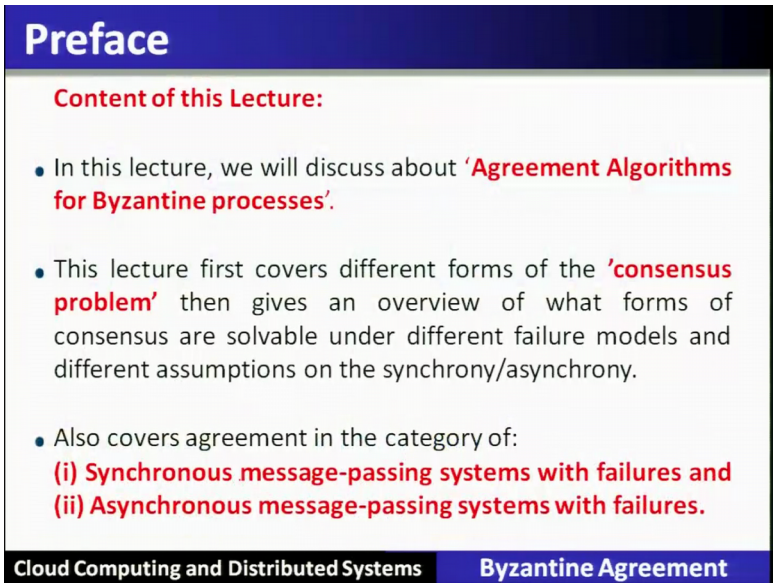
Cloud Computing and Distributed Systems
Dr. Rajiv Misra
Department of Computer Science and Engineering
Indian Institute of Technology, Patna

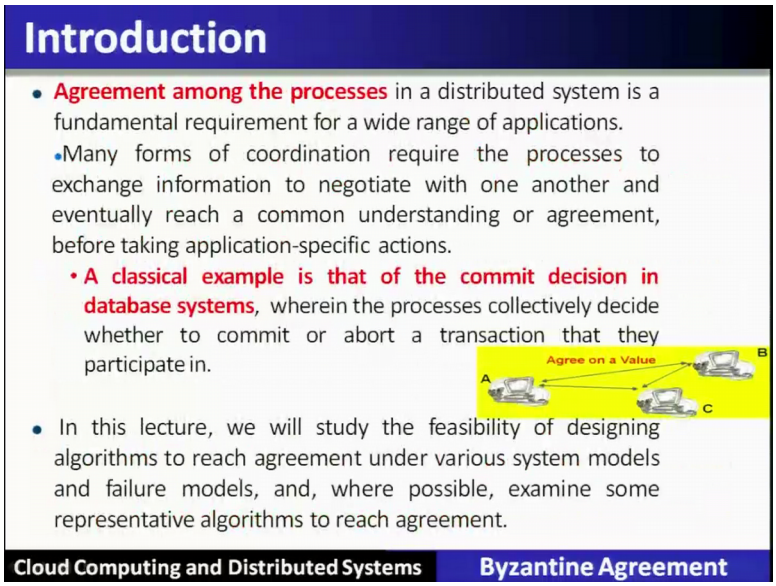
Lecture-14
Byzantine Agreement

(Refer Slide Time: 00:15)



Byzantine Agreement: Pre phase content of this lecture, we will discuss about agreement algorithms for byzantine processes. We will cover different forms of consensus problem. We will give the overview what forms of consensus are solvable under different failure model and under different assumptions on synchrony versus asynchrony. Also we will cover the range of agreement problems, such as in the category of synchronous message passing systems and asynchronous message passing systems with the failure models.

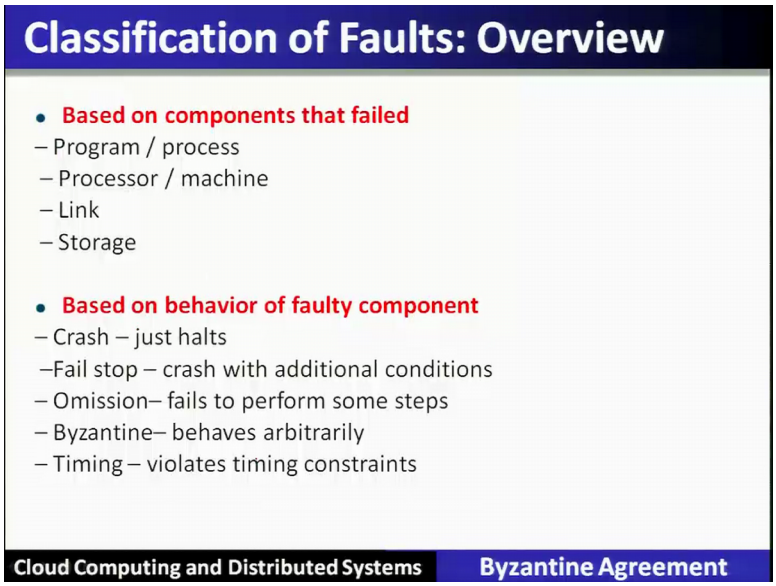
(Refer Slide Time: 01:02)



Introduction, the agreement among the processes in a distributed system is one of the fundamental requirements for wide range of applications. Many forms of coordination requires the processes to exchange the information to negotiate with one another and eventually reach a common understanding or the agreement, before taking the application specific action.

The classical example is that of commit decision in any database system. There processes collectively decide whether to commit or abort a transaction which they are participating in. This lecture we will study the feasibility of designing the algorithm to reach to a consensus under various system models and fault models. And examine the representative algorithms for ensuring the agreement in the system.

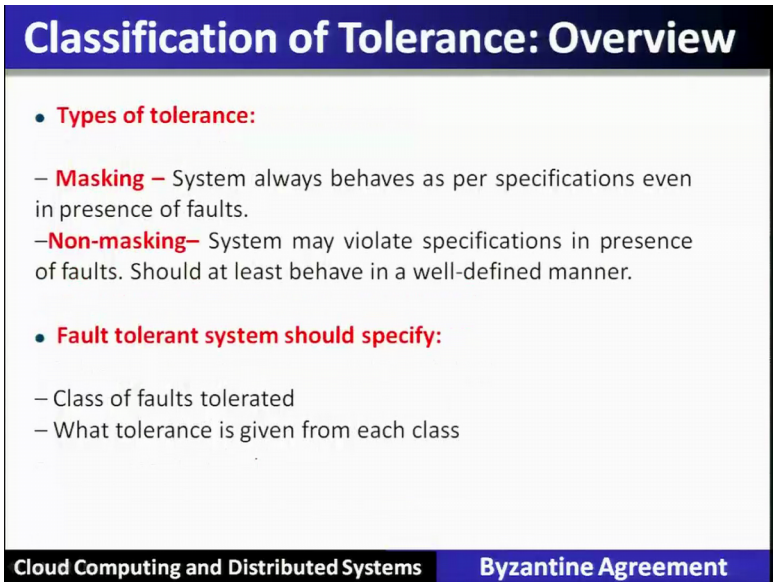
(Refer Slide Time: 02:11)



Before that, let us see some of the basics in contrast in relation to the faults. Classification of the faults: Based on different components, which are failed such as program or a process can fail processor or machine can fail the link that is the network link can also fail or the storage can fail. There are different possibilities of failure. Now, based on these failure possibilities, there are different fault model which are now evolved in this particular study of subsystems. So, the first model is called crash word or it is also called as fail-stop. So, that particular model will when suffers from the failure that is called a crash or a fault, it will stop functioning.

This is one of the most simplest model fault model which can be assumed. We have seen the algorithms in the previous lessons. The, another type of model fault model is called omission fault. Here it fails to perform some steps, the next model which is very general kind of model that is called byzantine fault model. It behaves arbitrarily; that means, it is behavior is not predictable, sometimes it may omit the messages. Otherwise, it is functioning ok, sometimes it is meant malfunctioning or maliciously functioning. This is all term in a very general fault model called byzantine fault model.

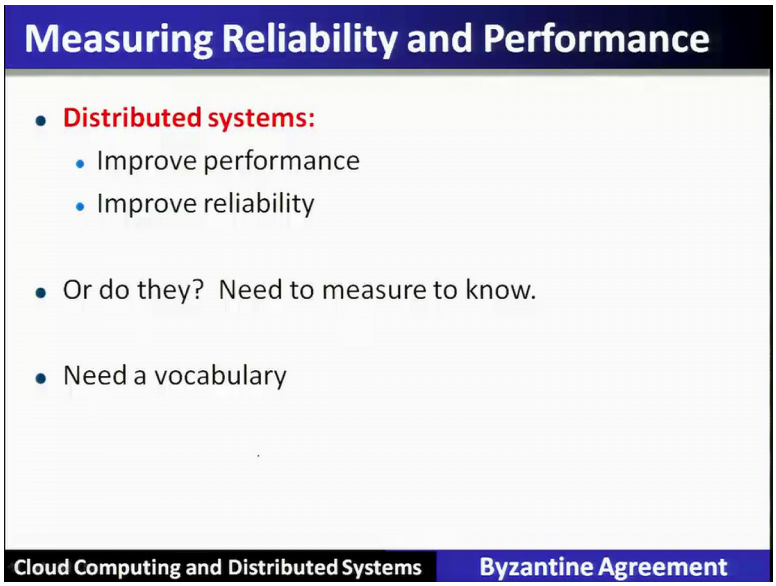
(Refer Slide Time: 04:21)



Timing which violates the timing constraint, is also a type of fault model. In this discussion we will consider the byzantine fault model, and we will see the agreement: what are the algorithms to deal with that. Now, there are terms which are called tolerance.

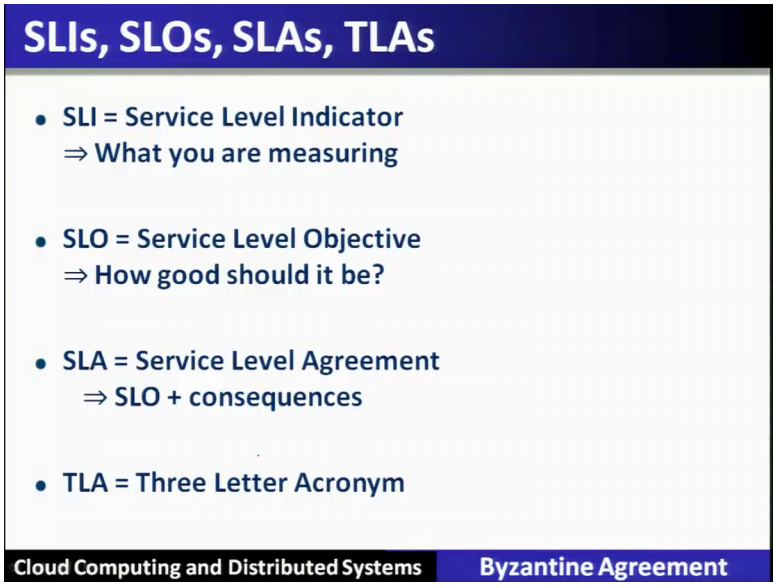
So, masking means system always behaves as per the specification even in the presence of faults. Non masking is system may violate the specification in the presence of fault, should at least behave in a well-defined manner. So, the fault tolerance system should specify what class of faults which are tolerated and what tolerance is given for each class.

(Refer Slide Time: 05:05)



Furthermore, to measure the reliability and performance in a distributed system; that means, there are various measures for performance and reliability, which are to be ensured while giving the services. So, we will understand we will see how to measure them that is performance and the reliability.

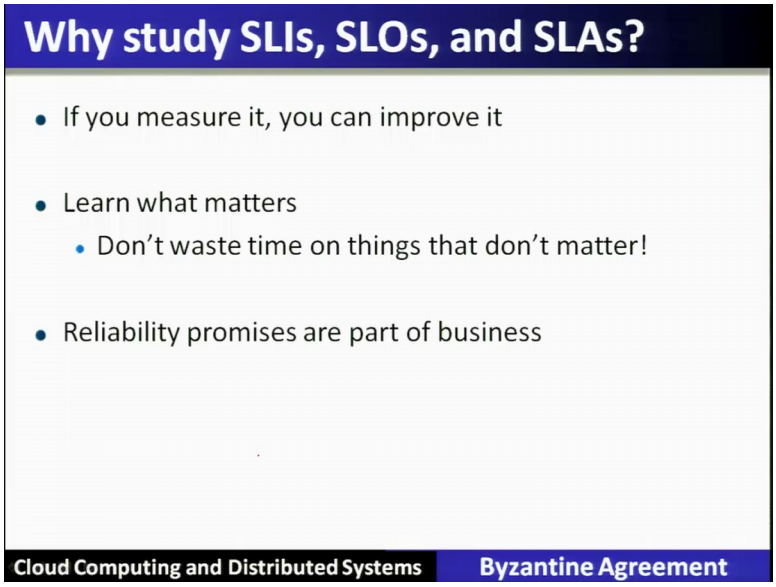
(Refer Slide Time: 05:30)



So, there are certain terms which are used by the industry when they provide the services in the form of a cloud or in a distributed system. Let us quickly review them. SLI, that is service level indicator, will indicate what are you measuring. Service level, objective

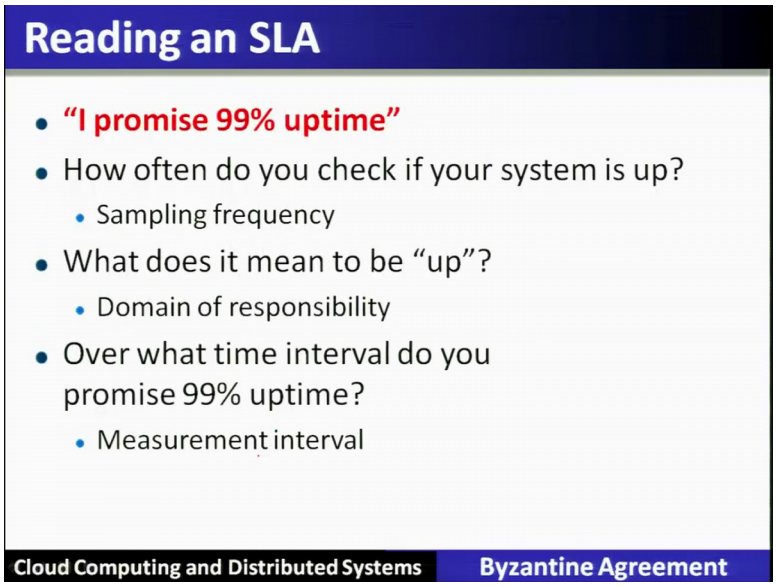
how good should it be, and service level agreement that is service level objective, plus what are the consequences. So, this is all specified in a 3 letter word that is called TLA.

(Refer Slide Time: 06:10)



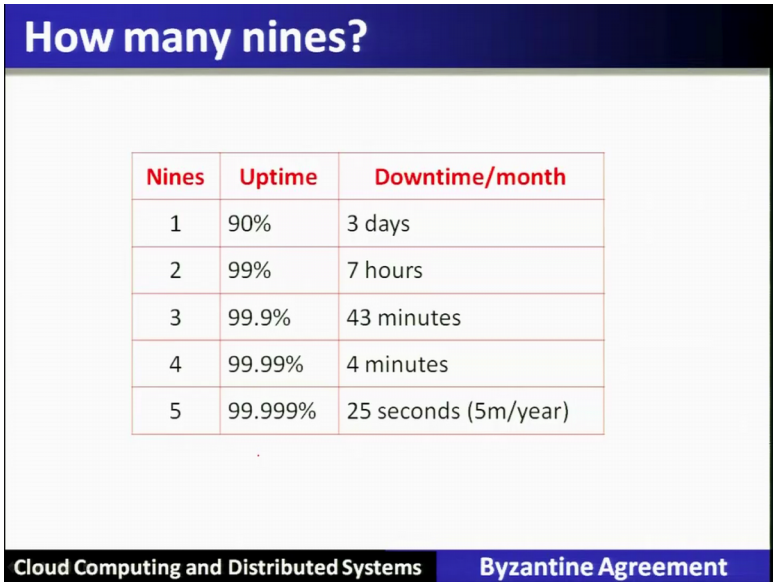
Now, why you want to measure it? Because, you are purchasing it as a service these guarantees and the service provider or the vendor also has to put various amount of resources. That means, he has to be at the cost to support this kind of guarantees in the pope in the terms of reliability. So, reliability promises are the part of the business. So, let us discuss it.

(Refer Slide Time: 06:38)



So, if the service level agreements says that I promise 90, 99 percent up time. What does this means? That means, to analyze this meaning is that you have to see that how often do you check if the system is up that is sampling frequency. What does it means to be up? That is domain of the responsibility over what time interval do you promise 99 percent uptime that is measurement interval.

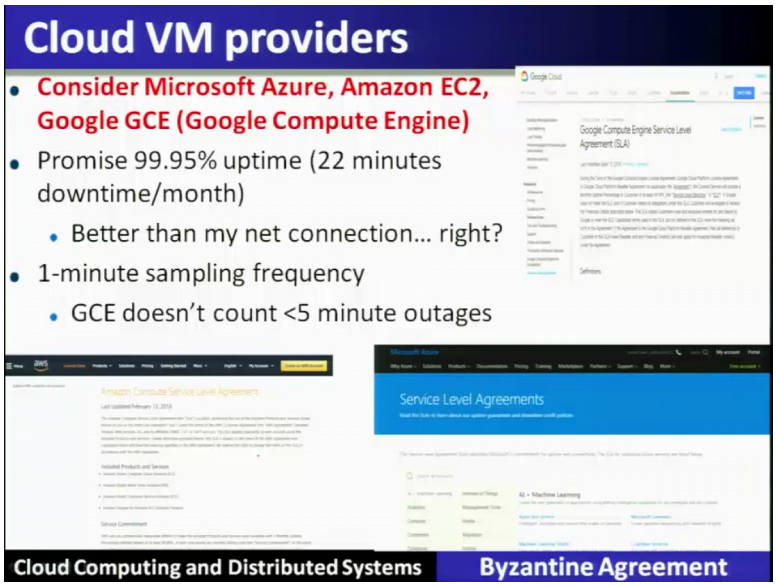
(Refer Slide Time: 07:14)



In all this can be measured or this can be specified how many 9's are there; that means, if there is only one 9. That is, if the uptime is 90 percent; that means, in a month that is in a monthly billing cycle 90 percent uptime comes out to be 3 days; that means, in a one month 3 days can be the down time. When it says that 2 9s that is 99 percent in a month that computes it to be 7 hours in a month. The system can be down before it can be repaired and kept up. 99.9, that is 3 9's works out to be 43 minutes in a month; that means, within 43 minutes the system is to be repaired and made available to the services.

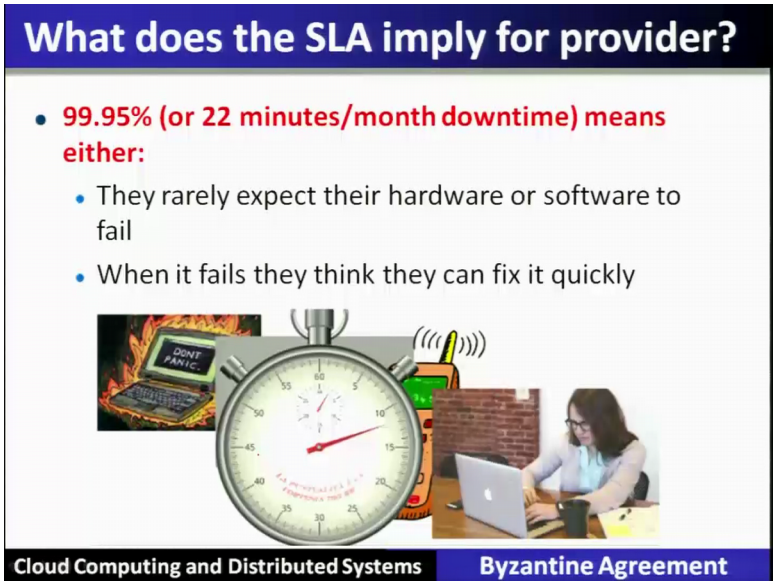
When it is 4 9's that is 99.99 that is works out to be 4 minutes; it is very quickly it has to be ensure it is maintenance. Now, when it comes out to be 5 9's that is 99.999 that is not in the month, that is in the year or in a month it comes out to be 25 seconds. That is 5 minutes per year that is a goal time. So, that kind of reliability if it is purchased by the service provider when service provider has to ensure; that means, it will provide this kind of uptime.

(Refer Slide Time: 08:55)



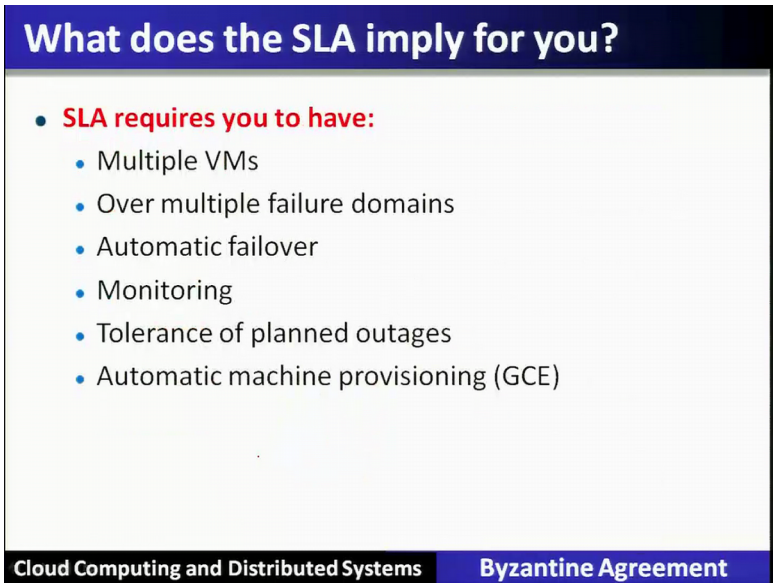
Cloud VM providers, blue chip cloud companies like, Microsoft Azure, Amazon, EC2, and Google compute engine, they all promise 99.9 5 that is 3 9's of up time that worked out to be 22 minutes' downtime per month. So, let us analyze it in terms of sampling frequency. They say that 1-minute sampling frequency; that means, in a 1 minute you have 60 seconds; that means, 59 seconds your system can be down. It can be up for one second and so on. So, there may be different tricks and place which need to be understood in this particular reliability game.

(Refer Slide Time: 09:53)



But at the end of the day to support 99.9 5 percent of time or; that means, within 22 minutes of downtime per month, the system is to be repaired and made available to the services. That is there rarely expect their hardware and software to fail, because if they fail then manually it is; that means, every person is sitting next to the computer, and these particular attending such system around the clock requires huge investment in terms of manpower and so on. That has to be fixed up quickly.

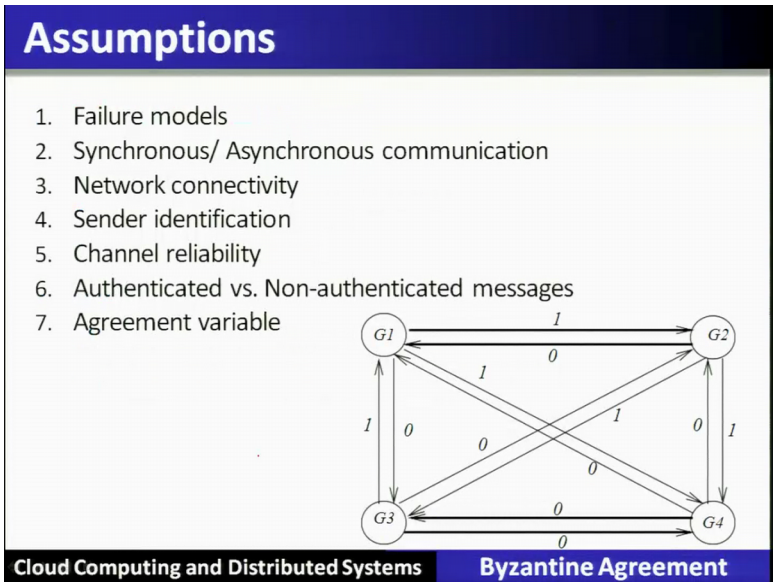
(Refer Slide Time: 10:43)



Now, as far as ensuring through service level agreement, the reliability means that the companies are required to support the services through multiple VMs; that means, whenever there is a failure in one site it can respond another VM at different site that the downtime can be reduced and the services continue to be service from the other new machine.

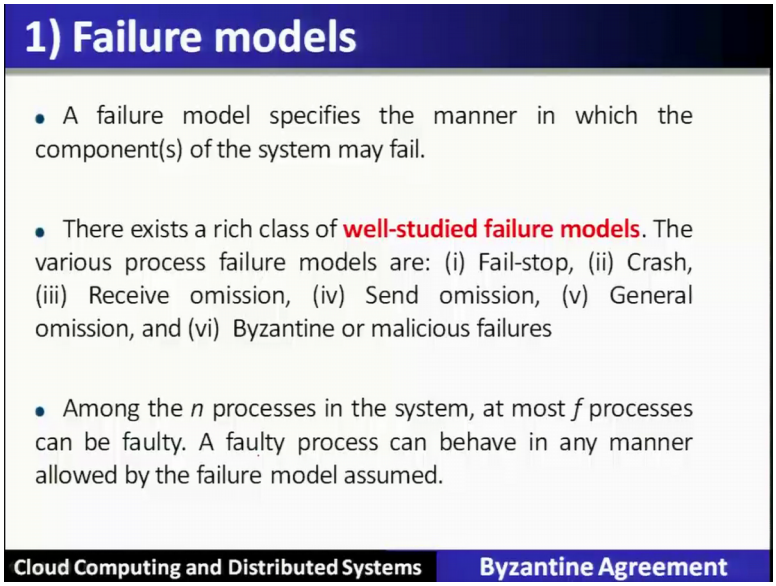
Then SLA also requires that over multiple failure domains, it ensures the reliability. And also ensures that automatic failover can be provisioned, continuous monitoring also is required and tolerance of the planned outages and automatic machine provisioning. So, all these things are required to be automated, then only that kind of service level agreement can be ensured.

(Refer Slide Time: 11:50)



So, let us assume for our byzantine fault or agreement algorithms, we will see some of the models. We will assume a fault model or a failure model we will also assume the synchronous or asynchronous communication. Network connectivity we will also assume how the sender is to be identified, then channel reliability authentication versus unauthentication message and agreement variable.

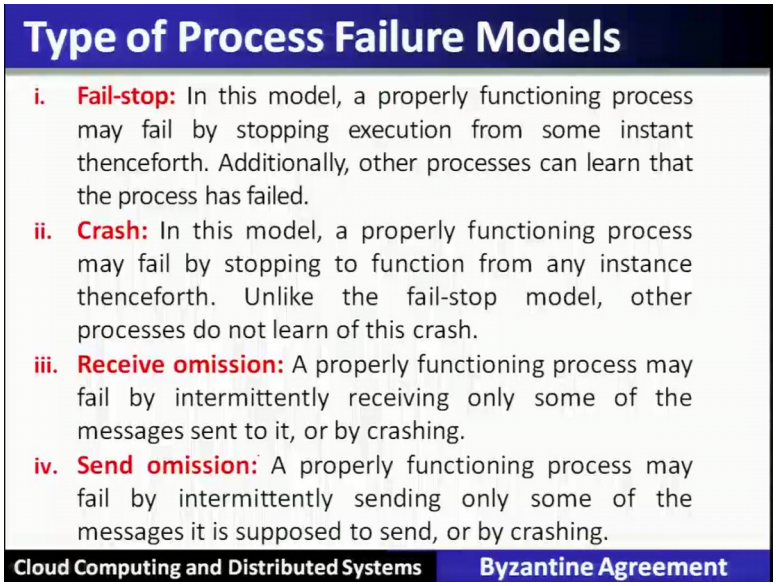
(Refer Slide Time: 12:22)



So, let us start one by one. So, failure model failure model specifies the manner in which the components of the system may fail. And the literature provides a well-defined failure

models which are fail-stop, crash, fault receive omission, send omission, general omission and byzantine or a malicious faults. So, in most of the study we assume that among n processes in the system at most f can be faulty. So, n f there are 2 parameters for such study in the system. So, a faulty process can behave in any manner allowed by the different fault models and such assumptions can be made.

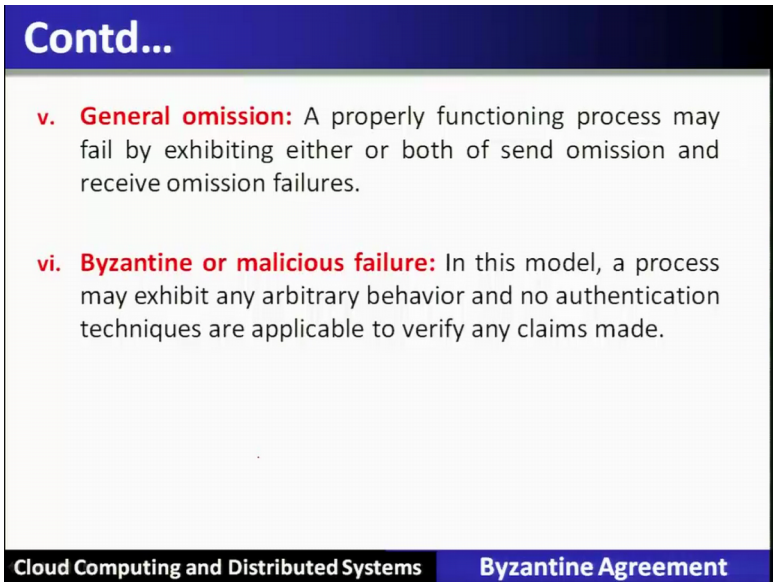
(Refer Slide Time: 13:15)



For example, the failure model or a fault model are of different types. The first one is called fail-stop in this model properly functioning process may fail by stopping the execution. From some instant henceforth additionally other processes can learn that the process has failed. Crash in this model a properly functioning process my fail by stopping to function from any instance henceforth.

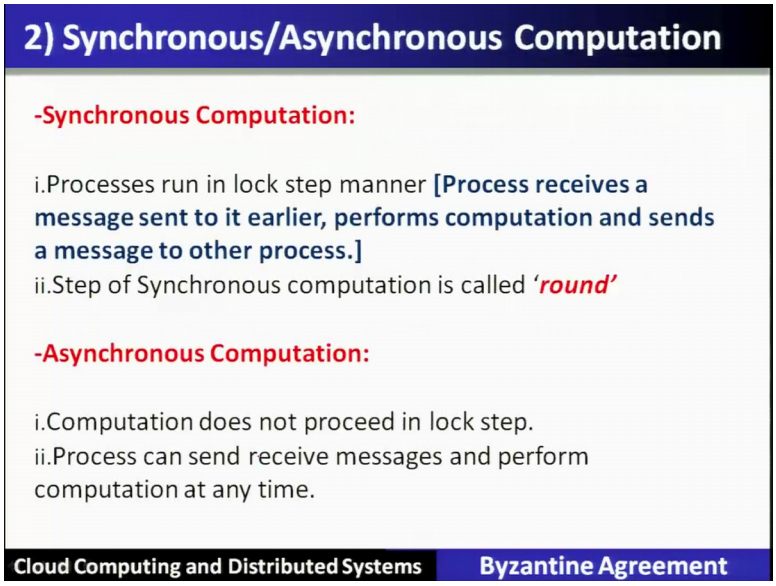
Unlike, fail-stop other processes do not learn of this particular crash. Receive omission a properly functioning process may fail by intermittently receiving only some of the messages sent to it or by crashing. Send omission or properly functioning process my fail by intermittently sending only some of the messages it is supposed to send by crashing.

(Refer Slide Time: 14:13)



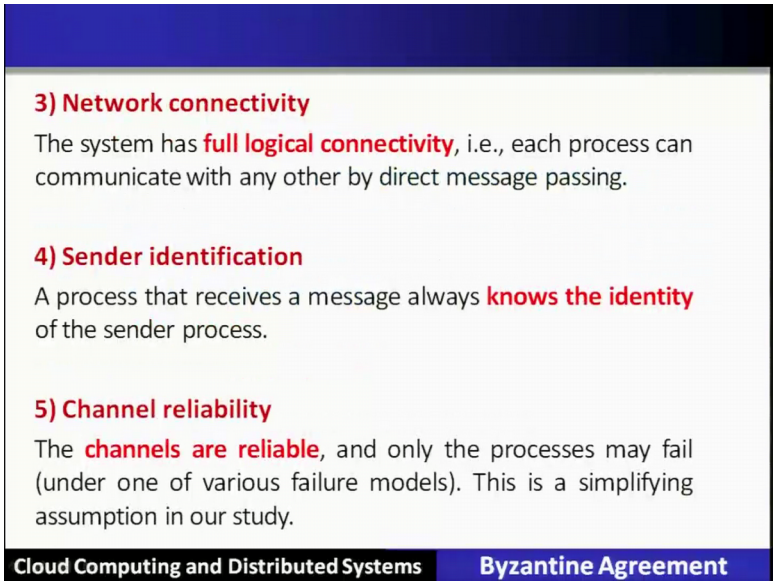
General omission or properly functioning process may fail by exhibiting either or both of send omission and receive omission failures. Byzantine or malicious failure in this model a process may exhibit any arbitrary behavior, and no authentication techniques are applicable to verify any claims made. This is called byzantine or a malicious fault this is most general kind of fault which may exhibit any arbitrary behavior due to the malfunction or due to the malicious activities. Hence, it is called byzantine or a malicious failure.

(Refer Slide Time: 14:57)



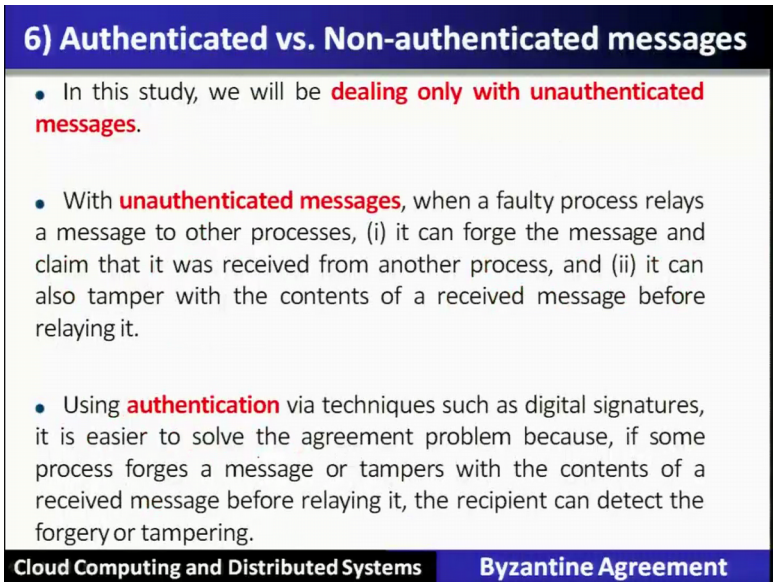
Computations are concerned there are two different models of computations which can be assumed in different agreement protocols. Synchronous computation, in this model the processes runs in a lock step manner; that is, the process receives a message performs the computation and send the message to the other process. So, the step of the synchronous computation is called a round. Whereas, asynchronous computation the computation does not proceed in a locked step the process can send receive message and perform the computation at any point of time.

(Refer Slide Time: 15:39)



Third one is called network connectivity. The system has full logical connectivity that is a assumed in that model. So, each process can communicate with any other by direct message passing. Sender identification a process that receives the message always knows the identity of the sender process. Channel reliability the channels are reliable. And only the processes may fail. This is to simplify and perform concentrated study of the subsystems.

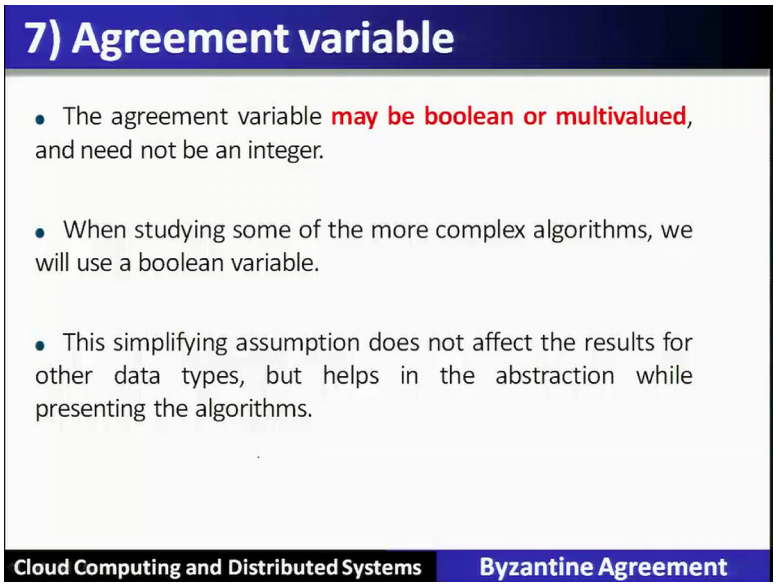
(Refer Slide Time: 16:23)



Authenticated versus non authenticated messages in this particular study we will be dealing only with unauthenticated messages. With unauthenticated messages when faulty process release a message to other process, it can forge the message and claim that it was received from another process.

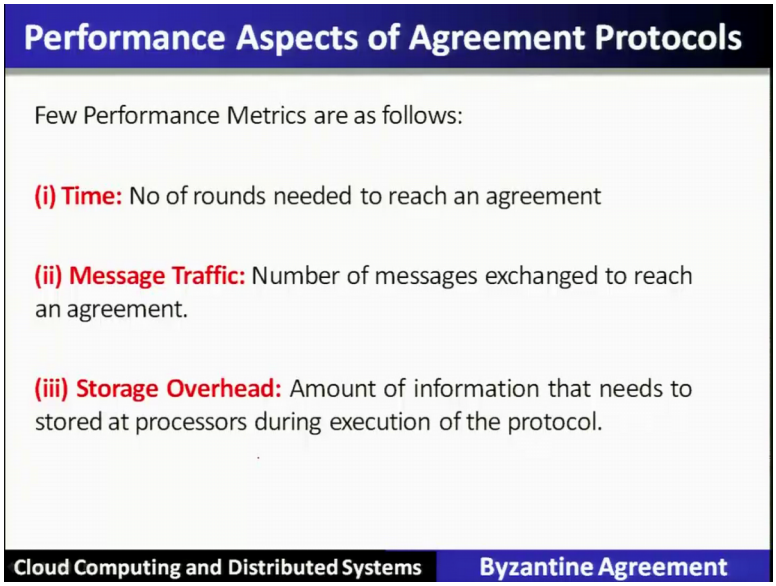
It can also tamper with the contents of the received message before relaying it. Now, using authentication techniques such as digital signature it is easier to solve the agreement problem because if some process forges a message or tampers with the content of the received message before relaying it the receiver can detected.

(Refer Slide Time: 17:05)



The agreement variable is another parameter. So, agreement variable maybe a Boolean or multivalued and need not be an integer. So, when studying some more complex algorithms we will only consider the Boolean variable for simplicity. This simplifying assumption does not affect the result of other data types, but helps in the abstraction while presenting the algorithms.

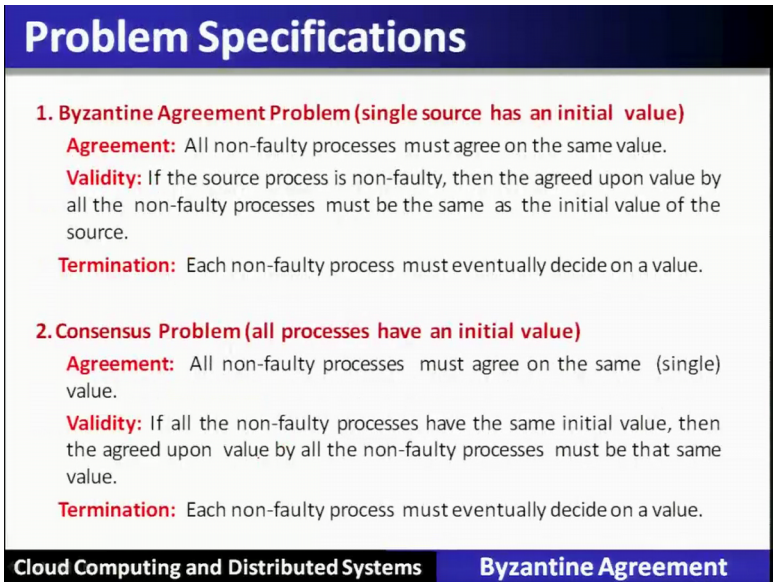
(Refer Slide Time: 17:27)



Performance aspects of the agreement algorithms; so, we will see some of the performance metrics; such as time which is measured in terms of rounds needed to reach

an agreement or terminate the agreement protocol. Second performance metrics is message traffic or that is the number of message exchanged in the algorithm to reach an agreement. Third one is the storage overhead that is the amount of storage required by the processor during the execution of the algorithm.

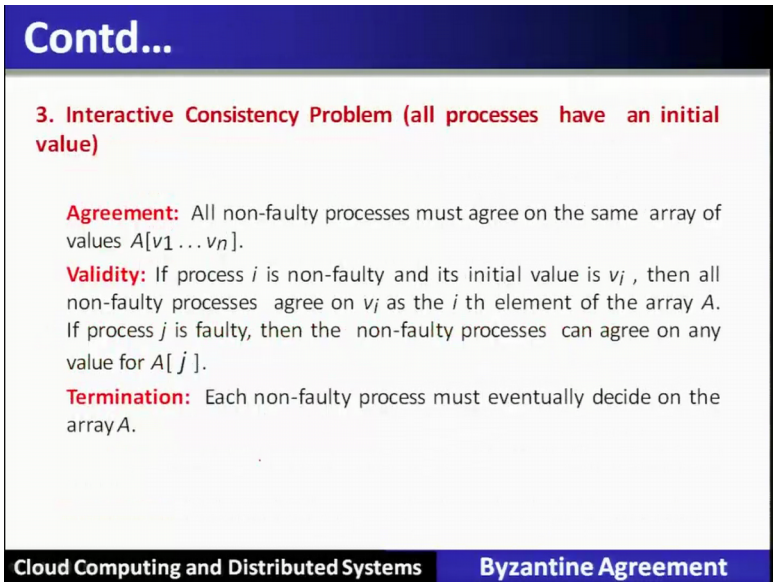
(Refer Slide Time: 18:15)



Let us see different variants of this agreement problem. The first problem is called Byzantine Agreement problem, which is having a single source with the initial value. So, it has 3 different conditions within it. Agreement says that all non-faulty processes must agree on the same value, that is the value of a source. Validity says that if a source process is non faulty, then the agreed upon value by all non-faulty processes must be the same as the initial value of the source. Termination says that each non faulty must eventually decide on the value.

Second barrier is called consensus problem. That is here all processes have an initial value, then it is called a consensus problem. Agreement says that all non-faulty processes must agree on the same single value; that means, they agree on only one value like byzantine. Validity that is if all the non-faulty processes have the same initial value, then the agreed upon value by all non-faulty processes, must be this must be that the same value termination is non faulty must eventually decide on a value.

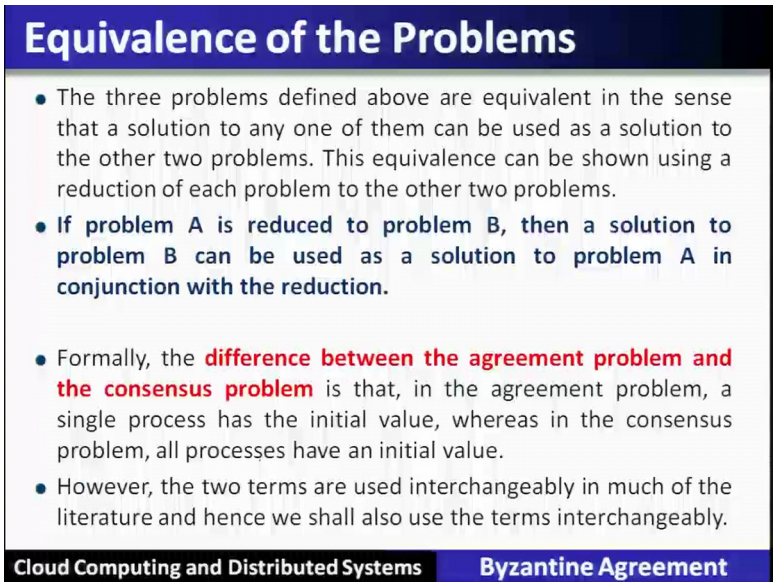
(Refer Slide Time: 19:47)



Third variant is called interactive consistency problem. Here all processes have an initial value. Agreement on non-faulty processes must agree on the same array of values. That means, if there are n different processes for there will be an array A, and there will be a particular value for each process.

So, this will be an array or a vector a of n different elements. And the vector or the array will be same in the agreement interactive consistency problem. Validity, that is if a process i is non faulty. And it is initial value is v i in that vector A. Then all non-faulty processes agree on v i as ith element of the array. If the process j is faulty, then the non-faulty processes can agree on any value for a of j. Termination that is each non faulty processes must eventually decide on an array a.

(Refer Slide Time: 21:01)



Now, let us see there are three different variants there equivalence. So, the three problems defined above are equal in the sense that a solution to any one of them can be used as solution to the other two problem. This is this equivalnce can be shown using a reduction. So, if a problem A is reduced to a problem B then the solution to a problem B can be used as a solution to a problem A in conjunction with the reduction.

So, let us see the difference between agreement and the consensus problem, is that in the agreement problem A single process has the initial value; where is an consensus problem all the processes have their own initial values; however, the two terms are used interchangeably in much literature and hence we shall use these terms interchangeably.

(Refer Slide Time: 21:58)

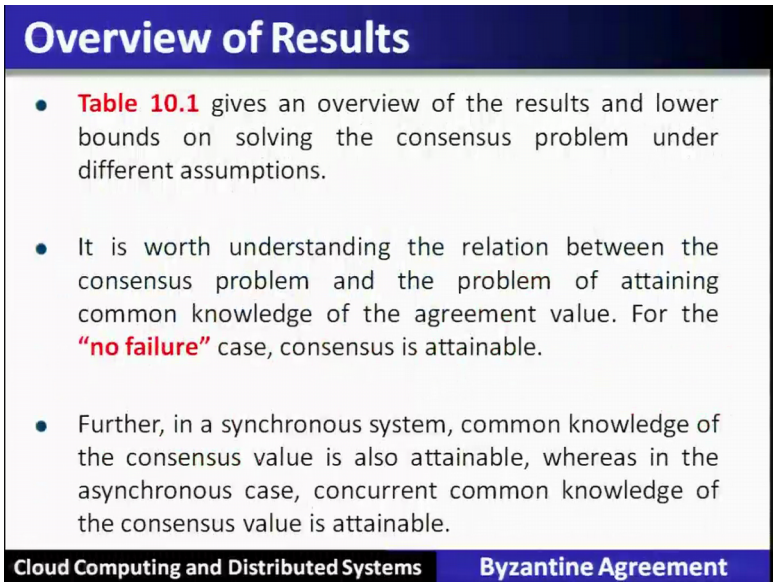
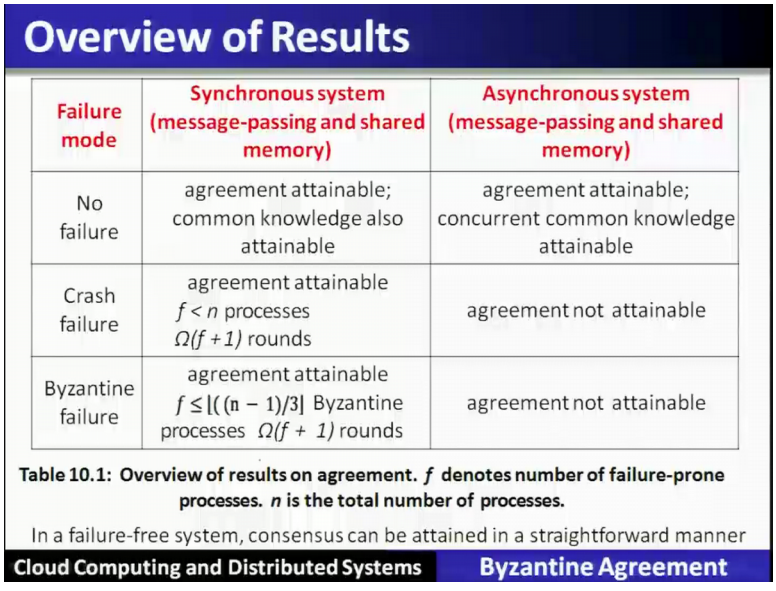


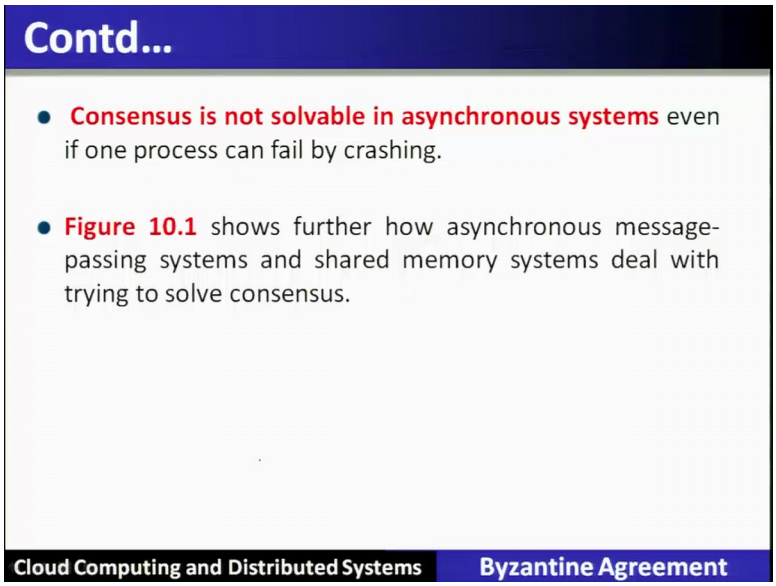
Table shows an overview of the result at the lower bound in solving the consensus problem under different assumptions. For that the synchronous system, the common knowledge of the consensus value is also attainable; where asynchronous system concurrent common knowledge of the consensus value is attainable.

(Refer Slide Time: 22:22)

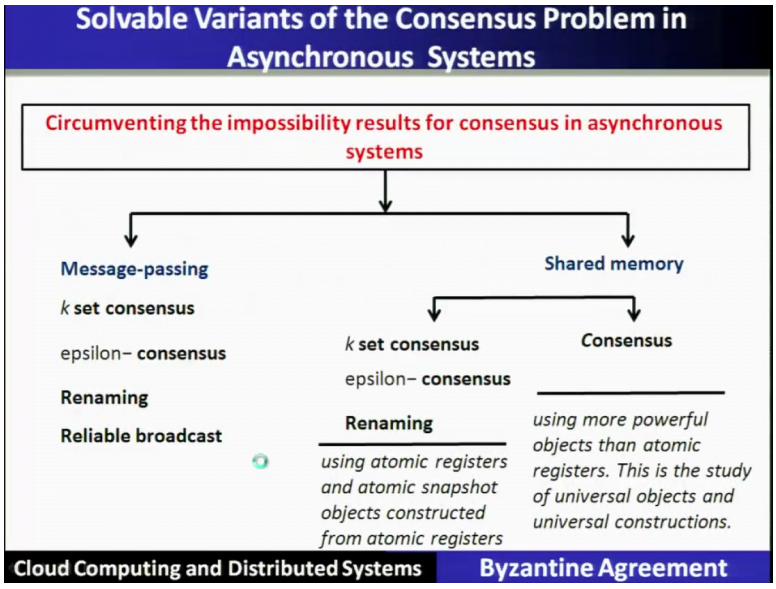


So, this particular table summarizes, it we will discuss each and every part in more details.

(Refer Slide Time: 22:30)

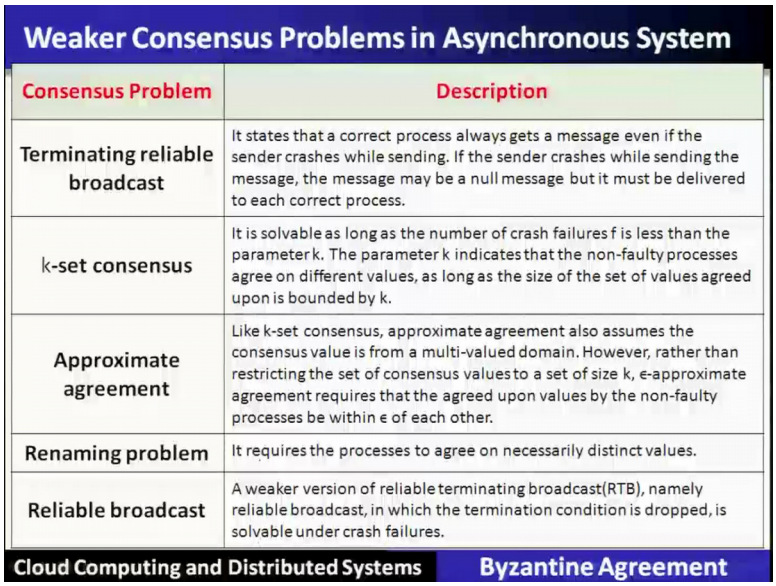


(Refer Slide Time: 27:31)



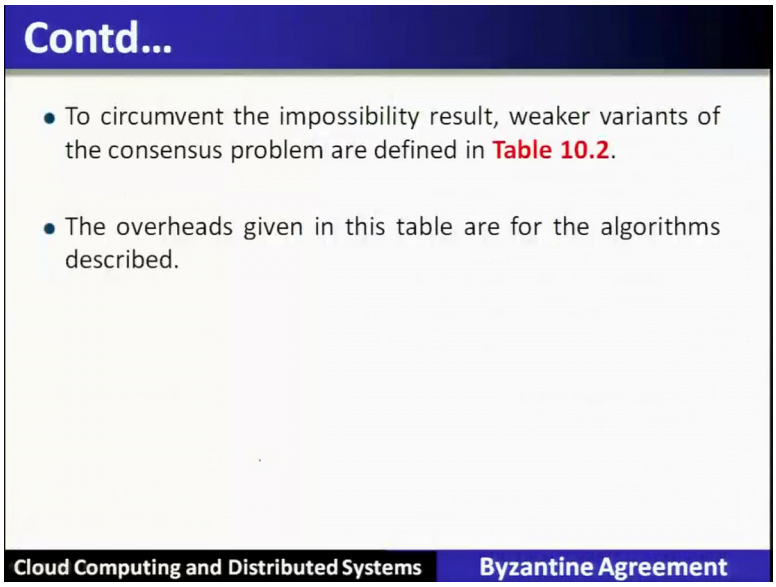
Now, there are some solvable variants of the consensus problem. It is shown that the consensus problem in asynchronous system is impossible to be solved. Therefore, we will consider the synchronous system, hence circumventing the impossibility result for consensus asynchronous system, can be classified into different ways message passing and shared memory. Message passing is our point of discussion.

(Refer Slide Time: 23:07)

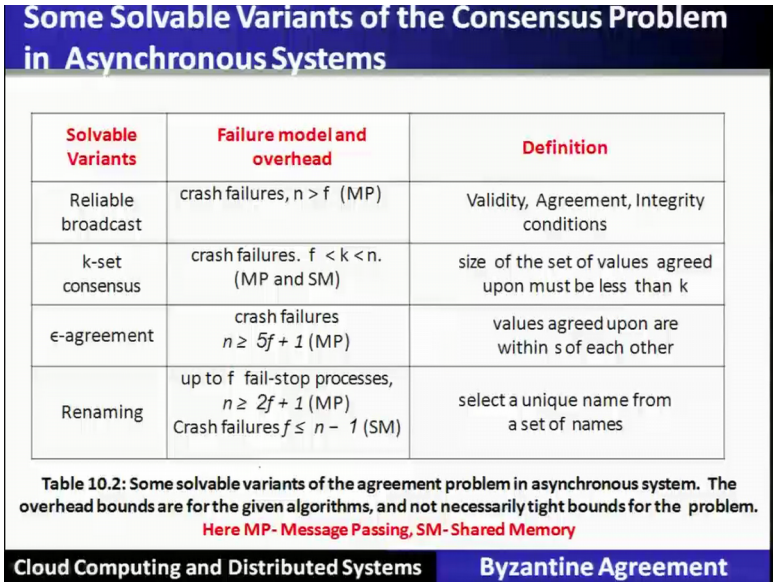


There are some weaker consensus problems in asynchronous system, that we will discuss in the end.

(Refer Slide Time: 23:13)

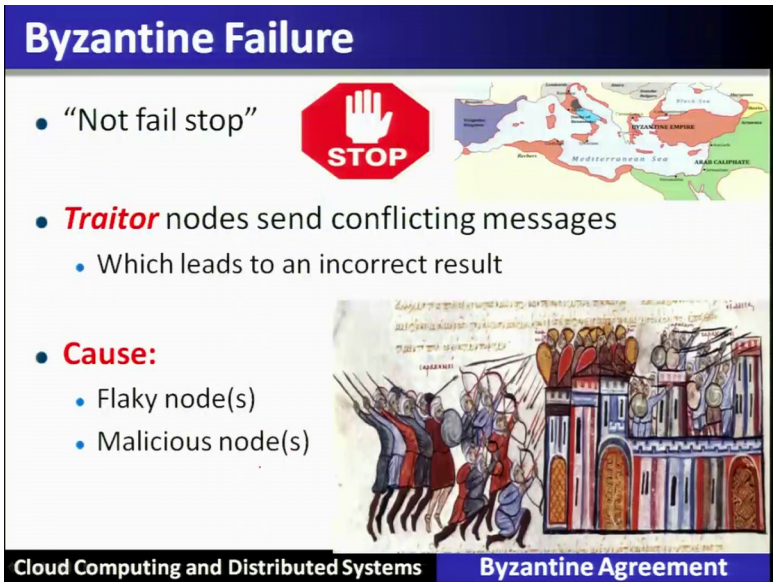


(Refer Slide Time: 23:14)



So, this is also summarized some solvable problem of consensus in asynchronous system. Now, agreement in a synchronous message passing system.

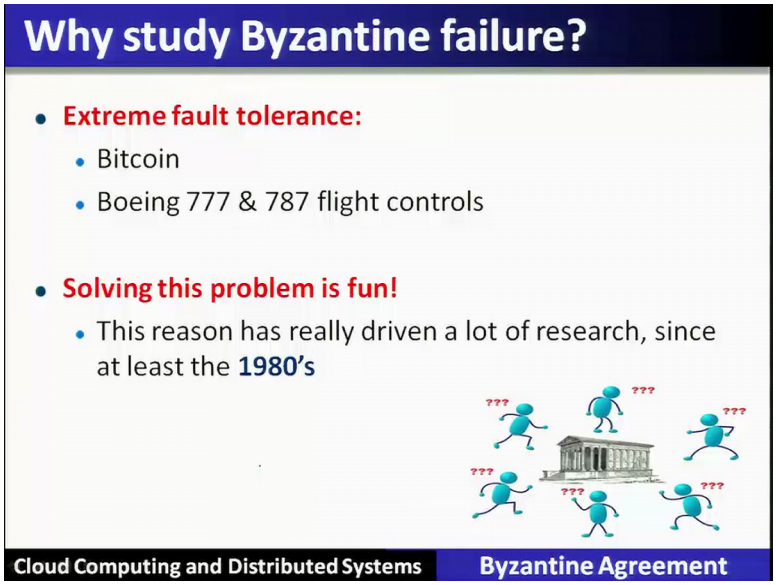
(Refer Slide Time: 23:24)



Let us see: what is the byzantine failure. It is not a fail-stop failure. Why because it is having an arbitrary fault model or a malicious faults can also be categorized as byzantine failure. The name byzantine fault model is derived out of a particular story in which there is a different army located around a particular mountain.

They cannot see each other, but they can communicate by sending the messenger physically from 1 to 2 another group and their generals can now decide whether to attack on another army. So, it may be possible that these generals some are creators, they may send the confusing messages. Hence, maliciously they are sending messages and this model is modeled as the byzantine failure. And hence the name is basically given up from that location which is called byzantine. The causes of these failures are in our distributed system or in a cloud computing system is due to the flaky nodes or the malicious nodes.

(Refer Slide Time: 25:08)

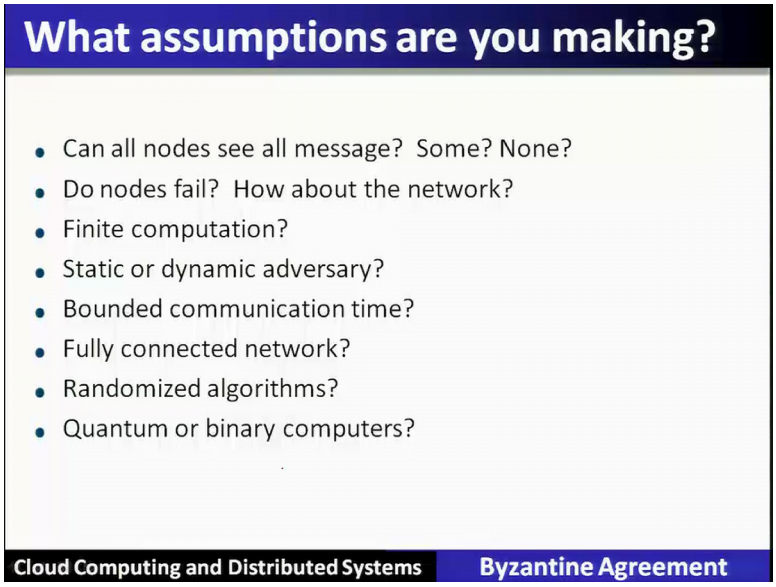


Now, the question is why we are considering this kind of fault model. Byzantine failures are most general kind of failure. So, the systems has to be build let us see the application, that is an extreme fault tolerance which is required in the case of bitcoin where the faults or this kind of failure or the behavior which is called byzantine failure is suspected from every agent in the bitcoin. Therefore, a byzantine fault model can be assumed while designing such system called bitcoin.

Another extreme fault tolerance which is required in the aircraft design that is in the Boeing planes at their flight control. So, that for any kind of failure the flight control systems are quite fault tolerant. So, this byzantine fault will give the extreme fault tolerance and these two examples which we have shown where such kind of fault model can be used to design the systems. Another reason of studying this byzantine fault is to

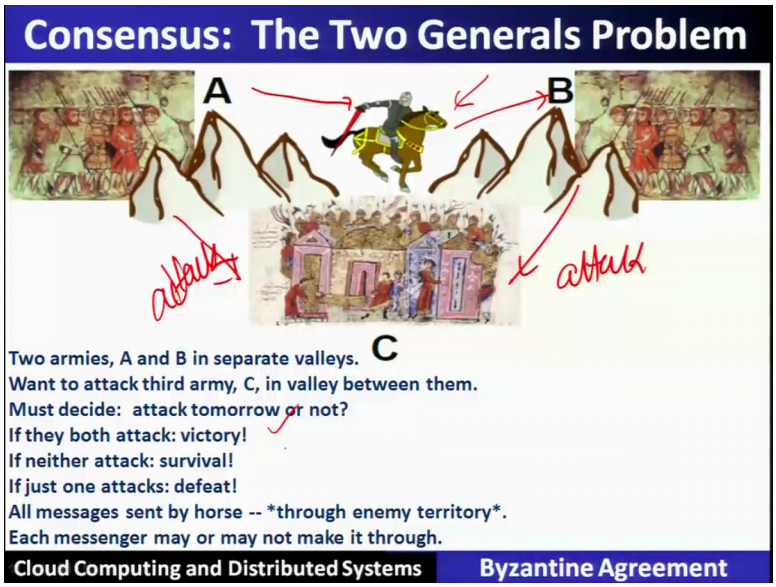
understand the research involved to deal with this kind of fault model and yet how they will be good get be designed.

(Refer Slide Time: 26:42)



There are different assumptions while discussing this kind of fault model, which are necessary to be understood let us understand this problem in more details.

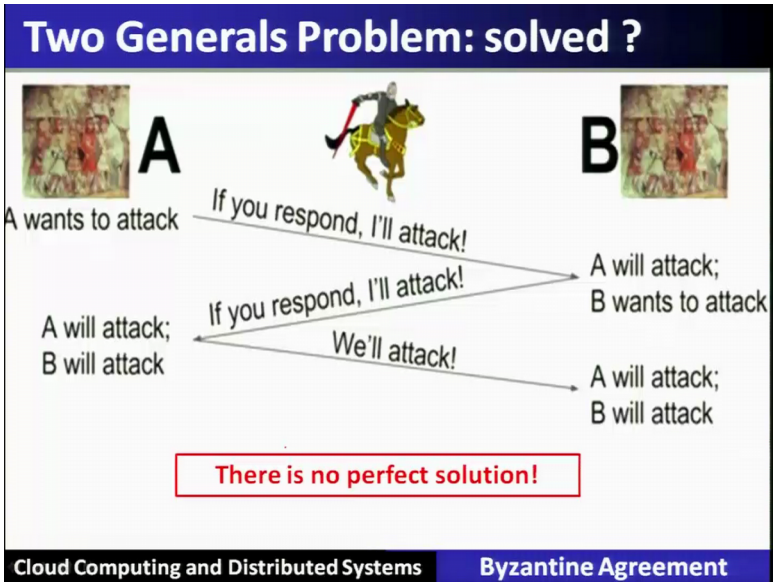
(Refer Slide Time: 27:02)



Now, agreement between two generals; let us say A and B which cannot directly communicate they can communicate through a messenger, which is shown here, who can go from who can take the message of A, and give it to the message B. Let us say that A is

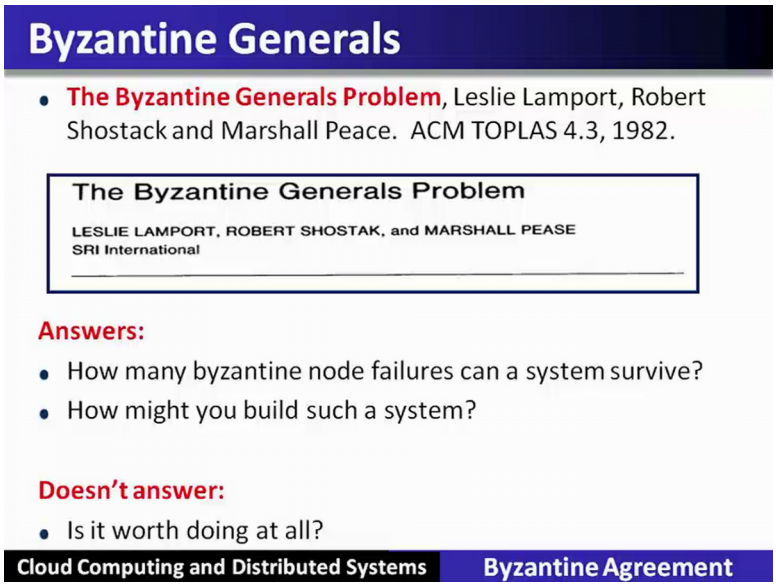
communicating to B to decide if both of them can attack together, because alone they cannot defeat the enemy that is C. So, they have to exchange the messages in this regard. So, A is sending a message that if they attack if they both attack then it will be a victory. And the other one is replying back with some message and again wait for the response of A.

(Refer Slide Time: 28:19)



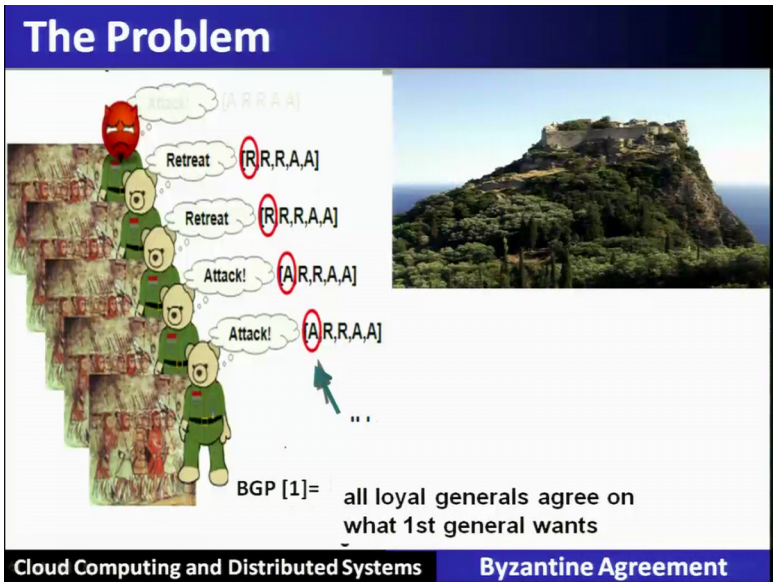
So, that means, a will send if you respond I will attack. This message will go from A to B and wait for the response of B. Now B will also response that if you respond I will attack and then again wait for as response. This message keeps on moving between A and B, and finally, they have to wait for the other sites perfect answer and never reaches to a consensus hence there is no perfect solution that is there is no consensus in this manner.

(Refer Slide Time: 29:00)



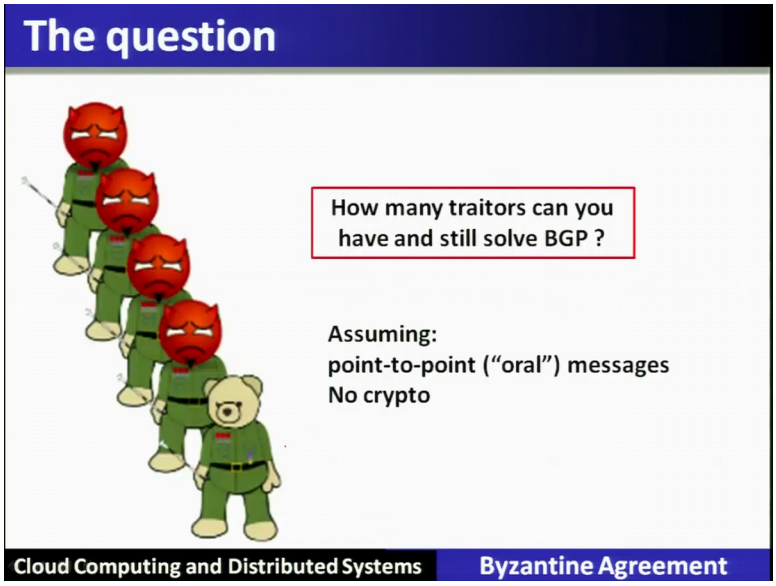
The byzantine general problem is studied by Leslie Lamport, Robert Shostack and Marshall Peace in their paper, which is called the byzantine general problem which is published in 1982. This particular paper has answered, the questions like how many byzantine nodes failure can the system be survive and how many how might you build such a system. But it does not answer is it worth doing worth solving this particular problem. Let us see in more detail about the solution.

(Refer Slide Time: 29:41)



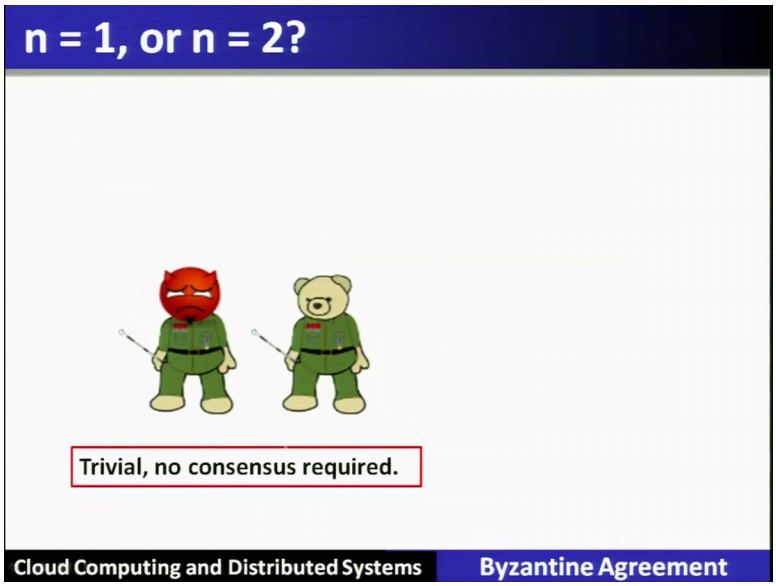
So, the problem is that different army commanders they are placed with their army around this particular mountain, which is called a byzantine. And they have to decide whether to attack or to retreat, and they have to evolve in a consensus what to do. Among them, some of them are traitors; that means, they may confused by sending a wrong message. But if there is a majority that we will see that if majority is correct or perfect; that means, they are loyal then they may reach to an agreement what is that particular how many traitors can be there yet you can get the proper decision.

(Refer Slide Time: 30:36)



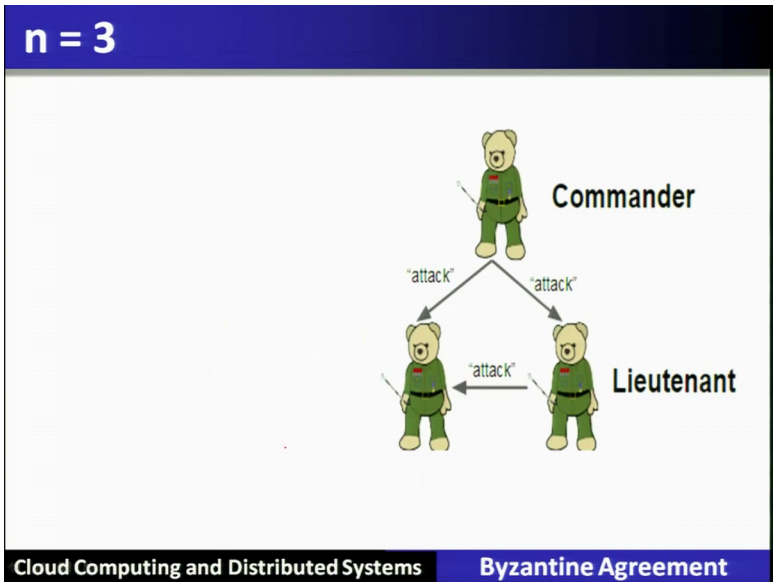
So, the question is how many traitors can you have still to solve the byzantine general problem. So, here we will assume that they communicate using oral messages and no crypto solutions is involved.

(Refer Slide Time: 30:56)



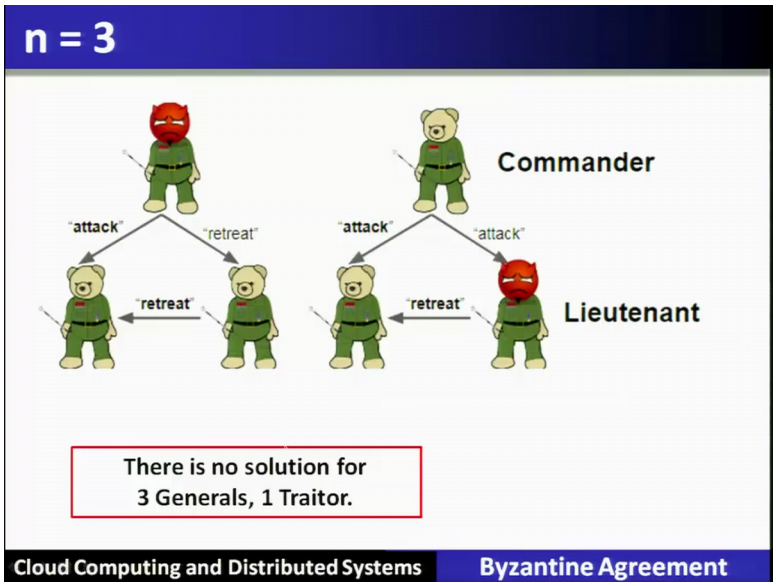
So, when n is equal to 1 or n is equal to 2 and means if 2 generals are one general is there, then it is trivial and no consensus is required here in this case.

(Refer Slide Time: 31:12)



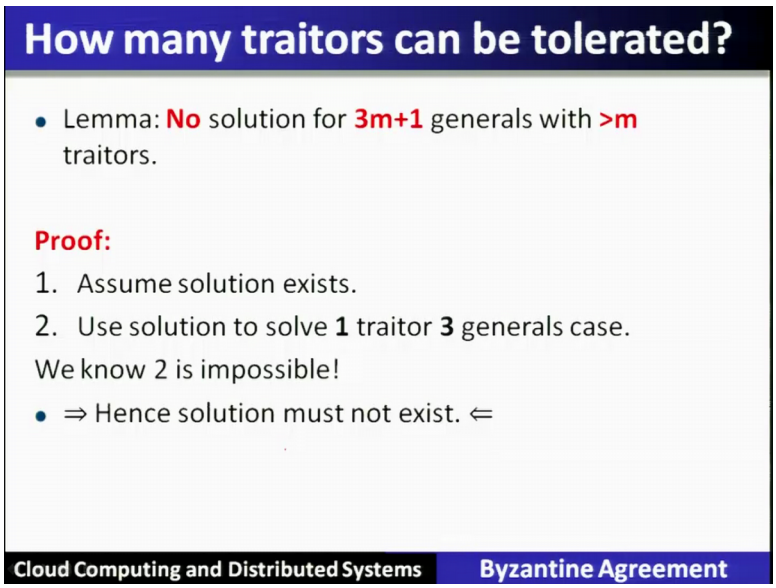
Let us see the condition when there are 3 different commanders. So, if in 3 different commanders if they say attack and this message will be exchanged.

(Refer Slide Time: 31:26)



But if one of them is traitor; the traitor will in first case we will give a confusing message to one of the commander he will say attack the other one he will send, retreat and on receiving this particular message they will exchange among themselves. And therefore, there is no majority, and this particular problem will not be able to be solved if there are 3 generals and among them one is traitor.

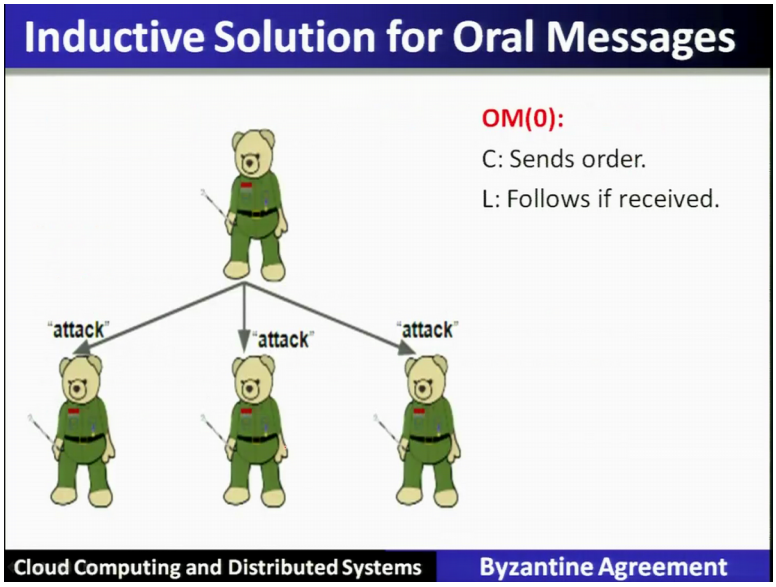
(Refer Slide Time: 32:02)



Therefore, how many such traitors can be there in a system. So, that it can be solved. So, there is a lemma that no solution for 3 m plus 1 generals with m is greater than traitors.

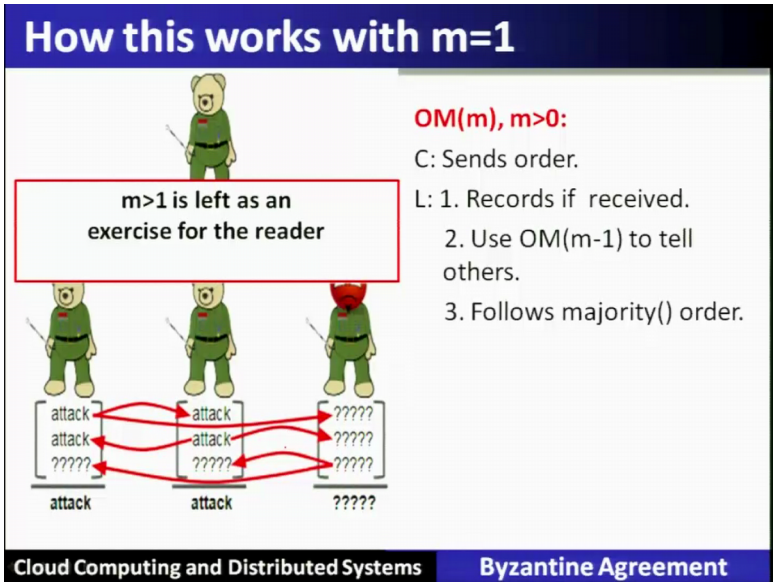
So, it says that if the number of generals are less than 3 m plus 1. Where m is the number of traitors there is no solution. So, we will see all these issues.

(Refer Slide Time: 32:37)



So, let us see that working of this algorithm.

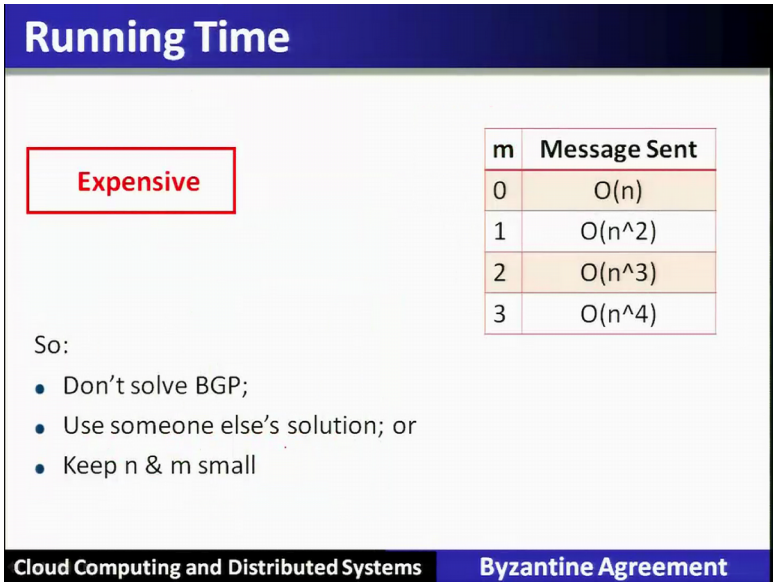
(Refer Slide Time: 32:43)



In this particular case, both of them will send the attack one commander will send the attack to all 3. And they will exchange their messages and finally, they will take the majority. One of them is traitor, then he will send a confusing message, and even if their

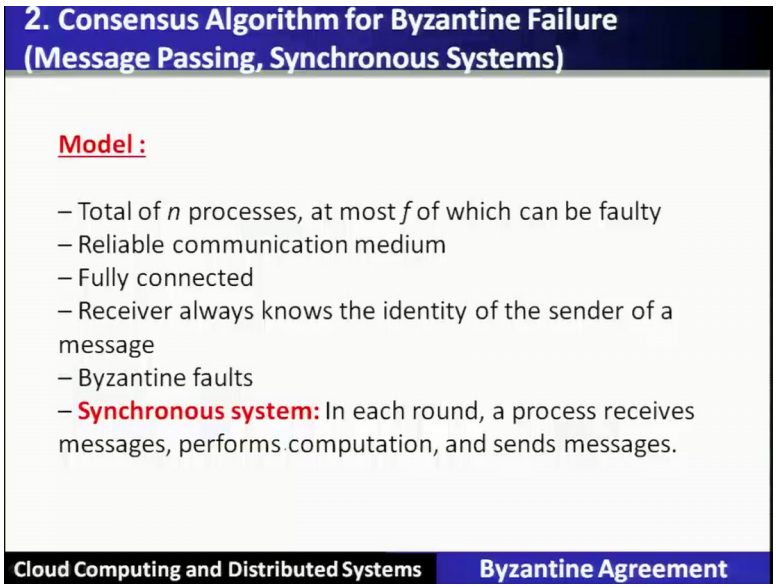
messages are exchanged among themselves at the lower end and if the majority if the if the majority is taken, they may reach to a consensus.

(Refer Slide Time: 33:17)



So, that we will see how this particular algorithm will work, but at the end of this particular study of this byzantine general problem, we will see that number of messages, which are exchanged is exponential and is called quite huge.

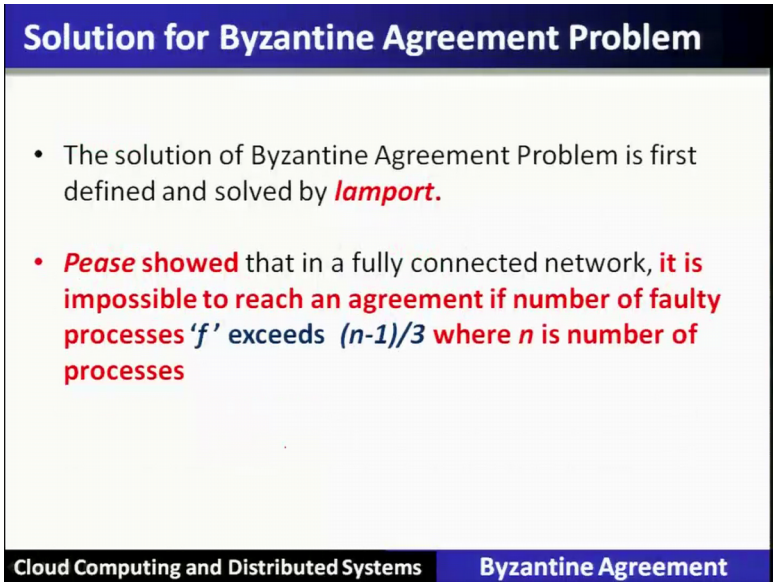
(Refer Slide Time: 33:32)



So, let us see the algorithm, which is called a Byzantine Agreement problem in the message passing system, and also the synchronous system. So, the model we will assume

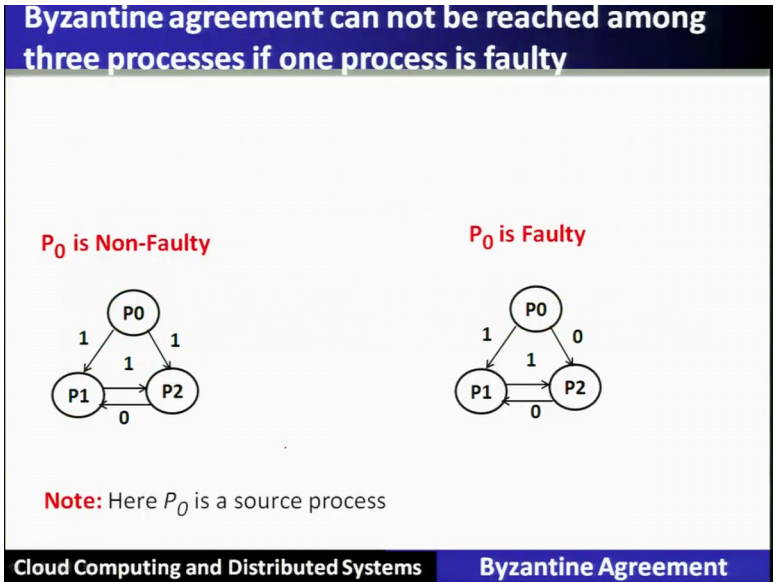
is let us assume that there are n processes out of that m f can be. Faulty reliable communication medium we will assume fully connected receiver always knows the identity of the server. And we will assume the fault model as byzantine fault and the system is the synchronous that is in each round process receives. The messages send and performs a computation and send back another round of messages.

(Refer Slide Time: 34:12)



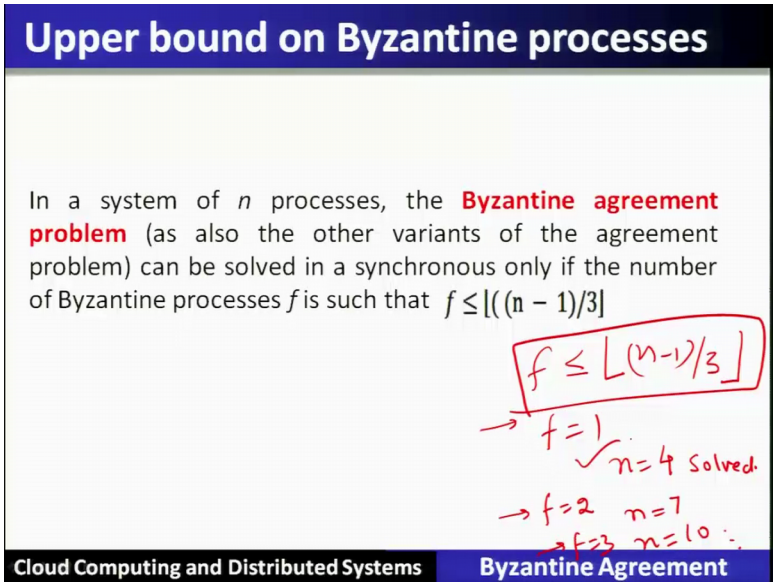
Solution to the Byzantine Agreement problem is first defined by Leslie Lamport. And peace showed that in a fully connected network it is impossible to reach an agreement if the number of faulty processes.

(Refer Slide Time: 34:32)



F exceeds n minus 1 by 3 where n is the number of processes. So, let us see example here, when this particular problem cannot be reached to an agreement. Here there are 3 different processes out of which one can be faulty, if one is faulty then agreement is not reachable, because here the f exceeds n minus 1 divided by 3.

(Refer Slide Time: 34:56)

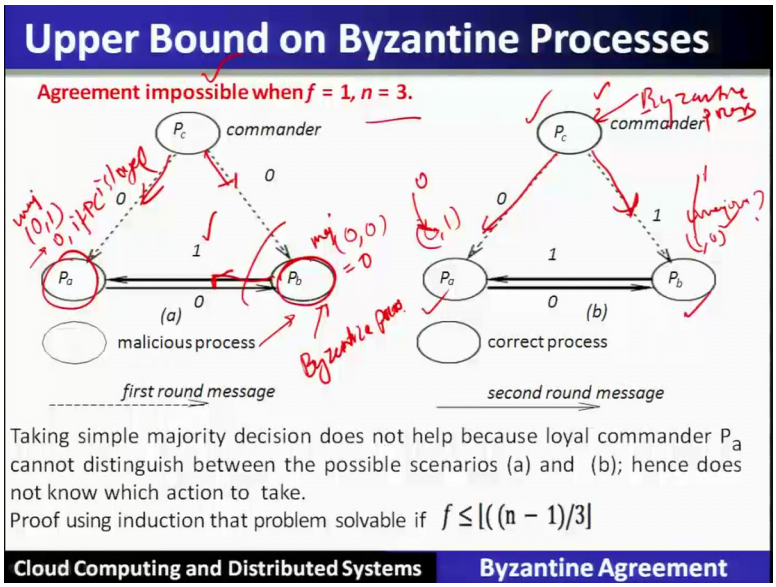


There is an upper bound which says that in a system of n processes. Byzantine Agreement problem can be solved in a synchronous model; if the number of byzantine processes f is bounded upper by n minus 1 divided by 3. So, if f is less than or equal to n

minus 1 divided by 3, then it can be solved. So, take the example, if f is equal to 1, then it comes out to be n is equal to 4 then it can be solved. Similarly, when f is equal to 2, n is equal to 7 and when f is equal to 3, then n is equal to 10 and so on.

So, this particular formula will give minimum number of on having knowing how many processes can fail, what will be the size of minimum number of nodes should be present to tolerate with the byzantine faults.

(Refer Slide Time: 36:29)



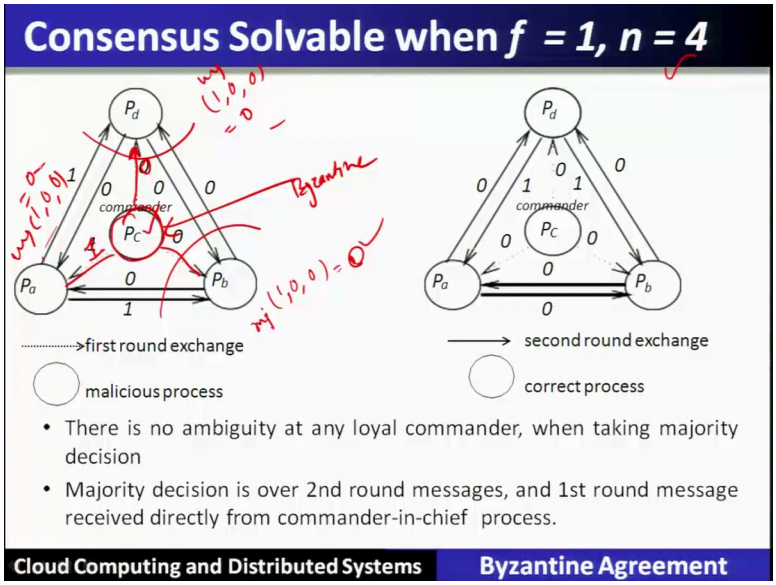
So, here the byzantine problem is impossible; when n is equal to when f is equal to 1 n is equal to 3, why? Because it requires minimum as per the bound n is equal to 4, but it is less than that. So, hence the Byzantine Agreement is impossible in this particular model. Let us see that if in the first case why it is impossible. We will assume that P b is malicious or a byzantine process.

So, the commander who is loyal, he will send both the values 0, both the ends. P b is the byzantine process, he will send the malicious way it will send after getting 0 he will send 1, because it is a malicious process. Whereas, P a is concerned that is also loyal. So, if it gets 0 and we will exchange it so, this P b will receive two different messages 0 and another 0. So, the majority of 2 0's will become 0. P a is concerned it will receive 0 and it will receive 1. So, majority cannot be calculated and if we assume that P c is loyal it is value is retained, then it will be 0 if P c is loyal.

But, if we see in another situation, when commander itself is malicious or a byzantine process; then it will send two different messages or a confusing messages to A and B. A on receiving the same message it will send, why because this is loyal. B on receiving the message it will send the same message let us see it will receive P b will receive 1 and 0. So, what is the majority? That is not known P a will receive 0 and 1, what is the majority that is not known. So, if we go by the first principle that the source is retained, then here it will be taken up as 0 and this will be taken up as one.

So, there is no consensus.

(Refer Slide Time: 39:28)

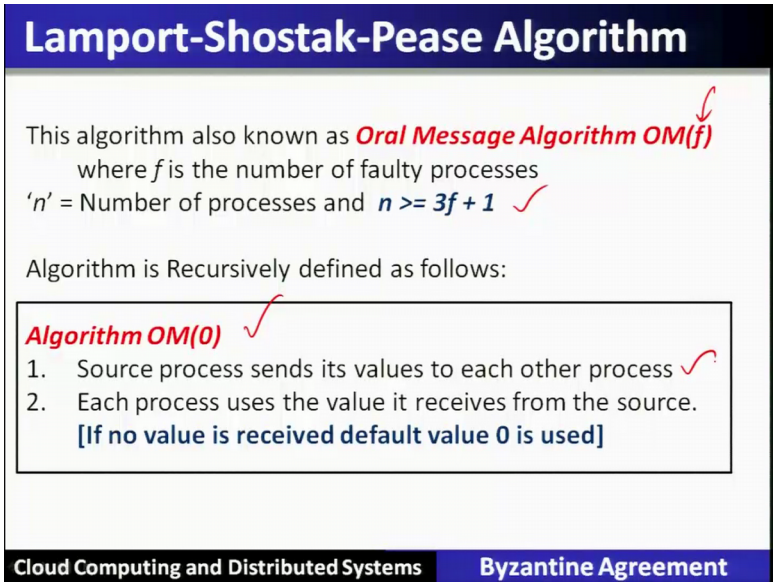


Now let us see that when n is equal to 4, f is equal to 1, then we will see here that consensus will be arrived. So, here let us see that the commander is the is byzantine. So, it will send different messages, it will send 0. But it will send 1 to P a and it will send 0. These messages will be further exchange. So, P a will send one to P d and P b. And one we take the total in values which is received by P b it is 1 coma 0 coma 0 and, if you take majority that comes out to be 0. Similarly, P d also will receive 1 0 0, and majority will be 0, and here also majority of 1 0 0 that becomes 0.

So, just see that here irrespective of the source being byzantine or a faulty the system is now reaching an agreement or a consensus of the same values at all other non-faulty processes. Sorry, hence through the example we have seen that if the condition that is the

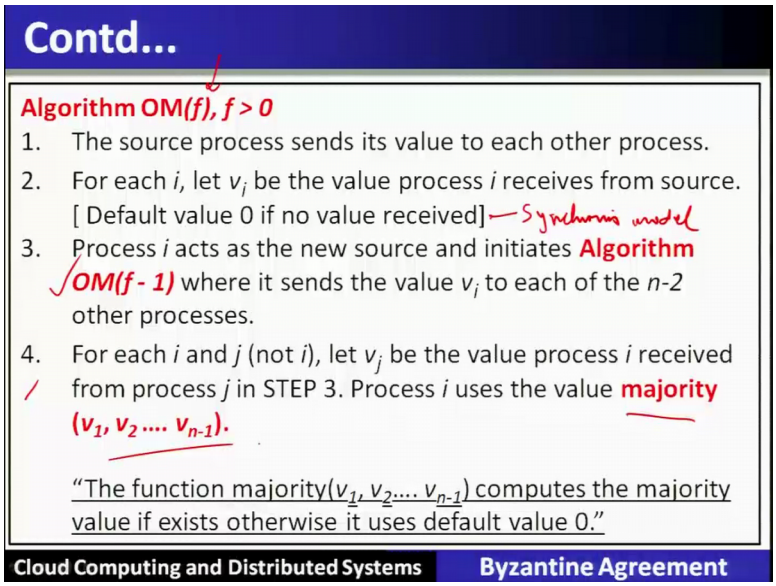
upper bound is satisfied, then this particular agreement problem can be solved let us see the algorithm for that which is called a Lamport Shostak Pease algorithm.

(Refer Slide Time: 41:03)



This algorithm also well known as oral message algorithm; where the input to this particular algorithm is f, f is the number of faulty processor. Here n is greater than or equal to 3 f plus 1. This particular boundary we have already seen. This particular algorithm is recursive algorithm. So, the base case of the recursion is when, when it is 0 then the source process will send it is value to each other process. And each process uses the value it receives from the source.

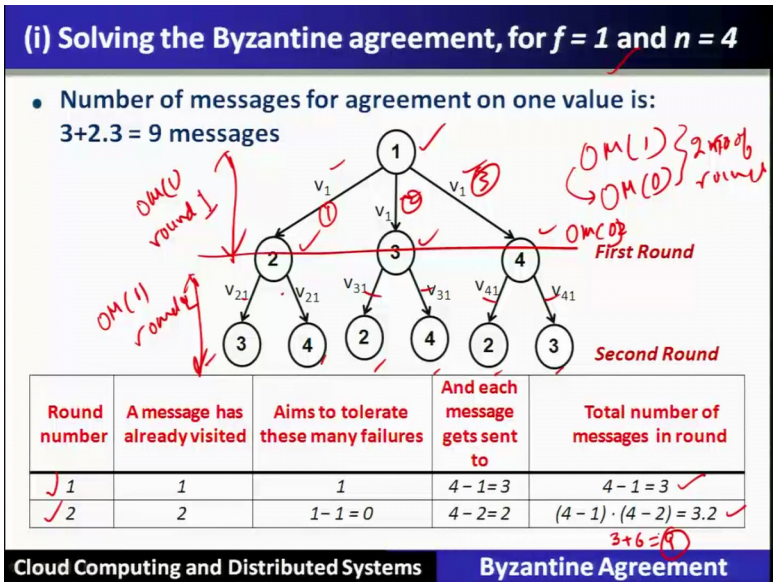
(Refer Slide Time: 41:51)



Now, it will be the next step will be the recursion, recurse, it will recurse on OM of f, when f is greater than 0. Source process sends it is values to each other, that we have seen for each i, let v i be the value the process i received from the source if there is no value is received then it will be 0, why because it is a synchronous algorithm.

We can assume such a value. So, process i will act as a new source and then initiates OM f minus 1. So, finally, what you will do? For each i and j, let be j be the values received by a process i in a step number 3; process i will compute the majority. So, the function majority computes the majority value if the if it exist. Otherwise it uses a default value of 0, here in this algorithm.

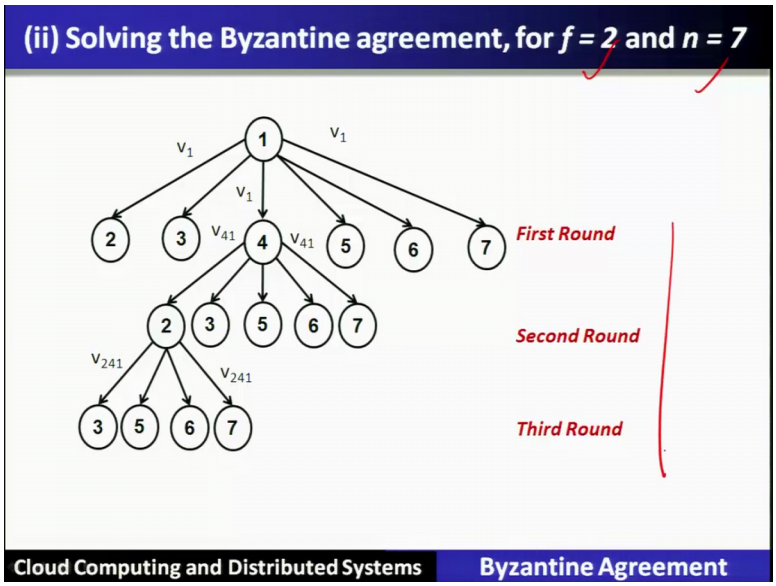
(Refer Slide Time: 43:02)



So, we have already seen the Byzantine Agreement, example also. Let us do the analysis of this particular algorithm. So, when f is equal to 1 this algorithm will be called OM of 1. Which in turn will call OM of 0 so; that means, there will be a 2 number of rounds. So, here 2 rounds are shown. In the first round, it will send the values, one of them will send the values the commander will send the values to the remaining 3 different process, let us say v 1, which in turn every process will become source and will invoke the round 2 with OM 0.

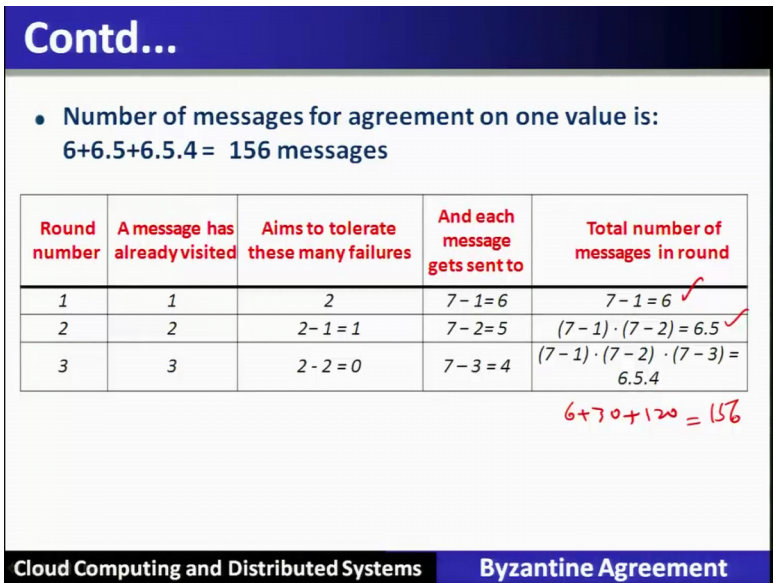
So, there will be 3 invocation of OM 0, 1, 2 and 3. And once the messages are reached at all the ends they will take the majority. So, let us count how many messages are exchanged. So, in the first round 3 messages are exchanged 1 2 and 3. In the second round so, in the second round there will be a 6 messages 1, 2, 3, 4, 5, 6. So, total number of messages will be 3 plus 6; that is 9 different messages will be exchanged here in this particular case.

(Refer Slide Time: 45:00)



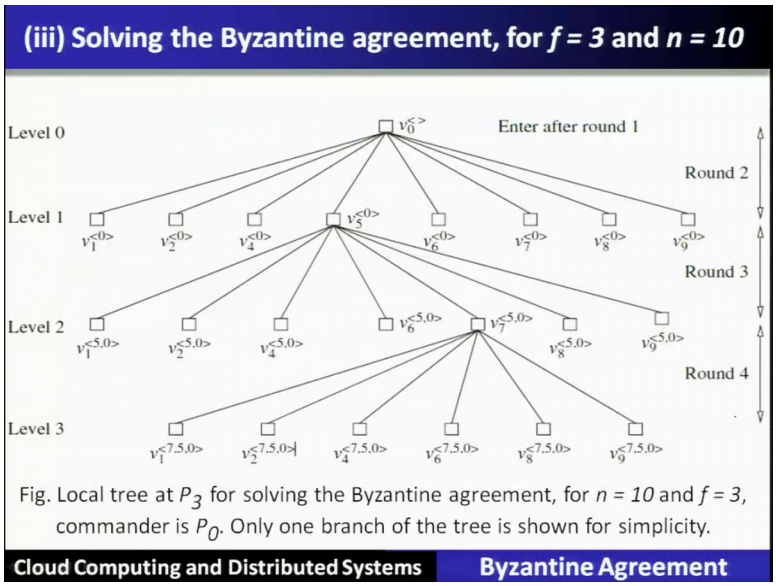
Now, when f is equal to 2, and n is equal to 7, there will be 3 different rounds.

(Refer Slide Time: 45:10)



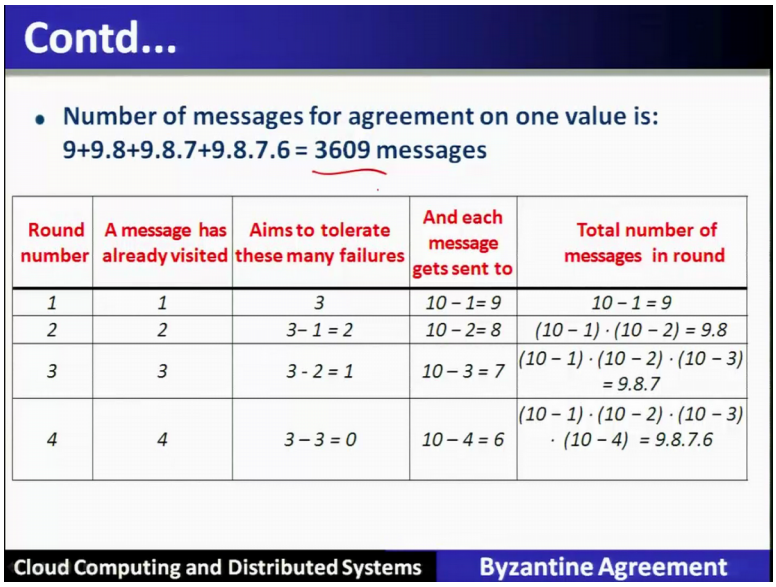
And if you see here how many messages will be exchanged? As you know that it will be n minus 1, that is first round will have 6 messages; second round will have 6 times 5 and third round will require 120 messages. So, as so that becomes 156 different messages are required when f is equal to 2 2.

(Refer Slide Time: 45:47)



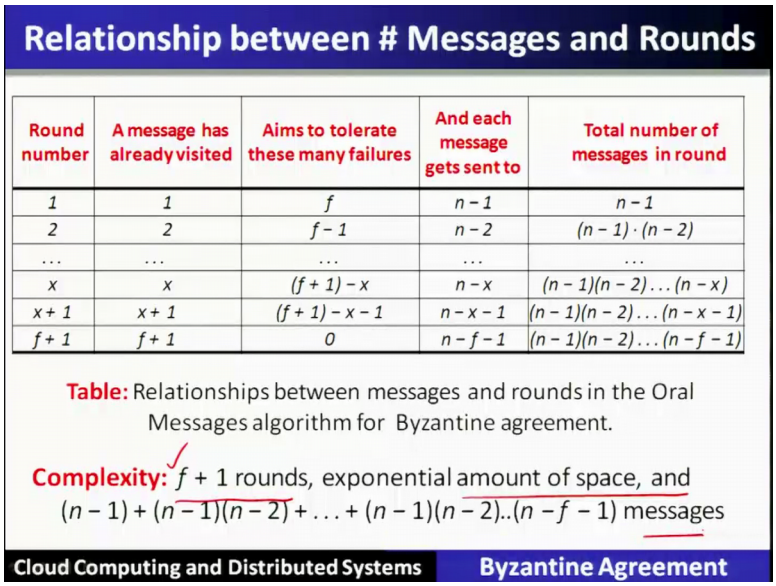
When f is equal to 3, you just see that entire worked out example is given in the slide is quite complicated.

(Refer Slide Time: 46:00)



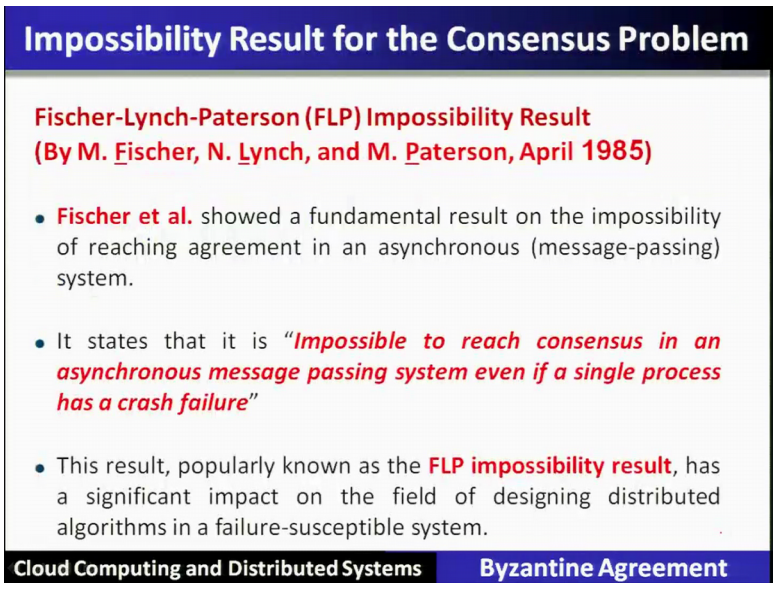
But we can summarize in the table how many messages are required. That is, 3609 too many number of messages are required 3 to tolerate 3 different faults. It becomes very, very costly in the system to support such kind of failure tolerance. But from this point onwards several studies are conducted and different solutions are there.

(Refer Slide Time: 46:28)



So, in summary we can see that the complexity, we can see that as per the time complexity it will be having f plus 1 number of rounds. But as whereas, the number of message complexity is concerned that also goes to an exponential number.

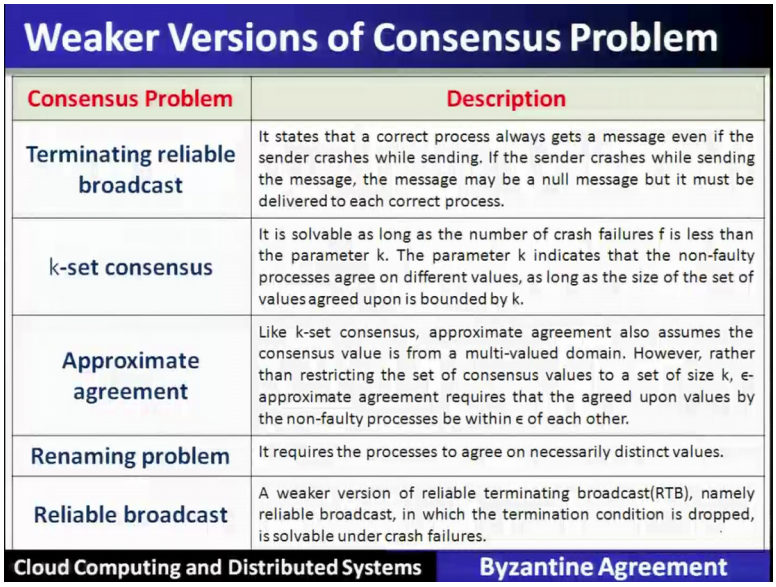
(Refer Slide Time: 46:54)



Now, agreement in asynchronous message passing systems with the failure; we have impossibility result for consensus problem in the asynchronous situation, given by Fischer Lynch and Paterson, FLP problem it is called. It is impossible to reach the

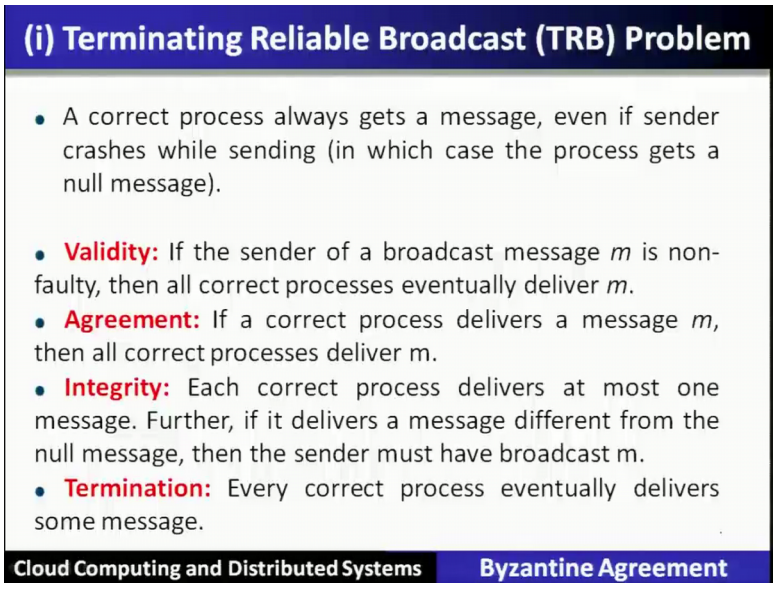
consensus in an asynchronous message passing system, even if a single process has the crash fault.

(Refer Slide Time: 47:19)



Now, this has led to the other development of a weaker version of a consensus problem, among them which is called a terminating reliable broadcast problem.

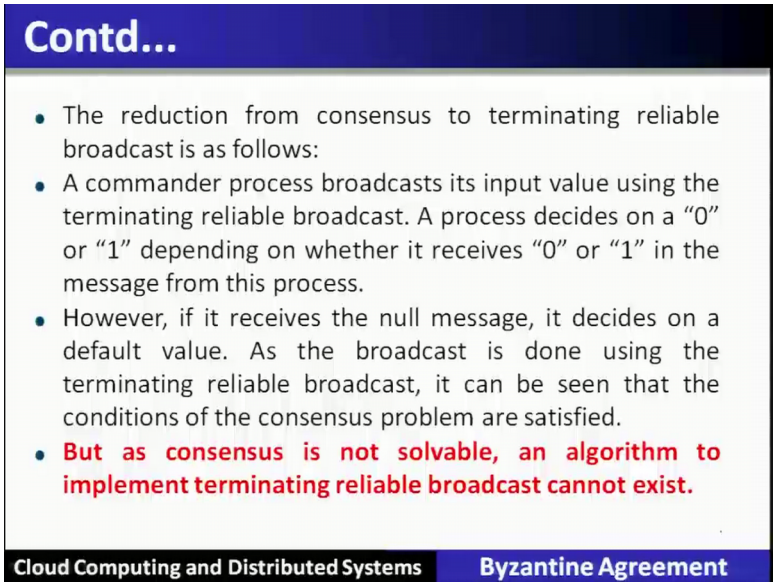
(Refer Slide Time: 47:32)



So, the correct process always gets a message even if the sender crashes, while sending there are 3 condition validity if the sender of a broadcast message m is non faulty. Then all correct processes eventually deliver if a correct process delivers a message then all

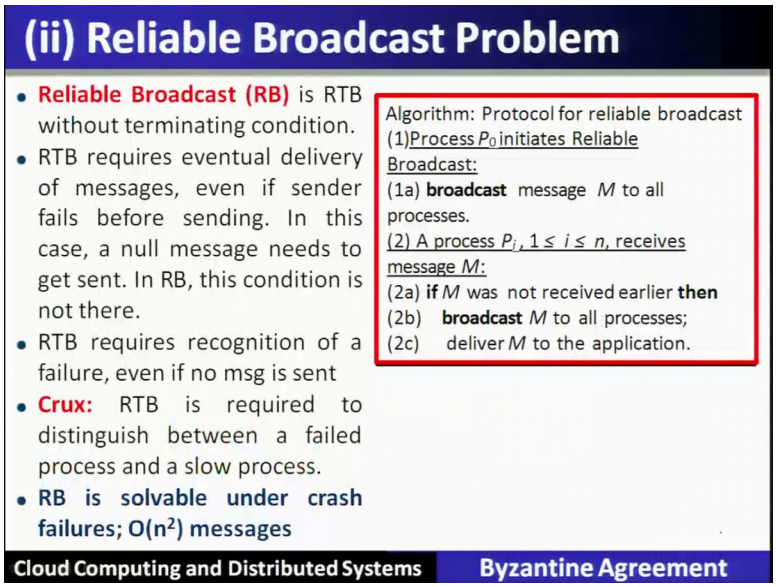
correct processes delivers it, that integrity and termination. Termination says that every process eventually deliver some message.

(Refer Slide Time: 48:01)



But here the consensus is not solvable the algorithm to implement terminating reliable broadcast cannot exist in this model.

(Refer Slide Time: 48:15)

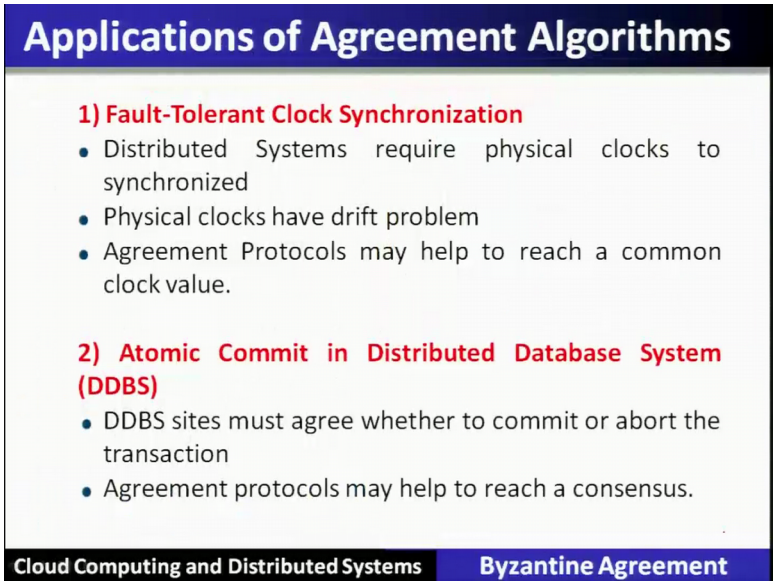


Therefore, a variation alteration of this where termination condition is not there, then it is called a reliable broadcast problem, that is the compromised problem it is. So, reliable broadcast problem is without terminating condition. So, therefore, it is preserving that

impossibility result, but modified the problem. So, reliable broadcast problem requires eventual delivery of the messages, even if the sender fails before sending. In this case null message needs to be get send. This condition is not there in the reliable broadcast problem which was assumed in the previous algorithm RTB.

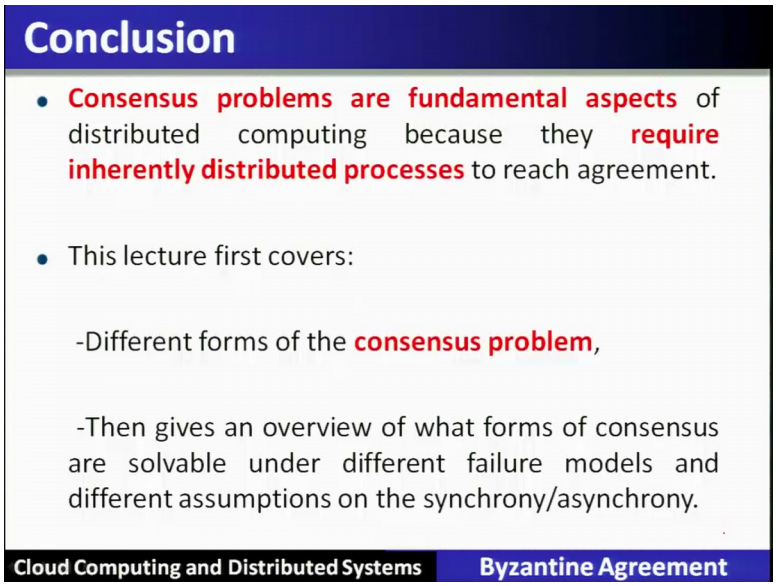
So, RTB is different this reliable broadcast RB is different than RTB. Why because, here the termination condition is not there, and this null message also is not there. This RB that is reliable broadcast problem is solvable under the crash fault with the complexity order n square.

(Refer Slide Time: 49:35)



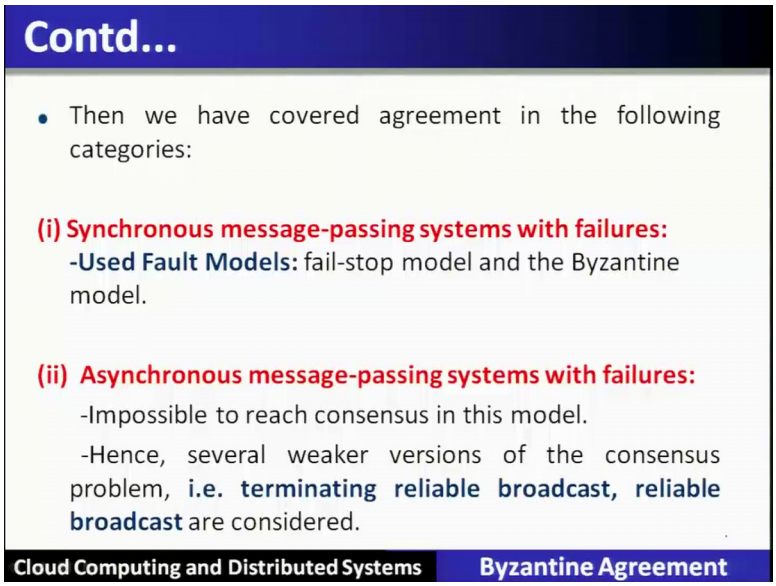
The applications of agreement algorithm are fault tolerant clock synchronization, atomic commit in distributed database systems.

(Refer Slide Time: 49:44)



Conclusion; so, consensus problems are fundamental in the study of a distributed system and the cloud computing system, where they require inherently distributed processes to reach to a common agreement. This lecture has covered different forms of consensus problem under the byzantine fault model. And we have then given the different form of consensus which are solvable under different fault models and in synchronous and asynchronous systems.

(Refer Slide Time: 50:27)



Synchronous message passing systems with failures can also support both fault fail-stop as well as byzantine fail model. In asynchronous message passing system it is impossible to reach consensus in this model hence the weaker versions of consensus problem we have seen; that is reliable broadcast problem. We have seen the terminating reliable broadcast TRB is impossible.

Thank you.