Slide 1:

Hello, I am Soham Bhattacharyya, and this is an explainer video on how timers work in computers and how synchronisation is achieved by distributed systems.

Slide 2:

It is *common* observation that *even* after rebooting the computer, we get the *exact* date and time. So, *how* is this possible? For the answer, we need to know that the clock system in a computer is comprised of two main components. First comes the Real Time Clock, which is powered by a source that is independent from the main power supply of the computer. This power source keeps the real time clock running even when the main power is shut off, for at least a few weeks. When this source is exhausted while the computer is shut down, the system loses track of time and asks for the current date and time when it is rebooted.

The other component is the System Clock which is maintained by the kernel of an operating system and is used to set the tasks and processes – their synchronization and scheduling, settings and managing interrupts, setting timer etc. It is directly dependent on the RTC. The system clock reports seconds and microseconds since a start point from the system boot up procedure. Basically the system clock is a digital signal emitter, which emits signal composed of binary.

Slide 3:

Coming to the Real time clock; like we already know, it has an alternate source of power so that it can continue to keep time even while the primary source of power is off or unavailable. The alternate source of power is usually a lithium battery, or, in newer systems, a supercapacitor because not only are they rechargeable, but also can be soldered and can supply power to battery backed RAM.

Most RTCs have a crystal oscillator. Generally an RTC is an Integrated Circuit but can also be in the form of software or be radio-based. The IC has a piezoelectric crystal. When a field is applied across the crystal and removed from it alternatingly, it generates electric signals of very precise frequency. The frequency is generally 32.768 kHz. Being exactly 215 cycles per second, it is a convenient rate to use with simple binary calculator circuits.

Slide 4:

The system clock is a software clock maintained by the kernel and used to implement gettimeofday as well as setting timestamps on files, etc. The system clock reports seconds and microseconds since a start point, defined to be the POSIX Epoch in POSIX based systems. This means, it is supposed to report wall clock time, which RTCs also do.

A key difference between an RTC and the system clock is that RTCs run even when the system is in a low power state, and the system clock can't. Until it is initialized, the system clock can only report time since system boot ... not since the POSIX Epoch. So at boot time, and after resuming from a system low power state, the system clock will often be set to the current wall clock time using an RTC. Systems without an RTC need to set the system clock using another clock, maybe across the network or by entering that data manually.

Slide 5:

The system clock is typically implemented as a programmable interval timer that periodically interrupts the CPU, which then starts executing a timer interrupt service routine. This routine typically adds one tick to the system clock and handles other periodic housekeeping tasks like pre-emption, before returning to the task the CPU was executing before the interruption.

Now, this programmable interval timer is a counter that generates an output signal when it reaches a programmed count. The output signal triggers an interrupt.

After the context switch on each periodic interrupt generated by the PIT, the timer interrupt service routine performs some tasks.

First, both the absolute time and elapsed time is updated. Absolute time is time kept in calendar date, hours, minutes, and seconds. Elapsed time is usually kept in ticks and indicates how long the system has been running since power up.

Then, a registered kernel function is called to notify the passage of a pre-programmed period. This information is used by the kernel in scheduling tasks.

Slide 6:

Finally, the interrupt is acknowledged, the necessary timer control register is reinitialised and the context is restored from the interrupt.

Slide 7:

Coming to distributed systems, also known as distributed computing, it is a system with multiple components located on different machines that communicate and coordinate actions in order to appear as a single coherent system to the end-user.

In a distributed system, each machine works toward a common goal and the end-user views results as one cohesive unit.

Each machine has its own end-user and the distributed system facilitates sharing resources or communication services.

Slide 8:

In distributed systems, there is need of proper allocation of resources to preserve the state of resources and help coordinate between the several processes. This is achieved via synchronisation.

Clock Synchronisation can be achieved in two ways:

External clock synchronization is the one in which an external reference clock is present. It is used as a reference and the nodes in the system can set and adjust their time accordingly.

Internal clock synchronization is the one in which each node shares its time with other nodes and all the nodes set and adjust their times accordingly.

In addition, there are 2 types of clock synchronisation algorithms:

Centralized is the one in which a time server is used as a reference. The single time server propagates its time to the nodes and all the nodes adjust the time accordingly. It is dependent on single time server so if that node fails, the whole system will lose synchronization.

Distributed is the one in which there is no centralized time server present. Instead the nodes adjust their time by using their local time and then, taking the average of the differences of time with other nodes. Distributed algorithms overcome the issue of centralized algorithms like the scalability and single point failure.

Slide 9:

Berkeley’s Algorithm is a clock synchronization technique used in distributed systems. The algorithm assumes that each machine-node in the network: either doesn’t have an accurate time source, or doesn’t possess a UTC server. It is intended for use within intranets.

In the first phase of the algorithm, a leader or a master node is chosen via an election process such as Chang and Roberts algorithm.

In the second phase of the algorithm, the Master periodically pings slave nodes and fetches clock time from theme using an algorithm like the Cristian’s algorithm

As part of Cristian’s algorithm, the Master Node first polls a slave node at time T0

Slide 10:

Then, the slave node fetches its own time using a get-time function from its kernel. Let this be TSlave. Finally, the slave node returns its time to the master at time T1. The effective time of the slave is calculated by adding the half of the round trip time to the time returned by the slave node.

Slide 11:

Finally, the Master node calculates average time difference between all the clock times received and the clock time given by master’s system clock itself. This average time difference is added to the current time at master’s system clock and broadcasted over the network.

Slide 12:

Now, coming to an example of a Distributed Synchronisation algorithm, the global averaging algorithm makes each node in the network broadcast its local time in the form of special ‘resynchronisation’ messages to all other nodes. After broadcasting, the clock process of the node waits for some time period t.

During this period, it collects ‘resync’ messages from other nodes and records their receiving times based on its own local clock time.

At the end of the waiting period, it estimates the skew of its own clock with respect to all other nodes.

To find the correct time, first, the average of the estimated skews is taken and used as a correction for its local clock.

Each node limits the impact of faulty clock by discarding highest and lowest estimated skews and then calculates average of estimated skew.

There are some

limitations of this algorithm which are as follow:

i. This method does not scale well.

ii. The Network should support broadcast facility.

iii. Due to large amount of messages network traffic increased.

Therefore, this algorithm may be useful for small networks only.