[Name: SALIPA GURUNG]

[Roll no. : 31]

Real Time System

Assignment 1

1. Define real time system. Compare and contrast Hard Real Time Vs Soft Real Time System with suitable examples.

Ans : A real time system is one whose logical correctness is based on both the correctness of the outputs and their timeliness.

Term	Hard Real Time System	Soft Real Time System
Definition	Systems where failure to meet response-time constraints leads to system failure.	System in which performance is degraded but not destroyed by failure to meet response-time constraints.
File size	Size of the data file is small or medium.	Size of the data file is large.
Response time	Response time is predefined that is in a millisecond.	Response time is higher.
Utility	Has more utility	Has less utility
Database	Has short databases	Has enlarged databases
Performance	Peak load performance should be predictable	Peak load can be tolerated
Safety	Safety is critical	Safety is not critical

Integrity	Have short term data integrity	Have long term data term integrity
Restrictive nature	Very restrictive	Less restrictive
Computation	In case of an error, the computation is rolled back	Computation is rolled back to a previously established checkpoint to initiate a recovery action.
Flexibility And laxity	Not flexible They have less laxity, generally provide full deadline compliance	More flexible They have greater laxity and can tolerate certain amounts of deadline misses
Validation	All users of Hard Real-time Systems get validation when needed	All users of Soft Real-time Systems do not get validation
Outcome of the deadline miss	Disastrous if system fails	System failure does not result in severe harm
Quality of service	Temporal	Best
Examples	1. Satellite launch 2. Railway signaling systems 3. Medical system 4. Flight control system 5. Missile Guidance system 6. Nuclear reactor control	1. DVD player 2. Telephone switches 3. Electronic games 4. Linux 5. Mobile communication 6. Weather monitoring systems

	system 7. Inkjet printer system 8. Safety critical systems are good examples of a hard real time system	7. Virtual reality 8. Personal computer 9. Online transaction system 10. Multimedia system 11. Web browsing

2. What do you mean by Deterministic system? Explain the design issues of the Real Time System.

Ans: A system is said to be deterministic if, for each possible state and each set of inputs, a unique set of outputs and next state of the system can be determined.

- In particular, a certain kind of determinism called *event determinism* means that the next step and outputs of the system are known for each state of inputs that trigger events.
- Finally, if in a deterministic system the response time for each set of outputs is known, then the system also exhibits temporal determinism.
- A side benefit of designing deterministic systems is that you can guarantee that the system is able to respond at any time, and in the case of temporally deterministic systems, when they will respond. This reinforces the association of control with real-time systems.

Issues in Real Time System Design

Most of the challenges come from the fact that Real time systems have to interact with real world entities. These interactions can get fairly complex. A typical real time system might be interacting with thousands of such entities at the same time. For example, a telephone switching system routinely handles calls from tens of thousands of subscribers. The system has to connect each call differently. Alson the exact sequence of events in the call might vary a lot.

1. Real time response
2. Recovering from issues
3. Working with distributed architecture
4. Asynchronous architecture
5. Race conditions and timing

Real time response

RTS has to respond to external interactions in a predetermined amount of time. Successful completion of an operation depends upon the correct and timely operation of the system. For example, a telephone switching system must feed dial tone to thousands of subscribers within a recommended limit of one second. To meet these requirements, the off hook detection mechanism and the software message communication involved have to work within the limited budget. The system has to meet these requirements for all the calls being set up at any given time.

Recovering from failures

RTS must function reliably in event of failures. These failures can be internal as well as external.

Internal failures can be due to hardware and software failures in the system.

- Software failures in a task
- Processor restart
- Board failure
- Link failure

External failure can be due to the external environment while performing in real world.

- Invalid behavior of external entities
- Inter connectivity failure

Working with distributed architecture

Most RTS involve processing on several different nodes. The system itself distributes the processing load among several processors. This introduces several challenges in design:

- Maintaining consistency
- Initializing the system
- Inter-processor interfaces
- Load Distribution
- Centralized resource allocation

Asynchronous communication

Most communication in the real world is asynchronous in nature, i.e. very few message interactions can be classified into the query response paradigm that works so well with remote procedural calls. Thus most RTS support state machine based design where multiple messages can be received in a single state. The next state is determined by the contents of the received message. State machines provide a very flexible mechanism to handle asynchronous message interactions. The flexibility comes with its own complexities.

Race condition and timing

RTS deals with timing issues by using timers. Timers are started to monitor the progress of events. If the expected events take place, the timer is stopped. If the expected event does not take place, the timer will timeout and recovery action will be triggered.

A race condition occurs when the state of a resource depends on timing factors that are not predictable. This is best explained with an example. Telephone exchanges have two way trunks which can be used by any of the two exchanges connected by the trunk. The problem is that both ends can allocate the trunk at more or less the same time, thus resulting in a race condition. Here the same trunk has been allocated for a incoming and an outgoing call. This race condition can be easily resolved by defining rules on who gets to keep the resource when such a clash occurs. The race condition can be avoided by requiring the two exchanges to work from different ends of the pool. Thus there will be no clashes under low load. Under high load race conditions will be hit which will be resolved by the predefined rules.

The main issue here is identifying race conditions. Most race conditions are not as simple as this one. Some of them are subtle and can only be identified by careful examination of the design.

3. Explain the misconception regarding the Real Time System.

Ans : The areas of research in real-time computing systems have not attracted enough attention. Part of the reasons are these misconceptions about the field.

1. **There is no science in real-time computing**
    - Mostly RTS designs are ad-hoc. But there can be ways developed to verify systems for timing bugs.

2. **Advances in supercomputers will take care of real time requirements**
    - The most important requirement of a real-time system is consistent output, not high throughput. Supercomputers would increase throughput but there might be scenarios where the processor might not be able to handle the traffic within the time-constraints.

3. **Real time computing is equivalent to fast computing**
    - Fast computing refers to minimizing the average response time whereas real-time computing refers to meeting deadlines. Predictability is the foremost goal, not speed.

4. **RTS research is performance engineering**
    - RTS research motive is to investigate the role time plays as a synchronization mechanism. Such problems are beyond traditional performance engineering.

5. **The problem is RTS design have all been solved in other areas of computer science or operations research**
    - In RTS, there are various exclusive problems which have not been solved in any other areas. Deriving the behavior of systems of arbitrary timings requires new techniques to be developed.

6. **It is not meaningful to talk about guaranteeing real-time performance, because we cannot guarantee that the hardware will not fail and the software is bug free or that the actual operating conditions will not violate the specified design limits.**
    - The goal should be to build a system to minimum time-constraint violations. There will always be situations of failure but that should not refrain us from building better systems.