Distributed Systems

Monsoon 2024

Lecture 2

International Institute of Information Technology

Hyderabad, India

Time in Distributed Systems

- Understand the notion of time in distributed systems.
- How to maintain clocks in a distributed system.
  - Physical vs. Logical time
- Various algorithms/protocols for maintaining logical time.

Physical Time and Synchronization

- Distributed systems run several applications that are critical.
- Why do these applications require to know the “time” in certain contexts ?

Physical Time and Synchronization

- Distributed systems run several applications that are critical.
- Why do these applications require to know the “time” in certain contexts ?
- Examples include:
  - Assigning Timestamps to events
  - Applications using time-outs to deduce certain actions/events.
  - Performance and Resource Usages
  - Scheduling decisions
  - And so on…

Physical Time and Synchronization

- Imagine a process running on your computer.
- How does it get to know the current time?

Physical Time and Synchronization

- Imagine a process running on your computer.
- How does it get to know the current time?
- In centralized systems, there is only a single clock. A process gets the time by simply issuing a system call to the kernel.
  - The time returned by the kernel is the “time”.

Physical Time and Synchronization

- No Global clock or shared memory
  - Each processor has its own internal clock and its own notion of time.
- Drift: Clocks can easily drift seconds per day, accumulating significant errors over time.
- Synchronize at the start
  - Different clocks tick at different rates, may not remain synchronized even though synchronized when they start.
- These issues clearly poses serious problems to applications that depend on a synchronized notion of time.

Physical Time and Synchronization

- Before we go to discuss time in distributed systems, let us look at an in-between situation: that of networked systems.

Protocols for Network Time Synchronization

- One of the first protocols for synchronizing time on a network is by Gusella and Zatti, 1983.
- Their protocol is used in the UNIX BSD 4.3 as adjtime() system call.
- We will review the protocol briefly.

Gusella and Zutti Protocol

- Let A and B be two machines on a network.
- A sends a timestamped message to B and B sends a reply to A.
- Before sending the message, process B computes the time taken for the message from A to reach B as $D_{AB} = \text{timestamp}_B - \text{timestamp}_A$.
- If e_A and e_B are the errors in the times at A and B, then, $\text{timestamp}_A = t_A - e_A$ and $\text{timestamp}_B = t_B - e_B$.
- We can rewrite $D_{AB} = t_B - e_B - t_A + e_A = T_{AB} + \delta - \text{Error}_{AB}$.
  - The quantity T_{AB} refers to the transmission delay with offset δ that we are trying to estimate.

Gusella and Zutti Protocol

- A sends a timestamped message to B and B sends a reply to A.
- Denote the transmission delay from A to B as D_{AB}, we can rewrite $D_{AB} = t_B - e_B - t_A + e_A = T_{AB} + \delta - Error_{AB}$.
- Add a subscript 1 to the above quantities to indicate one set of computations. So,
- $D_{AB} = t_{B1} - e_{B1} - t_{A1} + e_{A1} = T_{AB} + \delta - Error_{AB1}$.
- As B sends a reply to process A, the above calculations can be repeated at process A to obtain the following.

Gusella and Zutti Protocol

- $D_{AB1} = t_{B1} - e_{B1} - t_{A1} + e_{A1} = T_{AB1} + \delta - Error_{AB1}$.
- As B sends a reply to process A, the above calculations can be repeated at process A to obtain the following.
- $D_{AB2} = (t_{A2} - t_{B2}) - (e_{A2} - e_{B2}) = T_{AB2} - \delta - Error_{AB2}$.
- A can also compute the difference in the transmission delay between A to B and B to A as

$$\Delta' = \frac{D_{AB1} - D_{AB2}}{2}$$

$$= \delta + \frac{T_{AB1} - T_{AB2}}{2} - \frac{Error_{AB1} - Error_{AB2}}{2}$$

Gusella and Zutti Protocol

- Assume that the random variables corresponding to $Error_{AB1}$ and $Error_{AB2}$ are independent and symmetric.
- We can repeat the above experiment N times so that the average Δ' can be computed as follows.
- $\Delta' = \frac{\sum_{i=1}^{N} \frac{d_{ABi} - d_{ABi+1}}{2}}{N}$

$$= \delta + \frac{\sum_{i=1}^{N} \frac{T_{ABi} - T_{ABi+1}}{2}}{N} - \frac{\sum_{i=1}^{N} \frac{Error_{ABi} - Error_{ABi+1}}{2}}{N}$$

- The second and the third terms in the right hand side have a mean of 0.
- Hence, for large N, by appealing to the Strong Law of Large Numbers, the right hand side converges to δ

Gusella and Zutti Protocol

- The TEMPO protocol designates one computer in a local network as a master and the rest as slaves.
- The master is responsible for initiating and coordinating time synchronization.
- The master uses the following steps:
  - The master interacts with each of the slave machines, say S1, S2, · · · , and obtains estimates of Δ'_{S1}, Δ'_{S2}, · · · , with respect to each of the slave machines.
  - The master then computes the average of the quantities, Δ' S1 , Δ' S2 , · · · ,, as the network average error.
  - The master directs slave machine S_i to adjust its clocks by an offset equal to the difference of Δ' Si and the network average error

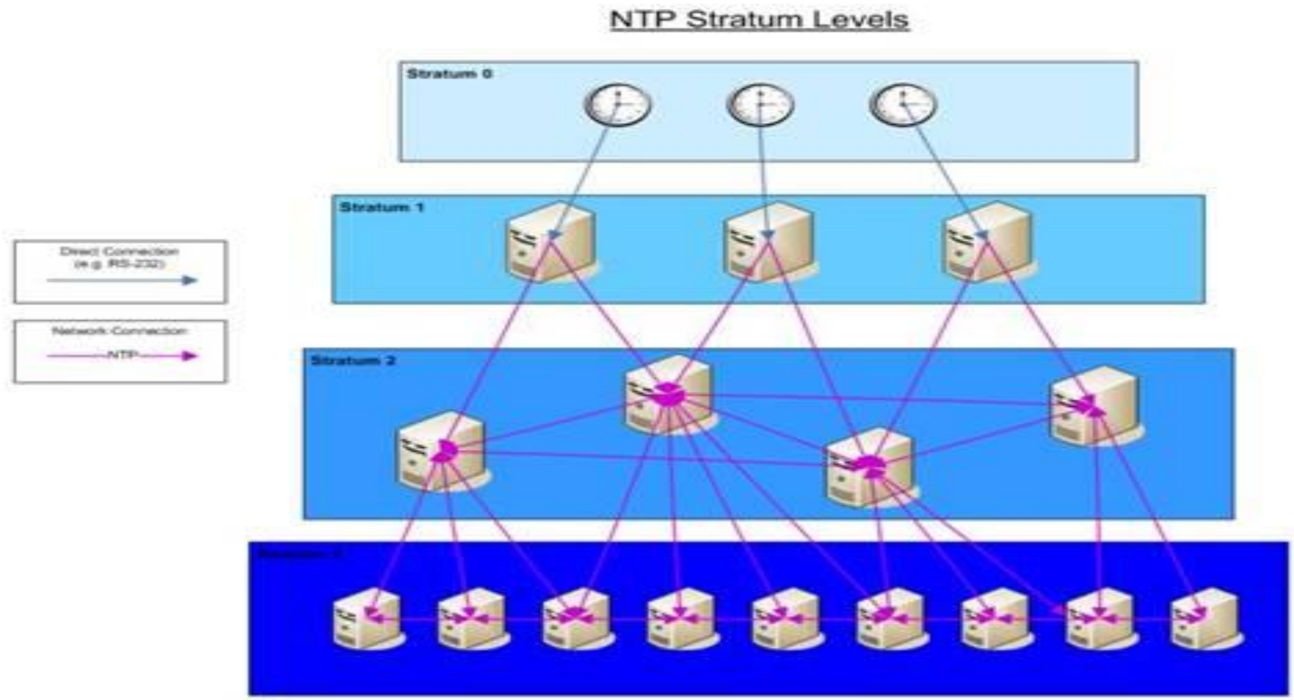
Guselli and Zutti Protocol

- Notice that the above protocol requires some slave machines to set their time to be in the past.
- The system call adjtime() handles these by increasing/decreasing the rate at which the clock moves.
- This can create situations that are not consistent even as the time is monotonic.

Network Time Protocol

- A protocol through which systems connected on a common network, such as the Internet, can synchronize time – up to an error.
- The NTP model has two entities: clients and servers.
- Clients get the current time from a server in the same network.
- The servers are arranged in a hierarchy.
  - This hierarchy can go as deep as 15 levels currently.
- NTP Servers get time in two ways
  - Either from one or more NTP servers in the next upper hierarchy
  - By having peer relationship with other NTP servers in the same hierarchy.

NTP Peers



An example of a typical NTP network (Image courtesy of Wikipedia)

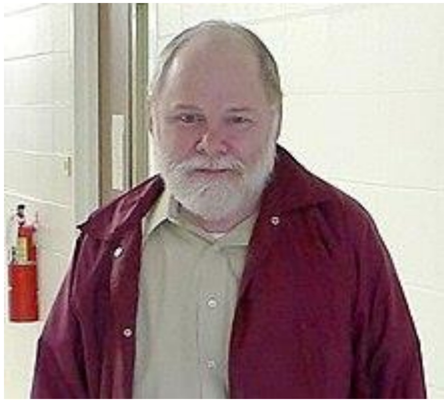
- Synchronizing once is not enough.
  - Clocks can drift over time in an unpredictable direction.
- Peers in an NTP set up exchange messages periodically to synchronize their relative times.
  - So, peers may have to synchronize amongst them.

NTP

- How do peers in NTP do this synchronization?

The Internet Time Keeper

- NTP is designed by David Mills, U. Delaware.
- Passed away recently in January 2024.
- Homepage:www.eecs.udel.edu/~mills



Synchronization Amongst Peers

- Let A and B be any two peers. Let C_a and C_b be the clocks at A and B.
- At time t, if $C_a(t) = t$, then we say that A has a perfect clock.
- The offset of a clock $C_a(t)$ measured as $C_a(t) – t$.
- The offset of a clock $C_a(t)$ relative to another clock $C_b(t)$ at time t is measured as $C_a(t) – C_b(t)$.
- The skew of a clock $C_a(t)$ is $F_a(t) – F_t$ where $F_a(t) = 1/C_a(t)$ and $F_t = 1/t$. [F for frequency].
- The skew of a clock $C_a(t)$ relative to another clock $C_b(t)$ at time t is measured as $F_a(t) – F_b(t)$.
- The drift of a clock C_a is the second derivative of C_a.

Synchronization Amongst Peers

- For a specified threshold ρ, we say that a clock is accurate if $1 - \rho \le dC/dt \le 1 + \rho$.

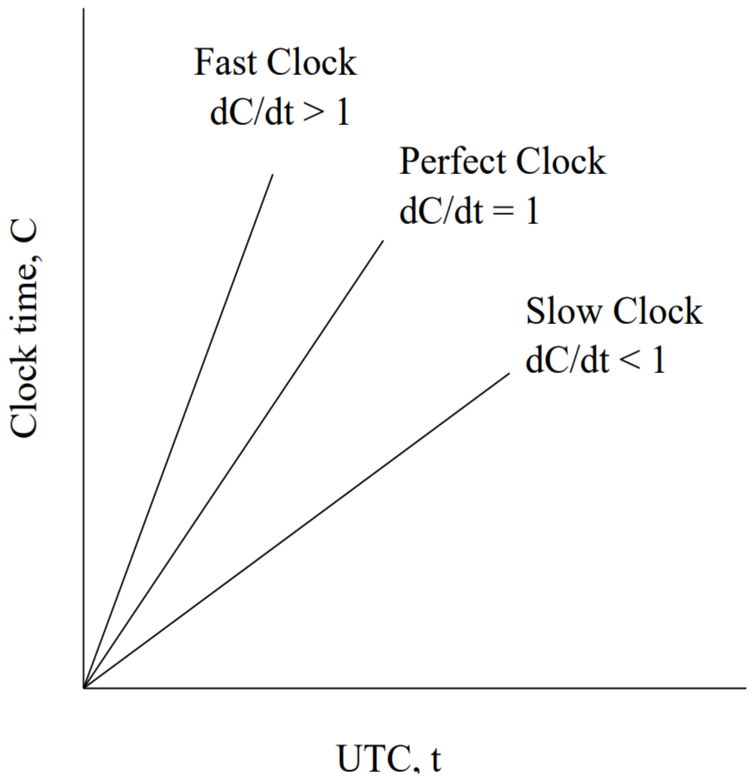


Figure 3.5: The behavior of fast, slow, and perfect clocks with respect to UTC.

The NTP Offset Estimation

- With the above notation, let us see how two peers estimate the relative offset delay.
- The basic idea is to perform several trials and choose the trial with the minimum delay.
- Each trial has the following format.

The NTP Offset Estimation

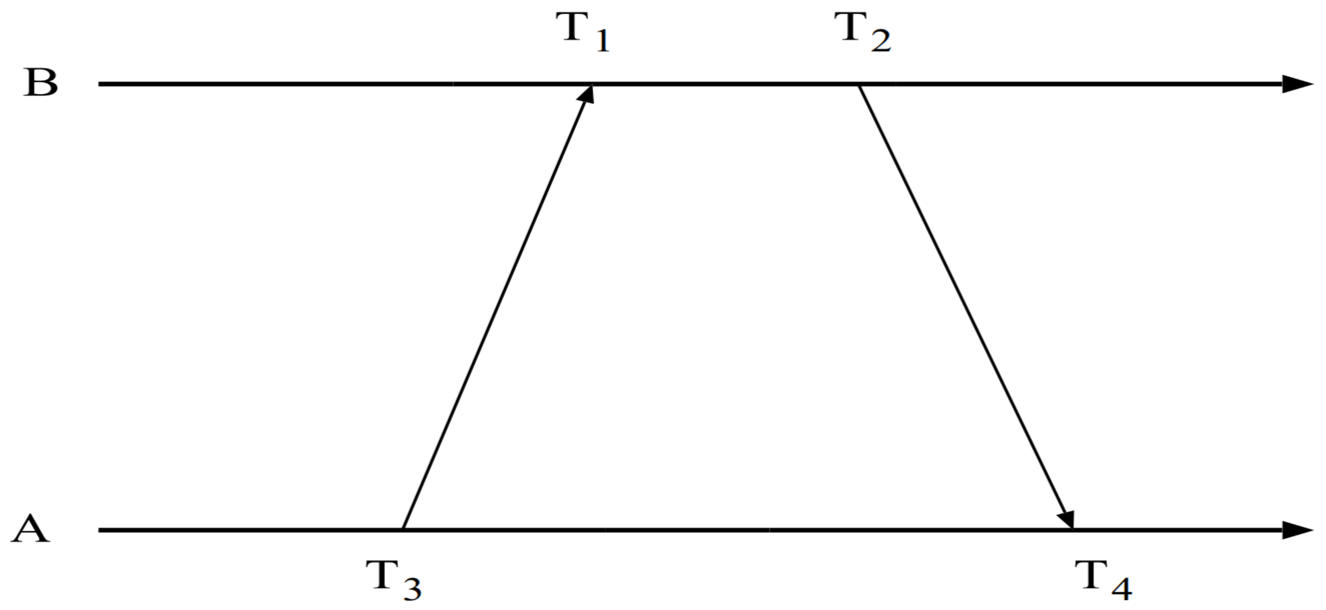


Figure 3.6: Offset and delay estimation.

- Assume clocks A and B are stable.
- Let T_1, T_2, T_3, and T_4 be the four most recent timestamps of message exchanges.

The NTP Offset Estimation

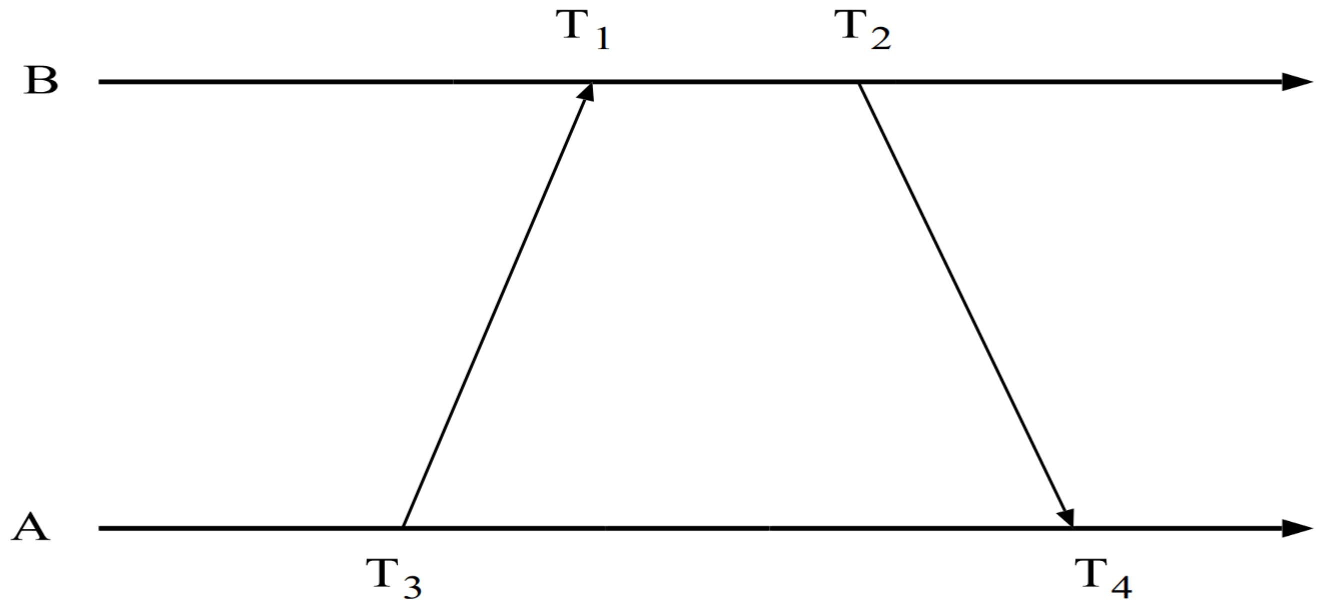


Figure 3.6: Offset and delay estimation.

- Let $a = T_1 - T_3$ and $b = T_2 - T_4$.
- If the network delay difference from A to B and from B to A, called differential delay, is small, the clock offset θ and roundtrip delay δ of B relative to A at time T_4 are approximately given by the following.
  - $\Theta = (a + b)/2$, $\delta = a - b$

The NTP Offset Estimation

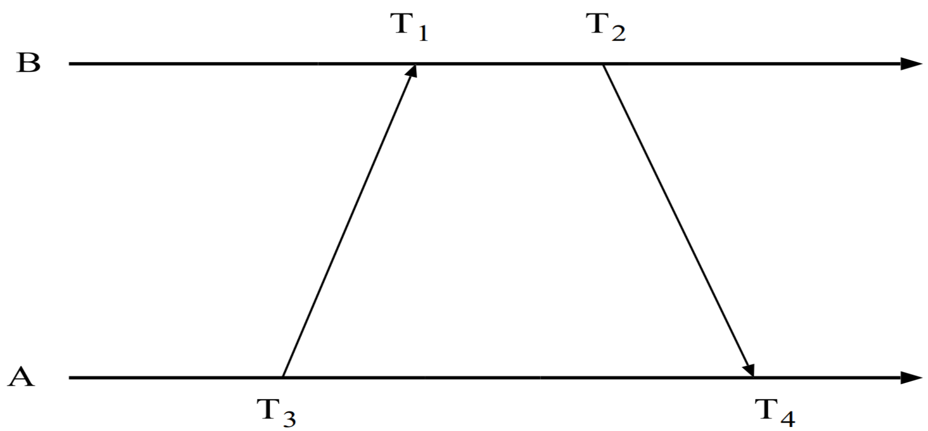


Figure 3.6: Offset and delay estimation.

- Let $a = T_1 - T_3$ and $b = T_2 - T_4$.
- If the network delay difference from A to B and from B to A, called differential delay, is small, the clock offset θ and roundtrip delay δ of B relative to A at time T_4 are approximately given by the following.
  - $\theta = (a + b)/2$, $\delta = a - b$
- Round-Trip Delay = $T_1 - T_3 + T_4 - T_2 = a - b$.
- Measured at B, $T_1 = T_3 + (\delta/2) + \theta$, and $T_2 = T_4 - (\delta/2) + \theta$. Add the two equations to get $2\theta = a+b$.

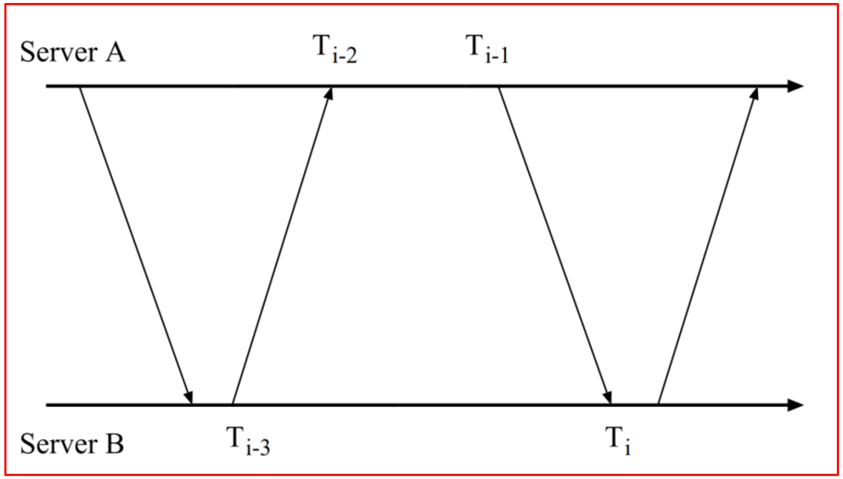
The NTP Offset Estimation

- A pair of servers in symmetric mode exchange pairs of timing messages.
- A store of data is then built up about the relationship between the two servers (pairs of offset and delay).
- Specifically, assume that each peer maintains pairs (O_i, D_i), where
- O_i - measure of offset (θ)
- D_i - transmission delay of two messages (δ).
- The offset corresponding to the minimum delay is chosen.

The NTP Offset Estimation

- The offset between A's clock and B's clock is O. If A's local clock time is A(t) and B's local clock time is B(t), we have

$$A(t) = B(t) + O$$



- Then,

$$T_{i-2} = T_{i-3} + t + O$$
$$T_i = T_{i-1} - O + t'$$

- Assuming $t = t'$, the offset O_i can be estimated as:

$$O_i = (T_{i-2} - T_{i-3} + T_{i-1} - T_i)/2$$

- The round-trip delay is estimated as:

$$D_i = (T_i - T_{i-3}) - (T_{i-1} - T_{i-2})$$

- The eight most recent pairs of (O_i , D_i) are retained.
- The value of O_i that corresponds to minimum D_i is chosen to estimate O.

The Next Steps

- Finding the offset is only one part of the NTP protocol.
- There exists a lot of follow-up tasks that systems do
  - How to (re)set the time using the offset?
  - How to choose peers and how to remove bad peers?
  - How to account for temporary glitches?
  - How to tolerate faults?

Why NTP Does Not Suffice?

- For distributed systems, there is no way to arrange the stratum of NTP servers.
- Plus, the accuracy of NTP may not be enough in a large scale distributed setting.
  - Typical accuracy is of the order of milliseconds.
  - Bad days, the errors can be of the order of 100s of ms.
  - LAN accuracy of the order of less than a ms.
- And the required peering support to allow for clock synchronization.

NTP Style Approaches at Global Scale

- Many consumer-facing planet scale organizations such as Google, Facebook, Amazon, also need a good time synchronization protocol.
- If NTP in its form does not suffice, how do they cope?
- The answer lies in engineering.

TrueTime

- Consider a setting where datacenters are spread across the planet.
- Each datacenter has one or more TimeMaster servers, also known as time servers.
- The TrueTime solution has two kinds of TimeMaster servers.
  - GPS Time Master: contain GPS receiver nodes and can interface with GPS signals and receive time infromation via satellites.
  - Armageddon Master: contain atomic clocks that are highly accurate.
- Using GPS based time information and atomic clocks is to account for good redundancy.
  - The factors that affect the failure of these two time systems are independent

TrueTime

- TimeMasters periodically exchange and compare their time with other TimeMasters.
- These are interconnected with dedicated, fault-tolerant, high-speed communication fabric.
- A client pings a set of TimeMaster servers of both kinds, the GPS TimeMasters and the Armageddon TimeMasters, across the network.

Other Related Systems

- Amazon internally uses a protocol called TimeSync for time service.
- This too is based on GPS time and atomic clocks.

- Read also about chrony by Facebook.

Moving Over to Distributed Systems

- Fortunately however, asynchronous distributed computations make progress in spurts.
- Causality: Dictionary meaning is to involve a cause, or marked by cause and effect
  - Fundamental to the design and analysis of parallel and distributed computing and operating systems.
  - Allows reasoning, analyzing, and drawing inferences about a computation.
- Causal precedence relation among the events of processes helps in
  - distributed algorithms design,
  - tracking of dependent events,
  - knowledge about the progress of a computation, and
  - concurrency measures

Why NTP Does Not Suffice?

- Therefore, a logical time is sufficient to capture the fundamental monotonicity property associated with causality in distributed systems.
- To define what logical time means, we have to first model distributed computation in terms of events.
  - Include communication channels.
- A bit of rough stretch with lots of notations and definitions.
  - Most definitions are intuitive but require the formal rigour.