TU Wien

The Role of *Time* in Embedded System Design

H.Kopetz
September 2009

Outline

- Introduction
- Model of Time
- RT Entities and RT Images
- Temporal Relations
- Time in Fault-Tolerant Systems
- Clock Synchronization
- Conclusions

Embedded Systems

- *Embedded Systems* enable the real-time computer control of physical devices and systems, ranging from mobile phones, to television sets, to automotive engines and to industrial robots (to take a few examples) in order to achieve an ***unprecedented level of utility and performance***.
- Embedded systems are also called *Cyber-Physical Systems* (CPS) to denote the emphasis and the close synergetic interactions of a ***real-time information processing subsystem*** (the *Cyber System*) with a ***physical system*** that is to be controlled (the *Physical System).*

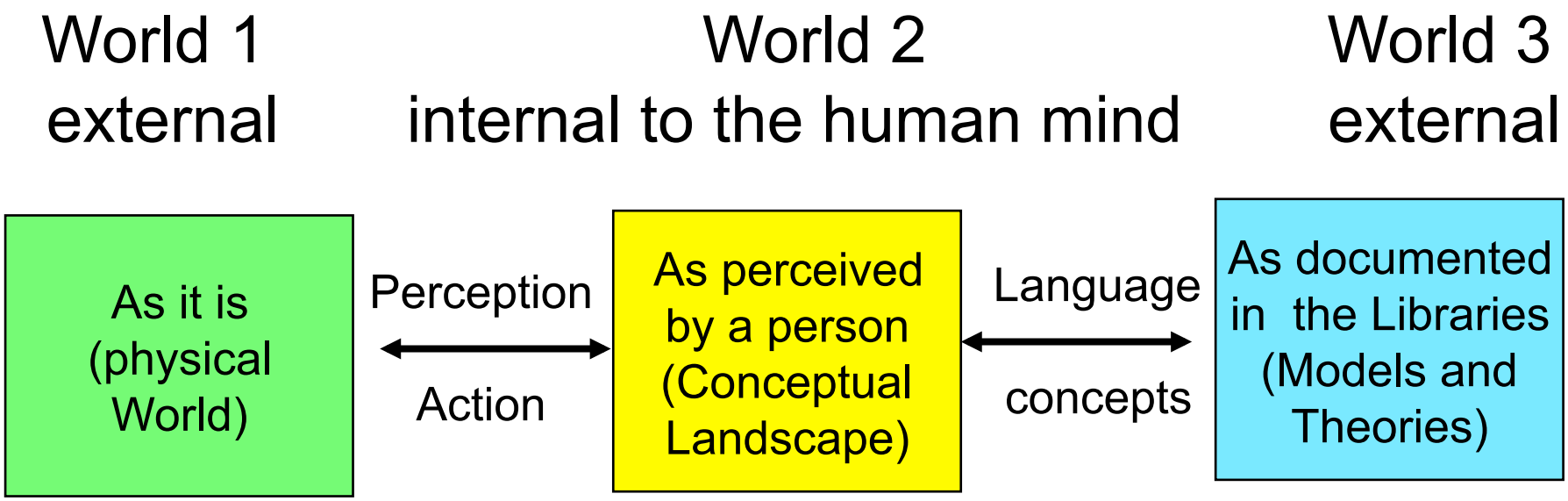
The Three Worlds

K. Popper distinguishes between three worlds

- World 1: The *physical world of natural structures and all types of biological systems*: the world of *facts*—the **physical system**
- World 2: The *subjective view of the world* by a person--the *conceptual landscape* or the *image* (**K.Boulding**)
- World 3: The *objective world of knowledge*, documented in models and theories and recorded in our libraries: the world of *constructs (e.g., software).*

K.Popper, *On the Theory of the Objective Mind*, Lecture in Vienna, Sept. 3, 1968, in *Objective Knowledge*,th Oxford University Press 1972 pp. 152-190, and *Three Worlds,* The Tanner Lecture on Human Values, Univ. of Michigan, April 7, 1978

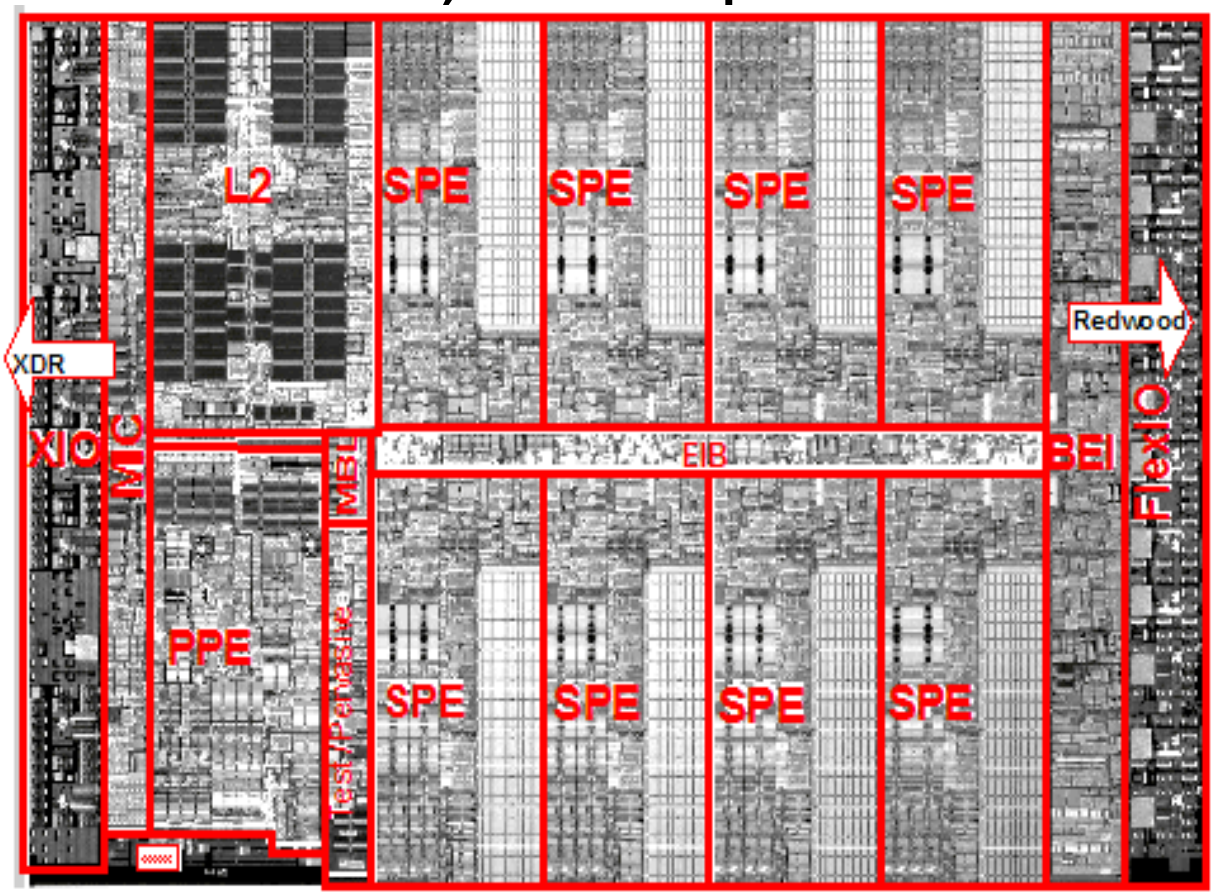
Relationship between the Three Worlds



The human mind is the ***Mediator*** between World 1 and World 3

The Cell Processor

2005: Joint development of IBM, Sony and Toshiba, 90 nm process 250 Mio Transistors,



Every one of these nine computers is more powerful than a PC of the 1980'ies

(256 kByte of local memory)

We Understand the World by *Modeling*

A *physical system* has a nearly infinite number of properties —every single transistor of a billion-transistor SoC consists of a huge number of atoms that are placed in space and have their own identity.

We need to *abstract*, to build *models*, that leave out the seemingly irrelevant detail, in order to be able to reason about properties of interest.

We take the view that certain properties, such as *determinism* or *complexity*, can only be assigned to *models of physical systems*, but not to the physical systems themselves, no matter whether these physical systems are *natural* or *man-made*.

What is a *Model?*

A model is a reduced representation of an *object system* that is formed in order to *understand* and *analyze* a chosen *aspect* of the object system.

The *purpose of the model* determines which properties of the object system must be retained in the model and which properties of the object system can be discarded.

Characteristics of a *Good* Model

A *good* model is characterized by the following properties:

- Utility (*fit for purpose*)
- Parsimony (*Occam's razor*)
- Understandability (*Simplicity*)
  - *Simple Model*: limited number of relations, such that the dependencies can be captured directly by the human mind (*Halfordl's relational cognitive complexity*}
  - *Complex Model:* relations among the entities of the model beyond the capability of direct human comprehension

Simplification Strategies

Our mental capabilities put a limit on the number of elements and relations that we can understand simultaneously--the psychologist *Halford* puts this limit at relations of at most rank five—beware of *concurrency*!

If a scenario exceeds these limits, we resort to simplification strategies:

- Abstraction (building of higher-level models)
- Partitioning (separation of concerns in *space*)
- Segmentation (separation of concerns in *time*)

Software for Dependable Systems

A recent (2007) report on *Software for Dependable Systems: Sufficient Evidence?* from the US National Academies contains as one of its central recommendations . .

One key to achieving dependability at reasonable cost is a serious and sustained commitment to ***simplicity****, including simplicity of critical functions and simplicity in system interactions.* ***This commitment is often the mark of true expertise.***

The *Major* Challenge of Science

Finding proper abstractions (concepts) and simple models to explain observed phenomena.

Example:

Newton's Conceptualization of *Power*, *Mass*, *Acceleration*

We need *scientific concepts* that are based on an *understandable model*, i.e., *a deliberate simplification of reality with the objective of explaining the chosen property of reality that is relevant for a particular purpose*.

Six *Desired* Properties of Scientific Concepts

- *Utility*: the new concept should serve a useful well-defined purpose.
- *Abstraction and Refinement*: the new concept should either be a *basic level concept* or an abstraction or a refinement of a *basic level concept*.
- *Precision*: The characteristic properties of the new concept must be precisely defined.
- *Identity*: The new concept should have a distinct identity and should be significantly different from other concepts in the domain.
- *Stability*: The new concept should be usable uniformly in many different contexts without any qualification or modification.
- *Analogy*: Similarities with other concepts should be pointed out.

Example: Celestial Mechanics

In celestial mechanics, when we are interested in the interactions between heavenly bodies, we build an *abstraction* where we put aside the diversity of our world and consider it to be a ***single mass point*--**the ultimate simplicity.

Purposes of Embedded System Models

to understand and analyze

- Behavior (Value, Temporal)
- Dependability
- Power and Energy
- Physical Form
- Cost
- etc.

Each purpose is served by a *different set of* model.

Modeling of Temporal Behavior

We need appropriate concepts:

- Model of Time
- *State*—Real-Time (RT) Entity
- Observation
- Real-Time (RT) Image
- RT Object

What is *Time?*

In many models of the natural sciences *time* is an independent state variable that increases monotonically.

The arrow of time

Past  **Future**

In classical physics, the domain of time is *dense*.

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.

Time Formats: NTP vs. PTP

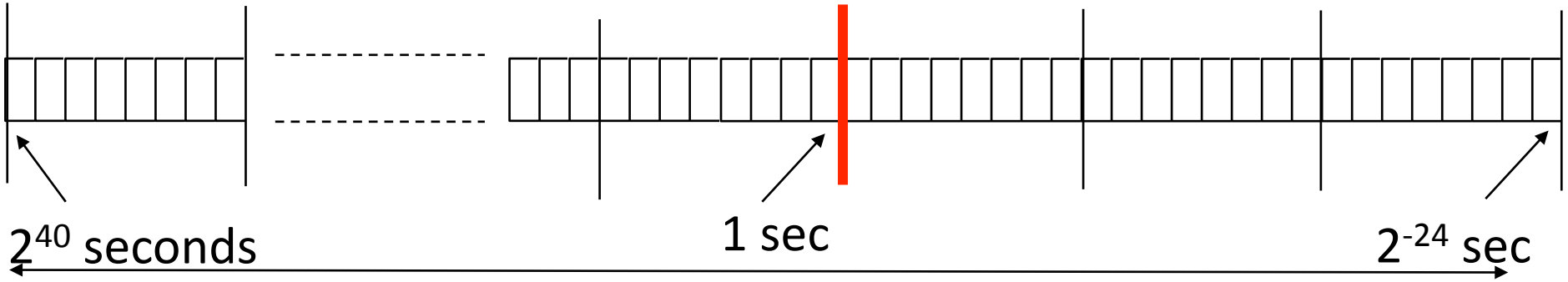
NTP	PTP
Network Time Protocol	Precision Time Protocol (IEEE 1588)
Timestamp Format 32-bit unsigned seconds 32-bit fractions of a second (resolving to 232 pico-seconds)	Timestamp Format 48-bit unsigned seconds 32-bit unsigned nanoseconds
Timescale UTC	Timescale(s) TAI
Prime Epoch 0 hour 1 January 1900	Allows for alternate timescales

Time Format--OMG Standard in the TTA

Time horizon

Elapsed seconds since January 6, 1980 at 00:00(GPS base).

Time granularity determined by precision of GPS

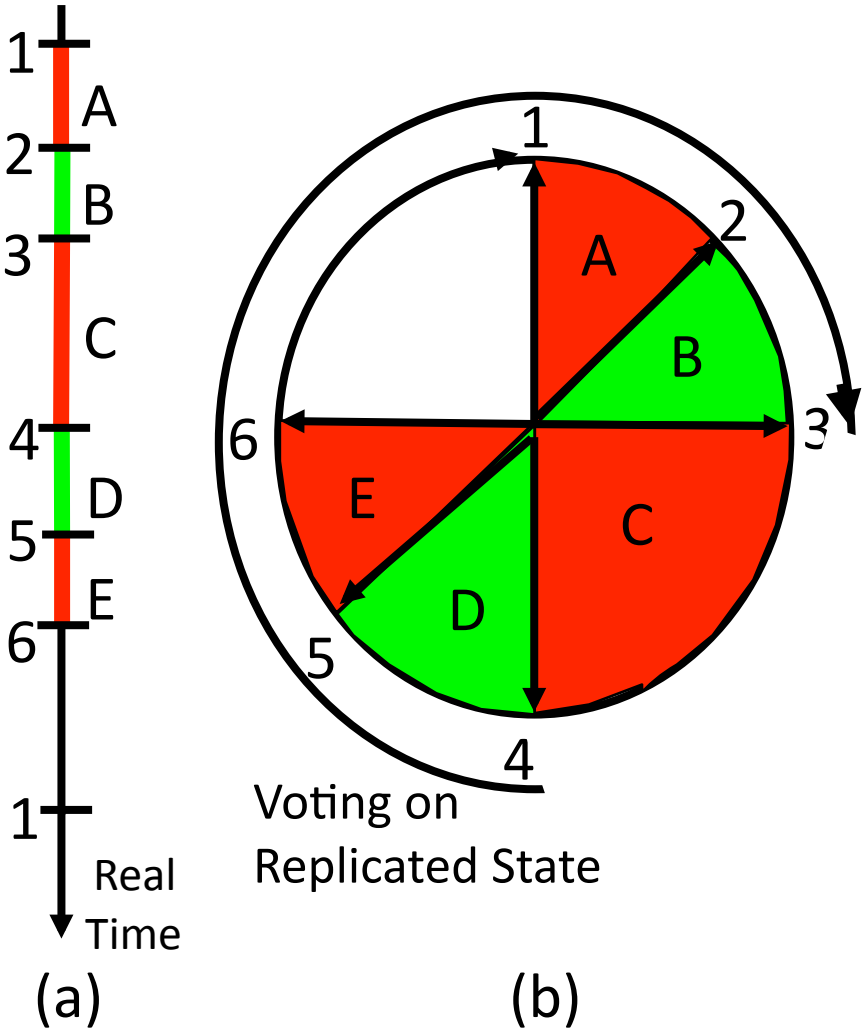


external time format (8 bytes)

Start of epoch: January 6, 1980 at 0:00:00 UTC
Granularity about 60 nanoseconds based on TIA

The occurrence of every significant event can be recorded with this global timestamp -- Related to IEEE 1588 Standard

Control Cycle: Cyclic Model of Time



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

Some Terms relating to *Time*

Instant: A cut of the Newtonian time line
Event: An occurrence at an instant
Clock: A device that contains a counter and increments this counter—i.e. generates events--periodically according to some law of physics (*microticks*).
Granularity: The temporal distance between two ticks of a digital clock,
Reference Clock: A hypothetical clock that has a granularity that is much smaller than the duration of any intervals of interest and which is in full agreement with the international standard of time.e.g.,
e.g., granularitiy of one *femto* second (10^{-15}sec)!

Gobal Time

Global Time: An *abstraction* that is approximated by the synchronized local times of the nodes in a distributed computer system.

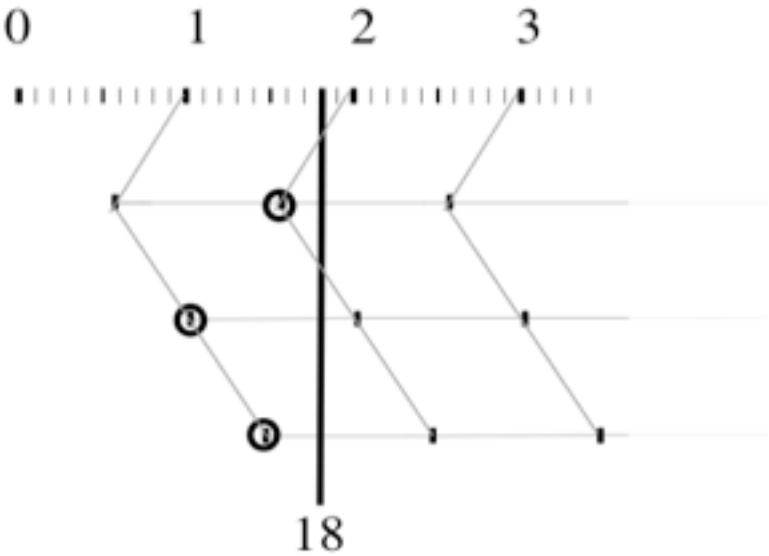
Precision: The maximum duration between two respective local ticks of the global time

Accuracy: The maximum duration between the global time and an external time reference

Timestamp: 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*).

Reasonableness Condition

In a distributed computer system, we call a global time ***reasonable***, if the granularity of the global time is larger than the precision of the clock synchronization.



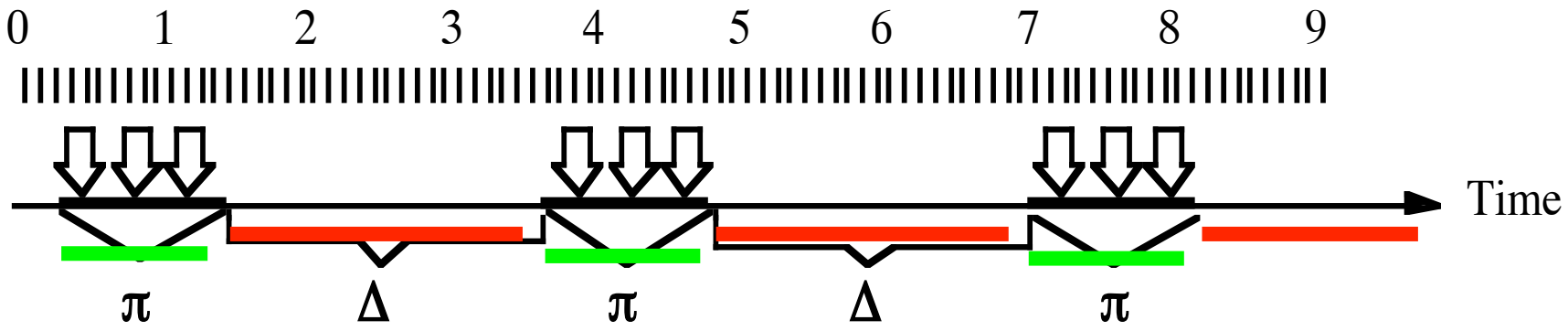
(a)

Concept of State *Mesarovic[5], p.45*

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.

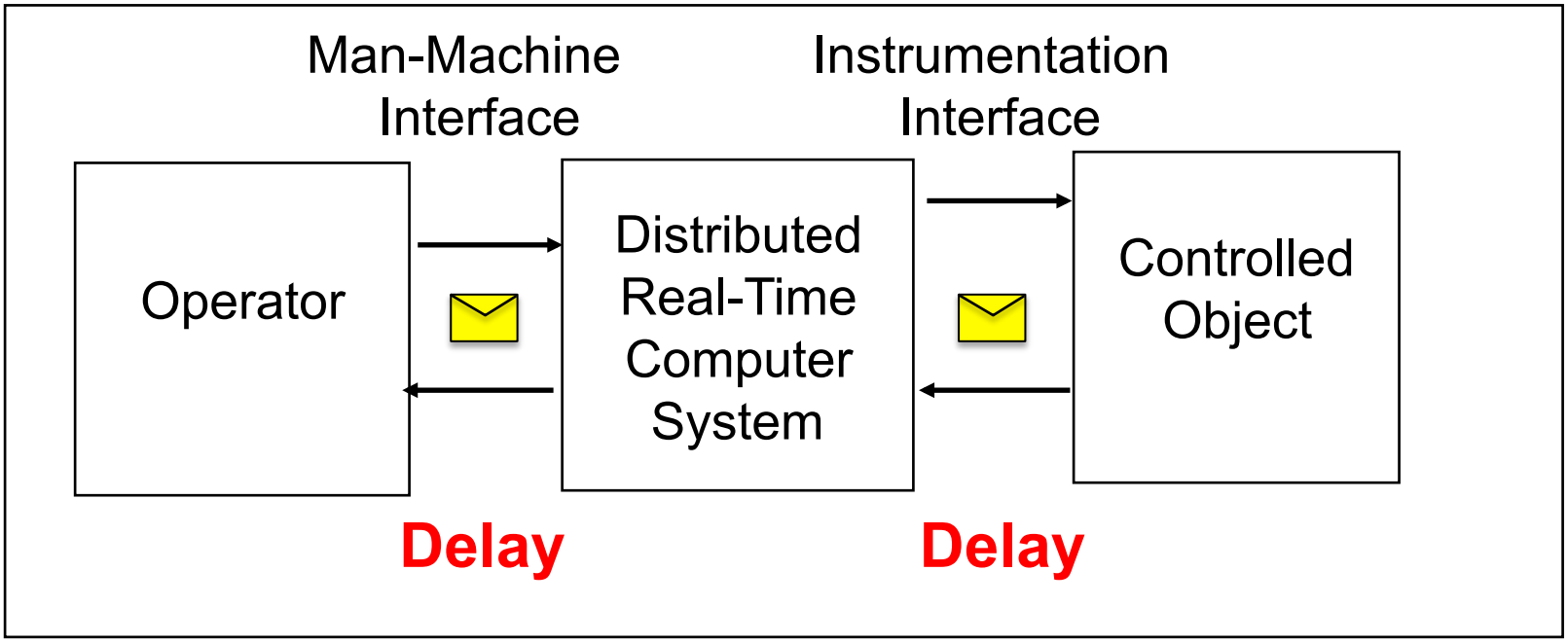
Sparse Time

If the occurrence of events is restricted to some active intervals on the timeline with duration π with an interval of silence of duration Δ between any two active intervals, then we call the time base π/Δ-sparse, or ***sparse*** for short, and events that occur during the active intervals ***sparse events***.



Sparse Event are only allowed during the active intervals

Model of an Embedded System



controlled by human intentions

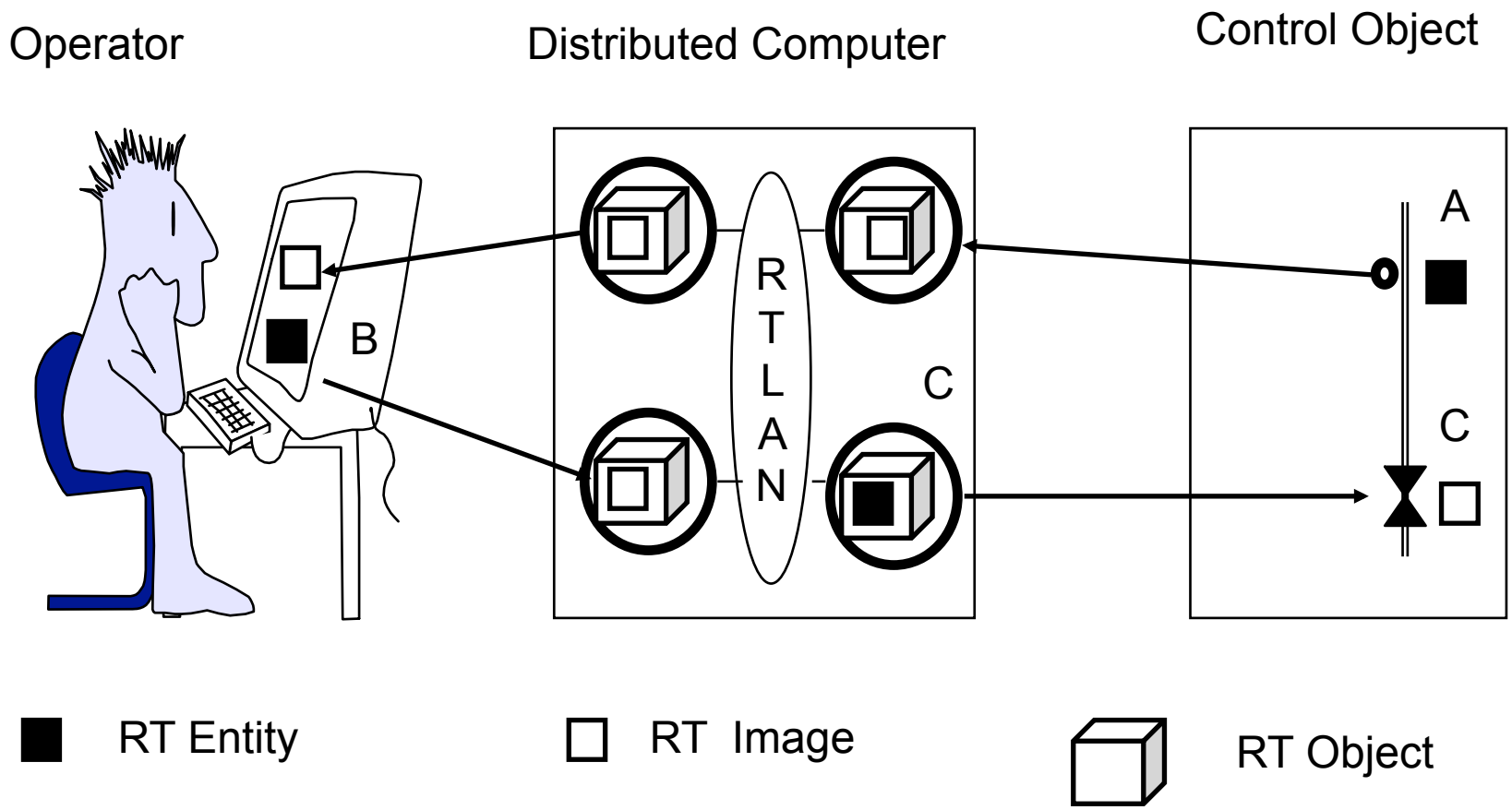
controlled by computer programs

controlled by physical laws involving time

Different Models of Physical Time

- The models of the controlled object are based on the concept of a *dense time* (*Newtonian time)*.
- A distributed computer system uses a *global time* which is an *approximate discrete representation* of the Newtonian time.
- *Sparse time* restricts the occurrence of events in a distributed computer system order that a consistent view of simultaneity can be established in the distributed computer system.

RT Entities, RT Images and RT Objects



Real Time (RT) Entity

A Real-Time (RT) Entity is a *state variable* of interest for the given purpose that changes its state as a function of real-time.

We distinguish between:

- Continuous RT Entities
- Discrete RT Entities

Examples of RT Entities:

- Flow in a Pipe (Continuous)
- Position of a Switch (Discrete)
- Setpoint selected by an Operator
- Intended Position of an Actuator

Attributes of RT-Entities

Static attributes of RT-Entity:

- Name
- Type
- Value Domain
- Maximum Rate of Change

Dynamic Attributes:

- Value set at a particular point in time

Sphere of Control

Every RT-Entity is in the Sphere of Control (SOC) of a subsystem that has the authority to set the value of the RT-entity:

- Setpoint is in the SOC of the operator
- Actual Flow is in the SOC of the control object
- Intended Valve Position is in the SOC of the Computer

Outside its SOC a RT-entity can only be observed, but not modified.

At this level of abstraction, changes in the representation of a RT-entity are not significant.

Observation

Information about the state of a RT-entity at an instant is captured in an observation.

An observation is an atomic triple

Observation = <Name, Time, Value>

consisting of:

- The name of the RT-entity
- The instant when the observation has been made
- The values of the RT-entity

Observations are transported in messages.

State and Event Observation

An observation is a *state observation*, if the value of the observation contains the full or partial state of the RT-entity. The time of a state observation denotes the point in time when the RT-entity was sampled.

An observation is an *event observation*, if the value of the observation contains the difference between the "old state" (the last observed state) and the "new state". The time of the event information denotes the point in time of the L-event of the "new state".

RT Images

A *RT-Image* is a picture of a RT Entity. A RT image is valid at a given point in time, if it is an accurate representation, both in the domains of value and time, of the corresponding RT Entity.

RT-Images

- are only valid during a limited interval of real-time.
- can be based on an observation or on a state estimation.
- can be stored in data objects, either inside a computer (RT object) or outside in an actuator.

RT Object

A RT-object is a “container” for a RT-Image or a RT-Entity in the Computer System.

A RT-object k

- has an associated real-time clock which ticks with a granularity t_k. This granularity must be in agreement with the dynamics of the RT-entity this object is to represent.
- Activates an object procedure if the time reaches a preset value.
- If there is no other way to activate an object procedure than by the periodic clock tick, we call the RT-object a *time-triggered* RT object.

Distributed RT-Object

A distributed RT-object is a set of replicated RT-objects located at different sites.

Every local instance of a distributed RT-object provides a specified service to its local site.

The quality of service of a distributed RT-object must be in conformance with some specified consistency constraint.

Examples:

- Clock synchronization within a specified precision
- Membership service with a specified delay

Temporal Relations

There are a number of important temporal relations among the concepts that we have introduced so far:

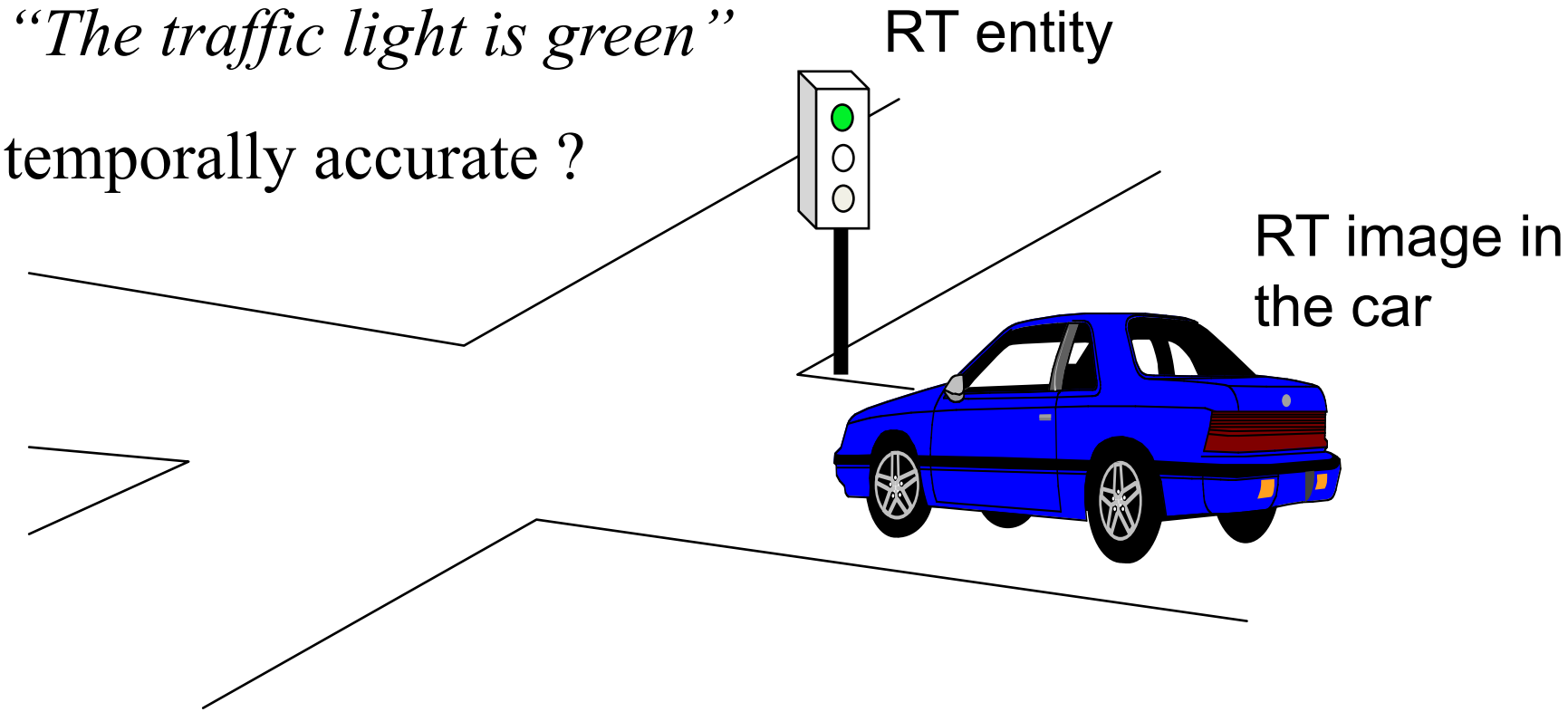
- Temporal Accuracy
- Temporal Uncertainty—Faithfulness of the Models
- Permanence of Data
- Idempotence
- Replica Determinism

Temporal Accuracy—Traffic light

How long is the RT image,
based on the observation:

"The traffic light is green"

temporally accurate ?



Temporal Accuracy--Definition

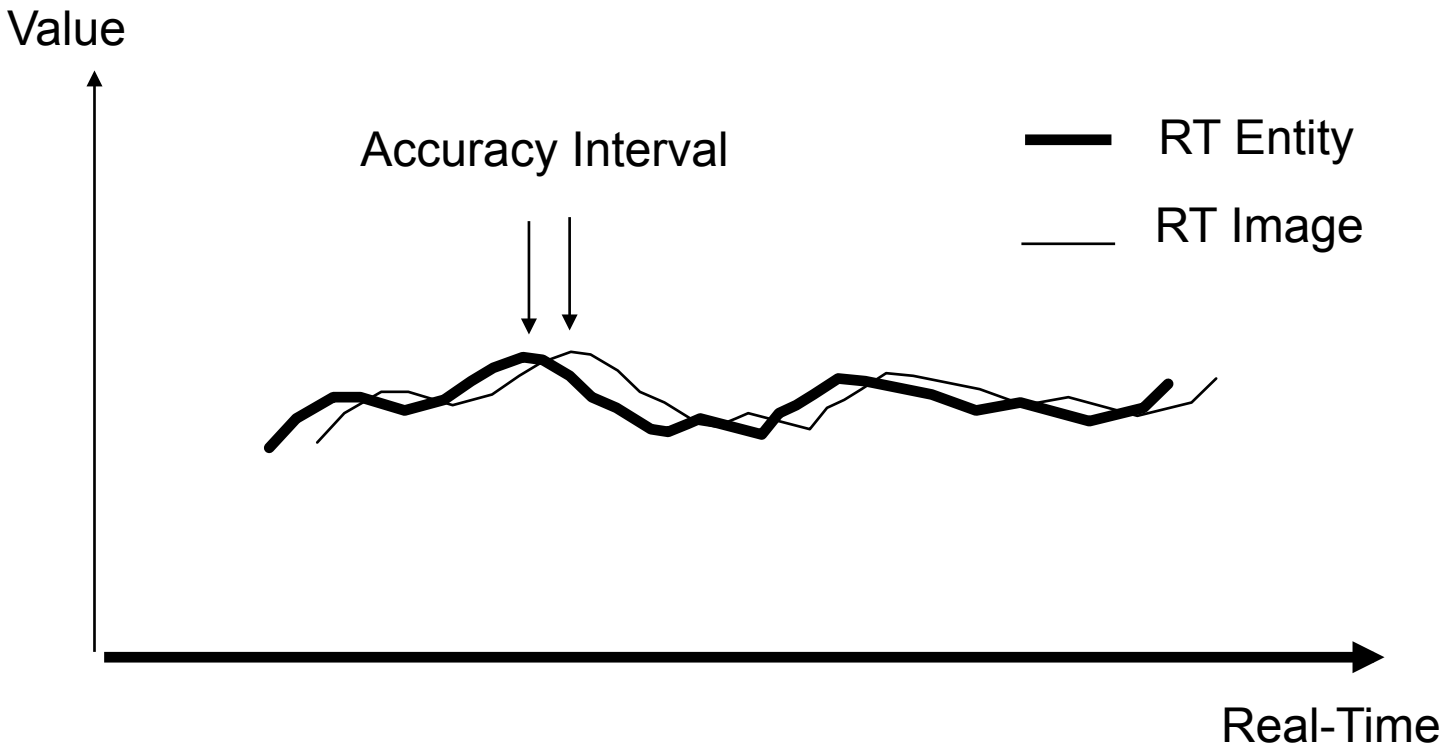
The temporal accuracy of a RT image is defined by referring to the recent history of observations of the related RT entity. A recent history RH_i at time t_i is an ordered set of time points $<t_i, t_{i-1}, t_{i-2}, \ldots . t_{i-k}>$, where the length of the recent history

$$d_{acc} = t_i - t_{i-k}$$

is called the temporal accuracy. Assume that the RT entity has been observed at every time point of the recent history. A RT image is temporally accurate at the present time t_i if

$$\exists\, t_j \in RH_i : Value\,(RT\,image\,at\;t_i) = Value\,(\,RT\,entity\;\;at\;t_j)$$

Temporal Accuracy of RT Objects



If a RT-object is updated by observations, then there will always be a delay between the state of the RT entity and that of the RT object

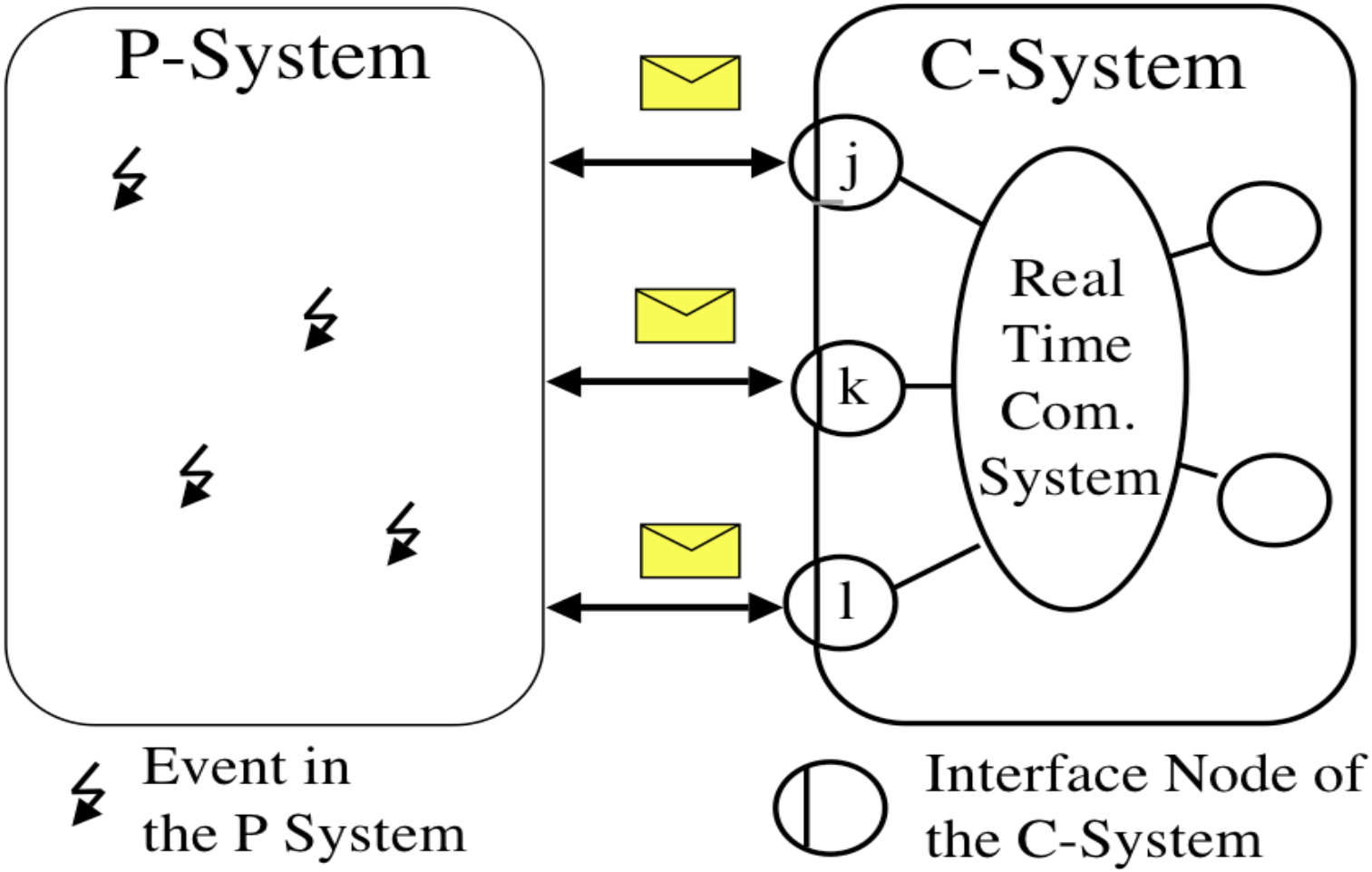
Example: Point of Ignition in an Engine

The point in time of ignition is a function of the following parameters:

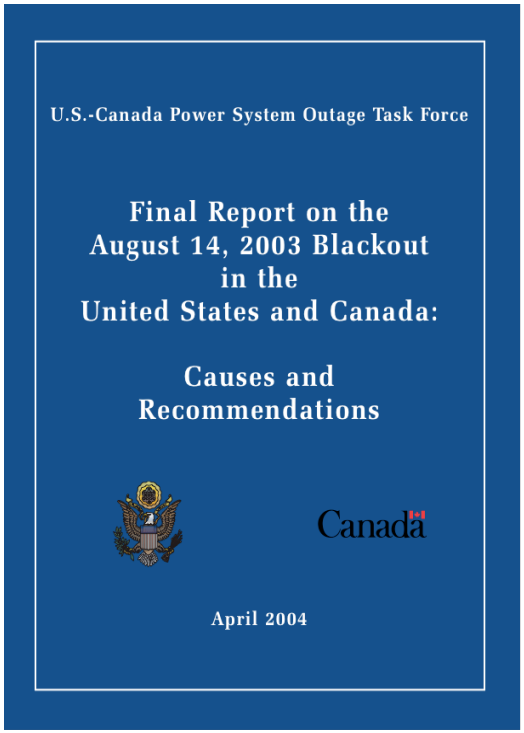
Parameter	Recent History (sec)
crank position	10^{-6}
h-state	10^{-3}
gas pedal position	10^{-2}
load	1
temperature	10^{+1}

There are seven orders of magnitude difference in the required temporal accuracy of the parameters.

Temporal Uncertainty—Faithfulness



US Blackout, August 14, 2003



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)

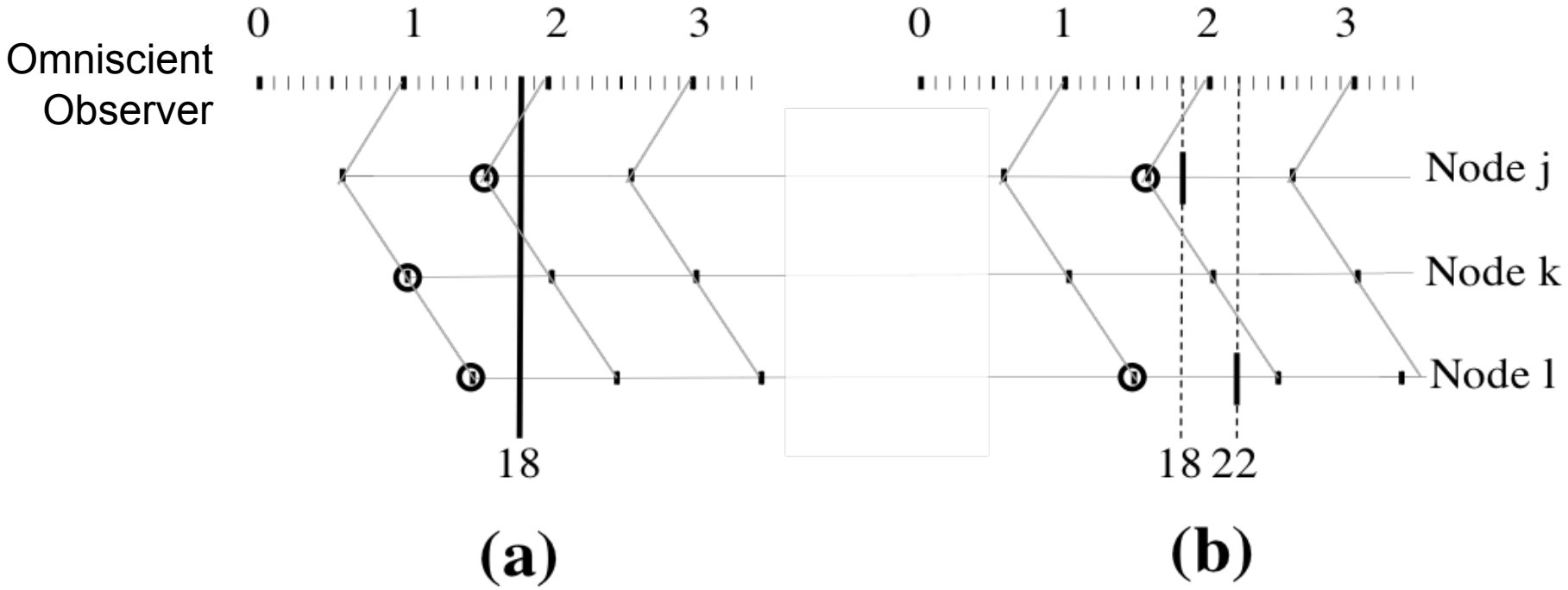
Temporal Uncertainty

If the events in the environment of the computer system are close together, then the temporal order of the events that is perceived by the distributed computer system may be different from the temporal order that has occurred in reality!

The precision of the global time limits the faithfulness of the computer model.

Time-Stamping Errors

Reasonableness Condition is Satisfied!

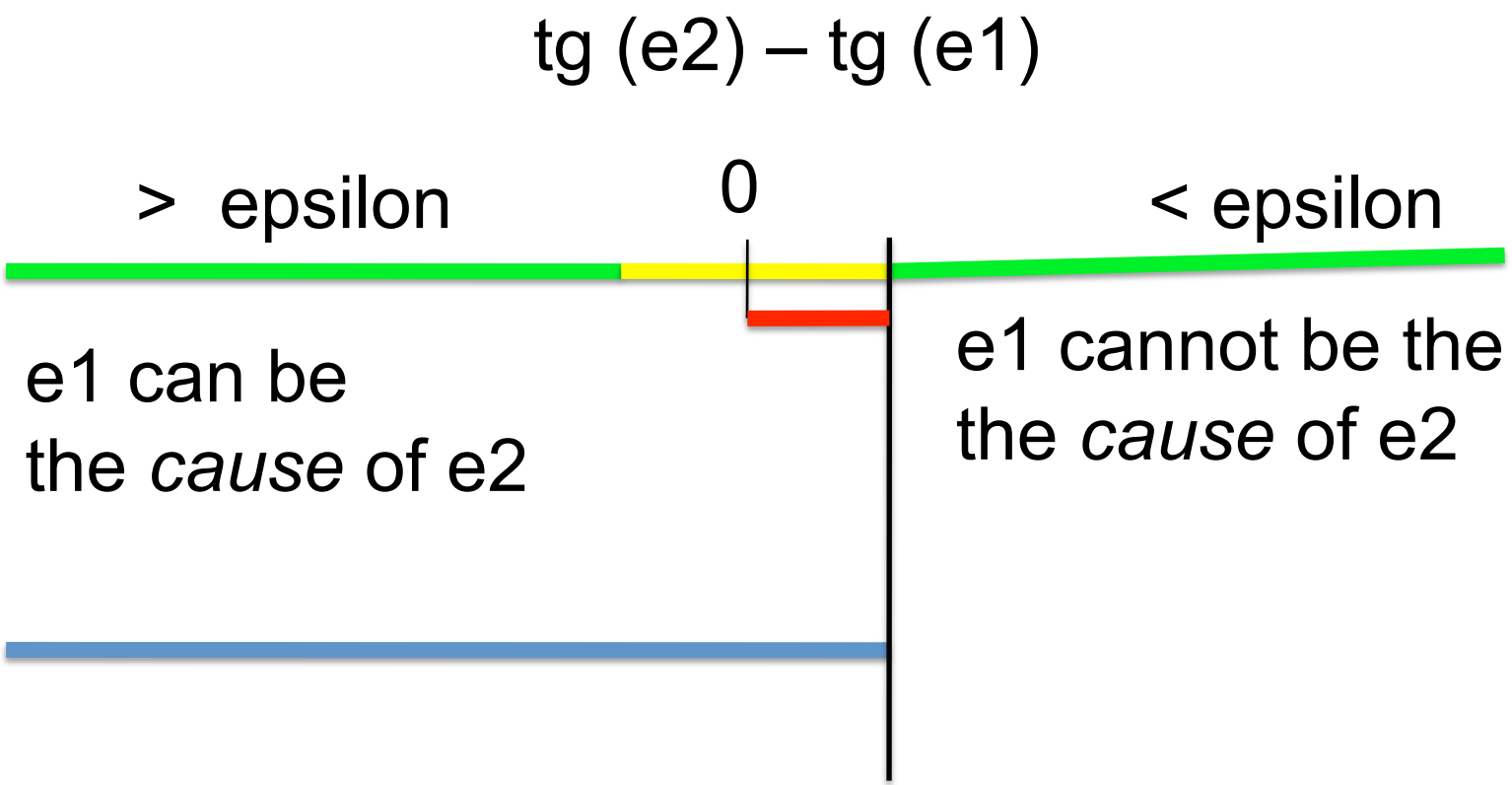


Difference one tick

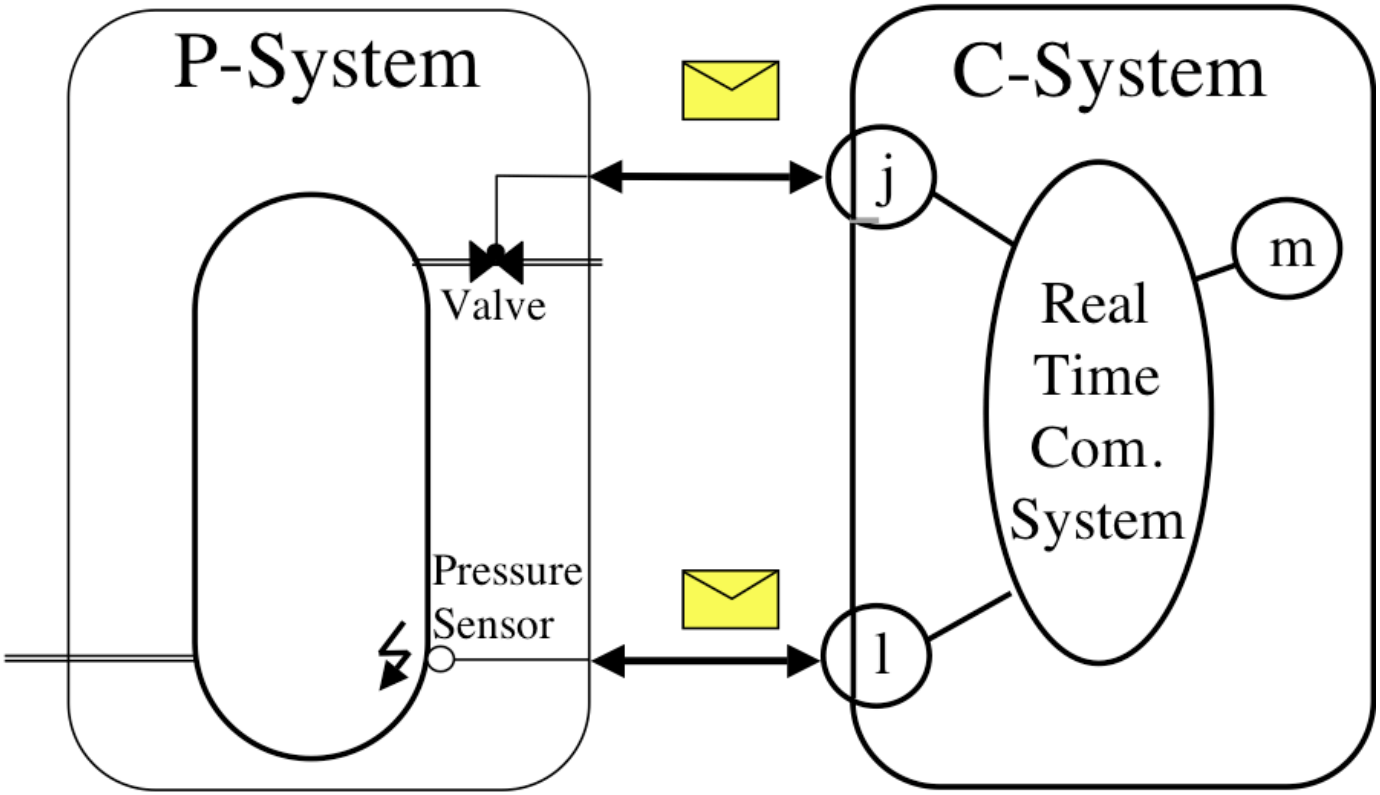
Temporal order reversed

Potential Causality

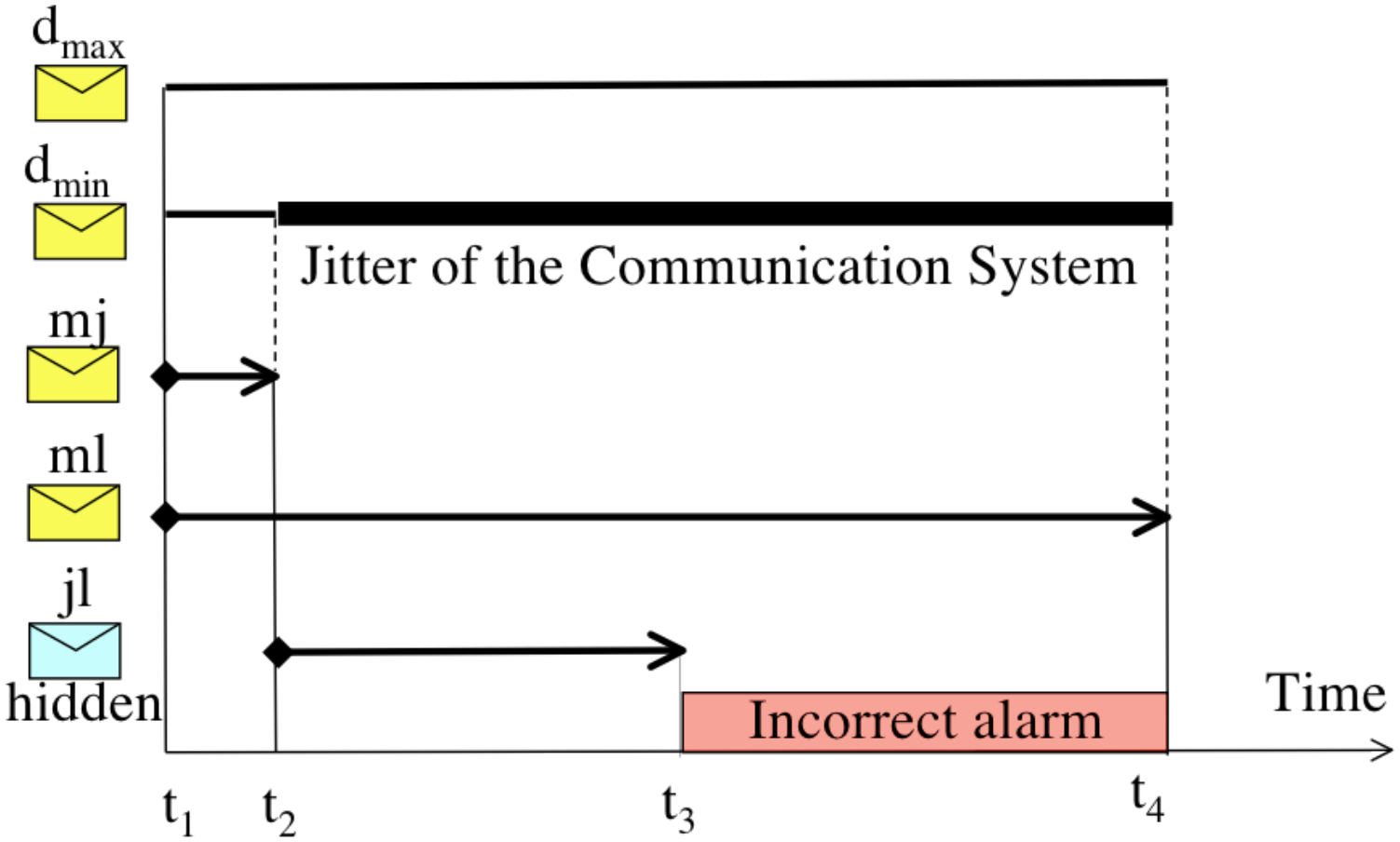
Given two events e1 and e2, can e1 be the *cause* of e2 (potential causality)?



Permanence of Data



Jitter Causes Incorrect Alarms



Permanence

Permanence is a relation between a given message M_i that has arrived at a RT-object O and all messages M_{i-1}, M_{i-2}, . . . that have been sent to this object before (in the temporal order) message M_i.

The message M_i becomes *permanent* at object O as soon as all previously sent messages have arrived at O.

If actions are taken on non-permanent messages, then an inconsistent behavior may result.

The *action delay* is the interval between the point in time when a message is sent by the sender and the point in time when the receiver knows that the message is permanent.

Action Delay

In distributed RT systems without a global time base the

maximum action delay: $d_{max} + \varepsilon = 2\, d_{max} - d_{min}$

In systems with a global time the

maximum action delay: $d_{max} + 2g$

In distributed real time system the maximum protocol execution time and not the "median" protocol execution time determines the responsiveness.

Accuracy versus Action Delay

In a properly designed RT system

$$\textbf{Action Delay} < d_{acc}$$

The accuracy is an application specific parameter, while the action delay is an implementation specific parameter.

What happens if this condition is violated?

Then we need state estimation!

Idempotence

Idempotence is a relation between a set of messages.

A set of messages is *idempotent*, if the effect of receiving more than one messages of this set is the same as the effect of receiving a single message.

Duplicated state messages are idempotent.

Duplicated event messages are not idempotent.

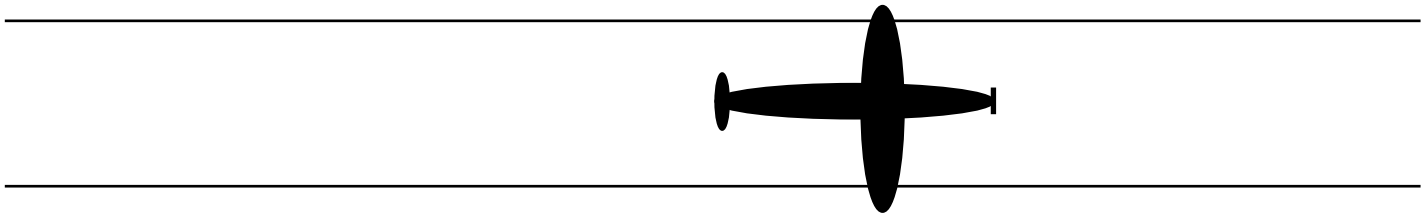
If the idempotence of a set of redundant messages is given, then the design of fault-tolerant systems is simplified.

Replica Determinism

If fault-tolerance is to be achieved by the replication of components, then the replicated component must produce identical results—the results must be *deterministic*.

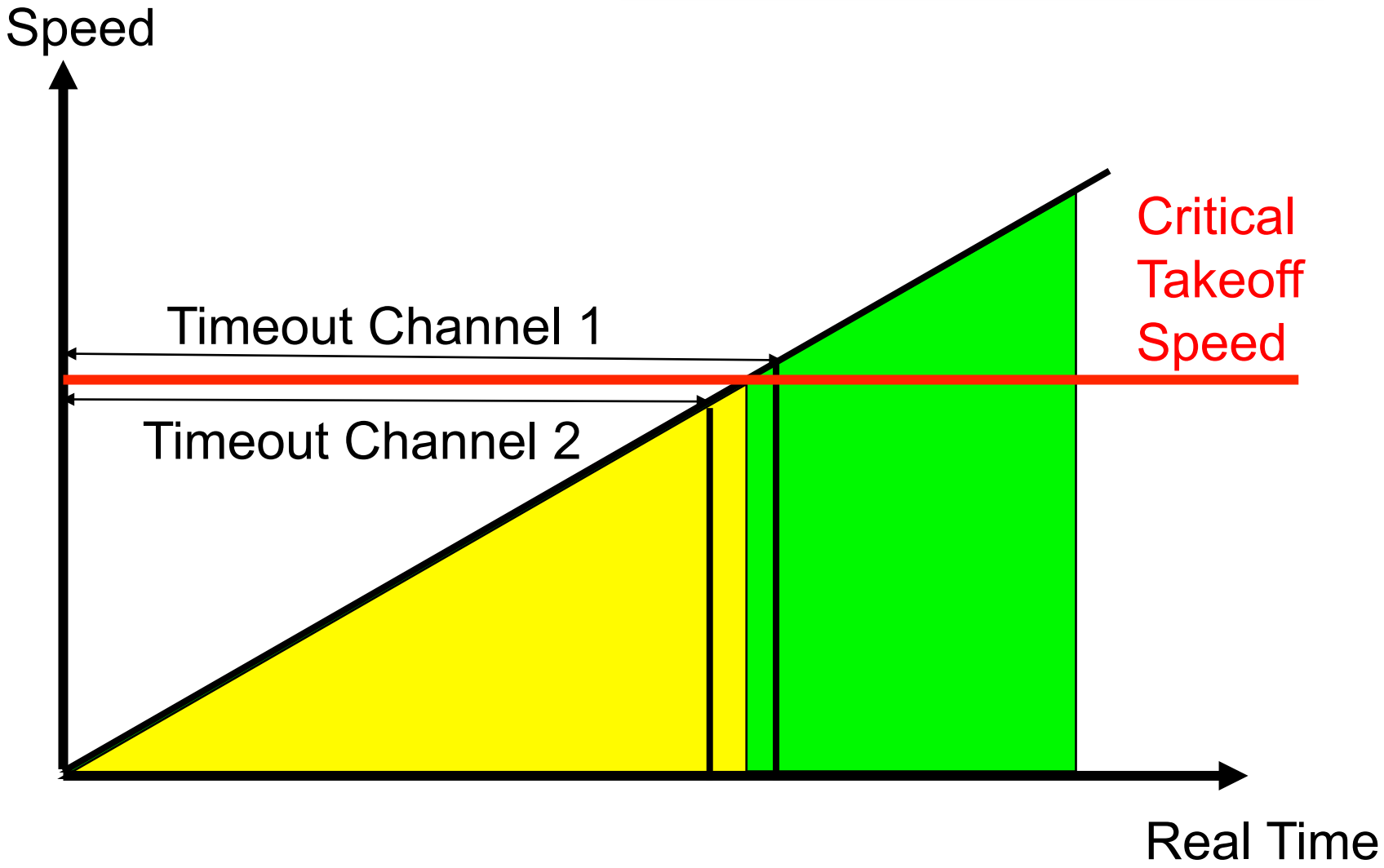
Example: Airplane on Takeoff

Consider an airplane that is taking off from a runway with a flight control system consisting of ***three independent channels***. Consider the system at the *critical instant* before takeoff:



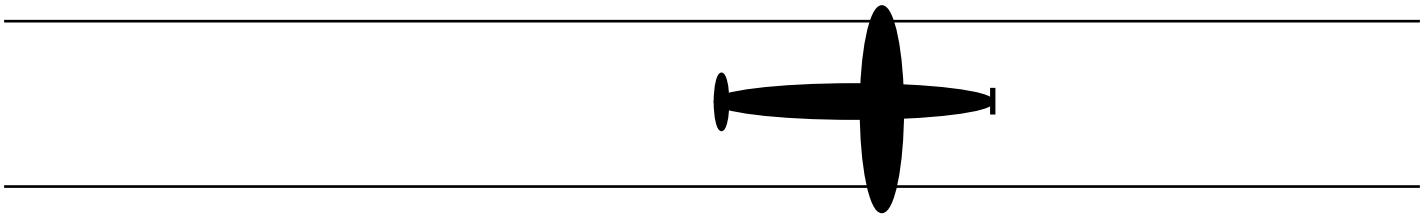
Channel 1 Take off Accelerate Engine
Channel 2 Abort Stop Engine

The Critical Role of Time



Example: Airplane on Takeoff

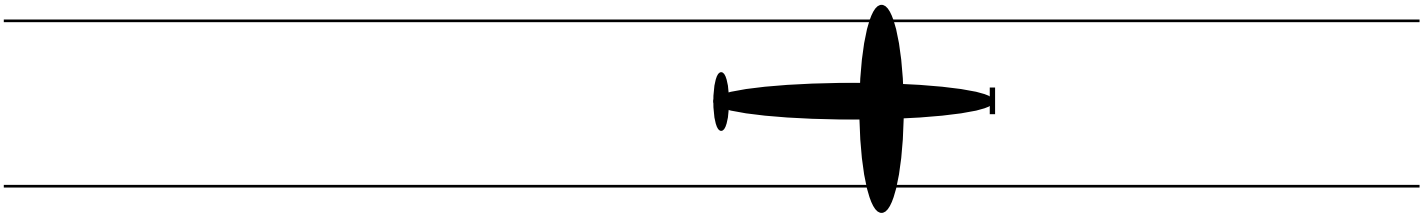
Consider an airplane that is taking off from a runway with a flight control system consisting of ***three independent channels***. Consider the system at the *critical instant* before takeoff:



Channel 1	**Take off**	**Accelerate Engine**
Channel 2	**Abort**	**Stop Engine**
Channel 3	Take off	*Stop Engine (Fault)*

Example: Airplane on Takeoff

Consider an airplane that is taking off from a runway with a flight control system consisting of ***three independent channels***. Consider the system at the *critical instant* before takeoff:



Channel 1	**Take off**	**Accelerate Engine**
Channel 2	**Abort**	**Stop Engine**
Channel 3	Take off	*Stop Engine (Fault)*
Majority	Take off	*Stop Engine (Fault)*

What is Needed to Implement TMR?

What architectural services are needed to implement Triple Modular Redundancy (TMR) at the architecture level?

- Replica Deterministic (*which includes timely*) Operation of the three Lanes
- Provision of an Independent Fault-Containment Region for each one of the Replicas
- Replicated Independent Communication Channels
- Synchronization Infrastructure
- Predictable Multicast Communication
- Support for Voting

Simultaneity: A Fundamental Problem

The ordering of simultaneous events is a fundamental problem of computer science:

- Hardware level: metastability
- Node level: semaphor operation
- Distributed system: ordering of messages

There are two solutions *within* a distributed system to solve the simultaneity problem:

- Distributed consensus--takes real-time and requires bandwidth (atomic broadcast)
- Sparse time

Simultaneity: Who Wins?

In case of simultaneity, one channel is free to decide which message should be transported first.

The other two channels must make the same decision!

This is a two-level problem:

- At first, is must be decided if two events occur simultaneously (difficult).
- Then, in the case of simultaneity, the same ordering algorithm must be executed in all subsystems to achieve consistent behavior (easy).

Definition of *Determinism*: First Try

First definition of *determinism*:

A model behaves deterministically if and only if, given a full set of initial conditions (the initial state) at time t_o, and a sequence of future timed inputs, the outputs at any future instant t are entailed.

This definition of determinism is intuitive, but it neglects the fact that in a real (physical) distributed system clocks cannot be precisely synchronized and therefore a system-wide consistent representation of time (and consequently state) cannot be established.

Determinism: Second Try

Let us assume

- Q *is a finite set of symbols denoting states*
- Σ *is a finite set symbols denoting the possible inputs*
- Δ *is a finite set of symbols denoting the possible outputs*
- $q_0 \in Q$ *is the initial state*
- $t_i, \in N$ *is the infinite set of active sparse time intervals*

then a model (*processing, communication*) is said to behave *deterministically* **iff, given** a sequence of *active sparse real-time intervals* t_i, the initial state of the system $q_0(t_0) \in Q$ *at* t_0 *(now),* and a sequence of *future* inputs $in_i(t_i) \in \Sigma$ **then** the sequence of *future* outputs $out_j(t_j) \in \Delta$ and the sequence of future states $q_j(t_j) \in Q$ is *entailed.*

Agreement Protocols

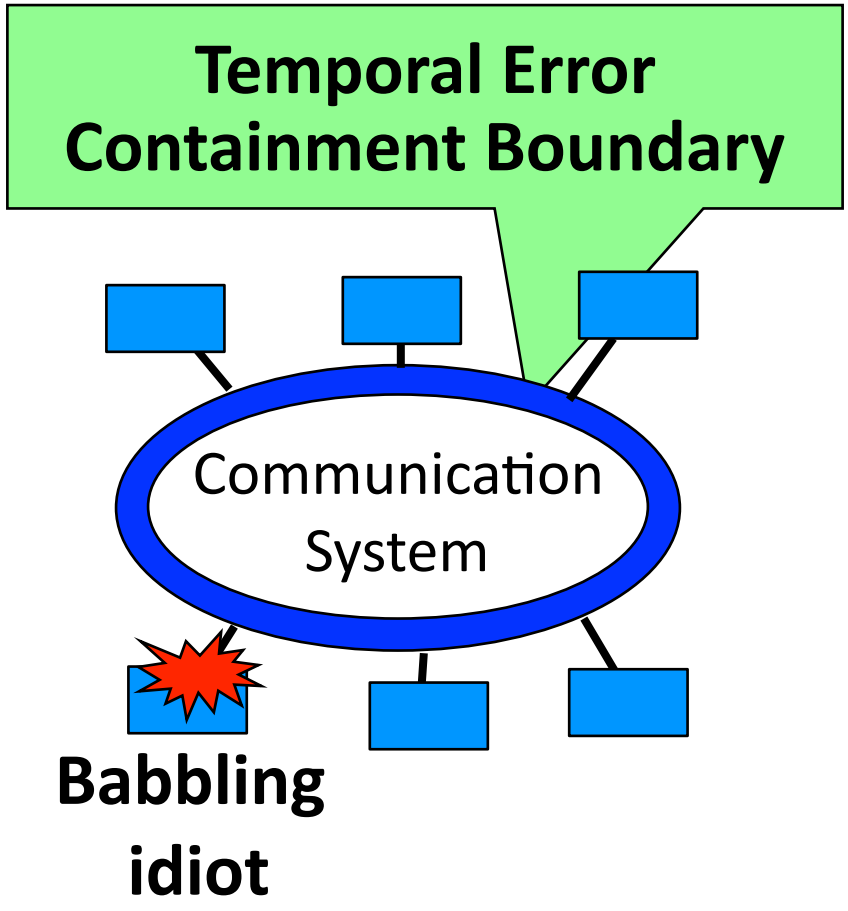
Due to the temp*oral uncertainty*--the limited faithfulness of our models—we cannot assume that two independent observers will come to the same conclusion about the simultaneity of events in the real world.

In order to arrive at a consistent view within the replicated computer systems, we must execute an agreement protocol at the interfaces between the real-world and the computer systems to establish a consistent order of events (although this consistent order might be different from the *temporal order* in the real world).

Temporal Error Containment by the CS

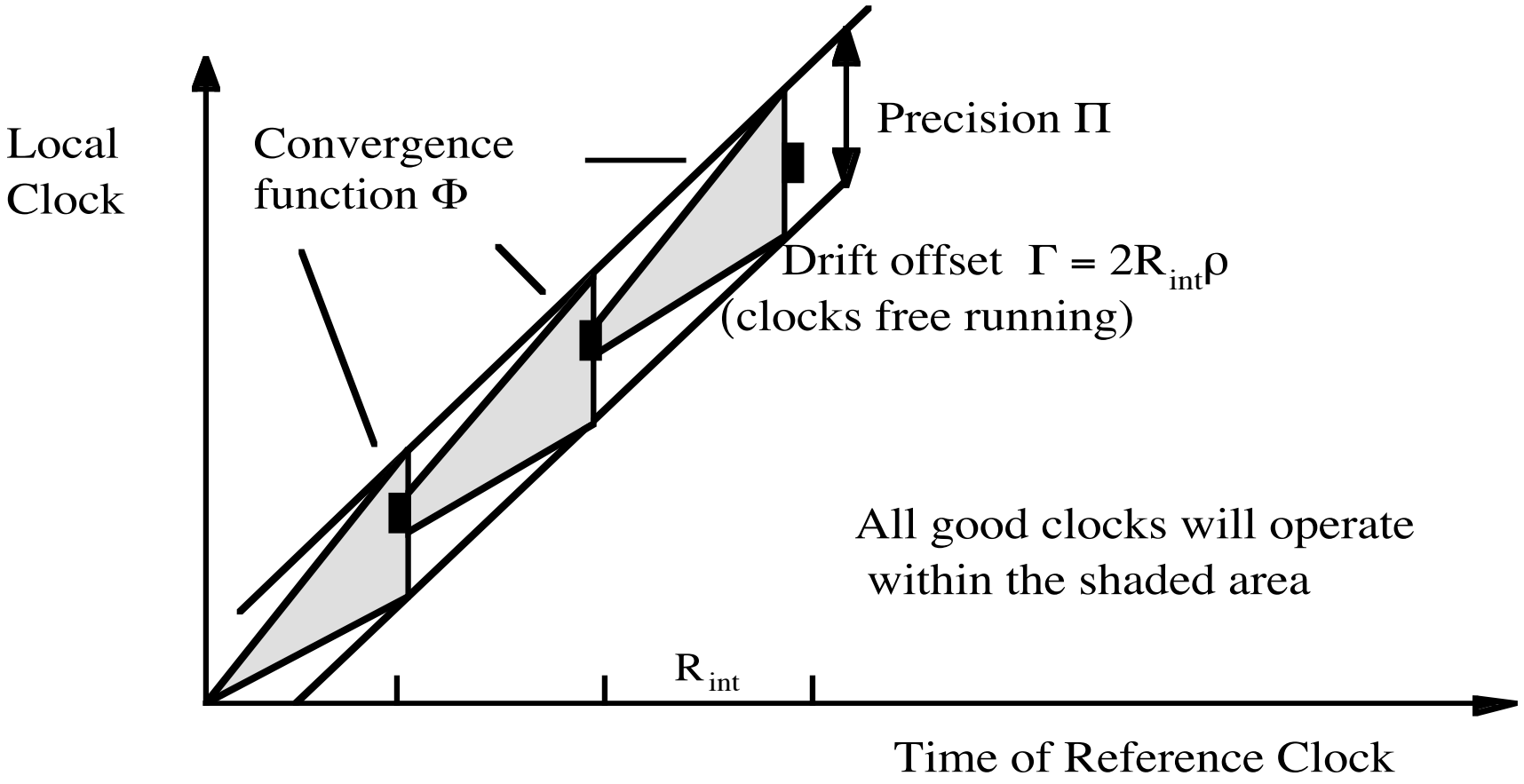
It is ***impossible*** to maintain the communication among the correct components of a RT-cluster **if the temporal errors caused by a faulty component are not contained**.

Error containment of an *arbitrary temporal node failure* requires that the Communication System is a self-contained FCU that has *temporal information* about the allowed behavior of the nodes-- it ***must contain application-specific state***.



The Synchronization Condition

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



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:

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

Distributed Clock Synchronization

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

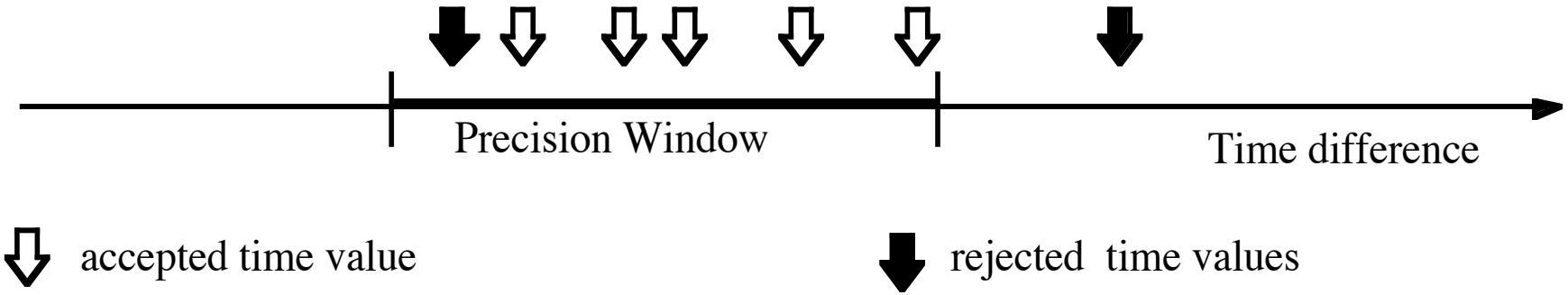
- Every node acquires knowledge about the state of the global time counters in all other nodes by message exchanges among the nodes.
- Every node analyzes the collected information to detect errors and executes the convergence function to calculate a correction value for the node's local global time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

Fault-Tolerant Algorithm

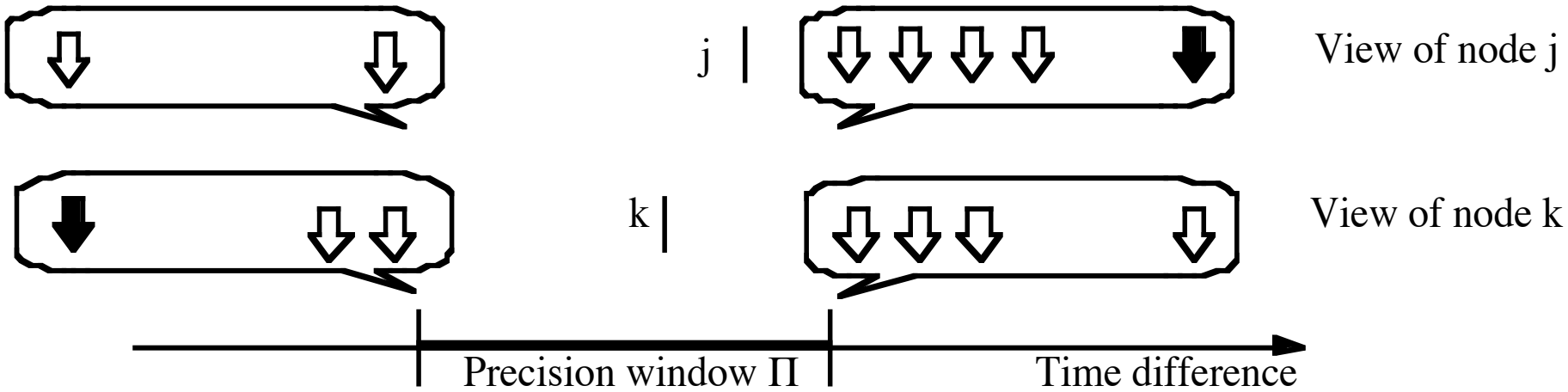
Every node measures the time differences between its own clock and all other clocks and rejects the k extreme differences, where k is the number of Byzantine faults that are to be tolerated.

If k=1, then



Fault Tolerant Average Algorithm

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



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

Some Impossibility Results

- It is impossible to precisely synchronize the clocks of a distributed computer system.
- It is impossible to decide, whether two events that occur in the controlled object have occurred simultaneously.
- It is impossible to describe the *state* of a physical process without a reference to physical time
- It is impossible to contain errors in a distributed computer system, if the shared communication system is not aware of the permitted temporal behavior of the nodes.
- It is impossible to contain a malicious error if there are less than four clocks (no authentication of messages)

Conclusion

- In embedded system design, physical time is a *first order citizen* that must be part of any realistic system model.
- A proper model of time, such as the sparse time model, can be a foundational element of the solution domain.
- The faithfulness of the temporal properties that are expressed in the model depends on the precision of the of the gobal time.
- Do not try to solve the *impossible*!