EEE499 - Real-Time Embedded System Design

Basic Real-Time System Terminology

ROYAL MILITARY COLLEGE OF CANADA
ELECTRICAL & COMPUTER ENGINEERING

GÉNIE ÉLECTRIQUE
ET GÉNIE INFORMATIQUE
COLLÈGE MILITAIRE ROYAL DU CANADA

RMC CMR

Outline

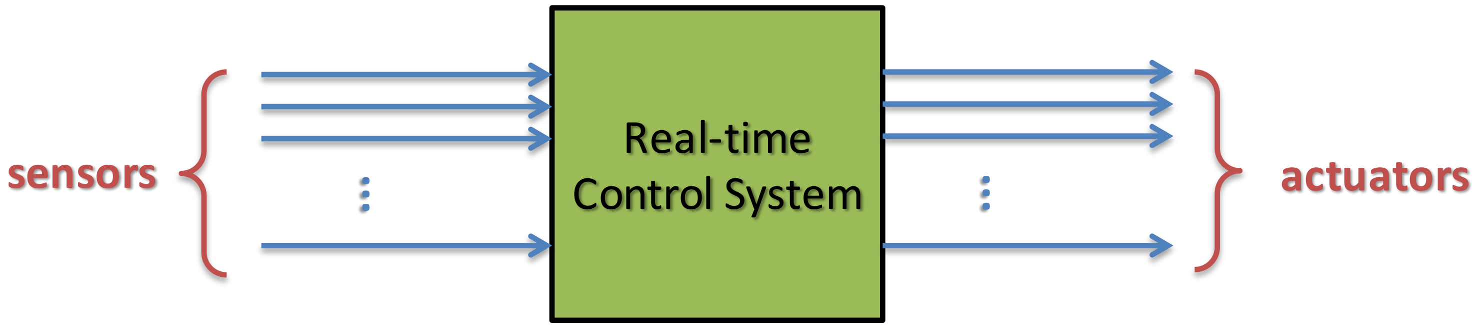
- Real-time definitions
- Soft versus hard real-time
- Timing attributes of RTS
- Jobs, tasks, processes & resources
- Specifying RTS timing constraints
- Embedded Systems

Temporary Definition

"A real-time system is required to complete its work and deliver its service on a timely basis" [1]

Real-Time Systems Definition

A simple model



Real-Time Systems Definition

1. "A real-time system is a computer system that must satisfy **bounded response-time** constraints or risk severe consequences, including **failure**" [2]
2. "A real-time system is one whose logical correctness is based on both the correctness of the **output** and their **timeliness**." [2]

Real-Time Systems Definition

3. "A real-time system is a software system that maintains an ongoing and **timely** interaction with its **environment**" [3]
4. "Any system in which the **time** at which output is produced is significant. This is usually because the input corresponds to some movement in the **physical world**, and the output has to relate to that same movement" [4]

Real-Time Systems Definition

1. Correctness
2. Timeliness
3. Interaction with environment
4. Consequences

A computer system is one whose logical correctness is based on both the **correctness** of the output and their **timeliness**. Where the timeliness is based an ongoing **interaction with the environment**. Failure to be logically correct has **consequences**.

Real-Time Systems Definition

Timeliness = Instantaneous ?

Real-Time Systems Definition

Timeliness ≠ Instantaneous

Real-Time Systems Definition

“All practical systems are ultimately real-time” [2]

Real-Time Systems Definition

Where do we draw the line?

Real-Time Systems Definition

Soft real-time

vs.

Hard real-time

vs.

Quality of service

Hard versus Soft Real-Time

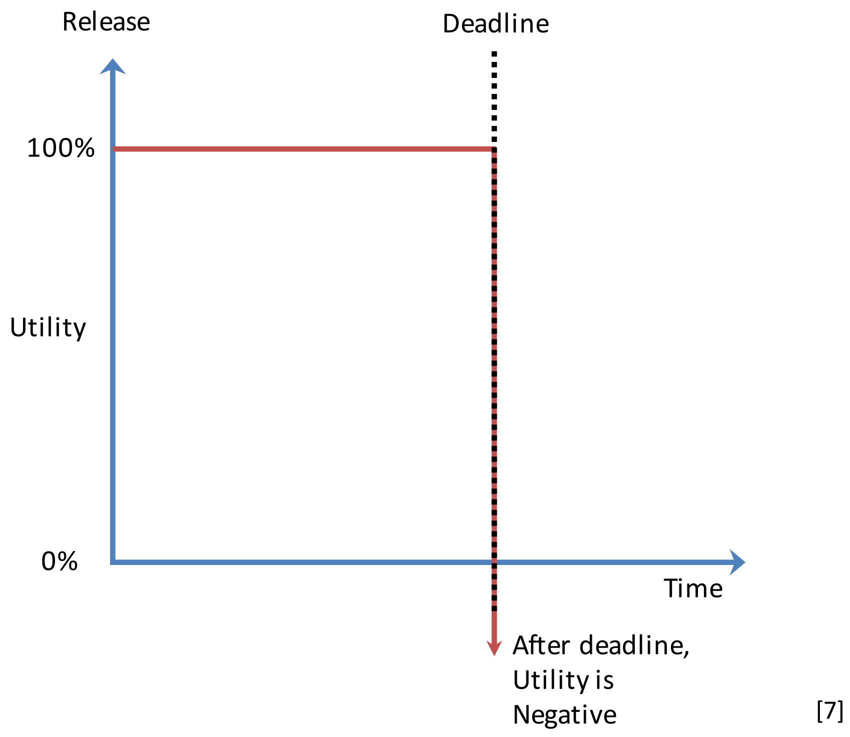
- there is no unanimous agreement within the field as to the exact boundary between hard, soft real-time and non-real-time.
- each definition is generally influenced by one of these points of view:
  - functional criticality
  - usefulness of late results
  - deterministic / probabilistic constraints

A good analogy is *guaranteed* versus *best effort* services

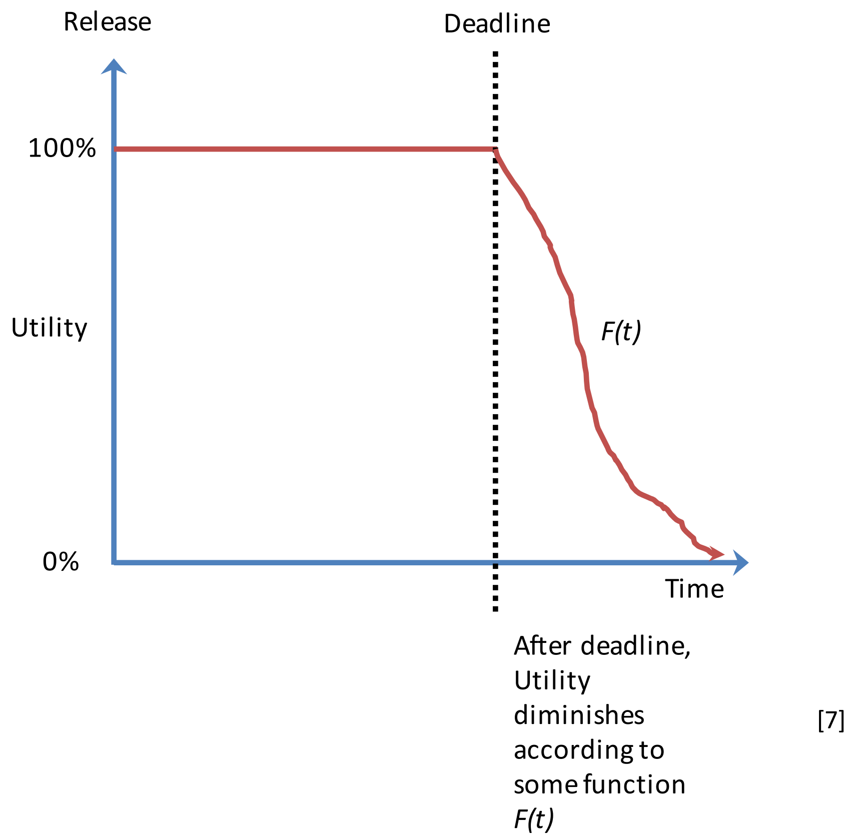
Hard versus Soft Real-Time

- **functional criticality**
  - the classification as to hard versus soft depends upon the consequences of a missed deadline
- **usefulness of late results**
  - the usefulness of a tardy hard real-time job decreases sharply while a soft real-time job more slowly degrades with tardiness
- **deterministic / probabilistic constraints**
  - a hard deadline must never be missed, while a soft deadline must be met x% of the time

Hard versus Soft Real-Time

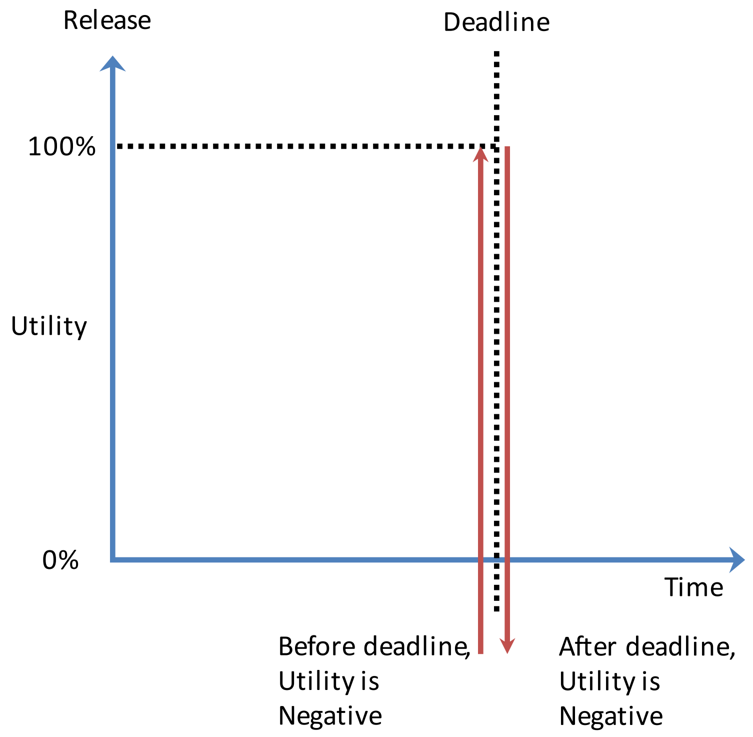


[7]

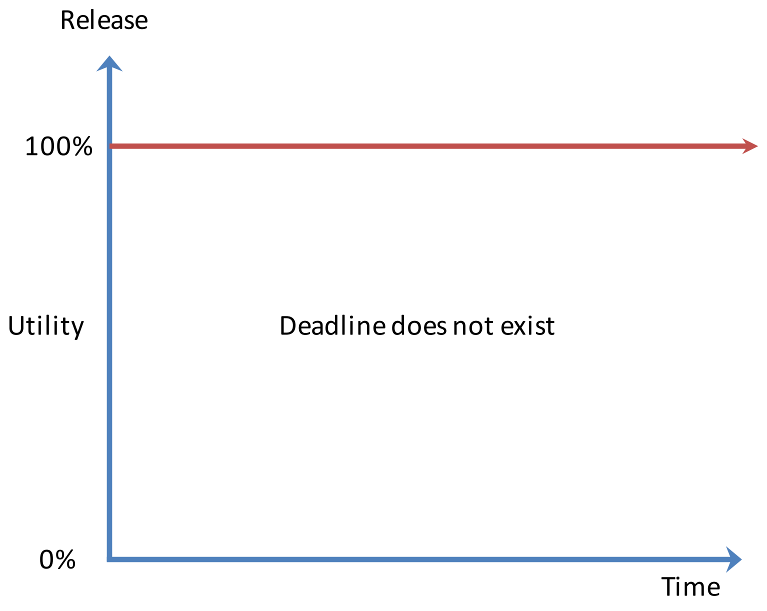

Release
Deadline
100%
Utility
F(t)
0%
Time
After deadline, Utility diminishes according to some function *F(t)*

[7]

Hard versus Soft Real-Time


Release
Deadline
100%
Utility
0%
Time
Before deadline, Utility is Negative
After deadline, Utility is Negative

[7]



[7]

Hard versus Soft Real-Time

- Liu's definition is particularly strict

  "The timing constraint is hard real-time if it must be <u>validated</u> that it is always met" [1]

- *validation requires*
  - *Proving the algorithm is correct, or*
  - *exhaustive test / simulation*

Hard Real-Time Systems

A hard real-time system is one which is made up of jobs/tasks with mostly hard real-time deadlines

Why do we need <u>hard</u> RTS?

- the real world may dictate it
  - flight control system
- safety
  - nuclear power plant
- high reliability / high availability
  - space probe, satellite
- high cost of recovery, speedy recovery
  - international monetary exchange system

How important is one missed deadline?

- there may be systems where we cannot see how just one missed deadline will make a difference
- and while this may be true, it is often easier to prove the correctness of a “hard” constraint than it is a probabilistic one

the effects of a missed deadline may have system ripple effects that are non-intuitive and extremely difficult to determine

Soft Real-Time Systems

A soft real-time system is one in which the jobs have soft deadlines

Soft Real-Time Systems

- timing constraints are generally more relaxed and validation is less rigorous
  - these systems may not be less complex/expensive - why not?
- timing constraints often specified probabilistically
  - the probability that a deadline will be exceeded by 10 msec is 1%
- examples:
  - telephone switching, multi-media, stock price quotation system

Jobs, Tasks and Processes

- a **job** is a unit of work that is scheduled and executed on a real-time system
  - – examples: read sensor, compute FFT, …
- whereas a **task** is a set of related jobs which collectively provide a system function
  - – example: the FCS control loop set of jobs above may combine to provide the function *control aircraft*

Jobs, Tasks and Processes

- a job executes on a special resource which we will call a **processor**
  - most commonly it is a CPU, but
  - note that this very general concept of a processor might also include a network or a disk (where execute now refers to transmit and access respectively)

Timing attributes of RTS

- purely cyclic
  - all tasks execute on a periodic basis
  - even I/O tasks are made periodic through polling
  - includes most digital control / monitor systems

Timing attributes of RTS

- mostly cyclic
  - while most tasks are periodic in such a system, a few tasks are asynchronous
  - an asynchronous task is one which could arrive at any time (unexpectedly)
    - example - keyboard input
  - includes most high-level control systems

Timing attributes of RTS

- asynchronous & somewhat predicable
  - most tasks are asynchronous and the duration between tasks executions may vary
  - however these variations are said to be *somewhat* predicable
    - bounded or statistically well behaved
  - includes signal processing, multimedia

Timing attributes of RTS

- asynchronous & unpredictable
  - most tasks again are asynchronous and usually have an associated high-run time complexity
  - either computationally unpredictable or are highly dynamic in terms of structure
  - E.g.: A core overheat in a nuclear power plant

Specifying RTS timing constraints

Release time

the instant a job is ready for execution

– why might it not be ready?

– does it execute right away?

Response time

the time difference between a job’s release and its completion

Specifying RTS timing constraints

Deadline (absolute)

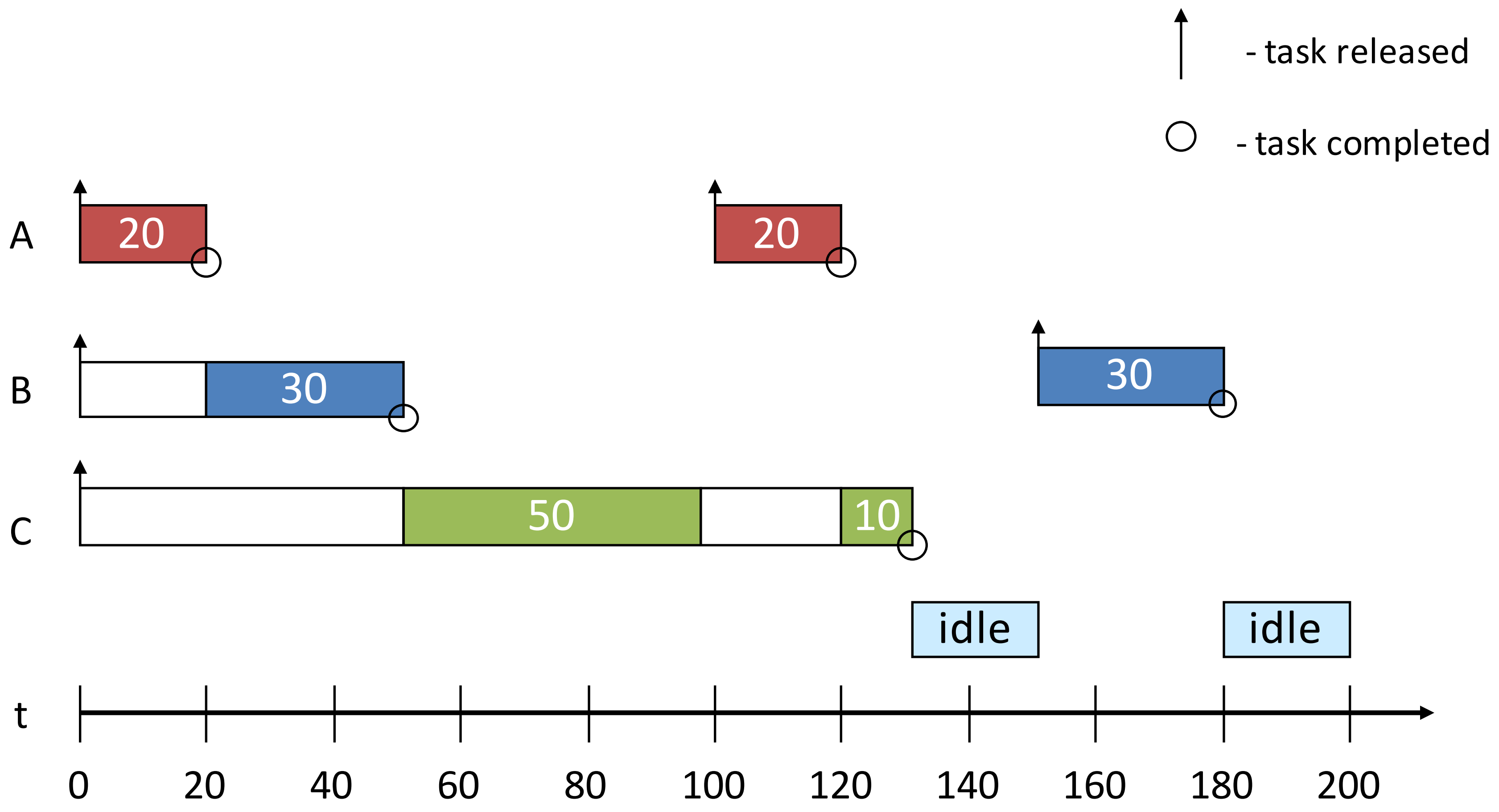
- Time by which a job must be completed
- a job with no deadline need not finish before ∞

Deadline (relative)

- the maximum allowable response time
- for periodic jobs, deadlines may be less than, equal to, or greater than the period

we will use “deadline” to refer to relative deadline

Specifying RTS timing constraints



Embedded Systems Definition

"An embedded system is a **special-purpose computer** completely contained within the device it controls and not directly observable by the user of the system. An embedded system **performs specific predefined services** rather than user-specified functions and services as a general-purpose computer does ." [7]

"[...] the point of an embedded system is to cost-effectively provide a more limited set of services in a larger system" [7]

Embedded Systems Definition

1. Special-purpose
2. Specific services
3. Cost effective

References

[1] Liu, J. W. S. Real-Time Systems. Prentice Hall, 2000.

[2] Laplante, P. A. and Ovaska, S. J. Real-Time Systems Design and Analysis, 4th Edition. Wiley, 2012.

[3] Selic, B. et al. Real-Time Object Oriented Modeling. Wiley, 1994.

[4] Daintith, J. and Wright, E. A Dictionary of Computing, 6th Edition. Oxford University Press, 2008.

[5] Smith, R. SOFT426: Real-Time Systems Course. Queen’s University, 2004.

[6] Perkins, C. Real-Time and Embedded Systems Course, University of Glasgow, 2007.

[7] Siewert, S. Pratt, J. Real-Time Embedded Components and Systems with Linux and RTOS. Mercury Learning and Infromation, 2016.