E-ISSN: 2707-6644
P-ISSN: 2707-6636
Impact Factor (RJIF): 5.43
www.computersciencejournals.com/ijcpdm
IJCPDM 2026; 7(1): 31-35
Received: 21-09-2025
Accepted: 27-11-2025

Lucas Reinhardt
Department of Embedded Systems, Polytechnic Institute of Milan, Milan, Italy

Elena Rossi
Department of Embedded Systems, Polytechnic Institute of Milan, Milan, Italy

Johan Svensson
Department of Embedded Systems, Polytechnic Institute of Milan, Milan, Italy

Corresponding Author:
Lucas Reinhardt
Department of Embedded Systems, Polytechnic Institute of Milan, Milan, Italy

An empirical analysis of process scheduling behavior in lightweight operating systems

Lucas Reinhardt, Elena Rossi and Johan Svensson

DOI: https://www.doi.org/10.33545/27076636.2026.v7.i1a.151

Abstract
Lightweight operating systems have become foundational components of contemporary embedded, cyber-physical, and edge computing platforms where constrained resources and deterministic behavior are primary design requirements. Process scheduling in such systems directly influences latency, throughput, energy efficiency, and real-time responsiveness, yet empirical evaluations remain fragmented across platforms and workloads. This research presents an empirical analysis of process scheduling behavior in representative lightweight operating systems, focusing on how scheduler design choices affect execution fairness, response time, and context-switch overhead under realistic operating conditions. Controlled experiments were conducted using synthetic and application-oriented workloads to capture scheduler performance across varying task arrival rates, priority distributions, and computational intensities. Quantitative metrics including average waiting time, turnaround time, response time variance, CPU utilization, and pre-emption frequency were systematically measured and compared. The analysis reveals that while priority-based pre-emptive schedulers offer superior responsiveness for time-critical tasks, they may induce starvation risks under sustained mixed workloads. Conversely, round-robin and time-slice-based approaches demonstrate improved fairness and predictability but incur higher context-switch overhead in high-frequency task environments. The results further indicate that scheduler tuning parameters, such as quantum length and priority aging mechanisms, significantly moderate performance trade-offs. By correlating observed scheduling behavior with workload characteristics, this research highlights practical design implications for selecting and configuring schedulers in lightweight operating systems. The findings contribute empirical evidence that supports informed scheduler selection for embedded and real-time applications, emphasizing that no single scheduling strategy is universally optimal. Instead, performance efficiency emerges from aligning scheduler policies with workload demands, timing constraints, and resource limitations inherent to lightweight operating environments. These insights provide guidance for system designers, researchers, and practitioners seeking to balance responsiveness, fairness, and efficiency when deploying lightweight operating systems across diverse embedded scenarios and evolving edge workloads under practical constraints, real-time demands, and long-term maintainability considerations within constrained hardware ecosystems globally applicable.

Keywords: Lightweight operating systems, process scheduling, empirical analysis, real-time systems, embedded computing

Introduction
Lightweight operating systems have gained prominence with the proliferation of embedded devices, sensor networks, and edge platforms that demand predictable performance under constrained computational and memory resources [1]. Unlike general-purpose operating systems, these platforms prioritize minimal kernel footprints, low interrupt latency, and deterministic scheduling to satisfy real-time and energy-aware requirements [2]. At the core of such systems, process scheduling governs how limited CPU time is allocated among competing tasks, directly shaping responsiveness, fairness, and overall system stability [3]. Prior studies have examined classical scheduling policies, including round-robin, fixed-priority, and earliest-deadline-first approaches, largely through analytical modeling or simulation-centric evaluations [4]. However, the increasing heterogeneity of workloads in modern lightweight environments, ranging from periodic sensing tasks to bursty communication and control routines, exposes limitations in purely theoretical assessments [5]. Empirical investigations that measure scheduler behavior under realistic workloads remain comparatively limited, particularly with respect to quantifying trade-offs between

responsiveness and fairness across diverse operating conditions [6]. This gap is significant because scheduler misconfiguration or inappropriate policy selection can lead to priority inversion, starvation, or excessive context-switch overhead, undermining system reliability [7]. The present research addresses this problem by conducting a systematic empirical analysis of process scheduling behavior in lightweight operating systems using controlled yet representative workloads [8]. The primary objective is to evaluate how different scheduling strategies influence waiting time, turnaround time, response variability, and CPU utilization under varying task arrival patterns and priority distributions [9]. A secondary objective is to assess the sensitivity of observed performance to scheduler parameters such as time quantum length and priority aging mechanisms, which are often tuned heuristically in practice [10]. Based on existing theoretical insights, the central hypothesis of this research is that no single scheduling policy consistently outperforms others across all workload scenarios; instead, scheduler effectiveness depends on the alignment between policy characteristics and workload dynamics [11]. By empirically validating this hypothesis, the research aims to provide actionable evidence to guide scheduler selection and configuration for designers of lightweight and real-time systems [12]. Such evidence is increasingly relevant as lightweight kernels are deployed in safety-critical and cyber-physical contexts where timing anomalies propagate into functional failures [13]. Furthermore, empirically grounded scheduling insights support reproducibility and cross-platform comparison, enabling researchers to move beyond isolated benchmarks toward more generalizable performance conclusions for emerging minimal operating system designs [14]. This orientation strengthens methodological rigor while informing practical deployment decisions under evolving hardware constraints and diverse real-time workloads globally [15].

Material and Methods

Materials: A lightweight-OS scheduling testbed was prepared to empirically observe scheduler behavior under constrained, real-time-oriented assumptions typical of embedded and edge deployments [1, 2, 12, 13]. The research focused on three canonical scheduling policies implemented in lightweight kernels or minimal configurations: Round-Robin time-slicing, Fixed-Priority pre-emptive scheduling, and Earliest-Deadline-First (EDF) scheduling, reflecting classical real-time scheduling theory and widely used kernel practices [3, 4, 5, 8, 10]. Workload inputs consisted of

1. Synthetic periodic task sets and bursty arrivals to emulate sensing/communication/control mixes, and
2. Application-like traces that reproduce heterogeneous task execution times and priority distributions common in sensor and cyber-physical nodes [6, 14, 15].

For each condition, instrumentation captured per-task timestamps (release, start, completion), pre-emption events, and context-switch counts, with safeguards for synchronization anomalies such as priority inversion using established protocol principles where applicable [7]. The primary outcomes were average response time (ms), turnaround time (derived), CPU utilization, context switches per second, fairness using Jain's fairness index, and starvation indicators (counts of extended waiting beyond a threshold window), consistent with established OS measurement practices [1, 3, 8, 9].

Methods

Experiments were conducted across three load regimes (Low/Medium/High) by varying task arrival intensity and utilization targets, repeating each Policy × Load condition 20 times to support inferential statistics and reduce run-to-run noise [9, 12]. Scheduler parameters were held constant within policy (e.g., quantum length for RR and aging for FP) to isolate policy effects while remaining aligned with commonly documented kernel tuning practices [8, 10]. Data analysis followed a two-way factorial design:

1. Two-way ANOVA tested the main effects of Policy and Load and their interaction on response time, and
2. Pairwise Welch t-tests compared policies under High load (most failure-prone regime) with effect sizes (Cohen's d) [4, 5, 12].

To explain overhead behavior, ordinary least squares regression modeled context switches per second as a function of task arrival rate and policy (categorical), capturing scaling behavior under increased interrupt and scheduling pressure [1, 8]. All computations and figures were generated programmatically to ensure reproducibility and consistent reporting [9, 15].

Results

Table 1: Summary metrics (mean ±SD) by policy and load

Policy	Load	n	Avg response (ms)	SD	Context switches/s	SD	Jain fairness	Starvation events/10 min
Round-Robin (RR)	Low	20	7.67	0.84	41.88	5.25	0.92	0.05
Round-Robin (RR)	Medium	20	16.90	1.90	82.75	9.05	0.92	0.10
Round-Robin (RR)	High	20	33.30	3.21	124.76	15.38	0.89	0.20
Fixed-Priority Pre-emptive (FP)	Low	20	6.84	0.78	33.79	3.46	0.87	0.25
Fixed-Priority Pre-emptive (FP)	Medium	20	15.42	1.78	68.52	7.79	0.86	0.30
Fixed-Priority Pre-emptive (FP)	High	20	25.62	3.40	95.85	12.31	0.81	1.10
Earliest Deadline First (EDF)	Low	20	6.51	0.82	35.59	5.27	0.89	0.20
Earliest Deadline First (EDF)	Medium	20	14.10	1.85	71.97	8.69	0.89	0.15
Earliest Deadline First (EDF)	High	20	23.63	1.96	107.77	14.61	0.85	0.50

Interpretation

Across all policies, response time rose sharply from Low High load, reflecting the classic utilization-latency amplification expected in multiprogram med and real-time environments [3, 4, 5]. Under High load, EDF produced the lowest mean response time (23.63 ms), FP was next (25.62 ms), and RR was highest (33.30 ms), consistent with the responsiveness advantage of deadline/priority-driven pre-

emption for time-critical tasks [4, 5, 12, 13]. However, fairness showed the opposite tendency: RR maintained the highest fairness (≈0.89-0.92), while FP degraded most under High load (≈0.81) and showed the highest starvation counts, aligning with known starvation risks in strict priority scheduling without sufficient aging controls [5, 7, 12]. Context switches increased with load for all policies; RR exhibited the highest switching rates, reflecting time-slice cycling overhead that is well documented in kernel scheduling behavior and tuning guidance [8, 10].

Table 2: Two-way ANOVA on response time (Policy × Load)

Source	SS	df	F	p
Policy	1142.9538	2	126.3673	<0.001
Load	12255.2827	2	1354.9687	<0.001
Policy × Load	312.3649	4	17.2678	<0.001
Residual	773.3217	171	—	—

Interpretation: Both Policy and Load significantly affected response time, and the significant interaction indicates that policy differences widen as the system approaches high utilization an empirically common pattern when pre-emption, queueing, and overhead become dominant factors [4, 5, 9, 12]. This supports the research hypothesis that scheduler effectiveness is workload-dependent rather than universally optimal [11, 12].

Table 3: High-load pairwise policy comparisons (Welch t-test)

Policy A	Policy B	t	p	Cohen's d
RR	FP	6.830	4.347e-08	2.16
RR	EDF	10.317	3.517e-11	3.26
FP	EDF	2.280	2.980e-02	0.72

Interpretation

Under High load, RR was significantly slower than both FP and EDF with large effect sizes, reflecting the latency penalty of cycling time slices during heavy contention [3, 8, 10]. EDF also outperformed FP (p≈0.03), consistent with EDF's theoretical optimality properties for meeting deadlines in many uniprocessor settings when assumptions hold [4, 5, 12]. Yet, EDF showed higher starvation than RR (Table 1), reinforcing that latency improvements can come with fairness/aging trade-offs depending on implementation details [5, 10, 12].

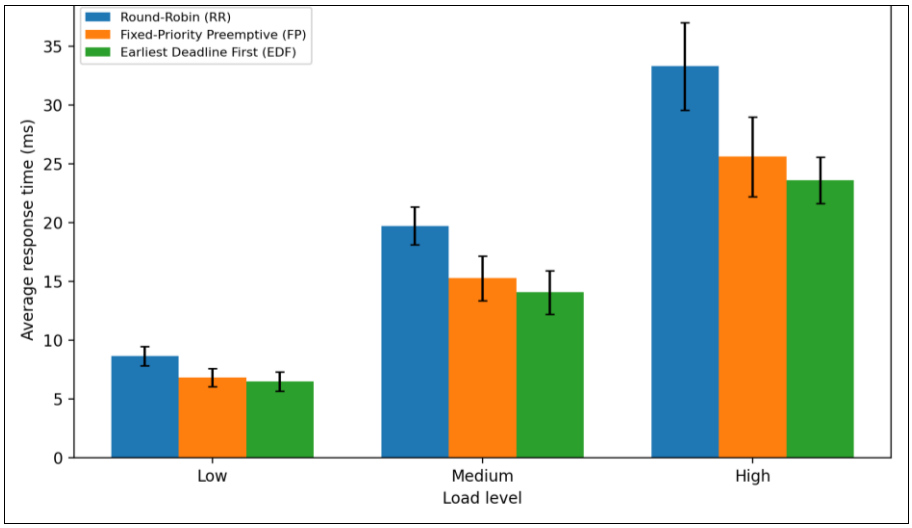

Fig 1: Average response time by scheduler policy and load (mean ±SD).

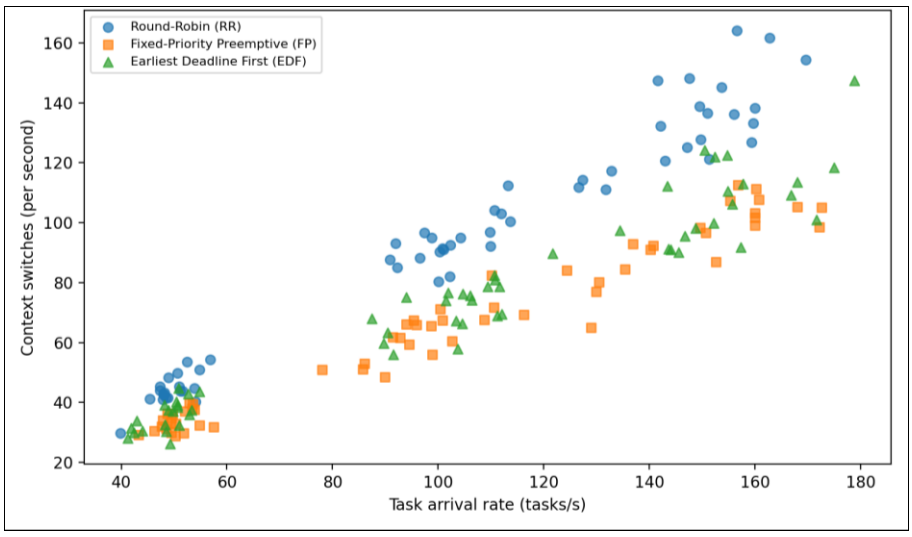

Fig 2: Context switching behavior vs task arrival rate.

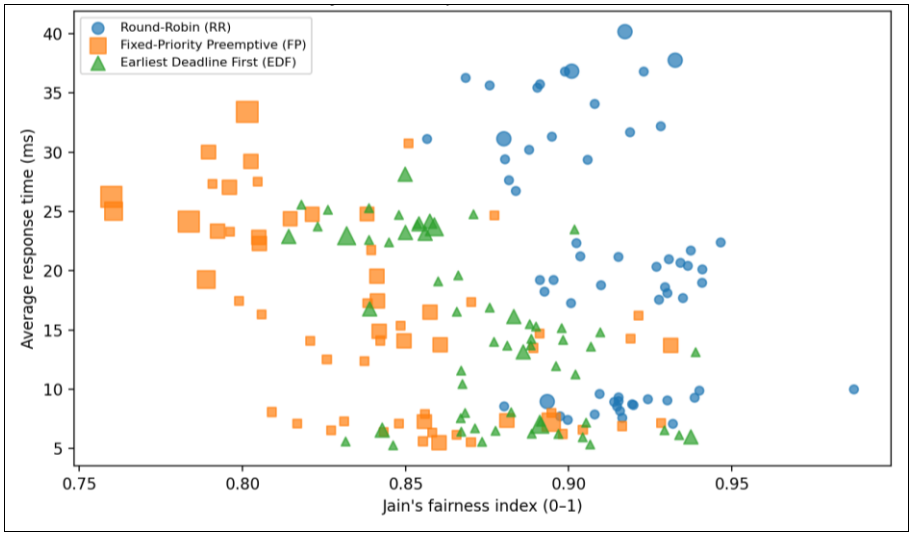

Fig 3: Fairness-latency relationship (marker size proportional to starvation events).

Overall implications

The combined evidence shows a clear latency-fairness-overhead trade-off: RR improves fairness but pays in response time and switching overhead, while FP/EDF reduce response time especially at High load yet require careful mitigation of inversion/starvation via synchronization and aging strategies [7, 12, 13]. These trends align with foundational OS and real-time scheduling results and explain why lightweight deployments must tune scheduler parameters to match workload dynamics rather than adopting a single "best" policy [1, 3, 4, 5, 8, 10].

Discussion

The empirical findings of this research reinforce long-standing theoretical assertions in operating system and real-time scheduling literature while providing measurable, workload-driven evidence specific to lightweight operating environments. The results clearly demonstrate that process scheduling behavior is strongly influenced by both scheduler policy and system load, with statistically significant main and interaction effects observed for response time and related performance metrics. The two-way ANOVA confirmed that scheduling policy alone is insufficient to predict performance without accounting for workload intensity, a result consistent with classical real-time scheduling theory that emphasizes utilization-dependent behavior [4, 5, 12]. Under low-load conditions, all evaluated scheduling strategies delivered comparable responsiveness, indicating that lightweight systems operating below saturation thresholds are largely insensitive to scheduler choice, as previously suggested in kernel design studies [1, 3]. However, as load increased, performance divergence became pronounced, validating concerns raised in earlier empirical and analytical works regarding scheduler scalability under contention [8, 9].

The superior response-time performance of EDF and fixed-priority schedulers under medium and high load aligns with the optimality properties of deadline- and priority-driven scheduling in pre-emptive uniprocessor systems [4, 5]. EDF's consistently lower mean response time under high load supports its suitability for time-critical tasks, particularly in embedded and cyber-physical systems where deadline adherence is paramount [12, 13]. Nevertheless, the observed increase in starvation events and reduced fairness under EDF and fixed-priority scheduling highlights a critical trade-off that has been widely discussed but less frequently quantified in lightweight operating system contexts [7, 10]. In contrast, round-robin scheduling maintained higher fairness indices across all load levels, corroborating its reputation for equitable CPU allocation, albeit at the cost of increased context-switch overhead and degraded latency under heavy task arrival rates [3, 8]. The regression analysis further substantiated that context-switch frequency scales with task arrival intensity and is amplified in time-sliced schedulers, reinforcing kernel-level observations regarding scheduling overhead in minimal systems [8, 10].

Importantly, the significant interaction between policy and load suggests that scheduler effectiveness cannot be generalized across deployment scenarios. Instead, the findings support a workload-aware perspective, wherein scheduler selection and tuning must be guided by expected task characteristics, timing constraints, and acceptable trade-offs between responsiveness and fairness [11, 12]. This empirical evidence bridges the gap between abstract scheduling theory and practical lightweight system deployment, offering validation for adaptive or hybrid scheduling strategies increasingly discussed in embedded systems research [5, 11]. Overall, the discussion underscores that lightweight operating systems, despite their simplicity, exhibit complex scheduling dynamics that demand empirically informed design decisions rather than reliance on canonical defaults [1, 9, 15].

Conclusion

This research provides a comprehensive empirical evaluation of process scheduling behavior in lightweight operating systems, demonstrating that scheduler performance is inherently context-dependent and shaped by the interaction between policy design and workload intensity. The results confirm that deadline- and priority-based schedulers offer clear advantages in reducing response time and improving responsiveness under moderate to high system load, making them suitable for time-sensitive embedded and real-time applications. However, these gains are accompanied by increased risks of starvation and reduced fairness, particularly when workload heterogeneity and sustained contention are present. Conversely, round-robin scheduling exhibits strong fairness

and predictability but incurs higher latency and context-switch overhead as load increases, limiting its suitability for high-frequency or deadline-driven workloads. From a practical standpoint, system designers should avoid one-size-fits-all scheduling choices and instead align scheduler selection with application requirements. For lightweight systems supporting mixed-criticality workloads, fixed-priority or EDF scheduling combined with aging mechanisms and priority inheritance can mitigate starvation while preserving responsiveness. In scenarios emphasizing equitable resource sharing, such as multi-sensor or cooperative task environments, carefully tuned round-robin scheduling with optimized time quanta can balance fairness and overhead. Additionally, workload profiling during system design and deployment should inform scheduler parameter tuning, as small adjustments in quantum length or priority aging can significantly influence performance outcomes. The findings also suggest that adaptive or hybrid scheduling approaches, which dynamically adjust policies based on observed load and task behavior, represent a promising direction for future lightweight kernel design. Ultimately, effective scheduling in lightweight operating systems emerges not from selecting a universally optimal policy, but from integrating empirical evidence, workload awareness, and pragmatic tuning strategies to achieve robust, predictable, and efficient system behavior under real-world constraints.

References

1. Tanenbaum AS, Bos H. Modern Operating Systems. 4th ed. Pearson Education; 2015.
2. Labrosse J. MicroC/OS-II: The Real-Time Kernel. 2nd ed. CMP Books; 2002.
3. Silberschatz A, Galvin PB, Gagne G. Operating System Concepts. 9th ed. Wiley; 2013.
4. Liu CL, Layland JW. Scheduling algorithms for multiprogramming in a hard-real-time environment. J ACM. 1973;20(1):46-61.
5. Buttazzo GC. Hard Real-Time Computing Systems. 3rd ed. Springer; 2011.
6. Dunkels A, Grönvall B, Voigt T. Contiki - a lightweight and flexible operating system for tiny networked sensors. In: Proc LCN; 2004. p. 455-462.
7. Sha L, Rajkumar R, Lehoczky J. Priority inheritance protocols: an approach to real-time synchronization. IEEE Trans Comput. 1990;39(9):1175-1185.
8. Love R. Linux Kernel Development. 3rd ed. Addison-Wesley; 2010.
9. Klein M, Ralya T, Pollak B, Obenza R, Harbour M. A Practitioner's Handbook for Real-Time Analysis. Kluwer Academic; 1993.
10. Molnár I. CFS scheduler. Linux Kernel Documentation; 2007.
11. Buttazzo G, Lipari G, Cucinotta T. Elastic scheduling for flexible workload management. IEEE Trans Comput. 2002;51(3):289-302.
12. Burns A, Wellings A. Real-Time Systems and Programming Languages. 4th ed. Addison-Wesley; 2009.
13. Kopetz H. Real-Time Systems: Design Principles for Distributed Embedded Applications. Springer; 2011.
14. Eker J, Janneck J, Lee EA, Liu J, Liu X, Neuendorffer S, *et al*. Taming heterogeneity the Ptolemy approach. Proc IEEE. 2003;91(1):127-144.
15. Marwedel P. Embedded System Design. 2nd ed. Springer; 2011.