Time and Order

Outline

- ♦ Time and Clocks
- ♦ Time Measurement
- ♦ Dense Time versus Sparse Time
- ♦ Internal Clock Synchronization
- ♦ External Clock Syncrhonization

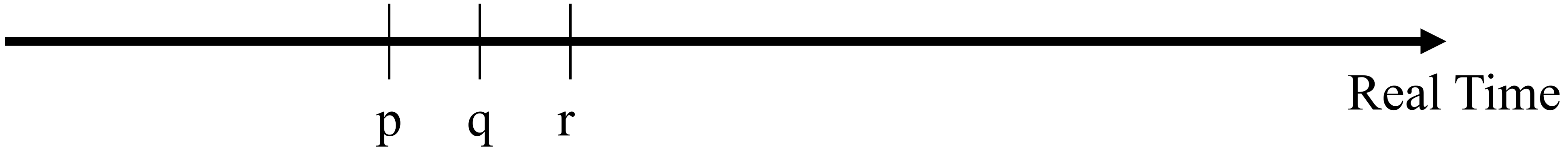
Instants are Temporally Ordered

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

(i) {T} is an ordered set, i.e., if p and q are any two instants, then either (1) p is simultaneous with q or (2) p precedes q or (3) q precedes p and these relations are mutually exclusive. We call the order of instants on the timeline the *temporal order*.

(ii) {T} is a dense set. This means that, if p≠r, there is at least one q between p and r.

The order of instants on the timeline is called the *temporal order*.



Durations and Events

A section of the time line is called a *duration*.

An *event* is a happening at an instant of time.

An event does not have a duration. If two events occur at an identical instant, then the two events are said to occur simultaneously. Instants are totally ordered; however, events are only partially ordered, since simultaneous events are not in the order relation. Events can be totally ordered if another criterion is introduced to order events that occur simultaneously, e.g., in a distributed computer system the node numbers where the events occurred can be used to order events that occur simultaneously at different nodes.

Causal Order

Reichenbach [Rei57,p.145] defined *causality* by a mark method without reference to time: "If event e1 is a cause of event e2, then a small variation (a mark) in e1 is associated with small variation in e2, whereas small variations in e2 are not necessarily associated with small variations in e1."

Example: Suppose there are two events e1 and e2:

e1 Somebody enters a room.

e2 The telephone starts to ring.

Consider the following two cases

(i) e2 occurs after e1

(ii) e1 occurs after e2

Alarm Analysis

A *primary alarm event* leads to a shower of *secondary alarm events* (*alarm shower*).

If the (partial) temporal order between alarm events has been established, it is possible to exclude an alarm event that *definitely occurred later* than other alarm events from being the primary event. A precise global time-base helps to determine the event set that is in this *definitely-occurred-later* relation.

Delivery order of messages in computer networks:

Temporal?
Causal?
Consistent?

Clocks and Timestamps

A *clock* is a device that contains a counter and increments this counter periodically according to some law of physics (*microticks*).

Let us assume there exists an external observer with an atomic clock that has a granularity that is much smaller than the duration of any intervals of interest. We call such a clock a *reference clock z* -- Precision, e.g., one femto second (10^{-15}sec)!

The *granularity* of a clock c is the number of microticks of the reference clock between any two consecutive microticks of c.

Given a clock and an event, a *timestamp* of the event is the state of clock immediately after the event occurrence, denoted by *clock (event*).

We assume that relativistic effects can be neglected.

Clock Drift

Clock Drift:

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

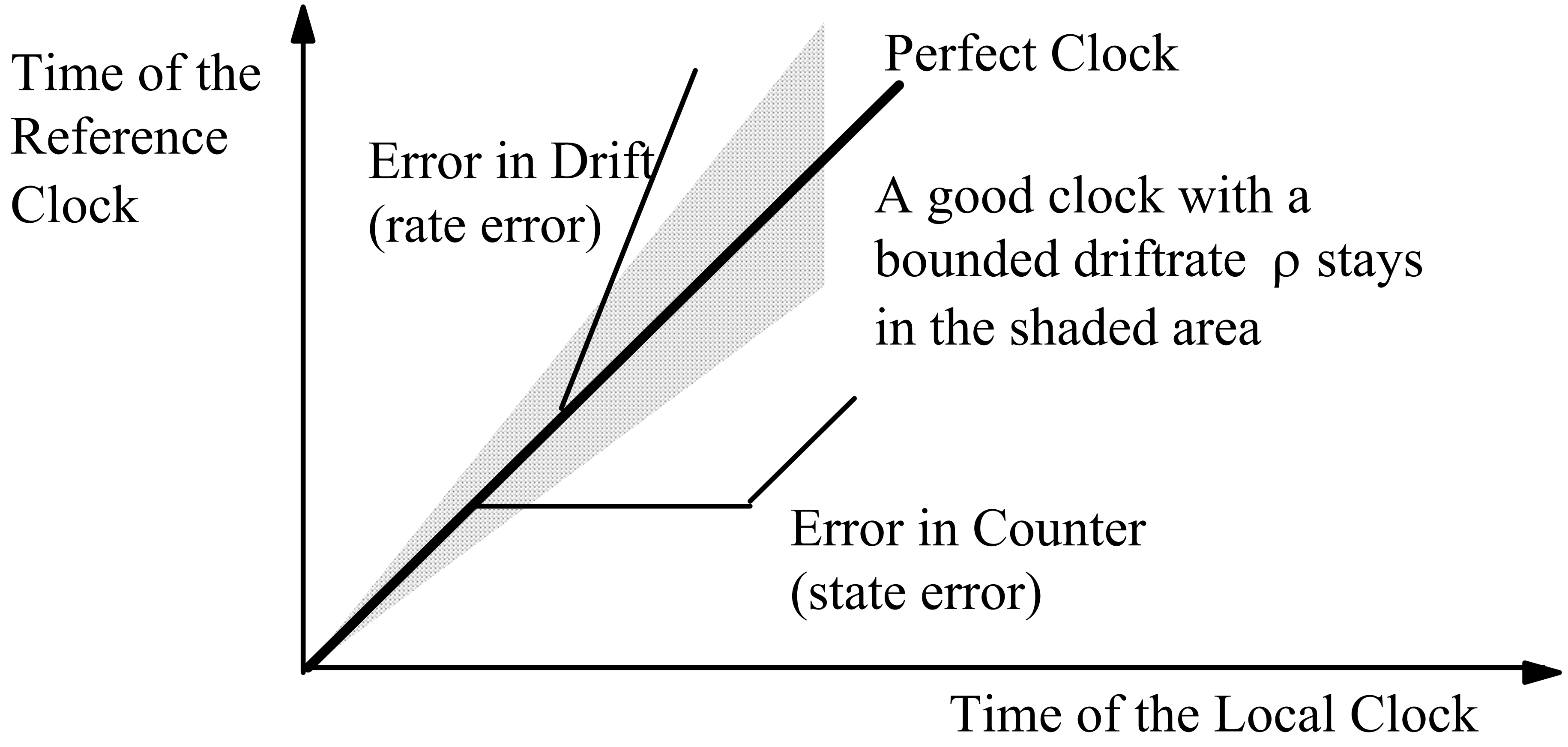
Drift Rate:

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

Perfect clock has drift rate of 0
Real clocks have drift rates from 10^{-2} to 10^{-8}

n^k nominal number of ticks of the reference clock within a granule of clock k.

Failure Modes of a Clock


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

Precision

Offset between two clocks j,k at tick i:

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

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

$$\Pi_i = \underset{\forall 1 \le j,k \le n}{Max} \{offset_i^{jk}\}$$

The process of mutual resynchronization of an ensemble of clocks in order to maintain a bounded precision is called *internal synchronization.*

Accuracy

The offset of clock k with respect to the reference clock z at tick i is called the *Accuracy*. The maximum offset over all ticks i that are of interest is called the accuracy of clock k. The accuracy denotes the maximum offset of a given clock from the external time reference during the time interval of interest.

This process of resynchronization of a clock with the reference clock is called *external synchronization*.

If all clocks of an ensemble are externally synchronized with an accuracy A, then the ensemble is also internally synchronized with a precision of at most 2A. The opposite is not true.

Time Standards

International Atomic Time (TAI): TAI is a physical time standard that defines the second as the duration of 9 192 631 770 periods of the radiation of a specified transition of the cesium atom 133. TAI is a chronoscopic timescale, i.e., a timescale without any discontinuities. It defines the epoch, the origin of time measurement, as January 1, 1958 at 00:00:00 hours, and continuously increases the counter as time progresses.

Universal Time Coordinated (UTC): UTC is an astronomical time standard that is the basis for the time on the "wall clock". In 1972 it was internationally agreed that the duration of the second should conform to the TAI standard, but that the number of seconds in an hour will have to be occasionally modified by inserting a leap second into UTC to maintain synchrony between the wall-clock time and the astronomical phenomena, like day and night.

A Problem with the Leap Second

Software Engineering Notes of March 1996 (p.16) reports on a problem that occurred when a leap second was added at midnight on New Year's Eve 1995. The leap second was added, but the date inadvertently advanced to Jan. 2. The synchronization of AP radio broadcast network depends on the official time signal, and this glitch affected their operation for several hours until the problem was corrected.

Making corrections at midnight is obviously risky:

(1) The day increments to January 1, 1996, 00:00:00.

(2) You reset the clock to 23:59:59, back one second.

(3) The clock continues running.

(4) The day changes again, and it's suddenly, January 2, 1996, 00:00:00.

No wonder they had problems.

Global Time

If there is a single reference clock available, all time measurements can be performed by this single clock that acts as a common "global" time.

In a distributed system clocks in order to generate a common notion of time, a "global time" in the distributed system.

However, such a global time is an *abstract notion* that can only be approximated by the clocks in the nodes.

It is possible to select a subset of the microticks of each local clock k for the generation of the local implementation of a global notion of time. We call such a selected local microtick a *macrotick* (or a tick) of the global time.

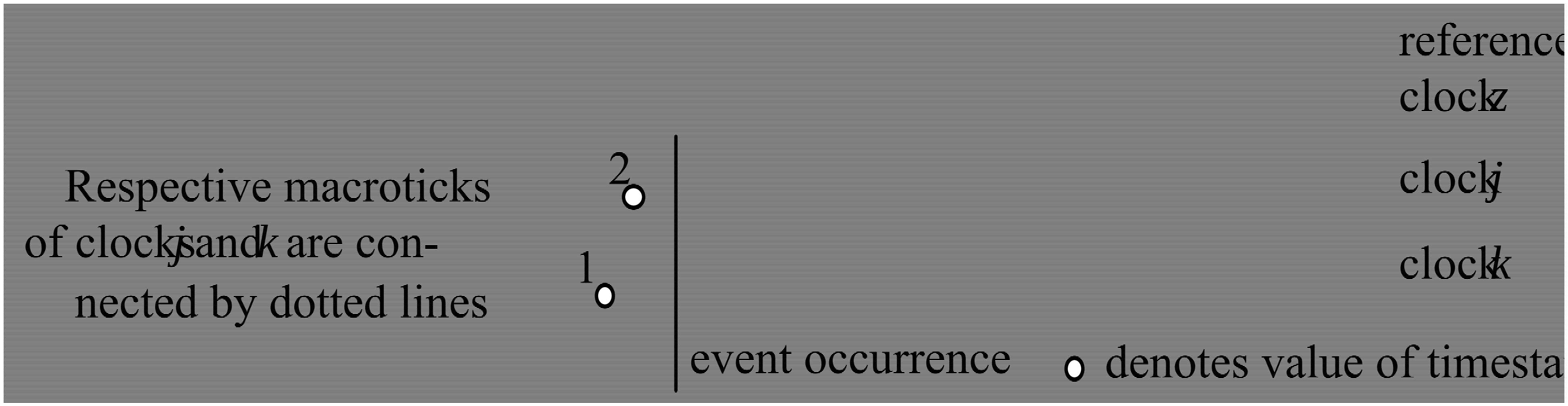
If no global time-base is available, then

- ♦ there are n independent local time references and the timestamps can only be related if they originate from the same clock.
- ♦ Interval measurements between events observed at different nodes are limited by the end-to-end delay jitter.
- ♦ the delay jitter of (ET) communication system determines the jitter in the control loops--this may be unacceptable for many real-time control applications.
- ♦ State estimation is very difficult, since the precise point in time of measurement of a process variable is not known.

Requirements of a Global Time Base

- Chronoscopic behavior, i.e. no discontinuities, even at the points of resynchronization
- Known precision
- High dependability
- Metric of the physical second

Reasonableness Condition

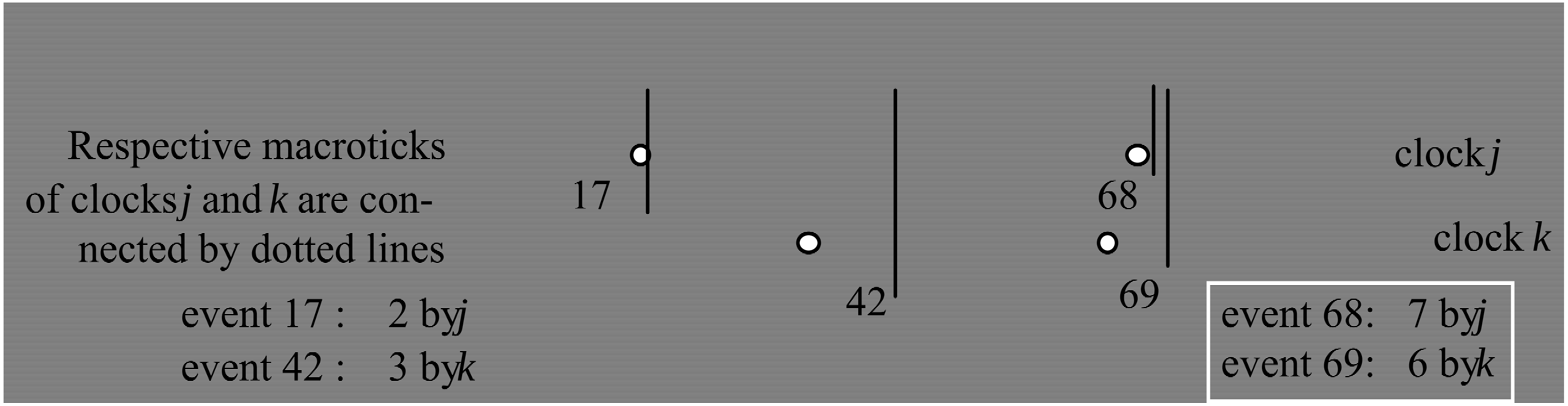


The global time t is called reasonable, if all local implementations of the global time satisfy the following reasonableness condition for the global granularity g of a macrotick:

$$g > \Pi$$

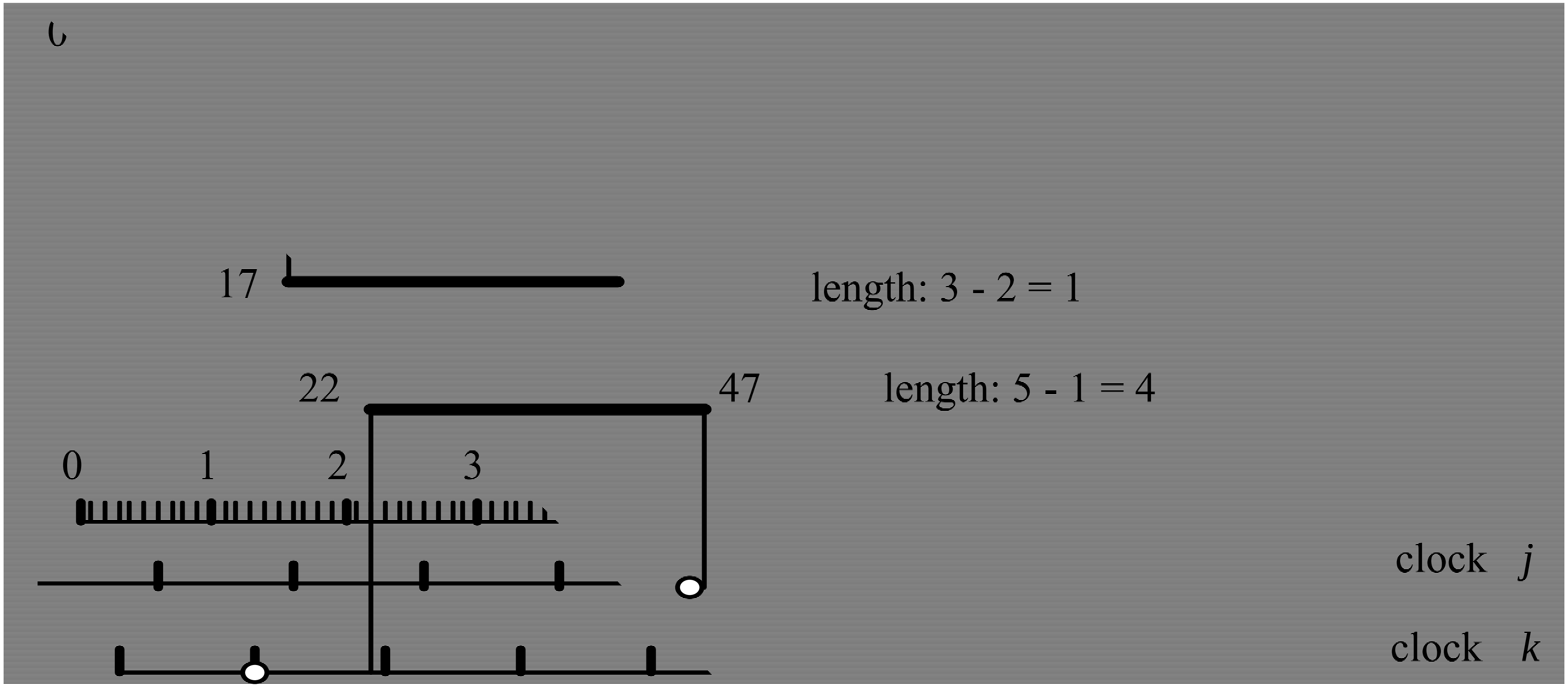
This reasonableness condition ensures that the synchronization error is bounded to less than one macrogranule, i.e., the duration between two macroticks.

One Tick Difference: What Does it Mean?



Because of the accumulation of the synchronization error and the digitalization error, it is not possible to reconstruct the temporal order of two events from the knowledge that the global timestamps differ by one.

Interval Measurement

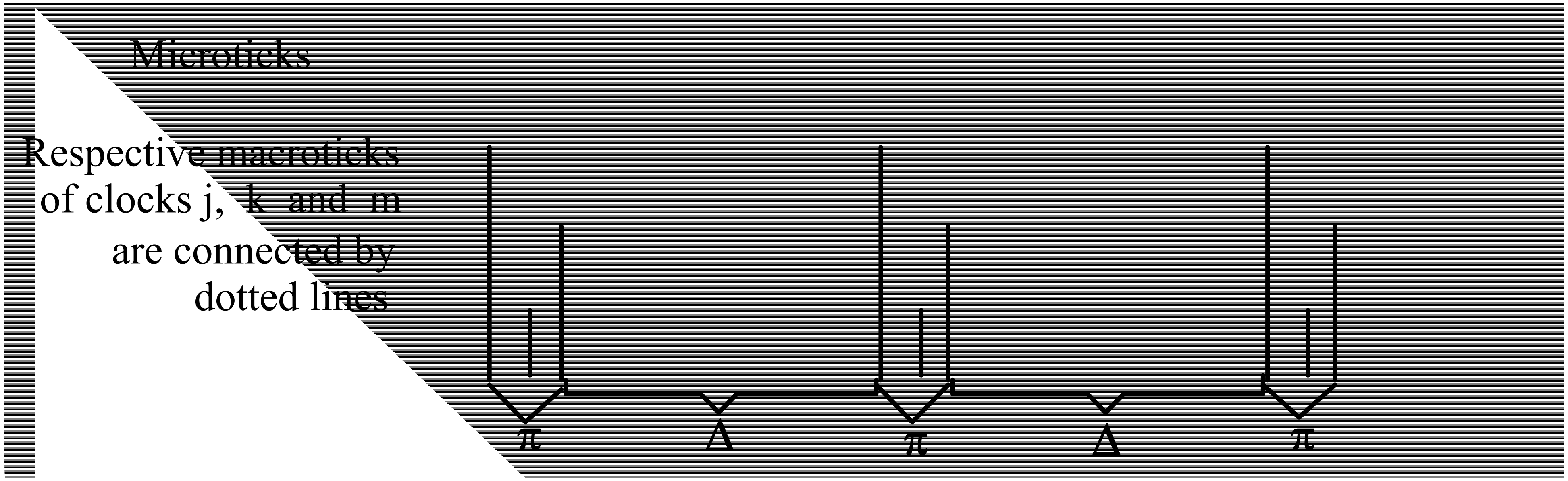


It follows: $(d_{obs} - 2g) < d_{true} < (d_{obs} + 2g)$

π/Δ Precedence

Given a set of events {E} and two durations π and Δ where π<<Δ, such that for any two elements e_i and e_j of this set the following condition holds:

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


Microticks
Respective macroticks of clocks j, k and m are connected by dotted lines
π
Δ
π
Δ
π

Fundamental Limits to Time Measurement

Given a distributed system with a reasonable global timebase with granularity g. Then the following fundamental limits to time measurement must be observed:

- ♦ If a single event is observed by two nodes, there is always the possibility that the timestamps will differ by one tick
- ♦ Let us assume that d_{obs} is the observed duration of an interval. Then the true duration d_{true} is

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

- ♦ The temporal order of events can only be recovered, if the observed time difference $d_{obs} \geq 2g$
- ♦ The temporal order of events can always be recovered, if the event set is 0/3g precedent.

Dense Time

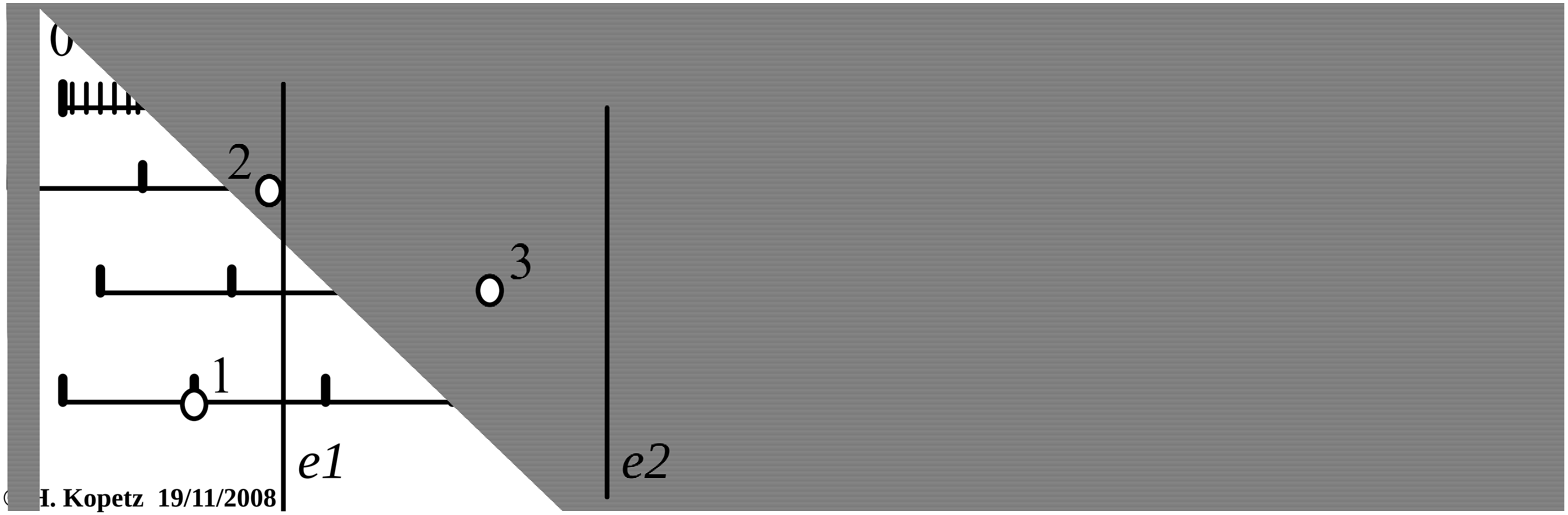
A timebase is dense if events can occur at any point of the timeline.

Consequences of these fundamental limits of time measurement in distributed systems:

- ♦ If a single event occurring on a dense timebase is observed by two nodes of the distributed system (e.g., to achieve redundancy in the observations), then an explicit agreement protocol is needed to establish a consistent view of the temporal point of event occurrence.
- ♦ If two events occur on a dense timebase, then it is impossible to always recover the temporal order of the events if they occur within an interval of 3g.

Inconsistent Order on Dense Time Base

Event e1 is observed by node j at time 2 and by node m at time 1, while e2 is only observed by node k that reports its observation "e2 occurred at 3" to node j and node m. Node j calculates a timestamp difference of one and concludes that the events occurred at about the same time and cannot be ordered. Node m calculates a timestamp difference of 2 and concludes that e1 has definitely occurred before e2.



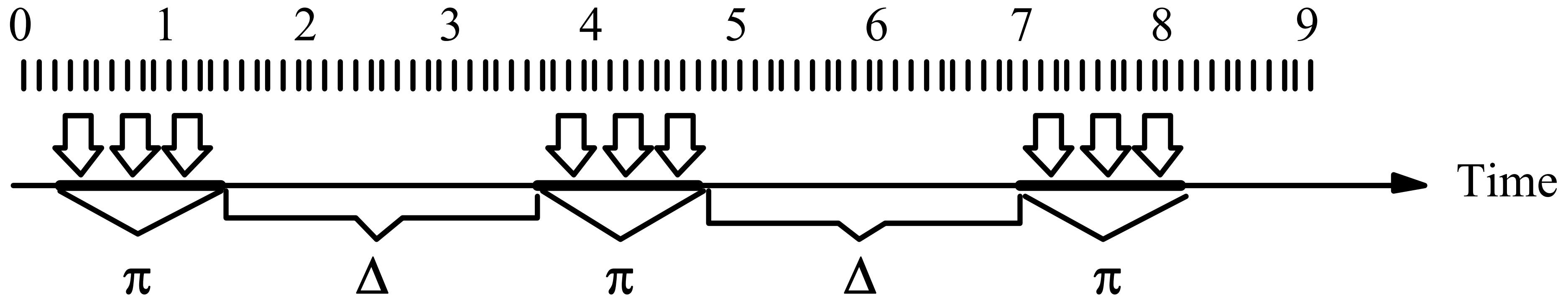
US Blackout, August 14, 2003

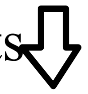
QuickTime™ and a
TIFF (LZW) decompressor
are needed to see this picture.

A valuable lesson from the August 14 blackout is the importance of having time-synchronized system data recorders. The Task Force's investigators labored over thousands of data items to determine the sequence of events, much like putting together small pieces of a very large puzzle. That process would have been ***significantly faster and easier if there had been wider use of synchronized data recording*** *devices.* (p.162 -bolds added)

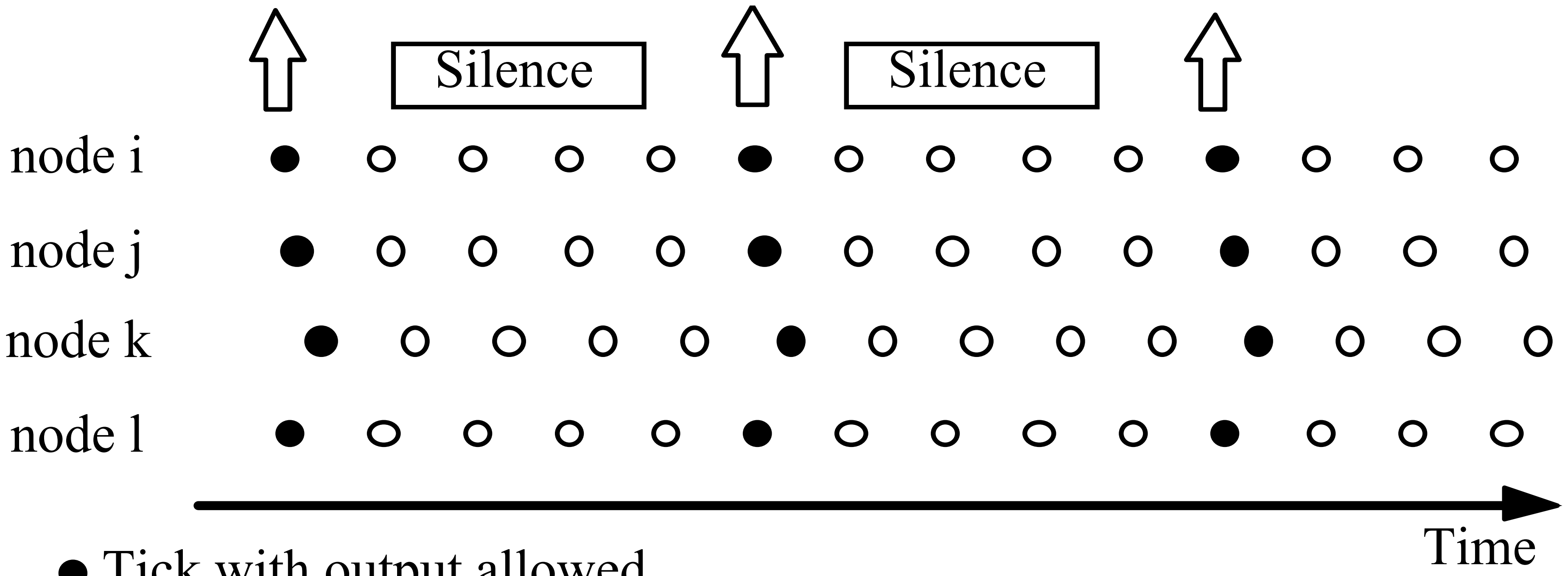
Sparse Time Base

If the occurrence of events is restricted to some active intervals with duration π with an interval of silence of duration Δ between any two active intervals, then we call the timebase π/Δ-sparse, or sparse for short.



Events  are only allowed to occur at subintervals of the timeline

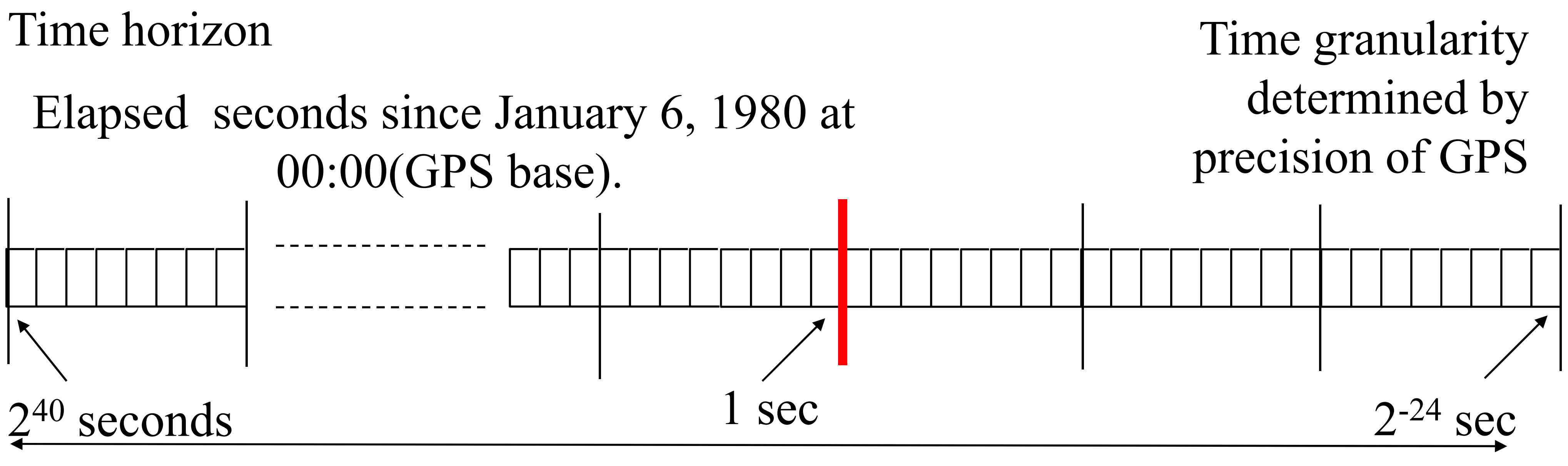
Space/Time Lattice



● Tick with output allowed

○ Tick with output not allowed

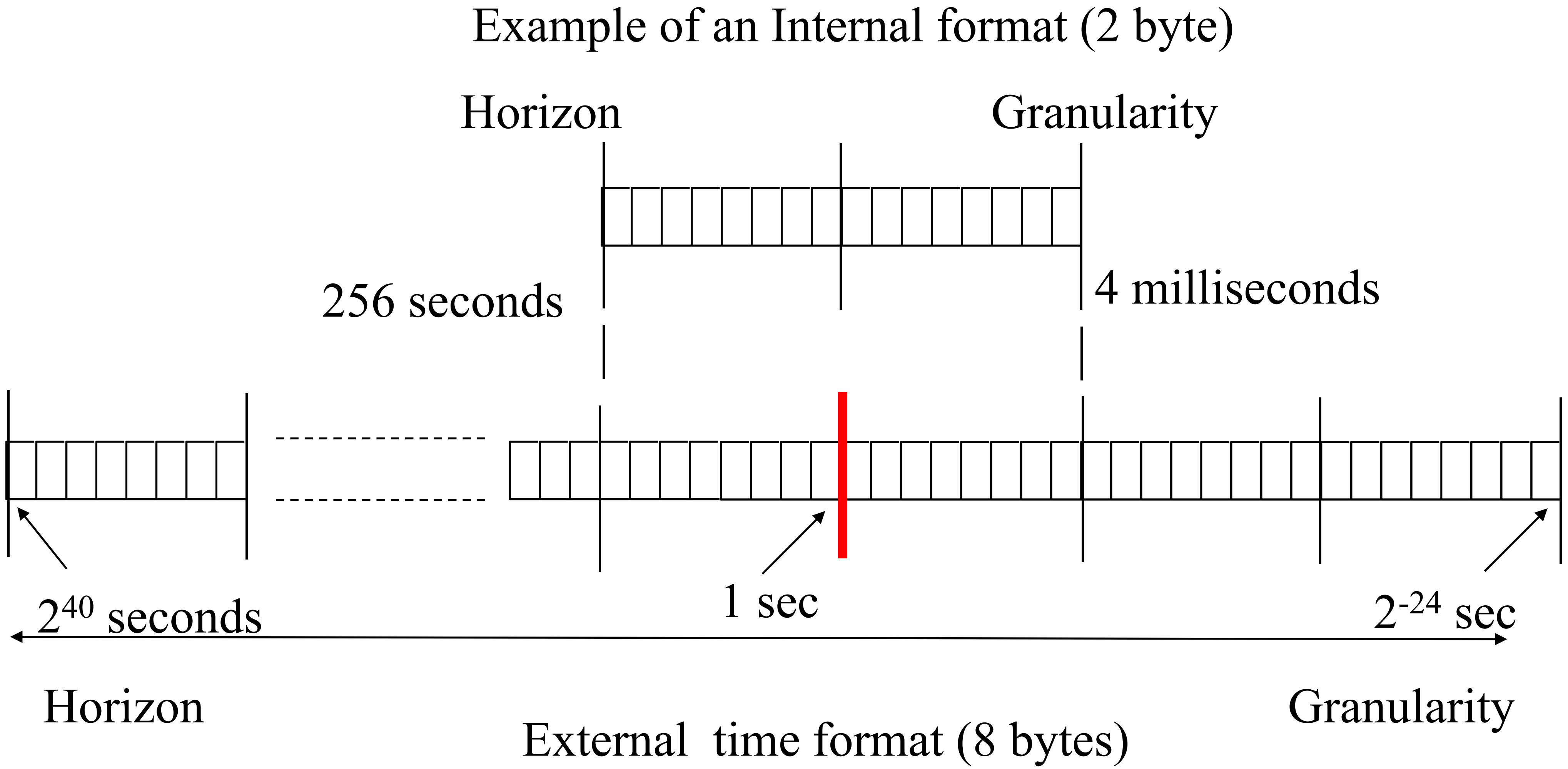
Uniform Time Format--OMG Standard


Time horizon
Elapsed seconds since January 6, 1980 at 00:00(GPS base).
Time granularity determined by precision of GPS
2^{40} seconds
1 sec
2^{-24} sec

external time format (8 bytes)

Start of epoch: January 6, 1980 at 0:00:00 UTC
Granularity about 60 nanosecond Horizon 34841 years
IEEE 1588 time format similar

Internal Time Format--limited Horizon and Precision



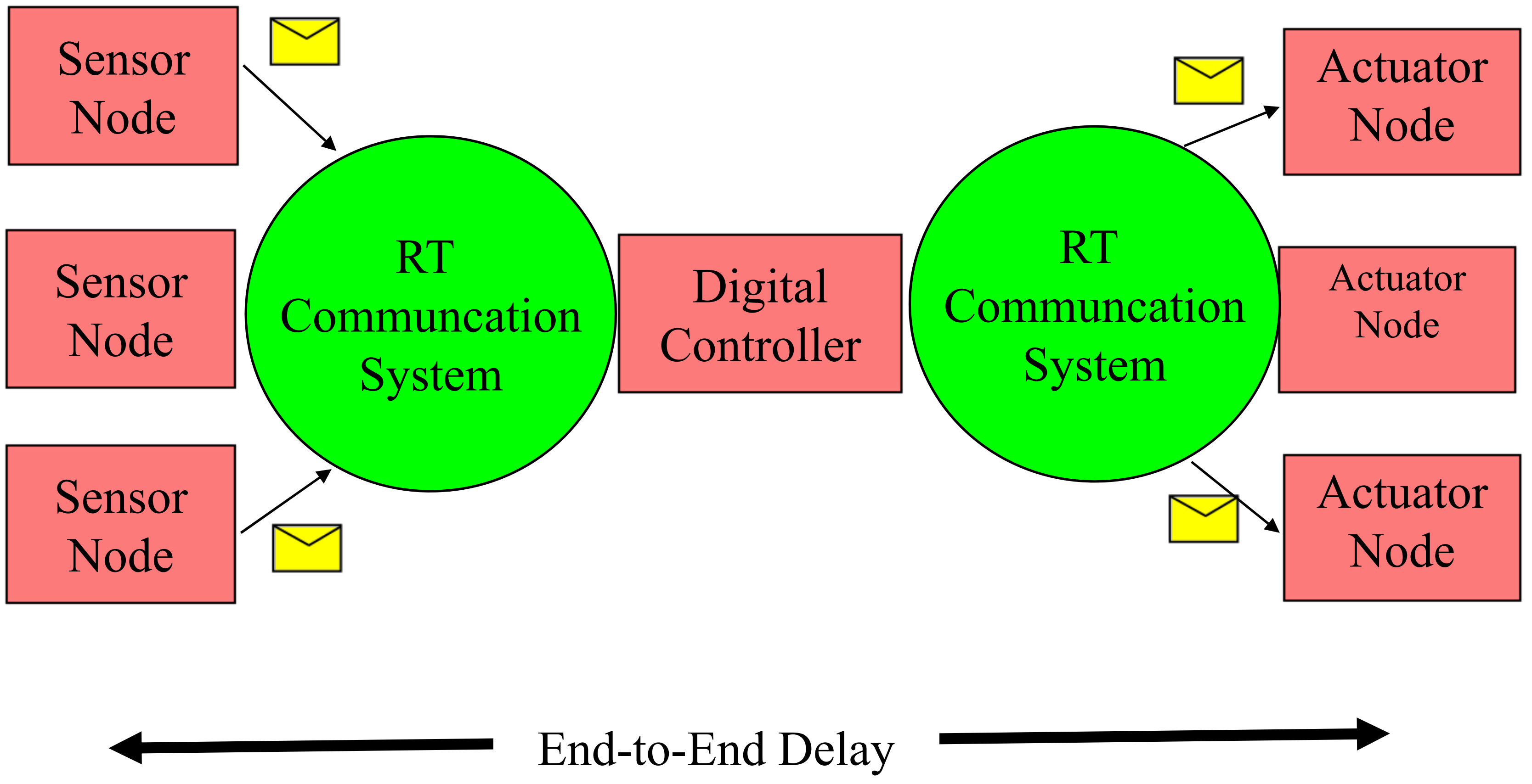
Time and State

In *abstract system theory (Mesarovic, p.45)*, the notion of *state* is introduced in order to separate the *past* from the *future*:

"The state enables the determination of a future output solely on the basis of the future input and the state the system is in. In other word, the state enables a "decoupling" of the past from the present and future. The state embodies all past history of a system. Knowing the state "supplants" knowledge of the past. Apparently, for this role to be meaningful, the notion of past and future must be relevant for the system considered."

A precise concept of time is a prerequisite for a precise concept of state.

A Typical Distributed Control System



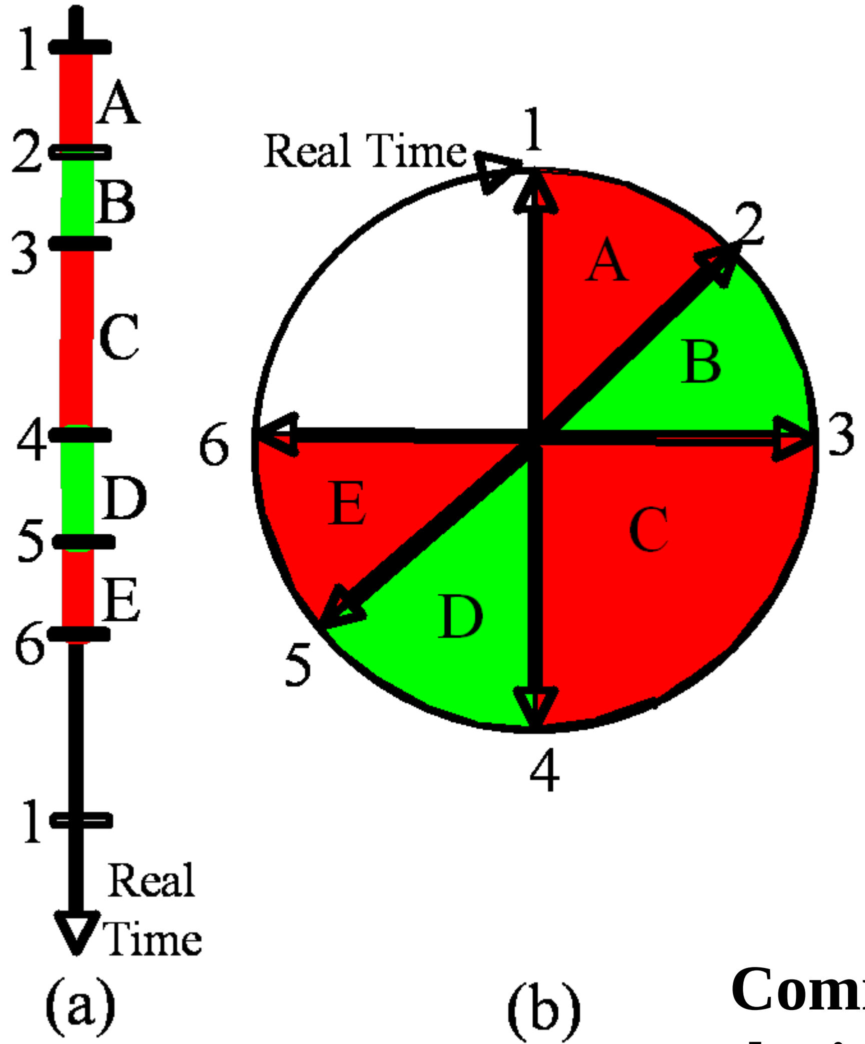
Global Time

Control System Requirements--Periodicity

Periodicity is not mandatory, but often assumed as it leads to simpler algorithms and more stable and secure systems. Most of the algorithms developed with this assumption are very sensitive to period duration variations, jitter at the starting instant. This is especially the case of motor controllers iin precision machines. Simultaneous sampling of inputs is also an important stability factor.

From : Decotignie, J., D., *Which Network for Which Application*, in *The Industrial Communication Technology Handbook*, R. Zuwarski, Editor. 2005, Taylor and Francis: Boca Raton. p. 19/1-19/15. -- *p. 19-4*

A Typical Control Cycle



1	Start of Cycle
A	Observation of Sensor Input
2	Start of Transmission of Sensor Data
B	Transmission of Input Data
3	Start of Processing of Control Algorithm
C	Processing of Control Algorithm
4	Termination of Processing
D	Transmission of Output Data
5	Start of Output to Actuators
E	Output Operation at the Actuator
6	Termination of Output Operation
1	Start of Cycle

Communication Services are only needed during the green intervals--the *Pulses*.

Cyclic Representation of Time

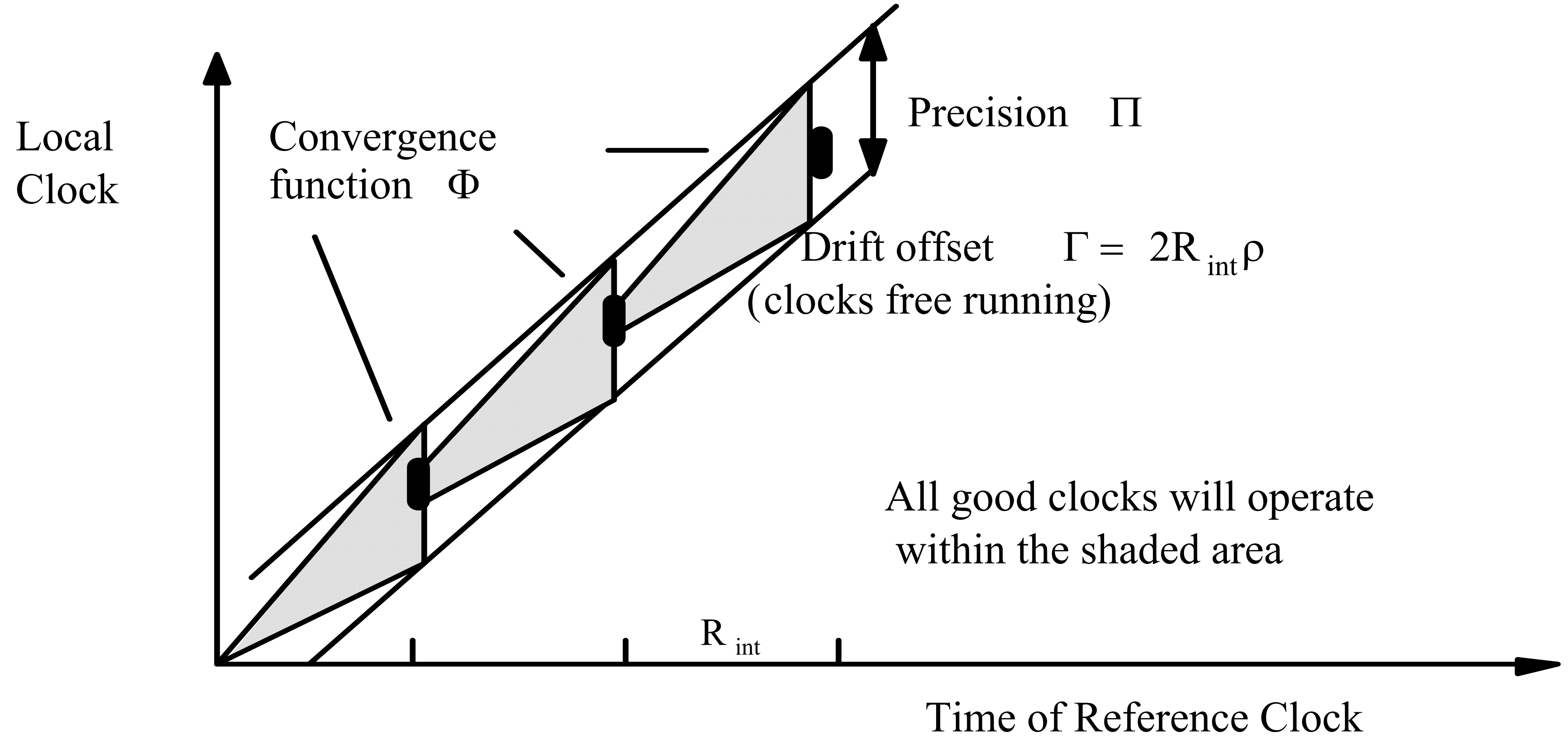
A cycle is a period of real-time between the repetitions of regular events:

- ♦ A cycle is specified by the duration of the period and the position of its start, the cycle start phase relative to a given time reference.
- ♦ In order to avoid the exponential explosion in the combination of cycles it is reasonable to demand that all cycles are in a harmonic relationship.
- ♦ If we use a binary representation of time, then a cycle can be defined by specifying a bit position in the time representation.
- ♦ The sparse time model is inherently cyclic.

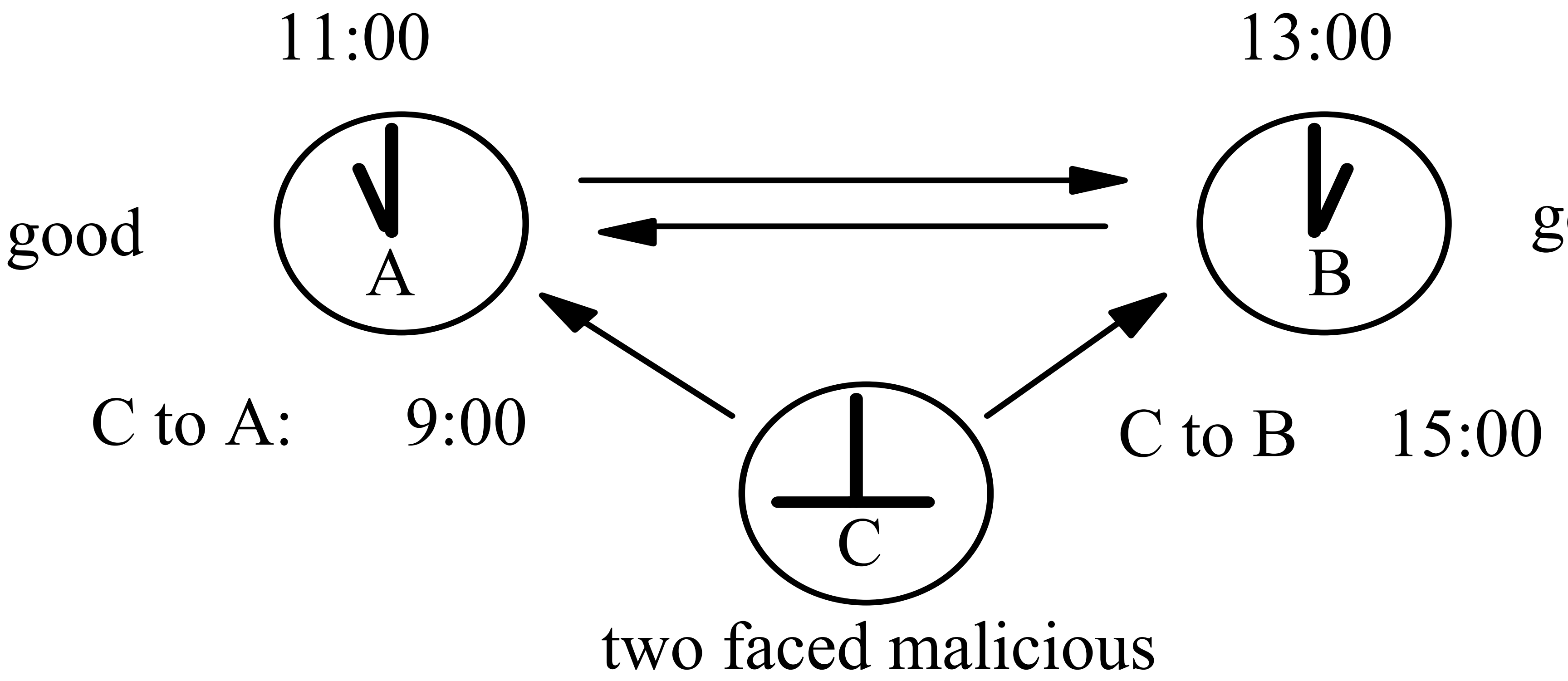
Internal Clock Synchronization

The Synchronization Condition

$$\Phi + \Gamma \leq \Pi$$



Malicious (Byzantine) Clocks



Total Number N of clocks must be $N \geq (3k + 1)$, where k is the number of malicious (Byzantine) faults.

Central Master Algorithm

A unique node, the central master, periodically sends its time counter in synchronization messages to all other nodes, the slave nodes. As soon as a slave receives a new time value from the master, the slave records the state of its local time counter as the time of message arrival. The difference between the master's time contained in the synchronization message and the recorded slave's time of message arrival, corrected by the latency of the message transport, is a measure of deviation of the two clocks. The slave then corrects its clock by this deviation to bring it into agreement with the master's clock.

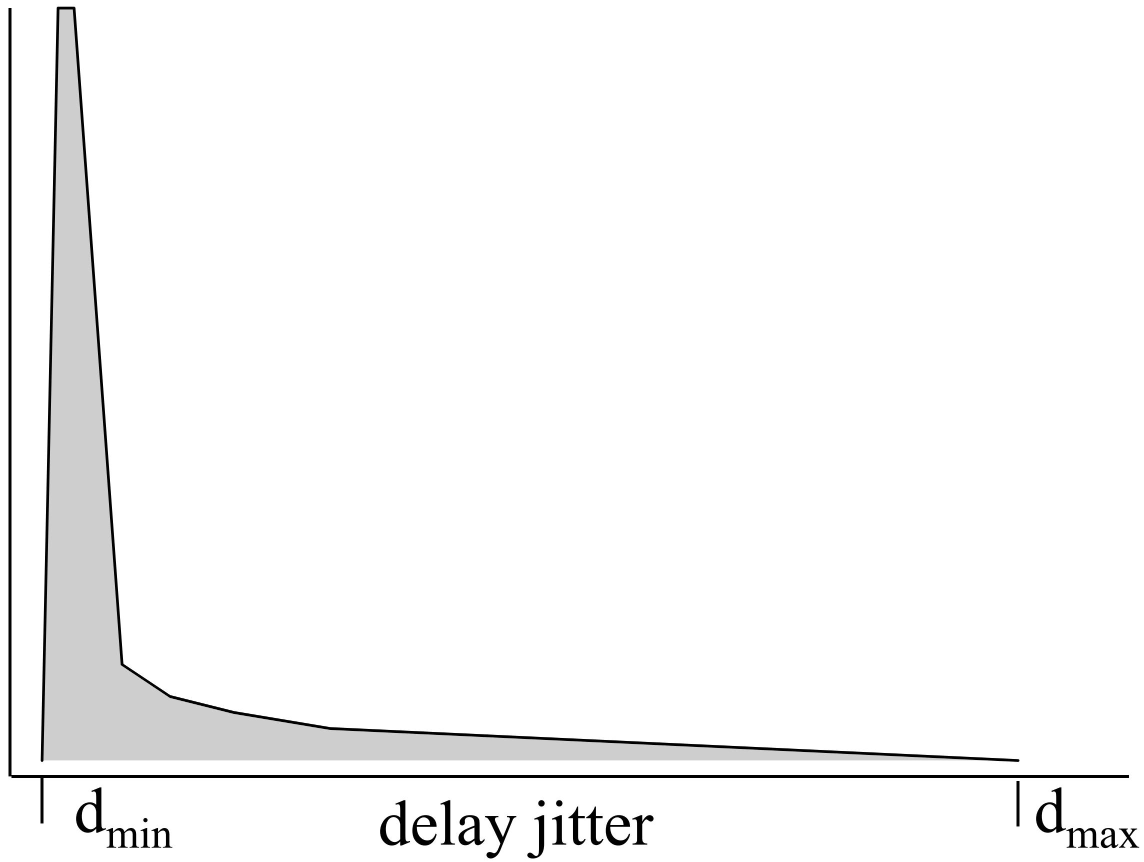
Precision of central Master Algorithm:

$$\Pi_{central} = \varepsilon + \Gamma$$

Delay Jitter

The difference d_{max} - d_{min} is called the *delay jitter* ε.

In standard OSI Protocols (with time redundant transmissions) the typical protocol execution time distribution is as depicted:



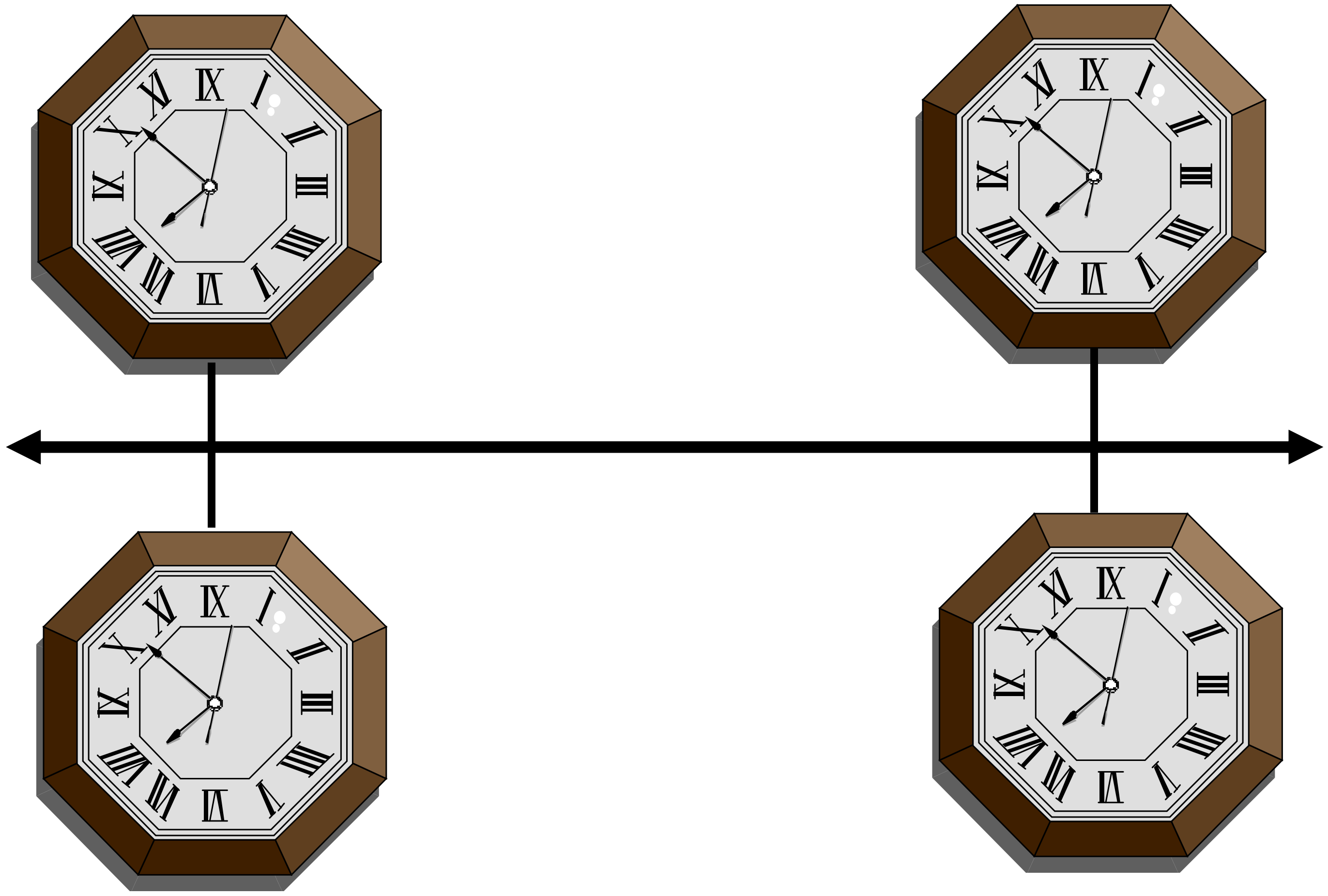
Distributed Clock Synchronization

Typically, distributed fault-tolerant clock resynchronization proceeds in three distinct phases.

- ♦ Every node acquires knowledge about the state of the global time counters in all other nodes by message exchanges among the nodes.
- ♦ Every node analyzes the collected information to detect errors and executes the convergence function to calculate a correction value for the node's local global time counter.
- ♦ The local time counter of the node is adjusted by the calculated correction value.

The algorithms differ in the way in which they collect the time values from the other nodes, in the type of convergence function used, and in the way in which the correction value is applied to the time counter.

How well can we synchronize clocks?



Convergence Function

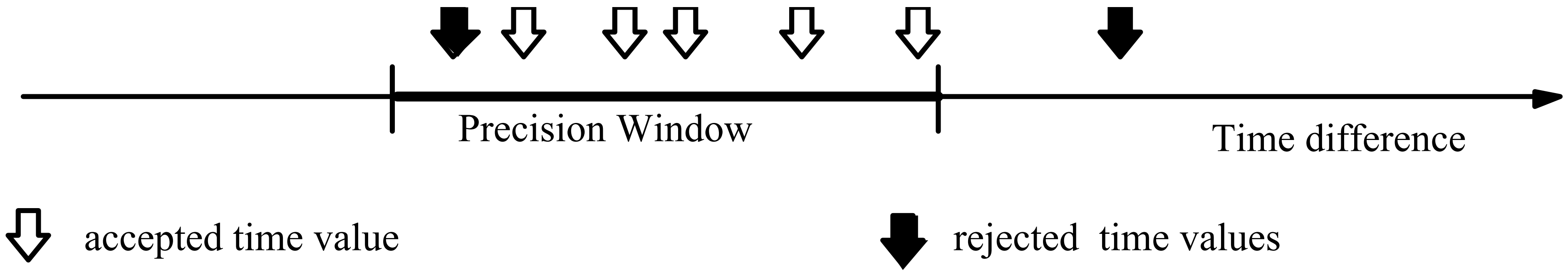
Examples:

- ♦ Average Algorithm
- ♦ Fault-Tolerant Average (FTA)
- ♦ Fault-Tolerant Midpoint
- ♦ Interactive Consistency Algorithms

Fault-Tolerant Algorithm

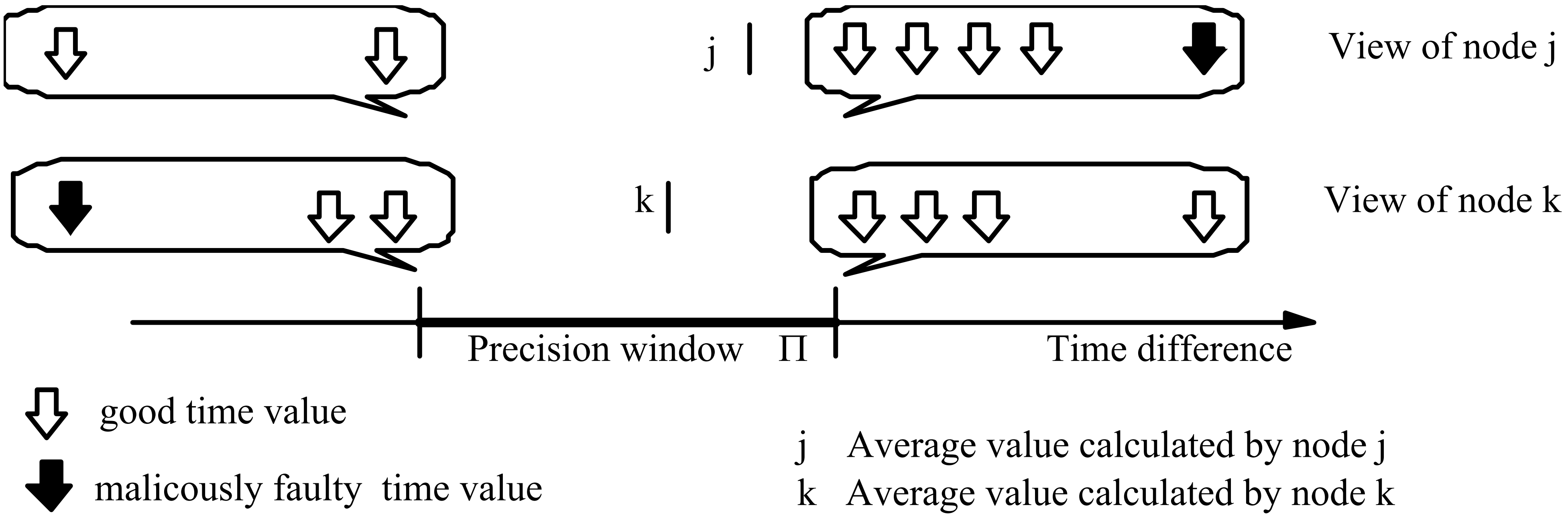
Every node measures the time differences between its own clock and all other clocks and rejects the k extreme differences, where k is the number of Byzantine faults that are to be tolerated.

If k=1, then



Fault Tolerant Average Algorithm

The worst scenario happens, if the Byzantine clock sets its (faulty) time values at different nodes at a different corner of the Precision window:


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

Precision of the FTA

Convergence Function

$$\Phi(N,k,\varepsilon) = k\Pi/\,(N-2k)+\varepsilon$$

Precision

$$\Pi(N,k,\varepsilon,\Gamma) = (\varepsilon+\Gamma)\frac{N-2k}{N-3k} = (\varepsilon+\Gamma)\mu(N,k)$$

where $\mu(N,k)$ is called the *Byzantine error factor* and is tabulated in the following table:

Number of nodes in the ensemble

Faults	4	5	6	7	10	15	20	30
1	2	1.5	1.33	1.25	1.14	1.08	1.06	1.03
2				3	1.5	1.22	1.14	1.08
3					4	1.5	1.27	1.22

Interactive Consistency Algorithms

To overcome the problem of the Byzantine error factor, every node sends its view of the ensemble to all other nodes in order that every node has the global view of the situation, i.e., it can find out which node has been cheating.

Every node takes this consistent global view, i.e., the matrix of time vectors, as the basis of the correction factor calculation.

Pro: $\mu(N,k) = 1$

Con: Extra round of communication

Limit to Internal Clock Synchronization

Lundelius and Lynch have shown that in a system with N clocks and a delay jitter ε it is impossible to synchronize clocks better

$$\Delta_{int} = \varepsilon \ (1-1/N)$$

The proof assumes that all clocks have perfect oscillators.

Critical Parameters

What are the critical parameters that determine the quality of the global time base?

- ♦ Drift offset $\Gamma = 2*R_{sync}*\rho$
- ♦ Delay jitter $\varepsilon = d_{max} - d_{min}$
- ♦ The occurrence of Byzantine failure is a rare event

The delay jitter is smallest, if the clock synchronization is performed very close to the physical level--by the hardware.

Compared to the delay jitter, the algorithmic effects are small.

Minimizing the Drift Offset

The Drift Offset

$$\Gamma = 2 * R_{sync} * \rho$$

can be minimized if the relative drift rates of the clocks are minimized.

Let us assume there is a rate master with a precise clock in each cluster.

All other clocks can periodically adjust their rate to that of the rate master.

Since clock-state correction is performed by a fault-tolerant algorithm, a failure in the clock rate correction will not cause a loss of the time base.

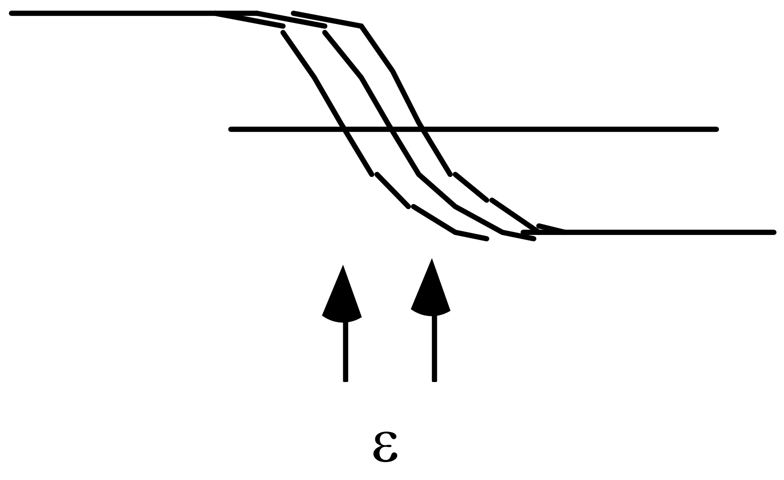
Reading the State of the Remote Clocks

synchronization message assembled and interpreted	approximate range of jitter
at the application software level	500μsec to 5 msec
in the kernel of the operating system	10μsec to 100μsec
in the hardware of the communication controller	1 μsec to 10μsec, or even better

Probabilistic Clock Synchronization:

Measure the time of a request/respond transaction that contains the time value of the partner clock and correct the clock value by half of the transaction duration.

Delay Jitter ε of the Signal Edge



The reading error ε is a fraction (less than one third) of a bitcell

External Clock Synchronization

External Clock synchronization is only possible, if the system has access to an external time reference.

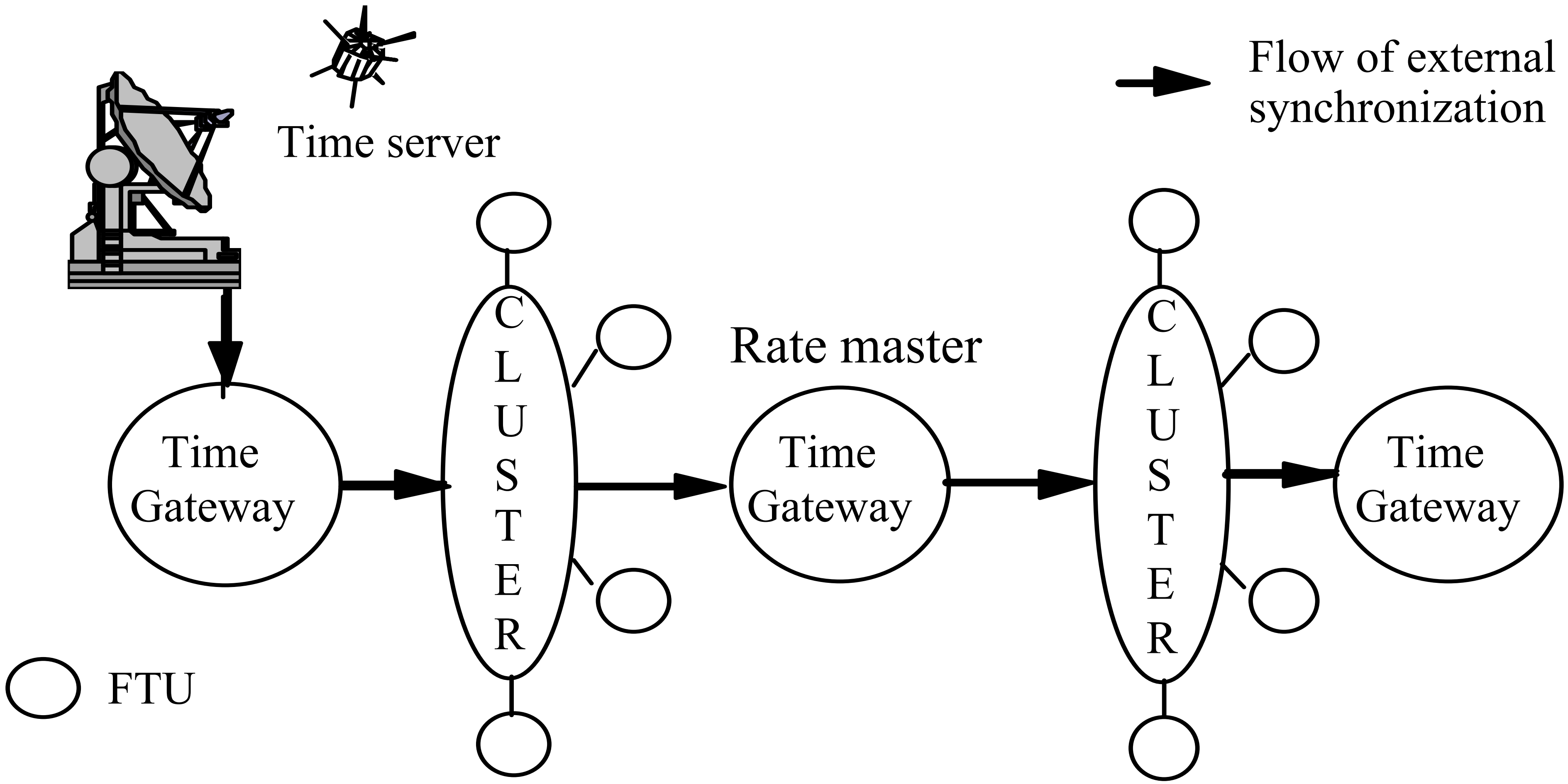
In the future, GPS will be an important time source that gives synchronization accuracy in the submicrosecond interval.

External and internal clock synchronization are complementary:

- ♦ Fault tolerant internal synchronization provides high availability and good short term stability.
- ♦ External clock synchronization provides long-term stability, but the availability may be lower.

If the rate master clock resides in a gateway to the external time reference, it can perform the external clock synchronization.

External Clock Synchronization (2)_

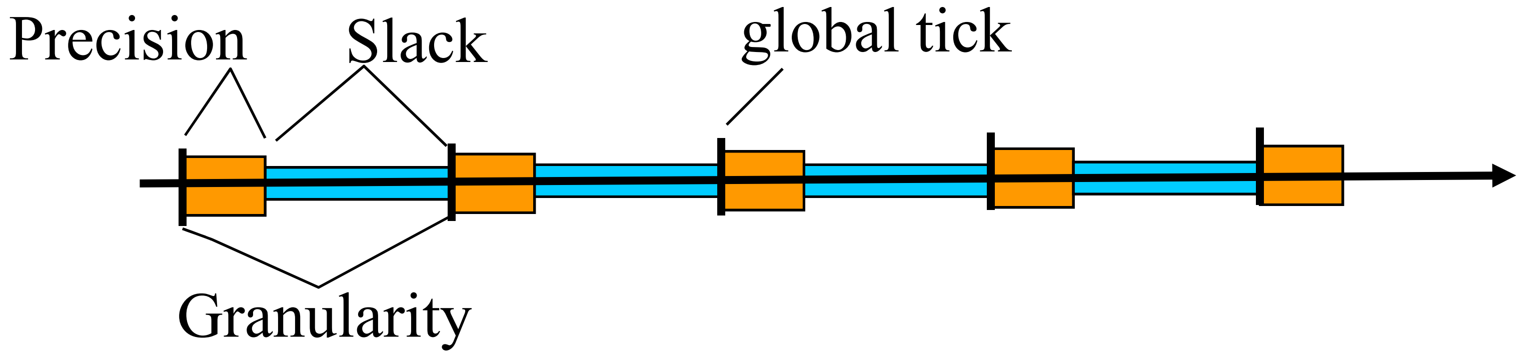


Quality Attributes of a Global Time Base

- ♦ **Precision**: Maximum difference between the respective ticks of the clocks in an ensemble
- ♦ **Accuracy:** Maximum difference between the tick of a clock and the corresponding tick of the external reference clock
- ♦ **Fault-Tolerance**: Number and types of fault the clocking system can tolerate
- ♦ **Blackout Survivability**: Blackout duration that can be tolerated without losing synchronism.

Blackout Survivability Interval

What is the maximum duration of a total communication blackout that can be tolerated without loss of synchronism?



How long will it take until the clocks drift so far from each other that they will leave the slack interval?

$$\mathbf{d^{blackout} = (g\text{-}\Pi)/2\rho = (g - \varepsilon)/2\rho - R}$$

(Assume Byzanthine error factor $\mu = 1$)

g	Granularity
Π	Precision
ρ	Drift rate
ε	Reading error
R	Resync Interval

Blackout Survivability Interval (ii)

$$\mathbf{d^{blackout} = (g-\Pi)/2\rho = (g - \varepsilon.\mu(N,k))/2\rho - R.\mu(N,k))}$$

Blackout survivability interval $d^{blackout}$ should be at least 5 times longer than the worst-case cold-start synchronization $dsync^{cold}$!

g	Granularity
Π	Precision
ρ	Drift rate
ε	Reading error
R	Resync Interval
μ(N.k)	Byzanthine error factor

Example:

$g = 1\ \mu sec$

$dsync^{cold} = 10$ msec then $\rho < 10^{-5}$

$d^{blackout} = 50$ msec

requires either high quality oscillators or dynamic rate calibration.

Dynamic Clock-Rate Calibration

Drift rate ρ of low quality clocks can be reduced by dynamic clock rate calibration.

Problem: How can we prove that a clocking system with a dynamic clock-rate calibration is *stable* within the *specified fault-hypothesis*?

We do not know of such a proof if there is dynamic feedback!

Solution: Avoid feedback by design.

Combine a fault-tolerant distributed clock-state synchronization algorithm with a central clock-rate calibration algorithm.

As a byproduct, this solution also eliminates the need for any special mechanism for external synchronization--simplification of the design.

Experiments and simulation have shown that with this method a drift rate of $< 10^{-5}$ can be achieved, even with low quality oscillators.