Earliest Deadline Late Server (EDLS)

Masrur-Jamil Prochchhod

May 2024

Abstract

In real-time systems, meeting task deadlines is crucial to ensure system reliability and performance. By using the idea of a "late server" to manage jobs that are past due, the Early Deadline Late Server (EDLS) scheduling algorithm has shown promise in addressing this issue. The EDLS algorithm is thoroughly reviewed in this paper, which also looks at its implementation issues, theoretical underpinnings, and real-world applications. We discuss the key features of the EDLS algorithm, including its scheduling strategy, deadline enforcement mechanism, and methods for handling late tasks. In addition, we investigate the performance characteristics of EDLS in relation to alternative scheduling algorithms, emphasizing its advantages and disadvantages under different conditions. Furthermore, we explore new developments and avenues for future study in EDLS scheduling, such as optimization strategies, integration with upcoming technologies like autonomous systems, and applications in many fields including cloud computing and cyber-physical systems. Our goal in writing this paper is to give researchers and practitioners a thorough grasp of the EDLS algorithm and how it affects the design and development of real-time systems.

1 Introduction

Among the various scheduling algorithms developed to address this challenge, the Early Deadline Late Server (EDLS) scheduling algorithm has gained notable attention. EDLS is designed to improve the handling of aperiodic tasks within the context of real-time systems, aiming to provide better performance in terms of deadline adherence while maintaining system efficiency. This algorithm addresses some of the limitations found in traditional real-time scheduling approaches by dynamically adjusting to the system's state and workload, thus ensuring more robust and predictable scheduling outcomes.

The primary objective of this paper is to provide a comprehensive review of the EDLS scheduling algorithm. This includes an in-depth look at its implementation, exploring how it can be practically applied within real-time systems, and discussing the theoretical foundations that support its operation. Furthermore, the paper will examine various real-world applications where EDLS has been successfully implemented, highlighting the benefits and potential improvements it brings to real-time task scheduling.

2 Importance of Meeting Task Deadlines in Real-Time Systems

Real-time systems are characterized by the necessity to process and respond to inputs within strict timing constraints. These constraints are often categorized into hard, firm, and soft deadlines. Hard deadlines are those where any delay is unacceptable and can result in system failure, as seen in safety-critical applications like pacemakers or automotive airbag systems. Firm deadlines, while still critical, allow for occasional missed deadlines without catastrophic consequences, though performance degradation can occur. Soft deadlines are more lenient, where occasional

deadline misses might only result in reduced quality of service rather than outright failure.

In all these scenarios, ensuring timely task completion is crucial. Failure to meet deadlines in hard real-time systems can lead to life-threatening situations or severe financial loss, while in soft real-time systems, it can lead to suboptimal performance and user dissatisfaction. Therefore, robust and efficient scheduling algorithms that can handle varying loads and ensure adherence to deadlines are essential components of real-time systems.

3 Brief Introduction to the Early Deadline Late Server (EDLS) Scheduling Algorithm

The Early Deadline Late Server (EDLS) is a scheduling algorithm tailored for real-time systems that manage both periodic and aperiodic tasks. EDLS is designed to dynamically allocate processing time to aperiodic tasks without significantly impacting the execution of periodic tasks, which are typically scheduled using algorithms like Rate Monotonic Scheduling (RMS) or Earliest Deadline First (EDF).

The fundamental principle of EDLS is to exploit the slack time—the time during which the processor would be idle—to execute aperiodic tasks. By doing so, EDLS aims to maximize the utilization of the processor while ensuring that periodic tasks meet their deadlines. The algorithm calculates the earliest deadline among the periodic tasks and schedules the aperiodic tasks during the intervals when the processor would otherwise be idle or underutilized. This dynamic adjustment helps in improving the response times of aperiodic tasks and ensures a more balanced and efficient system operation.

4 Objective of the Paper

The objective of this paper is multifaceted, focusing on a detailed examination of the EDLS scheduling algorithm from several perspectives:

1. **Review of EDLS**: The paper will provide an extensive review of the EDLS algorithm, discussing its origins, development, and the problems it aims to solve within the realm of real-time systems.

2. **Implementation**: The paper will delve into the practical aspects of implementing EDLS in real-time systems. This includes a discussion on the necessary system requirements, configuration parameters, and the steps involved in integrating EDLS with existing scheduling frameworks.

3. **Theoretical Basis**: A thorough exploration of the theoretical foundations of EDLS will be conducted. This will involve examining the mathematical models and algorithms that underpin EDLS, as well as analyzing its performance guarantees and limitations.

4. **Real-World Applications**: The paper will highlight various real-world scenarios where EDLS has been applied. This will include case studies from different industries, demonstrating how EDLS has been used to enhance system performance, reliability, and deadline adherence. Specific examples will illustrate the practical benefits and challenges encountered during the implementation of EDLS.

By providing a detailed review and analysis of the EDLS scheduling algorithm, this paper aims to contribute to the ongoing development and optimization of real-time systems. It seeks to offer valuable insights for researchers, practitioners, and engineers working in the field, facilitating better understanding and application of advanced scheduling techniques to meet the ever-growing demands of real-time processing environments.

5 Mathematical Algorithm

Initialization:

- Set the system clock to $t = 0$.

Figure 1: Grantt Chart of EDLS

- Initialize the set of tasks $\tau = \{\tau_1, \tau_2, ..., \tau_n\}$.
- Each task τ_i has attributes: period T_i, execution time C_i, and relative deadline D_i.

Task Arrival:

- When a task τ_i arrives at time t, compute its absolute deadline $d_i = t + D_i$.

Priority Assignment (EDF):

- Use Earliest Deadline First (EDF) to select the task with the earliest absolute deadline d_i.

Execute Task:

- Execute the selected task τ_i for its execution time C_i.

Update Virtual Deadline:

- After completing the task, update the virtual deadline $V(t)$.

Adjust Priorities:

- Recompute the priorities of remaining tasks based on their deadlines and the current time t.

Repeat:

- Continue the process until all tasks are completed.

Gantt Chart Representation:

```
Time:  0     1     2     3     4     5     6
Tasks: [1] [2] [2] [3]
```

At $t = 4$: Tasks τ_1 and τ_2 are ready again.
2[

6 Video Streaming Service Scheduler

Components:

- Frame Manager: Responsible for receiving and managing video frames from the video server.
- Frame Queue: Maintains the list of frames awaiting delivery.
- Scheduler: Implements the Earliest Deadline Late Server algorithm to select the next frame for delivery.

Workflow:

1. Frame Arrival: Video frames are received from the video server at irregular intervals. Each frame specifies its arrival time and deadline.
2. Frame Queueing: Newly arrived frames are added to the frame queue.
3. Scheduling: When it's time to select the next frame for delivery, the scheduler examines the frame queue and selects the frame with the earliest deadline among frames that arrived before or at the current time.
4. Frame Delivery: The selected frame is delivered to the user for playback.
5. Deadline Monitoring: The system monitors frame delivery to ensure that deadlines are met. If a frame exceeds its deadline, appropriate action is taken, such as buffering or quality adjustment.
6. Frame Completion: Upon frame delivery, the frame is removed from the frame queue.

Example:

Let's consider three video frames in the system:

- Frame 1: Arrival Time = 0 ms, Deadline = 30 ms

- Frame 2: Arrival Time = 20 ms, Deadline = 50 ms
- Frame 3: Arrival Time = 40 ms, Deadline = 70 ms

Assuming the frames are delivered in the order of arrival:

- At time 0, Frame 1 arrives and is scheduled for delivery.
- At time 30, Frame 1 completes its delivery.
- At time 20, Frame 2 arrives and is scheduled for delivery (since it arrived before Frame 1's deadline).
- At time 50, Frame 2 completes its delivery.
- At time 40, Frame 3 arrives but cannot be scheduled for immediate delivery since its deadline is after Frame 2's deadline.
- At time 70, Frame 3 completes its delivery.