Introduction to Real-Time Computer Systems

Lecture 1

Background

- This course is not about real-time system programming
- This course is not about embedded systems

- A good background about OS is required
  - process management
  - memory management
  - mutual exclusion, resource locking

- There will be no programming project
- There will be a lot of algorithmic discussions

Topics to Be Covered

- Intro to Real-time Systems
- Periodic Task Scheduling: RM, EDF, DM
- Aperiodic Task Scheduling
  - Fixed-Priority Servers: Background, Polling, Deferrable, Sporadic
  - Dynamic Priority Servers: Dynamic Sporadic, TBS, CBS
- Resource Access Protocols
  - Priority Inversion
  - Priority Inheritance, Priority Ceiling, Stack Resource Policy
- RTOS, Garbage Collection, Middleware
- Sensors and Actuators, Cyber-Physical Systems, IoT

EECS 223

Textbook

- Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications, 2nd ed. or 3rd ed.
  - By Giorgio C. Buttazzo

Grading:

- Midterm examination (30%): Tue Nov 7, 12:30 – 1:50
- Final examination (40%): Fri Dec 15, 10:30 - 12:30
- Homework or project (30%): 4-5 homework assignments

Office Hour

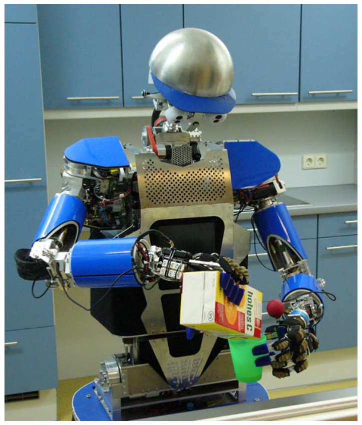
- Th 3-5, EH 4205

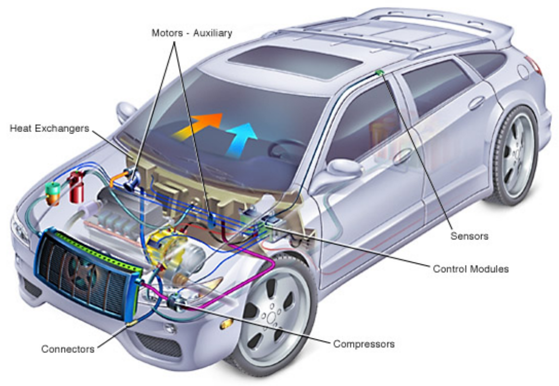
Web Site: https://eee.uci.edu/17f/18420/

Related: Embedded systems

- An embedded system is an electronic system that is
  - designed to perform a dedicated function.
  - oftentimes embedded within a larger system

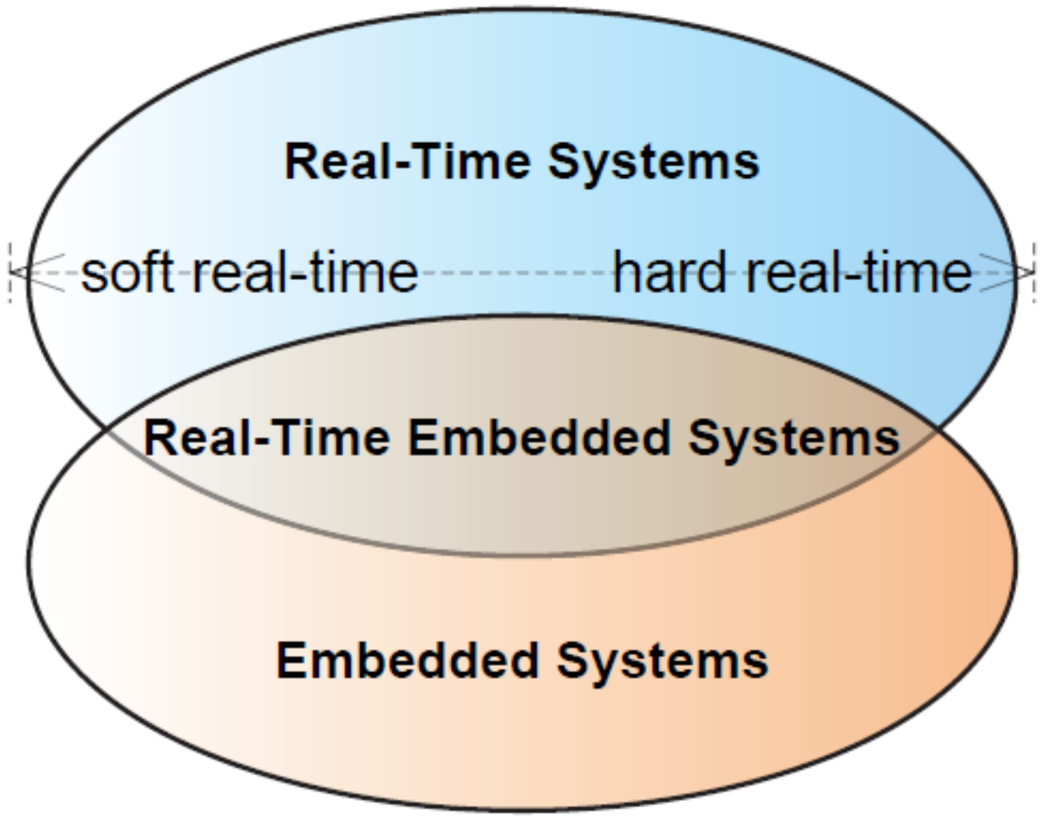
- Consumer electronics
- Industrial robots
- Command and control systems
- Medical equipment
- Automotive systems
- IoT





Spectrum of Real-Time Systems

A real-time system is called a *real-time embedded system* if it is designed to be embedded within some larger system.



What is a Real-Time System?

- The correctness of the system depends not only on the logical result of the computation, but also on the time at which the results are produced
- In other words, a computation must satisfy two aspects of correctness: *functional* and *temporal*.

  e.g. a homework must be submitted before the deadline;

  a running car must be stopped in time;
- Examples:
  - Industrial and automation system, medical instrument and devices, military defense system, communication systems
  - Smart Factory, Smart Car, Smart Home

Real-time systems

- A service request (event) is typically associated with a real-time computing constraint, or simply a timing constraint.
- Critical event has hard timing constraint, where the consequence of a missed deadline is fatal (unacceptable)
- Non-critical event has soft timing constraint, where the consequence of a missed deadline is undesirable but tolerable
- Real-time systems are those that can provide guaranteed worst-case response times to critical events, as well as acceptable average-case response times to noncritical events.

Soft Real-Time Systems

Soft timing constraints are typically expressed in probabilistic or statistical terms

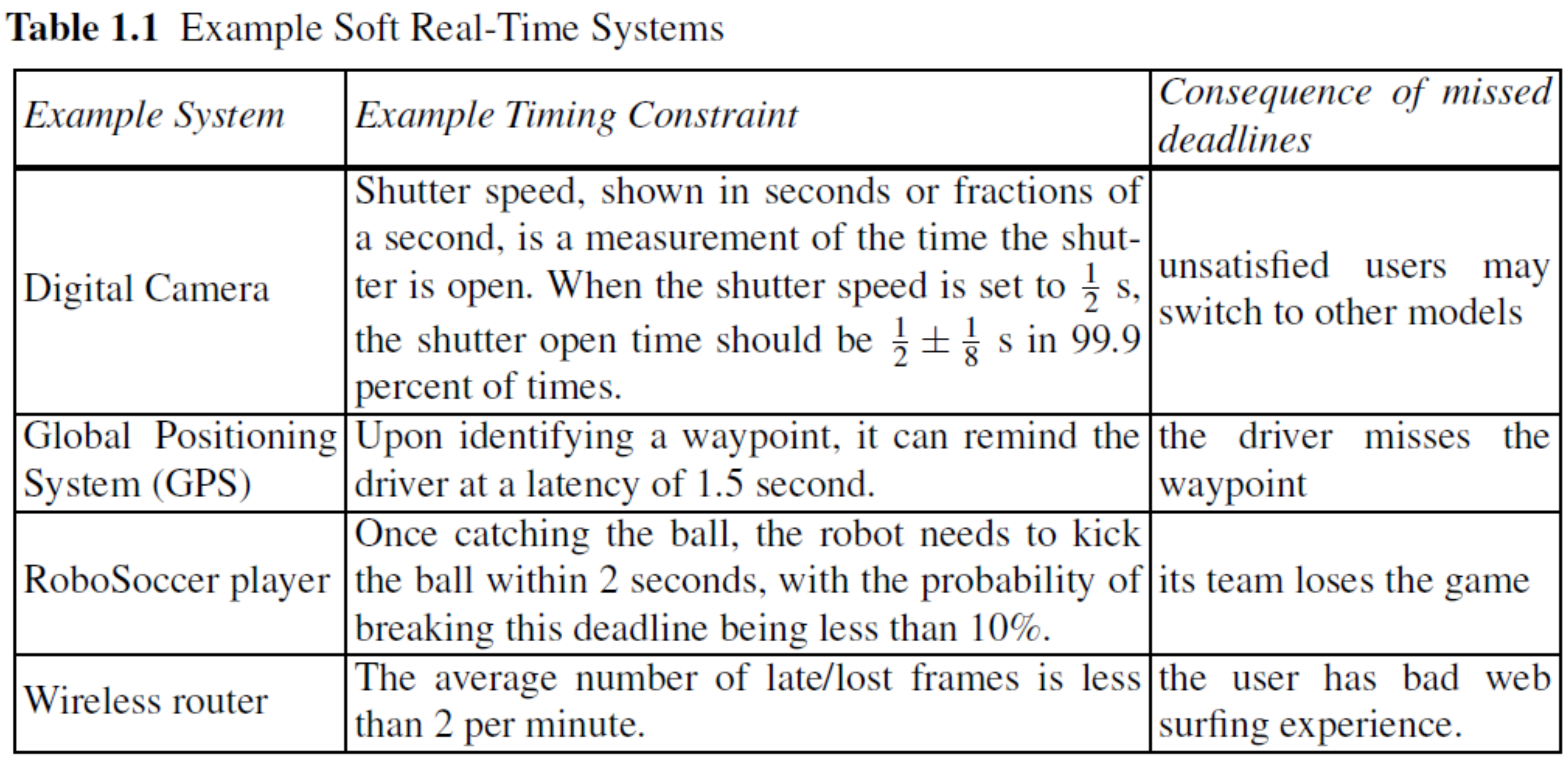


Table 1.1 Example Soft Real-Time Systems

Example System	*Example Timing Constraint*	*Consequence of missed deadlines*
Digital Camera	Shutter speed, shown in seconds or fractions of a second, is a measurement of the time the shutter is open. When the shutter speed is set to $\frac{1}{2}$ s, the shutter open time should be $\frac{1}{2} \pm \frac{1}{8}$ s in 99.9 percent of times.	unsatisfied users may switch to other models
Global Positioning System (GPS)	Upon identifying a waypoint, it can remind the driver at a latency of 1.5 second.	the driver misses the waypoint
RoboSoccer player	Once catching the ball, the robot needs to kick the ball within 2 seconds, with the probability of breaking this deadline being less than 10%.	its team loses the game
Wireless router	The average number of late/lost frames is less than 2 per minute.	the user has bad web surfing experience.

Hard Real-Time Systems

Hard timing constraints are typically expressed in deterministic terms

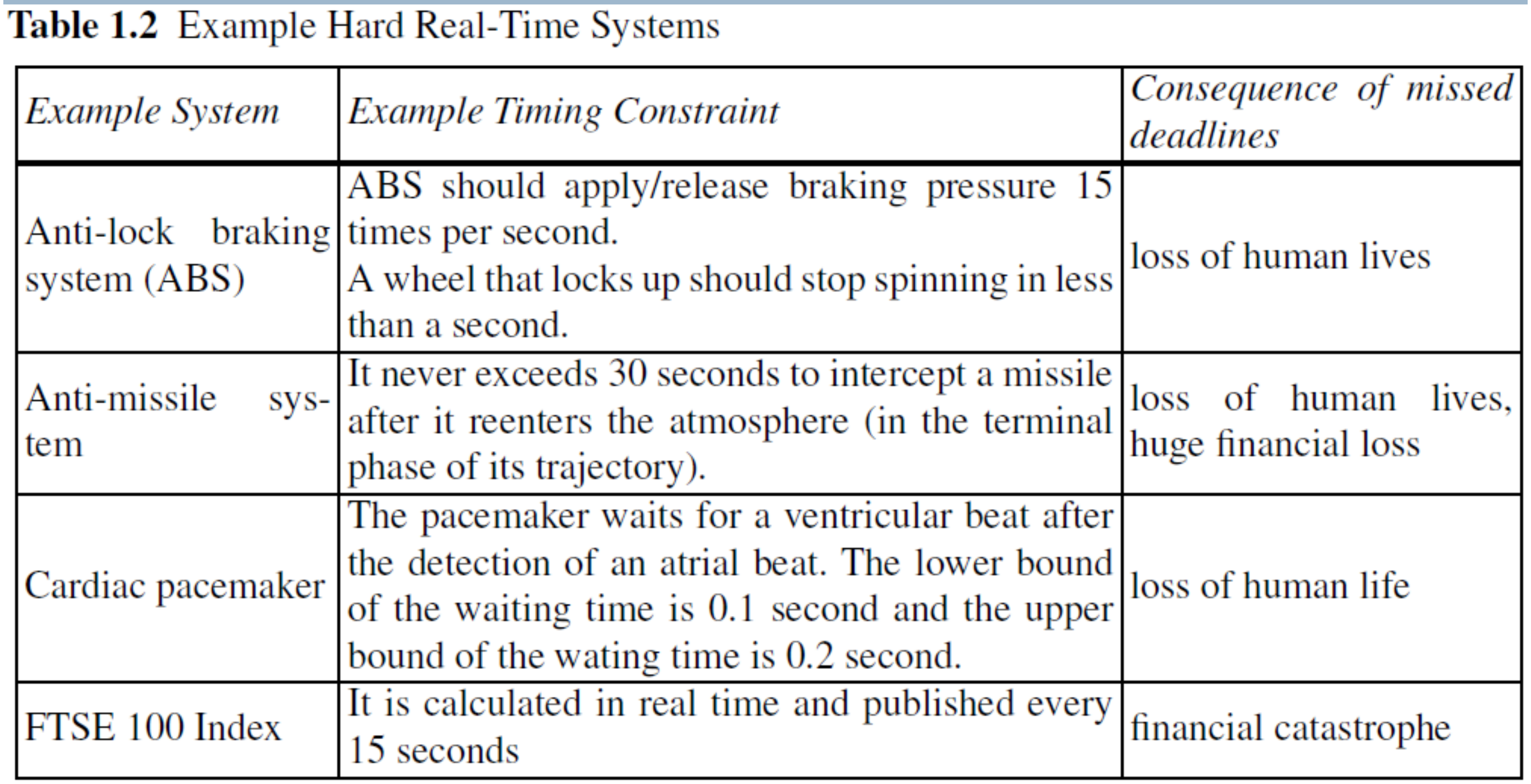


Table 1.2 Example Hard Real-Time Systems

Example System	*Example Timing Constraint*	*Consequence of missed deadlines*
Anti-lock braking system (ABS)	ABS should apply/release braking pressure 15 times per second. A wheel that locks up should stop spinning in less than a second.	loss of human lives
Anti-missile system	It never exceeds 30 seconds to intercept a missile after it reenters the atmosphere (in the terminal phase of its trajectory).	loss of human lives, huge financial loss
Cardiac pacemaker	The pacemaker waits for a ventricular beat after the detection of an atrial beat. The lower bound of the waiting time is 0.1 second and the upper bound of the wating time is 0.2 second.	loss of human life
FTSE 100 Index	It is calculated in real time and published every 15 seconds	financial catastrophe

Issues on Earlier Real-Time Systems

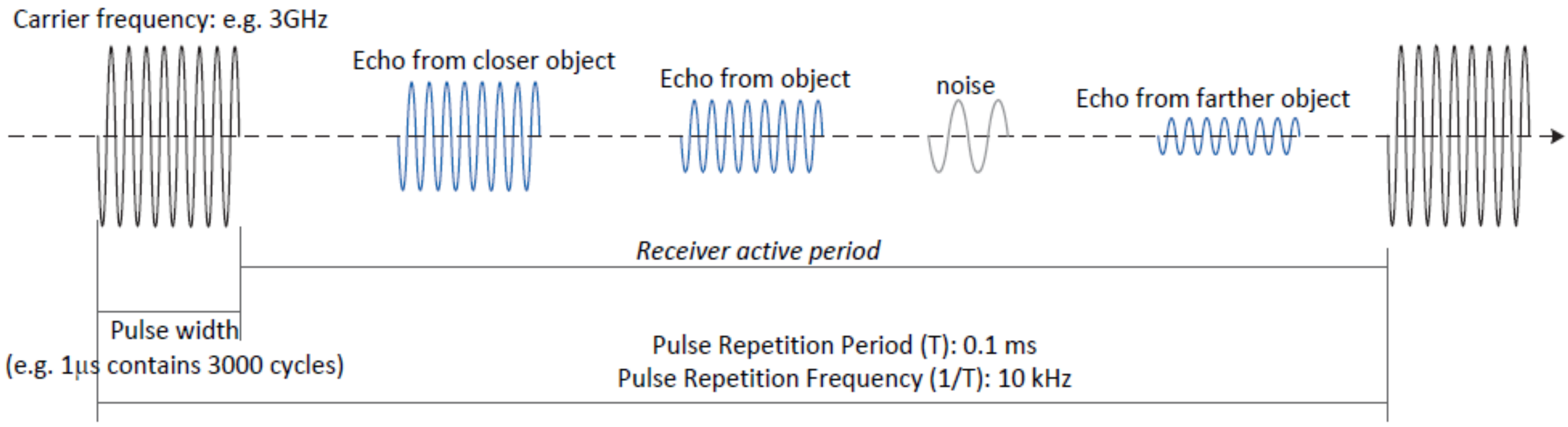
- Many classical control applications with stringent time constraints were implemented in special language, programming timers, writing low-level drivers for device handling, and manipulating task and interrupt priorities.
- The consequences of a failure can sometimes be catastrophic and may injure people or cause serious damages E.g.
  - Space shuttle delay due to 1.5% probability transient overload;
  - Patriot missiles miscalculation due to delay accumulation of 0.34 sec (or 687m, after 100 hours of operations)

http://sydney.edu.au/engineering/it/~alum/patriot_bug.html

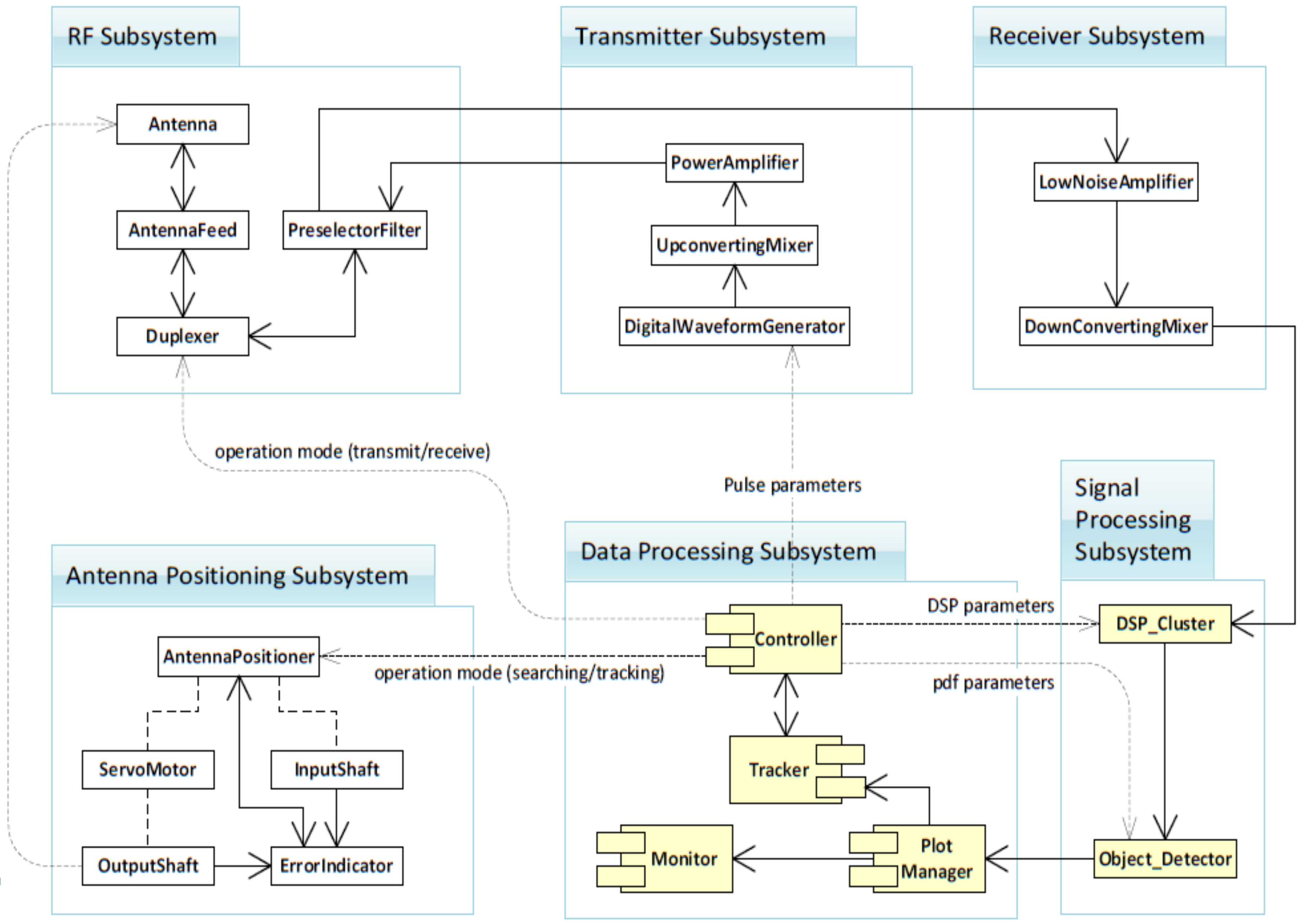
http://autarkaw.wordpress.com/2008/06/02/round-off-errors-and-the-patriot-missile/

Issues on Low-Level Programming

- **Tedious programming**. The implementation of large and complex applications in assembly language is much more difficult than high-level programming. Moreover, the efficiency of the code strongly depends on the programmer's ability.
- **Difficult code understanding**. Clever hand-coding introduces additional complexity and makes a program more difficult to comprehend.
- **Difficult software maintainability**. As the complexity of the program increases, the modification of large assembly programs becomes difficult even for the original programmer.
- **Difficult verification of time constraints**. Without the support of specific tools and methodologies for code and schedulability analysis, the verification of time constraints becomes practically impossible.


Carrier frequency: e.g. 3GHz
Echo from closer object
Echo from object
noise
Echo from farther object
Receiver active period
Pulse width
(e.g. 1μs contains 3000 cycles)
Pulse Repetition Period (T): 0.1 ms
Pulse Repetition Frequency (1/T): 10 kHz

Case Study: Radar System



Desirable Properties

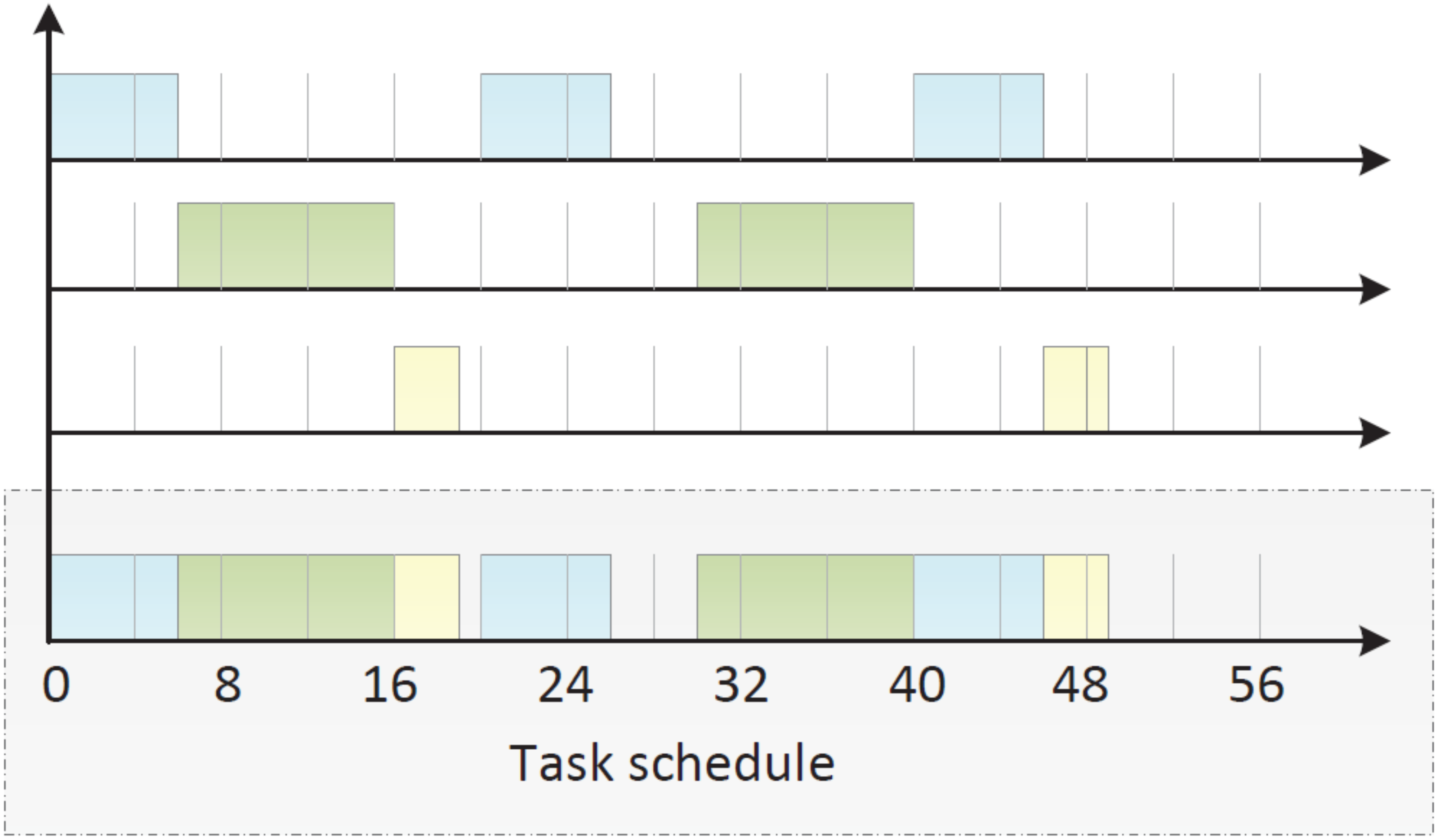
1. **Timeliness.** Results have to be correct not only in their value but also in time.
2. **Design for peak load.** Real-time systems must not collapse when they are subject to peak-load conditions.
3. **Predictability.** To guarantee a minimum level of performance, the system must be able to predict the consequences of any scheduling decision.
4. **Fault tolerance.** Single hardware and software failures should not cause the system to crash.
5. **Maintainability.** Possible system modifications are easy to perform.

Real-time Tasks Models

- The term ‘task’ is a design-time concept. Each task represents a unit of concurrency.
- A task with real-time constraints is called a real-time task.
- A task can be executed multiple times. Each individual run to completion of a task is called a *job* of that task.
- Periodic tasks: a periodic task can be executed repeatedly on a continuing basis. The inter-arrival times between consecutive jobs are almost the same, which is called its period.
- Sporadic tasks: 不定时的 a sporadic task is executed in response to external events which occur at random instants of time. The inter-arrival times between consecutive jobs may vary widely, and can be arbitrarily small (bursty occurrence). Sporadic tasks have hard deadlines.
- Aperiodic tasks: 非周期 an aperiodic task is also executed in response to external events which may occur at random instants of time. The inter-arrival times of aperiodic jobs may follow some probability distribution function. Aperiodic tasks have either soft or no deadlines.

A timing diagram showing the design of task schedule

Task A: period=20ms, execution time=6ms, deadline=20ms
Task B: period=30ms, execution time=10ms,deadline=30ms
Task C: period=40ms, execution time=3ms, deadline=40ms


Task schedule