Real-Time (Basics) (1)

What is time? Which time?

There may be several views of time in a distributed system:

- time amongst the several nodes (sites)
- time between a site and its users
- time between the system (controller) and its environment (controlled system

The "Real-time" problem:

Essentially, we need to synchronize system actions with the environment having its own pace (real time) and react in accordance to the evolution of the environment

Embedded Computer System

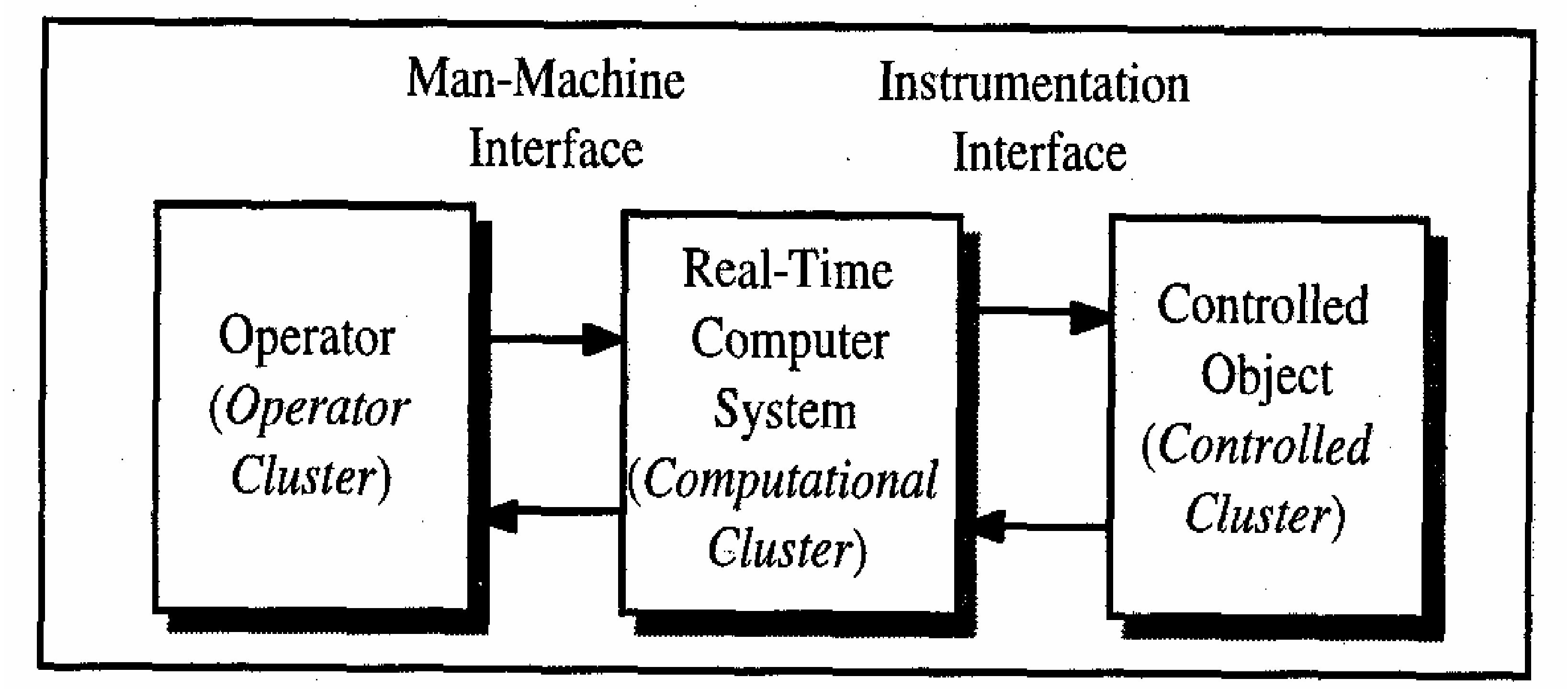
any computer system, made on purpose for a specific application, having no separate interface to the user (black box).

Most embedded systems are real-time systems and vice versa:

- oven (microwave) controller
- traffic lights control system
- driving (flight) control system (drive (flight)-by-wire)
- robot control systems
- multimedia computer game system
- virtual reality system

Real-Time (Basics) (2)

Coarse-grained block diagram of a real-time system:


Man-Machine Interface
Instrumentation Interface
Operator (*Operator Cluster*)
Real-Time Computer System (*Computational Cluster*)
Controlled Object (*Controlled Cluster*)

Real-Time (Basics) (3)

RT System
system whose computational progress is specified in terms of timeliness requirements dictated by the environment.

This definition implies properties and descriptions that sometimes are also used as definitions:

Any computer system where a *timely* response by the computer *to external stimuli* is *vital* is a real-time computing system.

Any computer system where the correct behavior of these systems depends not only on the value (result) of its computations, but also on the time at which the results are produced. As a consequence, a substantial fraction of the effort to design it goes into making sure that deadlines are met.

Any computer system providing at least one *real-time service*
---> RT and non-RT services (jobs) may coexist in the same system!

Examples of real-time services are:
- periodic read of a sensor
- activate a valve at a precise instant
- reply to a request or deliver a message in bounded time
- execute a task within a given interval

Real-Time (Basics) (4)

Classes of RT systems (due to varying constraints w.r.t. matching the timing requirements of the environment):

- *hard:* timing failures have to be avoided
  example: drive (fly)-by-wire

- *firm (mission-critical):* timing failures should be avoided , if occasionally not, exception handling is done successfully
  example: air traffic control system

- *soft:* occasional timing failures can be accepted
  example: on-line flight reservation system

Deadline: The instant at which a result (task) must be produced (executed)

Typically, real-world applications include hard and soft activities (services).

---> A hard RT system should be designed to handle adequately both hard and soft tasks.

---> Its objective should be to guarantee the individual timing constraints of the hard tasks while minimizing the average response time of the other tasks.

Real-Time (Basics) (4a)

Example: Driving a car

Some of the services (tasks) are hard, with varying deadlines depending on the environment, e.g. steering and braking.

Some are soft, like updating speed control.

Some even are non-RT , like tuning the radio.

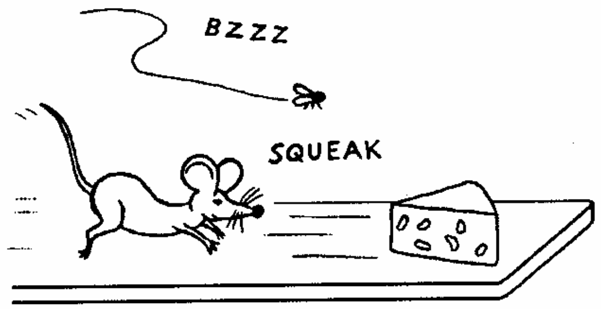
Real-Time (Basics) (5)

Misconceptions about the very nature of RT systems

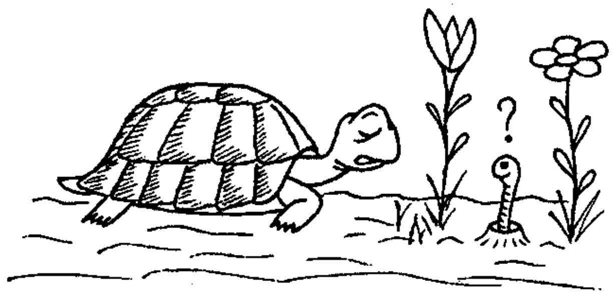
- RT systems = Fast systems

The word real indicates that the reaction of the computing system to events from its environment must conform to the needs of the environment.

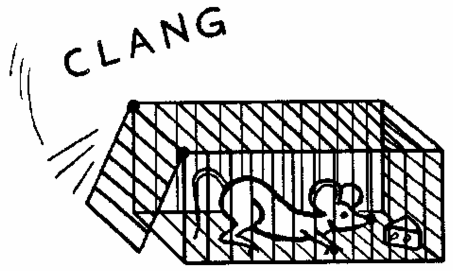
Analogy to nature (animal plays the role of a computer system):



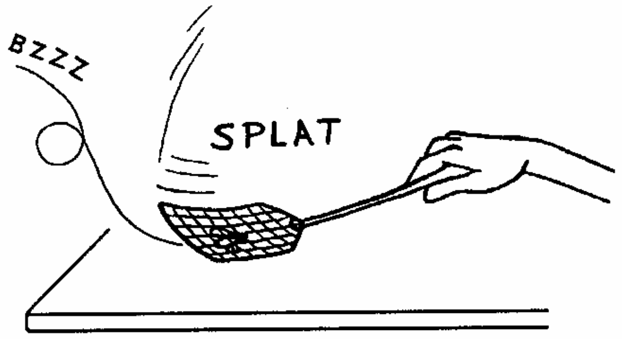
(a)



(b)



(c)



(d)

Real-Time (Basics) (6)

- Advances in computer hardware will take care of RT requirements

Advances in computer hardware will increase the computational speed in terms of millions of instructions per second (MIPS)

Reality: The more power we have, the more is spent elsewhere

- Wintel saga: The more powerful Pentiums become, the more power-hungry MS software gets
- Amdahl´s law: Perfect (linear) speedup in multiprocessor systems is not possible.

- RT = high performance, high throughput, low average response times

However, when several tasks have different timing constraints, average performance has little significance for the correct behavior of the system.

It is worth thinking about this little story:

“There was a man who drowned crossing a river with an average depth of 50 cm”

---> high performance does not meet the main property of RT systems, namely, to guarantee that the individual timing constraints of each hard tasks being executed will be met in **all possible circumstances.**

---> real-time *is not* about performance, it is about **predictability**

Real-Time (Basics) (7)

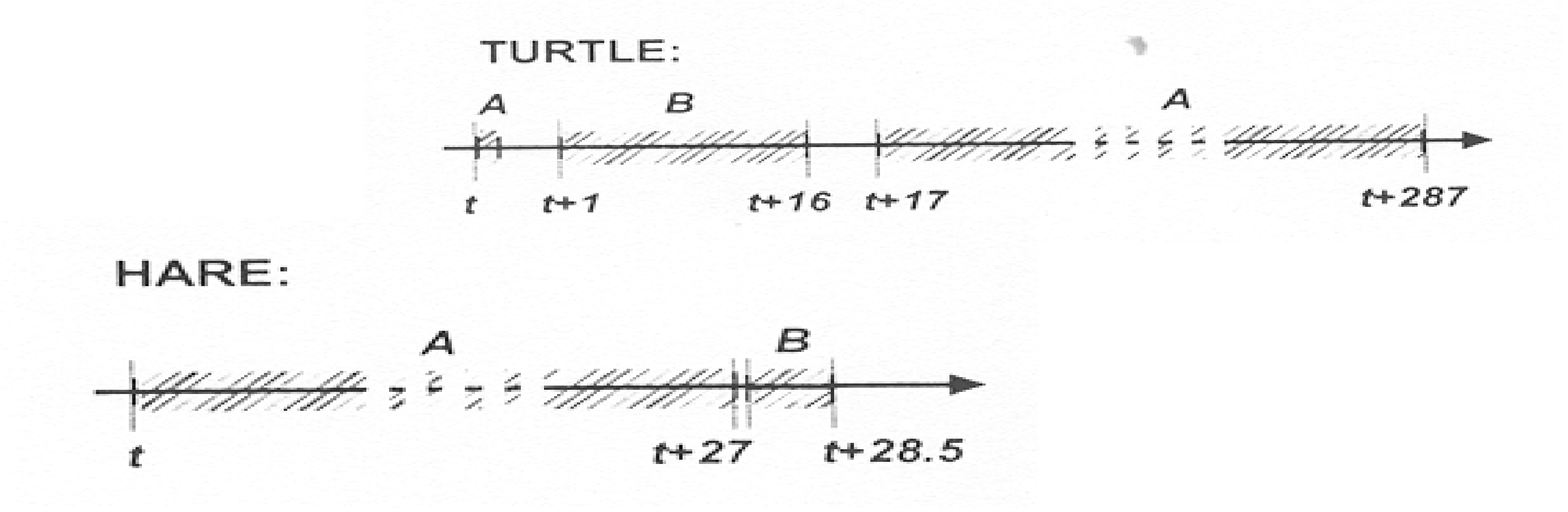
RT is mainly about **scheduling** needed resources such that deadlines are met

Analogy to real life: La Fontaine´s Fable of the Hare and the Turtle

Slow system TURTLE: relative speed s=1, context switch delay c=1, scheduling policy EDF
Fast system HARE: relative speed s=10, context switch delay c=0,01, scheduling policy FIFO

Mission: Executing tasks A and B where B arrives later, but at approximately the same time
Timing parameters for A: arrival time t, execution time X_A= 270, deadline D_A = t + 290
Timing parameters for B: arrival time t, execution time X_B= 15, deadline D_B = t + 28

Will the systems make it in time?



Real-Time (Basics) (8)

Main topic of current research:

How to ensure that (distributed) systems are timely, i.e. behave predictable under a number of unpredictable circumstances, including uncertainty, overload, faults?

The distinguishing property of RT systems is called ***p r e d i c t a b i l i t y.***

---> RT systems must be able to cope with

uncertainty
some timing parameters (execution times, latencies) may vary or even be unknown

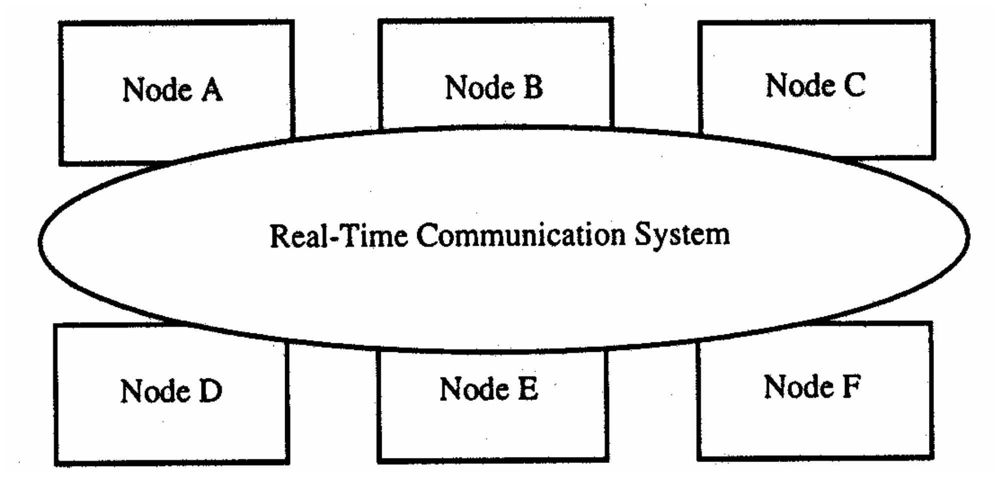
peak load
RT systems must be designed to manage all possible load scenarios including overload.

Faults
Critical components of the system have to be designed to be fault-tolerant.

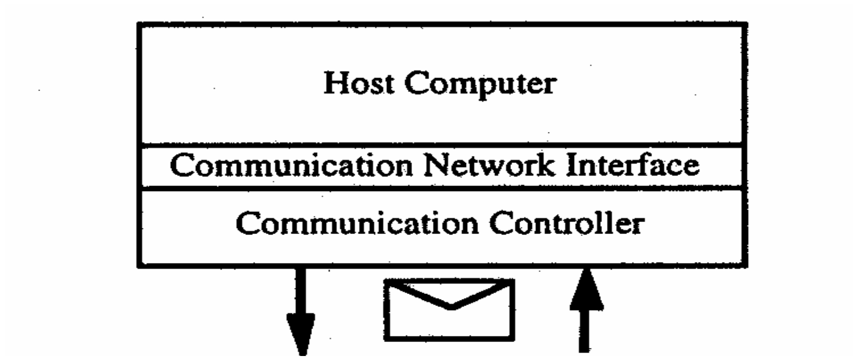
Real-Time (Basics) (9)

If **distributed,** timeliness guarantees have to be provided across all nodes interconnected by a network!

Distributed RT System:



Structure of a node:



Roadmap

Contents:

Real-Time paradigms (concepts) including

- Temporal and embedded systems specifications
- Time, Clocks, and clock synchronization
- **Scheduling the main computer resources**
  - CPU: Task scheduling
  - Memory: memory access protocols
  - communication network: Real-time communication protocols
- Real-Time Modeling
- Real-Time Platforms