Time, Clocks, and Global State

Armin R. Mikler

Overview

- Introduction
- Physical Clocks
  - Skew
  - Drift
- Clock Synchronization
  - In synchronous systems
  - Cristian's method
  - Berkeley's algorithm
  - The Network Time Protocol
- Logical Time and Clocks
- Events
  - Event Ordering
  - Causal Event Relationships
- Lamport's Algorithm
- Global States and Consistent Cuts

Introduction

Time is an important theoretical construct in understanding how distributed computations unfold!

In order to know at what time a particular event occurred at a particular computer, it is necessary to synchronize its clock with an authoritative external time source!

Distributed algorithms that depend on synchronized time have been developed. These include:

1. Maintaining consistency of distributed data;
2. Serialization of Transaction
3. Authentication and Authorization based on tickets/certificates (Kerberos)

Thanks Herr Einstein!!

The *Special Theory of Relativity* establishes the consequences that follow from the observation that the speed of light is constant for all observers, regardless of their relative speed.

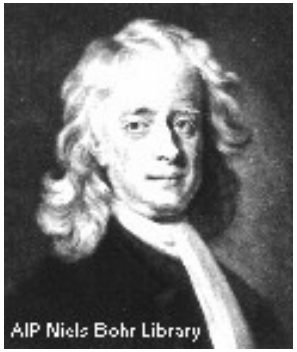
From this assumption, it has been proven that two events that are judged (perceived) to be simultaneous in one frame of reference are not necessarily simultaneous according to two observers in frames of reference that are in relative motion.

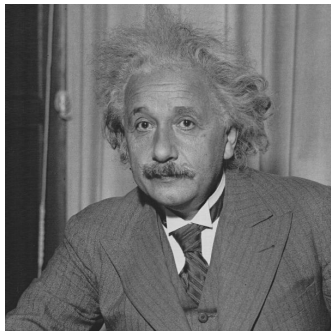
For example: an observer on Earth and an observer traveling away from Earth will disagree on the time interval between event.

In the extreme – the order of two events may be reversed for two different observers!

The reversal of events by two different observers cannot occur if the two events are causally dependent.

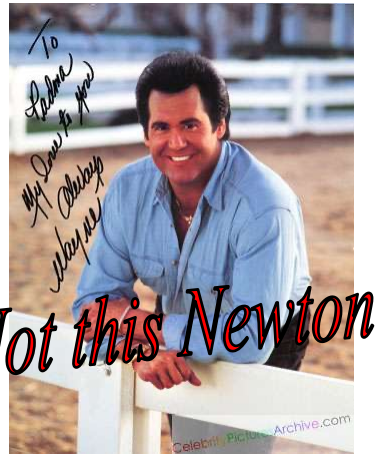
The physical effect follows the physical cause for all observers.





The elapsed time between the occurrence of events may however vary.

The timing of physical events was hence show to be relative to the observer, thereby discrediting *Newton's* notion of absolute physical time !!



CONCLUSION: There is no special physical clock in the universe to which we can appeal to precisely measure intervals of time !

Events, Clocks, and Ticks

A distributed system is generally viewed as a collection $\mathcal{P}$ processes p_i, i= 1,2,…N.

We define an event to be the execution of a single action by p_i.

Each process p_i executes on a single processor and has a state s_i associated with it.

We can view each process p_i as to execute a sequence of actions that fall in one of the following categories:

1. Sending a message;
2. Receiving a message;
3. Performing a computation that alters its state s_i;

The sequence of events within a single process p_i can be totally ordered, which is generally denoted by $e \rightarrow_i e'$.

We can then define the history or process p_i to be a the sequence of events that take place within:

$history(p_i) = h_i = \langle e_i^0, e_i^1, e_i^2, \ldots \rangle$

Why do we care about history ??

→We will later use the histories of concurrent processes p_i and p_j to argue that any allowable execution sequence (or schedule) is consistent with executing p_i and p_j sequentially.

Note: In order to globally order events that occur at different hosts in the DS, nodes must agree on *when theses events happen!*

This will lead to the notion of *serializability* – the property that assures that events (histories) of two or more concurrent processes can be ordered.

This necessitates the introduction of protocols that allow nodes to obtain the current time.

It further requires that nodes agree on what time it is when event take place.

Clocks, Drift, and Skew

Each node in the DS contains its own *physical clock* !

Physical clocks are HW devices that count oscillations of a crystal or quartz.

After a specified number of oscillations, the clock increments a register, thereby adding one *clock-tick* to a counter the represents the passing of time.

The operating system transforms the HW clock into a software-based clock by reading the clock register !

The OS reads the HW-clock value $H_i(t)$, scales it, and adds an offset to produce a SW-clock $C_i(t) = \alpha H_i(t) + \beta$.

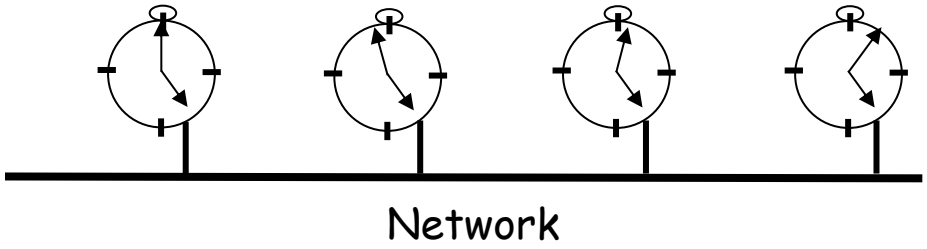
$C_i(t)$ approximates the physical time t at process p_i.

$C_i(t)$ may be implemented by a 64-bit word, representing nanoseconds that have elapsed at time t.

Successive events can be timed if the *clock resolutions* is smaller that the time interval between the two events.

Computer clocks, like any other clocks tend not to be in perfect agreement !!

Clock Skew



Definition:
Clock Drift is the effect that a clock experiences when its crystal is subject to physical variations and oscillates at different rates. On possible cause: *Temperature variation.*

Definition:
Clock Skew is the instantaneous difference between the readings of any two clocks !!

i.e., $|C_i(t) - C_j(t)|$

Drift in ordinary quartz crystals is generally limited to 10^{-6} sec/sec or 1 second every 11.6 days.
High Precision Crystals: 10^{-7} – 10^{-8}

Coordinated Universal Time (UTC)

What *is* the correct time ??

Since we cannot refer to any universal authority to answer this question, we must rely on the availability of highly reliable physical clocks.

Atomic oscillators have a drift rate of about 10^{-13}. The output of these clocks is used as standard for elapsed real time, known as ***International Atomic Time.***

The standard second has been defined by the 9,192,631,770 periods of transition between the two hyperfine levels of the ground state of Caesium-133 (Cs^{133}).

Seconds, days, months, and years are rooted in astronomical time.

UTC is based on atomic time but leap seconds are inserted or deleted to keep in step with astronomical time.

Clock Synchronization

So, how does UTC make it into the nodes of the DS ??

UTC Signals are transmitted from land-based radio stations and satellites covering many parts of the earth.

- Satellite sources include GPS
- Receivers are available commercially
- Land-based station accuracy = .1 – 10ms
- GPS accuracy is about 1μs
- UTC is available via phone line
- Accuracy over phone line is several ms

- Sources include: NIST

We need to distinguish:

1. External Synchronization
2. Internal Synchronization

Internal/External Synchronization

External synchronization refers to synchronization of process' clocks C_i with an authoritative external source S.

Let $D>0$ be the synchronization bound and S be the source of UTC.

Then $|S(t) - C_i(t)| < D$ *for* $i=1,2,\ldots,N$ and for all real times t.

We say that clocks C_i are *accurate* within the bound of D

Internal synchronization refers to synchronization of process' clocks C_i with each other.

Let $D>0$ be the synchronization bound and C_i and C_j are clocks at processes p_i and p_j, respectively.

Then $|C_i(t) - C_j(t)| < D$ *for* $i,j=1,2,\ldots,N$ and for all real times t.

We say that clocks C_i, C_j agree within the bound of D

Note that clocks that are internally synchronized are not necessarily externally synchronized. i.e., even though they agree with each other, the drift collectively from the external source of time.