Real Time systems

Lecture 1

Real-time systems are computing systems that must react within precise time constraints to events in the environment. As a consequence, the correct behavior of these systems depends not only on the value of the computation but also on the time at which the results are produced.

Examples of applications that require real-time computing include the following:

1. Chemical and nuclear plant control,
2. control of complex production processes,
3. railway switching systems,
4. automotive applications,
5. flight control systems,
6. environmental acquisition and monitoring,
7. telecommunication systems,
8. medical systems,
9. industrial automation,
10. robotics,
11. military systems,
12. space missions,
13. consumer electronic devices,
14. multimedia systems,
15. smart toys, and
16. virtual reality.

In many cases, the real-time computer running the application is embedded into the system to be controlled. Embedded systems span from small portable devices (e.g., cellular phones, cameras, navigators, ECG Holter devices, smart toys) to larger systems (e.g., industrial robots, cars, aircrafts). A high percentage of accidents that occur in nuclear power plants, space missions, or defense systems are often caused by software bugs in the control system.

1.2 WHAT DOES REAL TIME MEAN?

1.2.1 THE CONCEPT OF TIME.

The word *time* means that 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. The word *real* indicates that the reaction of the systems to external events must occur *during* their evolution. As a consequence, the system time (internal time) must be measured using the same time scale used for measuring the time in the controlled environment (external time).

Real time system <> fast system.

the objective of fast computing is to minimize the average response time of a given set of tasks, the objective of real-time computing is to meet the individual timing requirement of each task.

The environment is always an essential component of any real-time system. Figure 1 shows a block diagram of a typical real-time architecture for controlling a physical system. Rather than being fast, a real-time computing system should be predictable. And one safe way to achieve predictability is to investigate and employ new methodologies at every stage of the development of an application, from design to testing.

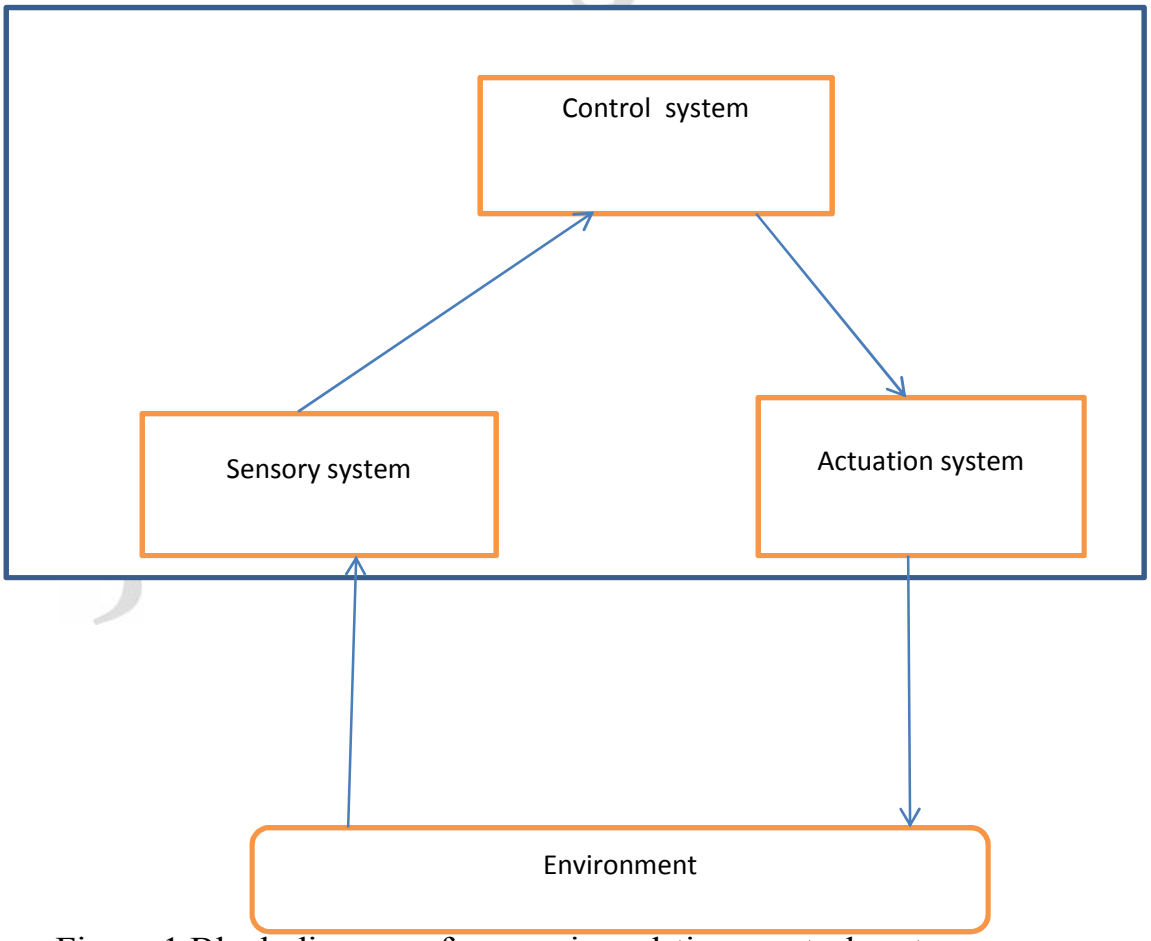

Figure 1 Block diagram of a generic real-time control system

At the process level, the main difference between a real-time and a non-real-time task is that a real-time task is characterized by **a *deadline*, which is the maximum time within which it must complete its execution**. In critical applications, a result produced after the deadline is not only late but wrong!
Depending on the consequences that may occur because of a missed deadline, a real time task can be distinguished in three categories:
Hard: A real-time task is said to be *hard* if producing the results after its deadline may cause catastrophic consequences on the system under control. E.g. Sensory data acquisition; data filtering and prediction; detection of critical conditions; data fusion and image processing; actuator servoing; low-level control of critical system components; and action planning for systems that tightly interact with the environment.

Firm: A real-time task is said to be *firm* if producing the results after its deadline is useless for the system, but does not cause any damage. E.g. Video playing; audio/video encoding and decoding; on-line image processing; sensory data transmission in distributed systems.

Soft: A real-time task is said to be *soft* if producing the results after its deadline has still some utility for the system, although causing a performance degradation . e.g. The command interpreter of the user interface; handling input data from the keyboard; displaying messages on the screen; representation of system state variables;
graphical activities; and saving report data.