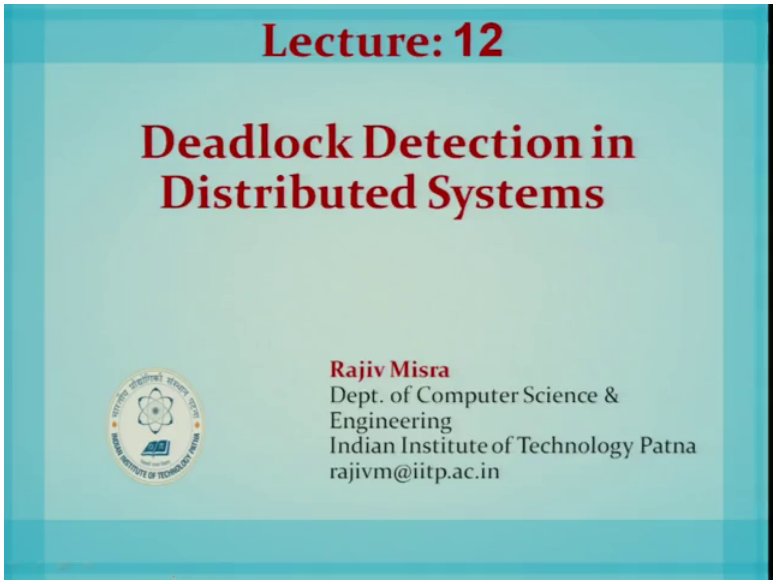
Distributed Systems
Dr. Rajiv Misra
Department of Computer Science and Engineering
Indian Institute of Technology, Patna

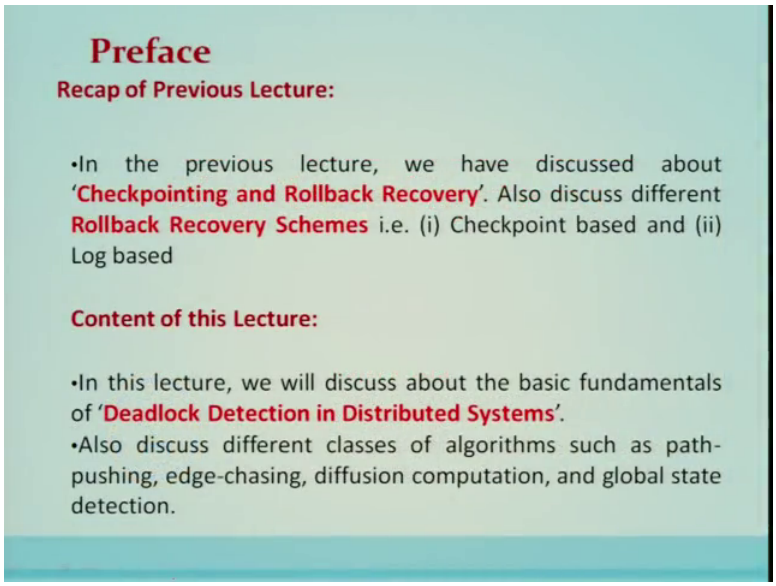
Lecture – 12
Deadlock Detection in Distributed Systems

Lecture 12; deadlock detection in distributed systems.

(Refer Slide Time: 00:19)



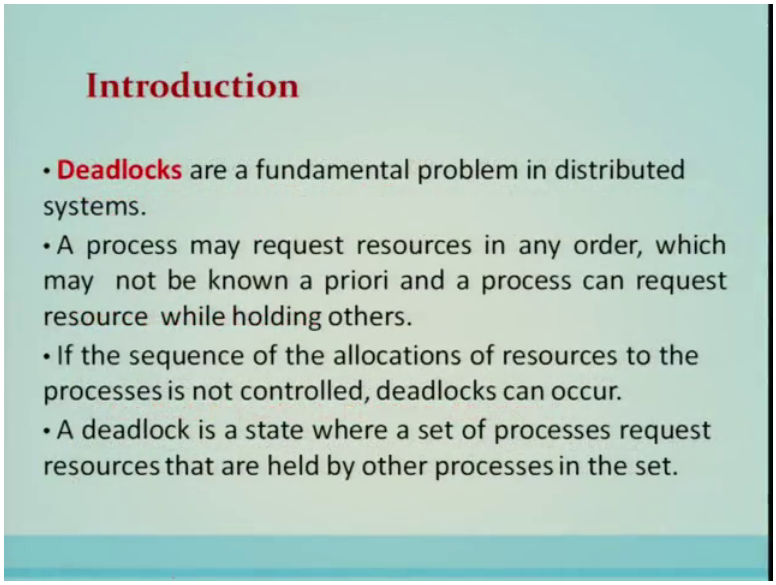
(Refer Slide Time: 00:21)



Preface recap of previous lecture in previous lecture, we have discussed about checkpointing and rollback recovery and also discussed different rollback recovery schemes that is checkpoint based and log based rollback recoveries.

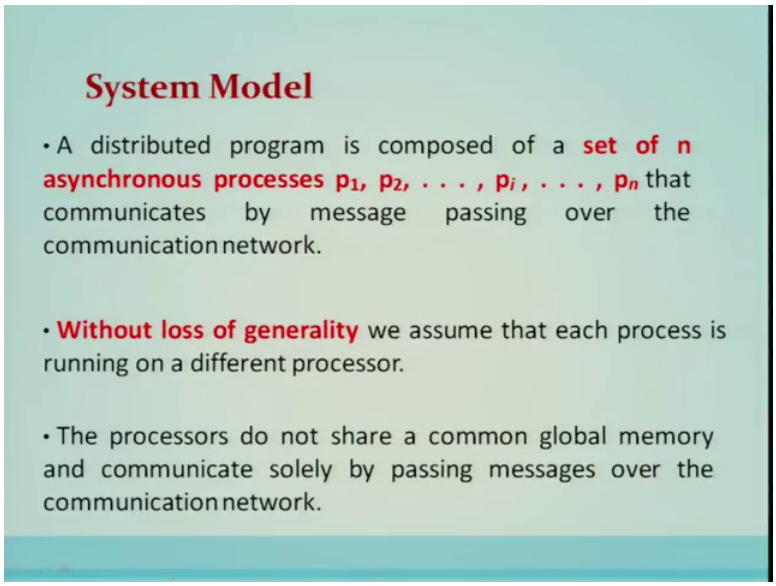
Content of this lecture in this lecture, we will discuss about basic fundamentals of deadlock detection in the distributed systems also discuss different classes of algorithms such as path pushing edge chasing diffusion computation and global state detection a methods for designing the distributed deadlock detection in distributed systems.

(Refer Slide Time: 01:03)



Introductions; deadlocks, deadlocks are the fundamental problem in the distributed system the process may request resources in any order which may not be known a priori and a process can request resource while holding others, if a sequence of allocation of resources to the processes is not controlled the deadlock can occur. So, a deadlock is state where a set of processes requests resources that are held by the other processes in the set that is the deadlock.

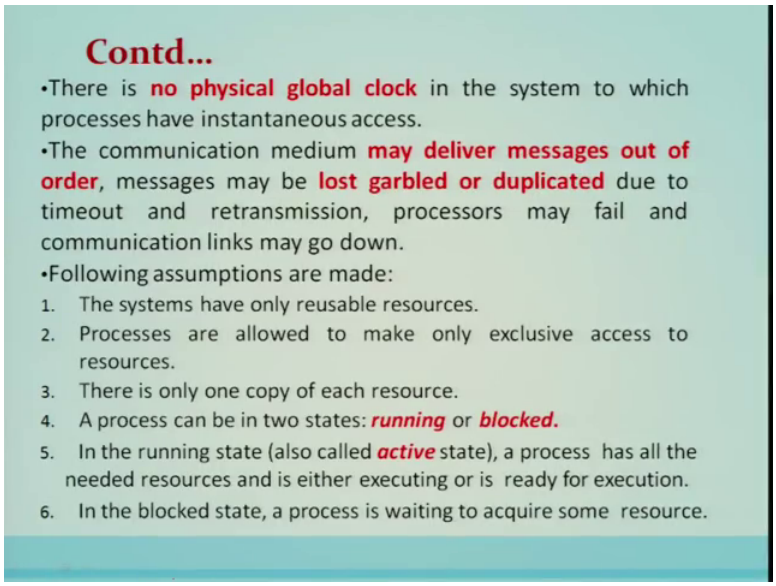
(Refer Slide Time: 01:43)



So, we are going to now define the system model which we are going to use in the deadlock detection algorithm a distributed program is composed of a set of n asynchronous processes p 1 to p n that communicates by message passing over the communication network without loss of generality we assume that each process is running on the different processor.

The processor do not share a common global memory and communicates only by passing messages over the communication network also; there is no physical global clock in the system to which the processes have instantaneous access the communication medium may deliver the messages out of order messages may be lost messages, may be garbled or they may be duplicated due to the timeout and retransmission processors may fail and communication link may go down all different possibilities can happen.

(Refer Slide Time: 02:50)

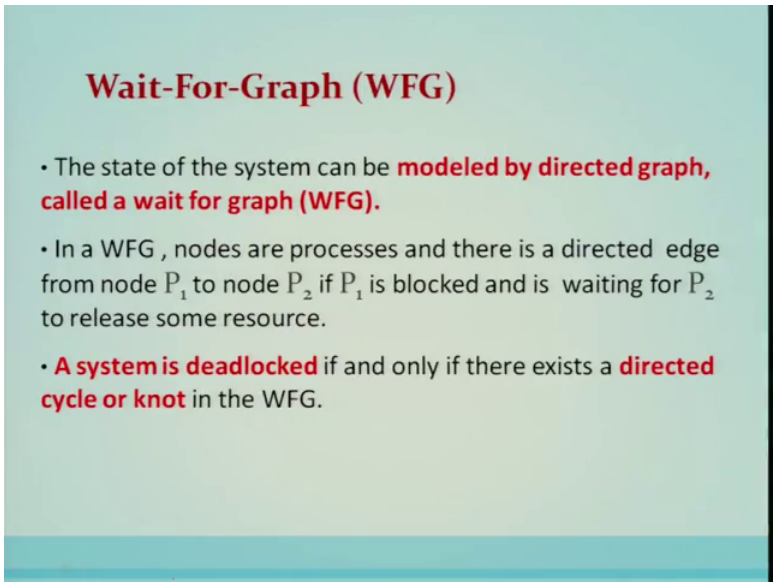


Because there are different set of processors and processes which are separated geographically and connected by the communication network and they exchange they communicate by passing messages through the communication network which may have these kind of problems.

So, the following assumptions are made the systems have only reusable resources first second assumption is processors are allowed to make only exclusive access to the resources third; there is only one copy of each resource a process can be in one of the 2 states that is running or a blocked.

In a running state, a process has all the needed resources and is either executing or is ready for the execution in the block state, a process is a waiting to acquire some resource, then we are going to define a data structure which is called a wait for graph which is used here in deadlock.

(Refer Slide Time: 04:20)



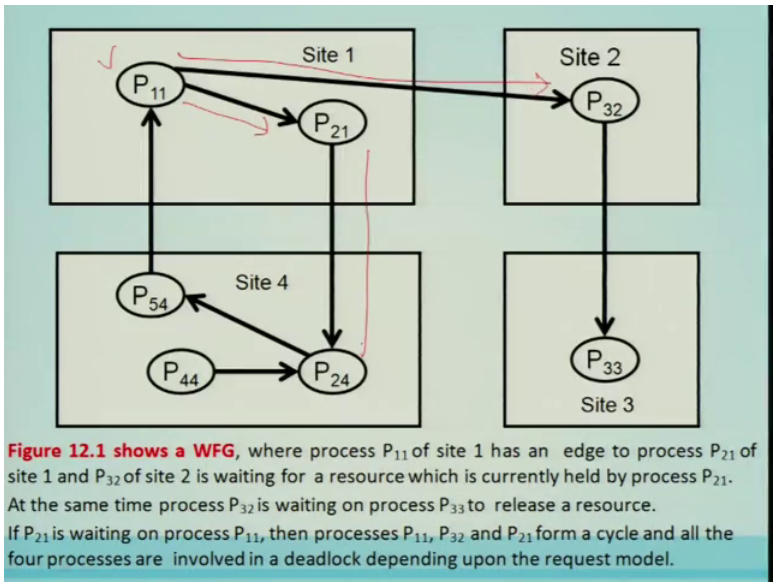
A state of the system can be modeled by directed graph which is also called as a wait for graph WFG in a wait for graph nodes are the processes and there is a directed edge from a node P i to P j, if p i is blocked and is waiting for P 2 to release some resource, then there is an edge. So, we can express wait for graph using P i and P j. So, this is the directed edge from P i to P j if process P i is blocked and this is waiting for P j to release the resources. So, this kind of graph is called wait for graph.

A system is deadlocked if and only if there exists a directed cycle or a knot in the wait for graph. So, for example, we can have another process P k now this P j is now waiting for P k and P k is waiting for P i. So, here there is a cycle in this particular graph. So, here the cycle means a set of waiting processes, this is a cycle this is a knot why because there is no; there is no outgoing edge out of this particular cycle and if any edge let us say P l is like this, then also it is a knot.

So, this is an example of a cycle or a knot and this set of waiting process is a system of deadlock depending upon the model of resources for which they are waiting. So, these particular models we are going to consider in the next slides.

So, this is an example of a wait for graph; now here the process P 11 of site 1, there is an edge to P 21, this edge is talking about and P 32. So, this particular process P 11, simultaneously waiting for 2 resources which can be released or which can be given by process P 21 and process P 32.

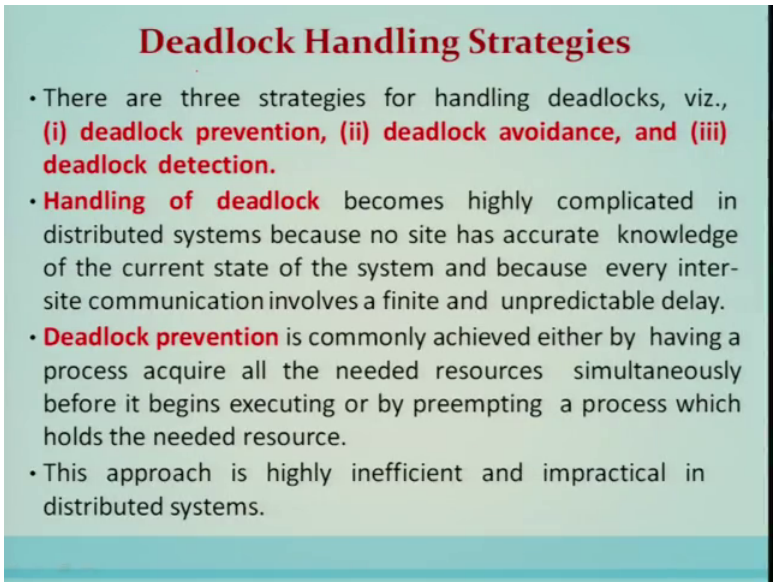
(Refer Slide Time: 07:18)



Now, if P 21 is also waiting for P 24 and P 24 is also waiting for P 54 and P 54 is waiting for P 11. So, by transitively we can say P 21 is also basically waiting for P 11, then it will form a cycle and all the 4 processes involved in the cycle, you used in the deadlock depending upon the request model now this is a cycle since this particular edge is outgoing. So, this cannot be a knot.

Now, preliminaries deadlock handling strategies there are 3 strategies for handling the deadlock that is deadlock prevention deadlock avoidance and deadlock detection handling of deadlock becomes complicated in distributed system where because no site has an accurate knowledge of the current state of the entire system and because every inter site communication involves a finite and unpredictable delay.

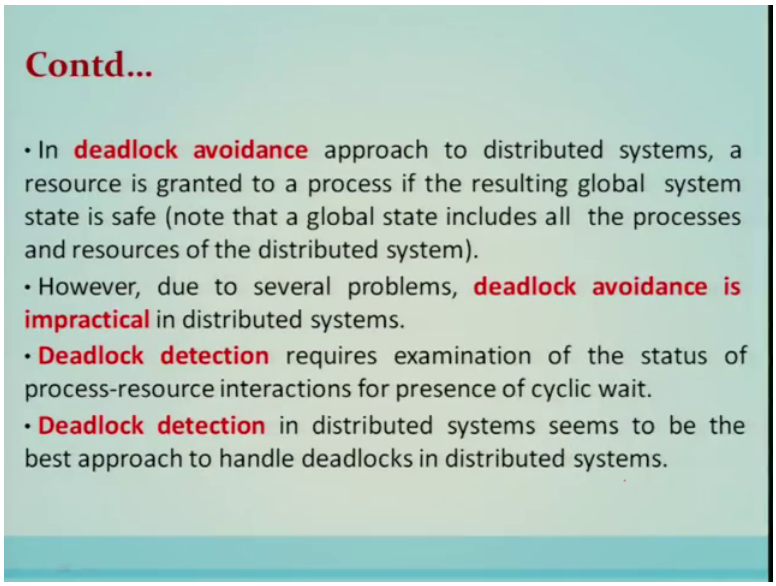
(Refer Slide Time: 08:20)



So, this particular scenario of the distribution or a distributed network in a distributed system makes the deadlock a bit difficult.

Now, another method another strategy for ending deadlock is called deadlock prevention is commonly achieved either by having a process that acquire all the needed resources simultaneously before it begins the execution or by preempting a process which holds the needed resources this approach is highly inefficient why because it is not always possible to acquire all the resources which are needed for the execution or it will be very costlier to preempt a process and release the resources and allocate the resource for the requesting process in order to continue. So, deadlock prevention is highly inefficient inapplicable or impractical for the distributed environment.

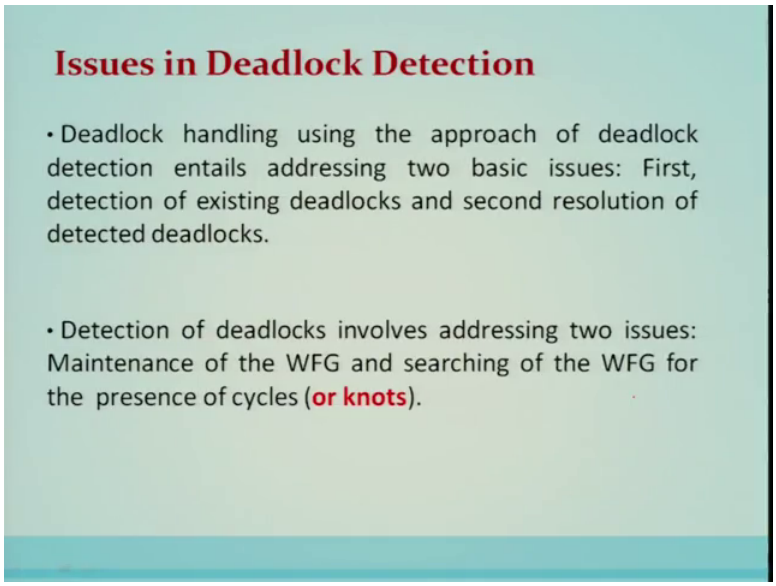
(Refer Slide Time: 09:38)



Now, another method of handling deadlock is the deadlock avoidance deadlock avoidance in distributed system is method deals with that the resources is granted to a process if the resulting global system state is safe note that global state includes all the processes and resources of the distributed system. So, here the resources are allocated in such a way that the resulting global state is safe so; however, due to several problems deadlock avoidance is also impractical in a distributed system.

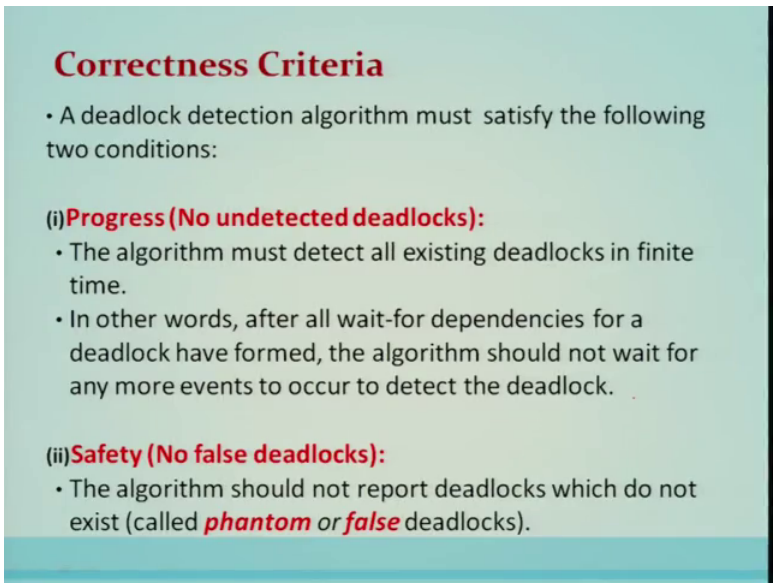
Deadlock detection requires examination of the status of process resource interactions for the presence of a cyclic wait. So, deadlock detection in a distributed system to seems to be best approach to handle the deadlock why because it is becoming a convenient in this model

(Refer Slide Time: 10:38)



Issues in deadlock detection; so, that is why we are going to discuss only one strategy that is deadlock detection; why because it is most practical in the distributed system scenario to handle the deadlocks deadlock handling using the approach of deadlock detection entails addressing 2 basic issues the first one deals with that the detection of the deadlock and second deals with the resolution of detected deadlocks. So, detection of deadlocks involves addressing 2 issues the first one is the maintenance of wait for graph WFG and second is searching this wait for graph for the presence of a cycle or a knot.

(Refer Slide Time: 11:28)

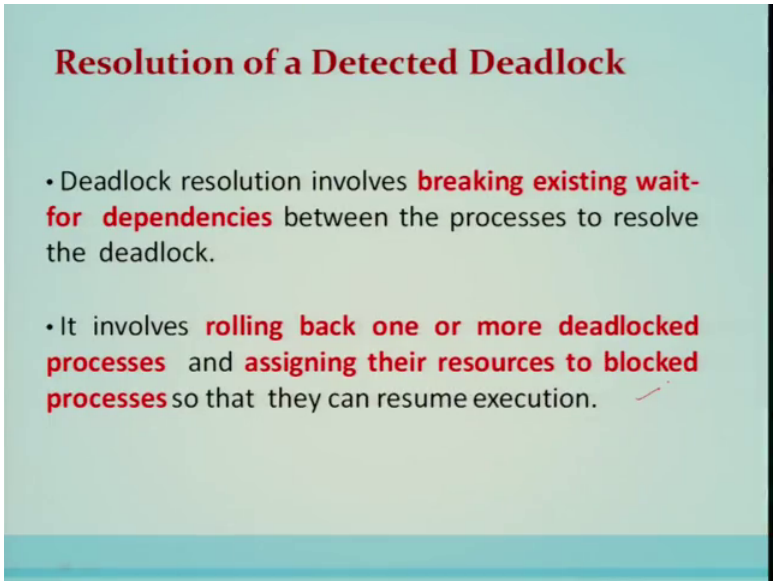


So, different algorithms will follow different schemes for these 2 methods correctness criteria a deadlock detection algorithm must satisfy the following 2 conditions to ensure the correctness of the algorithm.

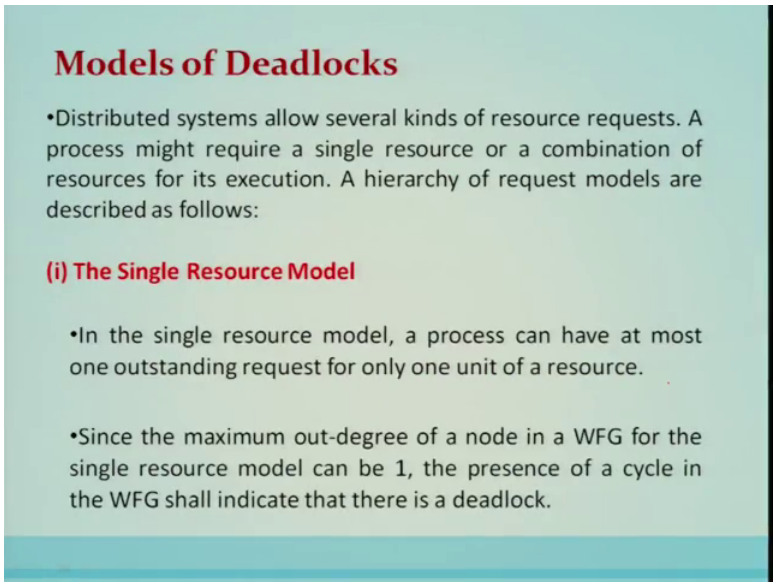
First is the progress the algorithm must detect all the existing deadlocks in a finite amount of time in other words after all wait for dependencies for a deadlock have formed that algorithm should not wait for any more events to occur to detect the deadlock.

Second condition for correctness of a deadlock is called safety that is the algorithm should not report the deadlock which do not exist and that condition is called phantom deadlocks or a false dreadlocks. So, safety ensures that it always report the correct state of the stable state that is called a deadlock and the second condition of correctness is the progress; that means, all the existing deadlocks must be identified in a finite amount of time.

(Refer Slide Time: 12:44)



(Refer Slide Time: 12:50)



Now, resolution of the detected dreadlocks; the deadlock resolution involves breaking the existing wait for dependencies between the processes to resolve the deadlock, it involves rolling back one or more deadlocked processes and assign their resources to the other blocked process so that they can resume the execution in the distributed systems.

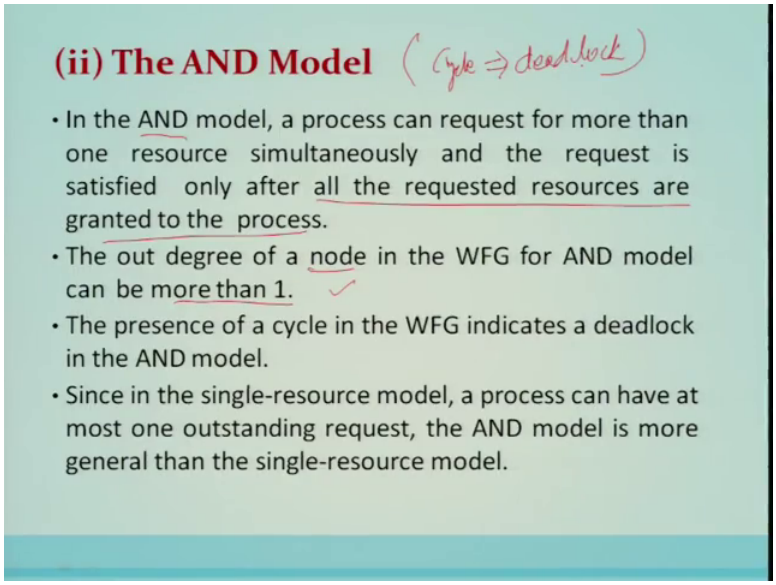
Models of deadlocks distributed system allows different kind of resource requests so; that means, they are represented by different model a process might require a single resource or a combination of resources for its execution.

A hierarchy of request models are described as follows the first resource request is the single resource model in single resource model a process can have at most one outstanding request for only one unit of resource. So, this is the maximum out degree of a node in wait for graph in this single resource model is one? So, the presence of a cycle in a wait for graph shall indicate that there is a deadlock. So, cycle in this particular single resource model will indicate that the system is in deadlock.

And model in and model a process can request for more than one resource simultaneously and the request is satisfied only after all the requested resources are granted to the process that is why it is called and model the out degree of a node here in wait for graph for and model can be more than one, the presence of recycle in a wait for graph indicates a deadlock in and model since in a single resource model a process can

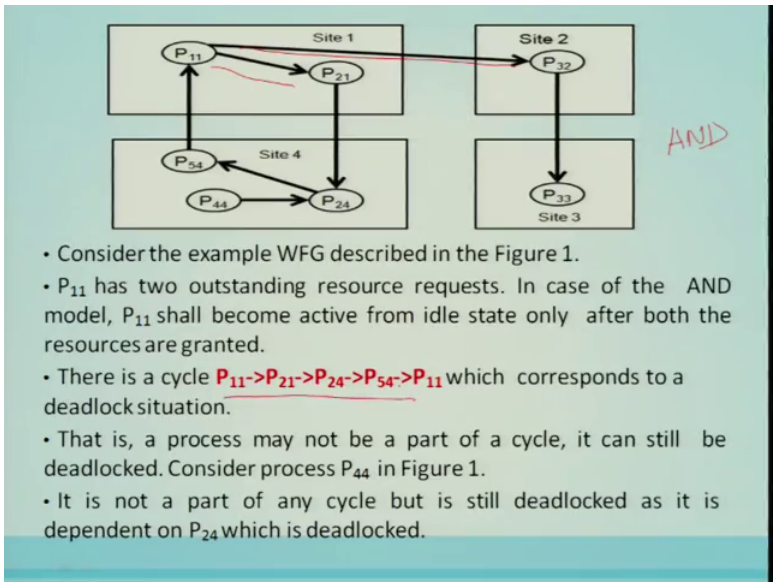
have at most one outstanding request the end model is more general than a single resource model.

(Refer Slide Time: 14:22)



Let us consider an example consider an example of a wait for graph in figure one and consider it as an AND model. So, P 11 has 2 outstanding resource requests that we have seen over here. So, P 11 shall become active from ideal only after both the resources are granted to P 11.

(Refer Slide Time: 15:01)



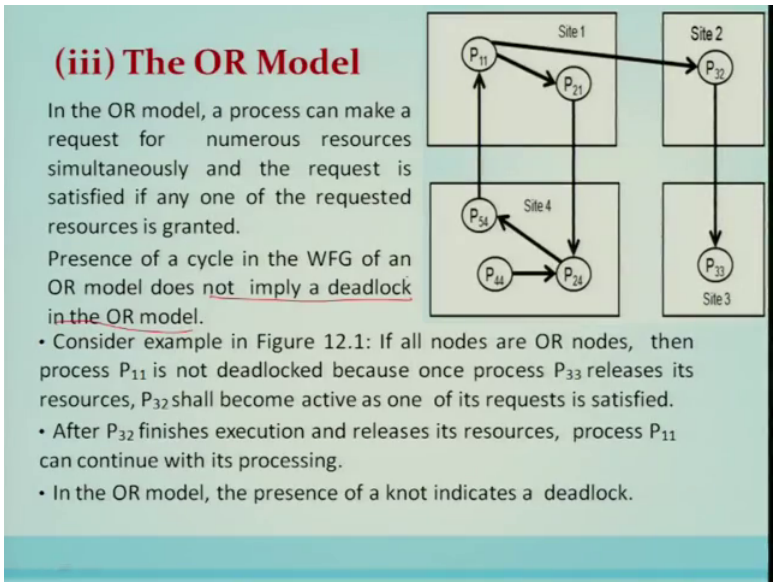
So, there is a cycle which corresponds to the deadlock situation why because in and model if there is a cycle; that means, it is a deadlock that is the process may not be in a part of cycle it still be a deadlock.

Now, another example is over here that this example says that a process may not be in a cycle yet it is in deadlock for example, P 44 is not in a cycle, but P 44 is in deadlock situation. So, it is not part of a cycle is deadlock, but it is deadlocked.

The or model in the or model a process can make requests for an numerous resources simultaneously and the request is satisfied if any one of the requested resources is granted presence of a cycle in a wait for graph in or model does not imply a deadlock in or model.

So, let us consider the example the same example here all the nodes are or node then the process P 11 is not in a deadlock why because if P 33 has finished its execution and releases the resources it will make the P 33 active.

(Refer Slide Time: 16:15)

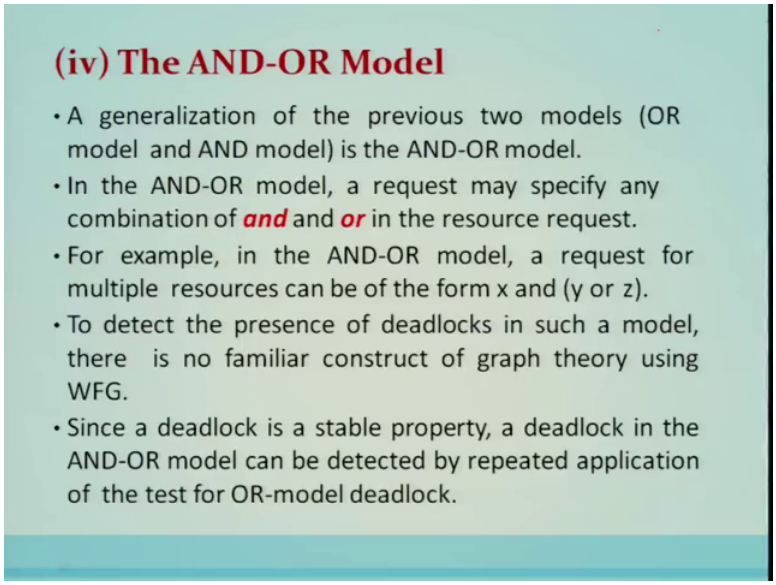


So, the once the resources are releases by released by P 33; it can be allocated to P 32. So, P 32 will also become an active after the resources are located.

So, after finishing P 32, this particular resource can be can be given or can be allocated to P 11. So, P 11 since it is an OR model. So, once that this particular resource is allocated. So, P 11 will become an active. So, it becomes an active, it can start its

execution and it will break the cycle. So, in the OR model the presence of a knot will indicate a deadlock basically here this particular condition is only a cycle not a knot. So, the presence of a knot in or model indicates the deadlock and OR model is a mixed model. So, it is a generalization of the 2 models and or model and or model a request may specify any combination of AND or in the request resource.

(Refer Slide Time: 17:45)



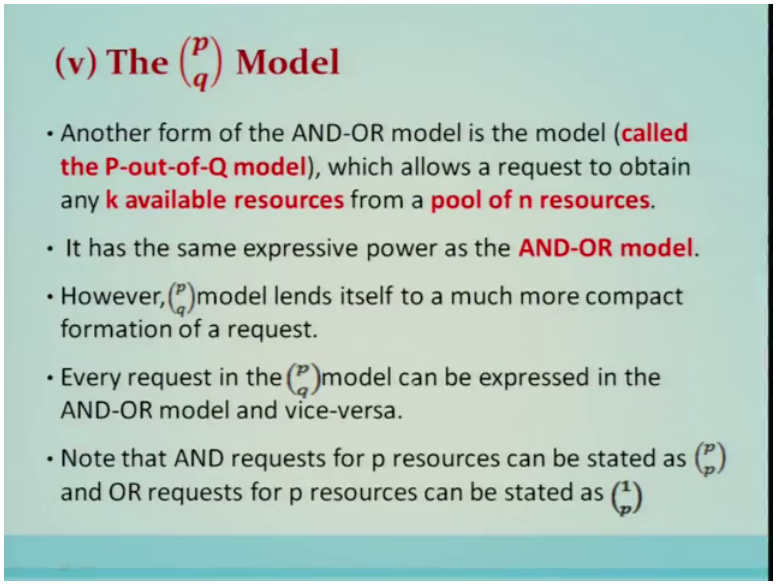
For example, in AND OR model a request for multiple resources can be of the form x plus y or z to detect the presence of a deadlock in such a model there is no familiar construct of a graph theory for that use, the wait for graph hence the deadlock is detected using its stable property.

So, a deadlock in AND OR model can be directed by repeated application of the test for the or model deadlock to find out the stable property and why because if the deadlock is nothing, but finally, the stable satisfying the stable property.

Now, another model is called P out of Q models. So, another form of and or model is called P out of Q model which allows the request to obtain any k available resources from a pool of n resources it has same expressive power as at as and or model we have seen earlier; however, P out of Q model lends itself to a much more compact formation of a request. So, every request in a P out of Q model can be expressed in the form of AND OR graph and vice versa note that and requests for P resources can be stated as P out of P; that means, all P resources are required that is the AND model and the OR

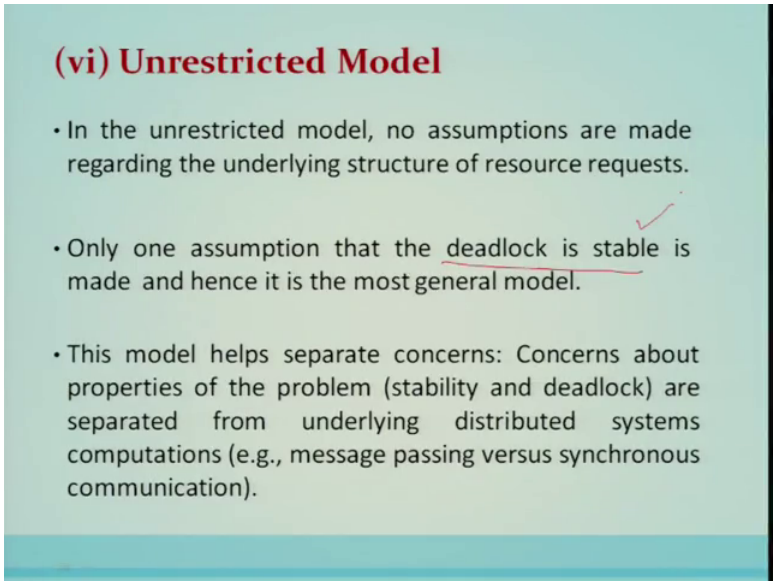
model request for the P resources can be stated as 1 out of P that is an OR model. So, P out of Q can be expressed in these 2 forms of OR and n model.

(Refer Slide Time: 18:45)



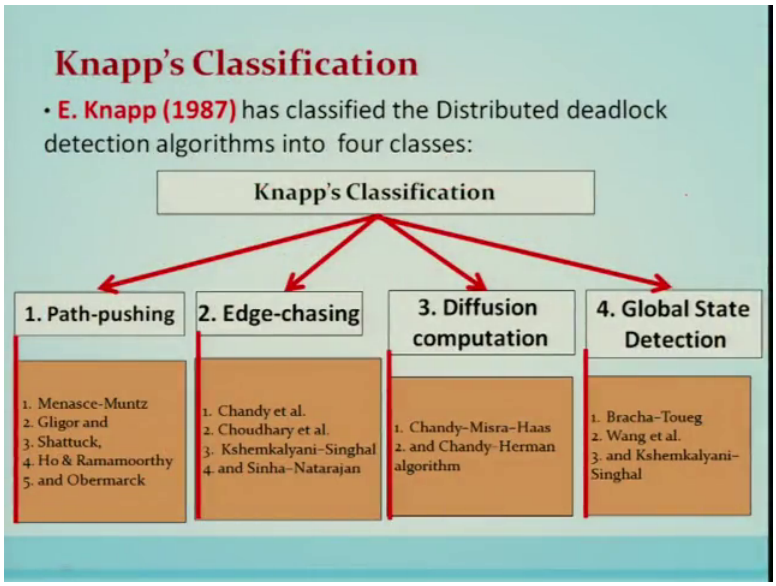
Unrestricted model in unrestricted model no assumptions are made regarding the underlying structure of the resource requests only one assumption that the deadlock is stable is made hence this is a most general model.

(Refer Slide Time: 19:29)



This model helps separate concern about the properties of the problem that is stability and the deadlock are separated from the underlying distributed computations.

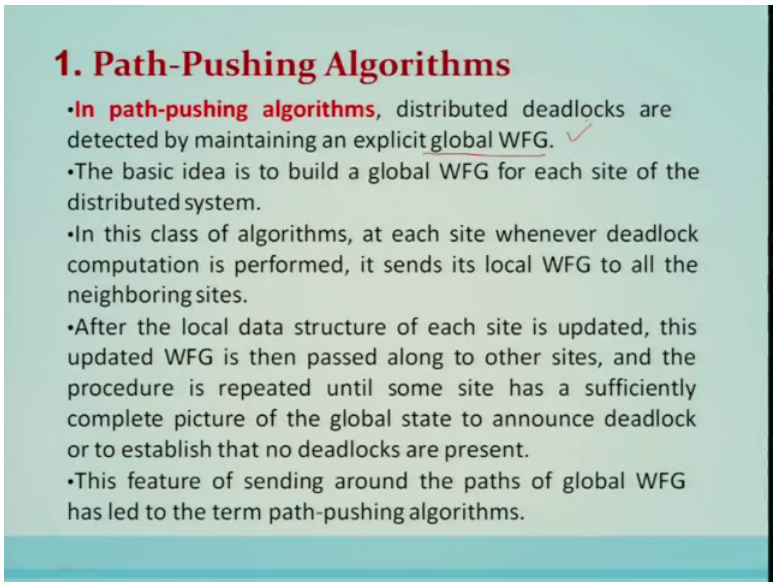
(Refer Slide Time: 19:55)



Classification of distributed deadlock detection algorithm classification of distributed deadlock detection algorithms Knapp's classification Knapp has classified the distributed deadlock detection algorithm in 4 classes. The first one is path pushing, second one is edge chasing, third one is diffusion computation, fourth one is global state detection; each of these classified methods to detect the deadlock is basically different algorithms are listed over here and these algorithm use this particular strategy that we will see that is 4 different strategies which will classify different algorithms distributed algorithms they are listed as path pushing edge chasing diffusion computation and global state detection let us see these strategies one by one.

Path pushing algorithms in path pushing algorithm distributed deadlock detection are detected by maintaining an explicit global wait for graph the basic idea is to build global wait for graph for each site of the distributed system.

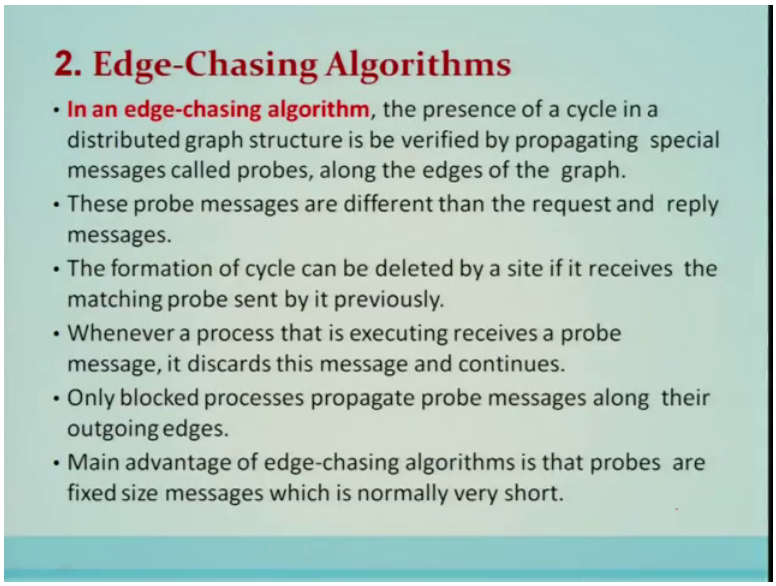
(Refer Slide Time: 21:12)



In this class of algorithms at each site whenever a deadlock computation is performed, it sends its local wait for graph to all the neighboring sites after the local data structure of each site is updated this updated wait for graph is then passed along to the other sites and this procedure is repeated until some site has sufficiently complete picture of the global state to announce the deadlock or to establish that no deadlocks are present.

The name path pushing here in this algorithm is used because the local data structure is sent along the path to different processes connected by the communication network hence the name is path pushing algorithm so; that means, the local wait for graph which is constructed by a particular node is sent along the path and once they are collected at all the ends. So, basically one node will get a complete picture of some site will get a complete picture of the global state and it will announce the deadlock are established that there is no deadlock present this feature of sending around the paths of the global wait for graph has led to the term path pushing algorithms . So, the algorithms which uses this strategy is called basically classified in these particular methods that is path pushing algorithms that we will see later on.

(Refer Slide Time: 22:56)



Edge chasing algorithms in edge chasing algorithms the presence of a cycle in a distributed graph structure is to be verified by propagating a special message called probes along the edges of the graph probe messages are different than the requests and reply messages of the computation.

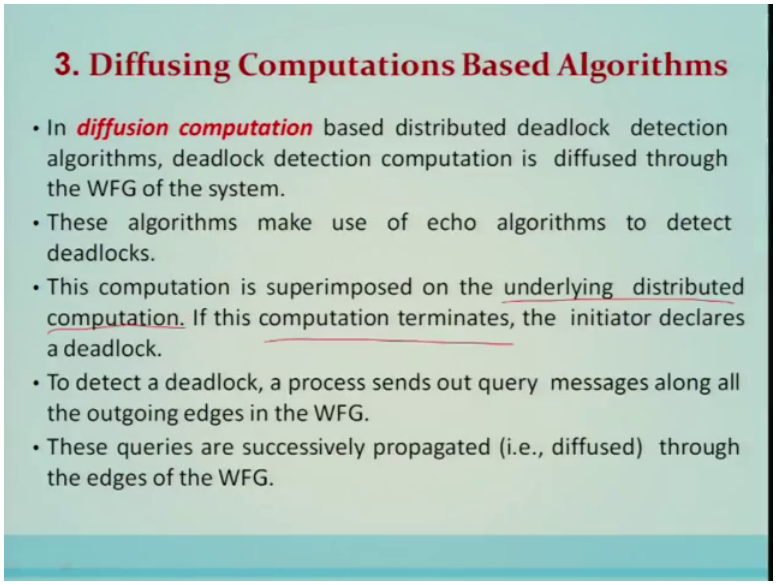
The formation of a cycle can be detected can be deleted by a site if it receives the matching probe matching probe sent by it previously. So, whenever a process that is executing receives a probe message it discards the message and continues; that means, if a particular process which is an active process and if it receives a probe. So, it will just discard this particular message why because the deadlock involves only the set of blocked processes and if the process is working that is not a part of the deadlock hence it discards the message and continues its current execution.

Only the blocked processes propagate the probe message along their outgoing edges. So, the main advantage of edge chasing algorithms is that probes are of fixed size and they are is small in size hence the overhead of message size is very very minimal and that is the advantage of edge chasing algorithms.

Diffusion computation based algorithms in diffusion computation based distributed deadlock detection algorithms deadlock detection computation is diffused through the weight for graph of the system these algorithm make use of echo algorithms to detect the

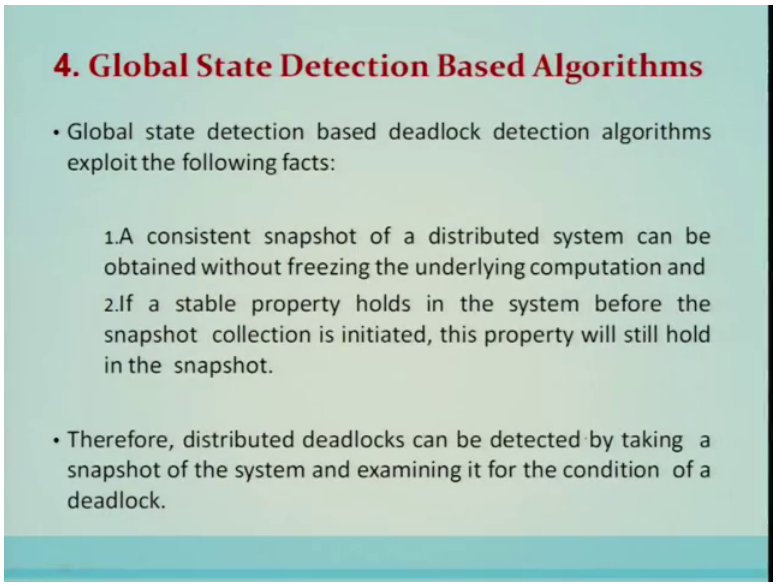
deadlock this computation is superimposed on the underlying distributed computation hence no separate execution for deadlock detection is taking place.

(Refer Slide Time: 25:01)



So, if the computation terminates the initiator declares the deadlock to detect a deadlock a process sends out query message along all the outgoing edges in the wait for graph these queries are successively propagated that is diffused through the edges of the wait for graph. So, when a blocked process receives first query message for a particular deadlock detection initiation it does not send a reply message until it has received a reply message of for every query it sent. So, for all subsequent queries for this deadlock detection initiation, it immediately sends back a reply message the initiator of the deadlock detection detects the deadlock when it receives a reply from every query it has sent out.

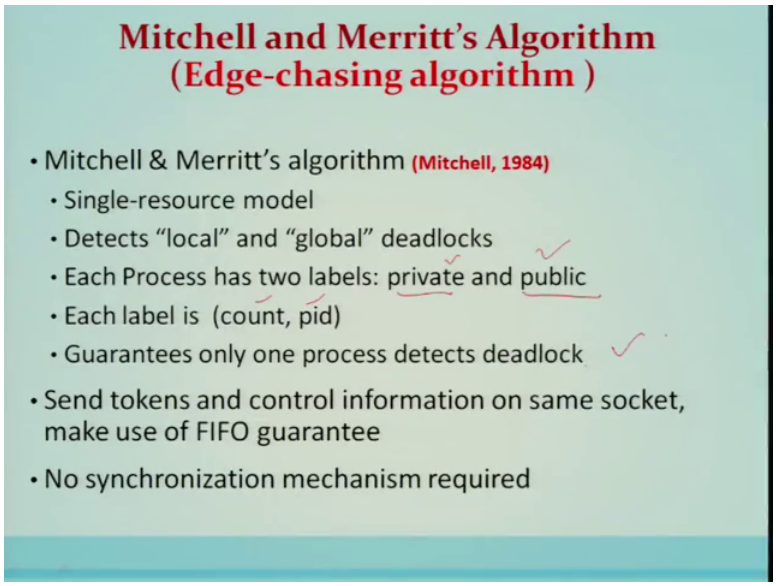
(Refer Slide Time: 25:56)



Global state detection based algorithms global state detection based deadlock detection algorithm exploit the following facts the first is that consistent snapshot of a distributed system can be obtained without freezing the underlying computation that we have seen in Chandy Lampert's algorithm. Now if a stable property holds in the system before snapshot collection is initiated, this property will still hold after the snapshot is available or it will be captured in the snapshot; therefore, distributed dead locks can be detected by taking a snapshot of the system and examining it for the condition of the deadlock.

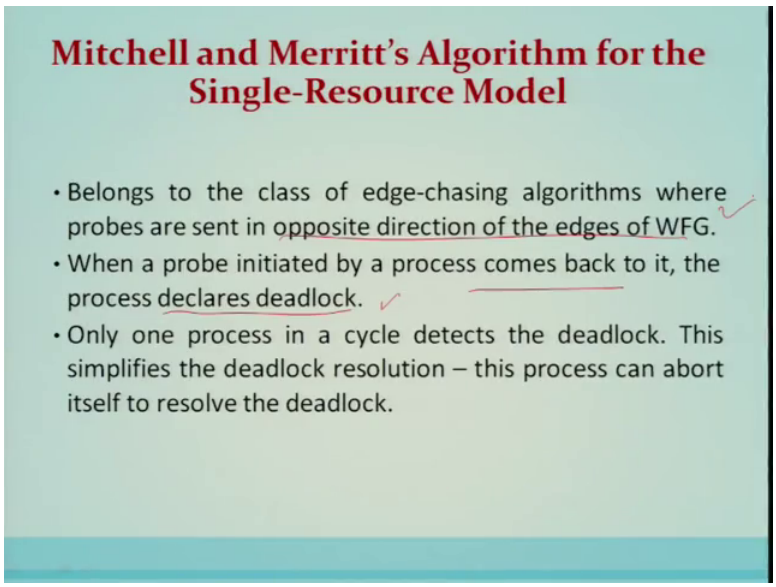
Deadlock detection algorithms now we are going to discuss deadlock detection algorithm and that is given by Mitchell and Marritt and this is called Mitchell Marritt algorithm and this particular algorithm is based on edge chasing approach which we have discussed in the previous slides; so, Mitchell Merritt's algorithm given in 1984 assumes a single resource model detects the local and global deadlocks each process has assumed 2 different labels private and public each label is accountant the process id guarantees only one process will detect a deadlock and that is why this particular method is popular.

(Refer Slide Time: 27:25)

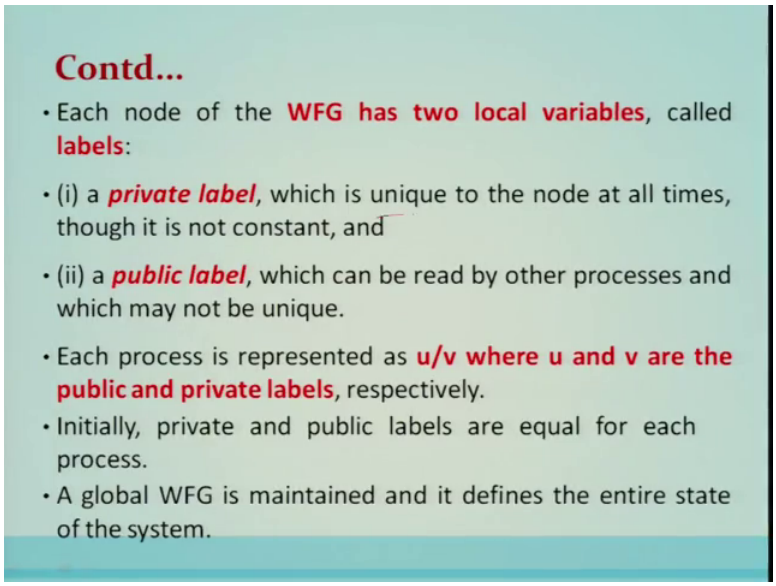


Send tokens and control information on the same socket and make use of FIFO guarantees, no synchronization mechanism is required in this algorithm this algorithm belongs to a class of edge chasing algorithms where the probes are sent in the opposite direction of the edges of the wait for graph when the probe initiated by a process comes back to it the process declares a deadlock only one process in the cycle detects the deadlock this simplifies the deadlock resolution this process can abort itself to resolve the deadlock.

(Refer Slide Time: 27:58)

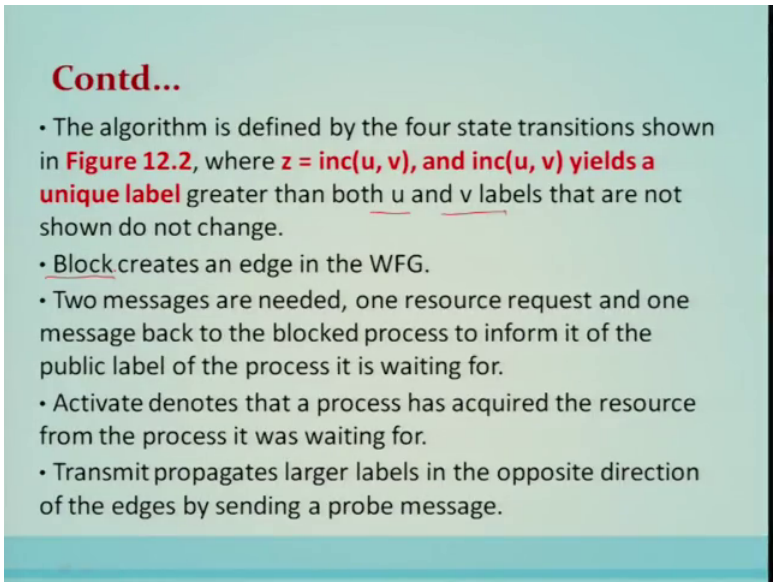


(Refer Slide Time: 28:07)



Each node in the wait for graph has 2 variables and they are called label private label and a public label private label is unique to the node at all the time though it is not a constant and the public level which can be read by the other processes and which may not be the unique each process is represented as u oblique v; that means, the private and the public label where u and v are the public and the private labels respectively. So, initially private and public labels are equal a global wait for graph is maintained and it defines the entire state of the system the algorithm is defined by 4 state transition shown in the next figure where z is nothing, but an increment of u v which yields a unique label which is greater than both x and y labels that are not shown to change.

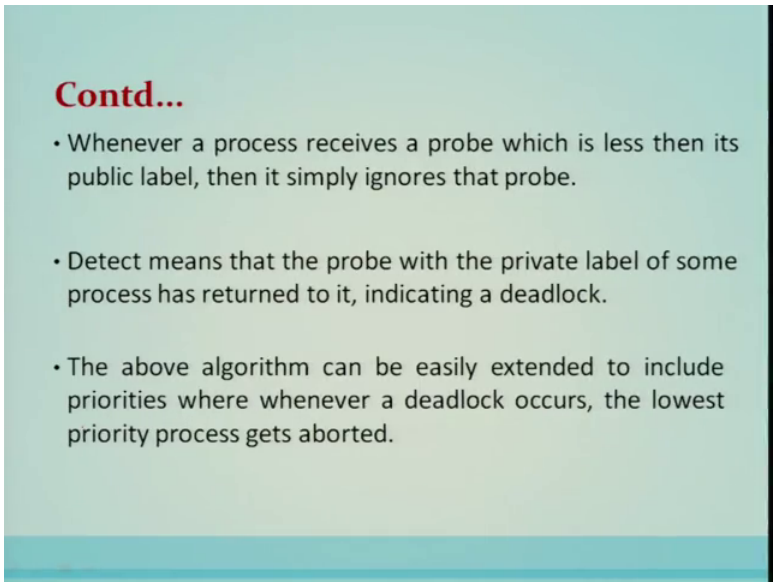
(Refer Slide Time: 29:28)



Now, another state is called a block state this will block will create an edge in a wait for graph that we will see in the next slide. Now 2 messages are needed one resource request and one the message back to the blocked state to inform it if the public label of the process is waiting for.

Now, another state is called an activate state activate state denotes a process has acquired the resource from the process it was waiting for now; the next is another state is called the transmit state. So, transmit propagates the larger label in the opposite direction of the edges by sending the probe messages.

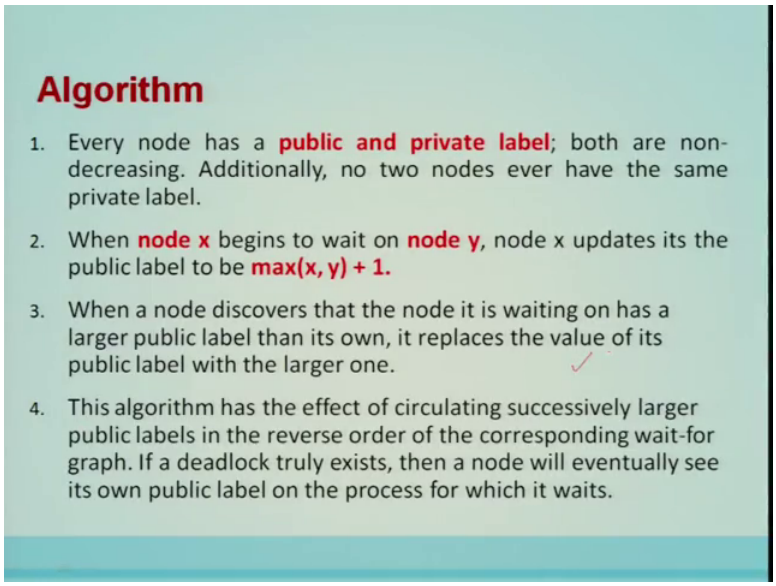
(Refer Slide Time: 30:28)



Let us see all these 4 different state transitions in this particular picture. So, this is the block activate transmit and detect you will explain. So, whenever a process receives a probe which is less than its public label it, then simply nodes that probe detect means that probe with the private label of some process has returned to it indicating a deadlock the above algorithm can be easily extended to include priorities where whenever a deadlock occurs the lowest priority process gets aborted and hence resolves the deadlock.

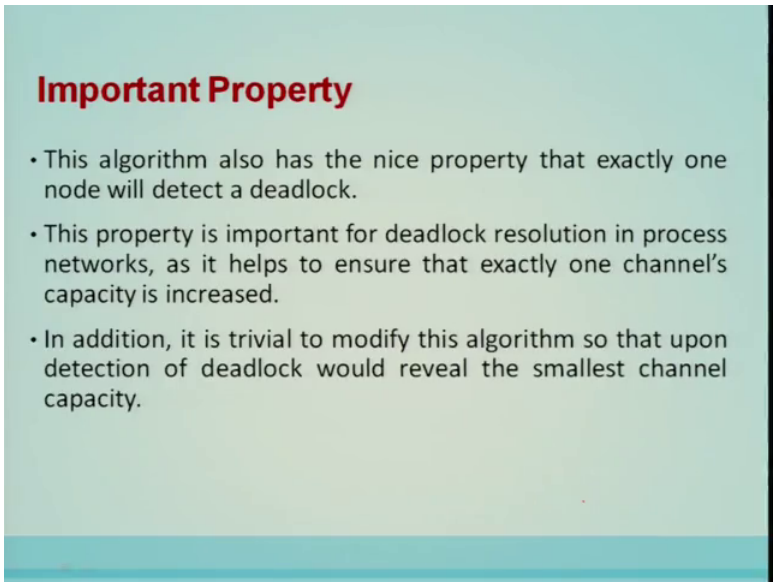
Now, we have to see the algorithm in more detail, the first steps is that every node has a public and a private label both are non decreasing additionally no 2 nodes ever have the same private label node x begins to wait on node y node x updates its public label to be max of x y plus 1 when a node discovers that a node; it is waiting on has a larger public label than its own it replaces its value of its public label with a larger one.

(Refer Slide Time: 31:25)



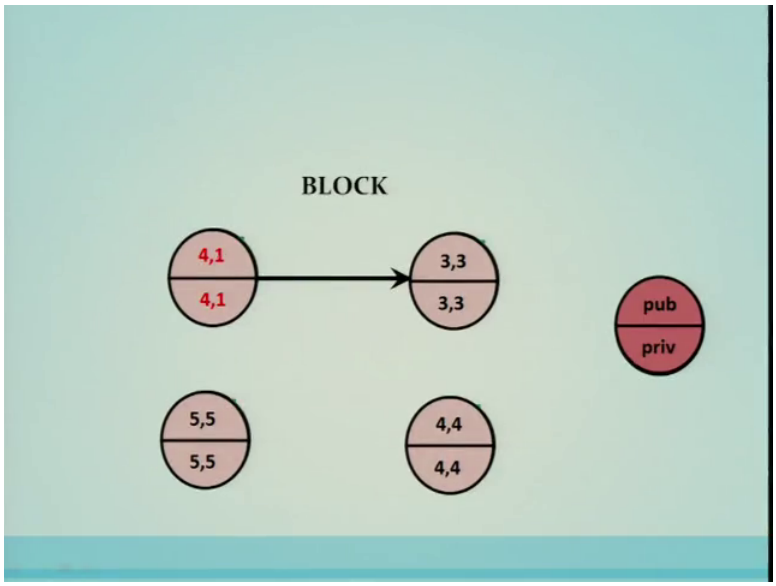
This algorithm has the effect of circulating successively larger public labels in the reverse order of the corresponding wait for graph if a deadlock truly exists, then the node will eventually see its own public label on the process for which it waits.

(Refer Slide Time: 32:01)



Let us see all these 4 different state transitions in this particular picture. So, this is the block activate transmit and detect you will explain. So, whenever a process receives a probe which is less than its public label, it is trivial to modify this algorithm. So, that upon detection could reveal the smallest channel capacity.

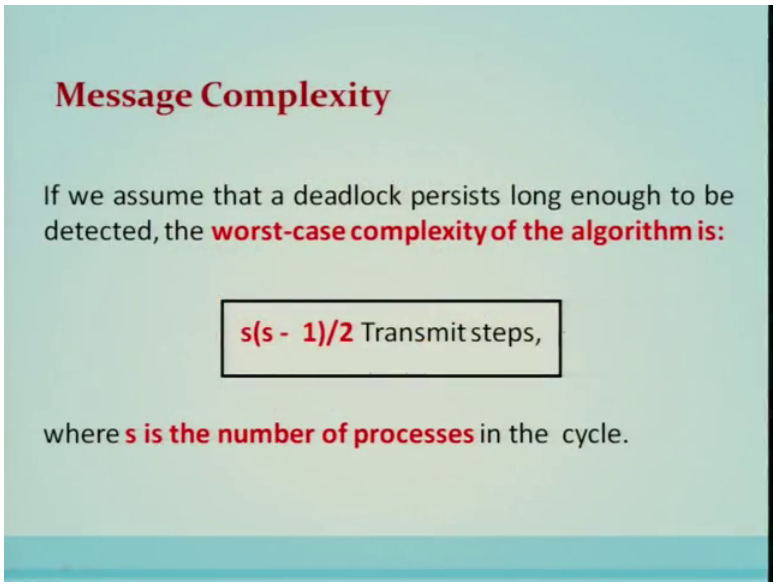
(Refer Slide Time: 32:28)



Let us understand this algorithm through an example. So, just see that here this particular node has basically the label both are same u and v. Now when it will block, then this will this label public label will increase that is x and y; you have to say max of these 2 a plus one. So, earlier it was one and it was 3. So, max was 3 plus 1 and that is equal to 4. So, 4 is now relabeled here and this edge will come in the block process. Similarly you just see that if you want to add an edge. So, you have to take the maximum 4 and 5 is 5 plus one that is that is 6. So, this will be 6 in the next slide we will see that 6 5.

Now, if you want to add a label like this then 3 and 6 is 6 plus one that is basically the seven. So, seven will be made over here. Now these labels will basically move on the opposite direction of these edges of this is a wait for graph now these labels will move in the opposite direction of these edges. So, that these labels get updated in the next step of this algorithm will show. So, it will transmit seven and will get updated, it will again transmit seven that is going in the opposite direction to take the higher values of the label and finally, when that same label 7 will come back to the same process; this process will understand that this system is in a state of deadlock because it has detected a cycle.

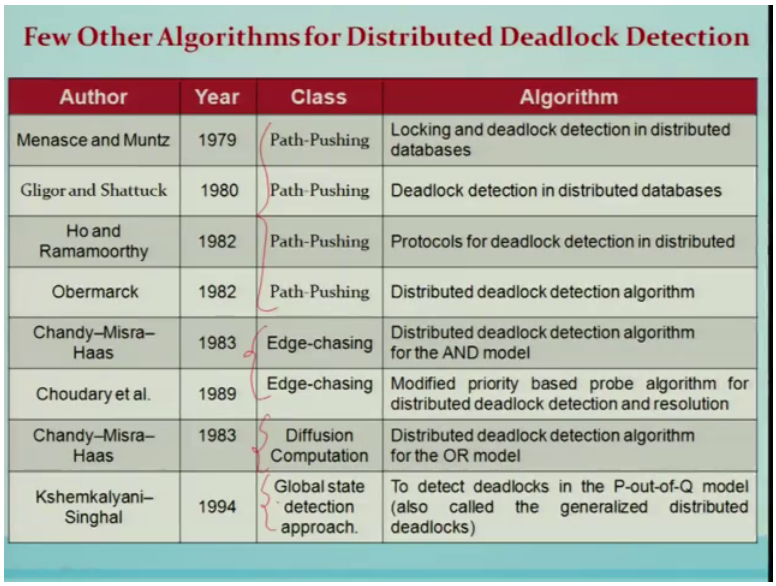
(Refer Slide Time: 34:32)



Now, message complexity of this algorithm if we assume that a deadlock persist long enough to be detected the worst case complexity of the algorithm is s times s minus 1 by 2 transmit steps where s is the number of processes in the cycle.

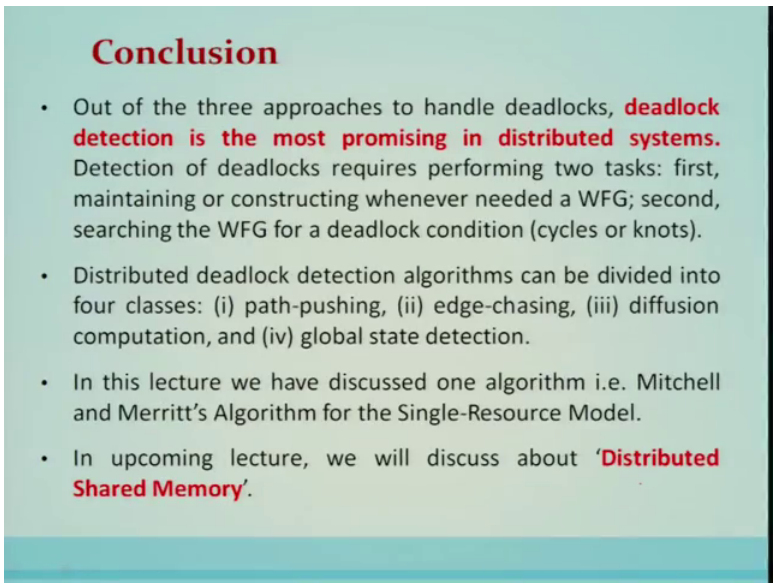
Now, other algorithms which deals with distributed deadlock detection are summarized here in this particular table; now as you can see that all these algorithms uses different strategies of the algorithm distributed algorithm design that is all these strategies we have discussed in previous slide that is either they are path pushing strategies of distributed algorithm design edge chasing diffusion computation and global state detection.

(Refer Slide Time: 35:15)



So, all these algorithms will basically follow that principles which we have covered for example, in a path pushing; it will generate a local wait for graph and push along it path. So, that a particular single node will have the complete information or a complete global state and then using that global state it will. So, based on that scenario these algorithms are basically designed similarly there is an edge chasing one example of edge chasing algorithm which we have seen similar algorithms are also using with a different resource request models.

(Refer Slide Time: 36:29)



Diffusion computation is an algorithm which is using or which is defined on an or request model. Now finally, a global state detection algorithm is given for resource a request model which is P out of Q model. Conclusion out of the 3 approaches to handle deadlocks deadlock detection is the most promising in the distributed systems detection of deadlock requires performing 2 tasks first maintaining the wait for graph and second searching the wait for graph to detect whether there is a situation to detect for a cycle or knot which will in turn depend model to say that whether there is a deadlock or not distributed; deadlock detection algorithms can be classified in 4 different classes that we have seen as push path pushing edge changing decision computation and global state detection.

So, in this lecture, we have discussed one algorithm that is the Mitchell and Merritt algorithm for a single resource model and that particular algorithm is based on the technique which is called in edge chasing in the upcoming lectures we distributed shared memory.

Thank you.