

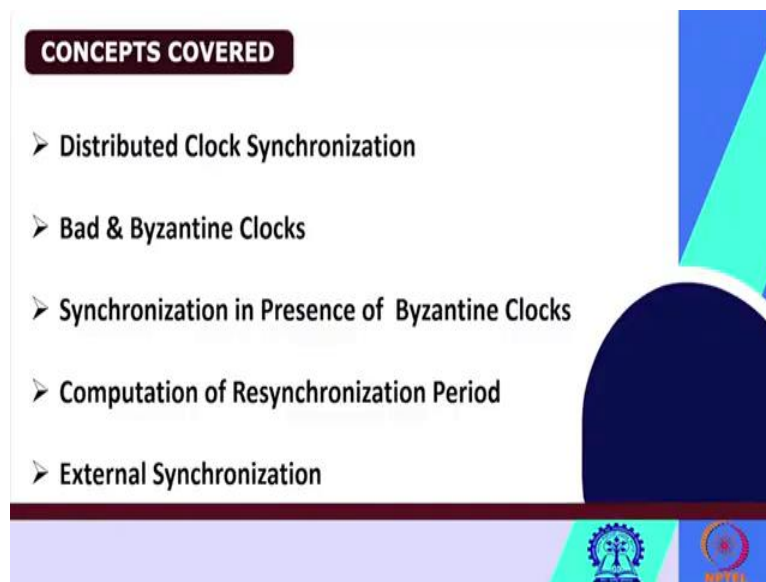
Real Time Systems
Professor Durga Prasad Mohapatra
Department of Computer Science and Engineering
National Institute of Technology Rourkela

Lecture 36

Distributed Clock Synchronization in RT Systems

Good morning to all of you, last class we discuss about Centralized Clock Synchronization in real time systems. So, today we will discuss about the Distributed Clock Synchronization in real time systems.

(Refer Slide Time: 0:00:32)



So, first we will see about how the distributed clock synchronization can be achieved. We will see, what do you mean by bad clock and what do you mean by byzantine clock. How synchronization can be made in the presence of the byzantine clocks. How the computation of resynchronization period can be done in distributed clock synchronization. Finally, we will see the details or maybe briefly about the external synchronization, external clock synchronization.

(Refer Slide Time: 0:01:05)

KEYWORDS

- Bad Clocks
- Byzantine Clocks
- Drift
- Resynchronization Period
- NTP (Network Time Protocol)

So, in this lecture also you have to know, what do you mean by a bad clock, what do you mean by a byzantine clock, what do you mean by drift or skew, resynchronization period and we will see what is NTP, Network Time Protocol.

(Refer Slide Time: 0:01:19)

DISTRIBUTED CLOCK SYNCHRONIZATION

- **Problem with centralized clock synchronization:**
 - **single point failure.**
- **Solution: Distributed Clock Synchronization.**

So, let us see why distributed clock synchronization is required. So, yesterday or last class we have discussed the details of centralized clock synchronization. I have told the drawbacks of centralized clock synchronization. So, the major bottleneck of centralized clock synchronization is single point failure, that means if this master clock fails then what will happen?

We cannot achieve synchronization, so it is a single point failure. So, if the master clock or the time server it fails then we cannot synchronize the clocks, that is the bottleneck of centralized clock synchronization. So, in order to overcome this problem of this single point failure, the solution is we should go for distributed clock synchronization. So, now let us see the details of this distributed clock synchronization.

(Refer Slide Time: 0:02:14)

DISTRIBUTED CLOCK SYNCHRONIZATION

- No master clock.
- All clocks periodically exchange their clock readings.
 - Each clock computes the average time and sets its clock accordingly.

What if some clocks are bad?

The slide features a diagram of four green circular nodes labeled C_1, C_2, C_3, C_4 connected by blue lines in a network topology. A yellow box with a red border contains the text 'What if some clocks are bad?'. A small video inset shows a man speaking. The slide has a blue and green background with a dark blue header and footer containing logos.

In distributed clock synchronization, there is no concept of master clock, as it happens in centralized clock synchronization. In centralized clock synchronization there is a master clock and according to the time value of the master clock all other clocks they will set their times, but in distributed clock synchronization there is no master clock used.

All the clocks in the distributed system, they periodically exchange their clock readings, their time values. Then when a clock, it receives what the time values or the clock readings of other clocks, then it computes the average time and then this average time is used as, to set the clock value at the particular clock. So, here all the clocks periodically exchange their clock readings to all the other clocks, then each clock when it receives the periodic message regarding the time values from other clocks, it computes the average time and then it sets its clock accordingly.

This is how the clocks in a distributed system, they are synchronized, but the problem, here I have shown the example of a distributed system, we are, I have taken four clocks and you see each clock tries and they are connected like this, now how does it happen? So, each clock periodically sends

their clock readings and then the other clocks on receiving those messages they find out the average time and sets their clock accordingly. But there is a problem here. What happens? If some clocks are becoming bad. So, what do you mean by bad clock here?

(Refer Slide Time: 0:04:06)

BAD CLOCKS

- Some clocks become bad and stops all of a sudden.
- **Bad clocks exhibit large drifts:**
 - Drifts larger than the manufacturer specified tolerance.
- Bad clocks can be identified and handled during synchronization
 - by rejecting the time values of any clock which differs by an amount larger than the specified bound.

DISTRIBUTED CLOCK SYNCHRONIZATION

- No master clock.
- All clocks periodically exchange their clock readings.
 - Each clock computes the average time and sets its clock accordingly.

What if some clocks are bad?

So, some clocks they become bad and if they become bad what will happen? They stop all of a sudden, so they will not work. Some clocks, they become bad and stops all of a sudden, and what other things they can do? These bad clocks, they exhibit larger drifts. Last class I was saying this drift or the skew normally it is specified by this manufacturer, but these bad clocks they will exhibit

larger drifts or we can say that which drifts these are larger than the manufacturer specified tolerance.

So, what is this specified value given by the manufacturer? It exceeds those values. So, the bad clocks exhibit very large drifts, that means drifts larger than the value or the manufacturer specified tolerance. So, then how to handle these bad clocks? These bad clocks can be handled, they can be easily identified, the bad clocks can be easily identified and they can be handled during the clock synchronization process, how? So, what we have to do, as I have already told you. That each clock, all the clocks periodically send their clock ratings.

So, when you will examine these clock readings or the time values and you see that any clock, the time value of any clock differs by a very large amount, then the specified bound given by the manufacturer, then what you will do, you will simply reject those values. I am repeating again. We can easily identify and handle the bad clocks during synchronization, how? We have to examine the clock values, those are received from the other clocks.

Then if you will see, if any clock value, it differs by an amount larger than the specified bound, maybe specified by the manufacturer, then we will simply reject those values, because that we will expect that, that clock value has come from a bad clock. So, that is why we have to reject those values. So, in this way, bad clocks can be identified and they can be handled during clock synchronization.

(Refer Slide Time: 0:06:21)

BYZANTINE CLOCKS

- A Byzantine clock is a two-faced clock:
 - It can transmit different values to different clocks at the same time.

Byzantine C_1 $t + e$ C_5
 $t - e$ C_2 C_3 C_4

So, anyway we can handle these bad clocks but there is another type of clock who also pose difficulty in this synchronization. These clocks are known as byzantine clocks. It is very important, what do you mean by byzantine clock? A byzantine clock is a clock, which is having two faces. What do you mean by that? So, byzantine clock is two faced clock, it can transmit different values to different clocks at the same time, at a particular point of time it may send some value sometime to one clock, whereas some other value to another clock. It can transmit different values, different time values to different clocks at the same time.

Let us take a small example. So, here I have taken an example of a distributed system, five clock I have taken. So, C 1 is a clock which transmits at a particular time, the time t minus e to clock C 2, but sends the time t plus e to C 5. So, what does it mean? At the same time the clock C 1, it is sending two different values. One is t minus e to C 2, another is t plus e to C 5, which is not correct.

So, it is a two-faced clock, because it is sending two different values of the time to two different clocks. So, we can say that C 1 is an example of byzantine clock. So, these byzantine clocks they also put problem while doing this clock synchronization. Let us see how we can make, how we can achieve distributed clock synchronization in the presence of byzantine clocks.

(Refer Slide Time: 0:07:59)

SYNC IN PRESENCE OF BYZANTINE CLOCKS

- Is it possible to have good clocks approximately synchronized?
 - Yes, if no more than one-third of the clocks are bad or byzantine.
- Assume:
 - There are n clocks in the system.
 - Each clock periodically broadcasts its time value at the end of certain interval.
 - The clocks are to be synchronized within ϵ time units of each other.

The slide features a video inset of a man in a light-colored shirt speaking. At the bottom, there are logos for IIT Bombay and NPTEL.

So, let us see clock synchronization in presence of byzantine clocks. Now, the question is, is it possible to have good clocks approximately synchronized? Is it possible? Because in a distributed system there are some clocks that are good, some clocks are bad, some clocks are byzantine. Now, I have already told you how to handle bad clocks during clock synchronization. Now, the question is, is it possible to have the good clocks at least approximately synchronized, at least can we synchronize the good clocks in the presence of byzantine clocks, the answer is yes, we can synchronize the good clocks. How?

If no more than one third of the clocks are bad or byzantine, so if in a system how many total number of clocks are there? If no more than one third of the clocks are bad or byzantine, then we can possibly synchronize the good clocks. So, let me just express in other way. If there are n number of total clocks and out of that m number of clocks are bad or byzantine then what we can say that in order to have good clocks approximately synchronized no more than one third of the total clocks are bad or byzantine.

So, if the total clocks are n it must be greater than $3m$ where m is the total number of bad or byzantine clocks. So, it is a straightforward formula. If we want to synchronize the good clocks, the condition is that the total number of clocks must be greater than $3m$, so what should be the minimum value, n must be equal to at least $3m$ plus 1, at least 1 is extra.

So, n must be $3m$, greater than $3m$, that means the total number of clocks n , must be or at least equal to $3m + 1$, at least 1 should be extra. So, if you will see, if you will take the total number of clocks is four then what is the equation then? $3m + 1$, that means m is here only 1, so that means if four number of clocks are there, we can synchronize the good clocks if only one clock is bad or byzantine.

If more than one clock is bad or byzantine we cannot synchronize the good clocks. So, in order to synchronize the good clocks in a system where n number of clocks are there, at best one clock may be bad or byzantine, then we can approximately set the good clocks, otherwise we cannot synchronize the good clocks.

Now, let us find out a formula for the resynchronization period. Last class we have already found out the formula for the resynchronization period in case of centralized clock synchronization, today, right now, we will find out another formula which will be used to calculate the resynchronization period in case of distributed synchronization.

So, let us assume that there are total n number of clocks present in the system. Each clock periodically broadcast its time, value at the end of certain interval. Each clock periodically broadcast its time value at the end of certain interval, this is what we can say that resynchronization period.

The clocks, they have to be synchronized within say epsilon time units of each other. So, epsilon already we have seen in last class. It is the drift. So, the clocks have to be synchronized within epsilon time units of each other. With this now let us try to find out the formula for the resynchronization period in distributed clock synchronization.

(Refer Slide Time: 0:12:06)

SYNC IN PRESENCE OF BYZANTINE CLOCKS

- When a clock receives a time broadcast:
 - Checks against its own time.
 - If it differs by more than ϵ time units,
 - Received time value is ignored.
- After each step, each clock averages all received good time values:
 - Uses this value as its time.

Now, let us see first, how synchronization can be made in the presence of byzantine clocks, if total n number of clocks are there out of that m number of clocks are bad or byzantine, then how to achieve synchronization. So, let us see first how synchronization can be made in presence of byzantine clocks. So, when a clock receives a time broadcast, I have already told you in distributed system, every clock periodically broadcast their time value. So, when a clock receives a time broadcast then what does it do? It checks against its own time. So, what is its own time right now?

So, when a clock receives a time broadcast, it checks against its own time, what is the right time in itself. It checks against its own time. Now, if its own time, it differs by more than epsilon time units, then definitely it has come from a bad clock and when it receives, the received time value is ignored. I am repeating again so when a clock receives a time broadcast, a time value, from another clock then it checks against its own, then it checks that time against its own time.

Then it calculates the difference, that means the difference between its own time and the received time. If the received time differs by more than some limit, a threshold, epsilon time units, where we know epsilon means the drift, so if the received time differs from its own time by more than epsilon time units, then the received time value is ignored, because it is assumed that that time has come from a bad clock.

And I have already told you, how to handle bad clocks, if it is greater than the specified limit by the manufacturer, that is epsilon time units, then we will assume that this time has come from a

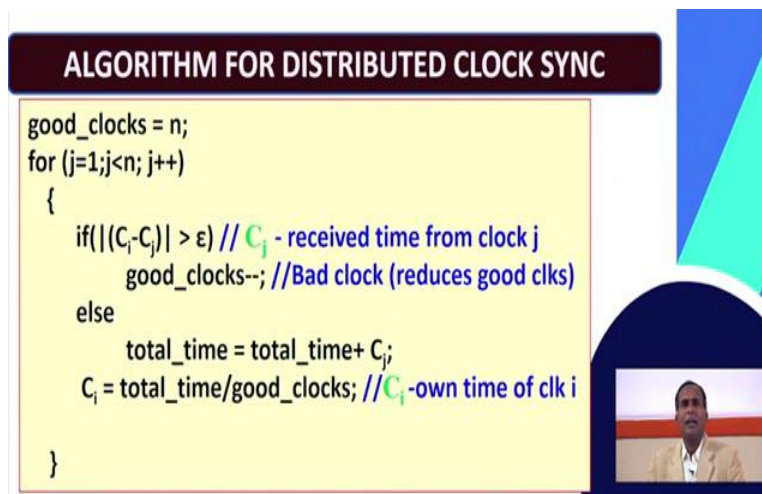
bad clock and hence we have to ignore this received time value. Else that time value is accepted. And how to synchronize then?

Let us see, so there are two possibilities I have already told you, either it may come from a bad clock or it is a good one, we have to see. So, now if the received time is greater than its own time, if it differs by more than epsilon time units, then the received time value is ignored. Now, after each step, otherwise if it is, if the difference is less than epsilon, then we have to consider these things.

So, what we will do in the next step? So, at each step each clocks, so that means if the difference is not more than epsilon time units, that means it is less than epsilon time units, then what will be done? Each clock will take the average of all the received good time values. That means it is definitely a good value, it is not what time value from a bad clock, it is time from a good clock.

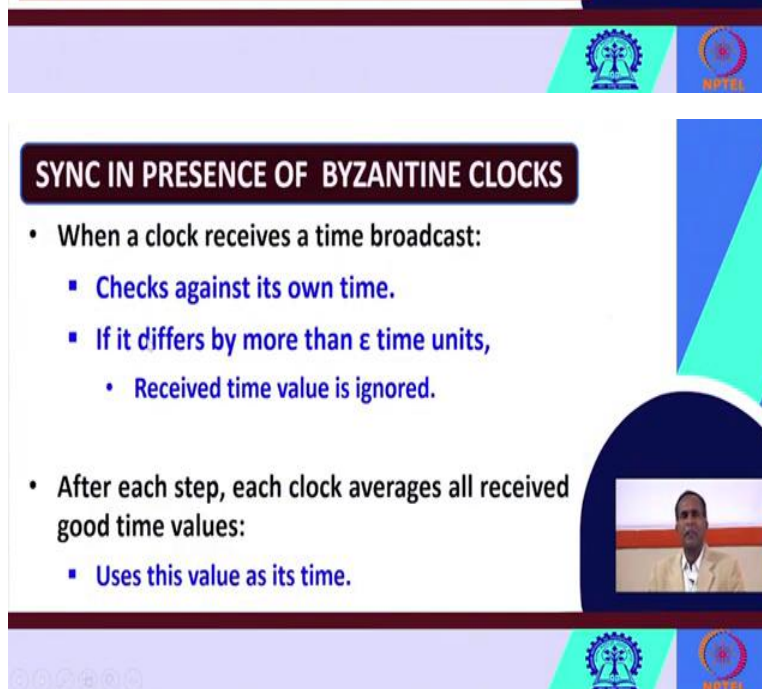
So, each clock will average, each clock averages all the received good time values. Then what is the average value? That will be used as this current time. It will use this, the clock will use this time value as its own time. It will now revise. So, now the clock will use this average value as its time. So, what are the previous values? Now, it will update it, it will set with this new average value, this is how synchronization can be made in presence of byzantine clocks.

(Refer Slide Time: 0:15:37)



```
ALGORITHM FOR DISTRIBUTED CLOCK SYNC

good_clocks = n;
for (j=1;j<n; j++)
{
  if (|(Ci-Cj)| > ε) // Cj - received time from clock j
    good_clocks--; //Bad clock (reduces good clks)
  else
    total_time = total_time+ Cj;
    Ci = total_time/good_clocks; //Ci-own time of clk i
}
```



- When a clock receives a time broadcast:
 - Checks against its own time.
 - If it differs by more than ϵ time units,
 - Received time value is ignored.
- After each step, each clock averages all received good time values:
 - Uses this value as its time.

So, what I have told, I have given here simple, I have given it in this form of a simple algorithm, you can implement yourself in any language. So, what I have done here, I have first assumed good clocks is equal to n , that means total number of good clocks is equal to n . Then for some j is equal to 1, j less than equal to n , j plus, plus. Where n is equal to total good clocks.

Now, what I have already told you, first to find out the difference. That means difference between the received value and the own value, if it is more than epsilon time units then you ignore it. that means bad clock will be, number of good clock will be reduced by 1 and number of bad class will be increased by 1.

So, if c_i minus c_j , where c_i is its own time c_j is the time received from clock j . So, if this value is greater than ϵ , then obviously this has come from a bad clock, hence you reduce good clock by 1. So, good clocks will be equal to good clocks minus 1. Else this is a good time value. So, now total time, it was there somewhere, you can initialize total time is equal to 0 or so. So, now total time is equal to the previous value of total time, plus this new value of c_j , because this is a good time value. Then you can find out the average, you can find out the revised value, so the c_i that means the own time of clock i , now will be set to the average value.

That means total time divided by the number of good clocks. This will give you the average value, now this average value will be set as the time of c_i . So, c_i is the own time of clock i . So, in this way you can write a simple algorithm for synchronization in distributed systems and this is an easy one. This algorithm you may implement yourself.

(Refer Slide Time: 0:17:19)

SYNC IN PRESENCE OF BYZANTINE CLOCKS

Theorem: In a distributed system with n clocks:

- A single Byzantine clock can make any two arbitrary clocks to differ by $3\epsilon/n$ sec.
- Here, ϵ is the maximum permissible drift between two clocks.
- m Byzantine clocks, can make 2 good clocks differ by at most $3\epsilon m$, in average computation.
- Hence, from the time computation by the individual clocks, it follows that the individual clocks will be synchronized within $3\epsilon m/n$.
- Let the time required for 2 clocks to drift from $3\epsilon m/n$ to ϵ be ΔT .

Now, there is an important theorem, we will see. Based on that important theorem as I was saying, I will find out the formula for the resynchronization period in distributed systems. What does this theorem says? This theorem says that, in a distributed system with n number of clocks a single byzantine clock can make any two arbitrary clocks to differ by some units, what is that unit 3ϵ by n second.

This is a very standard theorem. The proof is given in professor Rajib Mall's book, those who are interested they may go, I will not cover the proof here. You can go yourself. So, this theorem says

that in a distributed system with n number of clocks, a single byzantine clock, it can make any two arbitrary clocks to differ by how much unit? By 3ϵ by n second, where ϵ is the drift, is not it?

Where ϵ is the maximum permissible drift between two clocks. Very standard theorem, the proof is very easy given in professor Rajib Mall's book that you can see yourself. So, now what I have said, this is with a single clock. Now, I have to find out the equation if there are m number of byzantine clocks are there, then how the clocks, they differ, how the time values in any two clocks may differ?

So, here first let us see, m byzantine clocks can make two good clocks differ by at most $3\epsilon m$ in average computation where m is the number of byzantine clocks, hence from the above time computation, by the individual clocks it may follow that the individual clocks will be synchronized within $3\epsilon m$ by n , where m is the number of byzantine clocks. The individual clocks will be synchronized within what time? Within $3\epsilon m$ by n times, please see the, please relate, a single byzantine clock can make any two arbitrary clocks to differ by how much time? By 3ϵ divided by n second. So, this is a single byzantine clock.

If there will be m byzantine clock I am multiplying here m . So, 3ϵ by n , here I am multiplying m $3\epsilon m$ divided by n , hence, from the time computation, from the above time computation by the individual clocks, it follow that the individual clocks will be synchronized within what time? Within $3\epsilon m$ into n divided by n seconds.

Now, what is our requirement, our requirement to find out what should be the resynchronization period ΔT , as we have already calculated in last class for centralized synchronization, similarly we will find out the synchronization period, resynchronization period for this distributed clock synchronization.

So, let us assume that the time required for two clocks to drift form $3\epsilon m$ by n to ϵ , because I have already told you ϵ is the maximum permissible drift. Let the time required for two clocks to differ, from what value to what value, form $3\epsilon m$ by n to the maximum permissible drift that is ϵ by ΔT . So, what is happening during the time ΔT ?

(Refer Slide Time: 0:20:29)

COMPUTATION OF RESYNCHRONIZATION PERIOD

- During ΔT :
 - The time difference increases from $\frac{3\epsilon m}{n}$ to ϵ .
 - This must also be equal $2\rho\Delta T$.

- Or, $2\Delta T\rho = \frac{n\epsilon - 3\epsilon m}{n}$

- Or, $\Delta T = \frac{n\epsilon - 3\epsilon m}{2n\rho}$



SYNC IN PRESENCE OF BYZANTINE CLOCKS

Theorem: In a distributed system with n clocks:

- A single Byzantine clock can make any two arbitrary clocks to differ by $3\epsilon/n$ sec.
- Here, ϵ is the maximum permissible drift between two clocks.
- m Byzantine clocks, can make 2 good clocks differ by at most $3\epsilon m$, in average computation.
- Hence, from the time computation by the individual clocks, it follows that the individual clocks will be synchronized within $3\epsilon m/n$.
- Let the time required for 2 clocks to drift from $3\epsilon m/n$ to ϵ be ΔT .



SYNC IN PRESENCE OF BYZANTINE CLOCKS

- Is it possible to have good clocks approximately synchronized?
 - Yes, if no more than one-third of the clocks are bad or byzantine.
- Assume:
 - There are n clocks in the system.
 - Each clock periodically broadcasts its time value at the end of certain interval.
 - The clocks are to be synchronized within ϵ time units of each other.



During time ΔT , the time difference, it increases. From what value to what value, we have shown here, from $3\epsilon m/n$ to ϵ . Now, we know what is the, last class I have already told you, you can go those slides for centralized synchronization. I have already told you in centralized synchronization the maximum drift between any two clocks, that means what is the worst case, one clock is going in the negative direction, one clock is going in the positive direction.

So, I have told in last class, the difference will be what? $\rho \Delta T$ minus, again within minus bracket minus ΔT , so minus, minus will become plus, so it will become $\rho \Delta T$ plus $\rho \Delta T$ is equal to $2\rho \Delta T$. So, the same equation is also valid here. So, during ΔT the drift between two clocks, it must also be equal to $2\rho \Delta T$.

So, now I have already told you during ΔT , the time difference increased from this to this. $3\epsilon m/n$ to ϵ , so what is difference? It is $\epsilon - 3\epsilon m/n$. So, I have already, also told you, last class that the drift between two clocks, the maximum drift can be equal to $2\rho \Delta T$, so $2\rho \Delta T$ will be equal to how much?

Just find out this subtraction, $\epsilon - 3\epsilon m/n$, so you just solve it, from the minus operation, so you will get how much? Any $\epsilon - 3\epsilon m/n$. So, $2\rho \Delta T$ will be equal to how much? $\epsilon - 3\epsilon m/n$, you just simplify, you will get ϵ , minus $3\epsilon m/n$.

So, this will be equal to then, I want to find out what? delta T, because I am interested in find out the resynchronization period. So, delta T will be equal to how much? Simply 2ρ will come to this side, so $n\epsilon - 3\epsilon m$ by $2n\rho$, in this way we can find out the value of the resynchronization period.

But what is the value of n? I have already told you in terms of m, I have given this formula earlier that we can synchronize the good clocks if no more than one third of the clocks are bad or byzantine. That means n can be, n should be greater than $3m$ or I can say that n is equal to $3m$ plus 1. Where n is the total clocks m is the number of bad or byzantine clocks. So, this value of n I can now put here. So, in place of n how much I can put? $3m$ plus 1. So, let us put it.

(Refer Slide Time: 0:23:11)

The slide is titled "COMPUTATION OF RESYNC PERIOD cont ...". It contains the following text and formulas:

- We know that $n = 3m + 1$.
- Therefore, $\Delta T = \frac{(3m + 1)\epsilon - 3\epsilon m}{2n\rho}$
- Or, $\Delta T = \frac{\epsilon}{2n\rho}$

The slide also features a small video inset of a man in a suit, and logos for IIT Bombay and NPTEL at the bottom.

We know that n is equal to $3m$ plus 1 I have already told you. Therefore delta will be equal to how much?

(Refer Slide Time: 0:23:17)

COMPUTATION OF RESYNCHRONIZATION PERIOD

- During ΔT :
 - The time difference increases from $\frac{3\epsilon m}{n}$ to ϵ .
 - This must also be equal $2\rho\Delta T$.
- Or, $2\Delta T\rho = \frac{n\epsilon - 3\epsilon m}{n}$
- Or, $\Delta T = \frac{n\epsilon - 3\epsilon m}{2n\rho}$

The graph shows a series of green triangles on a blue horizontal axis. The vertical axis is labeled ϵ . The horizontal axis has a tick mark labeled $\frac{3\epsilon m}{n}$. The triangles represent the increasing time difference over time.

COMPUTATION OF RESYNC PERIOD cont ...

- We know that $n = 3m + 1$.
- Therefore, $\Delta T = \frac{(3m + 1)\epsilon - 3\epsilon m}{2n\rho}$
- Or, $\Delta T = \frac{\epsilon}{2n\rho}$

In this equation I am putting the value of n is equal to 3m plus 1. So, 3m plus 1 epsilon, minus 3 epsilon m, divided by 2 n rho. So, then you simplify on simplification you will get delta T is equal to epsilon by 2 rho n. So, delta T is equal to epsilon by 2 n rho. So, the resynchronization period in case of distributed system is or can be expressed as delta T is equal to epsilon by 2 n rho, is not it?

And I think you can relate to it with what we have seen in the centralized synchronization. In centralized synchronization this n value was not there, it was simply epsilon by 2 rho, but here delta T is equal to epsilon by 2 into n rho, because this is a distributed system. We are taking into

account the total number of clocks and n is the total number of clocks, that why n is coming in the denominator. So, ΔT is equal to $\epsilon / (2n\rho)$. Where ϵ is the maximum drift, n is this number of the total clocks and ρ also we have already told you in the last class. So, in this way you can find out ΔT is equal to $\epsilon / (2n\rho)$.

(Refer Slide Time: 0:24:26)

EXAMPLE - 1

- A distributed real-time system has 10 clocks:
 - Upto 3 clocks can be Byzantine.
 - Maximum drift to be less than $\epsilon = 1\text{ms}$.
 - Let the maximum drift rate between two clocks be $\rho = 5 \times 10^{-6}$.
- Determine the resynchronization interval.

Now, let us take a small example in order to illustrate this. So, suppose a distributed real time system has 10 number of clocks, and in these 10 number of clocks up to 3 clocks can be byzantine. You have already known n is equal to how much? $3m + 1$, so in 10 clocks, maximum 3 clocks can be byzantine.

Put the formula n is equal to $3m + 1$, so up to three clocks can be byzantine. The maximum drift to be less that ϵ unit that is 1 millisecond and the maximum drift rate between two clocks is known as ρ . That you have already known in the last class. So, let the maximum drift rate between two clocks be ρ is equal to 5×10^{-6} . Normally, ρ is specified by the clock manufacturer. What is the question? Question is determined the resynchronization interval, that means find out the value of ΔT . so let us find out.

(Refer Slide Time: 0:25:19)

ANSWER

- $\Delta T = \frac{\epsilon}{2n\rho}$
- $\Delta T = \frac{10^{-3}}{2 \times 10 \times 5 \times 10^{-6}}$
- $\Delta T = 10\text{sec}$



COMPUTATION OF RESYNC PERIOD cont ...

- We know that $n = 3m + 1$.
- Therefore, $\Delta T = \frac{(3m + 1)\epsilon - 3\epsilon m}{2n\rho}$
- Or, $\Delta T = \frac{\epsilon}{2n\rho}$



EXAMPLE - 1

- A distributed real-time system has 10 clocks:
 - Upto 3 clocks can be Byzantine.
 - Maximum drift to be less than $\epsilon = 1\text{ms}$.
 - Let the maximum drift rate between two clocks be $\rho = 5 \times 10^{-6}$.
- Determine the resynchronization interval.



We know the formula for delta T we have already derived, delta T is equal to how much? Epsilon by 2 n rho. So, delta T is equal to epsilon by 2 n rho. The value of epsilon is already given, how much? 1 millisecond, that means 10 to the power minus 3 seconds and 2n is how much? 10, because 10 clocks are there, what is the value of rho? The value of rho is given to be 5 into 10 the power minus 6. So, now you see that on simplification you will see that delta T is equal to 10 second. So, this is how you can find out the value of the resynchronization period. Let us quickly take another example.

(Refer Slide Time: 0:25:53)

EXAMPLE - 2

- **Suppose:**
 - A distributed system has 12 clocks.
 - The clocks are required to be synchronized within 1 msec of each other.
- **Assume:**
 - At best 2 clocks in the system are Byzantine.
 - Maximum drift rate of the clocks is 6×10^{-6} .
- **Determine:**
 - The rate at which the clocks need to exchange time values.
 - The total number of message exchanges required per hour for synchronization.



So, suppose you are having a distributed system, with 12 clocks, suppose distributed system has 12 clocks, the clocks are required to be synchronized within 1 millisecond of each other. So, epsilon is also given, the clocks are required to be synchronized within 1 millisecond of each other. That is epsilon is equal to 1 millisecond. We will assume that at best two clocks in the system are byzantine. Let us assume that at best two clocks in the system are byzantine.

The maximum drift weight that is rho of the clocks is given as 6×10^{-6} . so epsilon is given 1, n is given 12 and your rho is given as 6×10^{-6} , m is given as 2. what we have to determine? We have to determine the rate at which the clocks need to extend the time values. Find out the rates at which the clocks need to exchange the time values.

That means we have to find out what? The value of resynchronization period. The next question is find out the total number of message exchanges or the message overhead, required for our synchronization. So, let us do it quickly.

(Refer Slide Time: 0:27:07)

ANSWER

- Time difference due to 2 Byzantine clocks = $\frac{3 \cdot \epsilon \cdot 2}{n}$
- Resynchronization needed when clocks diverge by $2\Delta T\rho = \frac{n\epsilon - 6\epsilon}{n}$
- Or, $2\Delta T\rho = \frac{12\epsilon - 6\epsilon}{12} = \frac{\epsilon}{2}$
- $\Delta T = \frac{10^{-3}}{2 \times 2 \times 6 \times 10^{-6}} = 41.67 \text{ sec}$

The slide also features a video inset of a man in a suit speaking, and logos for IIT Bombay and NPTEL at the bottom.

COMPUTATION OF RESYNC PERIOD cont ...

- We know that $n = 3m + 1$.

- Therefore, $\Delta T = \frac{(3m + 1)\epsilon - 3\epsilon m}{2n\rho}$

- Or, $\Delta T = \frac{\epsilon}{2n\rho}$

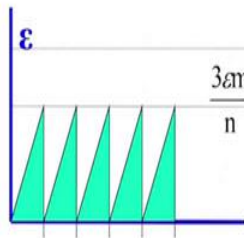


COMPUTATION OF RESYNCHRONIZATION PERIOD

- During ΔT :
 - The time difference increases from $\frac{3\epsilon m}{n}$ to ϵ .
 - This must also be equal $2\rho\Delta T$.

- Or, $2\Delta T\rho = \frac{n\epsilon - 3\epsilon m}{n}$

- Or, $\Delta T = \frac{n\epsilon - 3\epsilon m}{2n\rho}$



EXAMPLE - 2

- **Suppose:**
 - A distributed system has 12 clocks.
 - The clocks are required to be synchronized within 1 msec of each other.
- **Assume:**
 - At best 2 clocks in the system are Byzantine.
 - Maximum drift rate of the clocks is 6×10^{-6} .
- **Determine:**
 - The rate at which the clocks need to exchange time values.
 - The total number of message exchanges required per hour for synchronization.



Straightforward formula. The time difference due to two byzantine clocks, what is the formula? I have already given you, $3m\epsilon$ divided by n . what is m , m is the byzantine clocks, what is the byzantine clocks here? 2. So, it is equal to $3m\epsilon$ by n . So, the value of m I have put 2, because 2 byzantine clocks are there.

So, the time difference due to two byzantine clocks is equal 3 into ϵ into 2 by n . So, what is the resynchronization period? What is the formula? $2\rho\Delta T$ into n into ϵ minus $3m\epsilon$. This I have already told you here. This is the value for this $\rho\Delta T$, $n\epsilon$ minus $3m\epsilon$ divided by $2n\rho$.

I have to put the values of n , ϵ and ρ , so let us put the values $2\Delta T\rho$ is equal to $n\epsilon$ minus $3m\epsilon$ by n . What is the value m here? $3m\epsilon$, so $2\Delta T\rho$ is equal to $n\epsilon$, $n\epsilon$ minus $3m\epsilon$ by n . So what is the value of m here? 2. So, 3 into 2 becomes 6. So, $2\Delta T\rho$ is equal to $n\epsilon$ minus 6ϵ by n . Now, let us simplify. Or $2\Delta T\rho$ is equal to, what is the value of n ? I have already told you, 12.




12ϵ minus 6ϵ by 12, and this is coming to be how much? ϵ by 2, so $2\Delta T\rho$ is coming to be ϵ by 2, so what is the value of ΔT ? So, the value of ΔT on simplification is coming to be like this, ϵ by 2 and 2ρ will come to this side. So, the value of ϵ is how much? 1 millisecond that is 10^{-3} seconds into 2, then another 2 is here. So, 2 we have written and what left? ρ . What is the value of ρ ? Value of ρ we have seen 6 into 10^{-6} , here it is already given.

So, put the value of rho here 6 into 10 to the power minus 6. So, delta T is coming to be on simplification 41.67 seconds. So, in this way we have find out the resynchronization period.

(Refer Slide Time: 0:29:36)




ANSWER cont ...

- **Message overhead:**
 - Each node transmits $n-1$ messages per resynchronization interval.
 - Total message transmissions per resynchronization interval = $n*(n-1) = 132$.
- Number of resynchronization intervals per hour = $60*60/41.67 = 86.41$.
- Number of messages transmitted per hour = $86.41*132 = 11,406$.



EXAMPLE - 2

- **Suppose:**
 - A distributed system has 12 clocks.
 - The clocks are required to be synchronized within 1 msec of each other.
- **Assume:**
 - At best 2 clocks in the system are Byzantine.
 - Maximum drift rate of the clocks is $6*10^{-6}$.
- **Determine:**
 - The rate at which the clocks need to exchange time values.
 - The total number of message exchanges required per hour for synchronization.



ANSWER

- Time difference due to 2 Byzantine clocks = $\frac{3 \cdot \epsilon \cdot 2}{n}$
- Resynchronization needed when clocks diverge by $2\Delta T\rho = \frac{n\epsilon - 6\epsilon}{n}$
- Or, $2\Delta T\rho = \frac{12\epsilon - 6\epsilon}{12} = \frac{\epsilon}{2}$
- $\Delta T = \frac{10^{-3}}{2 \times 2 \times 6 \times 10^{-6}} = 41.67 \text{ sec}$

Now, we will quickly find out the message overhead, how much message it will require. The message overhead let us see, so similar type of problem we have done also in the last class. The message overhead can be calculated as follows. Each node transmits how many messages? Each node transmits n minus 1 messages per resynchronization period.

What is the value of n here? I have already told you there are 12 clocks. So, 12 minus 1, so 12 into 12 minus 1, so coming to be 132. The number of resynchronization intervals per hour will be how much? 16 into 16, divided by the resynchronization period, resynchronization period is 41.67 seconds.

So, the number of resynchronization intervals per hour is equal to 60 into 60 by 41.67 that is equal to 86.41. So, what is the number of messages transmitted per hour? Very simple, it is 6.41 into what is the total message transmissions per resynchronization interval it is 1 and 32. So, the total number of messages transmitted per hour is equal to 86.41, into 132, coming to be nearly equal to 11,406.

So, the total number of messages transmitted per hour is equal to 11,406. This is also known as the message overhead. So, the message overhead per hour is coming to be 11,406. So, please solve this problem again. Please you may practice some other practice problems and exercises and some examples you can do yourself. Similar types of questions will be asked in the examination.

(Refer Slide Time: 0:31:07)

EXTERNAL SYNCHRONIZATION

- Clocks must be synchronized with some real-time, such as UTC.
- Cristian's algorithm can be used if the time server is synchronized with real-time, somehow.
- But, Berkeley algorithm cannot be used. Why?
- UTC time is broadcasted from different sources around the world, for example:
 - National Institute of Standards & Tech. (NIST).
 - United States Naval Observatory (USNO).
 - GPS satellites, etc.

The slide features a dark blue header with the title in white. The main content is on a white background with a blue and green geometric design on the right. A small video inset shows a man in a suit speaking. At the bottom, there are logos for NIST and USNO.

Now, the last part is the external synchronization. I have already told you last class, there is another classification of the synchronization that is internal synchronization and external synchronization. I have told you that in internal synchronization one of the clock internal to the system or based on that timing value of the clocks must be, the time will be set according to the internal clocks, but in external synchronization the time will be set as per some clock which is external to the system.

So, in the external synchronization, the clocks must be synchronized with respect to what? With respect to some real time, and example of real time yesterday I have told, UTC, that universally coordinated time, which is based on what? TAI, is not it? UTC is based on, it is computed using TAI, international atomic time.

But how you can get these UTC values? Sitting at your place? You know, UTC time is broadcasted from different sources around the world, many organizations are there, many platforms, many places UTC time is broadcasted. Some examples I have given here. UTC timing is broadcasted from different sources around the world, for example the NIST, National Institute of Standards and Technology, United States of Naval Observatory and this national institute of physics laboratory, which is in U.K., through various GPS satellites etc.

So, this UTC time you can get from these places. Let us see, we have seen so many centralized synchronized algorithms, at least 2 we have seen, Cristian's algorithmic and Berkeley algorithm. Now, let us explore the possibility whether, can they be extended for external synchronization. If

we will see Cristian's algorithm can be used, it can be extended if the time server is synchronized with the real time somehow.

So, if because we know in Cristian's algorithm there is a master clock and we are using the time of the master clock. So, if the time of this time server of this master clock can be synchronized or it can be, if the time server it can be synchronized with the real time, maybe with the UTC time somehow, somehow if the time of the time server can be synchronized with the real time, such as UTC, then you can say okay Cristian's algorithm can be used or can be extended to achieve external synchronization.

But you see Berkeley algorithm cannot be used, why? Because in Berkeley algorithm the master clock it requests the timing value, other clocks they are sending the timing value to masters and it takes the averages, those kinds of things. So, that is why this Berkeley algorithm cannot be used, because in external synchronization the clocks must be synchronized with some real time.

So, where? In Berkeley algorithm the times are purely internal. So, that cannot be used for external synchronization. However, in Cristian's algorithm, if somehow we can synchronize the timing of the time server, with the real time, with the UTC then the Cristian's algorithm can be used for achieving external synchronization.

So, let us quickly summarize, Cristian's algorithm can be used for external synchronization if the time server or the master clock is synchronized with the real time somehow, but Berkeley algorithm cannot be used for external synchronization because it uses the internal times or the times of the internal clocks. This is fundamental about external synchronization, details we will skip.

(Refer Slide Time: 0:34:53)

NTP : NETWORK TIME PROTOCOL

- It is a protocol for time sync. in the internet.
- Follows a hierarchical architecture.
 - primary time servers (stratum 1) synchronize to national time standards via radio, satellite etc.
 - secondary servers and clients (stratum 2, 3,...) synchronize to primary servers in a hierarchical manner (stratum 2 servers synchronize with stratum 1, stratum 3 synchronize with stratum 2, and so on).

The slide features a dark blue header with the title in white. The main content is in blue text on a white background. A small video inset shows a man in a suit speaking. At the bottom, there are logos for a university (left) and NPTCL (right).

Now, let us quickly look at a protocol called as NTP, Network Time Protocol. NTP it is a protocol for time synchronization in the internet and let us see how does it work, NTP follows a hierarchical architecture. What do you mean by hierarchical architecture? The servers may be divided into mainly two types, like primary time serves and the secondary servers and the clients. The primary time servers may be kept at stratum 1 level.

They are synchronized to the national time standards, such as UTC. The secondary servers and the clients which maybe kept at stratum 2, stratum 3, et cetera. They are synchronized to their above servers that is prime servers, to the servers at the above level. In a hierarchical manner. What do you mean by hierarchical manner here? That stratum 2 servers, they synchronize with stratum 1, stratum 3 servers, they synchronize with stratum 2 and so on. So, this is what we call as this hierarchical architecture.

(Refer Slide Time: 0:35:58)

NTP : NETWORK TIME PROTOCOL

- Reliability is ensured using redundant servers.
- Communication is made by multicast (usually within LAN servers), symmetric (usually within multiple geographically close servers), or client servers (to higher stratum servers).
- Complex algorithms to combine and filter times.
- Synchronization is possible to within tens of milliseconds for most machines.
- Still it is just a best-effort service, no guarantees.
- Refer [RFC 1305](#) and www.eecis.udel.edu/~ntp/ for more details.

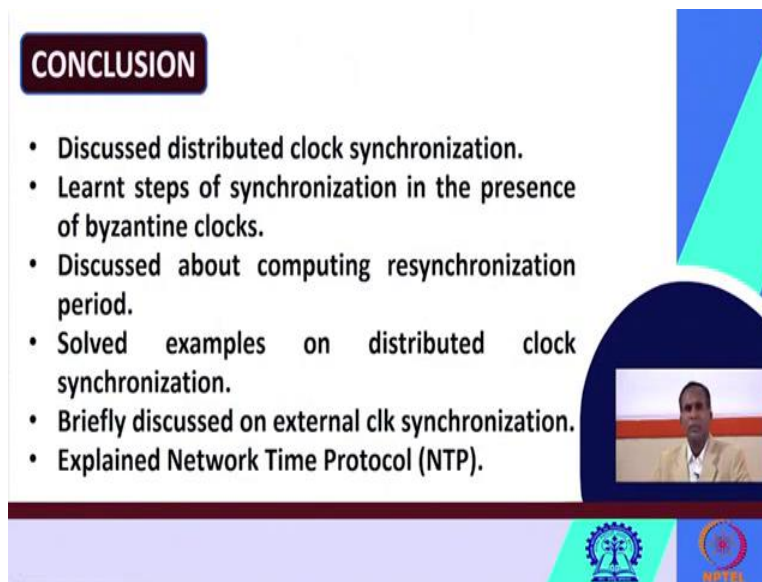
Then we will quickly look at some other important aspects of this network time protocol. So, in network time protocol the reliability, how it is ensured? The reliability is ensured using redundant servers. So, we can put redundant servers or you can put some more servers, extra servers. So, reliability is ensured by using redundant servers, the communication is made by using multi cast symmetric or client servers.

Communication is made by multi class that means usually within LAN servers. Symmetric which are usually within multiple geographically close servers or by client servers, client servers means here where stratum servers are there, communications is made by multi class where usually within LAN servers this may be done, symmetric, usually within multiple geographically close servers or client servers that means to higher stratum servers.

So, in this way a communication is made in network time protocol. So, network time protocol it uses complex algorithms to combine and filter the times. Please remember here that synchronization is possible within 10s of milliseconds per machine, using NTP synchronization is possible to what? Within 10s of milliseconds per most machine.

Please remember that NTP, still it is just best for service, but no guarantees are there, you cannot give any guaranty whether it is fully reliable or fully you have achieved this synchronization, nothing. You may refer RFC 1305 or this website I have given here www.eecix.udel.edu/~ntp/ for more details. You may refer both these sources for more details on NTP.

(Refer Slide Time: 0:38:03)



CONCLUSION

- Discussed distributed clock synchronization.
- Learnt steps of synchronization in the presence of byzantine clocks.
- Discussed about computing resynchronization period.
- Solved examples on distributed clock synchronization.
- Briefly discussed on external clk synchronization.
- Explained Network Time Protocol (NTP).

The slide features a dark red header with the word 'CONCLUSION' in white. Below the header is a list of six bullet points. To the right of the text is a small video inset showing a man in a light-colored shirt speaking. At the bottom of the slide, there are two logos: a blue gear-like logo on the left and a red and blue logo with the text 'NPTL' on the right.

So, today I have discussed about distributed clock synchronization, we have taken some problems, we have learned the steps of synchronization. How synchronization can be made in distributed systems with the presence of both byzantine clocks or bad clocks. We have seen how synchronization can be made in the presence of bad and byzantine clocks.

We have also discussed about computing the resynchronization period, ΔT , how we can compute the resynchronization period ΔT in distributed systems. We have solved some examples on distributed clock synchronization. We have also briefly discussed on external clock synchronization, we have also explained about a protocol called NTP, which stands for network time protocol.

(Refer Slide Time: 0:38:48)



REFERENCES

1. Rajib Mall, Real-Time Systems: Theory and Practice, 1st Edition, 2007, Pearson Education
2. C. M. Krishna & K. G. Shin, Real-Time Systems, 2017, Tata McGraw Hill Education

The slide features a dark red header with the word 'REFERENCES' in white. Below the header, two references are listed in black text. On the right side, there is a video inset showing a man in a light-colored shirt speaking. The slide has a decorative background with blue and green geometric shapes and logos at the bottom, including a gear-like logo and the NPTEL logo.

We have taken these details from these two books. Thank you very much for your patineceful hearing, thank you.