Chapter 3
Global Time

Overview

- Notions of causal order, temporal order, and delivery order

- External observers, reference clocks, and global time base
- Sparse time base to view event order in a distributed real-time system
- Internal clock synchronization to compensate for drift offset. Influence of the communication system jitter on the precision of the global time base.
- External time synchronization, time gateways, and the Internet network time protocol (NTP).

3.1 Time and Order

Time is one of the fundamental concepts in physics. Time is an independent variable that determines the sequence of states of a system.
To achieve consistent behavior in a distributed real-time system, all nodes must process events in the same (temporal) order. This requires some kind of global time base which provides globally agreed upon timestamps for all events.

Orders: Temporal Order

Continuum of real time can be modeled by a directed timeline consisting of an infinite set {T} of instants with the following properties:

- {T} is an ordered set (one instance always precedes the other, or vice versa, or they are simultaneous). We call the order of instants a temporal order.
- {T} is a dense set (you can always find a new instance between two existing instances)

A section of time is called a duration. An event takes place at an instant of time. Instants are totally ordered, events are only partially ordered (two simultaneous events are still different, while two simultaneous instants are equivalent). Simultaneous events can be ordered by adding an additional attribute, e.g. the number of the node at which event occurred.

Orders: Causal Order

Example: Rupture of fuel line from fuel tank to injection pump. Pressure changes abruptly, flow changes, after some time engine stops working. Causal order is temporal order plus something.
Example with two events e1 and e2:

e1: Somebody enters a room
e2: The telephone starts ringing

Two scenarios:

1. e2 occurs after e1
2. e1 occurs after e2

Causal order can only be established for case 2.

Orders: Delivery Order

Events are delivered to the different nodes in a real-time system via the communication network. The communication system guarantees that all nodes in the system see the sequence of events in the same order, the delivery order. The delivery order may be different from the temporal order, e.g. in case of prioritized messages.

Clocks

A clock is an objective method for measuring the progress of time.

Clocks: Physical clock

A physical clock is a device for measuring time. It contains a counter, and a physical oscillation mechanism that periodically generates an event to increase the counter. The periodic event is called the microtick of the clock.
The duration between two consecutive microticks is called the granularity g of the clock. The granularity leads to a digitalization error in time measurement.
Notation: $microtick_i^k$ means microtick *i* of clock *k*.

3.1 Time and Order

Clocks: Reference clock

We assume an omniscient external observer who can observe all events of interest. This observer possesses a unique reference clock z with frequency f^z. This clock is in perfect agreement with the international standard of time.

We call $1/f^z$ the granularity g^z of clock z. We assume that the frequency of the reference clock is very high.

Clock(event) denotes the timestamp generated by the use of a given clock to timestamp event e. When we timestamp events with the reference clock, z(e), this is called the absolute timestamp of event e.

The duration between two events is measured by counting microticks of the reference clock. The granularity g^k of clock *k* is given by the nominal number n^k of microticks of the reference clock *z* between two microticks of this clock *k*.

The temporal order of events that occur between any two consecutive microticks of the reference clock cannot be established from their timestamps.

3.1 Time and Order

Clocks: Clock drift

The drift of a physical clock *k* is determined by measuring the duration of a granule of clock *k* with the reference clock *z*, and dividing it by the nominal number n^k of reference clock microticks in a granule:

$$drift_i^k = \frac{z(microtick_{i+1}^k) - z(microtick_i^k)}{n^k}$$

For a good clock, the drift is very close to one. For notational convenience, thedrift rate has been introduced:

$$\rho_i^k = \left| \frac{z(microtick_{i+1}^k) - z(microtick_i^k)}{n^k} - 1 \right|$$

Typical drift rates are in the range of 10^{-2} to 10^{-7} sec/sec or better. Every clock has a non-zero drift rate. Thus any free-running clock leaves any bounded relative time interval after a finite time.

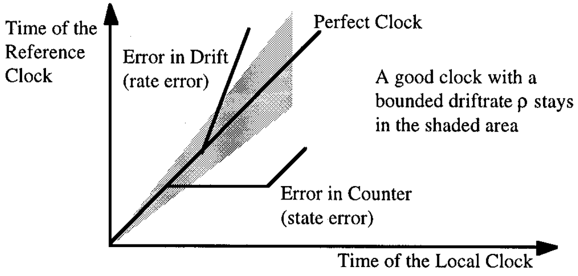
Example: Patriot missile tracking system.

3.1 Time and Order

Clocks: Failure mode of a clock

Clocks exhibit two types of failures:

- Counter becomes faulty, exhibiting bad counter values
- Drift rate exceeds specified limits


Time of the Reference Clock
Error in Drift (rate error)
Perfect Clock
A good clock with a bounded driftrate ρ stays in the shaded area
Error in Counter (state error)
Time of the Local Clock

Precision and Accuracy: Offset

Offset at microtick i between two clocks j and k with the same granularity is defined as

$$offset_i^{jk} = \left| z(microtick_i^j) - z(microtick_i^k) \right|$$

Precision and Accuracy: Precision

3.1 Time and Order

Given an ensemble of clocks {1, 2, ..., n}, the maximum offset between any two clocks of the ensemble is called the precision Π_i of the ensemble at microtick *i*:

$$\Pi_i = \max_{\forall 1 \le j,k \le n} \left\{ offset_i^{jk} \right\}$$

The maximum of Π_i over an interval of interest is called the precision Π of the ensemble. Because of the drift rate of any physical clock, the clocks in an ensemble will drift apart if they are not resynchronized periodically (internal synchronization).

Precision and Accuracy: Accuracy

The offset of clock *k* with respect to reference clock *z* at microtick *i* is called the $accuracy_i^k$. The maximum offset over all microticks I that are of interest is called the $accuracy^k$ of clock *k*.

To keep a clock within a bounded interval of the reference clock, it must be periodically resynchronized with the reference clock (external synchronization).

External synchronization automatically leads to internal synchronization, but not vice versa.

3.1 Time and Order

Time Standards

For distributed real-time systems, two time bases are relevant.

Time Standards: International Atomic Time (TAI)

TAI defines the second as the duration of 9192631770 periods of the radiation of a specified transition of the cesium atom 133. The definition agrees with astronomical observations. TAI is a chronoscopic time scale, i.e. it does not have discontinuities.

Time Standards: Universal Time Coordinated (UTC)

UTC is the basis for the time on the “wall-clock”. There is a known offset between the local wall-clock time and UTC which is defined by the timezone and daylight savings time.

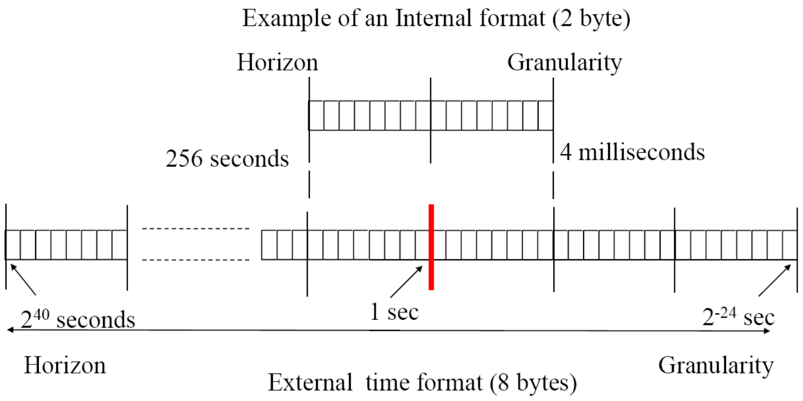
UTC introduces leap seconds once in a while, to maintain synchrony with astronomical phenomena, like day and night. Thus, UTC is not a chronoscopic time scale.

It was agreed that on January 1, 1958 at midnight UTC and TAI had the same value. Since then UTC has deviated from TAI by about 30 seconds.

3.1 Time and Order

Uniform Time Format (OMG Standard)

Start of epoch: January 6, 1980 at 0:00:00 UTC. Granularity is 59,6 ns approx 60 nanosecond, time horizon 34841 years. Similar to IEEE 1588. Time granularity determined by precision of GPS.



3.1 Time and Order

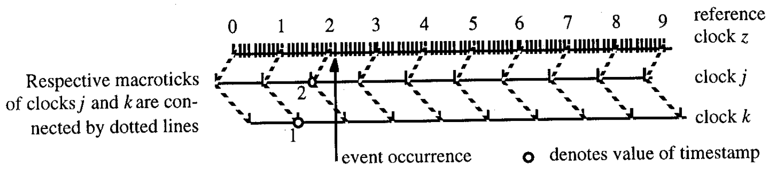
3.2 Time Measurement

3.2 Time Measurement

In a distributed real-time system, local real-time clocks are not tightly synchronized with the reference clock z, and thus reconstructing order of events or measuring time intervals is not straight forward.
A weaker notion of a universal time reference is therefore being used - the concept ofglobal time.

Global Time

To establish global time, we pick a subset of the microticks of each local clock k for the local implementation of a global notion of time, e.g. every tenth microtick. These selected ticks we call macroticks t_i^k.

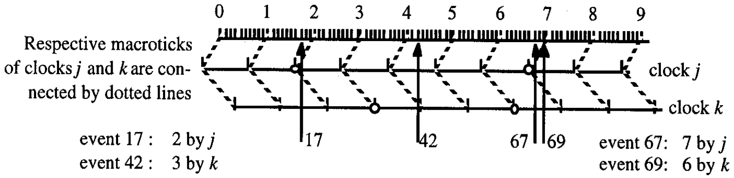

0
1
2
3
4
5
6
7
8
9
reference clock z
clock j
clock k
Respective macroticks of clocks j and k are connected by dotted lines
2
1
event occurrence
o denotes value of timestamp

Reasonableness condition for global time t: all local implementations satisfy condition that granularity g is larger than the global precision Π:

$$g > \Pi$$

In case this reasonability condition is satisfied, global timestamps for a single event differ by at most one tick.

This has a serious implication: we cannot reconstruct the temporal order if timestamps differ by just one tick. Example:



Event 17: 1 by k, 2 by j
Event 42: 4 by j, 3 by k
Event 67: 6 by k, 7 by j
Event 69: 7 by j, 6 by k

Thus, event 67 is time stamped with 7 by clock j, and event 69, which occurs later than event 67 as seen from the reference clock, is time stamped with 6 by clock k.

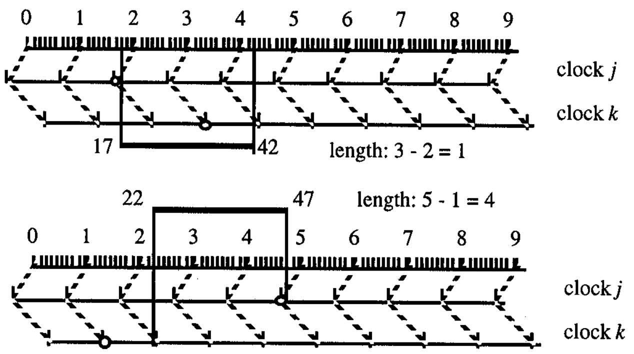
We can reconstruct the temporal order if the timestamps differ by at least two macroticks, because the sum of synchronization and digitalization error is always less than 2 granules.

Interval Measurement

Interval is defined by two events: start event and terminating event. The error measuring the duration of an interval is always less than $2g$. Thus, the true duration d_{true} is bounded by:

$$\left(d_{obs} - 2g\right) < d_{true} < \left(d_{obs} + 2g\right)$$

Example for interval of 25 microticks duration:

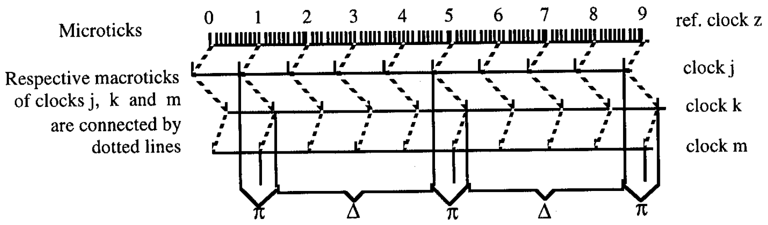


π/Δ-Precedence

π/Δ-precedence means that a subset of the events that happen at about the same time (and is thus close together within π) is separated by a substantial interval (at least Δ) from the elements in another subset.

Example for events that are generated by nodes j, k, m. Every node is to generate an event at 1, 5, 9. All events are generated locally at the same global clock tick within a small time interval π where $\pi < \Pi$. Events occurring at different ticks are at least Δ apart.

$$\left[\left|z(e_i) - z(e_j)\right| \le \pi\right] \vee \left[\left|z(e_i) - z(e_j)\right| > \Delta\right]$$


Microticks
0
1
2
3
4
5
6
7
8
9
ref. clock z
Respective macroticks of clocks j, k and m are connected by dotted lines
clock j
clock k
clock m
π
Δ
π
Δ
π

If π is zero, then any two events occur either at the same time or are at least a duration Δ apart.

Event Set	Observed timestamps condition	Temporal order can be established
0/1g precedent	$\left\|t^j(e1) - t^k(e2)\right\| \ge 0$	No
0/2g precedent	$\left\|t^j(e1) - t^k(e2)\right\| \ge 1$	No
0/3g precedent	$\left\|t^j(e1) - t^k(e2)\right\| \ge 2$	Yes
0/4g precedent	$\left\|t^j(e1) - t^k(e2)\right\| \ge 3$	Yes

To establish the temporal order of observed events, we need at least 0/3g precedence, i.e. the timestamps of two successive events is at least 2 macroticks apart. (One of the three g is eaten up by possible difference of two clocks on different nodes, see reasonableness condition).

3.2 Time Measurement

Fundamental Limits of Time Measurement

Four fundamental limits of time measurement in distributed real-time systems with a reasonable global time base with granularity g:

- If a single event is observed by two nodes, there is always the possibility that the timestamps differ by two ticks. A one tick difference does not suffice to recover the temporal order of events.
- If the observed duration of an interval is d_{obs}, the true duration d_{true} is bounded by

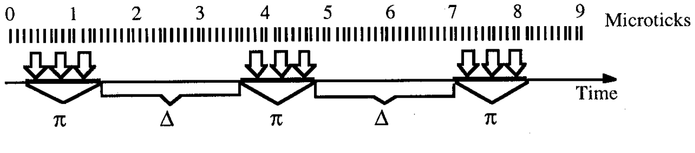
$$\left(d_{obs} - 2g\right) < d_{true} < \left(d_{obs} + 2g\right)$$

- The temporal order of events can be recovered from their timestamps, if the difference between their timestamps is equal to or greater than 2 ticks.
- The temporal order of events can always be recovered from their timestamps, if the event set is at least 0/3g precedent.

3.3 Dense Time versus Sparse Time

3.3 Dense Time versus Sparse Time

In case events can occur any time, we speak of a dense time base.
In case events are permitted to occur only in certain time intervals π, we speak of a sparse time base.
In a dense time base it can be difficult to establish temporal order.



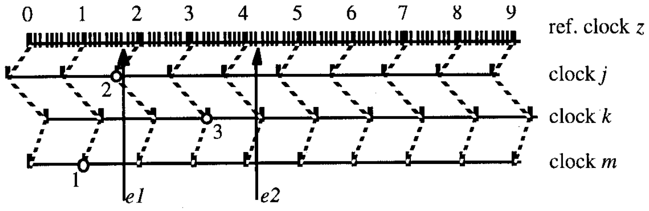
Events ⇩ are only allowed to occur within the intervals π.

3.3 Dense Time versus Sparse Time

Dense Time-Base

Example scenario with two events e1 and e1, 2.5 granules apart. Event e2 is only observed by node k, which reports its timestamp back to nodes j and m.

Node j: e1 observed at time 2; e2 from node k observed at time 3
Node m: e1 observed at time 1; e2 from node k observed at time 3



Node j calculates difference of one tick: events cannot be ordered
Node m calculates difference of two ticks: events can be ordered, with e1 definitely occurred before e2. Thus we have an inconsistent view.

To arrive at a consistent view, the nodes must execute an agreement protocol. However, such protocols come with a lot of communication and processing overhead.

Sparse Time-Base

3.3 Dense Time versus Sparse Time

Example scenario with two clusters A and B, not synchronized, B observes events generated in A.

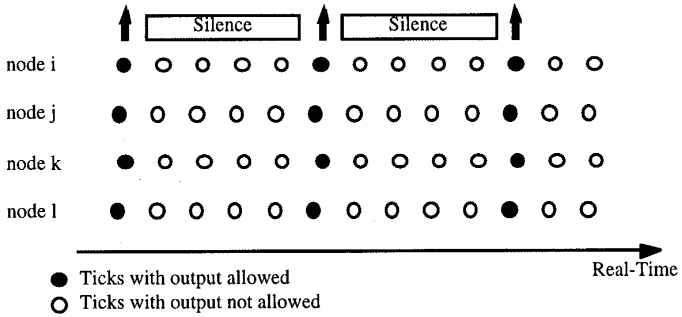
We have up to one granule difference in clocks between nodes in A and B, and we have up to one granule difference in clocks between nodes of A. Likewise, if we generate events in A that are within one granule, they can be time stamped by nodes within B with two ticks difference.

To establish temporal order in this scenario we therefore need at least 1g/4g precedence.

3.3 Dense Time versus Sparse Time

Space-Time Lattice

Example for a space-time lattice for 1/4g precedent event set:



At the instrumentation interface (between computer and controlled object), events occur on a dense time base. We need some kind of agreement protocol there, in case the controlled object is observed by more than one computer node.

Node failures also occur on a dense time base. In a TT architecture the recognition of failures can be restricted to a sparse time base, to avoid an agreement protocol.

3.4 Internal Clock Synchronization

3.4 Internal Clock Synchronization

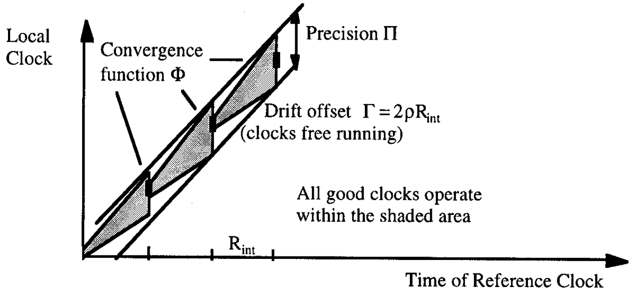
Purpose is to ensure that global ticks of all correct nodes occur within specified precision Π, despite varying drift rates.

Each node has a local oscillator generating the microticks. A subset of the microticks, the ticks or global time ticks, are taken to increment the local global time counter.

3.4 Internal Clock Synchronization

The Synchronization Condition

Global time ticks of each node are periodically resynchronized each resynchronization interval R_{int}. Convergence function Φ denotes offset of time values immediately after resynchronization.



Drift offset Γ and drift rate ρ are related as

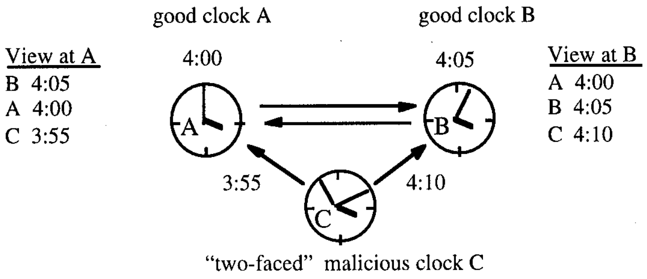
$$\Gamma = 2\rho R_{\text{int}}$$

For convergence function Φ, drift offset Γ, and precision Π holds:

$$\Phi + \Gamma \le \Pi$$

3.4 Internal Clock Synchronization

Byzantine error: “two-faced” clock. It displays to one node a different time than to another node. This can obstruct synchronization. Here each node sets its value to the average value of the ensemble. Neither node A nor node B will correct its value.



Central Master Synchronization

A standard algorithm during the startphase of synchronization is the central master algorithm. A maste node sends its time to all other nodes, and they set their local times to the transmitted time, corrected by the transport latency. Convergence function Φ is determined by the difference between fastest and slowest transmission to slave nodes of ensemble, thelatency jitter ε.
The precision of the central master algorithm is given by:

3.4 Internal Clock Synchronization

$\Pi = \varepsilon + \Gamma$. That means the convergence function is $\mathcal{E}$.

Fault tolerant Sync Algorithms

Precision of the FTA:

Assume a distributed system with N nodes, each one with its own clock (all time values are measured in seconds). At most k out of N clocks behave in a Byzantine manner.
A single Byzantine clock will cause the following difference in the calculated averages at two different nodes in an ensemble of N clocks:
Ebyz = Π /(N-2k)
For K byzantine errors:
EByz = k Π / (N – 2 k)
The convergence function is given by:
k Π / (N – 2 k) + $\mathcal{E}$

Combining the above function with the sync condition, we get the Byzantine error factor.

Distributed Synchronization Algorithms

Typically three phases in distributed clock synchronization algorithms:

3.4 Internal Clock Synchronization

1. Each node acquires knowledge about global time counters of all other nodes
2. Each node analyzes the data, to detect errors, and executes convergence function to correct its local global time counter. If correction term exceeds specified precision of ensemble, node must deactivate itself.
3. Local time counter is adjusted by calculated correction value.

Reading the Global Time: most important term affecting precision is jitter of time messages. Jitter depends on communication network, and at the level where synchronization takes place (application level, kernel, hardware). A small jitter is important to achieve high precision.

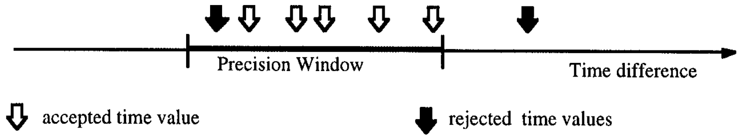
Impossibility Result: Maximum precision obtainable by given latency jitter and number of nodes is related as

$$\Pi = \varepsilon\left(1 - \frac{1}{N}\right)$$

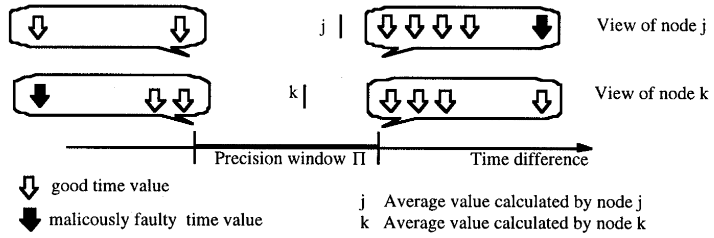
The Convergence Function:

Example showing an ensemble of 7 nodes and one tolerated Byzantine fault. The Fault-Tolerant-Average (FTA) algorithm takes average of five accepted time values shown.

3.4 Internal Clock Synchronization



Worst case scenario if all good clocks are at opposite ends of the precision window, and Byzantine clock is seen at different corners by two nodes.


j
View of node j
k
View of node k
Precision window Π
Time difference
good time value
malicously faulty time value
j Average value calculated by node j
k Average value calculated by node k

State Correction versus Rate Correction

3.4 Internal Clock Synchronization

Calculated correction value can be applied to local time immediately (state correction), or rate of clock can be modified such that the clock speeds up or slows down to bring it into better agreement with rest of ensemble (rate correction).

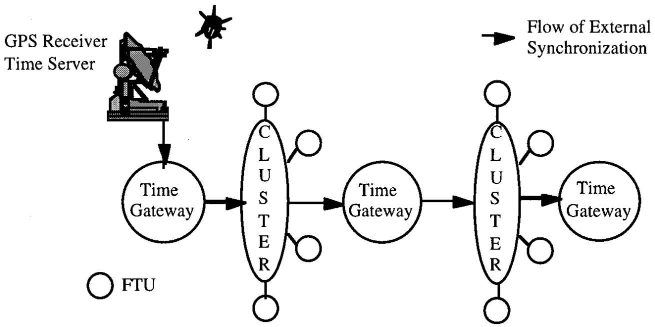
State correction leads to a discontinuity in the time base. If time is set back, the same time is reached again. This can cause nasty errors.

Rate correction is therefore preferable.

3.5 External Clock Synchronization

External synchronization links global time of a cluster to an external standard of time. A time server periodically broadcasts the current reference time via a time message.

Time scale must be based on constant measure of time (usually physical second), and must have a defined origin of time the epoch.

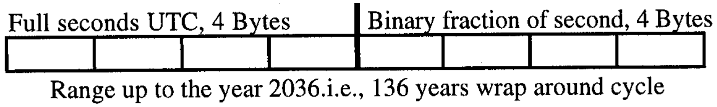


Principle of Operation

Time server periodically broadcasts time messages containing synchronization events, and information to place the events on the TAI scale. Synchronization is unidirectional; contrary to internal synchronization it is not a cooperative activity among nodes.
If time server sends an incorrect message, all subordinates will behave incorrectly. Some correction due to the inertia of time: the cluster monitors the server. A time gateway will only accept a message from the server if it is sufficiently close to its view of external time.

Time Formats

Format of the Network Time Protocol (NTP) of the internet:


Full seconds UTC, 4 Bytes
Binary fraction of second, 4 Bytes
Range up to the year 2036.i.e., 136 years wrap around cycle

Resolution is about 232 picoseconds.
NTP is not chronoscopic, since it is based on the UTC.

Time Gateway

1. It initializes the cluster with the current external time.
2. It periodically adjusts the rate of the global time in the cluster to the external time base.
3. It periodically sends the current external time to the cluster to allow new nodes to synchronize.

Points to Remember

3 Phases:

First phase: every node acquires knowledge about the state of the global time counters in all other nodes by the exchange of the messages.

Second Phase: every node analyses the collected information to detect errors and execute the convergence function to calculate a correction value of a local global time counter. -> if analysis says, sync condition nis not true, the node have to deactivate itself.

Third Phase: the local time counter of the node is adjusted by the calculated correction value.

1g:

Clock j ticks, event occurs, clock k ticks.

Global time stamp for event can differ by at most 1 tick.

2g:

At most 2g, to understand the temporal order of two events between two nodes.

3g:

Two events, and when a third node observes the events, then we need atleast 3g.

4g:

In sparse time base, when there are two clusters.

If cluster A generates a 1/3g precedent event set, then it is possible that two events that are generated at the same cluster-wide granule at cluster A will be time-stamped by cluster B with time-stamps that differ by 2 ticks. The observing cluster B should not order these events, because they have been generated at the same cluster-wide granule. Events that are generated by cluster A at different cluster wide granules (3g part) and therefore should be ordered by cluster B, could also obtain timestamps that differ by 2 ticks. Cluster B cannot decide whether or not to order events with a time stamp difference of 2 ticks. Cluster B cannot decide whether or not to order events with a time stamp difference of 2 ticks. To resolve this situation, cluster A must generate a 1/4g precedent set.

Example:

Either telephone ringing or bus arriving at the same time OR

If difficult, then atleast 3g apart in order to determine the temporal order.

Points to Remember