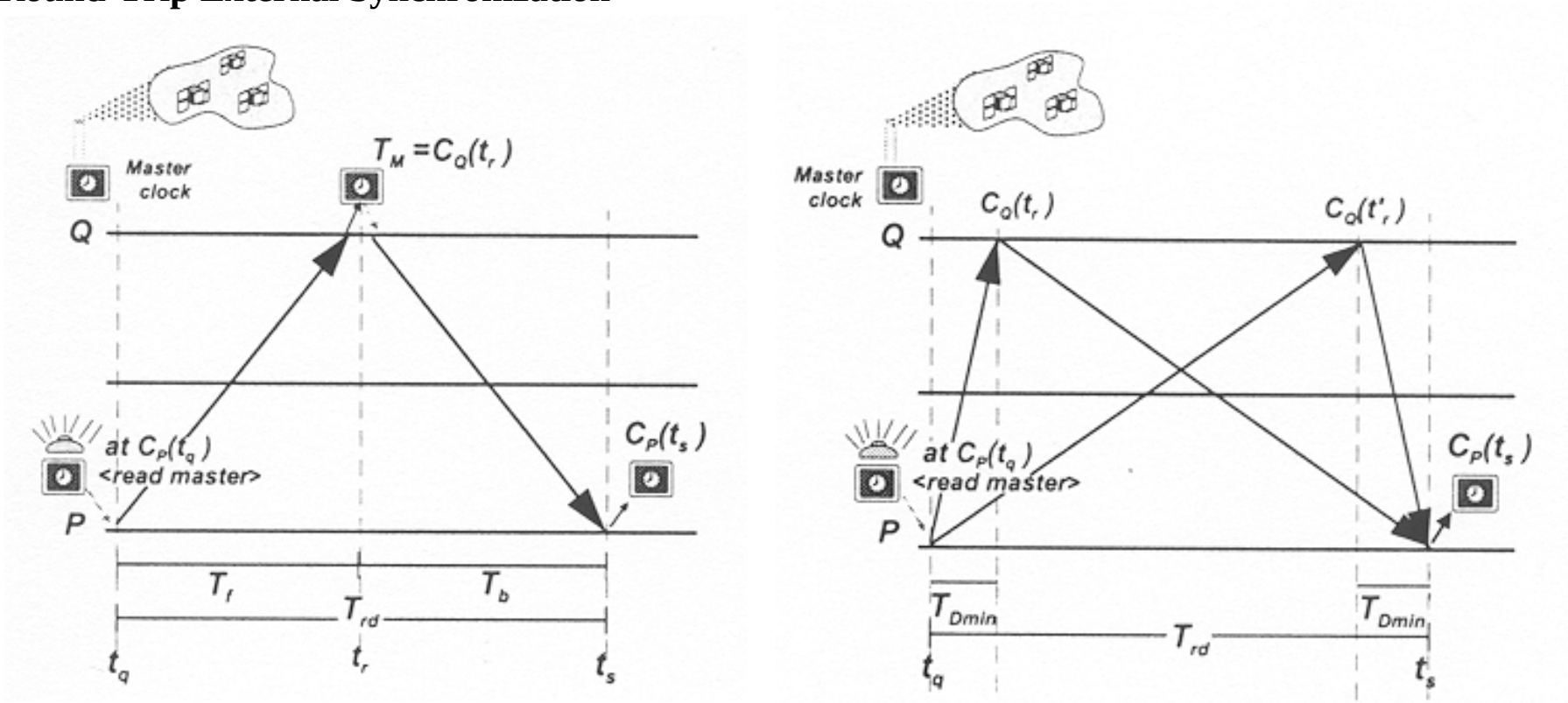
Real-Time (Paradigms) (16)

External Synchronization

Respective algorithms are not cooperative, but master-slave.

Simplest method: Multicasting of time by the master (used to synchronize GPS receiver units)

Round-Trip External Synchronization



Real-Time (Paradigms) (16a)

5. Input/Output

It deals with the *observation* of, and the *actuation* on, the environment performed by sensors and actuators.

Observation:
The act of acquiring and pre-processing the state of a RT entity, through one or more sensors.

It can be done by
- *sampling*
- *polling*
- *interrupt*

Actuation:
The act of issuing and post-processing a command to change the state of a RT entity, through one ore more actuators

It can be triggered
- *immediate*
- *deferred*
- *periodic*

Real-Time (Paradigms) (17)

7. Processor Scheduling

Scheduling is concerned with assigning needed resources in order to execute tasks such that the system meets the timing requirements. Scheduling is the backbone of a RT system and, therefore, is the most widely researched topic within RT systems.

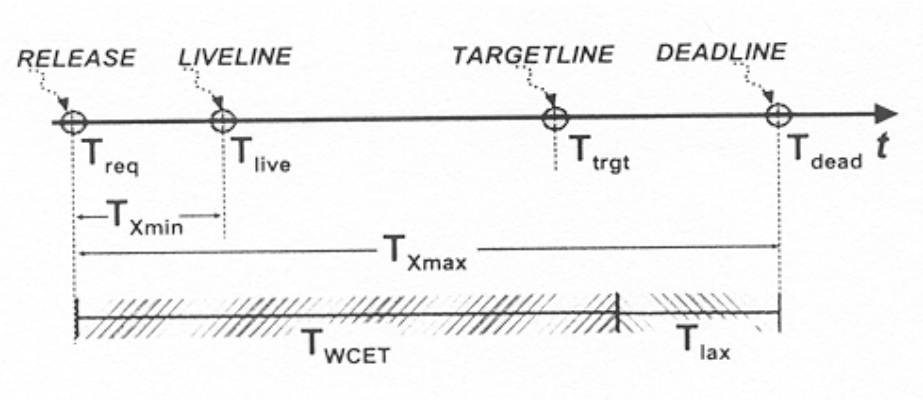
Policies of Non-RT (general purpose) systems aim at

- fairness
- high performance (throughput)
- high resource utilization

RT systems only aim at

- predictability, if necessary, in detriment of the other aims.

Important timing parameters in executing a task



Real-Time (Paradigms) (18)

Table of generic Timing Parameters specifying instants (events) and intervals (durations)

Not.	Designation	Description
T_{trg}	trigger instant	arrival instant of event causing the execution
T_{off}	deferral time	delay introduced before execution request (**offset**)
T_{req}	request instant	instant of execution request (**release**)
T_{Rmin}	min. inter-req. time	minimum interval between any two consecutive requests (equals request period T_R, for periodic tasks)
T_{Xmin}	min. termin. time	minimum elapsed time from request to termin. event
T_{Xmax}	max. termin. time	worst-case elapsed time from request to termin. event
T_{WCET}	worst-case exec. time	maximum task duration in continuous execution
T_{lax}	laxity	slack time available for execution ($T_{Xmax} - T_{WCET}$)
T_{live}	earliest term. instant	earliest that task may complete, also called **liveline**
T_{trgt}	typ. termin. instant	*desired* instant of completion (**targetline**)
T_{dead}	latest term. instant	latest that task may complete, also called **deadline**
T_{int}	max. interfer. time	max. time task can be suspended by higher pri. tasks
T_{blk}	max. blocking time	max. time task can be blocked by lower pri. tasks
P	priority	importance of task w.r.t timing (highest is often 0)
U	max. utilization factor	max. percent. of CPU utilization (T_{WCET}/T_{Xmax})

Real-Time (Paradigms) (19)

W.r.t. the flexibility of tasks regarding their timing constraints and functionality, they can be classified as:

Hard tasks

All timing constraints must be met and optimal functionality is delivered.

Critical tasks

Their activation can be triggered later than the given release time.

Redundant tasks

All timing constraints are met and the delivered functionality(accuracy) is not optimal (gracefully degraded) but still acceptable (correct in the sense of in compliance with the overall specification).

Soft (best effort) tasks

Missing the deadlines of soft tasks can be tolerated.

Real-Time (Paradigms) (20)

Classification of scheduling algorithms:

Preemptive
The task being executed can be interrupted at any time in order to assign the processor to another task according to the used algorithm.

Non-preemptive
A task, once started, is executed by the processor until completion.

Static
Scheduling decisions are based on static (fixed) task parameters.

Dynamic
Scheduling decisions are based on dynamic (possibly changing at system run-time) task parameters

calendar-based
Tasks are executed according to a resulting calendar (time schedule).

Priority-based
Tasks are executing according to assigned (fixed or dynamically changing) priorities.

Independent
Release time of tasks does not depend on the termination time of other tasks

Real-Time (Paradigms) (21)

Classification of scheduling policies:

Static (off-line) scheduling

schedulability analysis is done off-line, i.e. before run-time

---> the used scheduling algorithm has complete a priori - knowledge about all relevant task parameters, i.e. a deterministic system and environment is assumed

Dynamic (on-line) scheduling

schedulability analysis is done on-line, i.e. at run-time

---> the used scheduling algorithm must not (cannot) have complete a priori - knowledge about all relevant task parameters of all tasks

---> provides predictability w.r.t. individual task arrivals by running so-called acceptance tests

(Timing) Fault-Tolerant scheduling

trading predictability and enhanced throughput for potentially degraded functionality of individual tasks

Real-Time (Paradigms) (22)

Schedulability

A set of tasks is *schedulable* or *feasible* if all timing constraints (deadlines) are met by some algorithm.

An algorithm is *optimal* for a given task set if it fails to meet all deadlines only if no other algorithm can meet all deadlines, i.e. it always generates a feasible schedule if one exists.

Determining whether a given task set is feasible is called *schedulability testing.* The outcome can be

- *sufficient:* passing it indicates that it is feasible
- *necessary:* failing it indicates that it is not feasible
- *exact:* sufficient and necessary

Utilization-based Tests

- fail, if the generated schedule will use the CPU more than a given percentage
- are sufficient, but not necessary

□□□□

Real-Time (Paradigms) (23)

Processor utilization factor U:

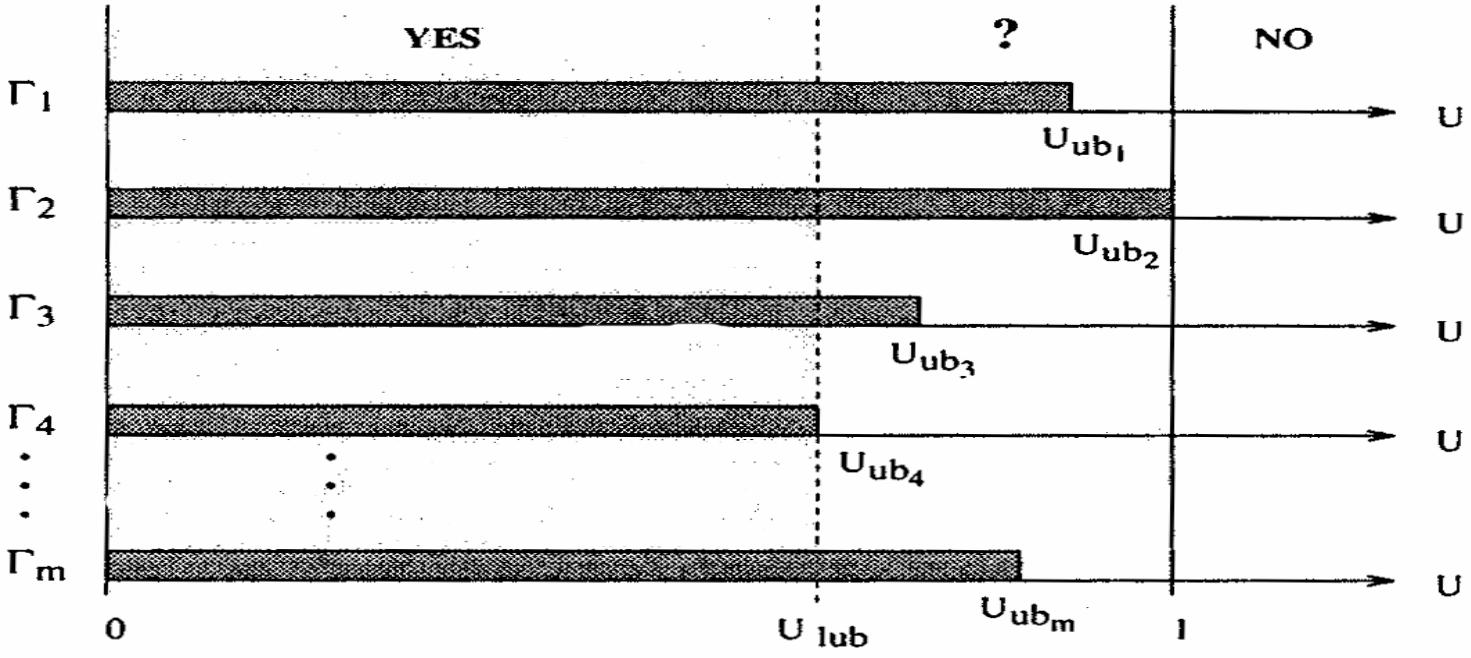
Given a finite set Γof n periodic tasks τ_i, U is the fraction of processor time spent in the execution of Γ, i.e.

$\mathbf{U = \Sigma\, C_i/T_i}$ (i = 1,...,n)

$\mathbf{U_{ub}\,(\Gamma,A)}$ is the upper bound of U for Γ under a given algorithm A in order to be feasible

$\mathbf{U = U_{ub}\,(\Gamma,A)}$ —> Γ is said to *fully utilize* the processor under A (full does not mean optimal utilization)

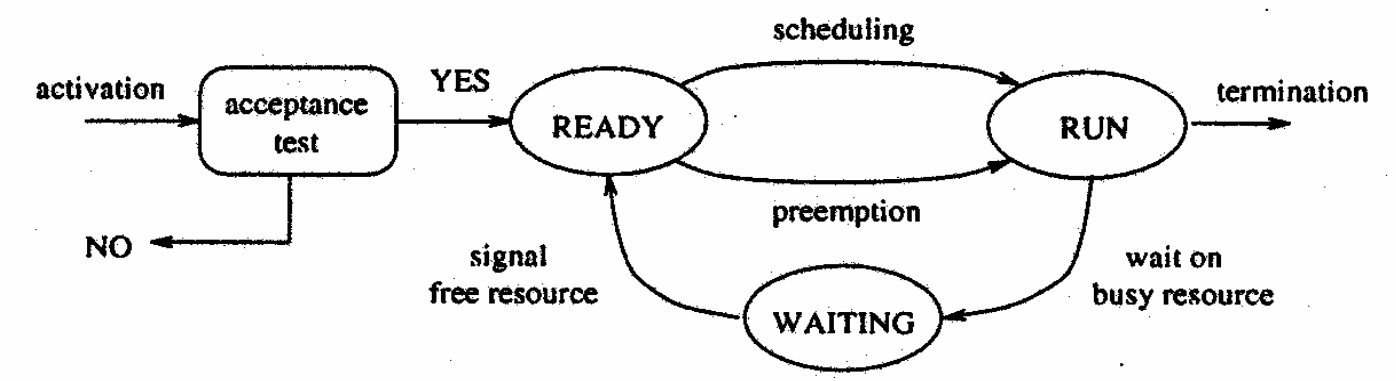
$\mathbf{U_{lub}(A) = \min U_{ub}(\Gamma,A)}$ is the least upper bound $U_{ub}(\Gamma,A)$ for all Γ



Real-Time (Paradigms) (24)

Response Time - based Tests

- determines for each task T_{max} by computing WCET + T_{int} and comparing it with $T_{dead.}$
- are exact



Acceptance Tests

- provide predictability w.r.t. individual task arrivals
- are sufficient

Rate-Monotonic Scheduling Algorithm (RM)

- designed for static scheduling of independent periodic tasks (all periods and WCET´s are known)
- the task´s priority is inversely related to its period ---> tasks with smaller periods have higher priorities
- it is preemptive and based on static priorities
- if for all tasks $T_{xmax} = T_R$, it is optimal among all fixed-priority algorithms
- $U_{lub} <= \ln 2$ is a sufficient condition for the schedulability test, $U_{lub} <= 1$, if the task set is *harmonic*, meaning that all periods are multiples of the smallest period