Instructions for Authors of SBC Conferences Papers and Abstracts

Giovani Ferreira[1], **Gustavo Coimbra**[1], **Eduardo San**[1]

[1]CEUB - Centro Universitário de Brasília
Caixa Postal 4488 – 70.904-970 – Brasília – DF – Brazil

Abstract.

Resumo.

1. Introduction

Nowadays, most business operations are supported by IT systems. Therefore, their availability is critical to keep business running smoothly and continuously. In order to provide high quality IT services, a well-managed data center is required to house computer servers, storage systems,network devices, and their associated components. Downtime of the data center can be costly resulting in production and business losses so that the high availability requirement of the data center is needed. Data center always requires a non-stopped service or 24x7 availability. The availability of data center mainly requires the protection of computer equipment and personnel. There are some potential hazards which may occur and impact the availability of data center. Fire is the most potential hazard which can create severe effects on data center. Fire can occur in a data center by mechanical failure, intentional arson, or natural causes. Data Center Physical Security Ontology for Automated Evaluation (Abstract).

Data Center is a facility used for housing a large amount of computer and communications equipment maintained by an organization for the purpose of handling the data necessary for its operations. Glossary of MMC Terminology. Available: http://msdn.microsoft.com/en-us/library/bb246417(VS.85).aspx.

2. Related Concepts

2.1. Time

It's important to understand what is time and it's related concepts. For this purpose, the definitions provided by [Kopetz 2011] will be considered:

- The flow of time is a directed time line that extends from the past into the future.
- A cut of the time line is called an instant.
- Any ideal occurrence that happens at an instant is called an event.
- The present point in time, now, is a very special event that separates the past from the future.
- An interval on the time line, called a duration, is defined by two events, the start event and the terminating event of the interval.
- The instant when a result must be produced is called a deadline.

In the context of deadlines, [Kopetz 2011] also provides a good definition, classifying them in three different ways:

1. Soft deadline: If a result has utility even after the deadline has passed,
2. Firm deadline: If a result does not matter after the deadline.
3. Hard deadline: If severe consequences could result if a firm deadline is missed.

2.2. Real-time systems

For [Kopetz 2011], a real-time computer system is a computer system where the correctness of the system behavior depends not only on the logical results of the computations, but also on the physical time when these results are produced. [Lichtenstein et al. 1985] states that a real-time system is a reactive system, that is, it must react to stimuli from its environment within time intervals dictated by its environment. According to [Stankovic and Ramamritham 1990], a real-time computer system can be classified in 2 different ways:

1. Static real-time system: where all deadlines can be guaranteed a priori.
2. Dynamic real-time system: large, complex, distributed, adaptive, contain many types of timing constraints, need to operate in a highly nondeterministic environment, and evolves over a long system lifetime.

The type of the deadline affects the real-time system. If the system must met at least one hard deadline, it's called a hard real-time computer system. If no hard deadline exists, then the system is called a soft real-time computer system. The design of a hard real-time system is fundamentally different from the design of a soft real-time system. While the first must sustain a guaranteed temporal behavior under all specified load and fault conditions, it is permissible for the second to miss a deadline occasionally [Kopetz 2011]. [Stankovic 1996] also points that hard real-time systems usually cause several consequences, even death, when missed an important deadline.

2.3. Tasks

For [Stankovic 1996] tasks can be classified as:

- Periodic task: activated every T units. The deadline for each activated instance may be less than, equal to, or greater than the period T.
- Aperiodic task: activated at unpredictable times.
- Sporadic task: an aperiodic task with the additional constraint that there is a minimum interarrival time between task activations.

Real-time scheduling is the process of creating start and finish times for sets of tasks such that all timing, precedence, and resource constraints are met.

3. The RTDMS

The RTDMS, that stands for Real Time Datacenter Monitor System, is a real time system aiming to provide a reliable and solid way to measure and monitor the conditions of a datacenter. Using sensors for temperature, humidity and flood, the RTDMS is able to quickly discover adverse conditions in a datacenter and reduce the warning time, so the support team have more time to take the appropriate interventions and avoid worse situations.

3.1. Tasks Model

The RTDMS have the following tasks model:

- Alarm Task: Responsible to warn that something went wrong in datacenter. The deadline is hard. To assure that the deadline will be fulfilled, this task blocks the execution of the other tasks and monopolizes the hardware.

Task Name	Period (ms)	Computation Time (ms)	Deadline Type
Alarm	Aperiodic	400	Hard
Check Flood	1000	30	Soft
Check Temperature	1000	820	Soft
Check Humidity	1000	20	Soft

- Check Flood: This task uses a rain sersor to check if there is a water flood in datacenter. The deadline is soft because if it fails sometimes, it's not a problem. An undetected flood a second and half ago does not reach very high levels in this hiatus. If the tasks detects a failure in the sensors, it activates the alarm task to warn that something is wrong.
- Check Temperature: Responsible to collect the data from the temperature sensors and treat them to provide a realiable measure. The deadline is soft, the temperature will not change abruptly in one second and half. The validation of the temperatures measured can deal with a failure of one sensor (measuring a absurd temperature value or not measuring anything).
- Check Humidity: Collect the data from humidity sensors and process it to provide a valid humidity measure. If the dealine fail sometimes, it's not a problem, humidity needs more time to significantly changes. This task can deal with failure of one sensor, correcting the measure and providing a correct final result.

3.1.1. Time Restrictions

The RTDMS time restriction is the warning time. In the moment that RTDMS detects an adverse condition, the alarm task must be execute instantly. The failure of this task, not warning or warning with delay, can cause big problems to the owners of the datacenter, including the loss of all data and equipments.

3.1.2. Scalability

The rate monotonic, which assigns a higher priority to a task with shorter period, was considered an optimal fixed priority algorithm by [Liu and Layland 1973], which proved that for a set of n periodic tasks with unique periods, a feasible schedule that will always meet deadlines exists if the CPU utilization is below a specific bound (depending on the number of tasks). The schedulability test for Rate Monotonic Scheduler is:

$$U = \sum_{i=1}^{n} \frac{C_i}{P_i} \leq n(2^{\frac{1}{n}} - 1)$$

Where U is the CPU utilization, C_i is the computation time of the task i, P_i is the release period (with deadline one period after), and n is the number of processes to be scheduled. The set of tasks for RTDMS was mathematically proved as scalable:

1. Check Flood:

$$C_1 = \frac{C_1}{P_1} = \frac{30}{1500} = 0.0200$$

2. Check Temperature:

$$C_2 = \frac{C_2}{P_2} = \frac{820}{1500} = 0.5467$$

3. Check Humidity:

$$C_3 = \frac{C_3}{P_3} = \frac{20}{1500} = 0.0133$$

- Schedulability

$$U = \sum_{i=1}^{3} \frac{C_i}{P_i} = 0.7600 \leq 3(2^{\frac{1}{3}} - 1) = 0.7797$$

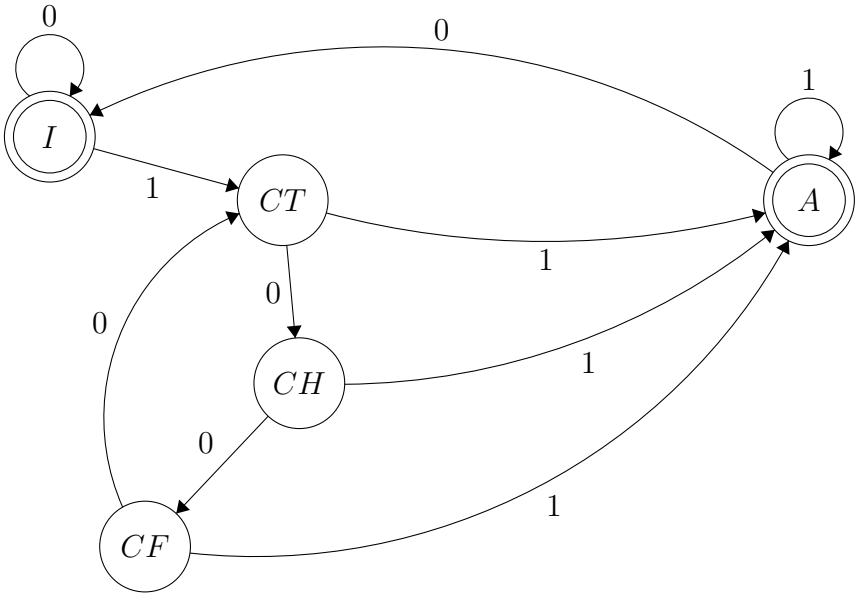
$$U = 0.7600 \leq 0.7797$$

Besides the mathematical proof, the RTDMS's set of tasks were also submitted to simulation in XXXX(nome da ferramenta):

[Imagem da ferramenta mostrando que eh escalonavel]

3.2. System's Flow

The system follows a flow that aims to provide the maximum of autonomy to the task with hardest deadline, executing it with total hardware control to ensure that the deadline will be fulfilled. The flow can be seen in the following state machine:



- Initial State (I): Outputs '1' when the system is activated;
- Alarm Task (A): Outputs '0' when is turned off manually;
- Check Flood (CF): Outputs '1' when detects a considerable amount of water or a failure in the sensor;
- Check Temperature (CT): Outputs '1' when identifies a temperature above what it should be;
- Check Humidity (CH): Outputs '1' when the humidity is above the acceptable limit.

4. Experiments and Evaluation

The prototype implementation of RTDMS consisted in the following hardware equipments:

- 1x - Arduino Uno
- 3x - Temperature and humidity sensors
- 1x - Rain sensor
- 1x - Buzzer
- 1x - Potentiometer

This prototype used the NilRTOS[1] library, that offers an implementation of the Rate Monotonic algorithm.

The prototype's source code[2] is available in github.

5. Conclusion

Referências

Kopetz, H. (2011). *Real-time systems: design principles for distributed embedded applications*. Springer Science & Business Media.

Lichtenstein, O., Pnueli, A., and Zuck, L. (1985). *The glory of the past*. Springer.

Liu, C. L. and Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61.

Stankovic, J. A. (1996). Real-time and embedded systems. *ACM Computing Surveys (CSUR)*, 28(1):205–208.

Stankovic, J. A. and Ramamritham, K. (1990). What is predictability for real-time systems? *Real-Time Systems*, 2(4):247–254.

[1]https://github.com/greiman/NilRTOS-Arduino

[2]https://github.com/giovanifss/Datacenter-Monitor