Real-Time (Modeling)(1)

Synchrony vs. asynchrony

Synchronous system model:
An algorithm or protocol is *synchronous* if it is possible to bound its action delays (processing and networking).

Examples for bounds are:
- processing delays have a known bound
- message delivery delays have a known bound
- rate of drift of local clocks has a known bound
- divergence between local clocks has a known bound

→ decisions can be taken based on the passage of time, e.g. message loss

time-bounded delivery (synchrony for messages):
Any message delivered is delivered within a known bound T_{Dmax} from the time of the send request

delivery time ($t^p_D(m)$):
interval between the *send (m)* event of message m and the *deliver$_p$ (m)* event at node p

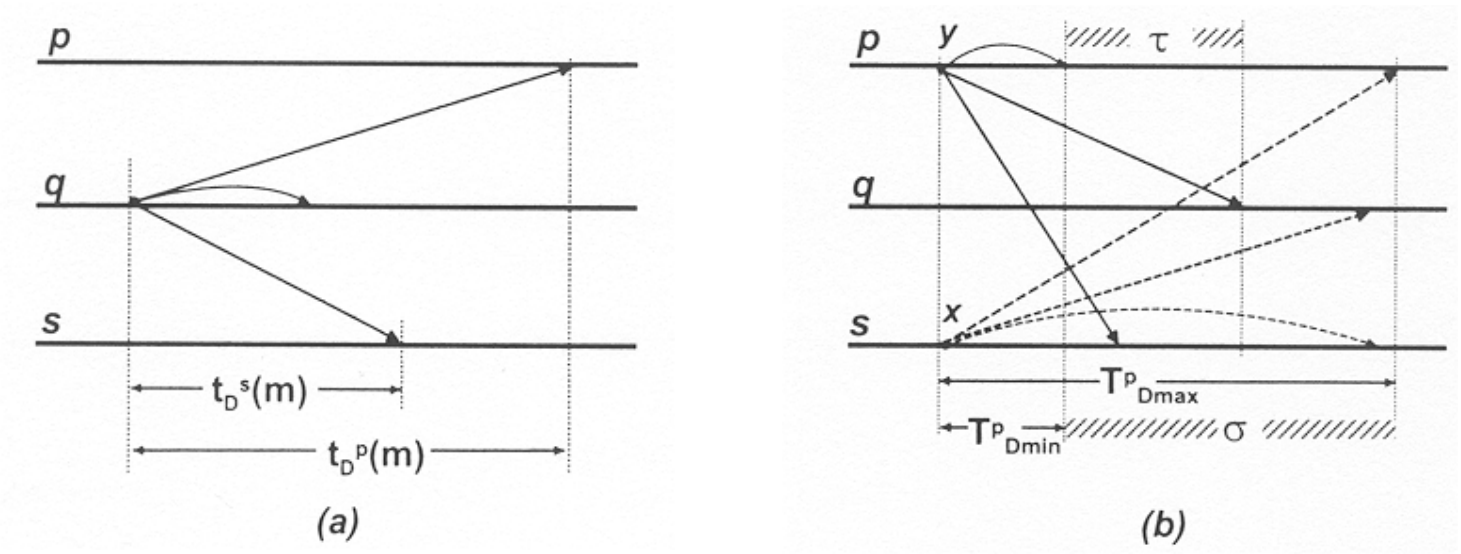
Con:
possible that either the synchrony of the system or the worst-case load scenarios of the application are difficult to determine --> the system may function incorrectly (lack of *coverage).*

Real-Time (Modeling)(2)

steadiness (σ):
greatest difference (max_p) between the maximum (T^p_{Dmax}) and minimum (T^p_{Dmin}) delivery times observed at any participant p.

tightness (τ):
greatest difference ($max_{m,p.q}$), for any message m, between ($t^p_D(m)$) and ($t^q_D(m)$), for any p,q.

Examples:



Note:
Achieving synchrony means guaranteeing timeliness properties (real-time system operation).

Real-Time (Modeling)(3)

Asynchronous system model:
Such systems are *time-free*, i.e.time is not existent in that model ---> Bounds do not exist!

Pro:
simple, adapted to environments giving very little guarantees

Con´s:
- no worst-case time bound for the duration of an execution (tasks, processes, RPC etc) can be guaranteed
- impossible to reach consensus if faults occur

Summary:
RT systems is materialized by t*imeliness specifications* ---> need for a synchronous system model
Synchrony is hardly to establish in large-scale, unpredictable and unreliable infrastructures
---> correct operation of the system is very difficult to achieve
Asynchronous models do not allow timeliness specifications.

Real-Time (Modeling) (4)

Problem:
What system model to use for applications with timing requirements running in environments with uncertain timeliness?

Possible solution:

Partial Synchrony models
Idea:
Relaxing the strong assumptions by considering that the system is not (a)synchronous *always or everywhere*
This complies to application environments where

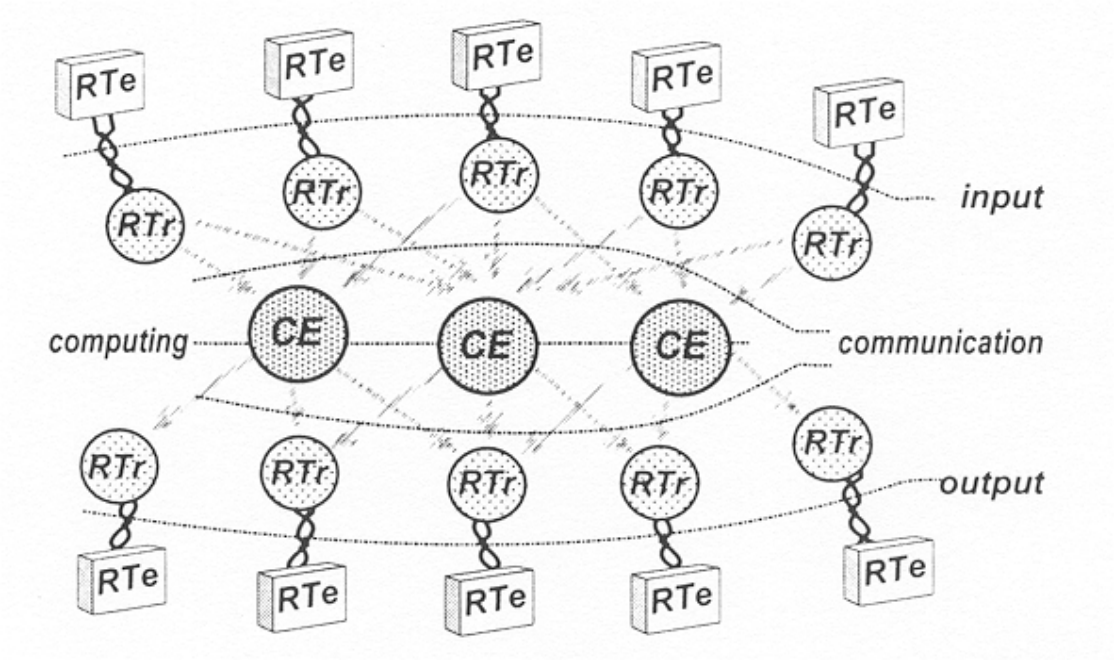
- synchronism is not a homogeneous property
- worst-case termination times or communication delays are hard to determine or totally different at different times

--->
timeliness parameters (e.g. processing or message delays, clock drifts) can be defined partly, i.e. hold on a subset of the system, or during limited periods of its operation. However, phases of asynchrony must be detected!

Real-Time (Modeling) (5)

A Generic Distributed RT System Model:

The model relies on the separation between input/output, communication, and computing.



This separation reduces the complexity of (simplifies) system design, which can be done in 3 separate steps:

- Ensuring that the RT_r correctly represents its Rt_e in all situations
- Ensuring that the communication mechanisms guarantee timely information flow between CE´s and RT_r´s
- Ensuring that the computing mechanisms produce correct (including timely!) results (outputs)

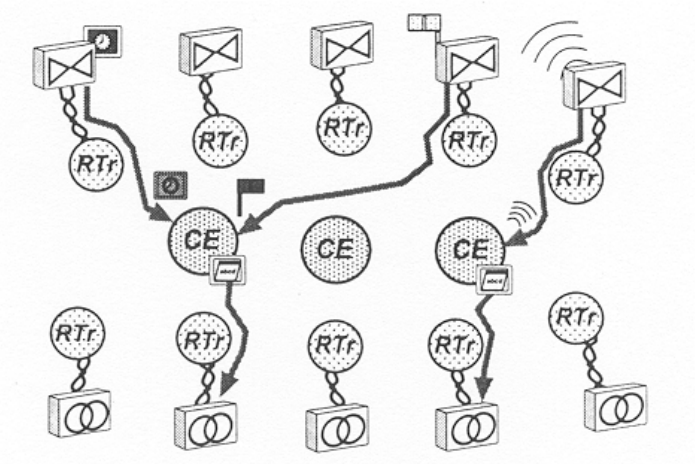
Real-Time (Modeling) (6)

1. The Event-Triggered Approach

Event-triggerd system:

One that reacts to significant events immediately and directly, meaning that the information flow retains an event-like nature up to the center of the system (CE´s)

Event-Triggered Architecture:



Properties:

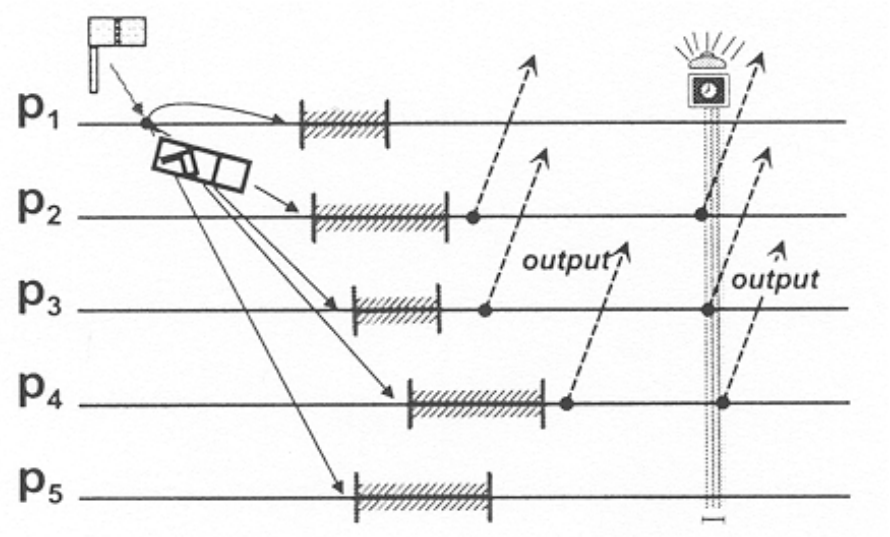
- definition of the operational envelope is by nature not rigid --> flexible w.r.t. overload and the unexpected
- irregular event distributions (sporadics, bursts, event showers) make predictability hard to guarantee
- on-line, preemptive priority-based scheduling of both periodic and sporadic tasks
- allows selective dissemination of information and allocation of resources (priorities, graceful degradation) in overload situations
- scalable

Real-Time (Modeling) (7)

Event-triggerd protocols:

default state is “idle”; on occurrence of an external event, it is transformed into an *event message* or a *message interrupt* sent to the center of the system.

Event-Triggered Timing behavior:



Properties:

- processing and outputs are event-driven ---> results in high jitter
- actuations can be triggered by (synchronized) clocks

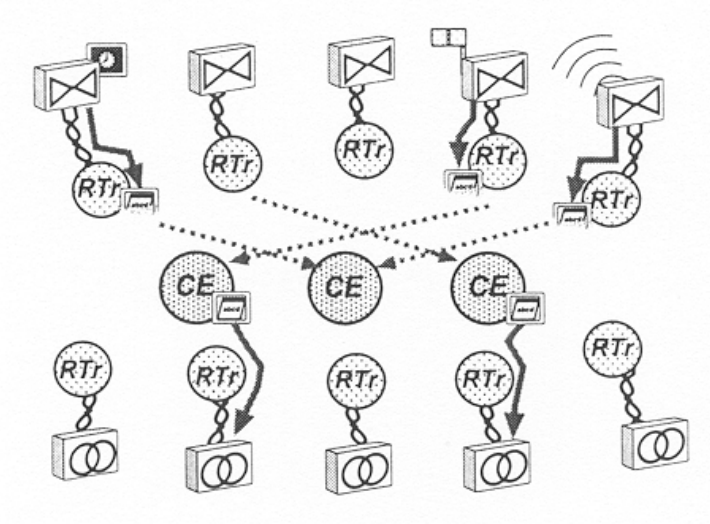
Real-Time (Modeling) (8)

2. The Time-Triggered Approach

Time-triggerd system:

One that reacts to events by transforming them into states at the representatives, and sending them to the CE´s at prespecified instants of time

Event-Triggered Architecture:



Properties:

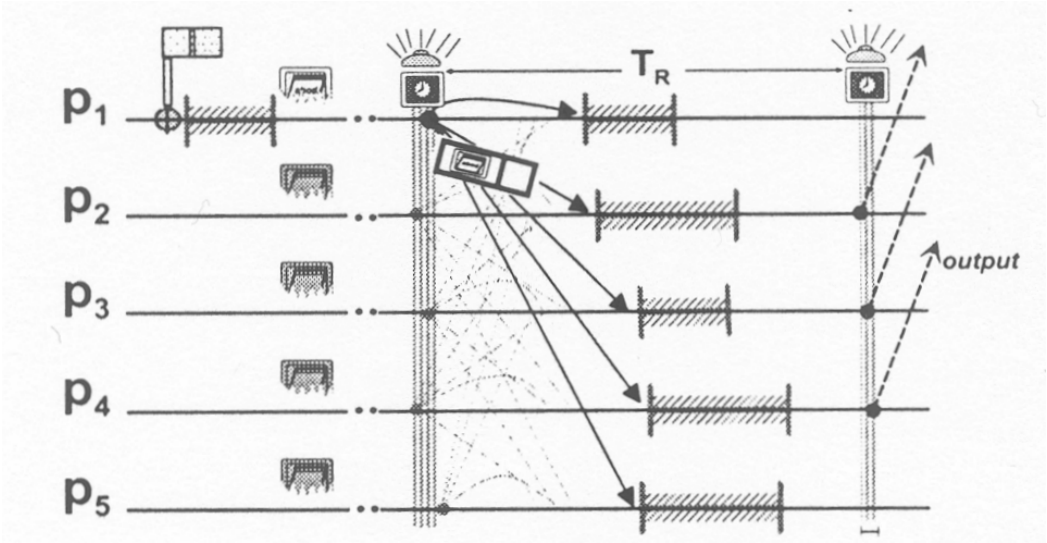
- regular (static, periodic) information flow betw. RT_r´s and CE´s --> resource reservation, no flow control
- predictability is easily achieved(TT response time only depends on the period) and simpler to verify
- schedule can be calculated off-line and it is static
- not scalable (increasing node number or adding new tasks needs redesign of node slots, communication, and scheduling) and cannot handle unexpected events

Real-Time (Modeling) (9)

Time-triggerd protocols:

Activity is triggered at environment-independent instants. The period in-between must be short enough to match the event arrivals and long enough w.r.t. dissemination and processing

Event-Triggered Timing behavior:



Properties:

- dissemination of inputs and outputs are time-driven at begin and end o f the period T_R
- system is always working at full load

Real-Time (Modeling) (10)

3. RT Communication Modeling

In order to *configure* the network, it needs to

- model the *traffic pattern,* belonging to the types aperiodic, sporadic, periodic
- separate traffic into *latency classes* according to urgency, associating corresponding latency bounds
- abstracting from physical particularities of the real network in order to achieve portability

Example: Abstract LAN Properties

Broadcast	correct nodes receiving an uncorrupted frame transmission, receive the same frame
Error Detection	correct nodes detect any corruption done by the network in a locally received frame
Network Order	any two frames received at any two correct nodes, are received in the same order at both nodes
Full Duplex	frames transmitted are also received at the sending node
Bounded Omission Degree	in a known time interval T_{rd}, omission failures may occur in at most k transmissions
Tightness	correct nodes receiving an uncorrupted frame transmission, receive it at real time instants that differ, at most, by a known interval T_{tight}
Bounded Transmis. Delay	any frame is transmitted by the network within a bounded delay T_{TXmax} from the send request

Real-Time (Modeling) (7)

4. RT Control

Control, historically, is the main application of RT systems. There are mainly two types:

- *continuous control:*

  guaranteeing that each of a certain number of variables stays within a maximum error from their pre-defined values, the so-called set-points

- *discrete control:*

  guaranteeing that a pre-defined sequence of actions take place at the appropriate times, dictated either by a pre-defined static schedule, or in reaction to external events, or both.

RT control, in essence, implies the solution of the following problems:

1) timely and correct observation of RT entities of the environment

2) timely and correct computation of the action to be executed next

3) timely and correct actuation of RT entities of the environment

ad 1) relates to determining a value at a given time or determining the time at which a given value shows up.

ad 2) addresses three aspects:

- the control algorithm must be logically adequate to the problem (semantically correct)
- observations must be used correctly over time (production of results within the required response time)
- scheduling of the necessary tasks must meet the given deadlines

ad 3) deals with the ability to position an action at a given point in time

Real-Time (Modeling) (8)

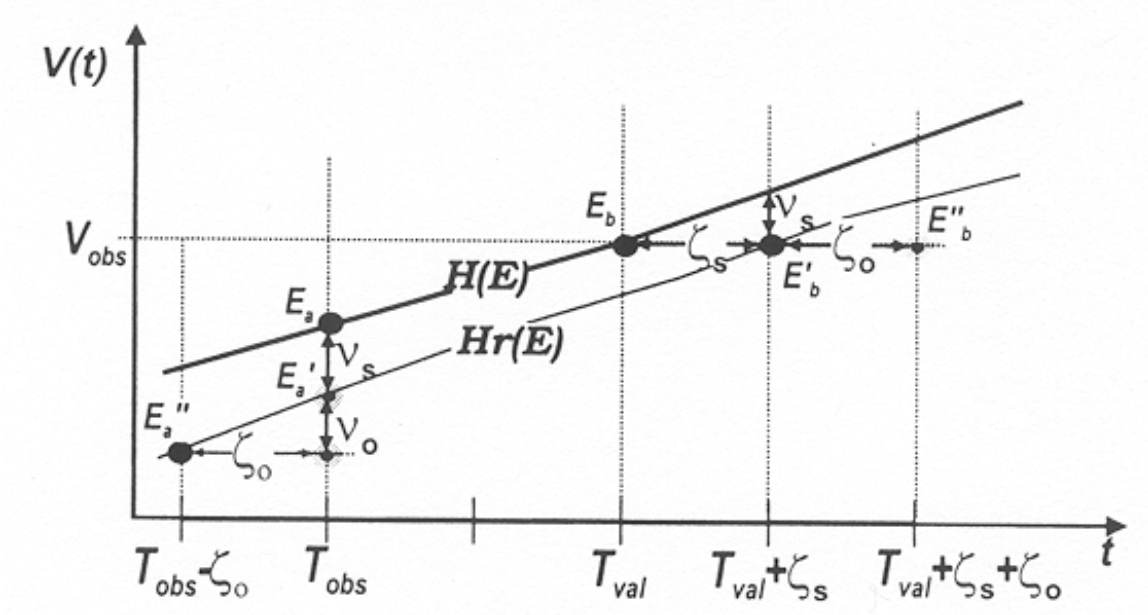
Quality of Control is affected by two time-domain factors, namely *response time* and *jitter*.

Fixed delays may be compensated by prediction or extrapolation, the remaining error due to jitter cannot.

It accumulates in the several phases of the control cycle:

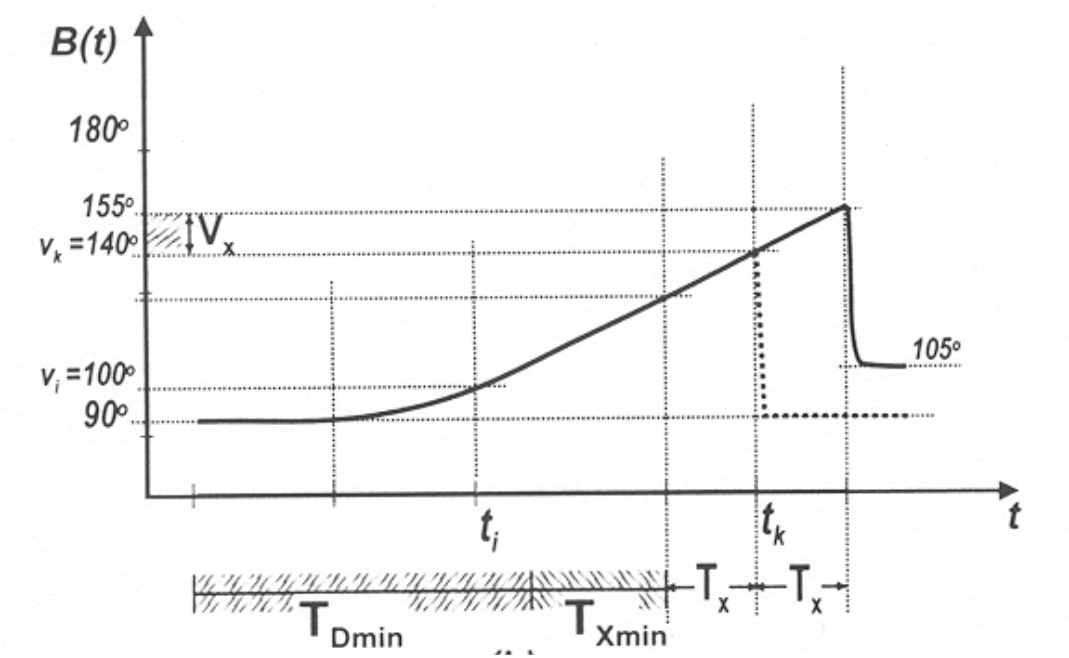
- observation jitter
- communication and execution time jitter
- positioning jitter of the actuation

Time of a Value



Real-Time (Modeling) (9)

Value over Time



The total timing error of the control loop is bounded by

$T_{ctl} =: T_o + T_a + T_x$

The bound on accumulated timing errors maps, for continuous variables, into a predictable value error bound

$V_{ctl} =: V_o + V_a + V_x$

Real-Time (Modeling) (10)

Distributed observation introduce two relevant problems:

- what is the minimum separation of observed events that can be ordered by global timestamps?
- what must be the minimum difference between the timestamps of two distributed observations such that their mutual physical order, i.e. the order of the events in the real time domain, can be determined?

Minimum separation is determined by the virtual granularity g of the virtual clock ticks.

Obviously, g must be larger than the precision π of the used synchronization algorithm, i.e. $g > \pi$

---> distributed local timestamps of the same global event are at most one tick apart

---> any two events with timestamps differing by at least 2 are ordered correctly

---> if the difference < 2, the physical order of the two events cannot be deduced from their timestamps

Real-Time (Modeling) (11)

5. RT Databases

In addition to regular data bases, RTDB´s have a time-value nature, i.e. their correctness depends on time.

In temporal terms, two problems have to be solved:

- keeping the value of each database item (temporally) consistent with its RTe
- keeping the values of a collection of database items mutually consistent

RT Transactions

In functional terms, depending on the action performed, RT transactions can be categorized in:

- write-only
- read-only
- update

Note: Temporal correctness might affect conventional solutions for logical correctness
(e.g. commit, abort of transactions)