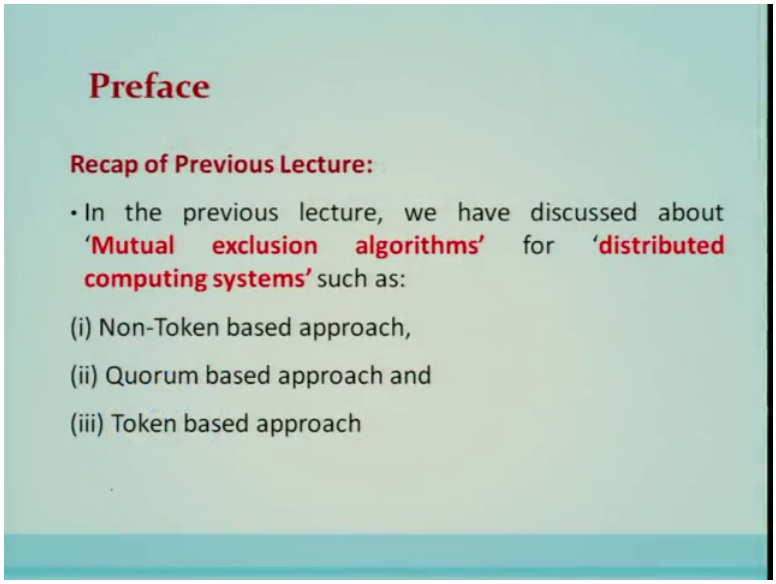
Distributed Systems
Dr. Rajiv Misra
Department of Computer Science and Engineering
Indian Institute of Technology, Patna

Lecture – 10
Consensus and Agreement Algorithms

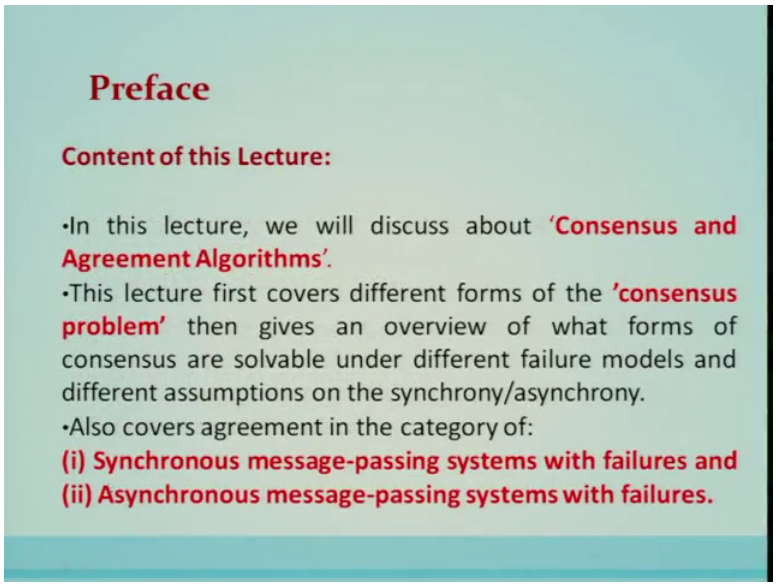
Lecture 10, Consensus and Agreement Algorithms.

(Refer Slide Time: 00:21)



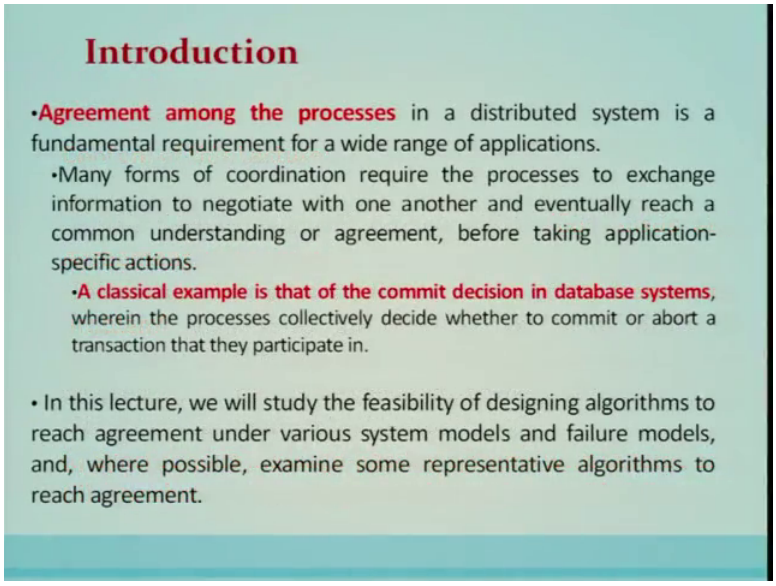
Preface, recap of previous lecture - in previous lecture we have discussed about mutual exclusion algorithms for a distributed computing system such as non-token based algorithms, quorum based algorithms and token based algorithms.

(Refer Slide Time: 00:37)



Content of this lecture, in this lecture we will discuss about consensus and agreement algorithms. This lecture first covers different aspects of consensus problem and then gives an overview of what forms the consensus are solvable under different failure models and different assumptions on synchrony and asynchrony. Also it covers the agreement in the category of synchronous message passing system with failures and asynchronous message passing system with failures.

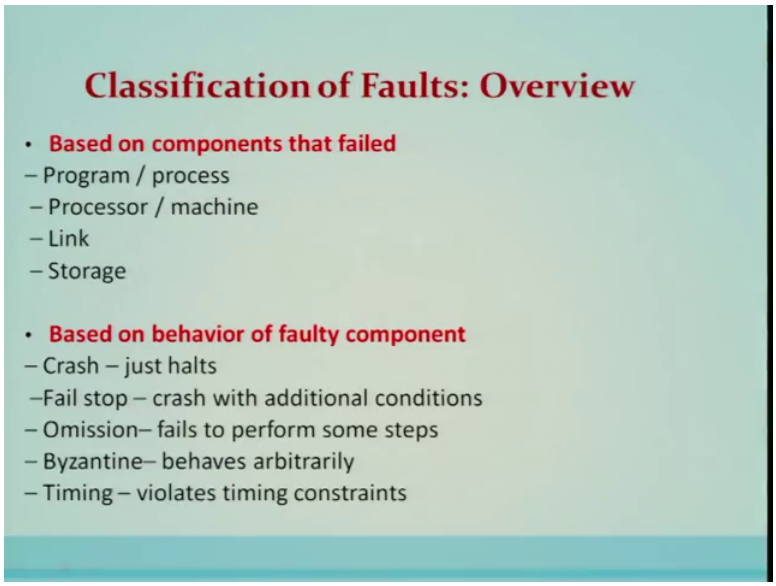
(Refer Slide Time: 01:14)



Introduction, agreement among the processes in a distributed system is a fundamental requirement for a wide range of applications. Many forms of coordination require the processes to exchange information to negotiate with one another and eventually reach. A common understanding or an agreement before taking and application specific actions a classical example is the commit decision in the database system where in the process collectively decide whether to commit or abort a transaction that they participate in.

In this lecture we will study the feasibility of designing algorithms to reach agreement under various system models and failure models and we are possible examine some of the representative algorithms to reach an agreement.

(Refer Slide Time: 02:12)

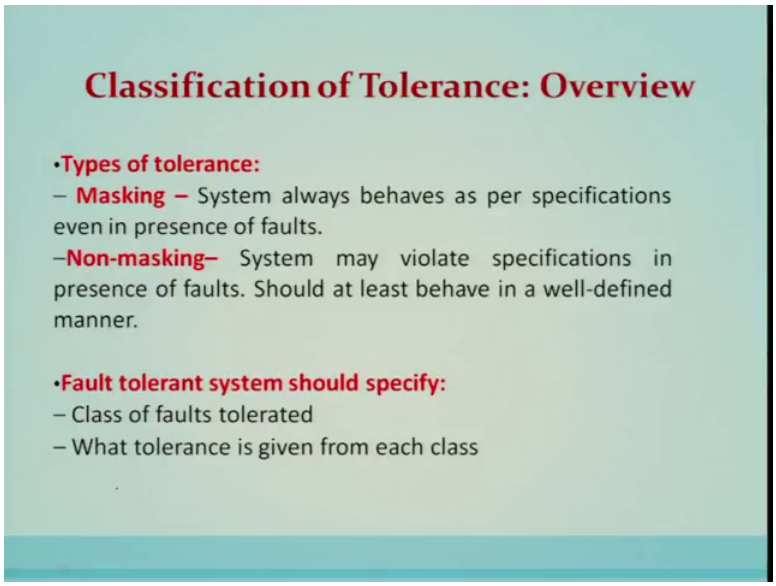


Now, we are going to see some of the fault models. So, let us classify the faults. So, based on the component that failed the components which are basically failing in the distributed systems are classified as either the program or a process or a processor or a machine or a link or a storage. So, they are prone to the failures and we assume that in this discussion that one of these components if it is failing how the algorithms are basically going to lead to a consensus or agreement which is required to run the applications. So, based on the behavior of faulty component, so the fault models different fault models we will be classified as follows.

So, the crash, crash fault there we assume that the system just halts after the problem. Fail stop, fail stop will crash with some additional conditions, crash is means that it will

halt. So, omission means it will fail to perform some by step; that means, some of the step will omit. Byzantine fault behaves arbitrary is the most general fault model. Timing, it violates the timing constraints. These are the different faulty components and they are basically used to model down failure models.

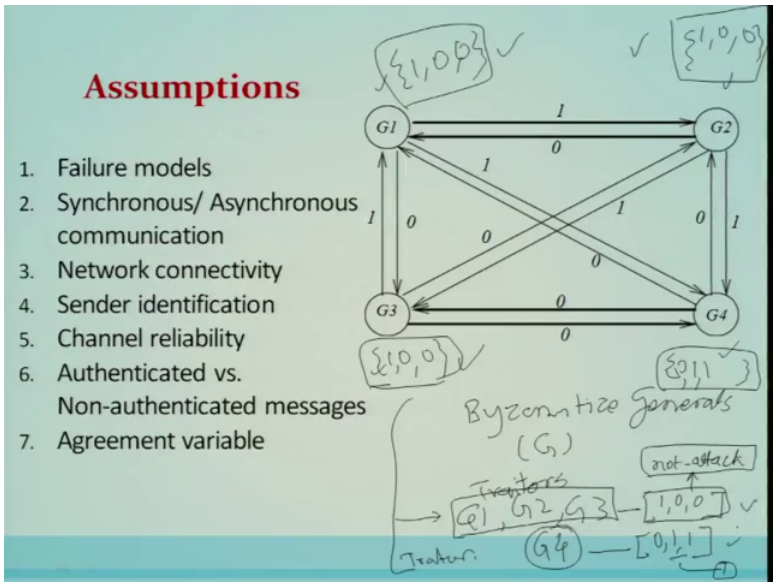
(Refer Slide Time: 03:39)



Now, classification of tolerance type of tolerance masking the system always behaves as per the specification in the presence of faults; that means, faults are masked. Non masking that system may violate specification in the presence of faults should at least behave in a well defined manner.

Fault tolerant system should specify the class of faults tolerated what tolerance is given from each class. So, these are basically some of the requirements.

(Refer Slide Time: 04:10)



Now, as far as different models are concerned different assumptions which we are going to use in this particular algorithm design the assumptions are we have to assume some failure models, so we have to see what are the different failure models possible. Then we have to make an assumption about synchronous and asynchronous type of communication.

Then another resumption we have to make about its network activity sender identification channel reliability authenticated versus non a authenticated message and agreement values. These are the different assumptions which we have to make in the algorithm we are going to discuss one by one in the next coming slides.

So, before that let us see the problems of an agreement in this particular figure which is shown who are here G 1 is basically this particular problem is about byzantine general problem that is why it is basically G, it is represented as G. So, here there are four general and they are leading the troops and that is why they are the group 1, group 2, group 3 and group 4 different generals they are located in the byzantine city and this particular hill is located in Istanbul nowadays.

So, at four different sites where they will not be able to see each other and located on the hill these four different byzantine generals they have to take the decision either to attack or not to attack. And if they take the decision simultaneously then they are going to win if they are not going to take decision simultaneously then basically they are going to

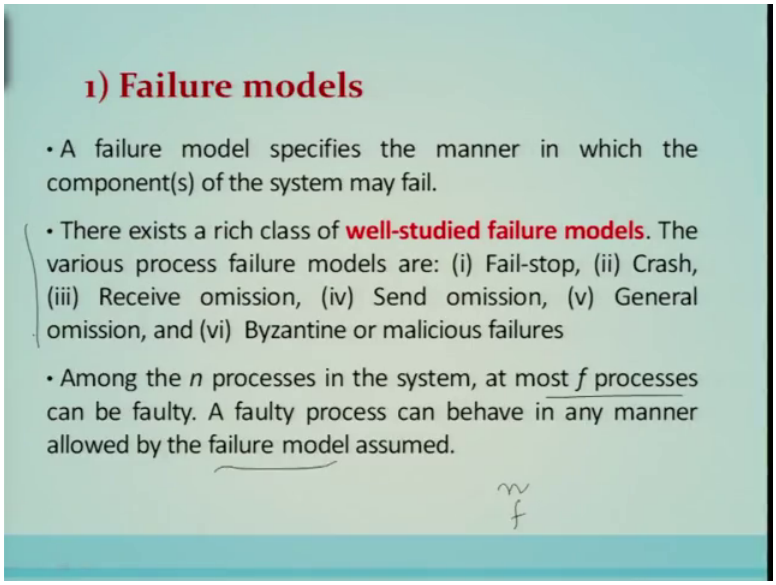
lose. Now some of these particular byzantine generals are the traitors and they will communicate whether to attack or not to attack they have to communicate with the help of message. So, here the message are nothing, but they are communicated through the messengers and the messengers are the messages are lost means if these messengers are caught by the army camp. So, here this G 1 G 2 G 3 G 4 they are byzantine generals and they are basically representing their troops at four different locations and these communication links are nothing, but they are the messengers they are able to they will be sending or they will be communicating with these different generals and once they are communicated securely then they have to follow the action whether to attack or not to attack.

.So, here you can see that there are some generals which are traitors also. So, depending upon how many generals traitor they have to take the decisions - 0 and 1, 0 means no attack and 1 means attack. So, let us assume that, so after the set of message exchanges G 4 will receive the messages 0 then 1 and 1 and G 3 will receive the messages from the other generals as 1 then 0 and 0 and G 2 will receive the messages as one then 0 and 0 and G 1 will receive the messages as 1 then 0 and then 0. Now can we just see that the messages which they received if it is coded in some form; that means, neither means it is very difficult to take this particular decision why because, so here it is 1 0 0 here it is 1 0 0. So, you can see that G 1 G 2 and G 3 they have reached to the same values, but as far as G 4 is concerned G 4 has a different value.

So, if let us say it is interpreted as G 1 G 2 G 3 will have the same value and the majority of it is 0s that is not to be attacked. As far as G 4 is concerned if we go by this then it will have 1 and let us say that it is having decision to attack. So, basically here we can conclude that this is basically a traitor and other non traitor that is the generals they are basically lead to a common value and that is not to attack. So, this basically values are arrived at after several message rounds. So, in still this particular problem is unsolved why because here G 1 G 2 G 3 they reach to common value and G 4 is not. So, everyone is not agreeing on a common value. So, the most of the applications they have to basically depend upon how the agreement is to be arrived at agreement of same value or agreement or set of values. So, these are the problems and this app has a wide applications and we are going to see how this particular agreement problem is going to be solved using algorithms and in what are the models and what are the system models

failure models where we are going to solve them. So, we are going to see these assumptions one by one.

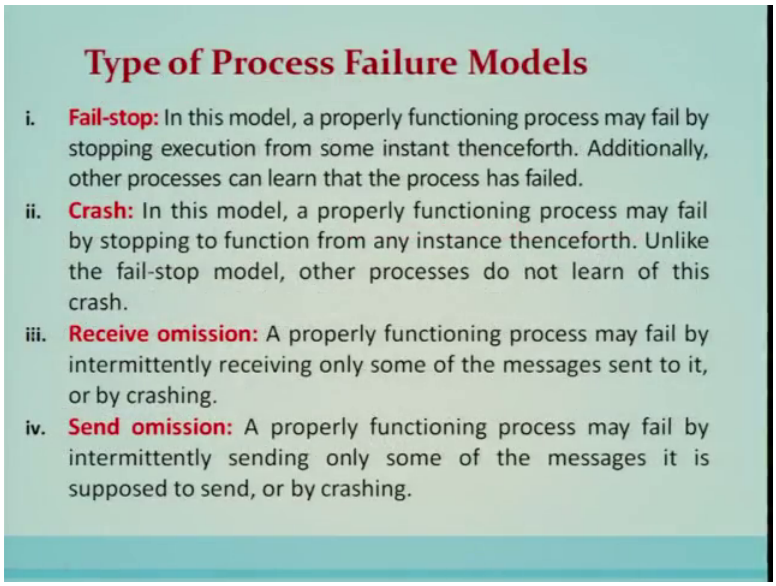
(Refer Slide Time: 10:14)



A failure model, a failure model specifies the manner in which the component of a system may fail. There exist a rich class of well studied failure models the various failure models are fail stop, crash, receive omission, send omission, general omission and byzantine or malicious failures.

Now here in failure models among n processes in the system, we represent it using n small n and we can also assume that there are at most f different processes can be faulty. So, a faulty process can behave in any manner allowed by the failure model which basically is to be assumed in the particular problem setting.

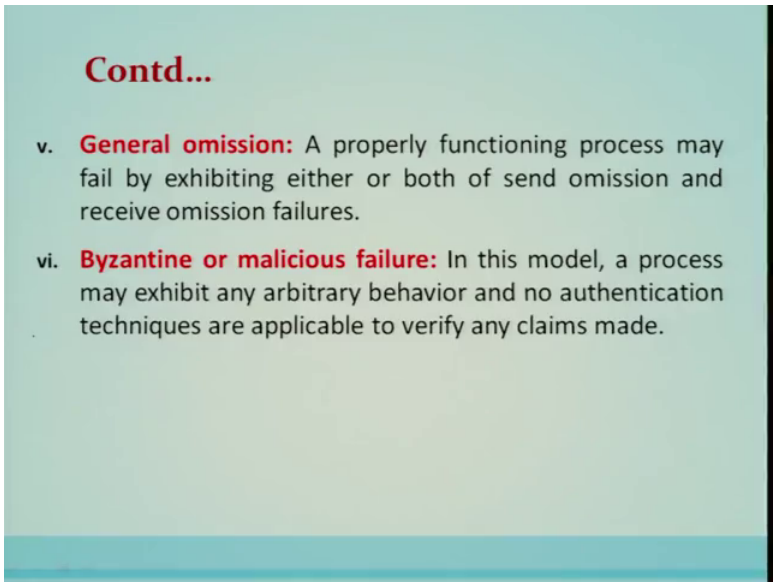
(Refer Slide Time: 11:07)



The types of failure models. So, the first type is called fail stop in this model a properly functioning process may fail by stopping the execution from some instance thenceforth, additionally other processes can learn that the process has failed this is called fail stop.

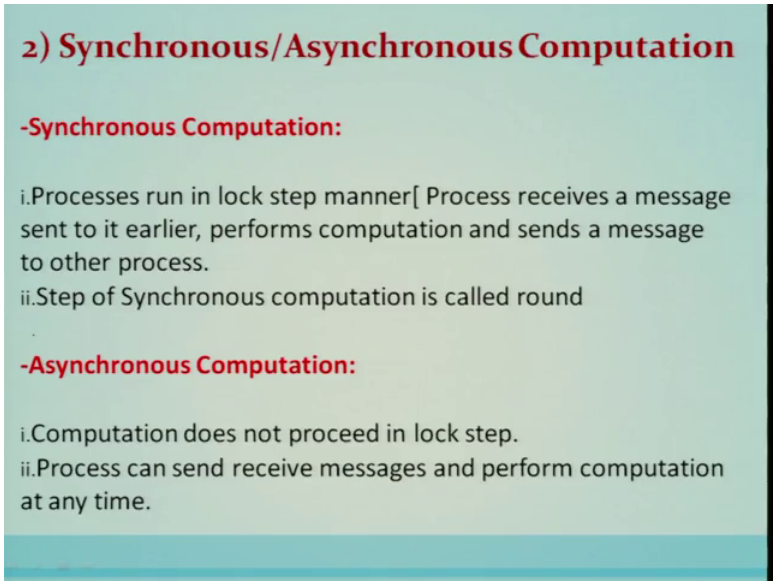
Crash, crash failure model in this model a properly functioning process may fail by stopping to function from any instance thenceforth unlike the failed stop model other processes do not learn of this crash. Third one is receive omission failure model a properly functioning process may fail by intermittently receiving only some of the messages sent to it or by crashing. Send omission a properly functioning process may fail by intermittently sending only some of the messages it is supposed to send or by crashing.

(Refer Slide Time: 12:07)



General omission a properly functioning process may fail by exhibiting either or both of send omission and receive omission failures. Byzantine or malicious in this model a process may exhibit any arbitrary behavior and no authentication techniques are applicable to verify any claims made.

(Refer Slide Time: 12:27)

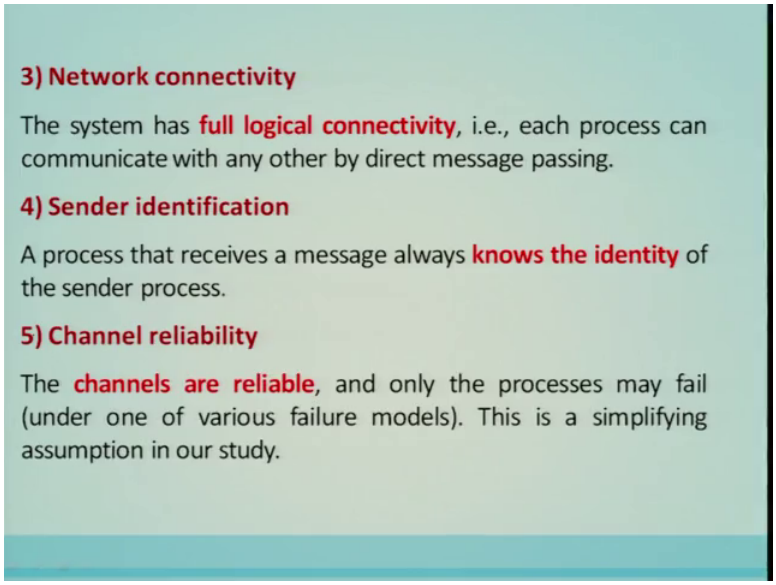


Now, the next assumptions in these algorithms are called about the synchronous oblique asynchronous computations Synchronous computation a process runs in a lock step

manner that is a process receives a message sent to it earlier performs the computation using those message and send the message to other process.

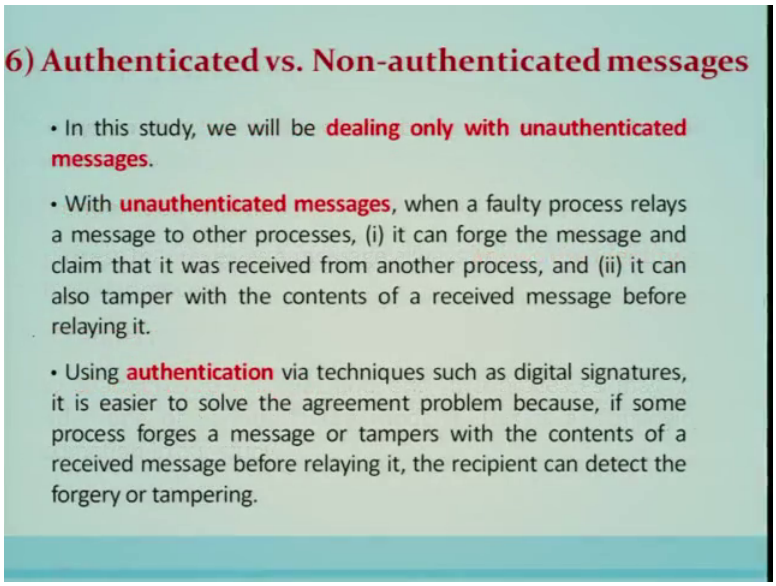
So, this is basically lockstep which the process runs in this particular discrete sequence. So, steps of synchronous computation is called around. A synchronous computation, computation does not proceed in this strict lockstep manner the process can send receive messages and perform the configuration at any point of time there is no bound in these delays of the messages which is assumed in a synchronous communication.

(Refer Slide Time: 13:22)



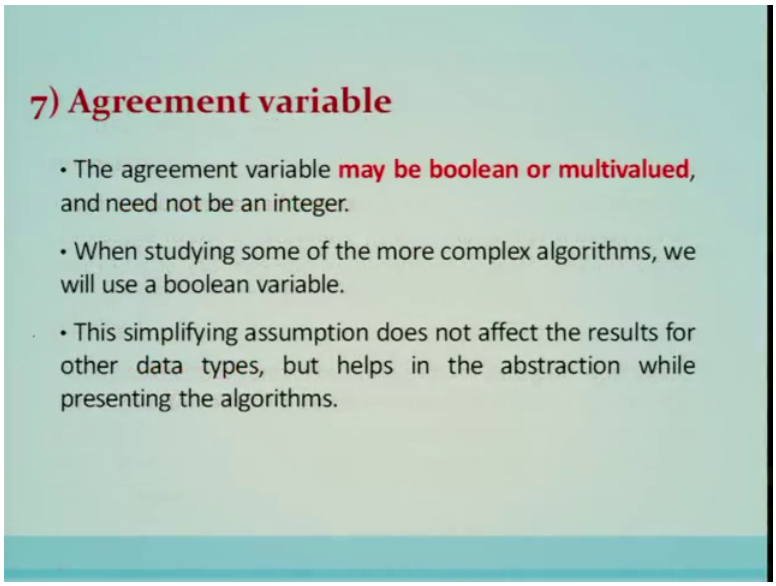
Third type of assumption is called about network connectivity; system has full logical connectivity that is the processes can communicate with a any other process by direct message passing. Fourth assumption is about sender identification a process that receives a message always moves the identity of the sender. Fifth channel reliability the channels are reliable and only processes may fail this is very important assumptions. So, the failure about processes we are going to or processes we have we are assuming under different failure models. So, this will simplify our study about different algorithms in this setting.

(Refer Slide Time: 14:04)



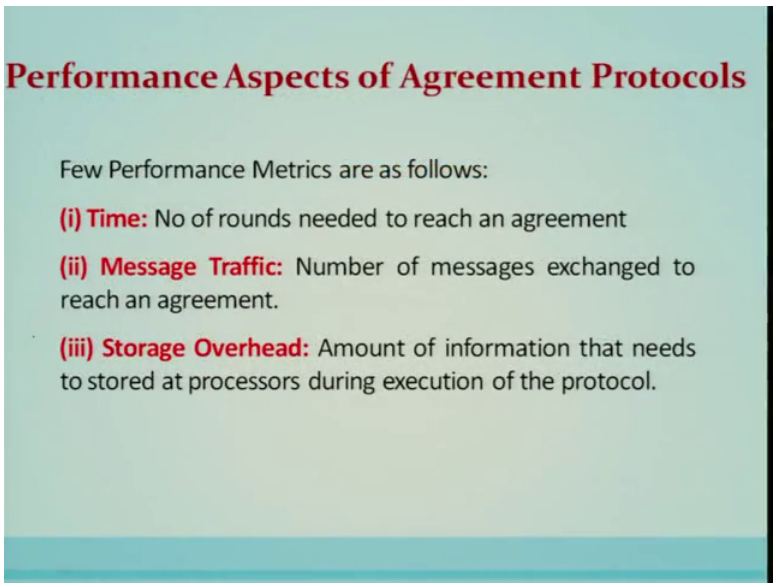
Authenticated versus non authenticated messages: In this part of the discussion we will be dealing only with the unauthenticated messages, with unauthenticated message when a faulty process relays the message to other processes it can forge the message and claim that it was received from another process and it can also tamper with the content of the received message before relaying it. Using authentication via the technique such as digital signature it is easy to solve the agreement problem because if some of the process forges the message or tampers with the content of the received message before relaying it the recipient can detect the forgery or tampering. So, we are going to use only the unauthenticated messages as per the algorithms are concerned.

(Refer Slide Time: 14:53)



Agreement variable, the agreement variable may be a boolean or a multivalued and need not be an integer when studying some more complex algorithms we will assume it as boolean variable. The simplifying assumptions does not affect the result of other data type, but helps in abstraction while presenting the more details of the insight of the algorithm.

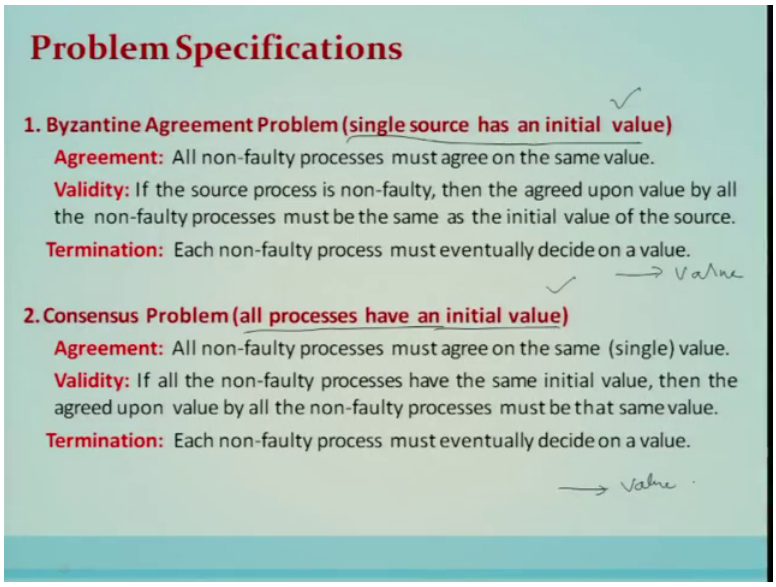
(Refer Slide Time: 15:19)



Performance aspects of the agreement protocol, few performance matrix matrices for the agreement protocols are as follows first is the time that is number of rounds needed to

reach an agreement message traffic that is the number of messages exchanged to reach an agreement, then storage overhead amount of information that needs to store at the processor during the execution of the protocol.

(Refer Slide Time: 15:48)



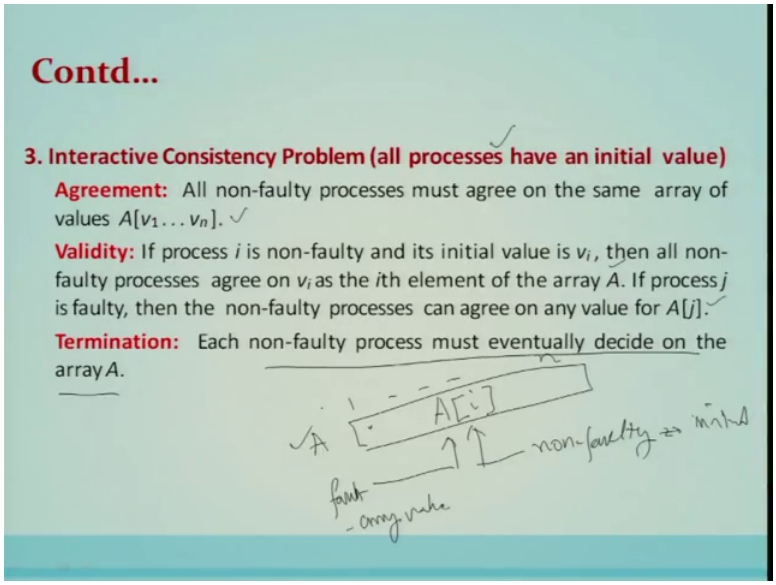
Problem specifications. So, the problem is specification under this agreement and consensus algorithms are as follows - the first one is byzantine agreement problem here the single source has the initial value. So, it has three conditions the agreement that is all non faulty processes must agree on the same value. Validity, if the source is non faulty than agreed upon value by all non faulty processes must be the same as the initial value of the source. Termination each non faulty process must eventually decide on a value.

Consensus problem - all processes have an initial value agreement all non faulty processes must agree on the same that is a single value. Validity, if all the on faulty processes have the same initial value then agreed upon value by all non faulty process must be that same value. Termination, each non faulty process must eventually decide on a value. So, may be that you have seen the difference if you have not noticed then again I am underlining it.

In byzantine agreement only one source has the initial value, in consensus problem all the processes not only single, but all the process have the initial values and in both of the cases they have to decide on a particular value, in both the cases they have to decide on a particular value. So, the only the problem setting is different here in consensus all the

process have their initial values we are in magenta in agreement only the single source has the initial value.

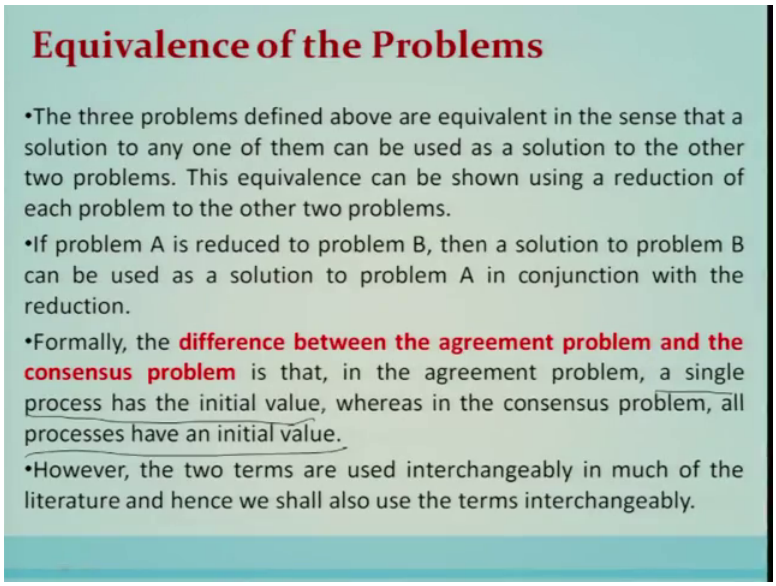
(Refer Slide Time: 17:36)



Third type of problem is called interactive consistency problem here all the processes have the initial values. So, agreement all non faulty process must agree on the same array of values. This you have to notice that this is the difference they are the earlier two methods they have to agree on a single value here they are agreeing on a set of values that is an array of values validity. If a process I is non faulty and its initial value is vi then all non faulty processes agree on v i as ith element of the array. If pj process j is faulty then the non faulty processes can agree on any value for A j.

Termination each non faulty process must eventually decide on an array. So, the array of a particular element shows that this value if they agree if it is a non faulty then they have to agree on that initial value and if it is a faulty then they have to agree on any value. So, this particular array will be built for n different processes 1 and so on up to n and so at the termination all non faulty process must decide on this particular array this array will be exchanged at all the non faulty. So, this is called interactive consistency problem.

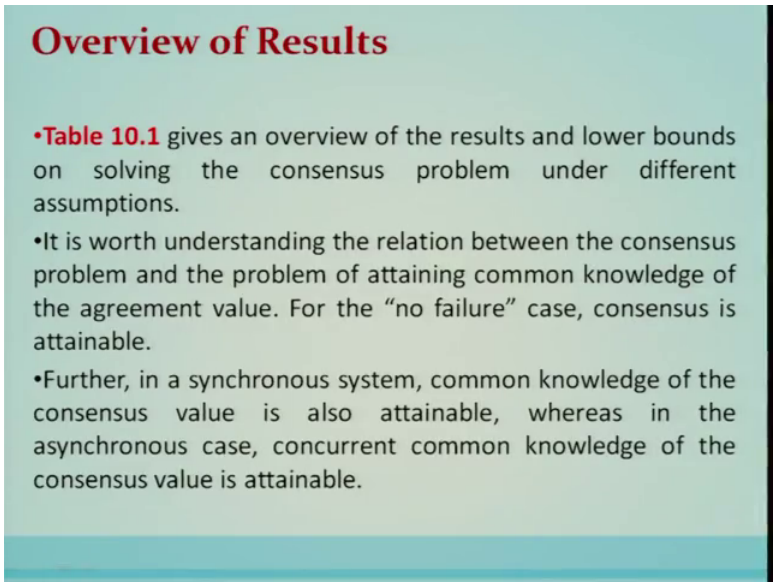
(Refer Slide Time: 19:10)



Among three problems there is an equivalence. So, three problems defined above are equivalent in the sense that a solution to one of them can be used as a solution to the other two problems this equivalence can be shown using a reduction of each problem to the other two problem. For example, if a problem A is reduced to a problem B, then the solution to a problem we can be used to solve the problem A in conjunction with the reduction.

Formally, the difference between the agreement and the consensus is that an agreement problem a single process has the initial value like in a byzantine agreement whereas, in consensus problem all the processes have the initial values. However, and they are equivalent; that means, if you know how to solve a byzantine agreement problem then basically the solution of that byzantine agreement problem can be used to solve the consensus problem and interactive consistency problem.

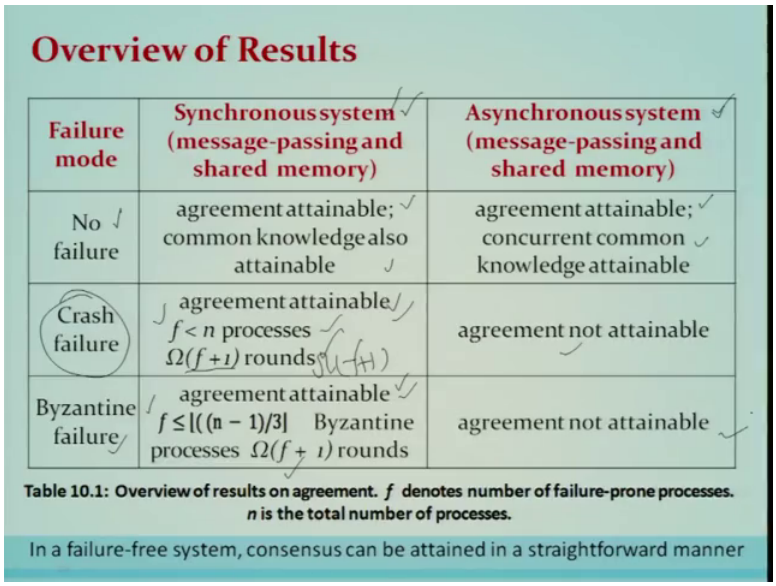
(Refer Slide Time: 20:14)



So, overview of the results, the following table gives an overview of the results and the lower bound on solving the consensus problem under different assumptions. It is worth understanding the relation between the consensus problem and the problem of attaining common knowledge of the agreement value. For the no failure case consensus is attainable that we will see in the next table.

Further in the synchronous system common knowledge of the consensus value is also attainable whereas, in the asynchronous case concurrent common knowledge of the consensus is attainable in no fault situation that we can see in this particular in this figure.

(Refer Slide Time: 20:50)

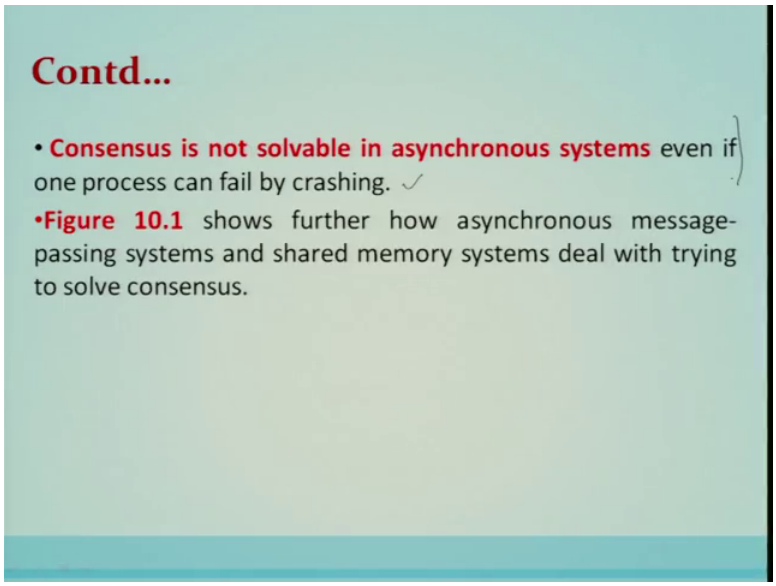


So, if the failure model is no failure and in the synchronous model the agreement is attainable and common knowledge is also attainable. If it is a synchronous message passing and a shared memory system then agreement is attainable and also the concurrent common knowledge is attainable, but if the failure model is a crash fault or a crash failure then agreement is attainable in the synchronous system and f the number of faulty processor is assumed to be less than the total number of processors. And it will be resolved that is agreement can be attainable in the f plus 1 of a lower bound number of rounds; that means, minimum f plus 1 round is required to basically achieve the agreement.

However, in a synchronous model even for the crash fault or a crash failure model agreement is not attainable at all. If the fault model is byzantine then the agreement is attainable in the synchronous system where the faulty number of processor is less than or equal to the floor of n minus 1 divided by 3. And this particular agreement is achieved in the lower bound of f plus 1 number of round where f is the faulty number of nodes.

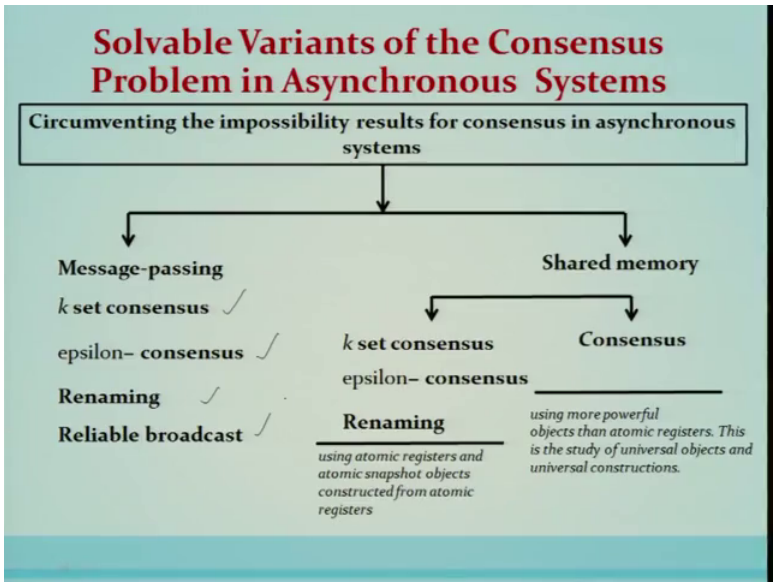
However in a synchronous model the agreement under byzantine failure is also not attainable because if it is not attainable in the crash fault then obviously, it will not be attainable in the byzantine fault.

(Refer Slide Time: 22:38)



So, consensus is not solvable in a synchronous system even if one processor can fail by the crashing. So, the next figure for that shows how a synchronous message passing system and a shared memory system deal with trying to solve the consensus problem.

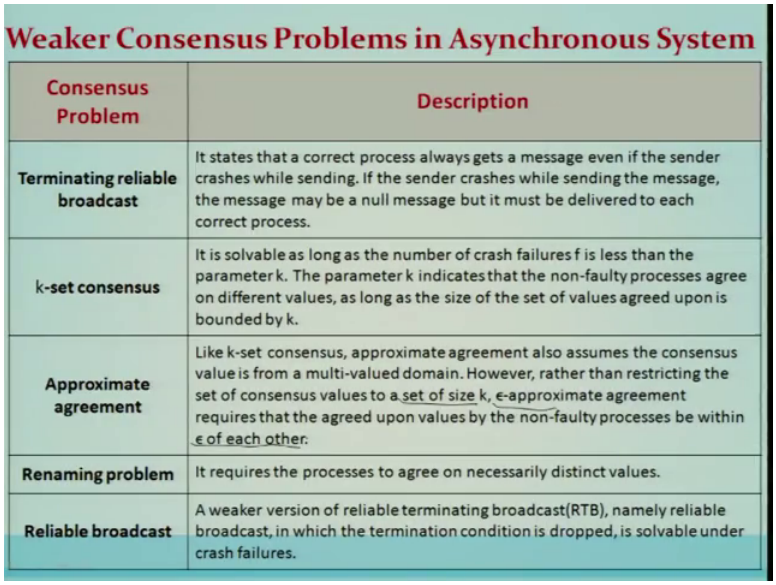
(Refer Slide Time: 22:57)



Now, since you know that if possibility of the consensus problem under asynchronous system, so what we can do is basically we can solve a variance of this consensus problem in this model that is in a synchronous system and that is shown here. So, under message passing system a variant of the consensus problem is called k set consensus, epsilon

consensus renaming and reliable broadcast that we will see in the next few slides in the more detail about them.

(Refer Slide Time: 23:33)



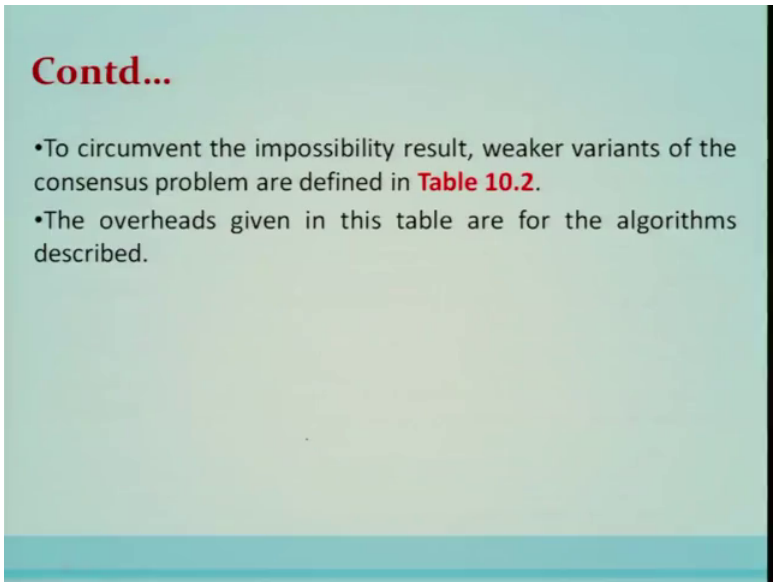
Weaker Consensus Problems in Asynchronous System

Consensus Problem	Description
Terminating reliable broadcast	It states that a correct process always gets a message even if the sender crashes while sending. If the sender crashes while sending the message, the message may be a null message but it must be delivered to each correct process.
k-set consensus	It is solvable as long as the number of crash failures f is less than the parameter k. The parameter k indicates that the non-faulty processes agree on different values, as long as the size of the set of values agreed upon is bounded by k.
Approximate agreement	Like k-set consensus, approximate agreement also assumes the consensus value is from a multi-valued domain. However, rather than restricting the set of consensus values to a set of size k, ϵ-approximate agreement requires that the agreed upon values by the non-faulty processes be within ϵ of each other.
Renaming problem	It requires the processes to agree on necessarily distinct values.
Reliable broadcast	A weaker version of reliable terminating broadcast(RTB), namely reliable broadcast, in which the termination condition is dropped, is solvable under crash failures.

Now, the weaker consensus problem in synchronous system we are just trying to understand what these problems are a terminating reliable broadcast it is states that a correct process always gets a message even if the sender crashes while sending it. k-set consensus it is solvable as long as the number of crash failures f is less than the parameter k the parameter k indicates that non faulty processes agree on a different values as long as the size of the set where our values agreed upon is bounded by k that is called k set consensus. Approximate agreement like k set consensus approximate agreement also assume the consensus value is from multi value domain; however, rather than restricting the set of consensus values to a set, set of size k it says that epsilon approximate agreement requires that the agreed upon value by the non faulty processor is within epsilon of each other.

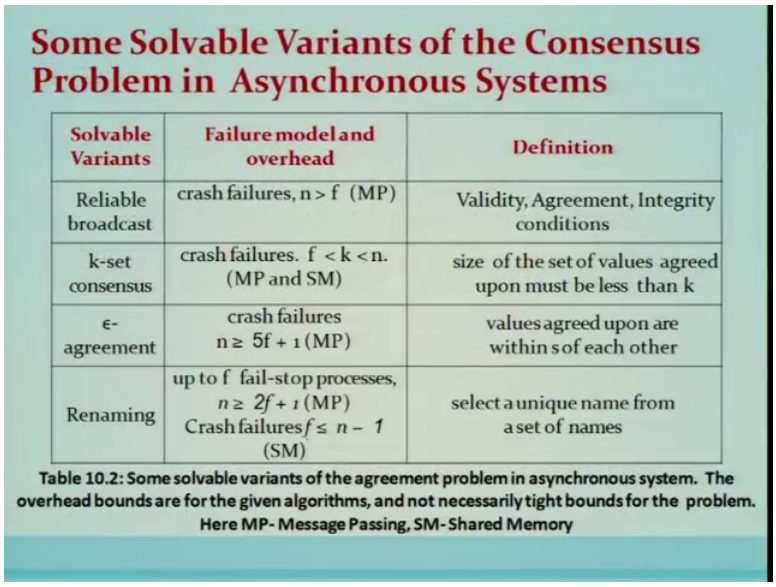
Renaming problem it requires the processes to agree on necessarily a distinct value. A reliable broadcast a weaker version of a reliable terminating broadcast, namely the reliable broadcast in which the termination condition is dropped is solvable under the crash fault.

(Refer Slide Time: 24:54)



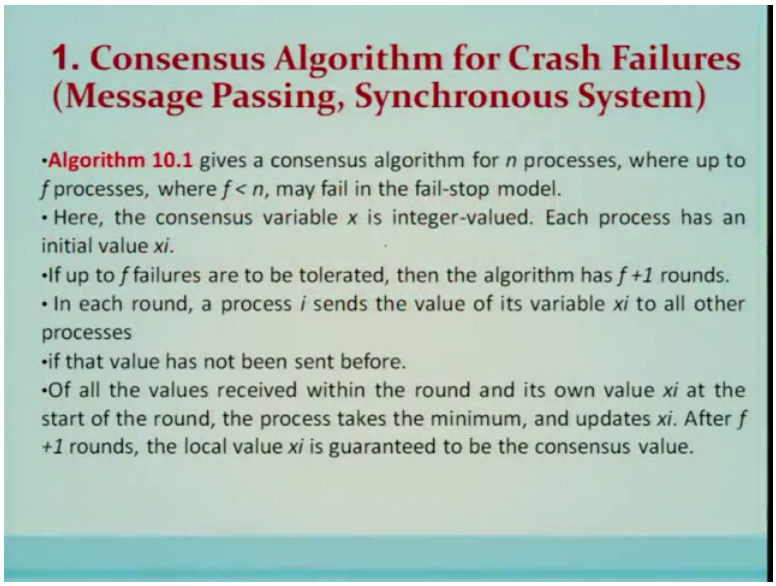
To circumvent the impossibility results the weaker variants are defined in the next table.

(Refer Slide Time: 25:01)



So, reliable broadcast failure model if it is crash fault it requires the over the condition n is greater than f similarly k set consensus. Then basically the value of k should be more than f and should be bound less than less than n. Epsilon agreement crash failures this particular model is working n is should be greater than equal to 5 f plus 1. Renaming up to f fail stop processes it can sustain and n should be greater than 2 f plus 1 and for the crash fault f is less than n minus 1.

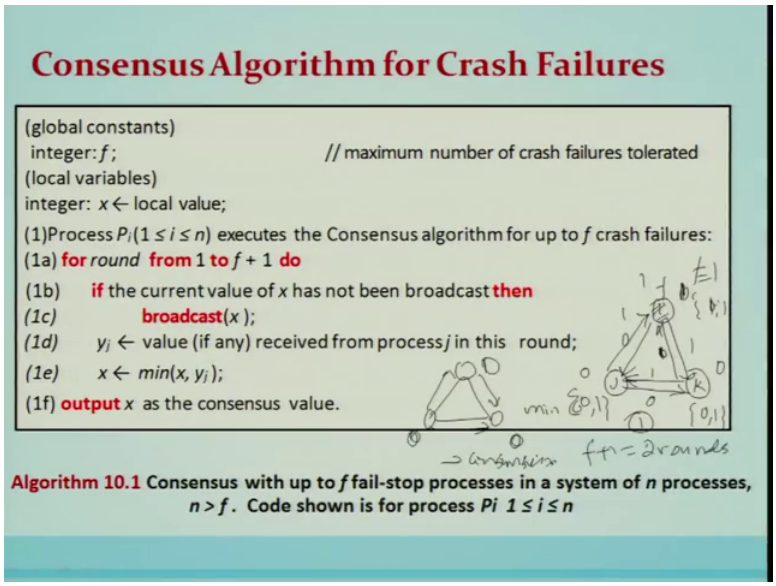
(Refer Slide Time: 26:01)



Agreement in synchronous message passing systems with failures: Consensus algorithm for crash failures message passing synchronous system. So, algorithm given in the next slide it gives the consensus algorithm for n processes where up to f processes where f is less than n may fail in a fail stop failure model. Here the consensus variable x is integer value; each process has initial value xi. If up to f failures are to be tolerated than algorithm has f plus 1 rounds, in each round a process I sense the value of its variable x I to all other processes if that value has not been sent before.

So, of all the values received within that round and its own value xi at that start of the round the process takes minimum and updates xi occur f plus 1 rounds the local value x I guaranteed to be the consensus value.

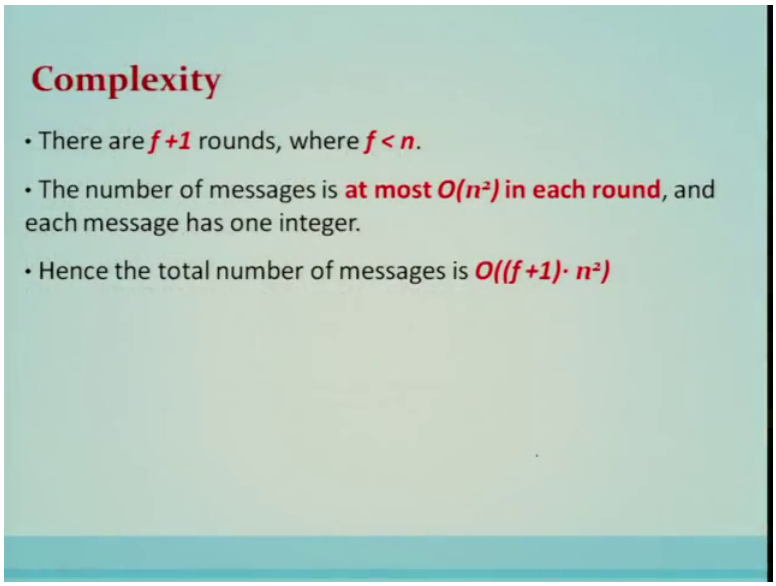
(Refer Slide Time: 27:06)



Let us understand it using this particular algorithm that is explained earlier. So, we can understand that if let us say that we have three different processors and let us say one is faulty. So, f is equal to 1 here in this case. So, the agreement requires f plus 1 that is equal to two rounds. If it is faulty let us say it will send 0 to 1 process and 1 to another process i j and k. Now, on receiving one on receiving 0 it will broadcast 0 over here and this particular process on receiving 1 it will broadcast 1 over here. So, this will complete one round in this one round and this particular process on receiving 1 it will send 1 over here and this on the receiving 0 it will send 0 over here. So, you can see that here it will receive it will receive 0 and 1 and this will receive 0 and 1, this will also receive 0 and 1 and the minimum if you take this will be having 0 0 and 0.

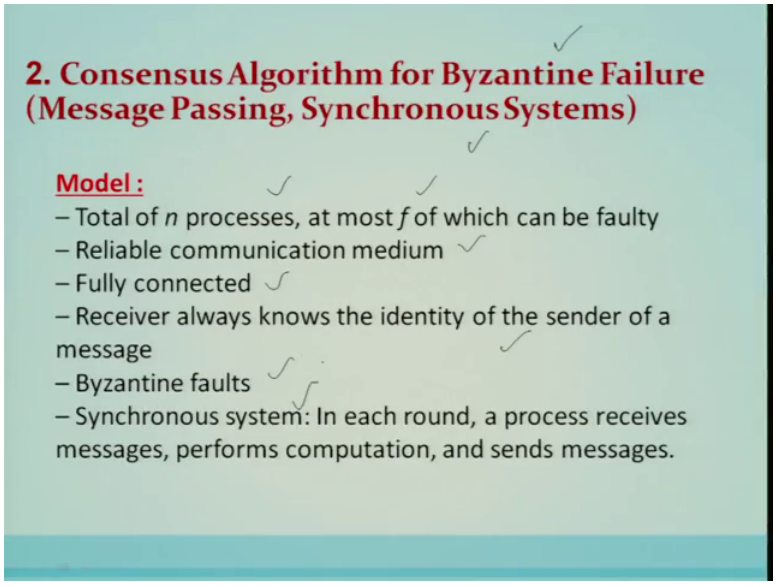
So, if let us say that it sends 1 back to him if it receives one then let us say it receives one in that case. So, it is 0 0 and 1. So, this is round number 1. So, similarly if you take another round of these particular messages and if this is having value 1 this is having value 0 this is having 0 if they exchange with each other and take the minimum then all of them will have the value 0 and they will reach to a consensus after two rounds.

(Refer Slide Time: 29:27)



So, the complexity of this particular algorithm is it requires f plus 1 rounds where f is less than n and the number of messages is order n square in each round and each message has one integers hence the total number of messages is f plus 1 into n square f plus 1 is the total number of rounds and in each round n square messages are required.

(Refer Slide Time: 29:57)

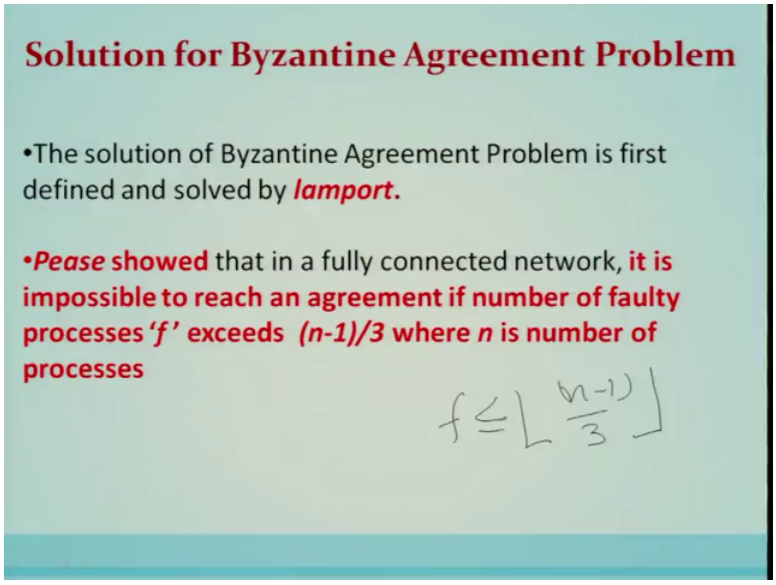


Now, the next most important algorithm in today's lecture is given here that is consensus algorithm for byzantine failures this algorithm is given by Leslie Lamport this is called Shostak Pease Lamport algorithm this is for byzantine failures. So, it is assuming a

synchronous system and the failure model is byzantine and we are going to see this particular algorithm.

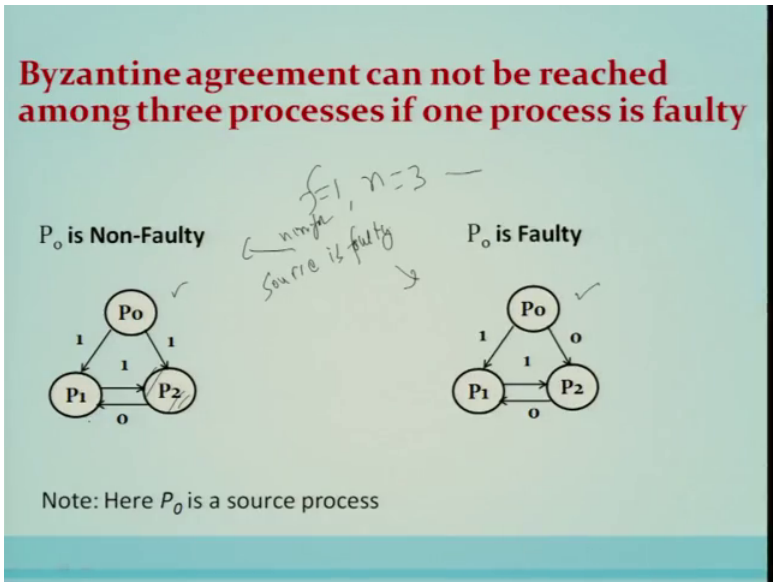
So, in this algorithm the model which is assumed is saying that there are total n processes and f can be the faulty the communication is reliable and it is fully connected topology receive always knows the identity of the sender and here the fault model is assumed as very general fault model that is byzantine and the synchronous system in each round a processor receives a message performs computation and sends. So, synchronous concentration model is assumed here.

(Refer Slide Time: 31:04)



So, solution of the byzantine agreement is first defined by first defined and solved by Leslie Lamport. So, Lamport Shostak Pease algorithm it is. So, Pease shows that show that fully connected network it is impossible to reach an agreement if the number of faulty processor f exceeds n minus 1 divided by 3 that is f should always be less than or equal to n minus 1 divided by 3.

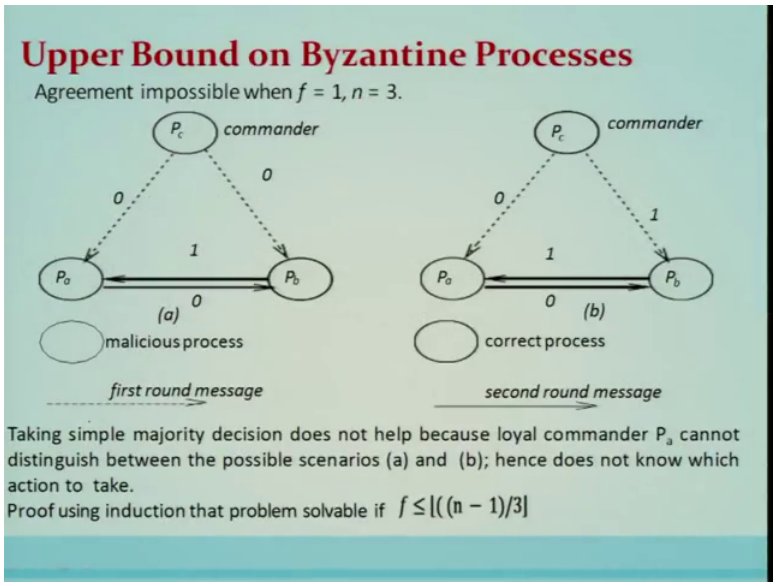
(Refer Slide Time: 31:36)



So, let us take this particular example whether this example shows that the condition where f is less than n minus 1 by 2 is violated over here; that means, if f is equal to 1 and n is equal to 2 this particular assumption is violated; that means, n minus 1 by 2 is not 1 in that case, but we are assuming 1 so obviously, as per the previous condition agreement byzantine agreement is not possible and we can see in this particular example.

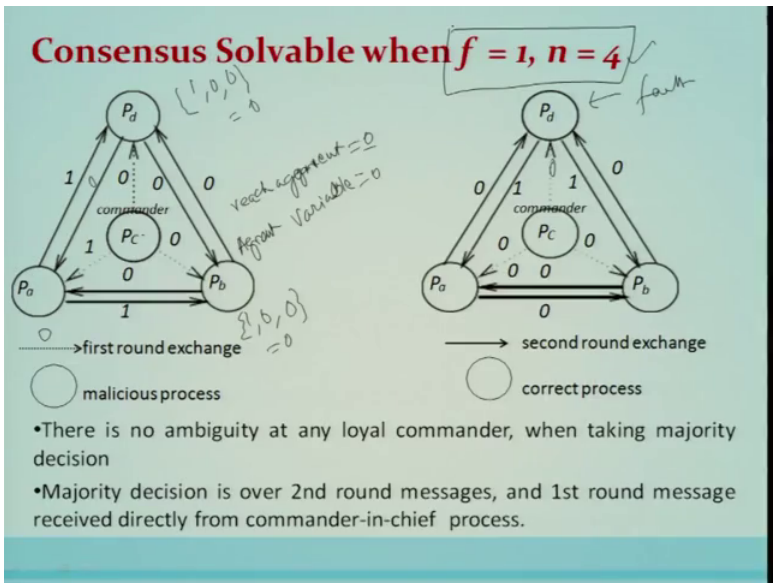
Here P 0 is faulty is non faulty and here P 0 is faulty so that means P 0 is the source, the source is faulty here in this case and source is non faulty in the other case. So, source is non faulty, but some other process is faulty let us say that P 2 is faulty. So, P 1 will send because it is non faulty same values to P 1 and P 2 and as far as the P 2s concerned it will send a different value because it is a faulty.

(Refer Slide Time: 32:59)



So, this agreement will not be reached here in this particular example. So, same example we can see that agreement is not possible.

(Refer Slide Time: 33:07)



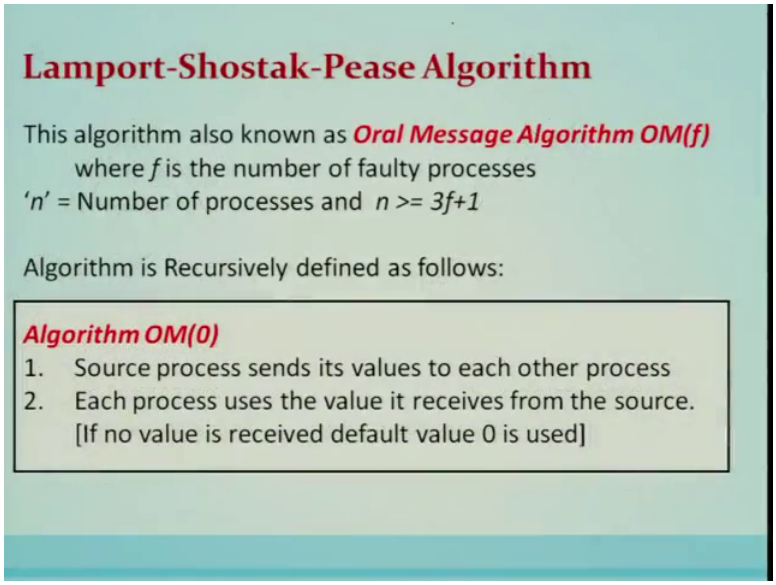
Now, agreement is possible when f is equal to 1 and the total number of processor is 4. So, agreement we can see how it is possible we can see about the commander P c. So, this is the source it will send the message 0 since it is faulty. So, it will send 0 to P d 0 to P b, but 1 to pa in the first column. So, P a after receiving this one it will send one to both

the neighbors, similarly P b after receiving 0 it will send 0 why because it is not faulty, similarity P d will send after receiving 0 at both the ends.

So, if we take these values which will be received here it is 1 and basically it is 0 and this is also 0. So, the majority is basically 0 here in this case here also if you see the values 1 0 and 0. So, the majority is 0 and here also majority is 0. So, in this particular case even if the source is faulty, it will reach to an agreement, reach an agreement and that value will be agreed upon value or agreement variable will be equal to 0.

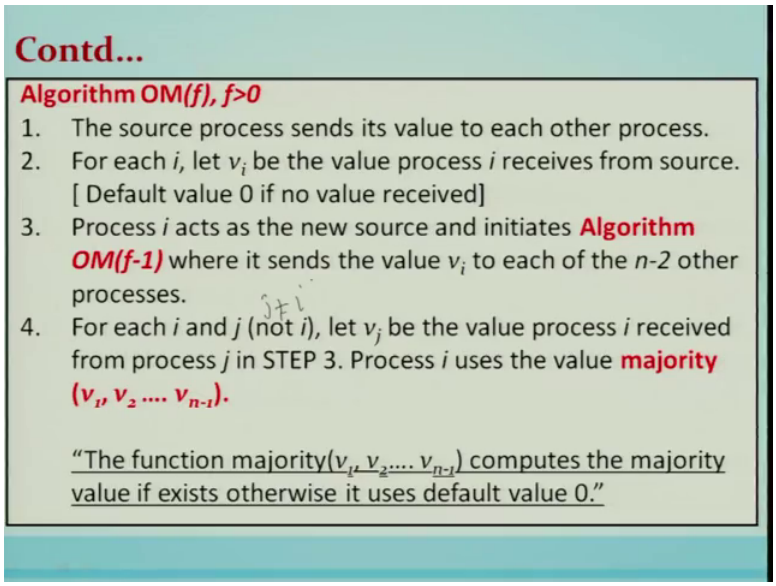
Similarly, here in this example also let us assume that this is there are some other node is faulty in that case. So, for example, you will receive 0 and this receives yeah this is faulty in this case why because after receiving 0 after receiving 0 it will send 1 and 1 in both the cases. So, we will see that in the same manner this also we will reach to an agreement and. So, as far as Lamport substrate Pease condition is concerned if the condition is satisfied with the number of faulty processor then the agreement is possible.

(Refer Slide Time: 35:32)



So, the algorithm is called as the Lamport's for as take Pease algorithm this algorithm also known as oral message algorithm OM f, f is the number of faulty processors. So, the number of processors n should be greater than or equal to 3 f plus 1. The algorithm is recursive and the base of the recursion that is OM 0 says that the source process sends its values to each other process. Now each process uses its value, value it receives from the source if no value is received the default 0 is assumed.

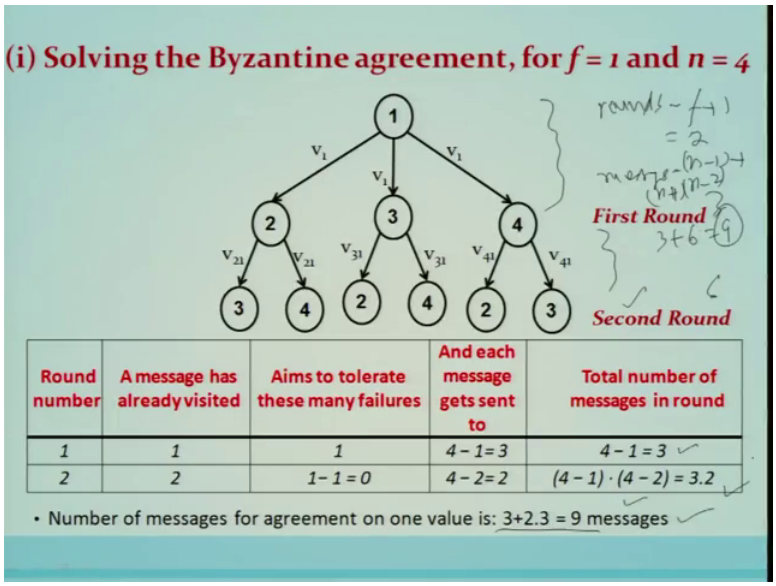
(Refer Slide Time: 36:12)



Now, recursion, recurse procedures of this algorithm OM, OM f, f is greater than 0 then the first steps is that source process sends its values to each other process now for all for each I let vi be the value the process I receives from the source now default 0 if the value is not received. Process ix as the new source and initiate algorithm OM f minus 1 where it sends the value vi to each of n minus 2 process.

Now finally, after this particular for loop is over in step number 2 then fourth step says that for each i and j, not i which is not equal j is not equal to i j is not equal to let v i be the be the be the process I received from the process j in step number 3, process i uses, the values using the majority function we want to v n minus 1 the function majority computes the majority value if exist otherwise it uses the default value as 0. So, this majority function is application dependent.

(Refer Slide Time: 37:35)

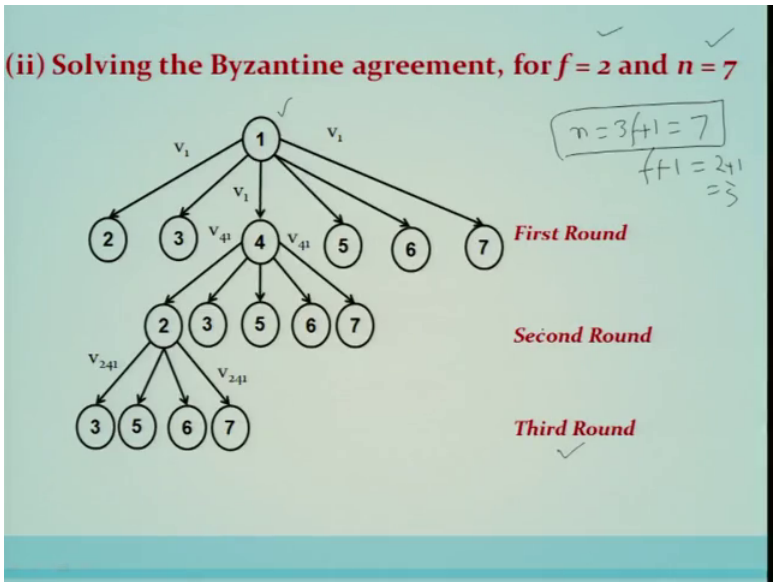


So, this basically will be the algorithm which is applied under the byzantine fault model and under the synchronous communication model, system model.

So, as we have seen that f is equal to 1 n is equal to 4 the same algorithm we have applied in the previous slide. Now, let us see how many messages, how many rounds will be there. Now as you know that it will require how many two rounds will be required. So, in each round the total number of messages, total number of messages in each round will be three for example, a source will send to its neighbors. Similarly, in the second round now each of these will send to its corresponding neighbors. So, basically 6 messages in second round and 3 messages in the first round, so total nine messages will be required here and the total number of number of rounds will be the 2 that is f plus 1 that is equal to 2 and the messages will be will be s equal to n minus 1 plus n minus 1 multiplied by n minus 2, that is n minus 1 means 4 minus 1 minus 1 that is 3 n 3 plus 6 that is 9 total 9 massages will be there.

So, if we generalize this particular total number of messages. Now if f is equal to, if f is equal to 2 then n is equal to 3 f plus 1 that is equal to 7.

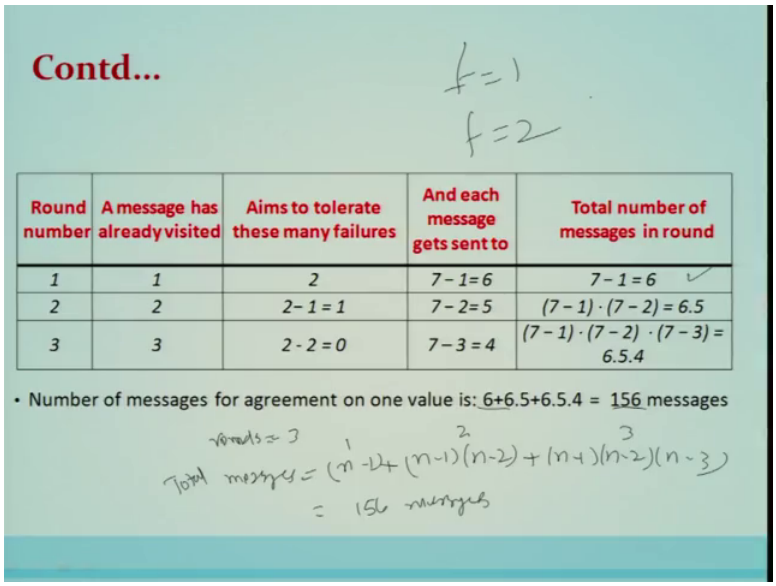
(Refer Slide Time: 39:20)



So, 7 is required and for the number of faulty processes are 2. So, this is particular model or the in this particular case byzantine agreement is possible. And how many number of rounds it is going to take? f plus 1 that is 2 plus 1 that is 3 rounds. So, here just see that in this particular picture 3 rounds are shown. And let us understand that in the first round the source will send the message to all other neighbors except itself. So, it will send to 2 3 4 5 6.

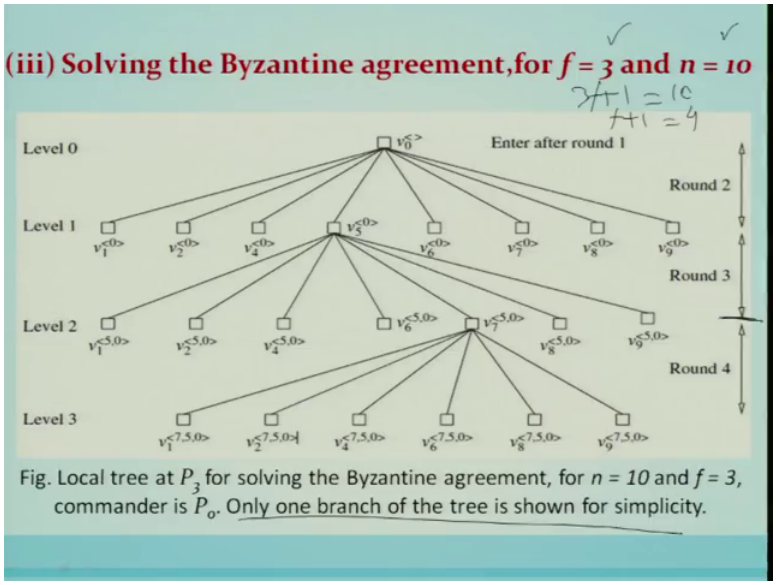
Now, every node will do this way; that means, we have to see how the node 4 is now doing it. Node 4 after receiving this v 1 it will send to its neighbors. So, its neighbors other than the previously defined neighbors or previously received messages or not to be included in that. So, it will send the message for 2 3 5 6 7, now among them to will in turn will send to its neighbors other than from where it has earlier previously received the message that is 3 5 6. So, third round when every node is complete the algorithm will finish and let us analyze how many number of messages will be communicated over here.

(Refer Slide Time: 41:02)



So, the number of, so total number of rounds as you have seen is equal to 3 and the messages required will be equal to n minus 1 plus n minus 1 times n minus 2 plus n minus 1 times n minus 2 times n minus 3 and if you see because this is round number 1, this is round number 2 and this is round number 3 different messages. So, total number of messages we have to sum total number of messages which are basically communicated in each round. So, if you sum them it comes out to be 156 different messages. So, we have seen two examples where f is equal to 1 and f is equal to 2, now we are going to see a more complicated example where f is equal to 3.

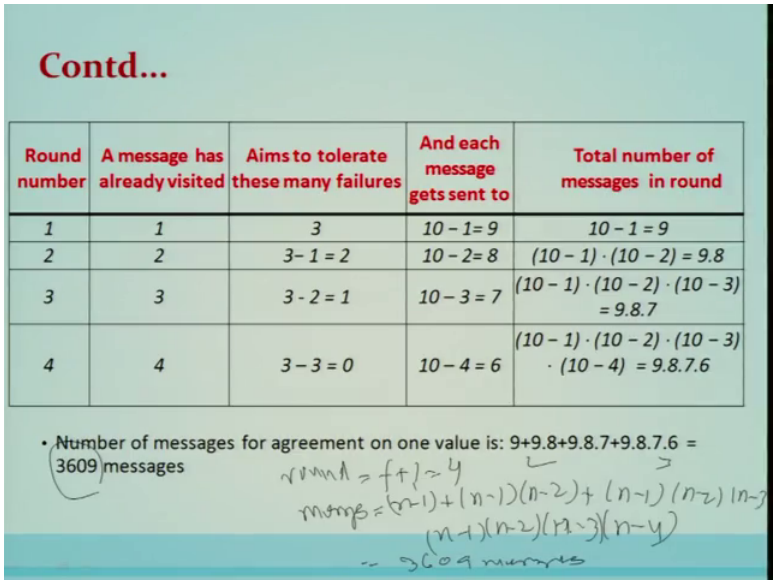
(Refer Slide Time: 42:02)



When f is equal to 3 the total number of nodes will be 10 3 f plus 1 that is equal to the n and the number of rounds will be f plus 1 that is 3 plus 1 that is 4.

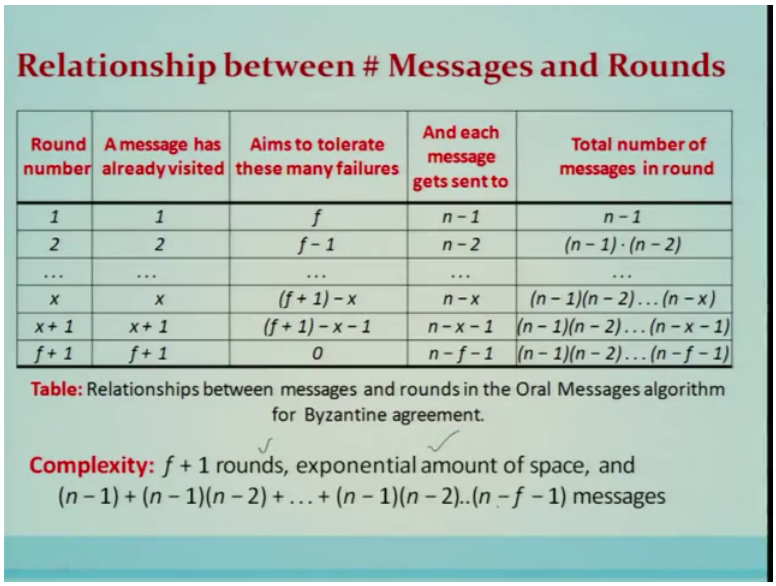
So, just see that in this particular figure number of rounds are 4 and as we have seen in the previous example that up to round 3 we have seen similarly the round 4 will also continue that every node will send the message to its basically neighbors and will not send to the other neighbors who have previously sent the message to that particular node. So, this example as written over here only one branch of the tree is shown for the simplicity what you can do is you can explore the complete examples at your end.

(Refer Slide Time: 43:05)



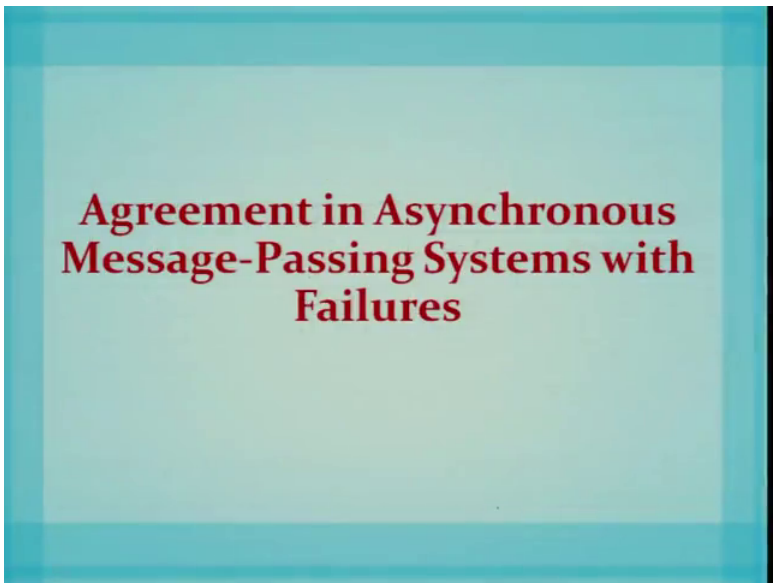
Similarly, if we see the message complexity number of rounds as I told you is equal to f plus 1 that is 3 plus 1, 4 and the message total number of message will be n minus 1 plus n minus 1 times n minus 2 plus n minus 1 times n minus 2 times n minus 3 1 2 and 3. So, and the fourth round will take n minus 1 n minus 2 n minus 3 and n minus 4. So, total number of messages if you count they comes out to be 3609 messages in this particular example.

(Refer Slide Time: 43:52)



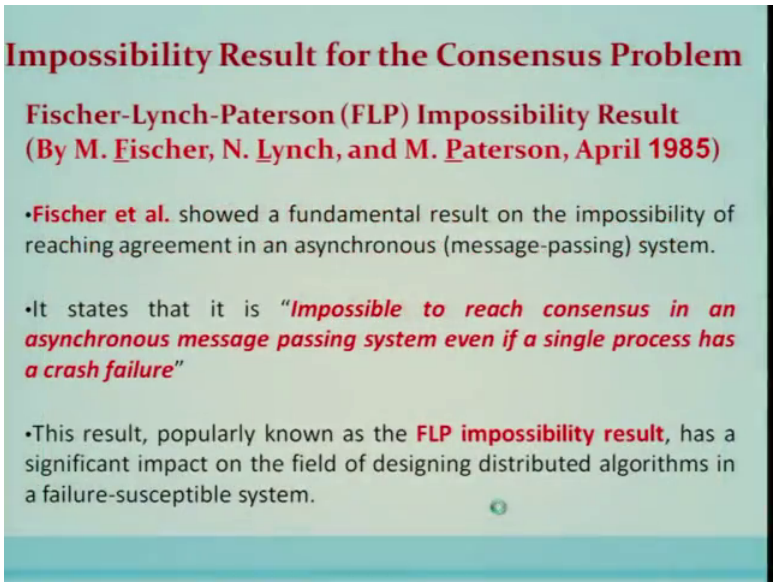
So, we can generalize these particular formula and so number of rounds is very simple calculation that is f plus 1 rounds, but as far as the messages are concerned it goes to an exponential that is why it is called exponential algorithm because the exponential of the order messages are required hear in this particular example, in this particular.

(Refer Slide Time: 44:18)



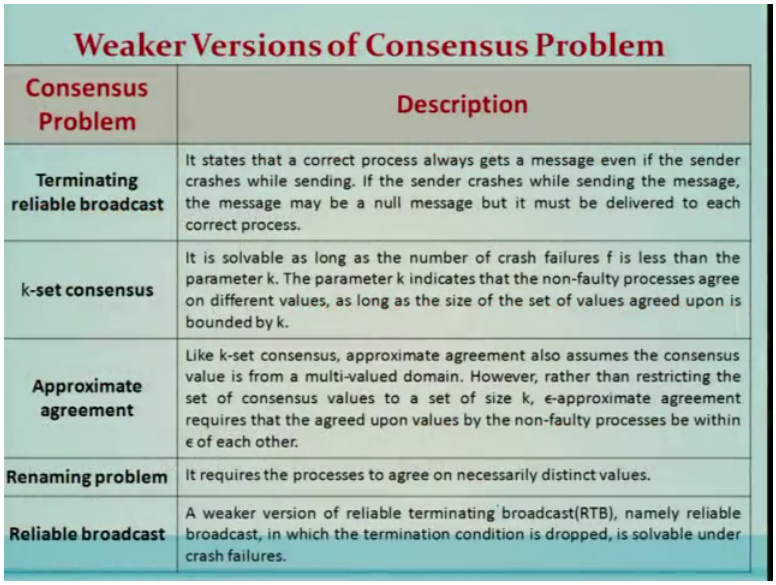
Exponential amount of space and messages are required.

(Refer Slide Time: 44:25)



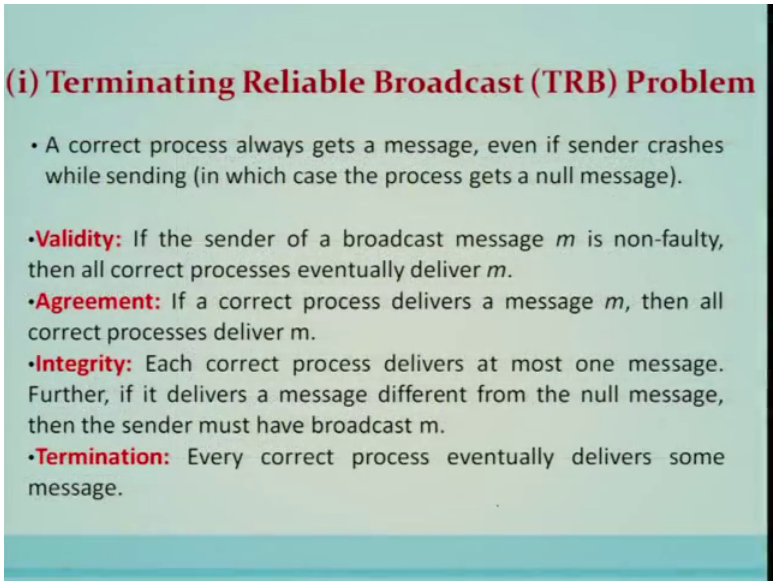
Now, agreement in asynchronous message passing system with failures impossible to result for consensus problem Fischer, Lynch and Paterson, FLP impossibility result it is called. So, in 1985 this particular impossibility result is called FLP impossibility result Fischer showed that the fundamental result on the impossibility of reaching an agreement in a synchronous system it states that it is impossible to reach consensus in an asynchronous message passing system even if a single message single process is has a crash failure, this result popularly known FLP impossibility to result has a significant impact in the field of designing distributed algorithm in a failure susceptible systems.

(Refer Slide Time: 45:10)



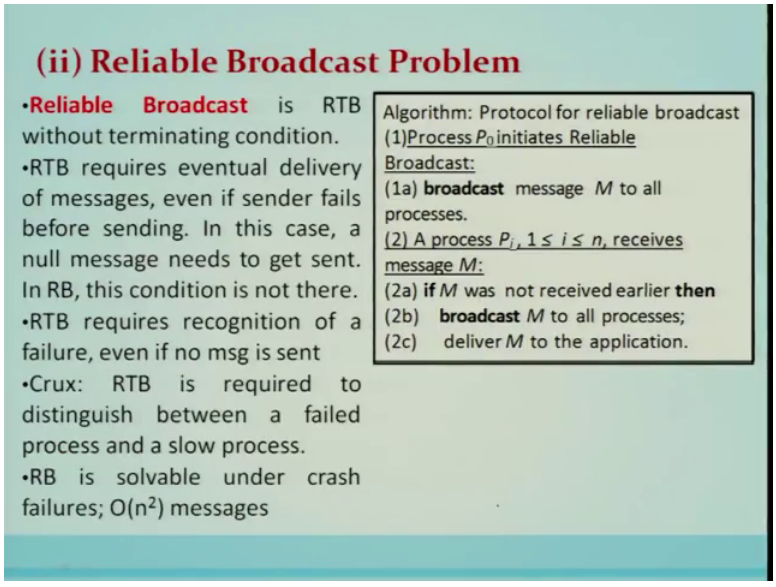
So, that is why weaker versions of consensus problem in a synchronous system a synchronous model of communication is basically devised and these are called terminating reliable broadcast k-set consensus, epsilon approximate agreement, renaming problem and reliable broadcast. Terminating reliable broadcast problem, a correct process always gets a message even if the sender crashes while sending it.

(Refer Slide Time: 45:31)



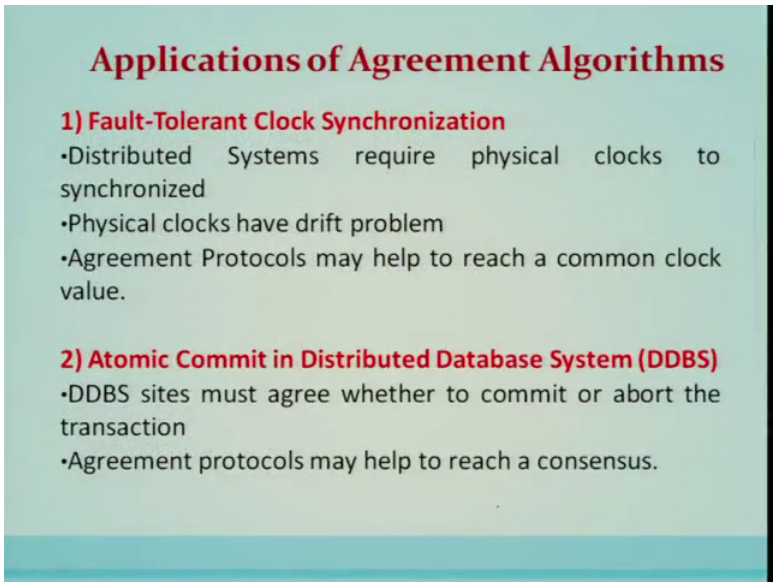
So, validity, if the sender of the broadcast message m is non faulty than all the correct process eventually deliver m.

(Refer Slide Time: 45:51)



Next problem is called reliable broadcast problem reliable broadcast problem is without terminating condition RTB requires that eventual delivery of the message even if the sender fails before sending it in this case the null message needs to be sent. In reliable broadcast this condition is not there, in reliable in RTB requires the recognition of the failure even if no message is sent. So, is solvable under the crash failure of the order n square.

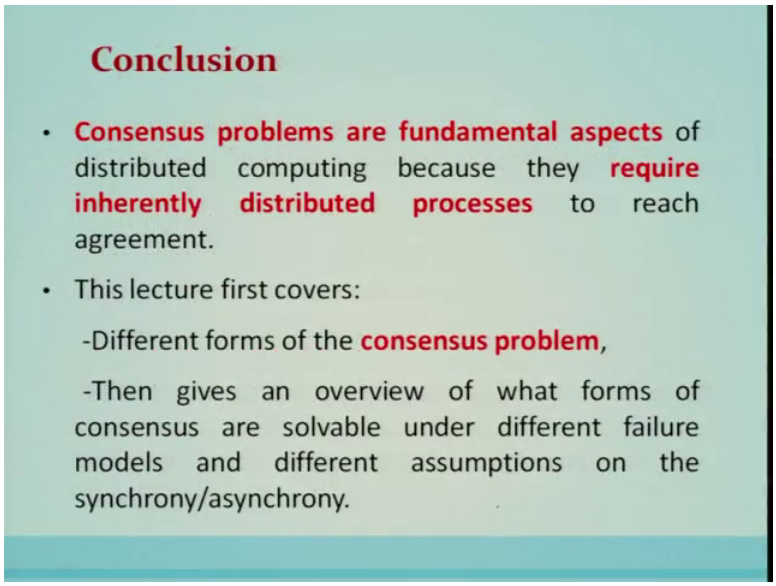
(Refer Slide Time: 46:21)



Now, applications of this agreement algorithm are many, so some of them are listed over here as first application is called as a fault tolerant clock synchronization distributed system required physical clock to be synchronized. So, agreement protocol may help them to reach a common clock value.

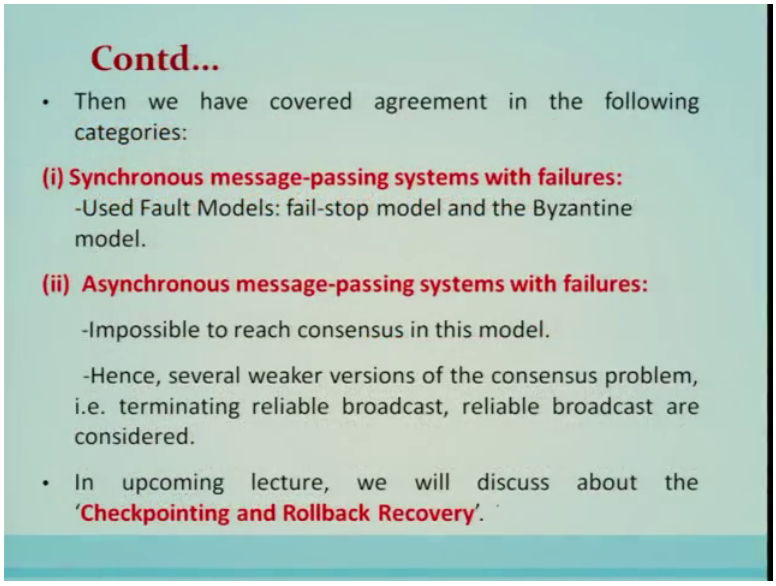
Atomic commit in distributed database system. So, distributed databases sites must agree on whether to commit or to about the transaction, agreement protocol may help to reach a consensus.

(Refer Slide Time: 46:54)



Conclusion, consensus problems are fundamental aspects of distributed computing because they require inherently distributed processes to reach an agreement or a consensus and which is essential in many of the application. So, this lecture covers different forms of consensus problem, then gives an overview of what forms the consensus are solvable under different models, failure models and different computation models.

(Refer Slide Time: 47:26)



Then we have covered the agreement n the following categories synchronous message passing system with failures use the fault model fail stop and byzantine models. We have seen two different algorithms a synchronous message passing system with failures we have we have shown the FLP impossibility condition in this problem setting, it is impossible to reach the consensus in this model. Hence several weaker versions of the consensus problematic terminating reliable broadcast, reliable broadcast are considered. In upcoming lectures we will discuss about check pointing and (Refer Time: 48:00).

Thank you.