Question A

Real time systems are used in a variety of computer science fields. Protection and space networks, networked multimedia systems, integrated automotive electronics, and so on are some examples. The correctness of a real time system's action is determined not only by the logical results of computations, but also by the physical instant at which these results are produced. A real time system's state varies as a function of physical time; for example, a chemical reaction tends to change its state long though its governing computer system has ceased.

A real-time machine can be decomposed into several subsystems based on this, including the managed object, the real-time computing system, and the human operator. A real-time computer device must respond to stimuli from the controlled object or the operator within time intervals set by its surroundings. A deadline is the point at which a result is made. The deadline is soft if the outcome is useful long after the deadline has passed; otherwise, it is solid. If missing a firm deadline could result in disaster, the deadline is difficult. Hard real-time services include command and control systems and air traffic control systems. Soft real-time applications include online transaction systems and airline reservation systems.

Real time system is becoming more and more popular. Air Traffic Control Networks, Networked Multimedia Systems, Command Control Systems, and other real time systems are examples. The correctness of the machine action in a real time system is determined not only by the logical outcomes of the computations, but also by the physical instant at which these results are produced. Real-time applications are categorised by a variety of perspectives, including factors external to the computer system and factors internal to the computer system. Hard and soft real time systems are given special attention. In hard real time systems, missing a deadline is disastrous, whereas in soft real time systems, it can result in a substantial loss. As a result, the most critical problem in these systems is the predictability of system behaviour.

Real time systems can be categorised in a variety of ways. The first two classifications, hard real time versus soft real-time, and fail-safe versus fail-operational, are based on the application's features, such as external conditions. The second three classifications, guaranteed-timeliness versus best-effort, resource-adequate versus resource-inadequate, and event-triggered versus time-triggered, are determined by the computer system's architecture and execution, such as internal variables.

Mixed systems combine two traditionally separate domains of computing: real-time and non-real time.  Real time systems respond to external events within a bounded interval of time (timeliness) and are often classified as hard real-time or soft real-time. Hard real time systems require a guarantee that all processing is completed within a given time constraint every time.

A late response may result in catastrophic consequences.  Thus, timeliness is a primary measure of correctness in a real time system.  Common examples of hard real time systems include nuclear power plants, avionics control systems, embedded braking systems for automobiles or trains, and signal processing systems employed by the Department of Défense.  Soft real time systems have a less rigorous notion of temporal correctness and the consequences of a late response are not catastrophic.  Examples of soft real time systems include telephone switches, on-line transaction systems, and electronic games.

Non-real time systems also have a notion of temporal correctness, but the response time is very subjective and seldom specified.  Most non real time systems strive for good average-case performance and tolerate occasional slow response times. Examples of non real time systems include desktop computers, workstations, information kiosks, and accounting systems.

Question B

Soft real time system is an operating system in which a critical real-time task gets priority over other tasks and retains that priority over other tasks until it completes. The response predefined time of soft real time systems are not very stringent, therefore missing the deadline only affects the performance and not the entire system. Size of the data file in soft real time systems is large. Soft real time systems are less restrictive in nature and the response predefined time of soft real time systems are not very stringent, therefore missing the deadline only affects the performance and not the entire system.  Soft real time systems miss deadlines occasionally. In a soft real time system a degraded operation in a rarely occurring peak load can be tolerated. Soft real time systems will slow down their response time if the load is very high. Most of the soft real time systems have larger databases and require long-term integrity of the system. In case of an error in a soft real time system, the computation is rolled back to previously established checkpoint to initiate a recovery action. Completion of task or activity by soft real time system probabilistic. Users of soft real time systems do not always obtain the validation. In this system response times are higher and peak load can be tolerated. Safety is not critical, it is less restrictive. In case of a soft real time system, computation is rolled back to previously established a checkpoint. An example of a soft real time system is 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 completed relatively on time the game will be enjoyable, and if not it may only lag a little.

A hard real time system also referred to as an immediate real time system is a hardware or software that must operate within the confines of a stringent predefined deadline. Usually, the application is considered to have failed if it does not complete its function within the specified timeline. The response predefined time of hard real time systems is in order of milliseconds and therefore, missing the deadline results in complete or massive system failure, and therefore this system should not miss the deadline. A hard-real time system is very restrictive. The response predefined time of hard real time systems is in order of milliseconds and therefore, missing the deadline results in complete or massive system failure, and therefore this system should not miss the deadline. Peak load performance should be predictable and should not violate the predefined deadlines. A hard real time system must remain synchronous with the state of the environment at all times. Most of the hard real time systems have small databases and occasionally require short-term integrity of the system. In case of an error in a hard real time system, the computation is rolled back or recovery is of limited use. Completion of the task or activity by hard real time systems is predefined or deterministic. The users of hard real time systems obtain validation when required. Hard real time is a system whose operation is incorrect whose result is not produced according to time constraint. In a hard real time system, the size of the data file is small or medium. The example of a hard real time system is the controlling the Airbag in every car, the controlling Airbag system should detect the crash and inflate rapidly the bag. The whole process takes more or less one-twenty-fifth of a second. Thus, if the system for example reacts with 1 second of delay the consequences could be mortal and it will be no benefit having the bag inflated once the car has already crashed.

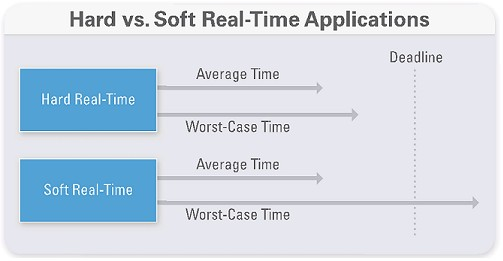


Figure @ Hard real time systems guarantee (when programmed correctly) that a deadline will be consistently met, while soft real time systems may periodically exceed the deadline.

Question C

Designing a real time system can be a difficult challenge. Most of the problem arises from the assumption that the real time system has to deal with real-world entities. These relationships can become extremely complicated. A standard real time system can communicate with thousands of such entities at the same time. The broadband wireless access system, for example, regularly manages communications from tens of thousands of customers. Each request is always linked differently by the system. Often, the exact sequence of events in the request can vary a lot.

In one case of the real time system that actually failed to do its job for example in data engineering is when the system itself failed to capture the deadline of the process that occurred in one of the ETL (Extract, Transform and Loading) process. Since there is no error detected, the data warehouse system allows the processes to continue processing the duplicate data until the storage runs out of capacity. This is not an issue if there is a person in-charge who keeps monitoring the query system from time to time, if a problem occurred, then the person-in-charge would simply stop the running processes. But it will be a major issue if it is implemented in the large-scale production stage where the most part of the system that is used by the stakeholders heavily relies on the data warehouse system itself. If the data warehouse is down due to resource bottleneck caused by the problem, then it will greatly affect the capability of the stakeholders to do their jobs since most of the data that they used came from the data warehouse itself.

The reason the real time system failed to detect the problem for this case is due to a script problem that was implemented in the running jobs. The script actually instructed the system to run the same process repeatedly, that’s why the system itself can’t calculate the deadline of the process. The problem can be solved and prevented by developing a clean script for the running jobs, and by monitoring the CPU load usage from the log files from time to time during the peak hours. Usually if the CPU load usage has reached more than 70% usage and took more than 10 minutes of execution time, then it clearly indicates a problem related to the job that runs the process. Once the problem has been identified, then it is up to the person-in-charge to stop the process and consult the person who runs the job.

In conclusion, designing a real time system can be a smart way whether we need to make sure that certain aspects of our software run in a certain amount of time, or whether we need to run our job reliably for a long period of time. If we are working on a mission-critical or safety-related project, the need for a real time system is obvious.

The real time system must react to external interactions in a predetermined period of time. Successful execution of the process relies on the proper and timely operation of the procedure. Design the hardware and software in the device to satisfy the real-time specifications. In order to satisfy these conditions, the off-clock detection mechanism and the software communication system concerned must operate under a short time budget. The system must satisfy these criteria for all calls made at any given time While the real-time technology provides rewards to the organization or individuals who used it, there is a time when the system has simply collapsed and caused a horrific tragedy for someone who relies heavily on it.

And where accurate scheduling and long-term durability are not absolute criteria for your project, installing a real time system will offer additional peace of mind that your software can continue to operate without interrupting the calculation or control procedure. If the device you are building will result in repair costs in the event that it is disrupted, the hardware and software costs needed to set up a real time system could well be worth investing in.

The real-time method would not actually make sense for any calculation or control project. Real-time operating systems usually run only one job at a time, and most real time systems do not have a user interface; in this case, a different system may be used to have user interfaces or user controls. Some projects also include hardware determinism where logic is applied to ASIC or FPGA. Yet, thousands of the real-time solutions are in the service today and will continue to be a feasible option for projects that require accurate scheduling and high reliability.

Reference

S. (2020, May 10). Difference between hard real time and soft real time system, from https://www.geeksforgeeks.org/difference-between-hard-real-time-and-soft-real-time-system/

Real-time computing. (2021, February 25).,from https://en.wikipedia.org/wiki/Real-time\_computing

Real-time operating system (rtos): Components, types, examples. (n.d.)., from https://www.guru99.com/real-time-operating-system.html

Laplante, P. A. Real-Time Systems Design and Analysis. 3rd edition Hoboken, NJ: Wiley; 2004; xxi, 505 p.

[Google Scholar](http://scholar.google.com/scholar_lookup?hl=en&publication_year=2004&pages=xxi+505&author=P.+A.+Laplante&title=Real-Time+Systems+Design+and+Analysis.)

Tanenbaum, A. S., Woodhull, A. S. Operating Systems: Design and Implementation. 3rd edition Upper Saddle River, NJ: Pearson Prentice Hall; 2006; xvii, 1054 p.

[Google Scholar](http://scholar.google.com/scholar_lookup?hl=en&publication_year=2006&pages=xvii+1054&author=A.+S.+Tanenbaum&author=A.+S.+Woodhull&title=Operating+Systems%3A+Design+and+Implementation.)