Dynamic Sporadic Server

Moiz Zaheer Malik
B.Eng. Electronic Engineering
Hochschule Hamm-Lippstadt
Lippstadt, Germany
moiz-zaheer.malik@stud.hshl.de

***Abstract*—In real-time systems, where tasks have different levels of critical importance, it is essential to serve aperiodic (irregular, event-driven) tasks while ensuring that the deadlines of high-priority periodic tasks are not violated. The Dynamic Sporadic Server (DSS) is a scheduling method designed for Earliest Deadline First (EDF) systems that addresses this problem.**

DSS is defined by a period T_s and a budget C_s, but unlike traditional sporadic servers, it does not restore the full budget at every period. Instead, when an aperiodic task arrives, DSS assigns it a deadline and restores only the amount of budget that was actually used. This approach allows the processor to reach full (100 percent) utilization while ensuring that all deadlines are still met. DSS improves the response time of aperiodic tasks without compromising the guarantees of periodic tasks.

Originally introduced by Spuri and Buttazzo[1], DSS has been widely studied for its efficiency in managing mixed task sets under EDF scheduling. This report reviews the theory behind DSS, describes its operation (including budget management), highlights its advantages over traditional methods, and discusses potential application areas.

Section Summaries

- **Abstract**
  This paper explains how the Dynamic Sporadic Server (DSS) helps in running aperiodic tasks without missing deadlines for periodic tasks. DSS works with EDF scheduling and tries to keep the processor busy as much as possible. It gives better response times and still makes sure that periodic jobs run on time.
- **I. Introduction**
  In real-time systems, we often have both periodic tasks (which repeat regularly) and aperiodic tasks (which come randomly). The problem is how to serve aperiodic tasks quickly without disturbing the periodic ones. DSS is a good solution for this in EDF-based systems.
- **II. Background**
  This section explains the basic types of tasks: periodic, sporadic, and aperiodic. It also talks about EDF scheduling and how it decides which task to run based on deadlines. The condition $U_p + U_a \leq 1$ is important to make sure deadlines are met when using EDF.
- **III. Sporadic and Dynamic Servers**
  Here, I explain the difference between the old Sporadic Server and the new DSS. The normal Sporadic Server works with fixed priority and resets budget differently. DSS works with dynamic deadlines and only adds back the budget that was used. This makes it more flexible under EDF.
- **IV. Dynamic Sporadic Server Simulation**
  I describe a small example where two periodic tasks run along with DSS, which serves incoming aperiodic jobs. It shows how DSS gets activated when needed, sets a deadline, and uses its budget. This helps to understand how DSS works in real scenarios.
- **V. DSS Simulation Overview**
  This part shows a C++ simulation of DSS and how it handles aperiodic requests while letting periodic tasks run on time. It also explains how the budget is used, when it's refilled, and how the system stays within the CPU usage limits.
- **VI. Comparison With Other Servers**
  DSS is compared with other types of servers like Polling, Deferrable, and normal Sporadic Servers. The table shows how DSS is better in some ways — especially in using CPU time more efficiently and responding faster to aperiodic jobs.
- **VII. Applications and Use Cases**
  DSS is used in many systems where real-time response is important. This includes cars (ECUs), robots, avionics, multimedia, and medical devices. It's useful where both regular and unexpected tasks happen, and all need to be handled without delays.
- **VIII. Limitations and Practical Considerations**
  DSS is helpful but not always easy to implement. It needs dynamic deadline management and careful handling of budget. It works well with EDF but not with fixed-priority systems. Things like context switching, server overload, and multicore systems also make it harder.
- **IX. Conclusion**
  DSS gives a smart way to handle aperiodic tasks in real-time systems using EDF. It keeps the system efficient and avoids deadline misses. Even though it's more complex than other servers, it offers good control

Reference Contributions

Here's how each reference helped me while writing this seminar paper:

- **[1] Magnus Bengtsson and Elias Karlsson (2020)**
  I used this to give a real-life example from the automotive industry in the applications section. It

helped me show how DSS can be used in ECUs to handle braking events without delaying periodic engine control tasks.

- **[2] Giorgio C. Buttazzo (2011)**
  This book was the main source for explaining how DSS works. I used it in multiple sections, especially for the schedulability condition $U_p + U_s \leq 1$, the rules for activation and replenishment, and the comparison table between different server types.
- **[3] John Kingston and Ying Wang (2023)**
  This was used in the limitations section when I talked about DSS on multicore systems. It helped explain why implementing DSS on multiple cores can be more complicated.
- **[4] Philip A. Laplante (2011)**
  I used this to describe the basics of periodic, sporadic, and aperiodic tasks. It was also helpful in explaining the need for server-based scheduling before introducing DSS.
- **[5] C. L. Liu and J. W. Layland (1973)**
  This paper was referenced for the EDF schedulability rule $U_p \leq 1$, which I included in the background section. It's important for understanding how EDF handles periodic tasks before adding aperiodic ones through DSS.
- **[6] Barry Sprunt, Lui Sha, and John Lehoczky (1989)**
  I used this to explain the original Sporadic Server. It was helpful when comparing DSS with older server models and showing how fixed-priority servers manage budget differently.
- **[7] Marco Spuri and Giorgio C. Buttazzo (1994)**
  This paper defines the Dynamic Sporadic Server itself. It was the main source for how DSS sets deadlines, manages budget, and maintains EDF feasibility. I used it throughout the theory sections.
- **[8] University of British Columbia (2025)**
  These course notes helped me when writing the simulation section. I followed their explanation of how the DSS activates, runs, and replenishes, and also used them for the timeline and logic in the C++ code example.

REFERENCES

[1] Marco Spuri and Giorgio C. Buttazzo. "Efficient Aperiodic Service under Earliest Deadline Scheduling". In: *Proceedings of the IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 1994, pp. 2–11.