ECEN 5623

REAL-TIME EMBEDDED SYSTEMS

HOMEWORK-1

SUBMITTED BY:

SOWMYA RAMAKRISHNAN

108684769

Sowmya.Ramakrishnan@colorado.edu

REAL-TIME EMBEDDED COMPONENTS AND SYSTEMS WITH LINUX AND RTOS-SAM SIEWART AND J0HN PRATT

CHAPTER 1 - EXERCISES

Q1

Examples of real-time embedded systems and how they meet the common definition of real-time and embedded:

- Commercial: vehicle systems for automobiles, subways, aircraft, railways, and ships: (ABS, Airbags, Navigation and Telemetry systems)
  - Real-Time: Vehicular components such as those mentioned above are bounded by deadlines. When the ABS/Airbags do not work on time, it may lead to hazardous events. Navigation should be provided when needed and telemetry (used to monitor individual vehicles) needs to be on-time.
  - Embedded: The components in vehicular systems are designed to perform specific and focused functions (ABS/Airbags for safety, Navigation systems for directions and Telemetry systems for monitoring) and hence are embedded in nature.
- Medical: radiation therapy, patient monitoring, and defibrillation
  - Real-Time: Medical equipments are strictly bounded by “time”. They are used in critical situations for extremely important tasks (involving life) and failure to adhere to time could have fatal consequences.
  - Embedded: These components perform extremely specialised functions and are embedded in nature.
- Robots having specified functionality (Line follower, manufacturing robots) are real-time in the sense that they need to act when they are supposed to (when time starts), need to follow up on the actions performed in other parts of the task and keep updating themselves. They are parts of bigger mechanisms and processes which perform concentrated functions, and this makes them embedded in nature.
- Home Automation System components like thermostats, sprinkler controllers, leak detectors and fire alarms are constrained in the

functions that they perform and hence are embedded in nature. These components have to act on time and for the specific period of time (thermostat till the temperature is controlled, leak detectors till leaks are present, fire alarms till smoke is detected etc) and this makes them real-time.

Q2

Section 3 of the paper by C. L. Liu and James W. Layland was read. An assumption is made that all requests for services are periodic.

As mentioned, the assumption is made based on its validity for process control, even though it is contradicting an opinion. In this section and paper, a task is completely characterized by its request period and run-time. For a task to not vary with time (constant run-time), and still not overlap with other tasks (even though they are independent, they can take place at the same time and this may cause errors), it must be periodic in nature-the constant interval between requests ensures that there is no overlap between services.

This is explained in the conclusion section of the paper. It says that the assumptions that requests be periodic and run-times be constant are the most important and least defensible of all five assumptions. In the absence of these, we would have to define the critical time zone for each task as the time zone between its request and deadline during which tasks having higher priority perform the maximum amount of computation. Unless detailed knowledge of run-time and request periods are available, run-ability constraints on task run-times would have to be computed based on assumed periodicity and constant run-time, using a period equal to the shortest request interval and a run-time equal to the longest run-time. A severe bound on processor utilization could be imposed by the task aperiodicity. Thus, the value of the implications of the assumption are great enough to make them a design goal for any real-time tasks which must receive guaranteed service.

This might serve to be a problem with a real application for the following reason:

The request for a real-time service need not and in most cases, will not be and cannot be made periodic. A real-time service is “time” dependent; and it requires start as soon as the request is made. The request cannot be

assumed to be made in a periodic manner, because it is something that is happening in real and nothing can really be predicted about it.

Thus, a problem exists with real applications when they are assumed to be periodic.

Q3

Hard Real-Time:

A service or set of services that are required to meet their deadlines relative to request frequency; if such deadlines are missed, there is not only no no utility in continuing the service, but in fact the consequences to the system are considered fatal or critical.

Soft Real-Time:

When a service can occasionally miss a deadline, and overrun it or terminate and drop a service release without system failure, these services are considered soft-for example, a video recorder compression and transport service might occasionally drop a frame when compression takes too long; as long as the video stream is not critical and an occasion dropout is acceptable regarding system requirements, this service can be considered a soft real-time service.

WHY they are different:

Real-time services are classified as Hard and Soft because they then serve as means to identify and differentiate extremely time and deadline- critical services from those that are not so critical with regard to meeting deadlines.

HOW they are different:

- In hard real-time systems, deadlines must never be missed. If they are missed, it becomes a case of total system failure and the task being completed or not ceases to matter. 'Timing' and 'deadline' are extremely vital factors in hard real-time services. If the deadlne to complete a task is 10seconds, it should be completed at exactly the 10th second-completion during the 5th or 9th or 11th is unaccepted and at most times, catastrophic.
- In soft real-time systems, there is some amount of elasticity and flexibility. Services must be completed, but there is no strict

implication if the deadlines are not met. If the deadlne to complete a task is 10seconds, it can be completed anytime between the 1st and 10th second- sometimes even a delay is accepted and does not lead to dire consequences.

This leads to another major difference between hard and soft real-time services:

- In hard real-time systems, failure to conform to timing constraints results in a loss of life or property. Hence, this scheduling is used extensively in mission critical systems of utmost priority.
- In soft real-time systems, completed tasks may have increasing value up to the deadline and decreasing value past it. There are no losses in terms of life or property and often it results in only some amount of degradation in quality (if deadlines are missed).
- In hard real-time systems, there is the concept of "system failure". In soft real-time systems, there is no concept of absolute "failure".
- In hard real-time systems, after the deadline passes, the result does not have any value-i. e; it is zero.
- In soft real-time systems, the result of the requests is not a worthless value after their deadline, rather it degrades as time passes after the deadline.

Examples:

Hard Real-Time Services:

- Apollo 11 lunar module descent guidance overload- CPU resouce overload on the Apollo 11 might have caused it to crash if it had not been rectified in time.
- Air France Flight 447 crashed into the ocean after a sensor malfunction caused a series of system errors. The pilots stalled the aircraft while responding to outdated instrument readings.
- Airbag system in vehicles should detect the crash and inflate rapidly. If the system reacts with even 1 second of delay, the consequences could be mortal, and it will be of no benefit having the bag inflated once the car has already crashed.

Soft Real-Time Services:

- A video game console runs software for a game engine. There are many resources that must be shared between its tasks. At the

same time tasks need to be completed according to the schedule for the game to play correctly. As long as tasks are being completely relatively on time the game will be enjoyable, and if not, it may only lag a little.

- Web browser- A URL takes some time in getting loaded in a browser. If the system takes more time than expected, the page obtained is not considered invalid- It only becomes a case of the system's performance not being up to the mark,
- Home automation: a software that automatically controls the temperature of a room (or a building). If the system has some delays reading the temperature sensors, it will be slower to react. Hoever, eventually it will end up reacting to the change and the temperature will be adjusted accordingly. In this case, the delayed reaction is useful, but it degrades the quality.

REFERENCES

(1) Real-Time Embedded Components and Systems with Linux and RTOS-Sam Siewart and John Pratt

(2) Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment-C. L. Liu and James W. Layland