TIME, CLOCKS AND CLOCKS SYSTEMS in REAL-TIME SYSTEMS

References [Kri974], [Nis97]

TIME and its REPRESENTATION
- **Time**
- **Real-time**
- **Clock time**
- **Ideal clock**
- **Real Clock**

EVENTS, STATES and INTERVALS
- **Intervals in Real Time**
- **Intervals in Clock Time**
- **Events, States and Temporal Intervals**

EVENT ORDERING
- **Casual (or happened-before or partially) ordering**
- **Global (or total) ordering**
- **Sync (or casual-global) ordering**

CLOCKS in REAL TIME SYSTEMS
- **Physical Clock**
- **Logical clock (or Lamport's clock)**
- **Vector clock**

TIME and its REPRESENTATION

Time

types of time:

1. metric, consisting of a set of time values, for measurement of time

2. continuously evolving process for expressing temporal references to events

set of time values:

1. dense set of time values

- **set has an ordering so that between any two elements in the set there is always a third element**
- **dense set is used in systems where the times of occurrence of any pair of events can be arbitrarily close to each other**

example: **rational and real numbers**

2. sparse set of time values

example: **natural numbers and integers**

Real-time

- objective or real time for expressing an ordered set of all definite and distinct events in the time space

Θ* - variable for the real-time = set of time values /non-negative real numbers/

properties of the real-time

- **dense**
- **totally ordered by relations**
- **uncountable**

- current time is a marker that represents current time continuously taking values of the real-time or the appropriate concrete representation

f(Θ*) - variable for the current time

properties of the current time

- **no values of current time are ever repeated i.e. it never assumes a value it has assumed before and always assumes a greater value than it previous values**
- **nobody controls the current time and nobody observes its progress directly**

Clock time

- concrete or clock time is for representation of time using clocks that contains discretized time values

***T* - variable for the clock time**

properties of the clock time

- **totally ordered by relations**
- **it may be controlled and observed**
- **values of the clock time may be repeatable in rudimentary representations**

granularity

1. Δ - actual or target granularity of the real time

- **a length of time between two elements in Θ* defining the actual or desired separation in real time between pairs of consecutive values in *T***

2. ∇ - calibrated granularity of the clock time

- **∇ is unit for measuring time in *T*, where a given clock measures the distances between pairs of elements in *T* in integral multiples of some ∇**

- clock time is described by the following four variables:

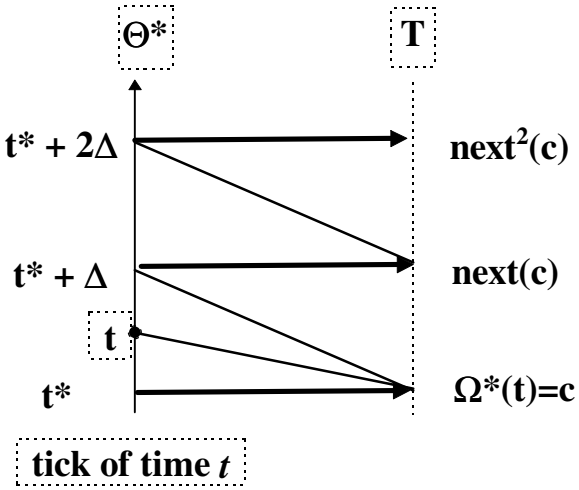
(*T, 0, next,* ∇)

functions of the clock time:

- **1. *next(t)***
  - **one-to-one successor function on rational number *N* that returns the value immediately before *t* in *T***

2. ***prev(t)***

- **inverse function of the *next* function**



3. ***measure(T -> N)***

where **for any *c* ∋ *T***

measure(0) = 0

measure(next(c)) = 1 + measure(c)

||c1,c2|| = ∇ * |measure(c1) - measure(c2)|

Ideal clock

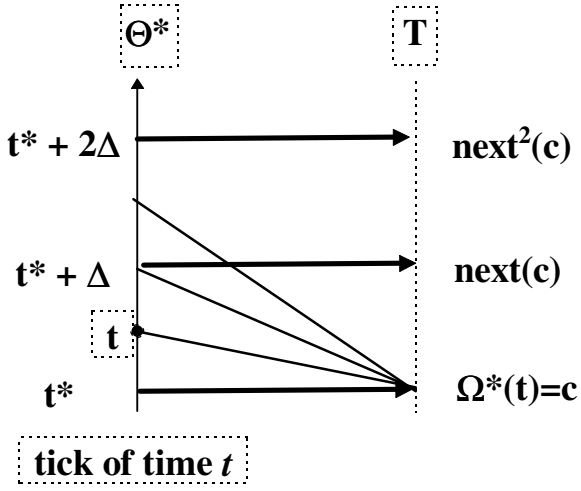
- a clock should map real time to an appropriate representation and it is described by the following function:

$\Omega^* : \Theta^* \rightarrow T$

properties of the clock

1. monotonicity

- **representation of time must progress, or at least not regress, with time**
- **representation of any two distinct time values of real-time must be in the same relation as the real time themselves**

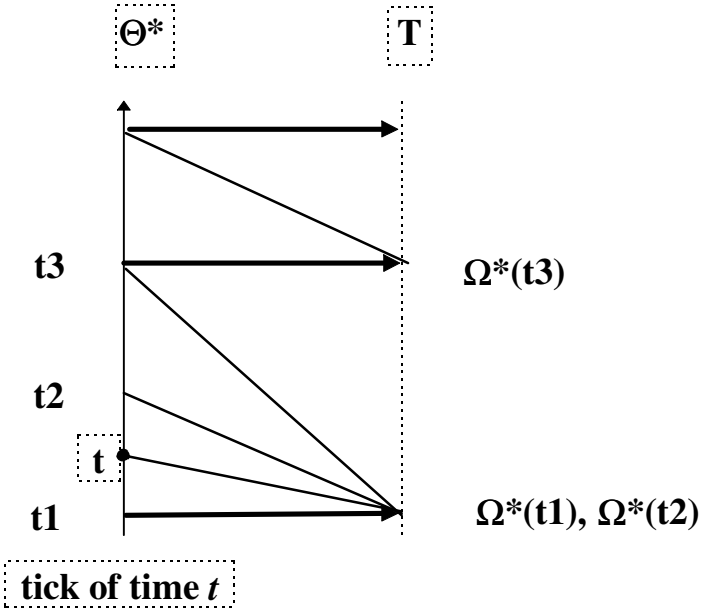


violation of monotonicity

$$\forall t_1, t_2 : \Theta^* \bullet (t_2 > t_1 \Rightarrow \Omega^*(t_2) \geq \Omega^*(t_1)) \wedge (\Omega^*(t_2) > \Omega^*(t_1) \Rightarrow t_2 > t_1)$$

2. uniformity

- **progress of temporal representation must be uniform relative to real time**



violation of uniformity

$$\forall t_1, t_2, t_3, t_4 : \Theta^* \bullet t_2 - t_1 = t_4 - t_3 \Rightarrow$$
$$\Rightarrow (||(\Omega^*(t_2), \Omega^*(t_1))|| - ||(\Omega^*(t_4), \Omega^*(t_3))||) \in \{-\nabla, 0, +\nabla\}$$

- **the calibrated granularity is the smallest non-zero measure of time that can be made in *T***

$$\exists\, t_1, t_2 : \Theta^* \bullet ||\Omega^*(t_1), \Omega^*(t_2)|| = \nabla$$

3. liveness

- **clocks must make progress at least every Δ units in real time**

$$\forall t : \Theta^* \bullet t \in \mathrm{dom}\Omega^* \Rightarrow \Omega^*(t + \Delta) > \Omega^*(t)$$

clock advancement

- **the clock works by continuous internal transitions governed by the independent progress of time**

- **these internal transitions are subject to the following real time constraints:**

  - **the clock must show the right clock representation of real time after every clock transition**

- **successive internal transitions must be separated by an arbitrarily small but non-zero interval ε of real time Θ^* where $\varepsilon << \Delta$**

- **for each clock tick there is unique contemporaneous update of display**

clock tick times

1. *t clock tick time**

- **value of real time for which is defined by the so-called clock tick function**

- **reading of the clock at real time *t* which is belongs to a discrete domain**

- **for a given real time value *t* the clock tick time is calculated as**

$$t^* = \min\{x | x \in \mathrm{dom}\Omega^* \wedge \Omega^*(x) = \Omega^*(t)\}$$

2. ω clock tick function

- **subset of the corresponding clock function**

$$w = \{(t^*, \Omega^*(t)) \bullet t : \Theta^* \mid t \in \mathrm{dom}\Omega^*\}$$

- **there is the following correspondence between the clock tick time and the clock tick function**

$$t^* = \omega^{-1}(\Omega^*(t))$$

- **there is a relationship between real time values and their tick times for every *t***

$$t^* \le t < t^* + \Delta$$

- **any event which appears to be an instantaneous event in clock time may be in fact a durative event in real time occurring in an interval smaller than the actual clock granularity**

$$\Omega(t_1) = \Omega(t_2) \Rightarrow t_1 \leq t_2 < t_1 + \Delta$$

standard time metrics

1. astronomical time metric

- **based on celestial observations that use conventional calendars**

2. external standard physical time

- **based on devices exploiting periodicity for example crystal oscillators or subatomic particles**
- **it has a much finer granularity and accuracy than most of the clocks thus,**
- **it should be corrected from time to time to keep in phase with astronomical time**

Real Clock

- there is no way of establishing the agreement between any two clocks as a result of the communication delays during the synchronisation, thus absolute correctness in time cannot be achieved

- problems in real clock:

- **excessive /negative of positive/ drifts,**
- **erratic display,**

- correct /"good"/ clock

$$1-\rho \le \frac{\delta_{ms}}{\delta_{sp}} \le 1+\rho$$

where **ρ - permitted deviation of clock rate from external reference time**

δms , δms- measured and specified difference in clock reading over a given interval of time

$$1-\rho \le \frac{\Delta}{\nabla} * \frac{||(\Omega(t_1),\Omega(t_2))||}{t_2 - t_1} \le 1+\rho$$

where **Ω - clock function to account features of the real clock and to show the external reference time**

- it uses the external standard physical time which is the most superior discretised time metric available

Δ/∇ - relationship between the actual granularity Δ in Θ and calibrated granularity ∇ in T

- conversion rate for relating time measured in T to external reference time Θ that must ideally close to unity when expressed in a single unit of measurement

- Δ varies with real time and clock rates over longer periods of time may be related to Δ/∇ by using an average value of Δ

$\Delta/\nabla<1$ - designates fast clocks

$\Delta/\nabla>1$ - designates slow clocks

EVENTS, STATES and INTERVALS

- actions, events are occurring in real-time and states, assigned to objects are changed by actions and events in real-time,

- there is an ordering among actions and events that describe their sequence in real-time,

- representation of time may contain points and intervals of time in real-time systems,

<u>problems</u> with point of time in real-time system:

1. instanteneity -> atomic event, but in real-time system are no atomic event

2. actions and events change states of objects, and there could be inconsistency in both understanding and describing real-time systems using point of time

<u>solution:</u> using interval of time instead point of time in real-time systems

- there are two types of interval:

1. bounded intervals:

- **open interval**

  (t_1, t_2) - contains all time values from t_1 to t_2 excluding of both t_1 and t_2

- **right-open /left/closed/ interval**

  $[t_1, t_2)$ - contains all time values from t_1 to t_2 including t_1 but excluding t_2

- **left-open /right/closed/ interval**

  $(t_1, t_2]$ - contains all time values from t_1 to t_2 including t_2 but excluding t_1

- **closed interval**

$[t_1, t_2]$ - contains all time values from t_1 to t_2 inclusively

2. unbounded intervals:

- **left-open unbounded interval**

  (t, ∞) - contains all time values from t to ∞ but excluding t

- **unbounded interval**

  $[t, \infty)$ - contains all time values from t to ∞ including t

Intervals in Real Time

- in real-time systems the right-open interval is used to describe actions and events

- the right-open interval is given by the following expression:

$$\Gamma = P(P\ R)$$

where Γ **- set of all possible right-open intervals,**

P - power set of the set concerned,

$$\Gamma = \{[\ t_1, t_2) \bullet t_1, t_2 : R$$

$$[\ t_1, t_2) = \{t \bullet t : R \mid t_1, \leq t < t_2\ \}$$

where $t_1 < t_2$

- evolution of a system should be given by subsets of intervals for example right-open intervals, where these subsets are totally disjoint intervals of time,

- these subsets are associated with states and events in order to specify intervals over which a given state obtains or a given event occurs,

$$O \subset P\ \Gamma$$

$$O = \{I{:}\Gamma \mid \forall x, y \in \Gamma \bullet x \neq y \wedge x \in I \wedge y \in I \Rightarrow cl\ x \cap cl\ y = \varnothing$$

where **O - set of subsets describing states and events**

I **- a right-open interval**

cl **- function giving the closed version of any fully or semi-open interval**

- termination of desirable events in real-time systems is described by the following subset:

$$O_f = \{I \in O \| card(I) \in N$$

where O_f - each element of this subset consists of a finite number of disjoint rigth-open intervals

card(I) - cardinality of *I*

N set of natural numbers

relation between intervals in real-time

- **an infinite number of events may occur in any interval, sequence of events may be described by relationship between intervals /temporal relationships/**

- **temporal relationships for *I, I'* :** Γ

I	**before**	***I'***	**right(*I*) < left(*I'*)**
I	**equal**	***I'***	***I* = *I'***
I	**meets**	***I'***	**right(*I*) = left(*I'*)**
I	**overlaps**	***I'***	**left(*I*) < left(*I'*) < right(*I*) < right(*I'*) v**
			left(*I'*) < left(*I*) < right(*I'*) < right(*I*)
I	**during**	***I'***	**left(*I'*) < left(*I*) $\wedge$ right(*I*) < right(*I'*)**
I	**starts**	***I'***	**left(*I'*) = left(*I*) $\wedge$ right(*I*) < right(*I'*)**
I	**finishes**	***I'***	**right(*I*) = right(*I'*) $\wedge$ left(*I'*) < left(*I*)**

Intervals in Clock Time

- for any time value within the right-open interval of real-time, its clock representation would line in the corresponding closed interval of concrete time

- an interval *I* in real time is measured as an interval I_c in a given clock representation,

$$\mathbf{I_c = \Omega^*(I)}$$

- relation between *I* and I_c to the end points is given by the following expressions:

$$\mathbf{t \in [t_1, t_2) \quad \Leftrightarrow t_1 \le t \wedge t < t_2 \Rightarrow}$$
$$\mathbf{\Omega^*(t) \in [\Omega^*(t_1), \Omega^*(t_2)) \Leftrightarrow \Omega^*(t_1) \le \Omega^*(t) \wedge \Omega^*(t) \le \Omega^*(t_2)}$$

- clock representation of the right-open interval of real time is equal to:

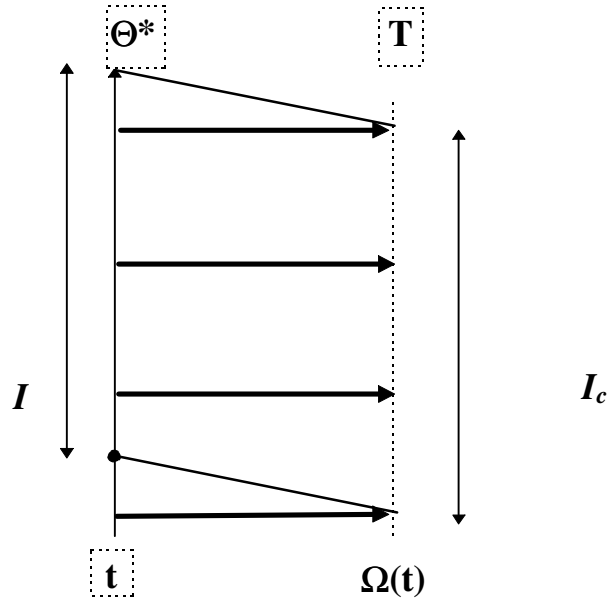
$$\mathbf{t \in [t_1, t_2) \Rightarrow \Omega^*(t) \in [\Omega^*(t_1), \Omega^*(t_2) \quad \text{if } t_2 \in dom\omega}$$

$$\mathbf{t \in [t_1, t_2) \Rightarrow \Omega^*(t) \in [\Omega^*(t_1), \Omega^*(t_2 + \Delta)) \quad \text{if } t_2 \notin dom\omega}$$

- the interval in clock time corresponding to any right-open interval in real-time may be given either as right-open or as closed interval

- if a particular clock time is within a certain interval of clock time then the corresponding real-time would lie in a right-open interval of real-time $[t_1, t_2)$ using the appropriate clock tick times corresponding to t_1 and t_2 in real-time

$$\mathbf{\Omega^*(t) \in [\Omega^*(t_1), \Omega^*(t_2) \Rightarrow t \in [t^*_1, t^*_2)}$$



relation between intervals in clock time

- **all relations but *meet*$_c$ relation are analogous to relations of interval in real time**

- **temporal relationship meet$_c$ for I_c, I_c' : Γ_c**

  I_c meets I_c' right(I_c) = left(I_c') v prev(right(I_c))=left(I_c')

connection between temporal relations of real time and clock time

- **temporal relations in real time and in clock time are identical except meet relation but they do have the same meaning for a given pair of intervals**

$$\forall I{:}\Gamma; I_c{:}\Gamma_c \bullet I \approx I_c \Leftrightarrow I_c = \Omega^*(I)$$

where ≈ relation leading from real time domain to clock time domain

Events, States and Temporal Intervals

state

- **characterises the condition of a system at a given instant in time or over a given interval of time,**
- **an abstraction of reality in which some of its selected features relevant to a given stage in remain unchanged over the interval of time concerned,**
- **is obtained at a moment or over a range of moments**
- **is homogenous (if a state obtained during an interval then the same state obtains at every moment in that interval) and dissective**

description of states

- **state may be described by a proposition that may describe all kinds of states**
- **temporal indexing of propositions leads to chronological logic of states**

event

- **event is either execution of a procedure or exchange of a message i.e. receiving or sending a message,**
- **characterises significant changes in a state of an object that may include several subevents but if an event is split up into subevents, then each subevent is not the same as the original events**
- **occurs over an interval which may be infinitesimal**

classes of events:

1. durative events

♦ **take time to occur and their advance may be described by an appropriate proposition**

2. punctual events

♦ **instantaneous events that cannot be caught in the act of happening**

3. momentary events

♦ **instantaneous events that have a proposition that describes a state for the instant when the event occurs i.e. it can be caught in the act of happening**

4. telic and atelic events

♦ **telic events are events which must complete in order for it have occurred**

♦ **atelic events are events without notion of completion**

5. once-only events

♦ **events that have only a unique occurrence or no occurrence**

6. repeated events

♦ **events that occur more than one time**

action

- **events performed by agents**

EVENT ORDERING

relation between events:

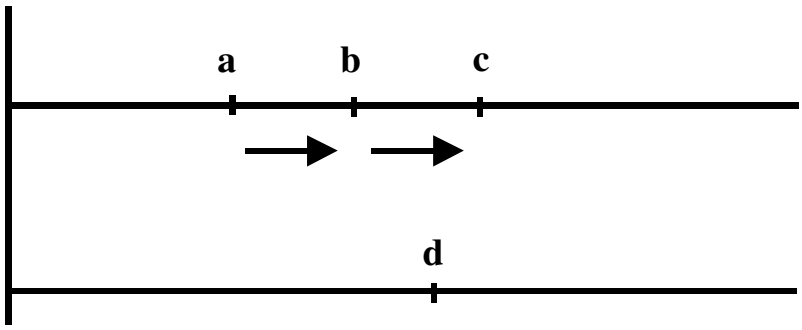
a. casually related events

- **event *a* causally affects event *b* if $a \rightarrow b$**

b. concurrent events

- **concurrent events *c, d* do not causally affect each other, i.e. $c \| d$ if $c \not\rightarrow d$ or $d \not\rightarrow c$**

- **concurrent events occur on processes that do not communicate with each other, thus nothing can be said about when they happened or which was first,**



problems:

- **events are ordered according to times shown on the local clock but local clocks cannot be perfectly synchronised in a distributed system,**

- **order of events occurring at different processes can be critical in a distributed system**

solutions:

- **application of event ordering approaches**

Casual (or happened-before or partially) ordering

- the relation captures the casual dependencies between events, i.e. whether two events are casually related or not,

- an event influences the system state, thus it affects the occurrence and outcome of future events,

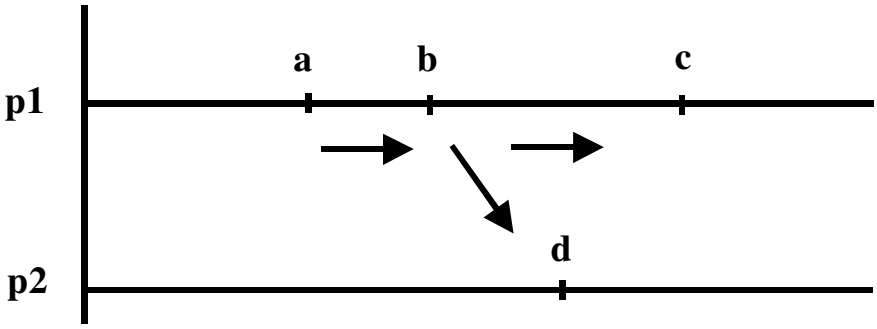
- the happened-before relation captures the casual dependencies between events, i.e. whether two event are casually related or not,

assumptions:

a. if two events occurred at the same process, then they occurred in order in which it observes them,

b. if the happened-before relation is true for two events, then the first might or might not actually caused the second because the happened-before relation describes only potential casuality, thus two events can be related by this relationship even though there is no real connection between them,

c. if a message is sent between two processes then event of sending the message happened before the event receiving the message,



$a \rightarrow b \rightarrow c$ **$a \rightarrow b \rightarrow d$**

rules of the relationship

a. if exists process p_i: $a \rightarrow^i b$, then $a \rightarrow b$

where ***a, b*** **- events** p_i **- process**

b. for any message *m*: a(*m*) → b(*m*)

where ***a*** **– event sending message *m***

b **– event receiving message *m***

remark: message cannot be received before it is sent or at the same time it is sent because the communication takes a finite amount of time

c. it is a transitive relation i.e. if *x, y, z* are events such that $x \rightarrow y$ and $y \rightarrow z$, then $x \rightarrow z$

problems:

- **at any time, a process executing in one node can have only approximate knowledge of the state of any other process in another node,**
- **even in the case when for any two nodes *i* and *j,* the order in which *i* sends messages to *j* is identical to the order in which *j* receives them from *i,* any two observable events in the system can be perceived as occurring in a different order by different nodes,**

Global (or total) ordering

problem:

- **no two events ever occur at exactly the same time thus, it is required to define the total order of events in DSs in order to avoid ambiguities,**
- **to determine that an event *a* happened before an event *b*, global time must be available, but distributed systems do not have common clock or synchronised clock,**

assumptions:

- **the casual ordering is only partial order on the set of all events, since some pairs of distinct events, generated by different processes, may have numerically identical timestamps,**

solution:

- **to use logical time instead of global time, i.e. a clock C_i assigns a number to an event where the number is the relative time at which the event occurred,**

rules of the relationship:

a. if *a* and *b* is an event occurring at process p_i and $a \rightarrow b$

then $C_i(a) < C_i(b)$

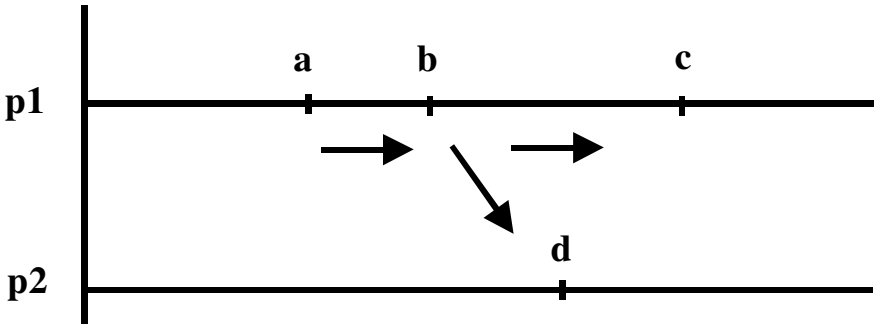
b. if *a* is an event occurring at process p_i then its clock C_i assigns the value $C_i(a)$ to it,

and *b* is an event occurring at process p_j then its clock C_j assigns the value $C_j(b)$ to it,

c. event *a* and *b* are totally ordered, i.e. $a \Rightarrow b$ if and only if

either $C_i(a) < C_j(b)$

or $C_i(a) = C_j(b)$ and $a \prec b$



a → b → c **a →b → d**

Sync (or casual-global) ordering

assumptions:

- **there are systems with some events that are casually ordered and other events that are globally ordered,**
- **casually ordered and globally ordered events can be processed in an arbitrary order, unless they are casually related,**
- **sync ordering first accepts those events that have been occurred but not yet handled in the DSs, so that they are processed before it, all later requests are processed after them,**
- **thus there is division of handling of events into past and future events,**

CLOCKS in REAL TIME SYSTEMS

types of clock in real-time systems:

1. physical clock

- **actual physical device that measures time**

2. logical clock

- **is software implementation of the physical clock,**
- **enables the reading of the physical clock with a predefined offset to counter against any discrepancy in time**

3. fault tolerant clock

- **high level abstraction of a system of clocks that is based on distributed architectures**
- **designed to function normally despite any possible clock failures**

time metrics in distributed real-time systems:

1. internal physical time

- **approximation of a processor to the external reference time**

2. local /physical/ time

- **reading of the local physical clock**

3. logical time

- **adjusted value of the physical clock reading that is used as the local time base**

4. approximate global time

- **approximation to a common time base among the processors constituting the distributed system achieved by internal synchronisation of time**

5. external reference time

- **external standard physical time that is the final reference for reasoning about the time**

Physical Clock

- actual physical device measuring time

International Atomic Time = TAI

- **produced by physical clock based on atomic oscillators**

  problem: 86400 TAI seconds is about 3 msec less than a mean astronomical day

  solution: adding leap second whenever the discrepancy between TAI and solar time grows to 800 msec

Coordinated Universal Time = UTC

- **international standard based on TAI but leap second is inserted or deleted to keep with astronomical time**
- **a shortwawe radio station WWV broadcasts a short pulse at the start of each UTC second,**
- **if a computer has a WWV receiver, all other computers in the computer system should be synchronised to it,**

clock in computer

timer = hw + sw to keep track of time that is based on hw clock interrupts keep track of sw activities

hw = quartz crystal that oscillates

sw = counter register is decremented by an interrupt that is generated by each oscillation of the crystal *n* times a second

interrupt = clock tick

each time when the counter gets to zero and interrupt is generated that increments the time register which is a sw clock C

ideal clock in computer

$C_p(t) = t$ $\qquad dC/dt = 1$

where **t** **- UTC time**

p **- computer**

C **- computer clock**

clock drift

- **the crystal-based clocks count time at different rates due to physical variations of crystals**

fast clock $\Delta/\nabla<1$ **or** $dC/dt >$

slow clock $\Delta/\nabla>1$ **or** $dC/dt < 1$

clock within a specification

$$1-\rho \le \frac{\Delta}{\nabla} * \frac{\|(\Omega(t_1),\Omega(t_2))\|}{t_2 - t_1} \le 1+\rho$$

$1 - \rho \le dC/dt \le 1 + \rho$

compensating clock drift

1. the time provided by UTC is greater than the time at a computer

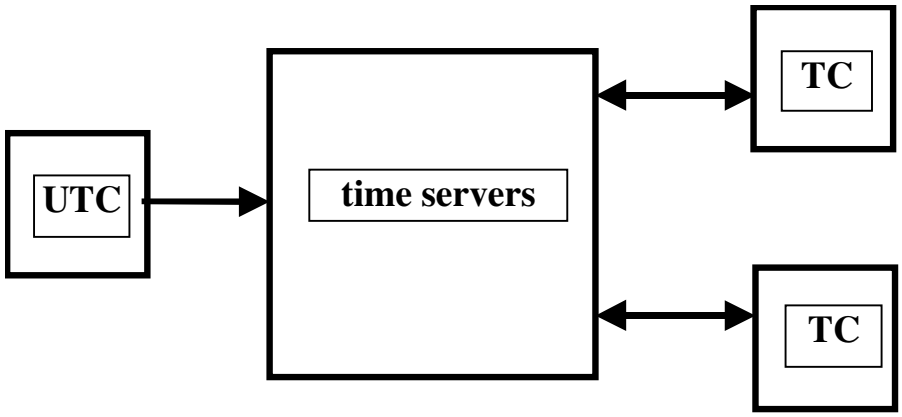
- **computer clock is set to the time provided by UTC**

2. the time provided by UTC is less than the time at a computer

- **computer clock is run slow until it is equal to the time provided by UTC**

- to provide the time for processes, a distributed time service architecture is applied to DSs that includes time clerks and time servers

- there is a time clerk (TC) on each client computer that may request time service from a time server (TS) that maintain up-to-date clock values and may have access to a universal real-time source UTC



time server architecture

- there is a three types of time access: time server access to the time; client access to time server; time server access to time server

- time servers access UTC sources provided by NIST (National Institute of Standards and Technology) at the accuracy near 1 msec by ACTS (Automated Computer Time Service) using modem service to NIST time through telephone lines or short wave radio station WWV, which regularly broadcasts UTC signals, i.e. a short pulse at the start of each UTC second, or Global Positioning System GSP

- there are two practical issues in implementing the time service:

1. compensating time delays

- **there is signal propagation and network communication delay where network delay may change and is several orders of magnitude higher than propagation delay,**

2. calibrating discrepancy

- **time servers may exchange UTCs with each other to attempt to reach a consensus on time by adjusting the UTC using previously agreed decision criteria**

Logical clock (or Lamport's clock)

- physical clocks tell whether an event happens before another unless they occur very closely,

- if the uncertainty of UTC is high or UTC intervals of events overlap with each other, it is not possible to determine the ordering of events,

- in many applications, events need not be scheduled or synchronised with respect to the real-time clock, it is only the ordering of execution of events that is concern,

problem:

- **it is impossible perfectly to synchronise clocks across a distributed system because timing information cannot be propagated and received instantaneously, thus an absolute global physical time is theoretically impossible to obtain in DSs and RTs**

solution:

- **in many systems computer may agree on the same time that may not be equal to the real time,**

- **the execution of a process is characterised by a sequence of events, for example execution of a procedure, and exchange of messages i.e. receiving and sending of messages, are events,**

- **assigning a number - timestamp - to an event where the number designates the relative time at which the event occurred, but these numbers have no relation to physical time,**

timestamp **= is a value of the logical clock for that event when it occurs, i.e.** $TS_i(a)$

logical clock **= there is a logical clock of each process** p_i **represented by a function** LC_i **which assigns a number or a timestamp** $TS_i(a)$ **to any event** a **at process** p_i

$$LC_i(a) = TS_i(a)$$

- **logical clock is a monotonically increasing sw counter, whose value may not have any particular relationship to any physical clock,**

- **the logical clock of a process is incremented by an arbitrary positive number when an event assigned to the process,**

- **each message *m* must be stamped with a timestamp *TS* which equals the sender logical clock at which the message was sent,**

- **upon receiving a message with a timestamp *TS*, a process compares its time with the attached to the message clock value,**

happened before conditions of the logical clock:

a. if *a* and *b* is an event occurring at process p_i and $a \rightarrow b$

then $TS_i(a) < TS_i(b)$

b. if *a* is an event sending a message *m* by process p_i with local timestamp $TS_i(a)$ and *b* is an event receiving the same message at p_j with local timestamp $TS_j(b)$,

then $TS_i(a) < TS_j(b)$

totally ordered conditions of the logical clock:

a. if *a* and *b* is an event occurring at process p_i and process p_j respectively, and $a \Rightarrow b$

then either $TS_i(a) < TS_j(b)$

or $TS_i(a) = TS_j(b)$ and $p_i \prec p_j$

remark: $\prec$ is an arbitrary relation that totally orders the processes to break ties

implementation rules of the logical clock:

a. if LC_i is incremented between any two successive events *a* and *b* in process p_i

then $LC_i(a) = LC_i(a)+d$ $(d>0)$

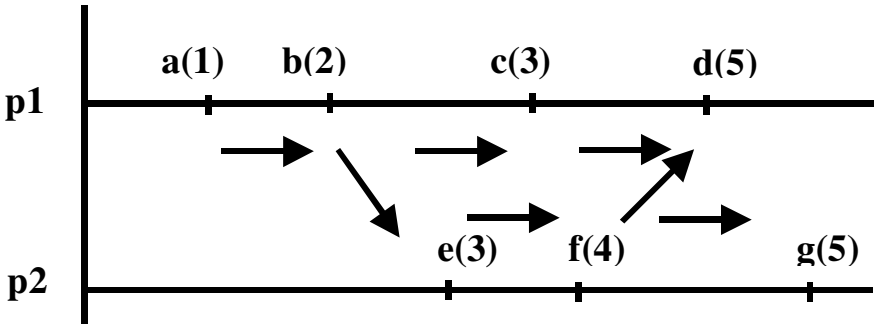
b. if *a* is an event sending a message *m* by process p_i

then a timestamp $TS_i(a)$ is assigned to a message *m*

if *b* is an event receiving the message *m* by process p_j the timestamp will be equal to a value greater than or equal to its present value and greater than $LC_i(a)$, i.e.

$$LC_j(b) = max\{LC_i(b), LC_j(a+d)\} \quad (d>0)$$

remark: ***d*** **usually has a value of one**

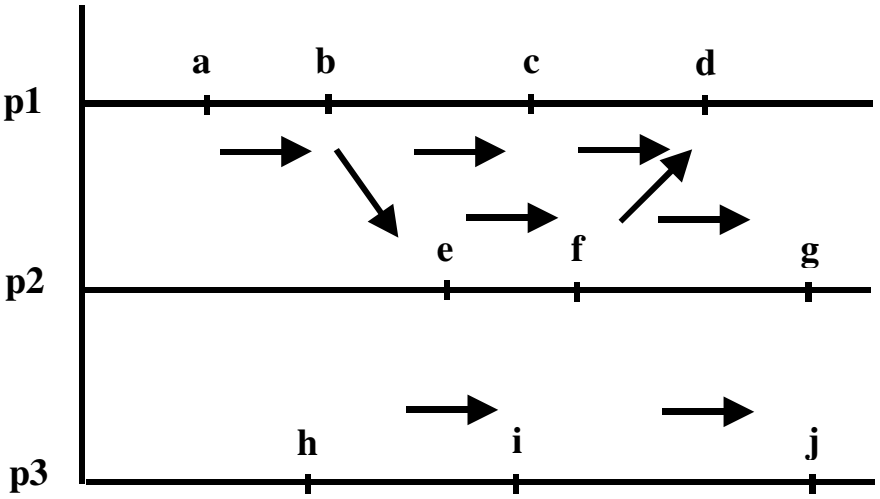


$a \rightarrow b \rightarrow c \rightarrow d$ $a \rightarrow e \rightarrow f \rightarrow d$

disadvantages of the logical clock:

- **each clock can independently advance due to the occurrence of local events in a process and a Lamport's clock system cannot distinguish between the advancements of clocks due to local events from those due to exchange of messages between processes,**

- **using timestamps, it is impossible to reason about the casual relationship between two events occurring in different processes by just looking at timestamps of the events,**


a
b
c
d
p1
e
f
g
p2
h
i
j
p3

Vector clock

- ,

problem:

- **using timestamps assigned by the logical clocks sometimes it is impossible the define casual relationship between two events occurring at different processes,**

solution:

- **there are *n* number of processes and each process p_i has a vector clock VC_i, which is an integer vector of length *n*, where *n* is number of co-operating processes,**
- **the vector clock of a process p_i VC_i is a function that assigns a vector $VC_i(a)=\{ TS_1, TS_2,\ldots TS_i[a],\ldots TS_n\}$ to any event *a*, where $TS_i[a]$ is the timestamp of event *a* at process p_i**
- **$VC_i[i]$ corresponds to process p_i own logical time and $VC_i[j]$ if $i \neq j$ is the best estimate value of the logical time at process p_j**
- **the best estimate values are obtained through the timestamps included in messages,**

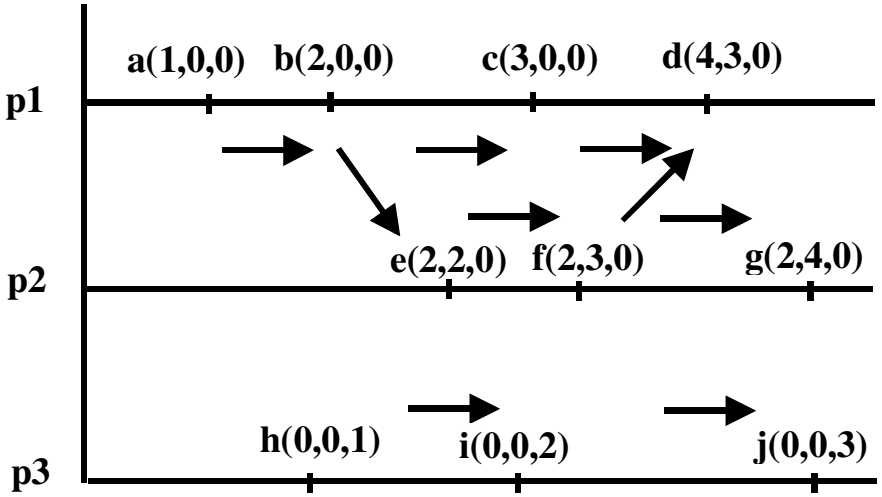
implementation rules of the vector clock:

a. if VC_i is incremented between any two successive events *a* and *b* in process p_i

then $VC_i(a) = VC_i(a)+d$ $(d>0)$

b. if *a* is an event sending a message *m* by process p_i

then a timestamp $TS_i(a)$ is assigned to a message *m*

if *b* is an event receiving the message *m* from a process p_i then the vector clock $VC_j(b)$ will be equal to a pair-wise maximum of $TS_i(a)$ and $TS_j(b)$ i.e.

$$VC_j(b) = max\{TS_i(a), TS_j(b)\}$$