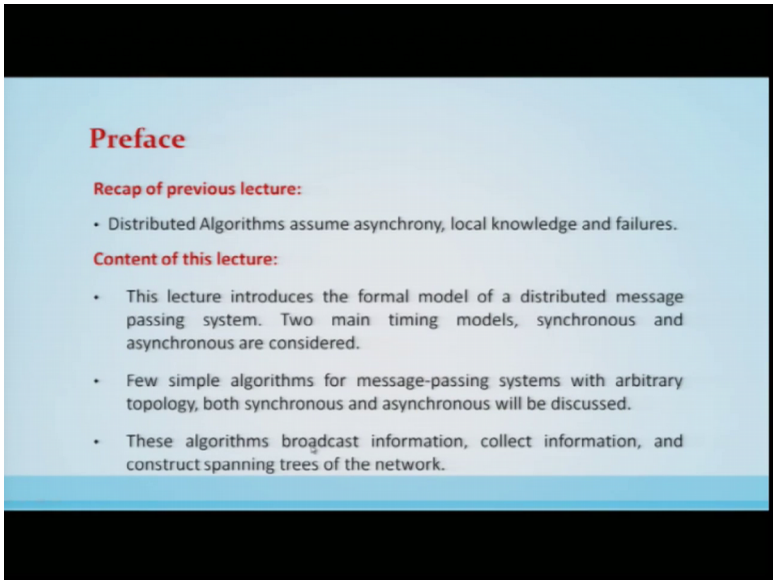
Distributed Systems
Dr. Rajiv Misra
Department of Computer Science and Engineering
Indian Institute of Technology, Patna

Lecture – 02
Basic Algorithms in Message Passing Systems

Lecture 2 is based on the basic algorithms on a message passing system.

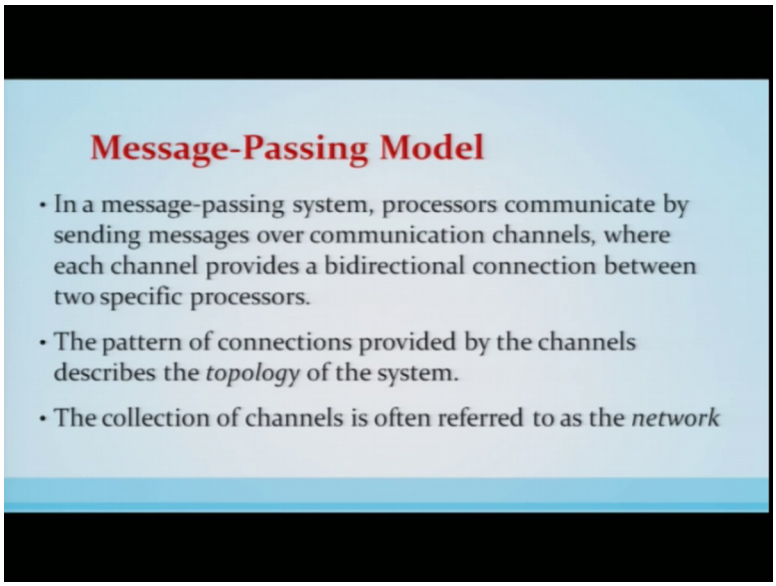
(Refer Slide Time: 00:20)



So, preface recap of the previous lecture distributed, we have seen in the previous lecture that distributed algorithms assume a synchrony local knowledge and the failures in this particular lecture, we are going to further on discuss these 3 issues; how using these 3 different constraints we are going to design the distributed algorithms. So, content of this lecture would be that the formal model of the distributed message passing system will be used to deal with this asynchrony and a local knowledge 2 main timing models which we are going to cover up is synchronous and asynchronous and few simple algorithms for message passing systems with arbitrary topology with both synchronous and asynchronous model we will be discussed.

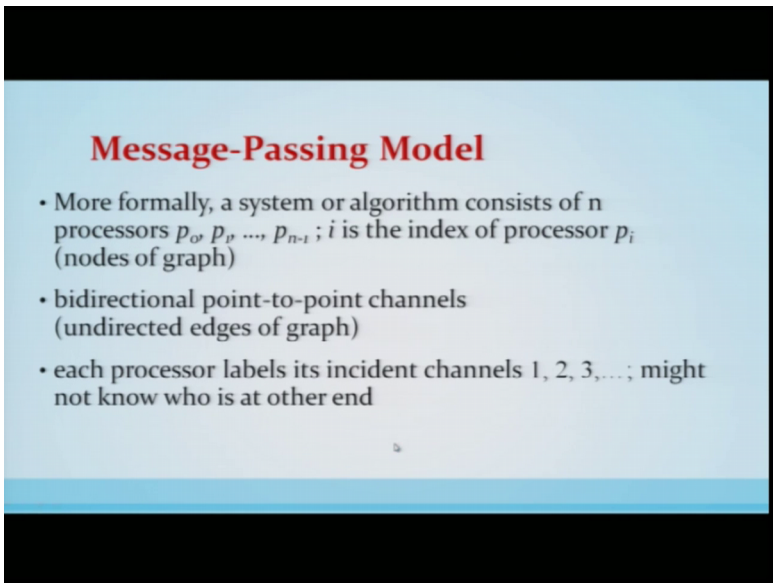
These algorithms which we are going to cover today will do the broadcast of information in the network they will collect the information and they will construct the spanning tree of the network. So, today's lecture will become the basic building blocks of distributed algorithms message passing model.

(Refer Slide Time: 01:33)



So, in a message passing model processors communicate by sending the messages over the communication channel where each channel provides a bidirectional communication between 2 processors. So, the pattern of connection provided by the channels describes the topology of the system the collection of the channel is referred to as the network.

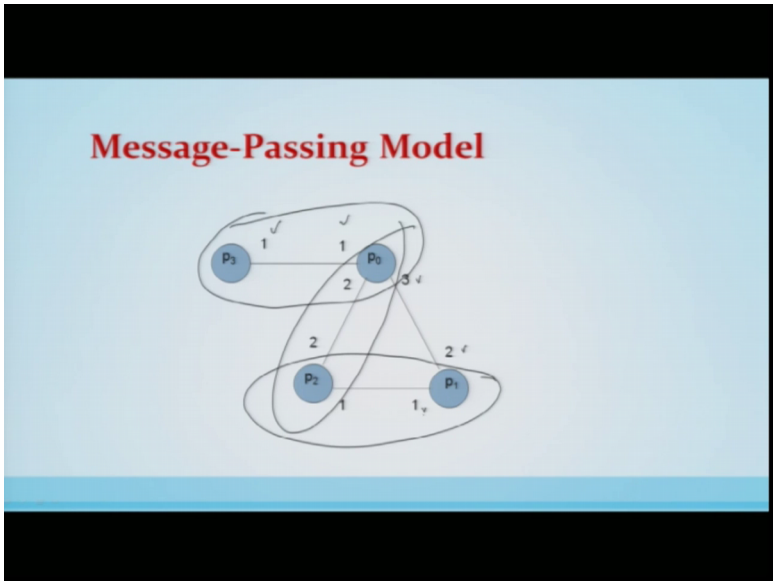
(Refer Slide Time: 01:55)



So, the under the message passing model the algorithms is basically consists of n different processors P 0 to P n minus 1 and this processors are indexed by i and they become the node of a graph in the topology which we are going to discuss now the next

component here in the algorithm is the bidirectional point to point channels. They are undirected edges of the topology graph each processor a labels its incident channels with these numbers 1, 2 and so on and this particular number is nothing, but the degree of the node. So, this particular each processor do not know who is at the other end because of the local knowledge.

(Refer Slide Time: 02:57)

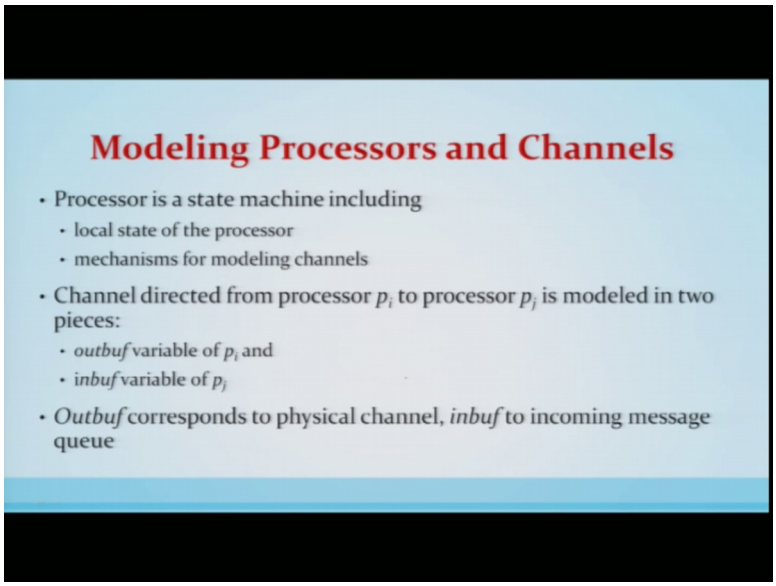


So, this particular model is assumed and this can be explained using this particular diagram.

Here you can see the node P 1 the 2 channels it has numbered as 1 and 2, similarly this particular channel which is connecting between P 0 and P 1 and P 0 is basically numbering the same channel as 3. So, basically because P 0 is not knowing what is basically the a channel number is assigned by P 1. So, it is only based on the local knowledge and this will form a message passing model. So, here you can see that in this particular topology P 0 and P 3 this particular channel both P 0 and P 3 they are giving the channel id as one similarly this particular channel which is connecting P 0 and P 2 both P 0 and a P 2 incidentally they have given the same number.

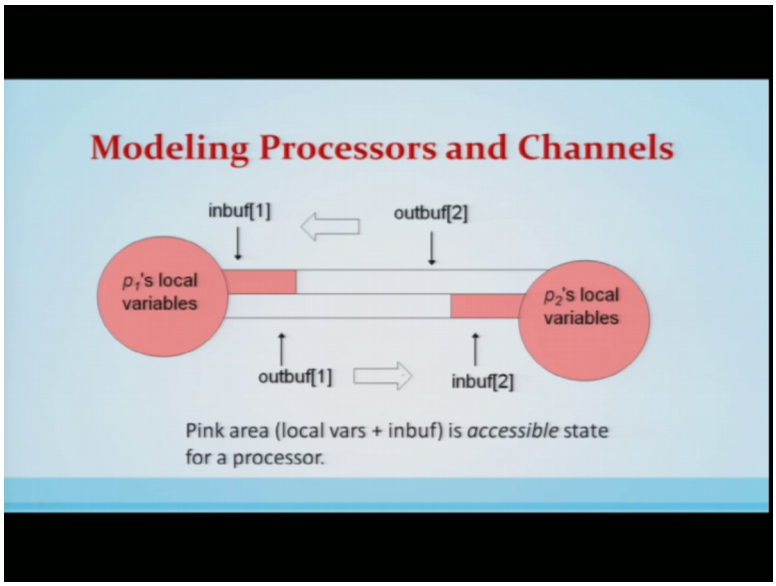
Similarly, the channel which is connecting P 1 and P 2 both P 1 and P 2; they are giving the same numbers.

(Refer Slide Time: 04:03)



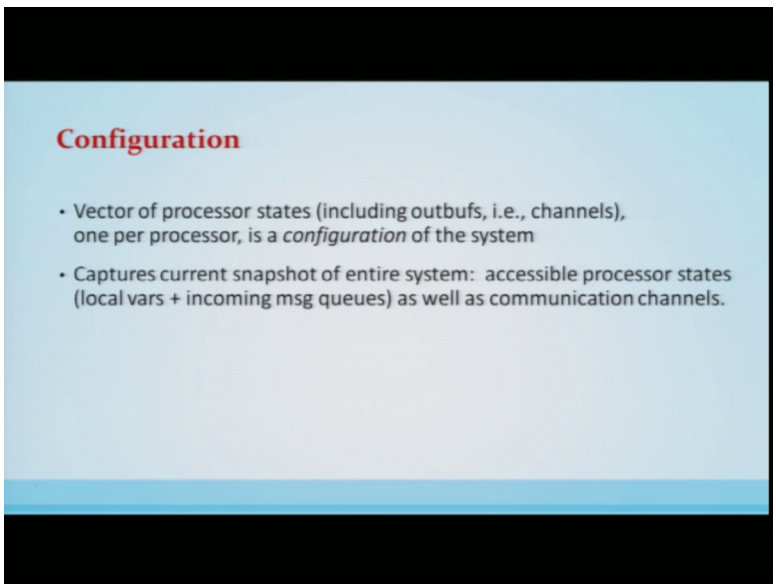
Now, modeling processors and the channels; so, the processor is a state machine including the local state of a processor mechanisms for modeling channels and this basically will be the processor. So, the channel is the directed from processor Pi to Pj is modeled into 2 pieces the first one is called outbuf that is the buffer variable of P 1 and inbuf variable of Pj. So, every channel has basically which is a connection of 2 processor Pi and Pj will require these 2 buffers. So, outbuf corresponds to the physical channel; that means, the messages which are sent on the channel, but not yet delivered that will be there in outbuf and inbuf is refers to the buffer to the incoming message queue and it is basically associated with the processors.

(Refer Slide Time: 05:07)



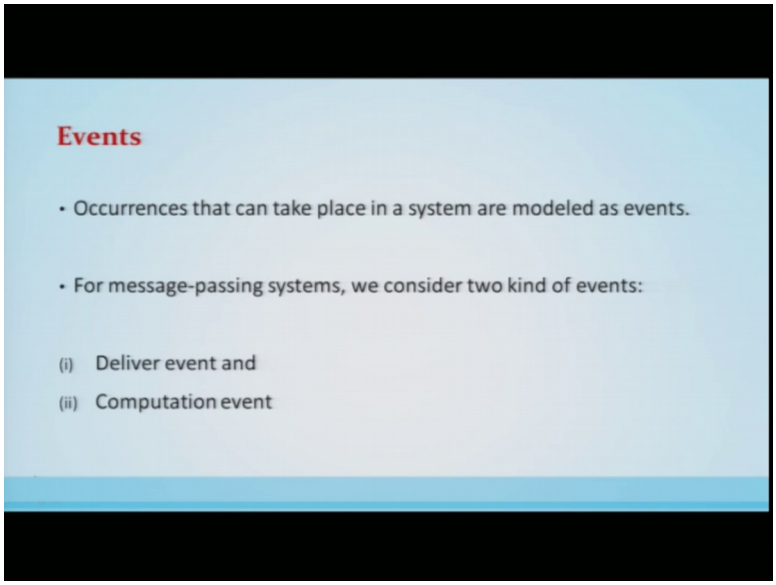
So, this particular connection of these 2 variables inbuf and outbuf can be explained using this particular diagram. So, here we can model the processor and channels like this. So, here you can see that P 1s local variable and its inbuf which is denoted using the pink color that is local variables and inbuf is an accessible state for a particular processor similarly P 2 is also having its own local variable and its inbuf which is also denoting P twos that is basically denoted with the index 2. So, that is P twos local variables and the white channels are basically indicating the outbuf that is nothing, but the communication channel where the messages are being sent.

(Refer Slide Time: 06:00)



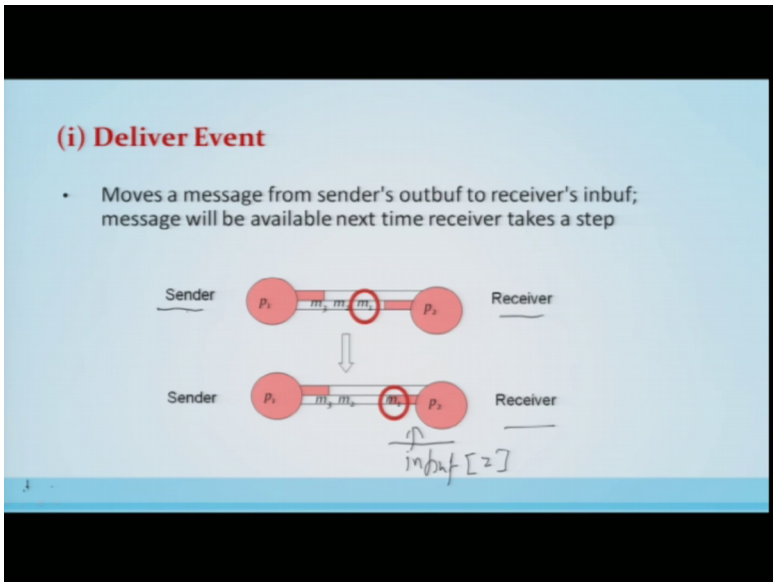
The configuration the vector of processor states including outbufs that is the channels one per processor is the configuration of a system captures this particular configuration captures the current a snapshot of the entire system that is accessible processor states local variables and incoming message queues as well as the communication channel is basically the configuration.

(Refer Slide Time: 06:24)



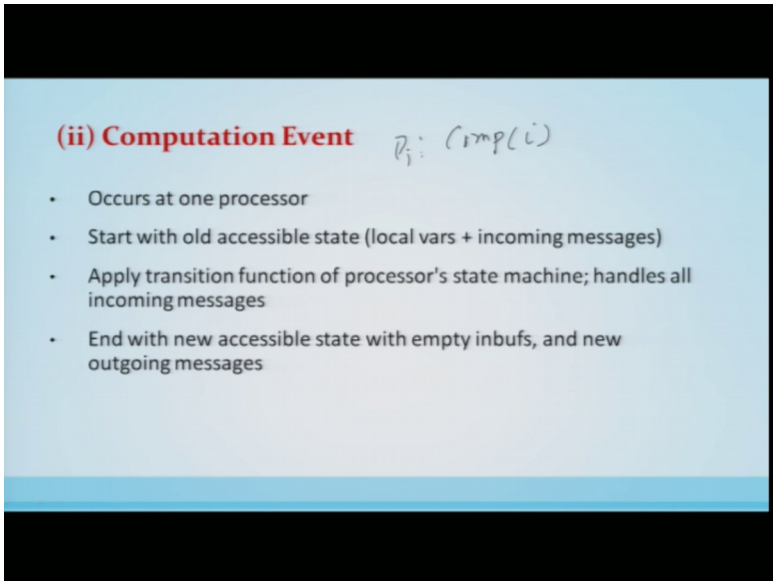
The next component here in this particular message passing model is called events. So, occurrence 2 types of event occur in the system, they are basically deliver events and computation events.

(Refer Slide Time: 06:46)



So, deliver event in a message passing system denotes that it moves a message from sender's outbuf to a receiver's inbuf that is the message will be available next time receiver takes a step. So, this particular deliver event can be understood by this illustrative diagram here the sender P 1 is having the in buff and the message m 1 let us say that sender P 1 want to send to the receiver P 2. So, you see next time in the step. So, m 1 will be delivered to the inbuf of P 2.

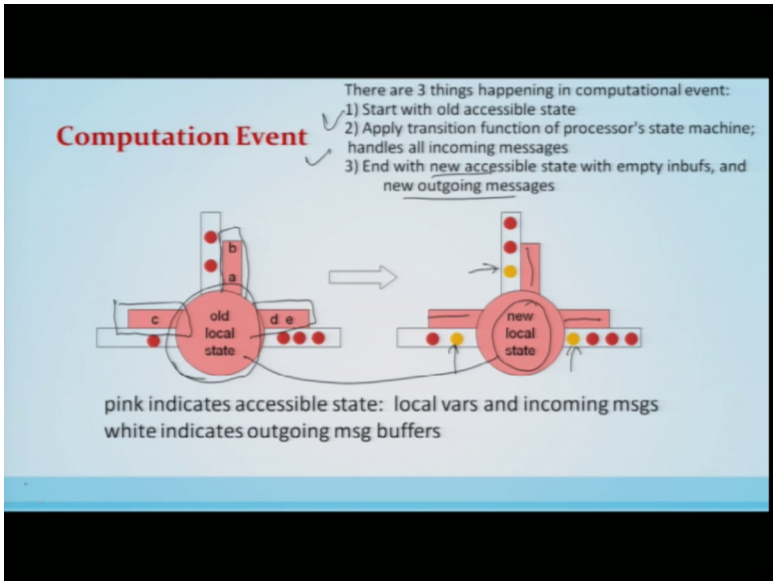
(Refer Slide Time: 07:38)



So, from sender to receiver, the message will be delivered using this particular event called deliver event; the next event which is possible in a message passing system is called computation event.

So, computation event is represented by computation compi will occur at one processor that is why for a particular processor Pi it is denoted as compi occurs at one process we start with the old accessible state which is nothing, but a local variables and the incoming messages apply the transition function of a processors state machine handles all the incoming messages now end with the new accessible state with an empty inbox and the new outgoing messages this computation event can be understood using this particular illustrative diagram.

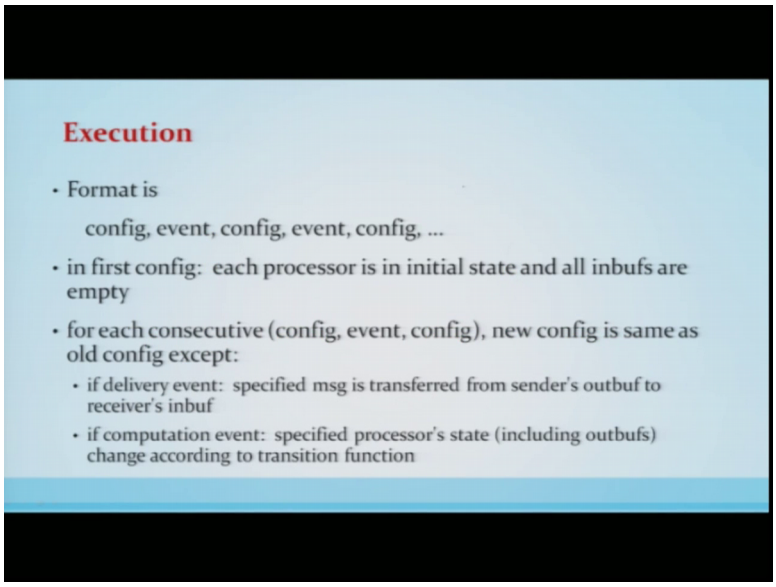
(Refer Slide Time: 08:23)



So, here you can see the this one the first state which is representing the old local state and its inbuf for this particular processors are having different messages in inbuf and for the 3 outgoing for the 3 channels associated with the 3 channels. So, computation event what it will do it will change the transition, it will make a transition in the states. So, it will it will provide you a new state and then basically you see that this particular associated inbuf will become empty and the new messages will be appearing in the outbuf of this particular state or that is associated with the channels of that particular process.

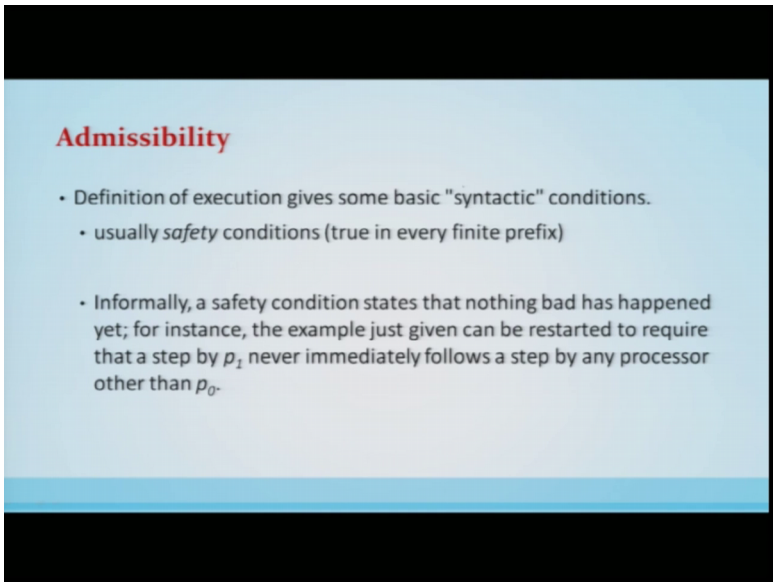
So, all 3 things will happen in the computation event. So, pink indicates the accessible states that are the local variables and incoming messages a white indicates the outgoing message buffers.

(Refer Slide Time: 09:40)



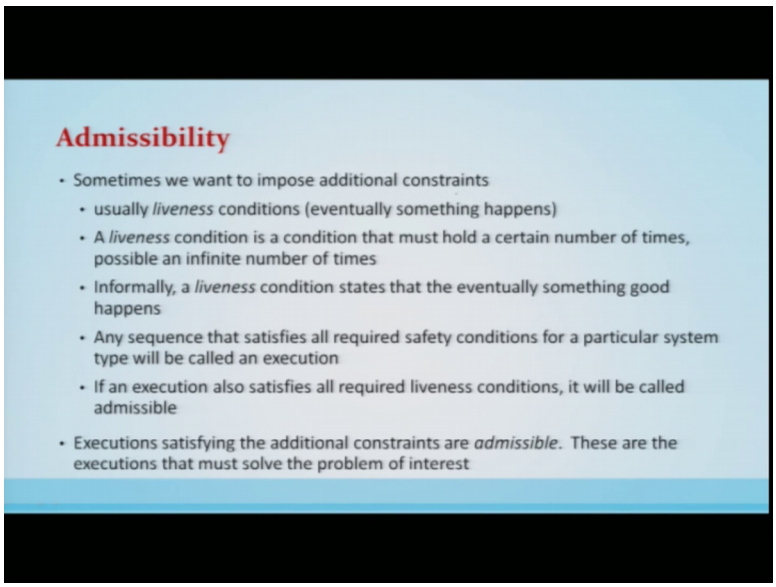
So, this is the computation event now the execution. So, execution in a in a message passing system is represented using a format which is alternating sequence of configuration event configuration event and so on. So, in the first configuration the processor is in the initial state and all the inbufs are empty for each consecutive this configuration event configuration, the new config is same as old except if the delivery event that is the specified message is transferred from senders outbuf to the receivers inbuf if the computation event happens then specified processors state change according to the transition function.

(Refer Slide Time: 10:21)



So, that is called execution now admissibility. So, the definition of the execution gives some basic syntactic conditions now usually safety conditions are specified as that nothing bad has happened yet and this particular safety condition ensures that the step P 1 never immediately follows a step by any other process other than P 0. So, P 1 is followed from P 0.

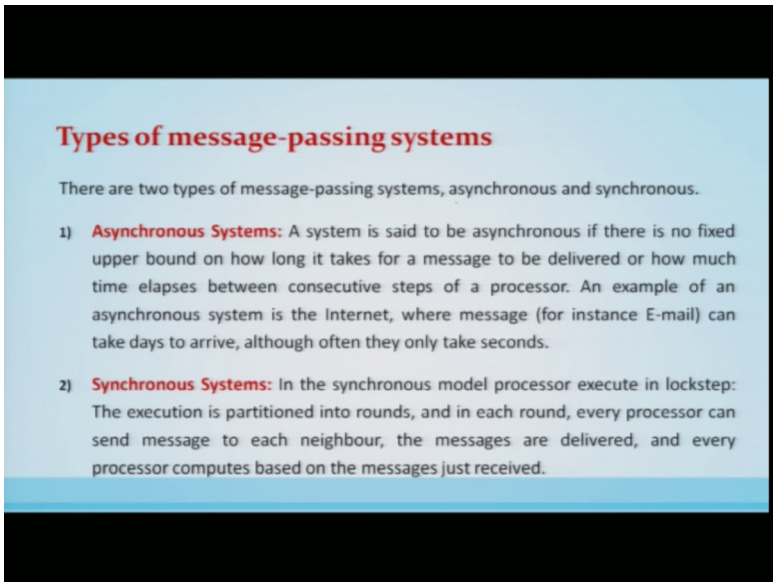
(Refer Slide Time: 10:56)



So, this will ensure a safety condition in the execution and further on we if we impose additional constraint which is called liveness conditions. So, liveness condition means eventually something good happens.

So, liveness condition is a condition that must hold a certain number of times possibly infinite number of times. So, eventually something good happens after that. So, any sequence that satisfies all the required safety conditions that we have discussed earlier for a particular system will be called an execution if an execution also satisfies the liveness condition that it is called an admissible. So, execution satisfying the additional constraints is admissible.

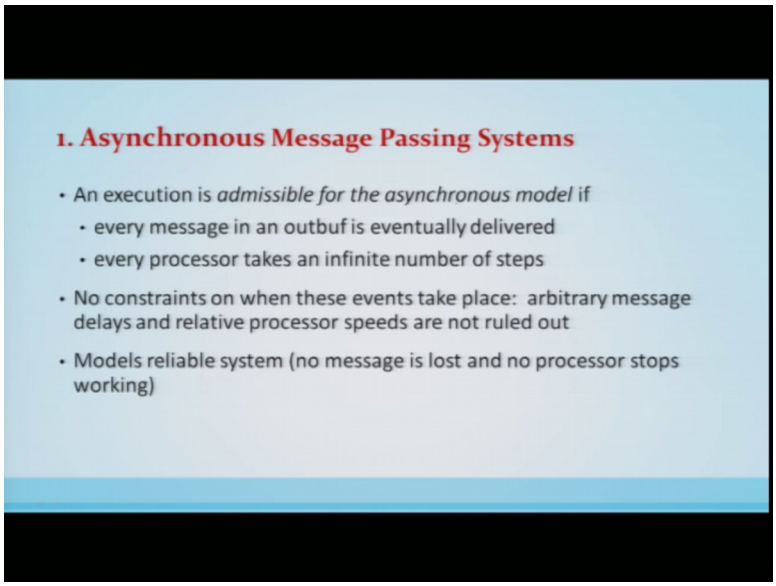
(Refer Slide Time: 11:39)



So, these are the executions that must solve the problem of interest now the type of message passing systems there are 2 types of message passing systems they are asynchronous and synchronous.

So, asynchronous system the system is said to be asynchronous if there is no fixed upper bound on how long it takes for a message to be delivered or how much time elapses between consecutive the steps of a processor so; that means, that means this asynchronous is basically dealing with the message delays and also the computation delays together and that is not known or unbounded sometimes. So, for example, an example of an asynchronous system is the internet where the message can take days to arrive, although often they may take seconds.

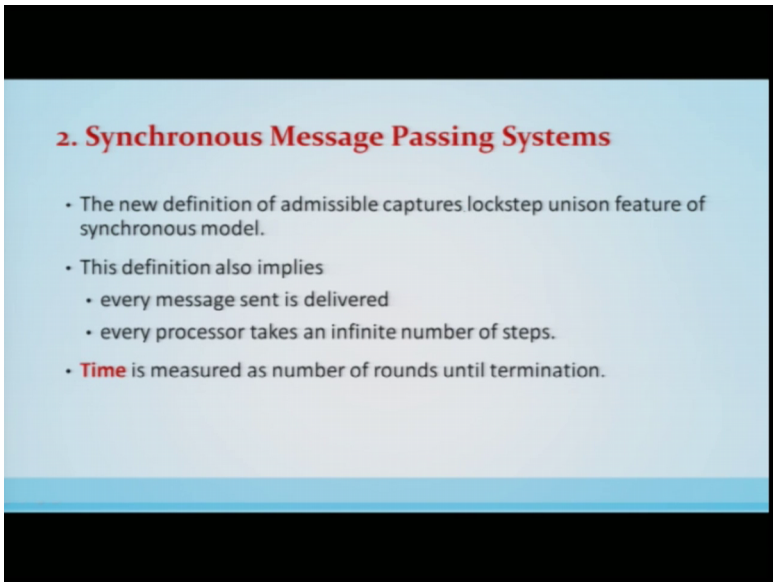
(Refer Slide Time: 12:46)



Now, the second type of system is called the synchronous system in synchronous model processors execute in locksteps, the execution is partitioned into the rounds and in each round each processor can send a message to each neighbor the messages are delivered and every processor computes based on the message just to received now asynchrony message passing systems asynchronous message passing systems is an execution is a admissible for asynchronous model if every message in the outbuf is eventually delivered and every processor takes infinite number of the steps no constraints on when these steps these events takes place arbitrary message delays and relative processor speeds are not ruled out.

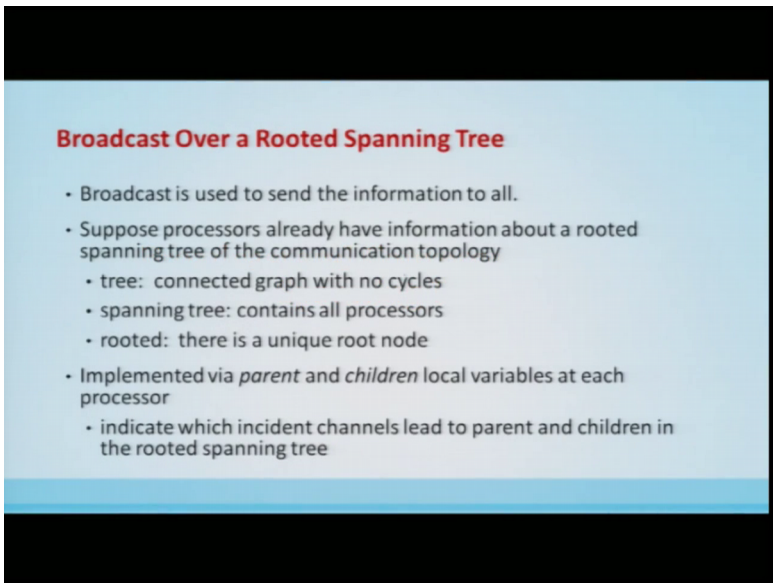
Models the reliable system that is no message is lost and no processor is stops. So, this is an assumption here when we cover up these 2 models for designing the distributed algorithms.

(Refer Slide Time: 13:24)



The second is the synchronous message passing system the new definition of admissible captures the lockstep unison features of a synchronous model this definition also implies every message sent is delivered every processor takes an infinite number of steps. So, time is measured as the number of rounds until the termination we have introduced the basic model for distributed algorithms which are based on synchronous and asynchronous model without any failure and using this particular model. Now we are going to discuss some of the basic algorithms and which is the basic building blocks.

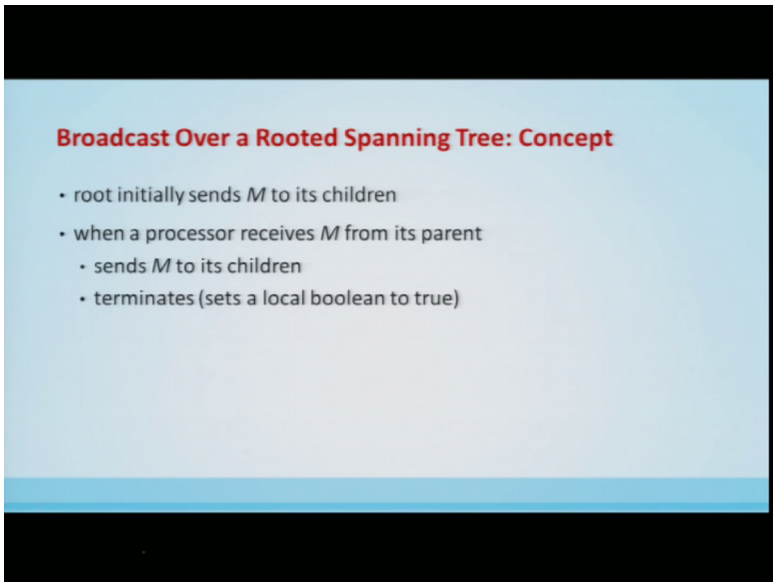
(Refer Slide Time: 14:18)



So, the first basic algorithm which we are going to cover up is about the broadcasting and the converge casting on a spanning tree. So, let us begin with this algorithm design broadcast over a rooted spanning tree now broadcast is used to send the information to call the nodes. Now suppose the processor already have the information about the rooted spanning tree of the communication topology. So, what is a spanning tree here tree is a connected graph with no cycles and that tree which can which is spans over all the processor is called the spanning tree and rooted means there is a unique root node of a tree is called a rooted spanning tree.

So, we assume in this particular algorithm a rooted spanning tree is given is already provided as the infrastructure facility and then we are going to introduce or we have to see the algorithm; that is called a broadcast. Now this particular rooted spanning tree can be implemented using 2 variables that is the parent and the child these are the local variables at each processor. So, this will indicate which incident channels lead to the parent and the children in the rooted spanning tree.

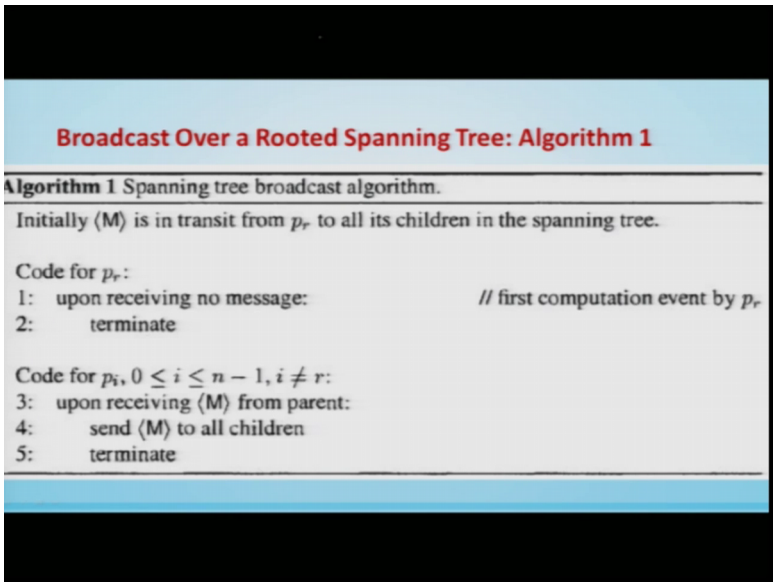
(Refer Slide Time: 15:27)



So, this particular; these 2 variables form the rooted spanning tree structure over the network.

Now, the algorithm; so, root initially sends message M to its children; now when the processor i receives M from its parent, then it will send M that is the message to its children and terminates; so, this will form the basic algorithm for broadcasting.

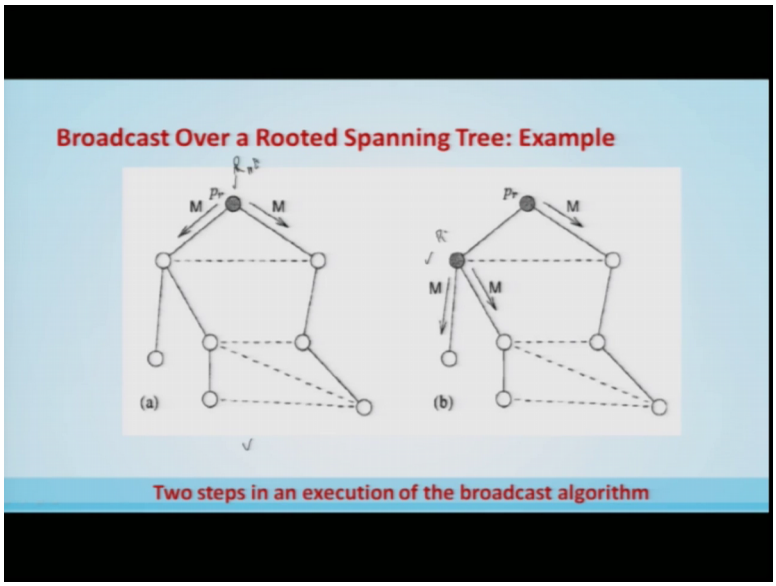
(Refer Slide Time: 15:55)



Let us understand here in this algorithm that is the algorithm one spanning tree broadcast algorithm initially m is a message is in transit from Pr Pr is a root from a processor which is required or which will initiate the broadcast to send the message to all its children in the spanning tree. So, the code for the root for that is Pr is goes like this to the steps step number 1 upon receiving no message, then basically it will terminate the code number or for the codes for the different processor other than other than root.

Upon receiving the M that is the message from the parent send m to all the children and then terminates.

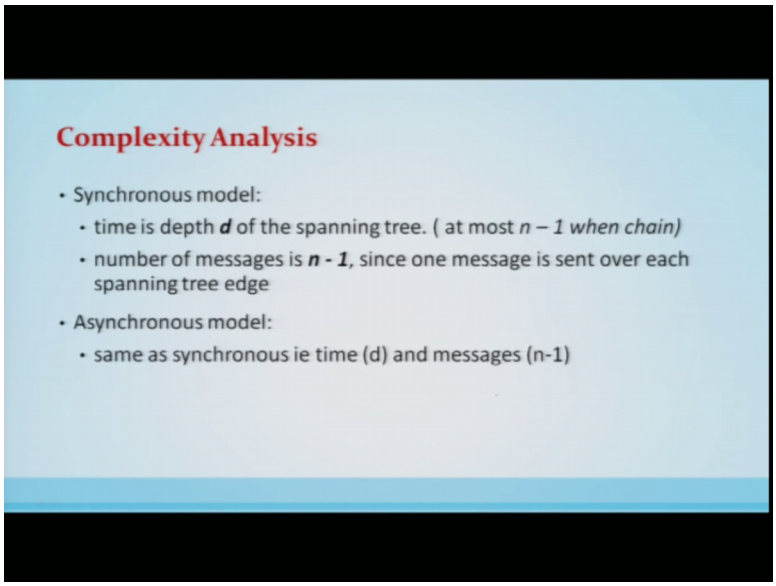
(Refer Slide Time: 16:54)



So, this will be the broadcast algorithm over a rooted spanning tree let us understand through this particular illustrative figure. So, here in the figure one you see that the root node pr will want to send the message m to all the other nodes. So, it will basically send to its outgoing channel because you see the structure is a spanning tree spanning tree is shown as the direct lines and dotted means the channels which are not in the spanning tree. So, M will be sent to the dark lines.

Now, after receiving after the m is received by the received node that is Pi will receive from its parent. So, then it will send to the other children which is shown as the dark nodes and so on. So, in this particular way, the steps will form and the message will eventually be delivered to all the nodes and this is called broadcasting.

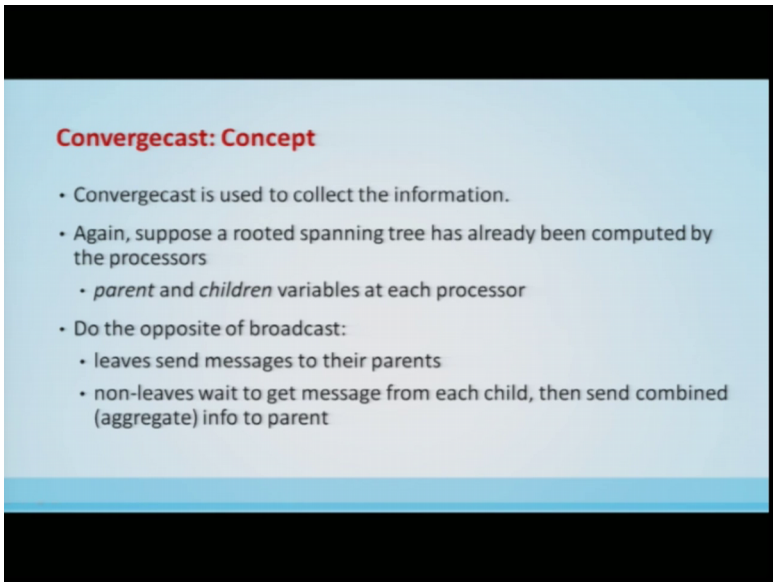
(Refer Slide Time: 17:57)



If we see the analysis that is the complexity analysis of this broadcast algorithm with on a rooted spanning tree if we consider the synchronous model then the time is the depth of a spanning tree; that means, how much time this particular algorithm is going to take to terminate at all the nodes is nothing, but the depth of a spanning tree.

The depth of a spanning tree can be at most n minus 1 when the topology is a chain. So, the number of messages here is n minus one by because only one message is sent over the each edge of a spanning tree and the total number of edges in any trees n minus 1. So, n minus 1 messages are required. Now asynchronous model also will have the same complexity that is the time is required is the depth d of a tree and the message is n minus 1.

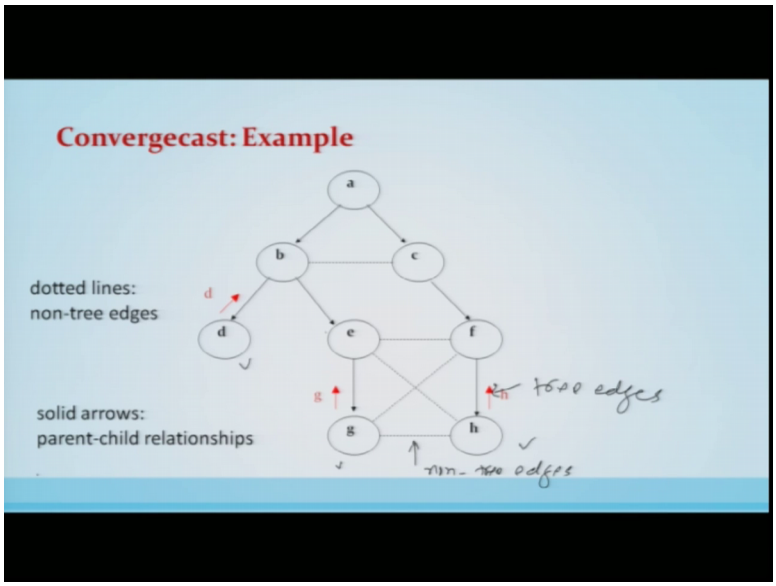
(Refer Slide Time: 18:56)



So, that was the simple algorithm for broadcasting. So, broadcasting algorithm is used in the distributed system or in a network to transmit the information to all the nodes.

The other simple algorithm which we are going to see is about the convergecast convergecast is an opposite of the broadcast. So, convergecast is used to collect the information again we assume a rooted spanning tree and where a rooted spanning tree is implemented using parent and children variable at each node now the leaves will send the information to their parents and a non leave will wait to get the message from each child.

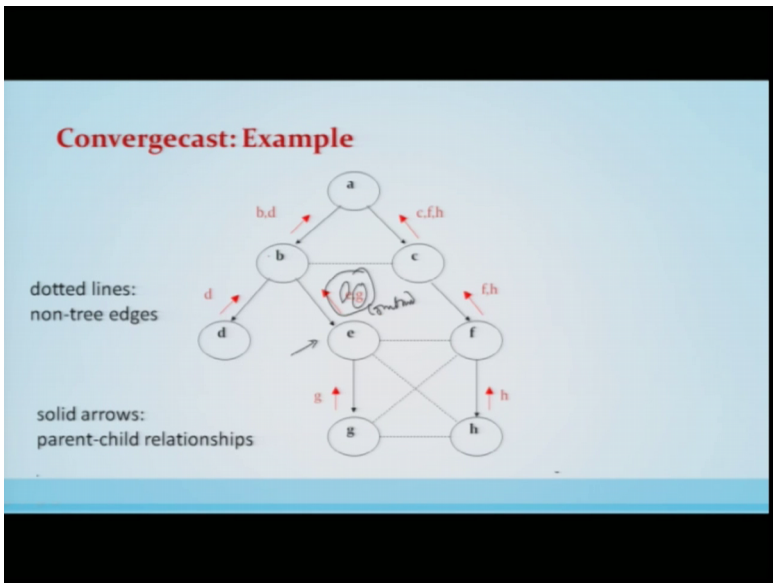
(Refer Slide Time: 19:47)



And then it will aggregate or a combined and then it will combine information will be transmitted to the parent take this particular example.

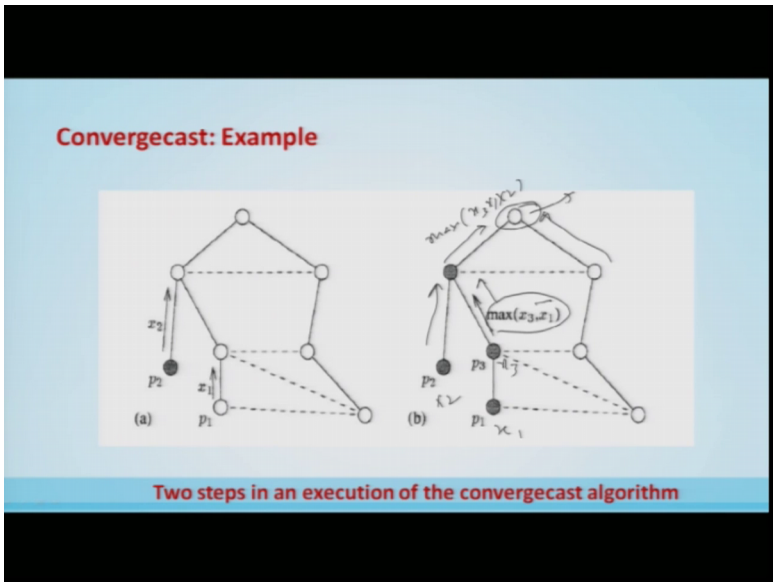
In this example again we follow the same convention the convention is that in this topology the dotted lines are non tree edges and the solid lines are basically the tree edges these are the tree edges. So, as you know the convergecast will start from the leaf nodes. So, all the leaf node will send the information to the parent like this you see the nodes d, g, h; they will send the information to their to their parent nodes.

(Refer Slide Time: 20:42)



And after receiving the information from all the childs; then it will aggregate and send the information to the parents. So, here you see; the node e will aggregate its information and the incoming information both are aggregated or combined and will send to its parent.

(Refer Slide Time: 21:10)

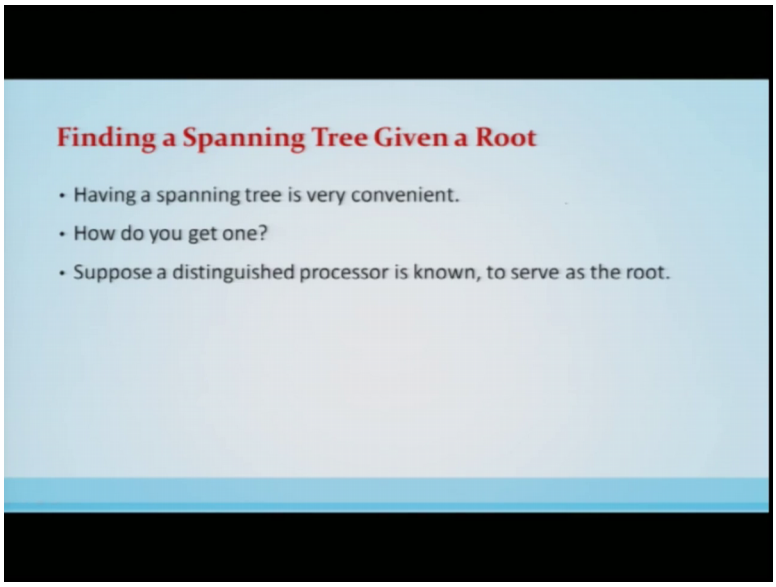


So, this is continued till all the nodes are able to communicate their information to the root node this is the convergecast another example same thing here we can see in the picture come the combination or the aggregate function is the maximum values. So, here P 1 is having x 1 and P 3 is having x 3. So, maximum of x 1 and x 3 is computed only that value will be basically sent to the parent similarly over here now here the max of x 3 x 1 and basically x 2 will be basically aggregated and sent to the to the root here in this case.

So, x 3, x 1 and x 2, similarly from this end also it will be done and from the root it will be basically again aggregated and the result will be there and this is called converge cast this aggregation function differs from application to application. So, this particular algorithm is useful for convergecast you can combine broadcast and convergecast together in some of the application where you want to send the information what together and when together in the convergecast will basically collect the information and deliver to the root node.

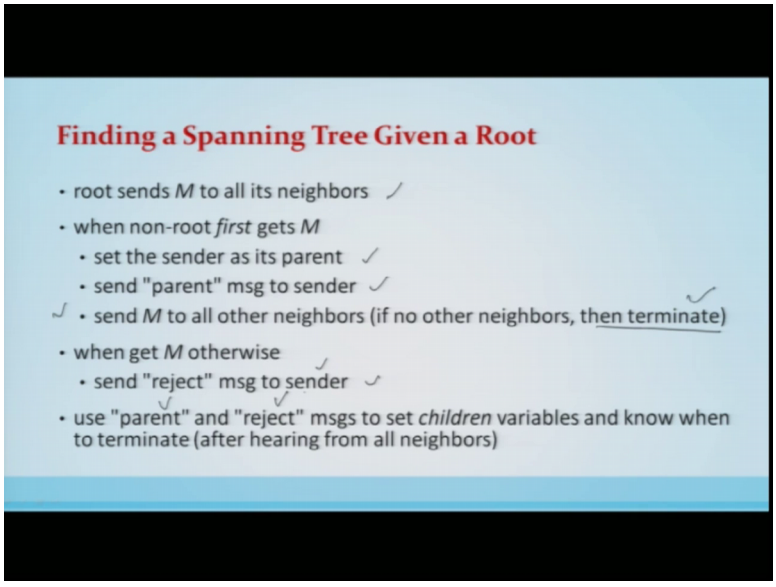
Now, these 2 previous algorithms assume or infrastructure that is called a spanning tree already in place now we are going to see how to construct a spanning tree in a distributed using distributed algorithm over the network.

(Refer Slide Time: 22:34)



So, the first algorithm which we are going to discuss is finding a spanning tree when a root node is given.

(Refer Slide Time: 22:41)

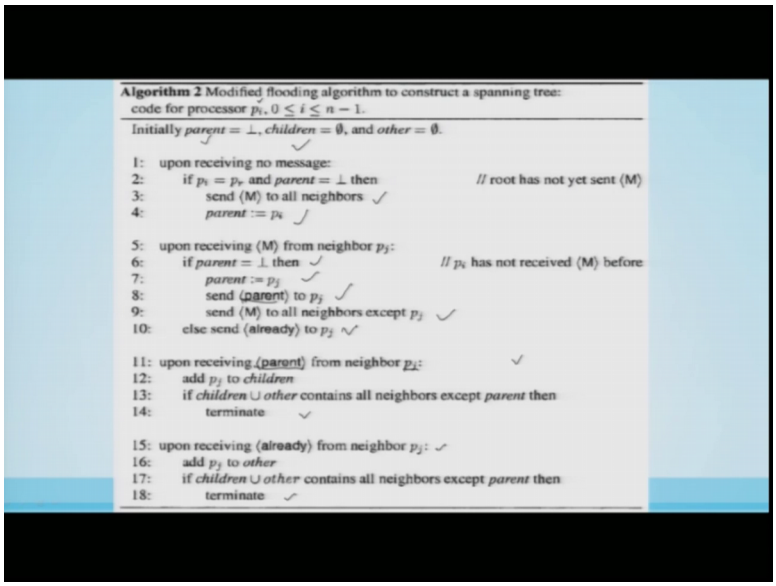


So, this algorithm is based on flooding algorithm. So, flooding algorithm is a very very simple algorithm where in a particular node initiates the flooding process with their

message m it will send to all its outgoing channels the channels which receives these messages will further send those messages on its children and. So, on this particular after transmitting those particular nodes will terminate and this is called a flooding algorithm using the modification of the flooding algorithm this particular spanning tree with a given root we are going to discuss.

So, let us see the algorithm. So, the root will send message m to its neighbors now when a non root node first gets m it will send the sender as its parent and also it will send a parent message to its sender the sender m will send m to all its neighbors that is children other than from where it has received and then it will terminate. So, when m is received otherwise means if it is and receive, then basically, it will send the reject message to the sender meaning to say that if the node has already set its parent and it is receiving m then it will send a reject to its basically parent now. So, that is why the parent and the reject there are 2 kinds of messages basically will be sent to the parent after receiving the message and then it will terminate the nodes.

(Refer Slide Time: 24:23)



So, this is the algorithm for finding a spanning tree of a root node in more details let us see this particular algorithm uses 2 variables parent and children parent is initialized to null for all the processors all the processors means this is the code for a processor pi. Similarly this particular code will run on all the processors from 0 to n minus 1 and there

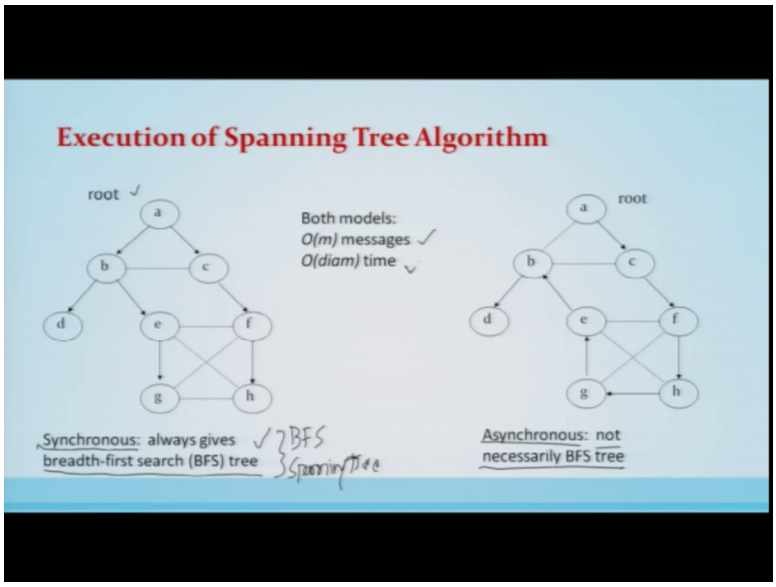
are 2 variables which will be initialized to parent will be initialized to null and children will be initialized to empty and others will be initialized to basically also empty.

Now upon receiving if the root has not yet sent, then it will send the message to all its neighbors and it will also send its parent is Pi because it is a root upon receiving message m from the neighbor Pi if the parent is empty as I told you then it will sent the parent and it will send the parent message to the parent and a message will be sent to all the nodes all the neighbors except Pj.

Else means if the parent is already set, then it will send already message to Pj. Now upon receiving the parent message from a neighbor Pj it will add Pj to the children. So, and the children union others will if contains all the neighbors except the parent, then it will terminate, then the node will terminate upon receiving already message from Pj, it will add Pj to the other if the children union other contain all the neighbors except the parent, then it will terminate when all the nodes terminates then the algorithm completes and basically a spanning tree will be constructed.

Let us see the example of the algorithm for a spanning tree construction for a given root.

(Refer Slide Time: 26:10)

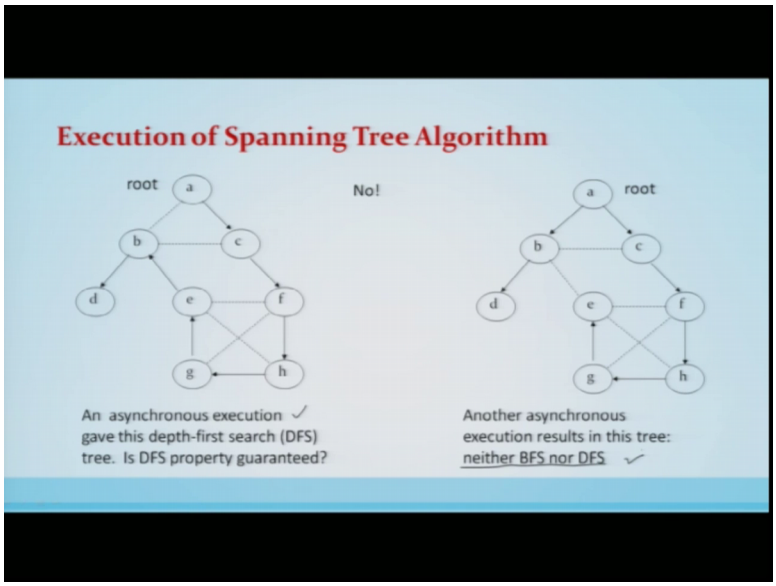


So, here we can see that if the algorithm runs with a given root let us see a; it will construct the spanning tree shown as basically the dark lines and dotted lines are non tree edges that will not be covered this particular tree is basically constructed when the model

is synchronous and you see this particular tree is a BFS tree. So, if the model is synchronous, then it will always constructs a spanning tree which is also a BFS a spanning tree. Now on the other hand if the model the timing model is the asynchronous then this particular tree is not necessarily the BFS why because the messages and the computational even the processors they have a different kind of delays.

So, that is why the asynchronous model will not necessarily gives the BFS always using this particular algorithm, but as far as the complexity is concerned total number of messages is M; M is the number of channels or the edges are there in that particular a graph and the time which it takes is of the order diameter that is.

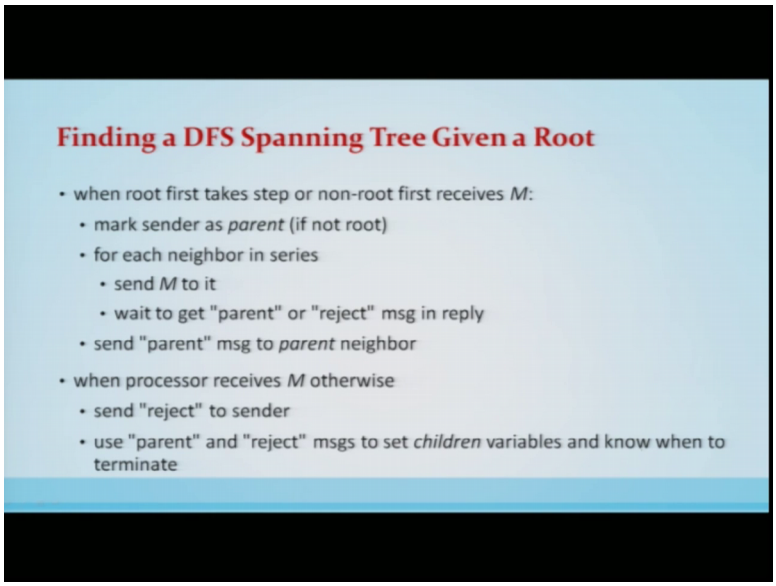
(Refer Slide Time: 27:29)



So, we can see here is that in asynchronous execution what kind of tree a spanning tree we are going to get kind means there are 2 kinds of the spanning tree we are talking about one is DFS and the other is BFS. So, as we have seen that in a synchronous mode of this algorithm this particular algorithm always gives a BFS, but if it is asynchronous then sometimes it will give BFS sometimes it gives a DFS or it may not give BFS or DFS.

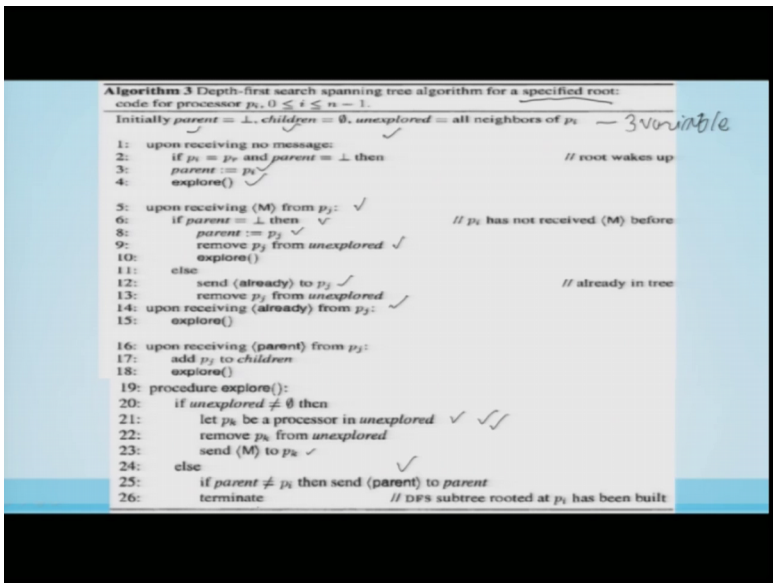
So, basically asynchronous executions in this tree neither BFS nor a DFS kind of tree is being resulted. So, basically it is a spanning tree only we without this particular property of a BFS and DFS if it is the model is asynchronous model.

(Refer Slide Time: 28:23)



Now, if the root is given, then how are we going to find out a DFS a spanning tree? So, this is the algorithm which we are going to discuss now.

(Refer Slide Time: 28:29)

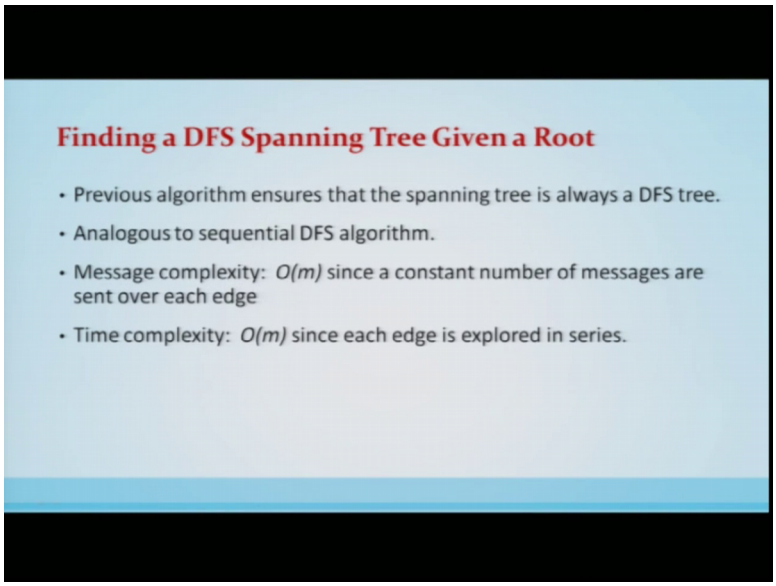


So, this algorithm definitely gives you the DFS spanning tree that is a spanning tree will be having this property and this is span in this particular algorithm requires a root to be specified; that means, a initiate of node to be known and it requires if the variables like parent which is initialize to nil and children is empty that unexplored is all the neighbors of a particular processors pi.

So, there are 3 different variables are required in this algorithm. So, we are going to see how these particular variables are used. So, initially all the parents will be empty or a null. So, let us see upon. So, the parent will start by initializing its parent variable as spi and it will explore. So, the if the parent is having a unexplored nodes in its variable then it will it will select pk that is the first processor you know explore remove pk from unexplored and send message m to pk and then upon receiving the message m from Pj if the parent is null the first time, it is receiving the message then it will set the parent from where it is receiving the message that is Pj and will remove Pj from unexplored it is a unexplored of this Pj and then again it will explore and the. So, basically that message it will send to one of the neighbors one of its neighbors and which will be taken out from the Pj and so on.

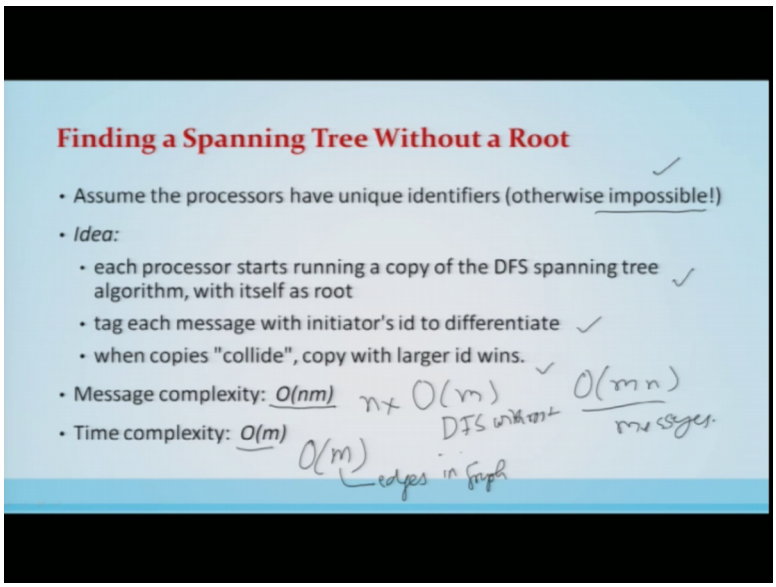
Now, if the message is already basically if the parent is not null then it will send already message to Pj because Pj has already set its parent variable. So, it will send already message back. So, upon receiving already message from pj, it will start exploring; that means, it will explore to the next possibility in; it is a unexplored node and so on. So, the termination will occur here when it comes to the explored and the explored is already exhausted and if the parent is not Pi then it will send the parent to the parent message to the parent and it will terminate and when the root will get the parent message then the entire algorithm terminates. So, this particular algorithm 3 will construct a DFS spanning tree; it is almost same as the sequential algorithm which we have seen on the on the DFS, but it will run on the distributed setting.

(Refer Slide Time: 31:15)



So, the previous algorithm ensures the spanning tree is always having a property which is called a DFS tree. So, as I told you it is analogous to the sequential DFS algorithm about message complexity. So, it is it will be of the order m since the constant numbers of messages are sent over the over the edges time complexity is also of the order m since each edge is explored in this particular series. So, this particular algorithm is having same order m for the message complexity and order m for the time complexity a challenge is that how to design that a DFS spanning tree when the root is not given.

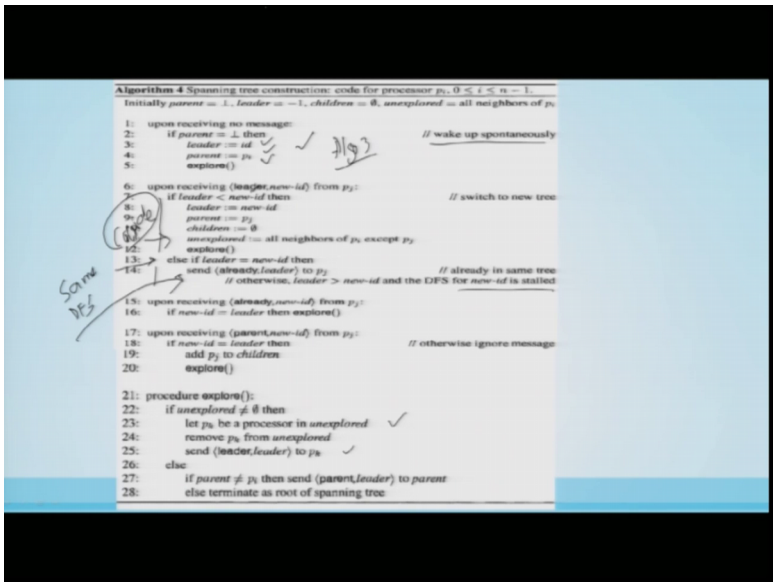
(Refer Slide Time: 32:01)



So, the next algorithm would be to find a spanning tree without a root node. Now we assume that the processors have the unique identifiers and here it is written otherwise impossible. So, impossible means there is impossibility result, this says that if the processors do not have the unique identifier then this algorithm will not work. So, this algorithm assumes that the processors have unique identifiers. So, the idea here in this particular algorithm is that each processor runs the copy of DFS algorithm which we have seen earlier with a root node with itself as a root and tag each message with the initiator id to differentiate.

Now, when the copies collide with a and the copy with a larger id wins. So, let us see first this particular algorithm and then we will discuss about the complexity of this particular algorithm.

(Refer Slide Time: 33:02)



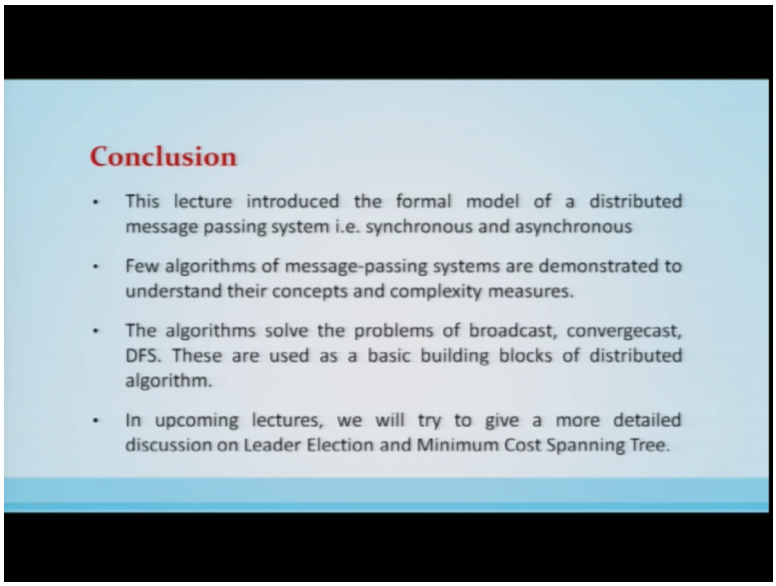
So, this particular algorithm is about the DFS when the root is not given. So, basically some of the nodes which will wake up they will start constructing DFS as per the algorithm number 3 so; that means, several DFS tree construction will be initiated if more than one nodes they wake up simultaneously and after that they will send the message they will explore so; that means, they will set its leader and its parent Pi and then it will explore.

So, in the explore; it will send the leader to one of the neighbor node through the channel and upon receiving this leader message from Pj the node has different possibilities. So,

for example, if the node has is already basically a part of another construction of a DFS tree then it will collide. So, this particular case will deal with the collide and if the node which has received this leader is on the same tree in the same DFS tree then leader is equal to new id and it will send already message to the already leader to the Pj otherwise what it will do; otherwise, it will stall the new construction of a DFS to grow further so; that means, collide in the sense if the collide and the incoming a new a incoming leader of the a node is a having higher id then it will basically survive and it will basically grow further otherwise it will stall will stall the growth.

So, there are 3 possibilities which are explained over here and rest of the algorithm steps are basically the same now the complexity of this particular algorithm is shown as the message complexity is of the order n times m why because order m is the complexity of the DFS with a root node and since n different processors can initiate it. So, it becomes order of m times n message complexity and the time complexity is of the order m that is the total number of in the edges or the channels in the graph.

(Refer Slide Time: 35:55)



So, in the conclusion we can see that this particular lecture has introduced you a formal model of a distributed message passing system for synchronous and asynchronous timing model with no failures. So, we have not assumed any failure and in this particular model we have seen some basic algorithms.

So, these basic algorithms in this particular model also we have seen; how to analyze; how to do the analysis that is the time analysis and the message complexity analysis the algorithms these algorithms which we have seen here will solve the problems of broadcasting converge casting and construction of a spanning tree that is a DFS. So, these are used as the basic building blocks of the distributed algorithm.

Now, in the upcoming lectures, we will try to give a more detailed discussion they are more complex algorithms and they are basically base are called the leader election algorithm and minimum cast spanning tree construction algorithm.

Thank you.