participants, t is the number of corruptions. So, for instance I am taking here n to be 4 and t to be 1 ok. So, this is how the EIG data structure will look like if n is equal to 4 and t equal to 1. So, it will be a tree with t plus 2 levels or layers ok. So, this is level 0 namely the root all these nodes are level 1 node then all the nodes here they are level 2 node and so on.

So, in this example I am taking t equal to 1 and a general EIG data structure tree data structure will have t plus 2 level levels layers. So, it is for t equal to 1 it will be 3, so we have 0 layer, 1 layer and 2 layer. So, total 3 layers or 3 levels ok.

(Refer Slide Time: 10:34)

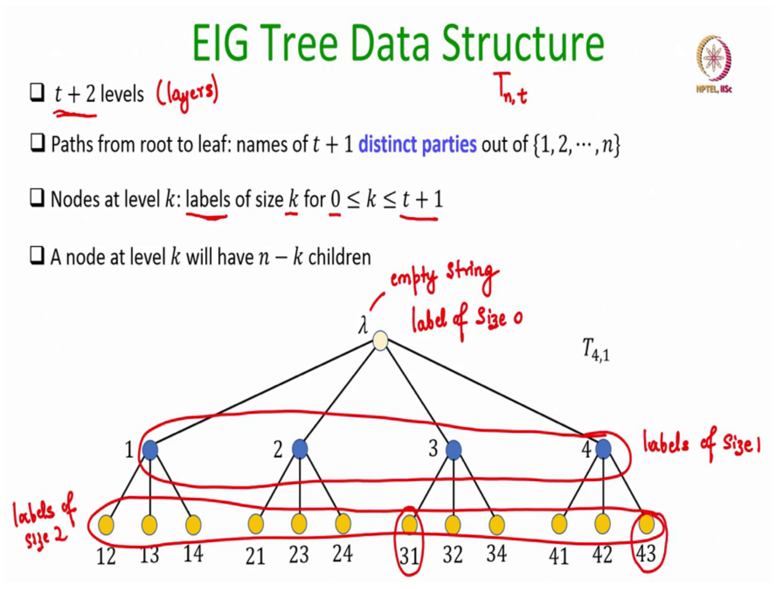


EIG Tree Data Structure

- ❑ $t + 2$ levels (layers) $T_{n,t}$
- ❑ Paths from root to leaf: names of $t + 1$ distinct parties out of $\{1, 2, \cdots, n\}$
- ❑ Nodes at level k: labels of size k for $0 \leq k \leq t + 1$
- ❑ A node at level k will have $n - k$ children

Now, there will be nodes at each layer right. So, if I consider a node at layer k or level k it will have $n - k$ children's ok. So, for instance if k is equal to 0 then this is root and it will have $n - k$, namely 4 number of children's 1 2 3 4. Now if I consider this node this node has, this node is at layer 1 level k equal to 1. So, it will have $n - k$ namely 3 number of children's and so on ok. Now, that is the number of children's that we will have at each level or each layer. Now each node will have some label ok.

(Refer Slide Time: 11:50)



EIG Tree Data Structure

- ❑ $t + 2$ levels (layers) $T_{n,t}$
- ❑ Paths from root to leaf: names of $t + 1$ distinct parties out of $\{1, 2, \cdots, n\}$
- ❑ Nodes at level k: labels of size k for $0 \leq k \leq t + 1$
- ❑ A node at level k will have $n - k$ children

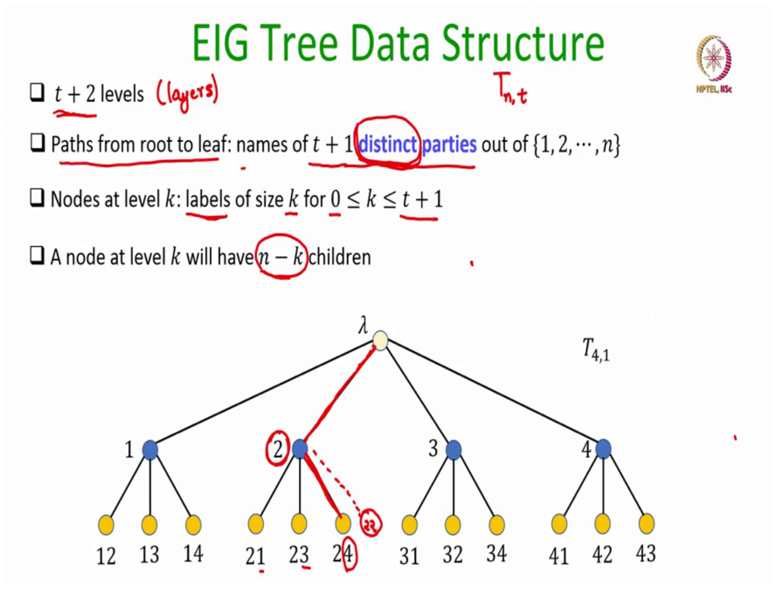
We will see the semantics of the label later on, but the labels will be of size k for all the nodes at level k ok. So, since we have t plus 2 levels starting from 0 to t plus 1, the root will have label of size 0.

So, that is why we are attaching an empty string as a label for the root node. Now, all the nodes at level 1 they have labels of size 1, all the nodes at level 2 layer 2 they have labels of size 2 and so on. So, for instance the label or the tag you can imagine label is nothing but some kind of tag, the tag associated with this node is 3 1, the string 3 1.

Whereas the tag associated with this node is the string 4 3 ok. So, that is about the label size ok. Now, how do we number the labels here? How do we assign the labels here? So, if you see the paths from root to leaf if I follow any path from root to any leaf right. It should be the case that the index or the identity of t plus 1 distinct parties should appear there.

That is the way we have arranged the tags for each node. So, we assume that we have n parties and their identifiers are 1 2 up to n and if I follow any path from the root to leaf the names of t plus 1 distinct parties should appear in that path.

(Refer Slide Time: 13:54)



EIG Tree Data Structure

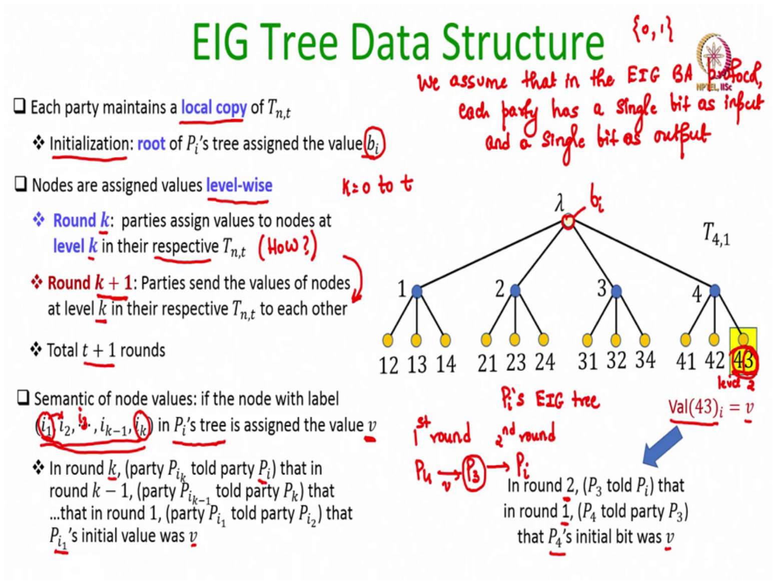
- ❑ $t+2$ levels (layers)
- ❑ Paths from root to leaf: names of $t+1$ distinct parties out of $\{1,2,\cdots,n\}$
- ❑ Nodes at level k: labels of size k for $0 \le k \le t+1$
- ❑ A node at level k will have $n-k$ children

So, for instance if I take this path. I go from say root, so you follow this highlighted path. Now I go from root to this node and then from that node I go to this node. So, what are the indices of the parties which appear along this highlighted path. So, we have node with index 2, followed by a node with index 4. So, we have party number 2 appearing in this path and party number 4 appearing in this path.

So, that is why in the tree; under the node with tag 2, we can only have children's where the tags could end with either 1 or 3 or 4, because we require the distinct parties to appear. So, we cannot have a child of 2 with tag 2 2, because that violates this requirement that the names of t plus 1 distinct parties should appear from the root to leaf ok.

And that is why as we go from one layer to the subsequent layer the number of children's decreases. Because when we are at layer k; that means, that we have already come from the root to k up to depth k; that means, we have already encountered the identities of k distinct parties. So, for any candidate children we have only n minus k identities left to assign as a particular tag or a particular label. So, that is why at level k a node will have only $n-k$ childrens because of this, distinct parties rule. So, that is the EIG data structure

(Refer Slide Time: 16:13)



EIG Tree Data Structure

{0,1}

We assume that in the EIG BA protocol, each party has a single bit as input and a single bit as output

- Each party maintains a local copy of $T_{n,t}$
  - Initialization: root of P_i's tree assigned the value b_i
- Nodes are assigned values level-wise

  $k = 0$ to t
  - Round k: parties assign values to nodes at level k in their respective $T_{n,t}$ (How ?)
  - Round $k+1$: Parties send the values of nodes at level k in their respective $T_{n,t}$ to each other
  - Total $t+1$ rounds
- Semantic of node values: if the node with label $(i_1, i_2, i_3, \ldots, i_{k-1}, i_k)$ in P_i's tree is assigned the value v
  - In round k, (party P_{i_k} told party P_i) that in round $k-1$, (party $P_{i_{k-1}}$ told party P_k) that …that in round 1, (party P_{i_1} told party P_{i_2}) that P_{i_1}'s initial value was v

In round 2, (P_3 told P_i) that in round 1, (P_4 told party P_3) that P_4's initial bit was v

So, now we will focus on the semantics of the tags associated with each node or the label associated with each node what exactly it signifies and how exactly the parties maintain this tree, what values are assigned to the nodes and so on. So, in the EIG protocol which we are going to see soon each party will maintain its own copy of the tree ok and it will try to assign values to each node in the tree. How the values are assigned? So, we will have the initialization process where in the ith party's copy of the tree, the root is assigned the value b_i.

So, we assume that in EIG BA protocol each party has a single bit as input and a single bit as output ok. So, the input of the ith party is b_i, which could be either 0 or 1 which is a private input known only to the ith party. Later on we will see protocols, where the input of the party could be any binary string, but right now we are making the things very simple.

We assume that the domain over which the EIG protocol is executed is the set $\{0, 1\}$ and each party will have an input from this set and an output from this set. So, as part of the initialization each party will assign the value b_i to the root node. And since the input b_i is available only with party P_i, no party will be knowing the value of the root node in other parties copy of the EIG tree.

Now the protocol proceeds and it will take total $t + 1$ communication rounds and what the parties will do is, they will exchange information round by round. And in each round what they are going to do is, they are going to assign value layer by layer to their respective copies of the EIG tree. So, for $k = 1$ to $t + 1$, the parties are going to do the following.

In round k where k could be any round in between 1 to t plus 1, the parties to the following. They assign values to the nodes at level k in their respective copies of the tree, how? We are going to see that very soon right. So, how basically that is determined by the next step? In round k plus 1 parties send the value of the nodes at level k in their respective trees in their respective copies of the tree at level k to every other node ok. They basically exchange the value of the nodes in their respective copies of the tree at level k and those values are exchanged in round number k plus one.

So, I should rather call it here K equal to 0 to t, so there are total $t + 1$ rounds. So, they have already done the initialization and now in round number 1, what they will do is each party will send the value of its root node to every other party, which will be obtained by the end of round 0. And then in round 1 what they are going to do is they will assign values to the node at the next layer depending upon whatever values they have received in the previous round.

And then again they will exchange the value of those nodes in that round, which will be received at the end of that round and will be processed in the next round and so on ok. How exactly that exchange of information happens? And what values are exactly assigned to the respective nodes? We will see soon that. But that is the way protocol operates ok.

So, now let us try to understand the semantic of node values in a tree. So, whatever I am explaining here is with respect to the ith party's tree. So, you can imagine that the same holds for every other party's copy of the EIG trees. So, imagine if there is a node with this tag. It has a tag, it has a label of size k, where it has k distinct identities i_1, i_2, i_3, i_{k-1}, i_k.

And suppose if the ith party assigns the bit v to that node in its EIG tree, what does that signify? It signifies the following it signifies that the starting party P_{i_1}'s initial value was v, which it reported to the next party namely, P_{i_2} in round number 1, and this information was communicated by party P_{i_2} in the next round to the party appearing in the label here. And continuing like that in round k the party with index i_k namely the last index has told party P_i about this.

So, it is like a chain of information which originated at round number 1 with the party P_{i_1}, who actually communicated its initial value to some node, who then talk about that to some other party in the next round, that P_{i_1} has communicated this value to me in the previous round and this chain of information then it comes to the ith party in round number k.

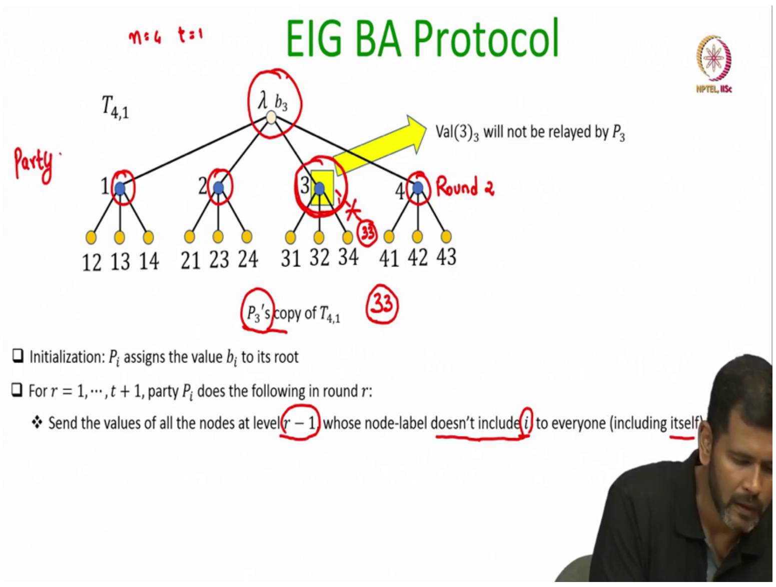
Then party P i assigns the value v to the node with this tag. So, for instance P_i assigns a value v to the node with tag 43, then I write $\text{Val}(43)_i = v$.

And this signifies that in round 2, why round 2? Because this node is at level 2. At round 2, P3 namely the party who is occurring here at the last position has told Pi, that in the previous round, namely round number 1, the previous party in the layer in the label which is P4, in this case has told P3 that P 4's initial value was bit v. So, what has happened here in that in first round P4 would have told P3 that P4's initial input for the byzantine agreement protocol is v.

And P3 communicates this information to Pi in the second round that hey, P4 has told me about this in the previous round then the node with tag 43 will be assigned the value v in the ith parties EIG tree. Now, notice that if party P 3 is corrupt then it might report an incorrect value of P4's input to Pi. So, when P3 is telling Pi that P4 has told P3 that its input was v.

It is not necessary the case that P3 is providing the right information regarding P4's input to Pi, it might be wrong as well. But Pi would not be knowing whether P3 is corrupt or not, whatever value P3 is reporting to Pi, that value will be assigned to this node in Pi's EIG tree. That is the semantic of the labels associated with the node.

(Refer Slide Time: 27:04)



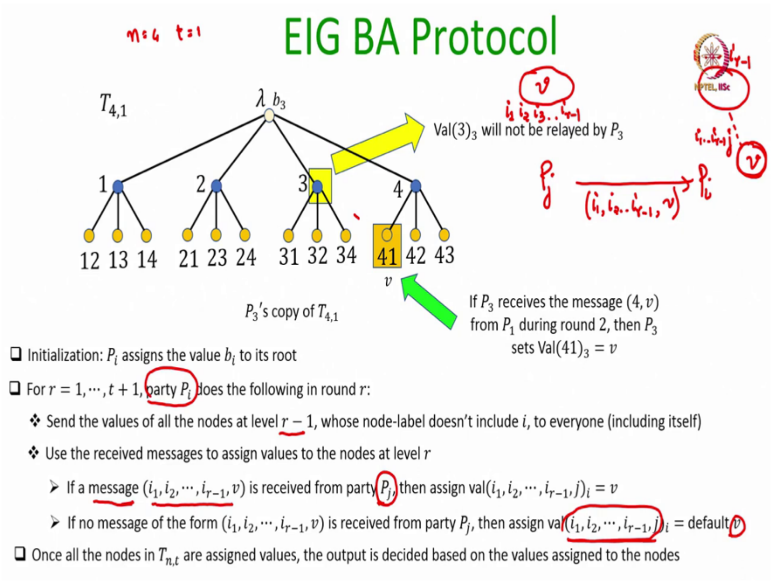
EIG BA Protocol

- Initialization: P_i assigns the value b_i to its root
- For $r = 1, \cdots, t+1$, party P_i does the following in round r:
  - Send the values of all the nodes at level $r-1$ whose node-label doesn't include i to everyone (including itself)

So, now let us see how the protocol assigns the values to the various nodes. So, as I said the initialization is very simple. Again I am explaining the process with respect to P3’s copy of the EIG tree, assuming $n = 4$ and $t = 1$. So, P3 will first assign the value b3 to its root node. And for $t + 1$ rounds, Pi will do the following in round r.

So, in round number r the party Pi will send the values of all the nodes in its copy of the EIG tree at the layer r – 1, provided the tag for those node does not include the identity i, to everyone else. So, when I am saying that it sends the value of all the nodes to everyone else including itself. So, remember that at each level there could be more than one node. So, the values of all the nodes in P i's copy of the EIG tree will be communicated except for those nodes whose label include i ok.

(Refer Slide Time: 31:01)



n=4 t=1
$T_{4,1}$
λ b_3
1
2
3
4
12 13 14 21 23 24 31 32 34 41 42 43
v
P_3's copy of $T_{4,1}$
$i_1 i_2 i_3 .. i_{r-1}$
v
$i_1 .. i_{r-1} j$
v
P_j
P_i
$(i_1, i_2 .. i_{r-1}, v)$

EIG BA Protocol

Val(3)$_3$ will not be relayed by P_3

If P_3 receives the message $(4, v)$ from P_1 during round 2, then P_3 sets Val(41)$_3 = v$

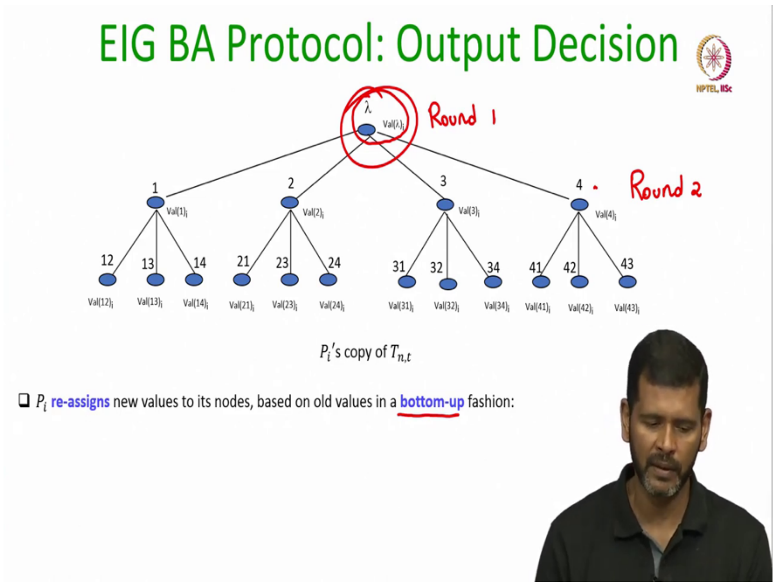
- Initialization: P_i assigns the value b_i to its root
- For $r = 1, \cdots, t+1$, party P_i does the following in round r:
  - Send the values of all the nodes at level $r-1$, whose node-label doesn't include i, to everyone (including itself)
  - Use the received messages to assign values to the nodes at level r
    - If a message $(i_1, i_2, \cdots, i_{r-1}, v)$ is received from party P_j, then assign val$(i_1, i_2, \cdots, i_{r-1}, j)_i = v$
    - If no message of the form $(i_1, i_2, \cdots, i_{r-1}, v)$ is received from party P_j, then assign val$(i_1, i_2, \cdots, i_{r-1}, j)_i$ = default v
- Once all the nodes in $T_{n,t}$ are assigned values, the output is decided based on the values assigned to the nodes

Now, what does the parties do based on the values that they receive in round r? So, remember when I say each party is doing this step; that means, party i will be receiving values from the other nodes which they are communicating right, they. So, every other party Pj would be sending to Pi the value of P j's nodes at layer $r - 1$, which will be received by Pi at the end of round r, which it will use to assign values to the children at layer number r ok.

So, for instance, if a message of this form is received from Pj by Pi. So, there is P j ok who has sent the following information, Pj is claiming that there was a node which was assigned the value v in its EIG tree which was at layer r minus 1 and whose tag was i1, i2, i3 up to i(r-1). So, if P j says that ok the value of the node in my EIG's tree which has the tag i1, i(r-1) is v, then P i uses that information to do the following, so Pi will also have an EIG tree. In its local copy of the EIG tree, there will be a node which has the tag i1, i2, up to i(r-1).

And under that there must be a child node whose tag will be i1, i2, i(r-1) followed by j, to that node Pi will assign the value v. However, if this message which is expected from Pj does not arrive to Pi at the end of round r, and remember i will be knowing that round r starts at this clock time and ends at this clock time. So, within this much clock time this expected message from Pj is supposed to be received by Pi. But if no such message comes then it assumes that ok Pj would send me this default value. So, default value is taken as some bit v and that is used by P i to assign or decorate the node with the tag i1, i2, i sub(r-1) j ok. Now once all the nodes in P i's EIG tree is assigned values the protocol ends and then Pi is going to locally decide an outcome for the protocol.

(Refer Slide Time: 34:20)



EIG BA Protocol: Output Decision

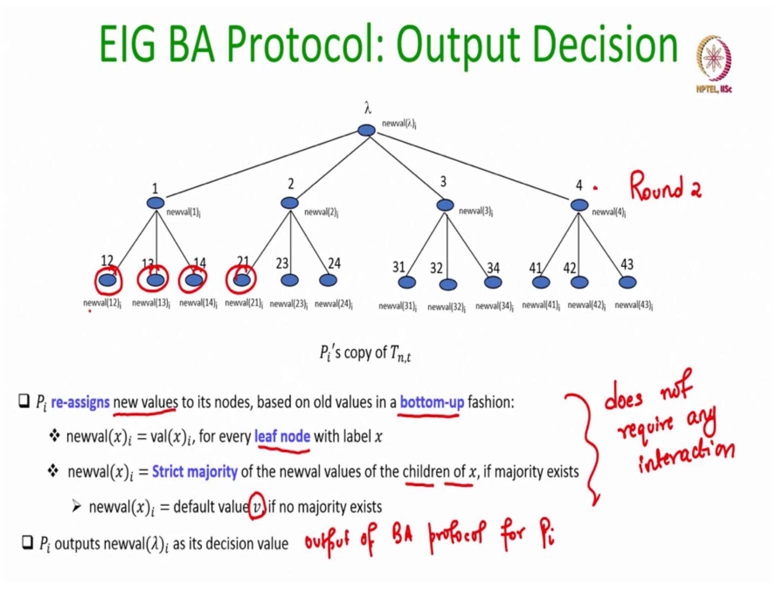
P_i's copy of $T_{n,t}$

- P_i **re-assigns** new values to its nodes, based on old values in a **bottom-up** fashion:

So, let us see the output decision how the output is decided. So, assume that in round number 1 initial values are communicated in round number 2, the values at layer one are communicated and then all the values are all the nodes are assigned in Pi's copy of EIG tree. Now what Pi does is, it reassigns new values to its nodes based on the values that it has received in a bottom of fashion to come up with a new value for the root node in its EIG tree.

So, remember the old value of the root node is Pi's own input, namely the input for the BA protocol. But that is not going to be the output for the BA protocol, the output is going to be decided in a different way. So, what Pi does is, this process of re assigning new values is done in a bottom up fashion. So, the new values for the leaf nodes is going to be whatever it was earlier ok.

(Refer Slide Time: 35:35)


λ
newval(λ)$_i$
1
newval(1)$_i$
2
newval(2)$_i$
3
newval(3)$_i$
4
newval(4)$_i$
Round 2
12
13
14
21
23
24
31
32
34
41
42
43
newval(12)$_i$
newval(13)$_i$
newval(14)$_i$
newval(21)$_i$
newval(23)$_i$
newval(24)$_i$
newval(31)$_i$
newval(32)$_i$
newval(34)$_i$
newval(41)$_i$
newval(42)$_i$
newval(43)$_i$
P_i's copy of $T_{n,t}$
NPTEL, IISc

EIG BA Protocol: Output Decision

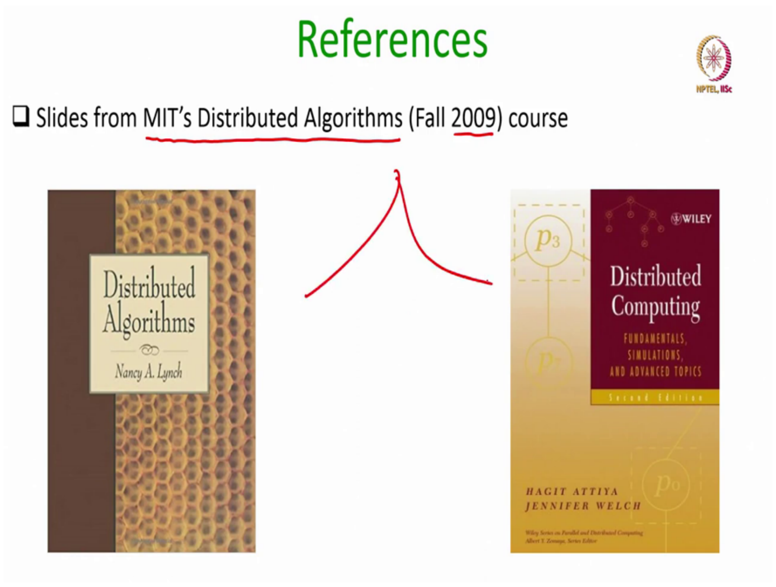
- P_i **re-assigns** new values to its nodes, based on old values in a **bottom-up** fashion:
  - newval$(x)_i$ = val$(x)_i$, for every **leaf node** with label x
  - newval$(x)_i$ = **Strict majority** of the newval values of the children of x, if majority exists
    - newval$(x)_i$ = default value v if no majority exists

  does not require any interaction
- P_i outputs newval$(\lambda)_i$ as its decision value output of BA protocol for Pi

So, whatever was the value of this leafs node they will be retained as it is as their new Val values, they would not change. But now as we go to the higher level from the bottom the new values are taken to be the strict majority of the new values of the children's of that particular node, if at all majority exist. It could be the possibility majority does not exist, if the majority does not exist then the new value is taken to be a default value. And then we will go to a one layer up and keep on doing this process till a new value has been computed for the root node ok.

And remember this is done locally this process does not require any interaction ok, because now all the nodes in Pi's EIG tree has have been populated. Now once the new value for the root node is computed that is taken to be the output value, that is the output of BA protocol for Pi ok. That is the protocol, I hope it is clear. In the next lecture we will analyze this protocol, we will demonstrate the protocol and in the subsequent lectures we will analyze the security properties.

(Refer Slide Time: 37:15)



References

❑ Slides from MIT's Distributed Algorithms (Fall 2009) course

Distributed Algorithms
Nancy A. Lynch
WILEY
Distributed Computing
FUNDAMENTALS, SIMULATIONS, AND ADVANCED TOPICS
Second Edition
HAGIT ATTIYA
JENNIFER WELCH

So, the references which are used for demonstrating the EIG protocol are the following. So, I have used the slides from MIT's course on Distributed Algorithms which is slightly an old course, but very informative. And of course, you can find the full details of EIG BA protocol in either of these textbooks.

Thank you.