Real - Time Systems

Prof. Dr. Rajib Mall

Department of Computer Science and Engineering

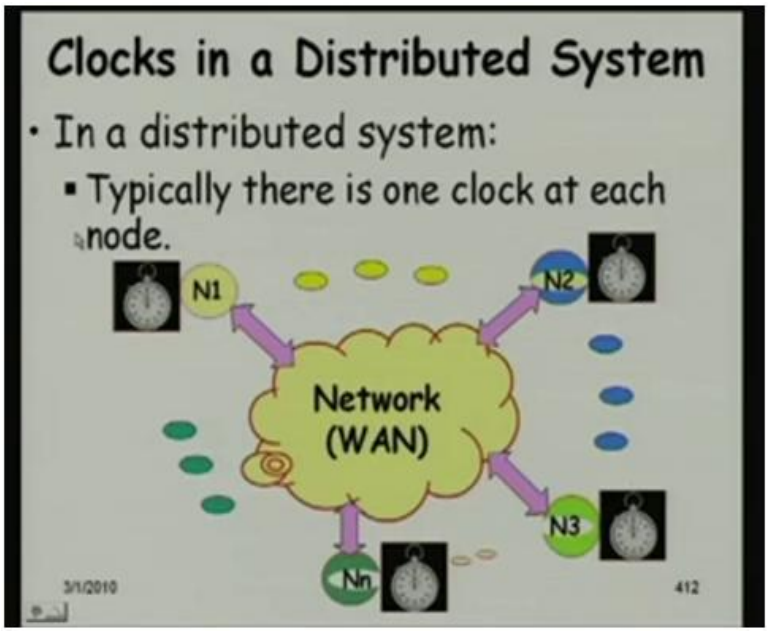
Indian Institute of Technology, Kharagpur

Lecture No. # 19

Clock Synchronization in Distributed Real-Time Systems

Good morning. So, let us get started with today's discussion. Last time, we were discussing the scheduling problem. Task scheduling and distributed real time systems. And we had said that these such systems, distributed systems are becoming more and more important in real time applications. So, we will see one of the major problems in these distributed real time applications is Clock Synchronization.

(Refer Slide Time: 00:55)



So, let us look at, how clocks can be synchronized in a distributed real time system. So typically, there is one clock at each node, and they are connected through a wide area network. So, the transmission time here is quite significant right, when compared to a multiple pressure system.

But what are the uses of a clock in a distributed system? Can anybody answer this question? What do you think?

Time stamping.

Yes. So, why do we need to time stamp? What sort of events? He said time stamping the events; what sort of events? And why do we need to time stamp? To know these sequence of materials and to have this time stamp.

Yeah, right, that is one thing that measures time stamping. To know what are the order of message that have been sent from different nodes or may be even from the same node right. So when the messages are sent from the same node; clock synchronization is not a problem, because it will time stamp on its own time and you can order it. But when messages are sent from different nodes, clock synchronization is important. So, time stamping is one. Any other application you can think of is?

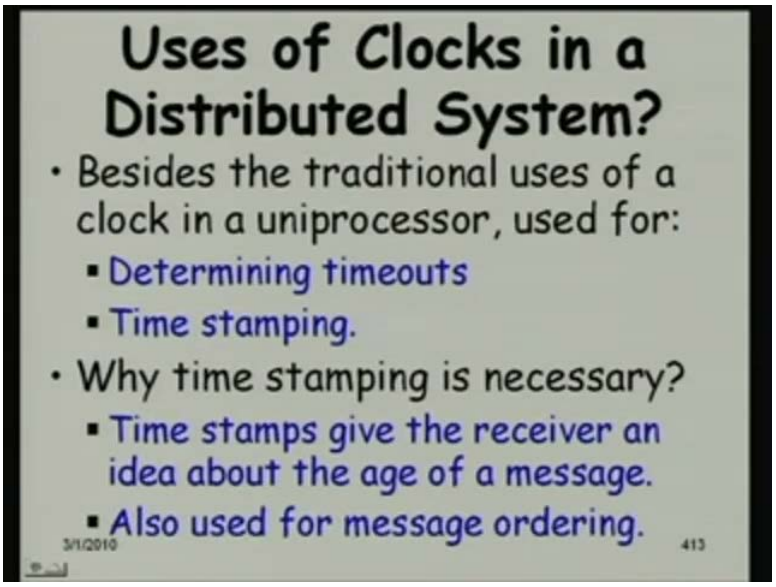
What is the latest information?

I mean why do we need, what sort of latest information you are talking of?

Node loaded and all. What is the latest information?

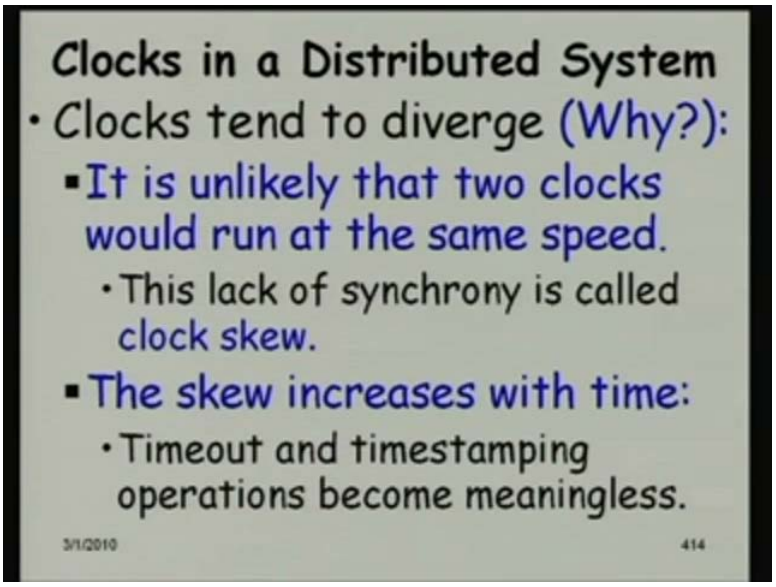
Yeah. So, he gives another one that finding the load position at an instant in different nodes. Yes.

(Refer Slide Time: 02:42)



So, let us see some of the applications and then we will see how to synchronize? So, we had seen that in the traditional use of a clock in a uniprocessor is to determine whether, a task completes in time right. We had said that there will be a time out, if it does not complete in time and some exception situations need to be handled, that is it niches its dead line. But in a uniprocessor, sorry in a in a distributed system, we need to determine the message time outs. You cannot weight keep on weighting for messages and it will time out. And also time stamping as you had said, and the time stamps give an idea about the age of the message, that also you had said and we can order the messages.

(Refer Slide Time: 03:32)



Now, the clocks in a distributed system, they tend to diverge it, said that there are clocks across different nodes and they tend to diverge. Why why do they diverge? Anybody has an idea?

(( ))

Sorry Error.

Error in the clock.

Error in the clock I mean, What do you mean by error in the clock?

(( )) standard value.

But why should it drift?

Because of that some environmental condition like.

So environmental conditions why should that affect clocks? He says environmental conditions make clocks drift, but why should they make the clocks drift at all.

(( ))

Not clear, can you please repeat?

Sir, Clocks are made of quartz crystals.

Quartz crystals.

So, like they are tend to be affected by the temperature, surrounding temperature.

Anybody has any other idea? He says that the clocks are crystal clocks, and the crystal frequency is affected by environmental conditions, such as temperature, such as temperature. Anybody has any other idea?

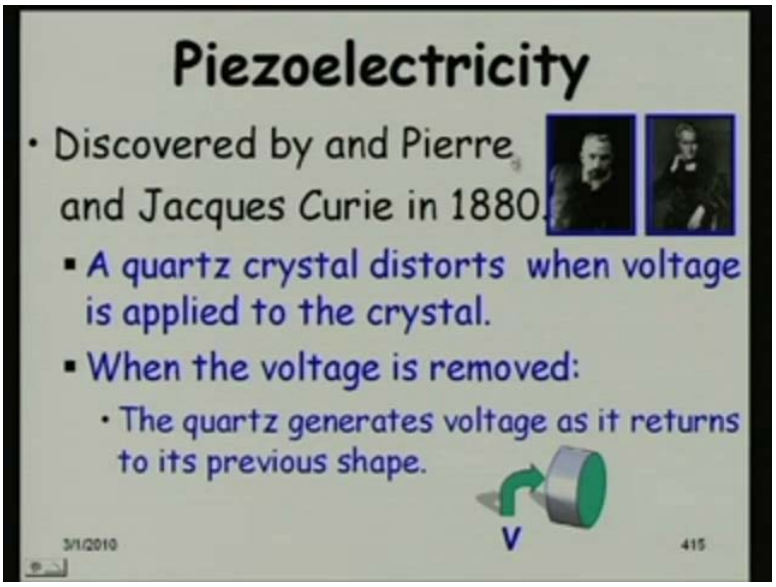
Power failure sort of treatment.

Power failure? No, power does not fail I mean, then the system will fail. See, after all the clock is basically a small integrated circuit mounted on the board. The processor board will find a small IC is the clock. Let us discuss this; May be, you will have a better idea. So, in a distributed system, it is very unrealistic to expect two clocks to run at the same speed. And this will will call it as clock skew.

Sir, why do not they do not have seems to be.

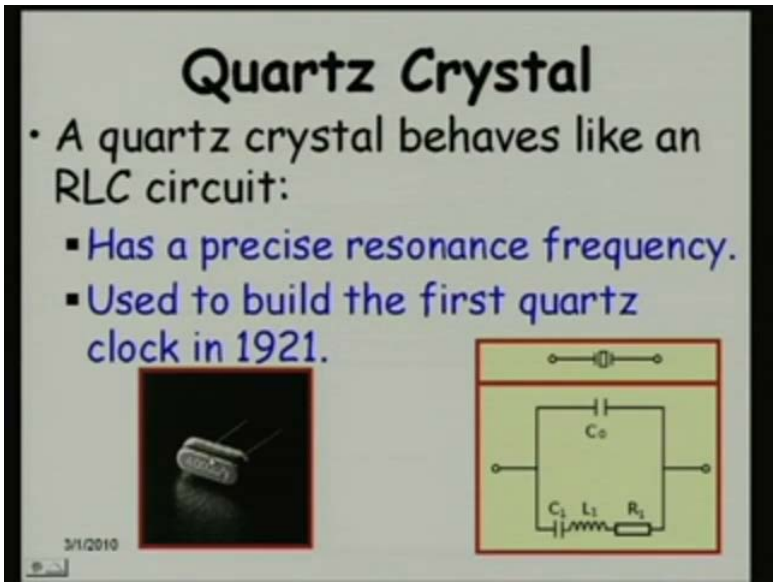
We will we will come, we will investigate. So, we will investigate the that is precisely the question, that I will asking that, why the clocks do not run at the same speed. We are going to investigate that. So, because they run at different speed, the skew increases with time, and once we are not sure of what is the skew, see if the clocks have a constant skew, like one clock differs from another by 1 second, another by 5 seconds etcetera, if these are constant, then it is not a problem. We can know the precise time of the different nodes, but this keep on increasing and therefore, time out and time stamp operations become meaningless.

(Refer Slide Time: 06:34)



As, some of you were mentioning, the clocks that are there in the computer are quartz clocks, based on the principle of Piezoelectricity, discovered in 1880, long back by the brothers Pierre Curie Pierre Curie and Jacques Curie. The main concept here is that, when a voltage is applied to a quartz crystal, it distorts. So, these are quartz crystal, if you apply a voltage, it will distort. And if you withdraw the voltage, the quartz would return to its previous shape and in the process it will generate in turn the voltage right. So, you had applied a voltage, it has distorted and you remove the voltage, it comes back from it is distorts into the original shape and then it generates a voltage. So, these are the principle in 1880, but much later about 50 years later somewhere, around 1925, or something 23. This was applied to develop a clock.

(Refer Slide Time: 07:54)



The idea is that, a quartz crystal is like a RLC circuit, capacitance, inductance and resistance. And you know that every RLC circuit has a precise resonance frequency. So, just see here this is the quartz crystal, and it acts like a equivalent circuit of this is this one and RLC circuit.

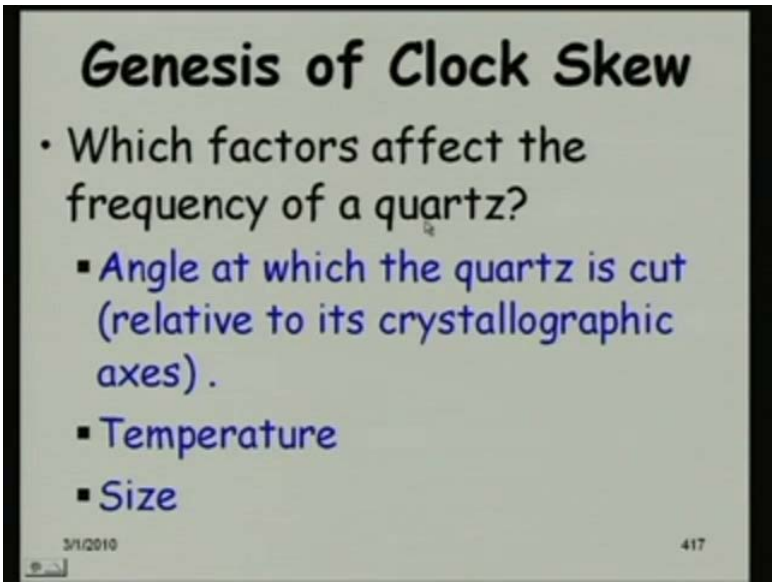
Sir, what is C 0?

These are some capacitance values; C 0, C 1 are capacitance; L 1 is a inductance; R 1 is a resistance.

Sir, this RLC is the bottom.

No, no this entire thing is a RLC circuit. This is a equivalent of this, it is a RLC circuit. It will have its own resonance frequency right. So, how to model this as in this form etcetera, let us not worry about it, but this is a model of this. Let us assume that, there is a capacitance here across and there is also capacitance, inductance, resistance involved here. Now, the first quartz clock was built in 1921, using this principle, and now you find it in the form of a IC mounted on your mother board of a computer.

(Refer Slide Time: 09:26)



But, given these that these are quartz crystals, why should the frequency change? The clock frequency change across different nodes, many reasons: One is that the angle at which the quartz is cut relative to the crystallographic axis that determines the frequency right.

So, you can guarantee that two quartz crystals or cut exactly on the same angle. Like, temperature and the size of the crystal. So all this make the frequency very widely, and you know, even if you maintain the temperature and same different locations right, you keep them in the same room still the clocks will diverge. But, just out of curiosity nothing to do with this discussion, that why quartz; why not see, what what is special about quartz? Why do use this as the clock? Why not any other material?

We have some perfect crystals that we have so.

Not really, quartz is not the perfect crystal structure.

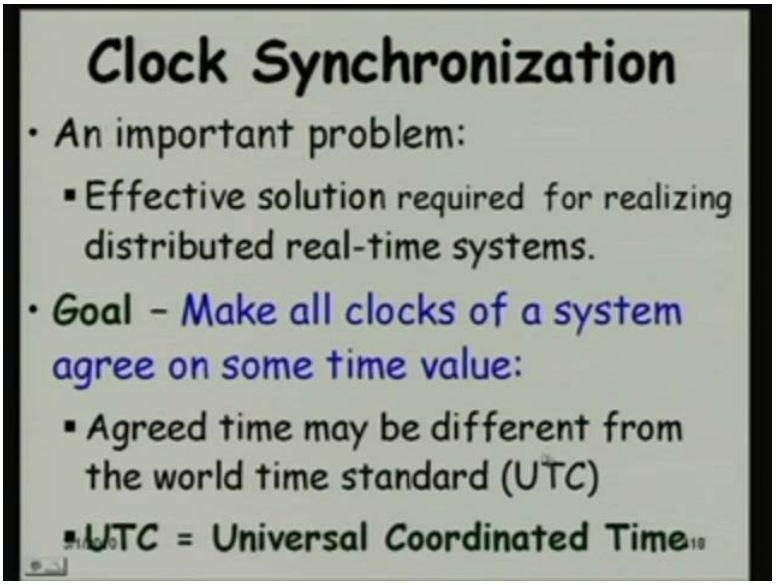
(( ))

I am sorry.

(( ))

No, but there are many other materials they will behave similarly. It is not the only Piezo-electric material. So, the reason is that, it has elastic property right. It is a one of the and also it is not a good conductor of temperature; otherwise you know, temperature will very largely affect its frequency; today, the steel or something, its frequency will largely be determined by temperature, even though steel is also elastic. So, quartz is a elastic material, even if it gets distorts small amount, it will return to its original shape and also it is not a good conductor of temperature.

(Refer Slide Time: 11:42)



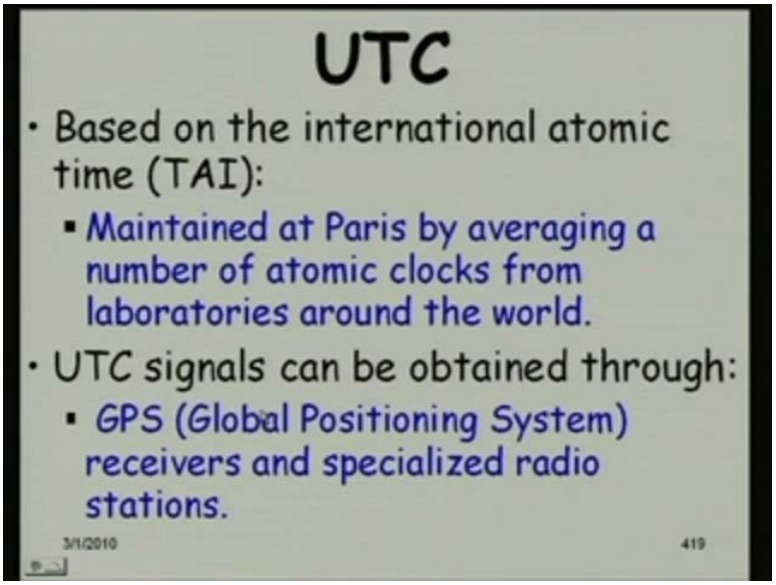
So, whatever you do, keep them in the same temperature etcetera, the clocks will diverge. Given that fact, how do we make clocks agree on some time value? So that we can know the précised time in the different nodes: one node can know the time at which some event occurred. For example: he was saying one of the event might be, what was the load at some instant? So, the agreed time that we will discuss can be different from the wall time standard. The wall time standard is called as UTC, Universal Coordinated Time, but this not the full form, is not it? UTC can be otherwise it should have been UCT. Anybody knows why it is UTC is called as Universal coordinated time?

May not, that may not be any.

If you have look at it its very funny actually. The French for the universal coordinated time is CUT I did, I mean that is a French word actually. And because there was a

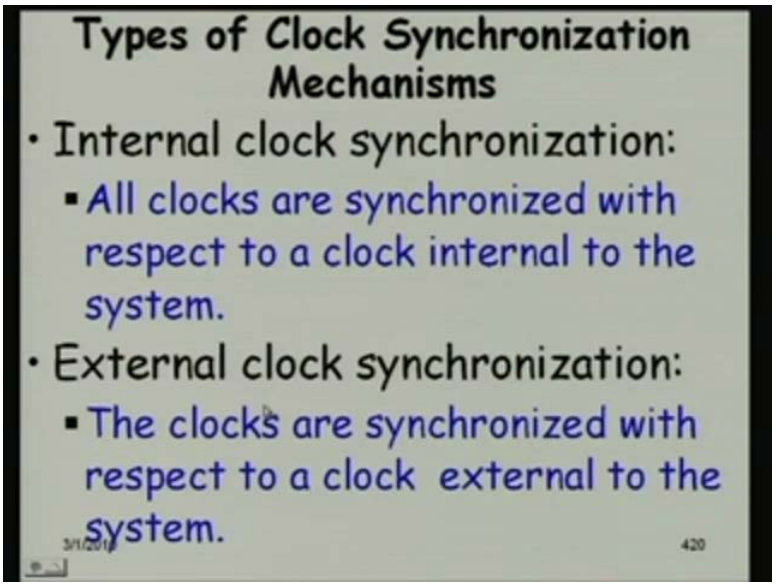
controversy, that it should be named as Universal Time Coordinated Universal Time, or Time Coordinated Universal or something right. So they took a middle path and they just made made it UTC, which neither is a full form in French, nor in English anyway. So, lets it is Universal Coordinated Time:

(Refer Slide Time: 13:36)



Based on the International atomic time same here TAI. It is maintained at Paris, by averaging a number of atomic clocks from laboratories around the world. And UTC signals can be obtained through GPS receivers; and also these are transmitted by some special radio stations. So, one thing is that the clocks can be made conformant to the UTC, but that is, that would be much more expensive than making them agree on some standard time value right. So, UTC is not used in most of this Distributed systems. They determine a time based on their own times.

(Refer Slide Time: 14:25)

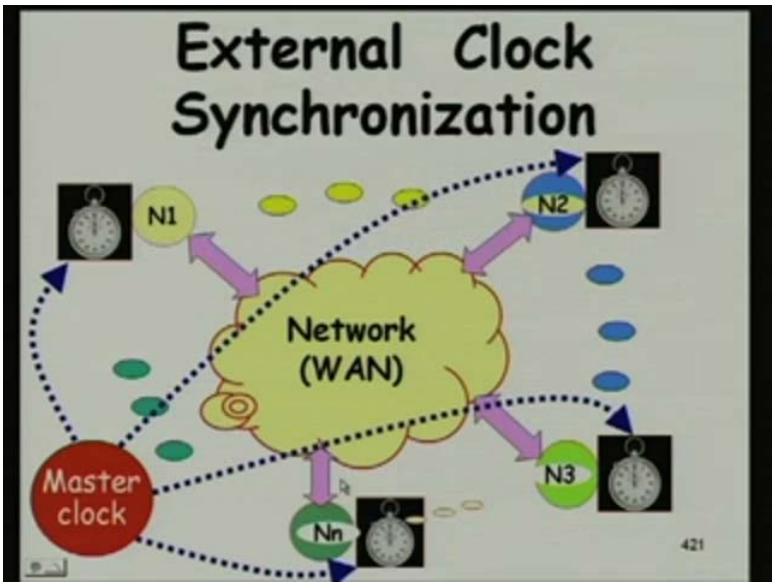


How can they should be made to?

We will that is what we are going to discuss? How the clocks will agree on a standard time? I will just going to discuss that. So, those are the clocks synchronization mechanisms; roughly there are two main mechanisms. Now, one is called as a internal clock synchronization; the other is called as the external clock synchronization. As the name says, internal clock synchronization will be basically the time value will be determined based on some internal clock. Clock Internal to the system; may be one clock, may be many clocks, but all the clocks are internal.

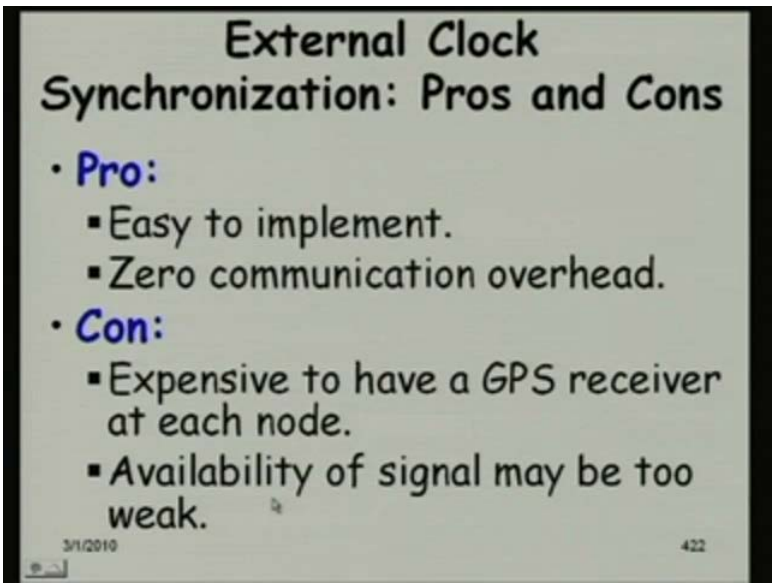
Whereas, in external clock synchronization; we use external clock, like the UTC. So, the in in a internal clock synchronization, all clocks are synchronized with respect to a clock internal to the system. Whereas, in the external clock synchronization the clocks or the system are synchronized with respect to some clock, which may be external to the which is external to the system. For example: UTC, but it can be any other clock.

(Refer Slide Time: 15:38)



In an external clock synchronization, we will have a master clock which can be UTC or any other clock, which, transmits time to all these clocks, frequently, and they keep on setting their time according to the master clock. The implementation this of this is simple, but unfortunately it is not used; as we were saying…

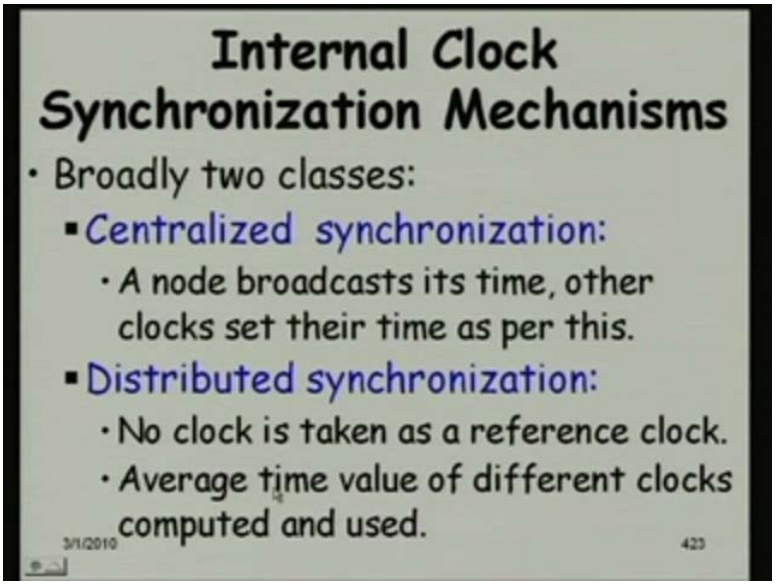
(Refer Slide Time: 16:08)



Let us see the advantages and disadvantages of the external clock synchronization: easy to implement; just keep on setting their time, sensing the time and setting it zero communication over head on the internal network, because these are broadcast to these

nodes. But the disadvantage is that, it is expensive to have a GPS receiver at each node, and possibly that is the reason, why external clock synchronization is not used in distributed real time systems. And also, depending on the location where it is deployed, the availability of the signal may be weak; it may fade and so on.

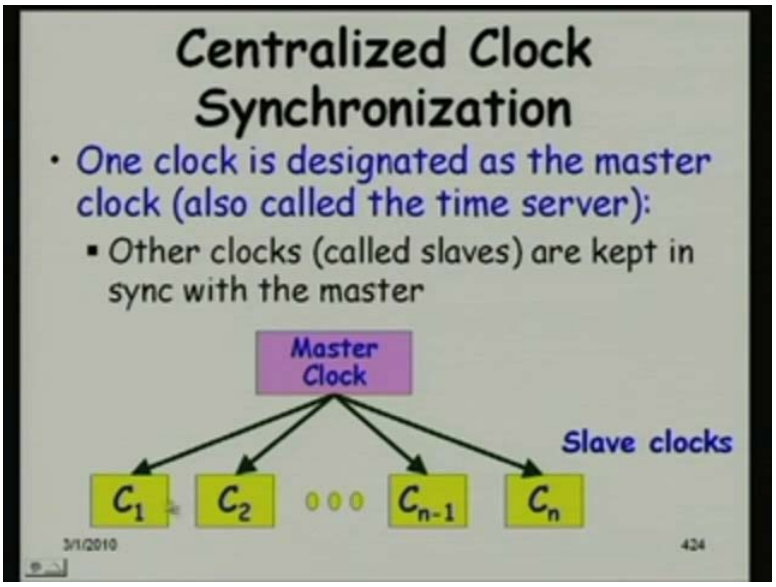
(Refer Slide Time: 17:00)



So, the one that is actually used is internal clock synchronization and if we look at internal clock synchronization mechanisms, there are two types: one is the centralized synchronization, and the other is distributed synchronization. In a centralized synchronization as the name says, we have one master clock - one of the clock of the system will be, it will be the central clock or the master clock and the signals, the clock values from that will be used to synchronize all other clocks, whereas in a distributed synchronization; the clock values across many clocks will be used to synchronize the different clocks.

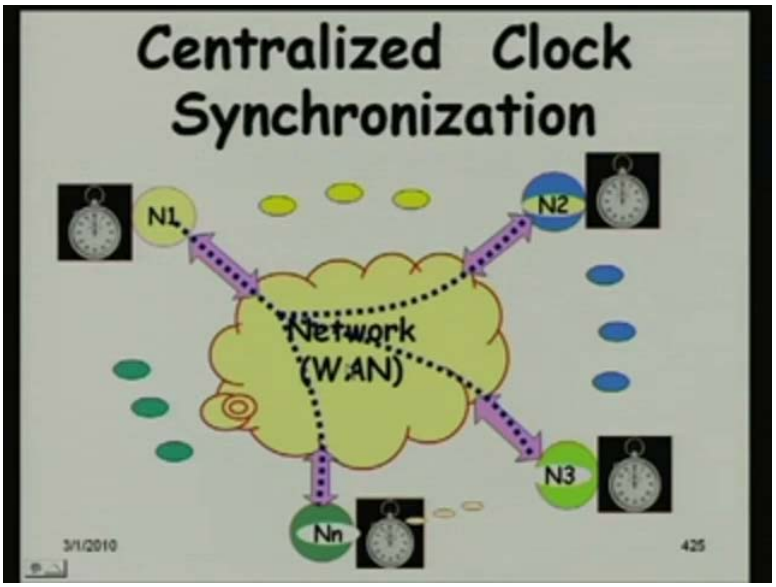
So here, one of the clock, the the node will broadcast its time. The node of the clock broadcast its time, and other clocks the different nodes will their set their time according to this. Whereas in a distributed clock, no clock is taken as the master clock or the reference clock and the the clock values, the average of all clock values not all clock values as we will see, many of the clock values are used to set the time, which is agreed by all clocks.

(Refer Slide Time: 18:29)



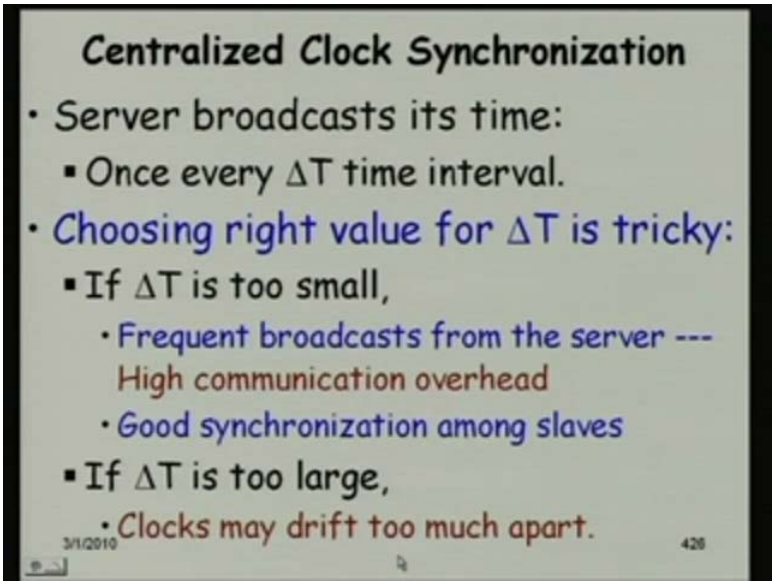
Now, first let us look at the centralized clock synchronization, because we will see that this is also not very popular. It is not that it is not used, but not very popular, because it has some problems. So, here one of the clock is designated as the master clock, also called as the time keeper or a time server, and the other clocks are slave clocks and they keep in sync with the master. So, this is the master clock, and all other clocks this keeps on send, transmitting broadcasting the time values and the slave clocks keep on setting their time, according to the master clock, simple mechanism.

(Refer Slide Time: 19:15)



The system, it just keeps on broadcasting, the time and they keep on setting.

(Refer Slide Time: 19:23)



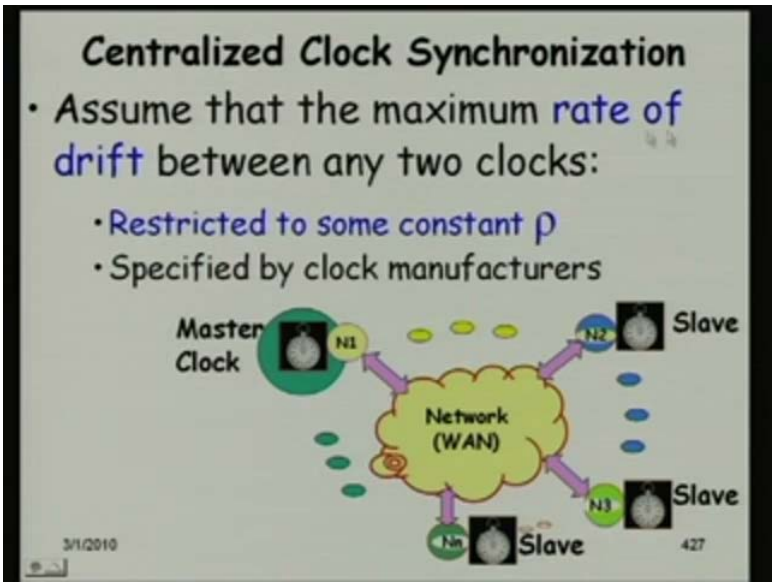
The server broadcasts its time, every delta T time interval. Now, let us see slightly more detail of this. So periodically broadcasts and the cycle is delta T, but one of the main problem here, not problem one of the issues that we need to address, if we are going to use a centralized clock synchronization is to fix the value of delta T, how frequently it should transmit? Because if there is transmits too frequently, it is a problem, and what is the problem it transmits?

(( ))

Yeah, communication over it simple thing. The clocks will be in synchrony, but there will be large communication over it and if it transmits less infrequently sorry less frequently then the clocks will go out of sync.

So, delta T is too small their frequent broadcasts from the time server and result in high communication over head, but there will be good synchronization among the slaves. On the other hand if delta T is too large, then the clocks will not be synchronized towards the end of this delta T time interval right. They will get synchronized and as the time progresses, they will become unsynchronized; they lose synchrony and again after delta T they will get synchronized right. So, during this interval delta T, the clocks might drift too much which will be unacceptable.

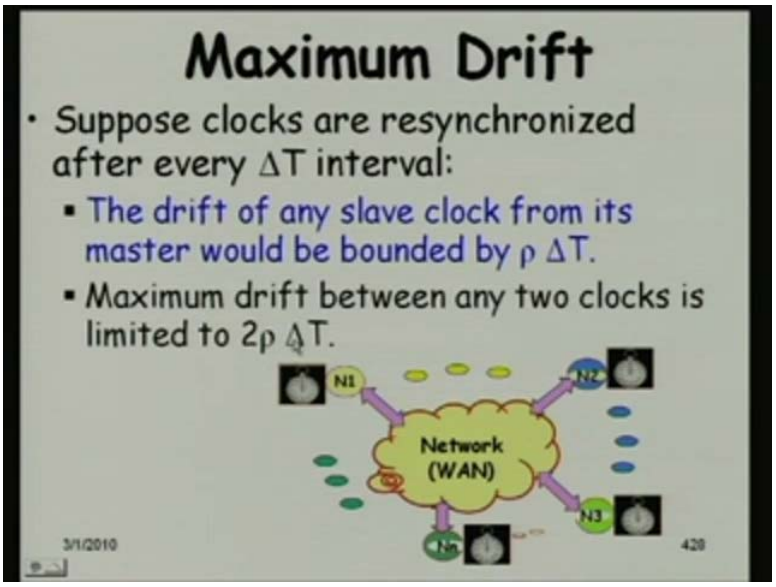
(Refer Slide Time: 21:11)



Now, let us assume that the maximum drift between clocks, that is permissible by our application is some rho, see we had earlier discussed about scheduling issues and so on, and there we had assumed, I think some communication time we had assumed right in distributed systems. But this is another thing that we need to assume, that what is the maximum drift between clocks and that we need to bound. Let us say, compared to our task completion times which is of some milliseconds, several milliseconds, we need to have a rho small enough, so that it will not really affect the schedulability of the tasks right. We need to fix it to some value it should not exceed rho.

Now, what the clock manufacturers actually give, for this clocks is actually the rate of drift; it is not the maximum drift, but the rate of drift. The rate of drift is how much drift between the clocks will occur per unit time, and that is rho. I am sorry, I think I told you that is the maximum drift no, this is the rate of drift maximum rate of drift. So, this is what the manufacturers will tell you. That see, we have manufactured these clocks and the quartz is got precisely and all those, and but we can guarantee you, a maximum rate of drift, that is drift per unit time is rho; it can be less than rho.

(Refer Slide Time: 23:01)



Now, suppose the clocks are resynchronized after delta T interval, then the maximum drift of a one clock (( )) that is one slave from the master will be rho delta T. Do you agree with this?

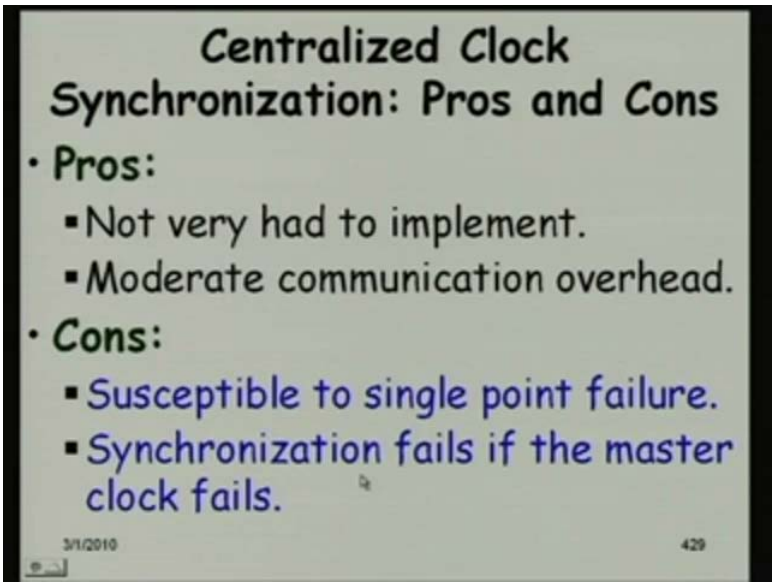
Yes.

Because the rate of drift is rho, and they synchronize set their time with the master, and let us say the communication time is known to be constant, because if the communication time is you can fix a bound on that; then again this will not work. Now, the maximum drift between any two clocks is 2 rho delta t. So, why is this 2 rho delta T.

One may drift.

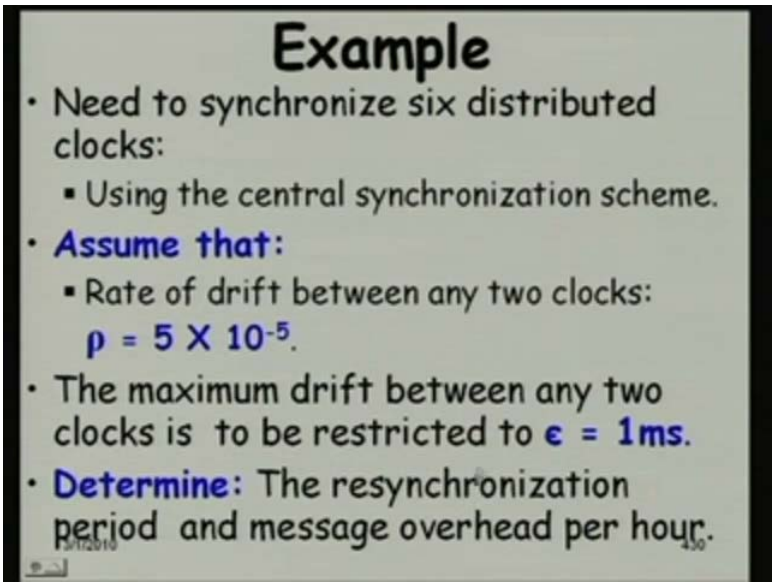
Yeah. So, one may drift become a slow clock and another is a fast clock. So, one has becomes slower by rho delta T from the master and another has become faster by rho delta T. So between these two slaves it will be two rho delta T.

(Refer Slide Time: 24:06)



Now, let us see what are its advantages and disadvantages: the advantages is that not really very difficult to implement; just designate one clock, good clock is the master which transmits the time frequently, need to calculate the time frequency at which it will transmit. The communication overhead is moderate, because its after all one clock, one node just keeps on broadcastings broadcasting every few seconds. But one major problem of the centralized clock synchronization is single point failure. What if the master clock fails? So, if you application is actually is a safety critical. Failure cannot be tolerated; then this centralized clock synchronization is not a good solution; the clocks will fail if the master clock fails.

(Refer Slide Time: 25:14)



Now, let us just work out an example using this simple thing that we discussed. Suppose we have 6 clocks in a distributed system, and we want to use the centralized synchronization scheme and we can assume that the rate of drift between the master and another clock or between any two clocks is 5 into 10 to the power minus 5; do not have a unit here, because its rate of the drift. Rate of the drift this much time per unit time. Now, let us say we want to maintain the maximum drift between any two clocks to be restricted to 1 millisecond.

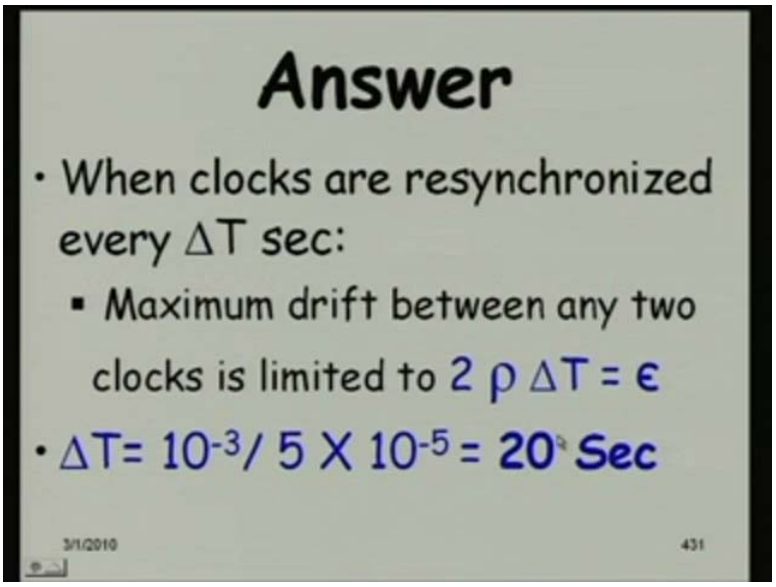
So, what is the frequency, with which the master needs to broadcast and also need to calculate what? How many messages need to be transmitted per hour? Can you do it or should we look at the answer.

2 rho delta T equal to 1.

2 rho delta T equal to 1. Yes…

(( ))

(Refer Slide Time: 26:30)



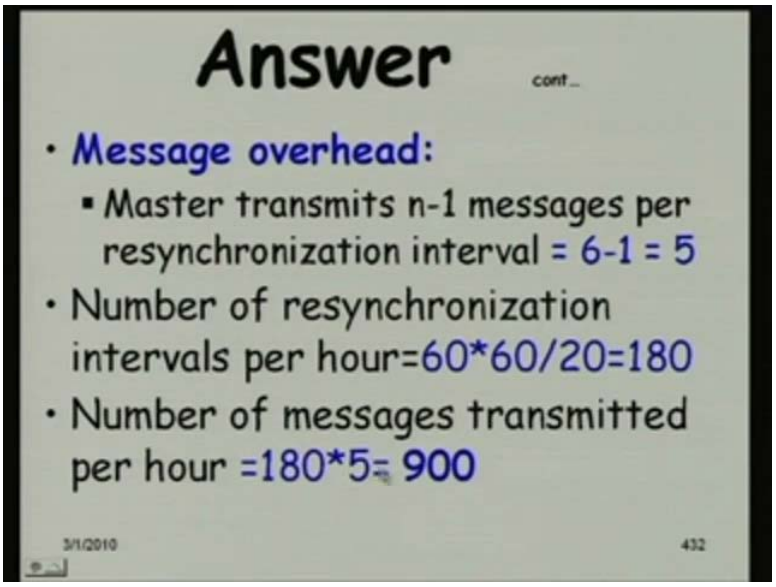
Not very difficult. So, he already has said that 2 rho delta T is equal to epsilon which is equal to one millisecond or in other word delta T is equal to10 to the power minus 3. it is 1 millisecond right divided by 5 into 10 to the power minus 5 which is equal to 20 second. So, every 20 second, it needs to transmit the clock value.

But what about the message overhead? The master needs to transmit n minus 1 messages per resynchronization interval. So, where is the calculation all right, in the previous slide some of you were mentioning something, where is the calculation all right. Yes, 10 to the power minus 3 by 5 into 10 to the power minus 5, so 100 by 5.

Divided by 4 divided by 2 is there.

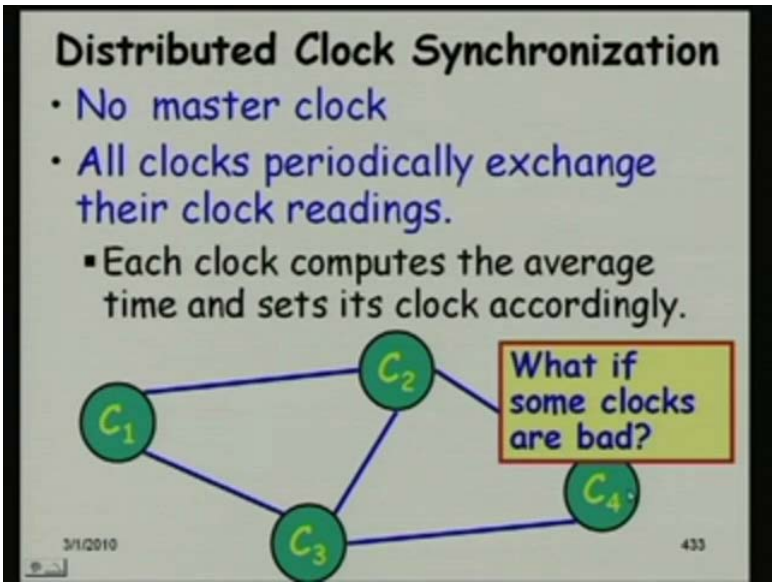
2 is there, yeah yeah right, is that 5 into 10 to the power minus 5. Yes, I think

(Refer Slide Time: 27:54)



I that is mistake I have done here, forgotten 2 here, when I have taken this side. So, 2 into 5 into 10 to the power minus 5 right. So, this will become 10second right . So, that is a mistake I have done here, will calculating this anyway. So, that is will be 10 second; transmission broadcast time is 10 second, now how many messages need to be transmitted, the master transmits 5 messages. For each synchronization resynchronization interval, so that is 60 into 60 divided by 10, it should be right. So, which will be 360 and then a number of messages transmitted per hours will be 360 into 5 right1800.

(Refer Slide Time: 28:27)



Simple thing. Now, let us see the distributed clock synchronization. So, here as we were saying is no master clock. And here all the clocks they exchange their time values and each clock computes the average of the time values, and then sets it is clock accordingly; this is the simplest thing that can happen.

So, there are many clocks here, the distributed system you just keep on broadcasting their own time frequently, periodically and frequently, and then after a clock receives all the time values it computes the average.

Help time (( )).

All the clock values that are received.

All the…

All other nodes have sent values, is it not? Including its own clock. So, it just finds out the average of all clock values in the system. See, everybody computes the average of clocks right C 1, C 2, C 3, C 4, these averages; these also averages C 1, C 2, C 3, C 4.

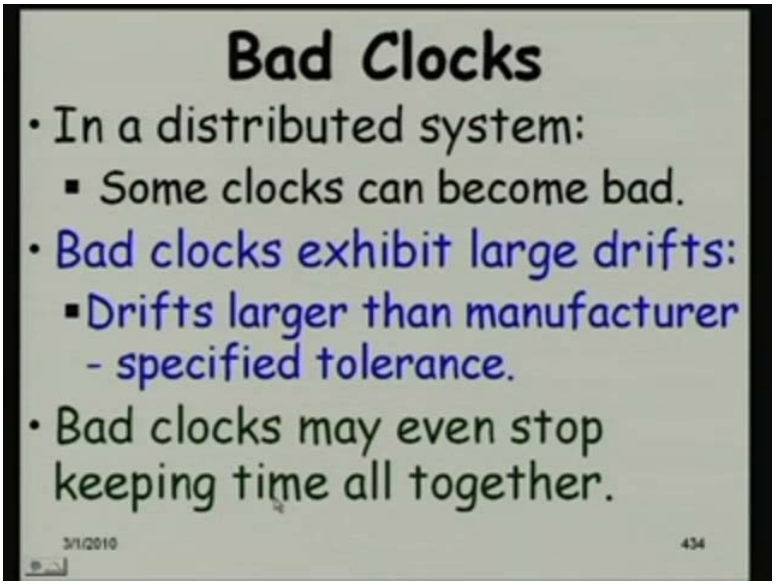
So, if all are averaging, all the clocks then they will agree, is it not? Then the clock they will agree on the time, because after all they are averaging the same set of clocks. But

what if some clocks are bad, a bad clock may not send time or it may send erroneous time right, instead of sending of a 5 hour and 40, it will send some 6 hour and 50.

But still there is more the average will be still same.

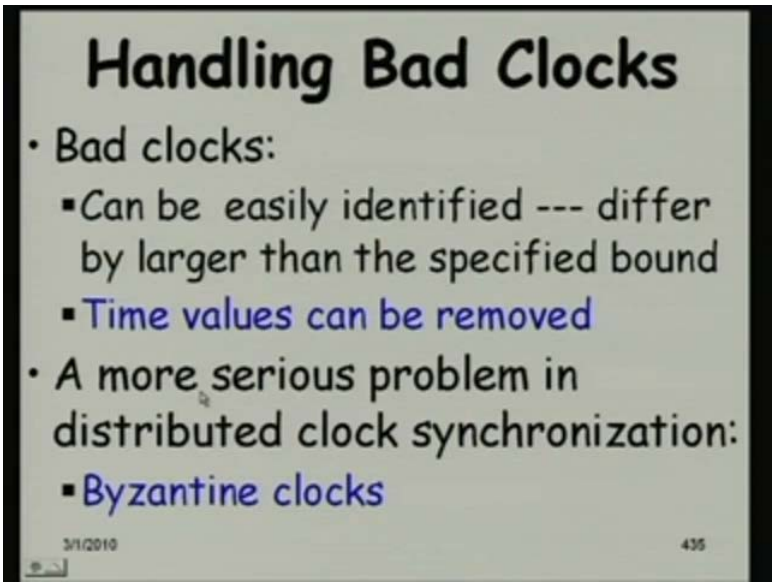
No, but the bad clock might not always send 6 hour 40 right. It might send 7 hour sometime; it may not send any time, some time it will fail right; it was sending and then the bad clock failed. So, then everybody will have to reset their time back, is it not? So, once a clock becomes bad the time value will jump is it not? Either plus or minus. So, the clock was already differing and then suddenly it has stopped calculating sorry stopped sending the time that will be a problem; it is not that all clocks can still agree and that is not a problem.

(Refer Slide Time: 31:03)



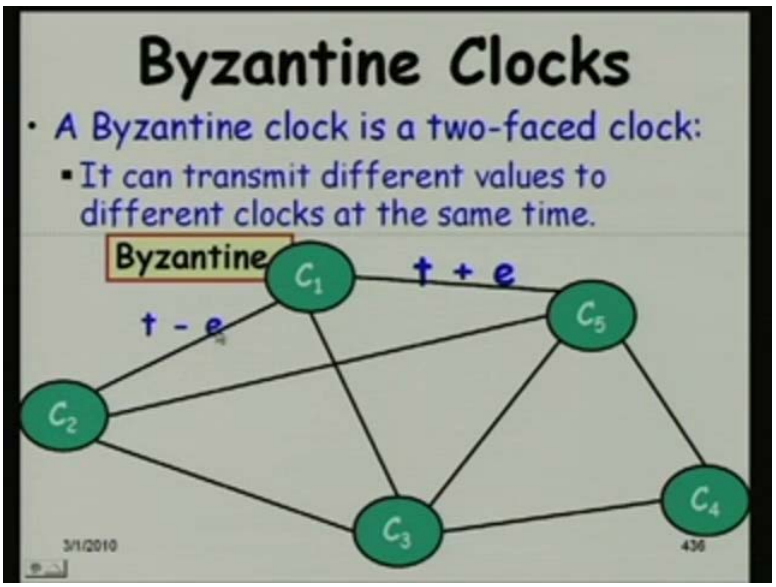
So the solution he says there that see it is not very difficult to handle that even if after sometime some clocks can become bad, but the thing is that the bad clocks can be identified by large drifts. So, if it is 5 hour 40 and it has sent a time 6 hour 50 can immediately each node will know, this is a this is a bad node, bad clock; and they can eliminate it from computation. So, if a clock drifts larger than the manufactured specified tolerance, then the time values can be ignored, and the bad clocks may even stop keeping the time altogether; they might not transmit the time does not matter.

(Refer Slide Time: 31:50)



So, handling bad clocks is easy; they can be easily identified, because they differ by large values and the the time values can be removed easily, but a more serious problem is the byzantine clocks; bad clocks are not a big problem can be easily handled; the problem is the byzantine clocks.

(Refer Slide Time: 32:21)

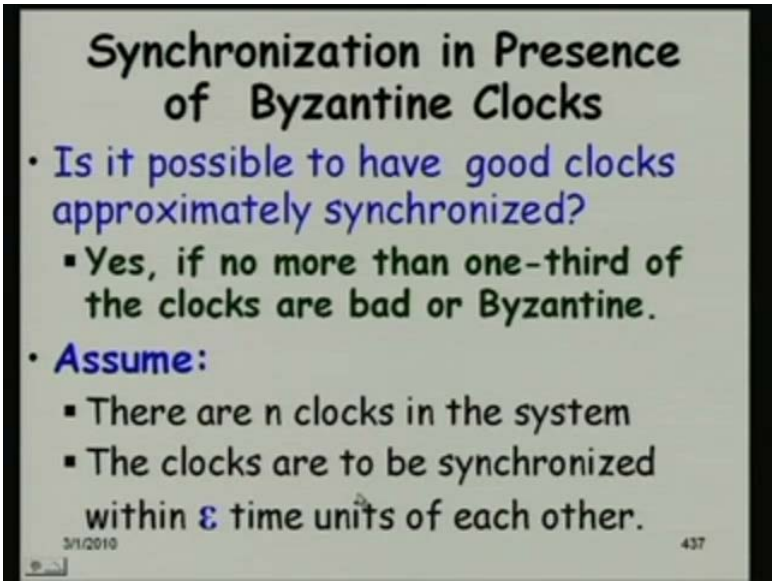


A byzantine clock is a two faced clock; actually, it is based on some story of byzantine generals; I think some Rome or somewhere, near Rome actually. There was a kingdom byzantine and there the generals go to different places, their position in different places

and they coordinate their attack. So, the problem is that what if some of the generals try to confuse the other generals and trick them at attacking the wrong time. So, they tell one general one thing or they confuse them. A byzantine general will confuse the other general he will tell one general that see, it is not the time to attack, and will tell another general that see, you should attack now right; so that they will lose it.

So, a byzantine clock transmits different values to different clocks at the same time; see here, C 1 is a byzantine clock and it is transmitting some t plus e to C 5 and t minus e to C 2; see if e was large enough let us say t is the time at C 5. If e was large then C 5 would have eliminated that, but it is a acceptable time value within the bound; but if here also it is within bound, but it is showing two different values. Now, if they compute the average, there will be a difference in time, but how much difference in time. If if there is a byzantine clock and we know that the maximum drifts between clocks is let us say rho into delta T. So, what will be the effect of a byzantine clock on the clock Synchronization? We will, we will just discuss about that.

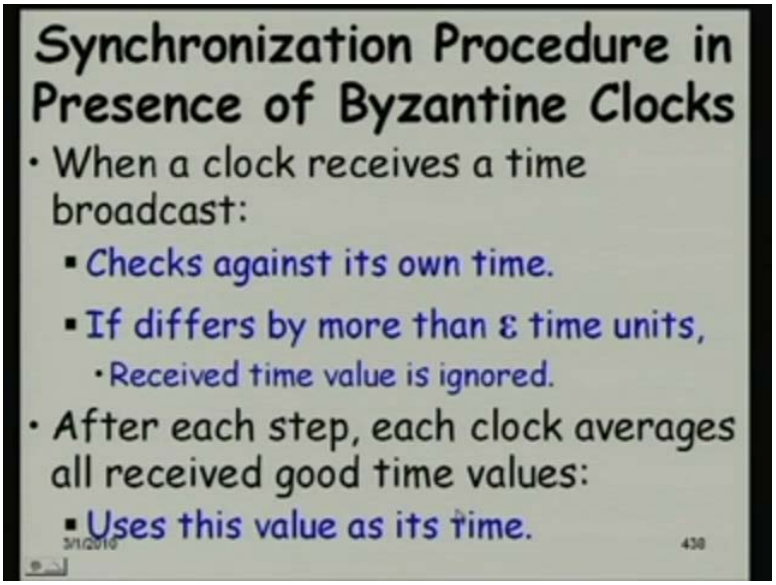
(Refer Slide Time: 34:42)



But the first question is it, is it possible to have good clocks synchronized in the presence of byzantine clocks? Answer is yes, we can synchronize them if no more than one-third of the clocks are bad or byzantine. So, out of 10, if we have 3 clocks bad, then we can synchronize them, but if out of 10, 4 are bad, then whatever we do, we cannot keep them synchronized or let us say we have 9 clocks, if 3 are byzantine, then we cannot

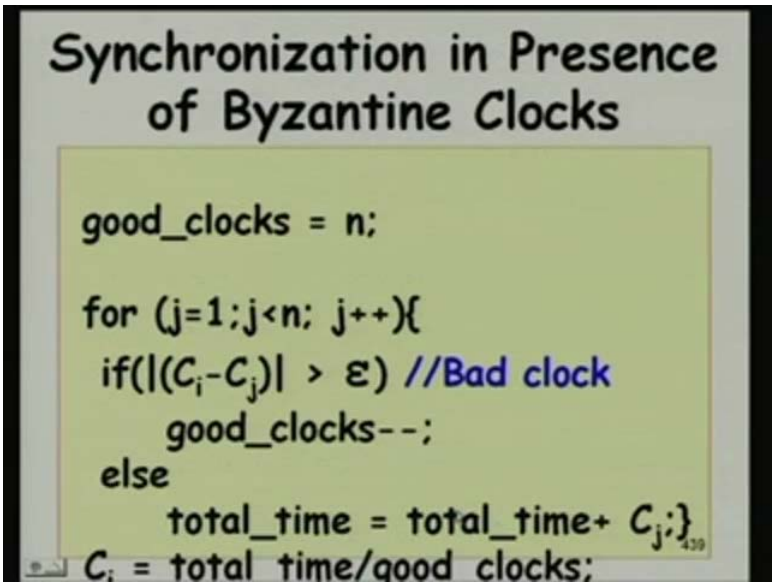
synchronize them; it should be less than one-third. Now let us show, how to do this? Why one-third of the clocks? If they are bad we can synchronize and no more than one-third. Assume that there are n clocks in the system and the clocks are to be synchronized within some epsilon time units of each other.

(Refer Slide Time: 35:58)



Now, when a clock receives a time broadcast, it checks against its own time. If it differs by epsilon time units, then the received time values are ignored, because it is has drifted more than epsilon, right the drift is to be restricted to epsilon, is it not? So, if it has drifted more than epsilon, the clock is bad, time value is ignored; but this does not eliminate the byzantine clocks, right it is only eliminate the bad clocks; and each clock averages the time value is good times and uses this as the time value.

(Refer Slide Time: 36:44)



So, this is the pseudo code for that simple thing here. See, if the good clocks to start with this n, right a global variable said to n. Now, it computes, this is one interval for synchronization. So, what it does is, if the clock drift is more than epsilon, it is a bad clock, it just reduces the number of good clocks and if it is a good clock, if it is within this bound, it just computes the total, and then it just computes the average. So, that will be total time divided by good clock right; C i is its own time, and j is the time received from all other clocks, n minus 1 other clocks. So, it will average out the time values of n minus 1 clocks, and then had its own time, and then average it out right.

Sir, in the previous slide, yes, that C I minus C j. Yes, I t should be greater than epsilon.

What did we have? Let us see.

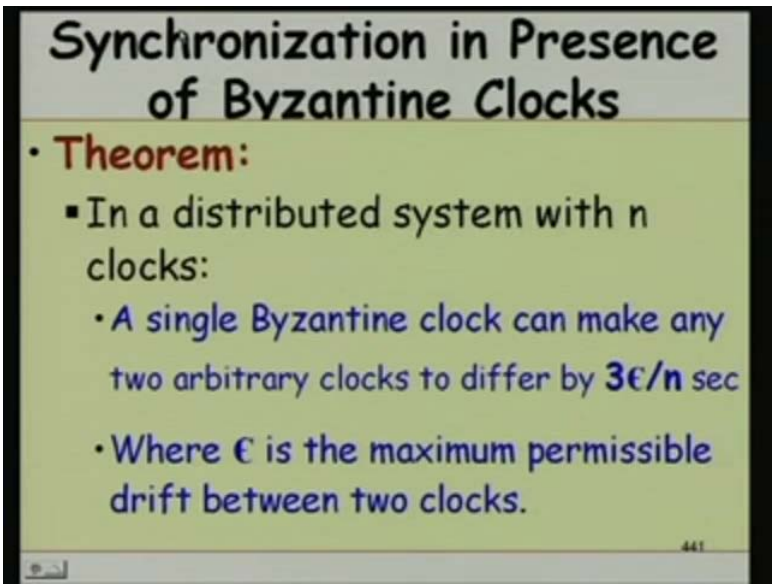
C i minus C j; so, if it is greater than epsilon, yes, then we will call it a bad clock. Yes. Suppose, C I has drifted by epsilon C j has drifted by 2 epsilon.

No, no see every clock see our goal is to keep two clocks synchronized within epsilon. We do not bother about which is drifted. So, if the time difference between the clock that has received it and another clock is more than epsilon, then the other clock will be assumed to be bad, is that ok?

Suppose one has drifted the epsilon otherwise…

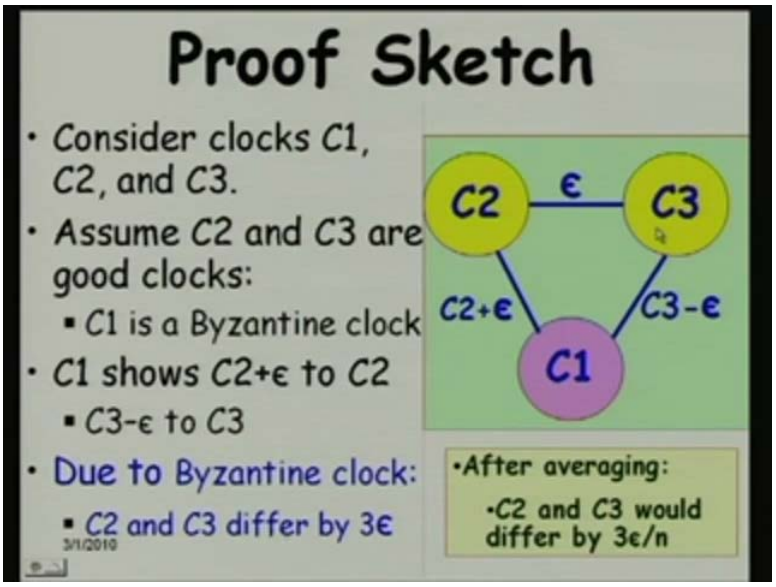
Do not do not matter. I mean, it does not matter, whether the clock has drifted or not; our goal is to keep them synchronized within epsilon that is given. So, within delta T, if they have drifted more than epsilon, you know that the other clock is bad. This clock assumes, the local clock assume that the other clock is bad, it has drifted too much more than the permissible amount, is that ok?

(Refer Slide Time: 39:26)



So now, let us look at a result; the result is the effect of a single byzantine clock. So, this we will state in the form of a theorem that a single byzantine clock can make any two arbitrary clocks to differ by 3 epsilon by n second, where epsilon is the maximum permissible drift between two clocks that is a C minus epsilon mod, sorry C i minus C j mode is greater than epsilon. So, that is same thing. So, epsilon is given to be the maximum permissible drift between two clocks, and n is the number of clocks in the system and what this theorem claims is that the effect of a single byzantine clock would be to make two arbitrary clocks at most differ by 3 epsilon by n. I think this was the question that I was asking you, that, if we have one byzantine clock, what will be its effect on other clocks? Because it will confuse the other clocks, is it not? It sent a high value to something and a low value to something, but and since the bad values it cannot send really bad values it will be sending within the epsilon limit. So, one it will be sending plus epsilon and another minus epsilon right and a effect of that is to make two clocks, differ by 3 epsilon by n. Let us see, how we get that not very hard?

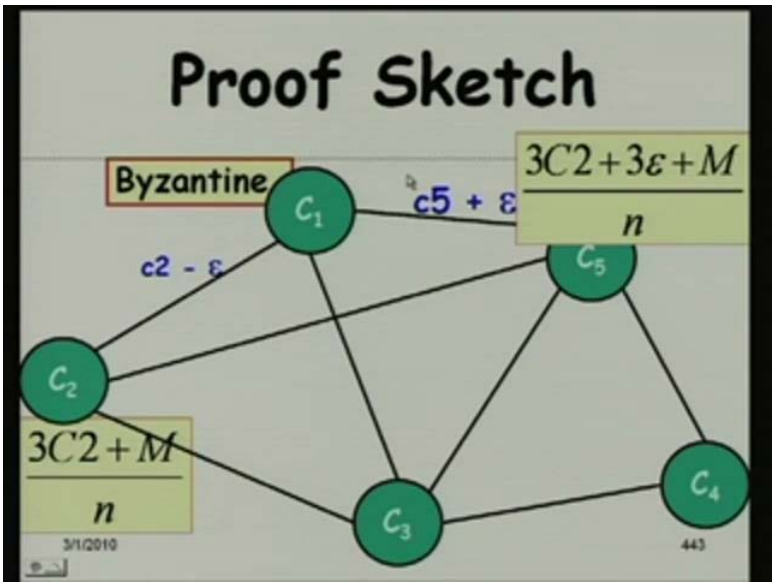
(Refer Slide Time: 41:04)



See here, we have a bad clock here C 1, let us assume three clocks. So, this is the sketch of the proof, will consider for three, you can generalize to n very easily. So, the sketch of the proof is like this. We assume that one is bad; of course, in this situation you know one is bad out of three, we cannot even make them synchronized within any given bound, any given bound not only epsilon. So, just for our own understanding, let us assume that C 1 is a bad clock, it is not a bad, it is a byzantine clock; C 1 is a byzantine clock send C 3 minus epsilon to this and send C 2 plus epsilon to this one; right and C 2 and C 3 also can differ by epsilon. See there. So, C 2, C 3 are the good clocks; C 1 is the byzantine clock, and C 1 transmits C 2 plus epsilon to C 2 and C 3 minus epsilon to C 3.

Now, due to byzantine clock C 2 and C 3 differ by 3 epsilon, do you agree with this? Because they themselves differ by epsilon at the most, if at you can assume some time t here and t plus epsilon here or you can assume t here that may be easier, it does not matter; so, t here, t plus epsilon here. So, this becomes the time value computed here will be t plus t plus epsilon 2 t plus epsilon, is it not? So, we will just see the calculation in more detail; so, it is not hard to see that they differ the two clocks will differ by, when they compute the total time right, this clocks computes the total time from all clocks from C 2 and C 3 and this clock computes the total from C 2 and C 1 then the total will be differing by 3 epsilon and then after averaging they will differ by three epsilon by n, that is the sketch of the proof.

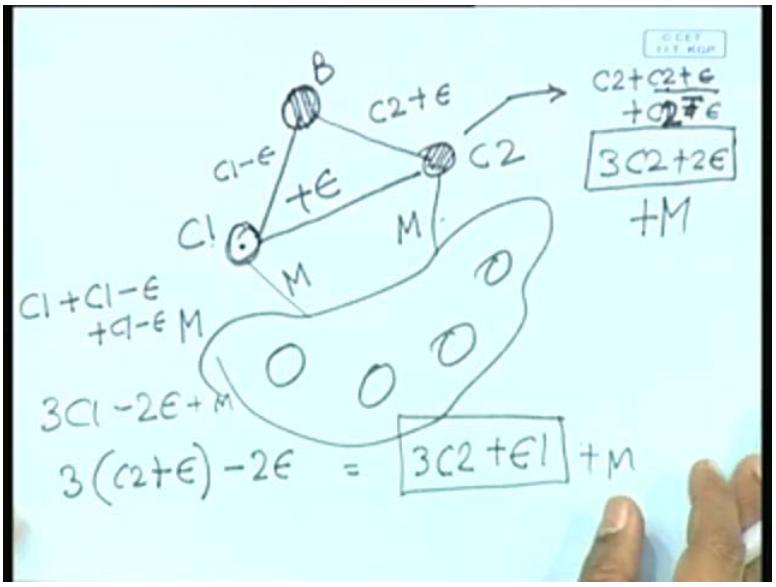
(Refer Slide Time: 43:45)



So, let us see the calculations here. So, see here C 1 is the byzantine clock; I should have used a different color here that would be a easier anyway. So, this transmits C 2 minus epsilon to C 2 and C5 plus epsilon to C 5 and this will compute, the total let us say due to all other clocks C 3, C 4 etcetera. This is m; some part has got missed here. So, this will be 3 C 2. So, this is C 2 and this is they in turn differ by epsilon. Let us see, this one this as part is caught here. So, let us see the value that is computed by here; let us assume m is the time value received from all other clocks, sum of the time values from all other clocks.

Now, let us see the impact of these three clocks that we have shown here. So, the value computed by C 5 will be 3, C 2 plus 3 epsilon by n, and this will be 3 C 2 plus m by n. Let us assume that this is C 2 and this will be at at most C 2 plus epsilon, is it not? And this is sorry and this is C 1 and this is C 2. So, this has got C 2 minus epsilon. So, it will become 3 C 2 plus m by n and this will become 3C 2 plus epsilon plus m by n is that ok? Because it has got the time value transmitted from here is C 2 and this has got time value transmitted from C 5 which is C 2 minus epsilon. So, this will become should we calculate out the details of the how the clocks send their time and they are averaged. So, let me just calculate here; see there are many clocks here.

(Refer Slide Time: 46:17)



So, let us assume that these are the three clocks we are worried about, and we can extend that the that argument to any other clocks any other three clocks. So, this is the one, which is byzantine and let us assume this is C 1 and this is C 2; and C 1 and C 2 differ from each other by epsilon. So, the byzantine clock B will transmit C 1 minus epsilon to C 1, let us assume; and it transmits C 2 plus epsilon to C 2, let us assume is the worst case. So, the sum calculated by this one this clock C 2 will be C 2 plus C 2 plus epsilon minus sorry plus C 1, let us say plus epsilon, it sends a plus epsilon here, they differ by plus epsilon so, no this is now this is transmitted its own time. So, that is ok. C 2 plus epsilon right right. So, this will be 3C 2 plus 2 epsilon right in this situation; now what about this one? This will be its own time is C 1 plus it has got C 1 minus epsilon here, C 1 minus epsilon, and what about the time it has received from here and of course, this will get M from all others let us say it gets m, the same M is got from all others that you can ignore, but what about this one here.

(Audio not clear from 48:19 to 48:23)

Sorry, this is wrong, is it? Yes sir. Why? C 2 is obtained here C 2 plus epsilon. So, C2; C 2 plus the second one C 2 plus epsilon

And this has got C 2 plus epsilon, that is all right; so, 3C 2 plus 2 epsilon.

Actually, what we have to add is C 2 plus C 1 plus C 2 plus epsilon; we are taking C 2 two times.

No, No three clocks there are C 2, 3C 2 plus 2 epsilon. Now, let us see here. So, this one we have C 1 is its own time and it has got C 1 minus epsilon here and it has got C 1 minus epsilon here right. So, this is 3C 1 minus 2 epsilon; but we know that C 1 is C 2 minus epsilon, is it not? it. So, this becomes 3 C 2 minus epsilon, C 1 is C 2 plus epsilon right minus 2 epsilon C 1 is C 2 plus epsilon. So, this becomes 3C 2 plus epsilon plus m of course, and this becomes 3 C 2 plus 2 epsilon plus m. So, that differ by three epsilon just do the calculation.

Only one epsilon, we are getting only one, but now we are getting only one.

One epsilon.

That in the first calculation; if C 1 plus C 2 plus that given by byzantine.

Let us see, C 1 is C 2 plus epsilon and…

What we assume? C 1 we are taking C 1 as reference. C 2 is equal to C 1 plus epsilon; we assume that, C 2 is this.

No, no see the byzantine clock is sending here C 1 minus epsilon, if it is any less than that this will reject it. This will send C 2 plus epsilon if it is any more than that it will reject it.

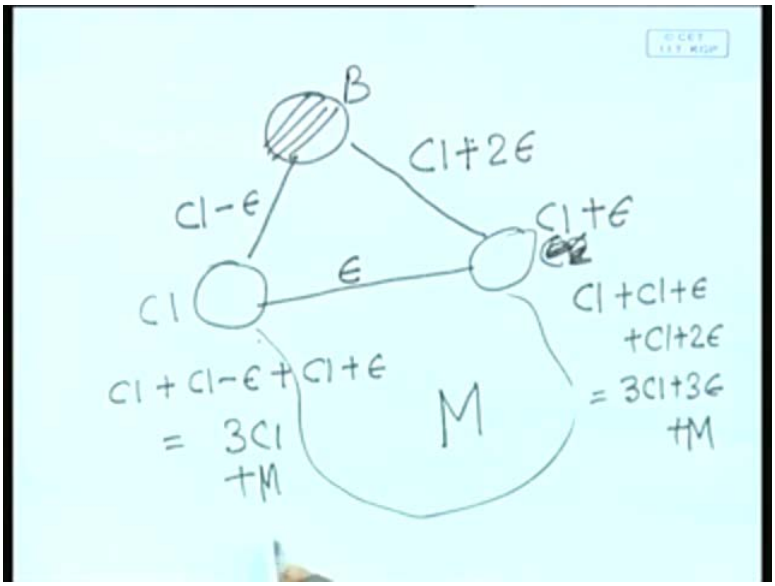
Actually C 2 is ahead of C 1. C 2 is ahead of C 1, so C 2 is equal to C 1 plus epsilon. C 2 is C 1 plus epsilon, yes. So, this would be minus right.

Here, we will get again the difference only in that calculation we can directly get 3 epsilon. I am sorry.

Instead of replacing the C 2, C 1 is the the C 2 is instead of C 1 why do not we just as it is, and then

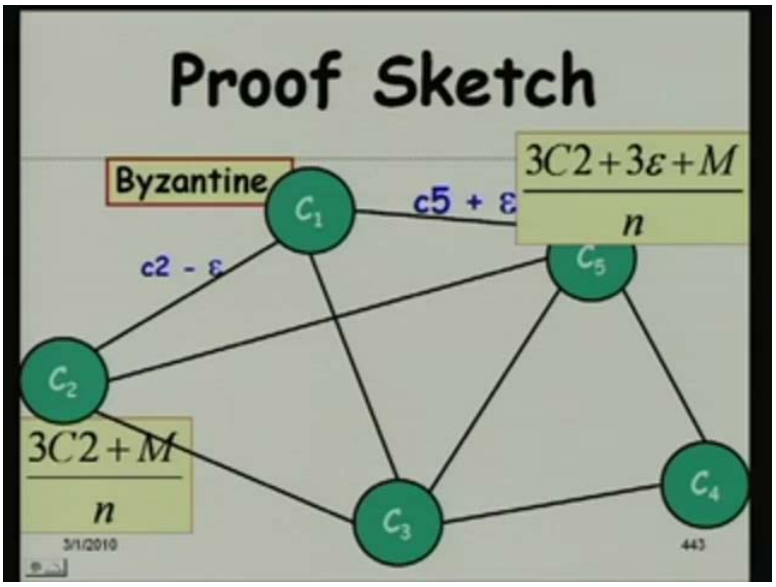
Use just one clock value is it? Let us let us do that. I think, it is become unnecessarily complicated.

(Refer Slide Time: 51:31)



So, let us assume one is the byzantine clock, and we have one clock C 1 and another clock C 2, and this sends C 2 we will use it as a C 1 plus epsilon C 1 that we will simplify as he says. So, we will use this clocks value as C 1 plus epsilon and the byzantine clock will send here C 1 minus epsilon, and to this it will send C 1 plus epsilon sorry C 1 plus 2 epsilon, C 1 plus 2 epsilon. So, the time value that this gets this clock is C 1 plus C 1 minus epsilon plus C 1 plus epsilon. So, this becomes 3C 1, and we have other clocks of course, which have a constant time value that is transmitted plus m and this one gets C 1 from here plus C 1 plus epsilon; its own time plus C 1 plus 2 epsilon. So, this is 3C 1 plus 3 epsilon Plus three epsilon plus m. So, this simplifies just considering one clock rather than no using 2 C 1, and 2 values C 1, C 2. So, that was leading to unnecessarily complications. So, easy to see here that the effect of the byzantine clock is to make two clocks, even when we have n clocks in the system to differ by 3 epsilon and the average of that after it is computed it will become 3 epsilon by n.

(Refer Slide Time: 53:36)



So, we will just stop here, and we will continue with discussing about how to compute the frequency with which we need to transmit clock values. So that, even in the presence of byzantine clocks, when one-third of the clocks, less than one-third or byzantine we can still keep the clocks synchronized within a given bound under drift. So, what do we think? Will the bound of the drifts? Let us say we want to keep them within epsilon; So, should they transmit more frequently or should they transmit less frequently in the presence of byzantine clocks? Let us assume that the two situation; one is there are no byzantine clock; another is there are byzantine clocks. So, should they transmit more frequently or less frequently? More frequently, because you know, we have to we want to let us say keep them synchronized within one millisecond. Let us, say the effect of one-third byzantine clock is to make them drift by 0.5 millisecond. So, by the time the drift becomes 0.5, we needs to we need to have a broadcast. So, we will stop here and will continue our discussion in the next hour. Thank you.