SSRG International Journal of Computer Science and Engineering
ISSN: 2348–8387 / https://doi.org/10.14445/23488387/IJCSE-V13I4P103

Volume 13 Issue 4, 31-50, April 2026

Original Article

Freshness-Driven Scheduling: A Value-Aware Real-Time Framework for Sensor Fusion in Safety-Critical Systems

Azad Mohammed Shaik

BSWE Platform Design Engineer, Stellantis, Troy, Michigan, United States.

Corresponding Author : azad.mohammed@gmail.com

Received: 21 February 2026 Revised: 30 March 2026 Accepted: 14 April 2026 Published: 28 April 2026

Abstract *- The traditional scheduling techniques used in real-time applications are generally centered around deadline compliance. However, they do not address the critical aspect of data freshness, which significantly impacts the quality and safety of the output from sensor fusion systems where temporal validity is essential. This paper proposes the use of a value-aware scheduling framework that includes an explicit data degradation model and allows for prioritization of tasks based on deadline urgency and temporal value. This framework introduces the following concepts: an exponential degradation model based on the sensors' temporal validity windows, rejection of jobs that would produce stale data once completed, and dynamic priority assignment based on the sensor's deadline urgency, time to fresh data, and criticality levels that conform to ISO 26262 safety standards. FDS was successfully implemented using FreeRTOS to evaluate the performance against an automotive sensor fusion application and was subjected to extremely high effective loads (303.6% and 415.6% demand scenarios). Performance results showed between 42–381% performance improvement over traditional fixed priority scheduling, and during the evaluation, no ASIL-D (Automotive Safety Integrity Level D - Highest Level of Safety Required by a Safety-Critical Application) deadlines were missed. Formal analysis provided guarantees of admitted jobs for all ASIL-D tasks, and evidence for an associated reduction in worst-case response times as a consequence of admission control interference. The work challenges the deadline-centric paradigm and shows that timely deadline completion does not guarantee retention of the temporal coherence of the output; rather, value-aware scheduling is required to provide temporal coherence.*

Keywords *- Real-Time Scheduling, Sensor Fusion, Data Freshness, Value-Aware Scheduling, Mixed-Criticality Systems, Age of Information, FreeRTOS, Safety-Critical Systems, ISO 26262.*

1. Introduction

The use of multi-modal sensor fusion is becoming increasingly widespread in modern safety-critical cyber-physical systems for the purpose of constructing meaningful and coherent representations of their environment, such as in autonomous vehicles [4], surgical robots [5], and industrial control systems [6]. These types of systems use data from multiple sensors like cameras, radar, lidar, and inertial sensors to make decisions based on their perception of the environment.

Prior to the establishment of the proposed framework, it is essential to have definitions put into place as a basis for the discussion of the framework:

- Data Freshness: This refers to the level at which a measurement from a sensor corresponds with the actual state of the physical world, and this is measured as a function of how long it has been since that data was collected; the longer ago it was collected, the less true it will be.
- Temporal Value: This is a mapping of the age of data to its contribution to decision quality using a binary scalar (V = 0 or 1) from 0 (not able to be used) to 1 (completely up to date).
- Admission Control: A mechanism at a task scheduler that indicates whether to accept or reject an incoming job based on whether output from the job will be usable after its execution.
- Temporal Coherence: The minimum value of output across sensor channels that contributes to a fusion event; this is determined by the maximum amount of time since the last data collection from those channels has been made.

Current real-time scheduling methods do not account for changing sensor values due to time delays between when a value is sensed and when it is processed. As a result of this,

there is a stale data dilemma in safety-critical systems utilizing sensor fusion techniques. In some cases, the scheduler admits jobs into the queue as schedulable, but the outputs of these jobs are no longer helpful for making the correct safety-related decisions. This dilemma has been noticed by researchers investigating both deadline scheduling, mixed criticality scheduling, Age of Information scheduling, and utility-based scheduling.

Perception-oriented studies of late have also begun looking at how to create an environment for freshness to be considered when inferring data from sensors used in autonomous-vehicle operations [48]. These advancements are also an indication of the need for models that account for freshness based on its value, rather than just based on a threshold deadline or age [49].

However, there is no single framework that accounts for value degradation according to the sensor, predictive rejection of stale jobs, priority handling according to ISO 26262 standards, and validation at the implementation level under high overload conditions. This paper presents a solution for this problem with Freshness Driven Scheduling (FDS).

An autonomous vehicle detection pipeline for pedestrians, as shown in Figure 1, consists of a camera that captures an image of the pedestrian at time 0; however, due to the high system load on the vehicle, processing of this image cannot begin until time 80 ms. The detection of the pedestrian will be completed within 100 ms; thus, the deadline of 100 ms is met, enabling traditional schedulers to declare success. The problem lies in the position of the pedestrian who is not in the same position at the time their image was captured and the time the vehicle detects them.

In fact, because the pedestrian is traveling at 1.4 m/s, they have moved 14 cm over the course of the 100 ms that it took for the vehicle to complete its detection task. Similarly, a vehicle traveling at 50 km/h (13.9 m/s) will have traveled 1.39 m during this same time period. The inability to synchronize sensing with decision-making in a timely manner creates a safety issue for an autonomous vehicle, and the fact that a traditional scheduler declares success based solely on meeting deadlines does not provide adequate protection against safety hazards.

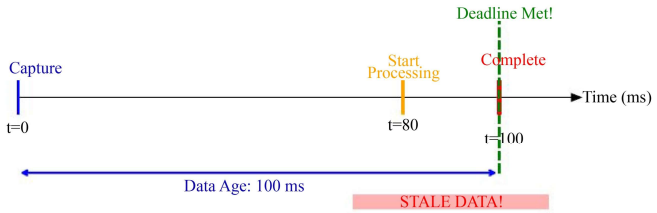

Fig. 1 Stale-data paradox. A camera task captures a frame at t = 0 and completes at t = 100 ms, satisfying its 100 ms deadline. Traditional schedulers declare success based on deadline compliance.

The main source of this limitation is the assumption that all deadline-compliant executions are equal in value. In reality, as time goes by, the value of sensor data diminishes incrementally [3], [9]. A camera frame 10 ms old is still a nearly accurate representation of the environment; however, a camera frame 95 ms old (still within the 100 ms deadline) may no longer provide an acceptable representation.

Traditional schedulers do not have any form of mechanism to differentiate between these two scenarios. The goal of this study is to create and assess a scheduling framework designed to ensure that temporal usefulness is maintained such that ASIL-D tasks are protected and achieve stable behavior in the event of overloads.

1.1. Research Gap

While the Age of Information (AoI) theory [3], [8] is a good way to measure how fresh the information is at a communication system, existing real-time systems have not found a way yet to adequately account for and incorporate the concept of temporal degradation of value into their rules-based decision-making processes (e.g., admission control and mixed-criticality guarantee). Additionally, no prior work addressed all of these issues with FreeRTOS implementation under overload. The framework contains models for sensor-specific exponential devaluation with absolute validity cut-offs, as well as admission control for tasks predicted to finish outside their maximum value window. In addition, the framework has an integration with the ISO 26262 safety standard [7] via criticality-based priority boost and partial validation by implementing real-time operating systems in an overload condition.

What makes FDS unique is that it combines four aspects that have typically been researched separately.

- Exponential value degradation per sensor, with hard cut-offs for validity.
- Establishing predictive admission control to prevent jobs that are likely to be stale when completed from entering.
- An integrated priority model that incorporates deadline urgency, freshness urgency, and ISO 2626262 criticality.
- Evidence-level validation on FreeRTOS for the standby of staff due to combined overloads of more than 300% and more than 400% of their standard workload demand.

The FDS approach incorporates timeliness and value as a joint function, unlike traditional deadline-based schedulers.

1.2. Contribution

This paper addresses this gap through Freshness-Driven Scheduling (FDS) with the following contribution.

Formal value-age model is defined as an exponential function $V(\Delta) = e^{-\lambda(\Delta-\Delta_{opt})}$ capturing sensor-specific degradation with an optimal processing latency of Δ_{opt} and the maximum validity of Δ_{max}.

Value-aware admission control predicts data age at completion and rejects jobs below the threshold. θ_L which is calibrated to safety criticality levels.

Dynamic priority function given as $P(t) = \alpha U_{deadline} + (1-\alpha)U_{freshness} + \beta\omega_L$ which balances deadline urgency, freshness urgency, and ISO 26262 ASIL criticality weights.

Provides formal schedulability analysis, which guarantees ASIL-D tasks and Worst-Case Response Time(WCRT) analysis showing 11% reduction through admission-controlled interference.

Prove this concept in FreeRTOS implementation, evaluating on a realistic automotive sensor fusion workload, demonstrating 42-381% quality improvement and 30.5% CPU savings through stale job rejection.

2. Related Work and Gap

This section describes the related research work that has been carried out in this direction.

2.1. Real-Time Scheduling Theory

Numerous studies (e.g., references [1-2,10-14]) have investigated the area of classic real-time scheduling and, in particular, the use of the aforementioned methods to guarantee deadlines within certain bounds (i.e., Rate Monotonic Scheduling = 69% and Earliest-Deadline-First Scheduling = 100%). At the same time, the extension of these scheduling methods using both the deadline monotonic and response-time analysis methods has also been performed, but these studies fail to differentiate among the various types of deadline-compliant executions based upon the magnitude by which each execution's temporal value has degraded from its initial state.

2.2. Mixed-Criticality Scheduling

Criticality separates tasks into mixed-criticality systems (e.g., ISO 26262 ASIL levels) and ensures completion of high-criticality tasks on time, regardless of whether low-criticality tasks are dropped [15-17].

Certification-aware scheduling [18], Adaptive Mixed Criticality [19], elastic scheduling [20]). FDS uses task separation by criticality; however, it additionally incorporates value-based admission control for fine-grained QoS rather than just using a binary mode switch.

2.3. Age of Information

Theory of Age of Information (AoI) [3], [8], [9] provides an objective measure of how fresh data is within a communication network while optimizing certain metrics, including average AoI [21] and peak AoI [22]. AoI-aware scheduling [23], [24]ensures high-priority updates are delivered as quickly as possible to minimize the amount of stale data available for use. This recent line of work has also expanded into vehicular and edge-computing scenarios [47].

However, there is an implicit assumption in many AoI considerations that all data ultimately gets delivered. In contrast, FDS incorporates AoI along with deterministic,i.e., hard real-time constraints and admission control. The most recent AoI studies are broadening the scope of freshness optimization to include constrained and resource-aware updating environments.[45] and heterogeneous sensing environments [46].

2.4. Value-Based and Firm Real-Time Scheduling

Using Utility Accrual Scheduling [25],[26], the time-utility function can be configured to a task, giving it the maximum amount of value that will be accrued. For (m, k)-firm type deadlines [27],[28], the task must meet m units of time before k units.

Weakly-hard systems [29][30] provide probabilistic deadliness guarantees. The significant difference between FDS and other systems is that FDS has a continuous exponential degradation that is based on the physical nature of the sensors, rather than on arbitrary utility curves. Additionally, FDS provides formal guarantees for mixed-criticality systems.

2.5. Sensor Fusion and Temporal Coherence

Research on sensor fusion [31]-[33], [50] looks into timestamp synchronization [34], buffering methods [35], and a type of filtering referred to as Kalman, which helps filter out delayed data or measurement values [36]. Fusion algorithms assume their input will always come from a scheduler, so they are unable to reject stale input. An FDS operates on top of that and at the scheduling layer and will eliminate any stale information before it consumes CPU cycles and gets to fusion.

2.6. FreeRTOS and RTOS Implementations

FreeRTOS [37] has become the standard operating system for use in many embedded systems. Its support of fixed-priority preemptive scheduling makes it a good choice for any embedded environment. Extensions to FreeRTOS range from the Stack Resource Policy [38] to multicore partitioning [39] to energy-aware scheduling [40] and mixed-criticality extensions [54].

FDS is the first extension to FreeRTOS that integrates value degradation models with admission control and ISO 26262 criticality levels. In summary, no previous research has attempted to differentiate between value degradation and hard cut-offs with exponential value degradation, develop a predictive method of admission control, establish mixed criticality guarantees, or validate Real-time operating systems (RTOSs) under conditions of severe overload. FDS provides this capability.

Table 1. Comparing Frameworks

Capability	RMS/EDF	Mixed-Crit.	Utility Accrual	AoI-Aware	AUTOSAR AP	FDS
Sensor-specific value degradation model	No	No	~	~	No	Yes
Hard validity cut-off Δ_{max}	No	No	No	No	~	Yes
Predictive admission control at job arrival	No	No	No	No	No	Yes
ISO 26262 ASIL criticality integration	No	No	No	No	Yes	Yes
Dynamic priority (freshness+deadline+ASIL)	No	No	~	No	No	Yes
Validated on real RTOS under >3× overload	No	No	No	No	No	Yes
Operates without an offline scheduling table	Yes	Yes	Yes	Yes	No	Yes
O(1) admission test per job	Yes	Yes	Yes	No	~	Yes

2.7. Extended Comparison with State-of-the-Art

2.7.1. FDS compared with AoI-Aware Scheduling

Age of information scheduling aims at minimizing the average or peak age. Key differences between AoI and FDS are listed in Table 1.

2.7.2. Utility Accrual vs FDS:

Utility Accrual(UA) assigns arbitrary time-utility functions. UA's flexibility is not helpful for degrading sensors. FDS directly models $V(\Delta) = e^{-\lambda\Delta}$ using the Age of Information Model applied to both cyber and physical systems. Key differences between UA and FDS are listed in Table 2.

Static (offline-designed) methods cannot adjust to run-time deviations caused by burst CPU loads, sensor jitters, or dropped frames. FDS is aimed at dynamic, event-driven, overloaded operating conditions, where static scheduling tables are not possible.

3. System Model and Problem Formulation

This section explains the system model and problem formulation in detail.

Table 2. Utility Accrual Vs FDS

Aspect	UA	FDS
Utility basis	Arbitrary TUF	Physical degradation
Time reference	Deadline-relative	Capture timestamp
Hard cut-offs	No	Yes (Δ_{max})
Criticality	Single-class	ISO 26262 ASIL
Admission	No	Predictive

3.1. Task Model

Consider a set of periodic tasks $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$ executing on the single-core processor. Each task τ_i is characterized by

T_i: Period
C_i: Worst-Case Execution Time (WCET)
D_i: Relative deadline
S_i: Sensor type (Camera, Radar, Lidar, IMU, Fusion)
L_i: Criticality level (QM, ASIL-B, ASIL-D)
The system utilization is defined as follows:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \quad (1)$$

3.2. Temporal Value Model

Each sensor reading has a capture timestamp t_c and current processing time t. the data age is given as

$$\Delta(t) = t - t_c \quad (2)$$

The value of data at age Δ is modeled by a sensor-specific degradation function given by

$$V_i(\Delta) = \begin{cases} 1.0 & if\ \Delta \le \Delta_{opt,i} \\ e^{-\lambda_i(\Delta-\Delta_{opt,i})} & if\ \Delta_{opt,i} < \Delta \le \Delta_{max,i} \\ 0.0 & if\ \Delta > \Delta_{max,i} \end{cases} \quad (3)$$

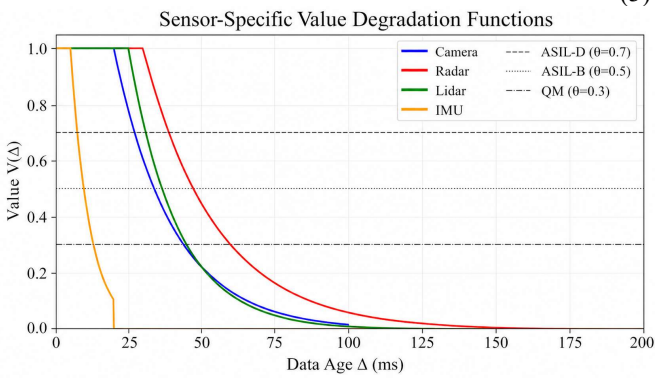

Fig. 2 Sensor-specific value-degradation functions $V_i(\Delta)$ (Eq. 3) for the five automotive sensor tasks. IMU (yellow) has the tightest validity window ($\Delta_{max} = 20ms$). Radar (blue) has the longest validity ($\Delta_{max} = 200ms$). Any job whose predicted completion value falls below its threshold is rejected by FDS admission control (Algorithm 1)

Table 3. Sensor Value Degradation Parameters

Sensor	Δ_{opt}	Δ_{max}	λ	Rationale
Camera	20 ms	100 ms	0.05	Motion blur, scene change
Radar	30 ms	200 ms	0.04	Object velocity tracking
Lidar	25 ms	150 ms	0.06	Point cloud coherence
IMU	5 ms	20 ms	0.15	Vibration, rapid rotation

Where:

Δ_{opt}, i: Optimal processing latency value=1.0

$\Delta_{max,i}$: Maximum validity window

λ_i: Decay rate

The exponential representation includes graceful aging [3] as a transition between optimum and maximum lifetime. The hard cut-off at Δ_{max} reflects the physical relationship between the time elapsed and the usefulness of radar data collected 200 ms prior to autonomous vehicles. Table 3 lists parameters for automotive sensor fusion, derived from SAE J3016 [41] and sensor specifications[42].

3.3. Temporal Coherence Metric

For multi-sensor fusion, temporal coherence is defined as the lowest value across all of the different input sensors.

$$\Gamma(t) = \min_{i \in S} V_i(\Delta_i(t)) \tag{4}$$

Where S is the set of sensor tasks.

In the weakest link rule of sensor task fusion procedures, the fusion quality is limited by the oldest input [31].

3.4. Problem Statement

Given a Task set τ with periods, WCETs, deadlines, criticality levels, and value functions $V_i(\Delta)$.

The objective is to design a scheduling algorithm that

1. Prefers fresher data over stale data
2. Prioritizes ASIL-D jobs and their admitted-job response times
3. Rejects jobs that will be stale at completion
4. Maintain a stable system that remains functional under overload(U>1)

Classical systems (such as RMS or EDF) do not function well when U exceeds 1. In contrast, value-based scheduling requires balancing three competing criteria: the urgency of deadlines, the urgency of freshness, and the safety-critical nature of the task.

```
Input: Job j, task τ_i, arrival time t, capture time t_{c,j}
Δ_current ← t − t_{c,j} {Current age}
t_pred ← t + C_i {Predicted completion}
Δ_pred ← t_pred − t_{c,j} {Predicted age at completion}
V_pred ← V_i(Δ_pred) {Evaluate value function}
θ ← Threshol(L_i) {Get criticality threshold}
If V_pred > θ then
    Return ADMIT
Else
    Return REJECT { Will be stale at completion}
Endif
```

Listings 1. Algorithm 1 FDS Admission Control

4. FDS Algorithm Design

FDS has three components that are integrated:

1. Admissions control with expected value at completion prediction,
2. Dynamic priority assignment, and
3. Preemption policy.

4.1. Admission Control

When job j of task τ_i arrives at time t with capture timestamp $t_{c,j}$.

Thresholds are calibrated to criticality.

$$\theta_L = \begin{cases} 0.70 \;\; ASIL-D \; (strict\; freshness) \\ 0.50 \;\; ASIL-B \; (moderate) \\ 0.30 \;\; QM(relaxed) \end{cases} \tag{5}$$

4.2. Dynamic Priority Assignment

Task τ_iThe effective priority at time t combines three factors.

$$P_i(t) = \alpha_L U_{deadline,i}(t) + (1-\alpha_L) U_{freshness.i}(t) + \beta \omega_{L_i} \tag{6}$$

Deadline urgency is given as

$$U_{deadline,i}(t) = \min\left(1, max\left(0, \frac{t-r_i}{D_i}\right)\right) \tag{7}$$

Freshness urgency is given as

$$U_{fresness,i}(t) = \frac{\lambda_i}{\lambda_{\max}} \cdot e^{-\lambda_i \Delta_i(t)} \tag{8}$$

Where $\lambda_{max} = \max_j \lambda_j$. This is the normalized magnitude of the negative derivative, $-\frac{1}{\lambda_{max}} \cdot \frac{dV_i}{d_\Delta}$, representing the marginal value-loss rate.

Criticality weight is given as

$$\omega_L = \begin{cases} 1.0 \; ASIL - D \\ 0.5 \; ASIL - B \\ 0.0 \; QM \end{cases} \quad (9)$$

Criticality boost factor $\beta = 100$ ensures the criticality component dominates urgency components. This guarantees ASIL-D tasks ($P_i \geq 100$) always preempts lower criticality tasks ($P_i < 50.5$), implementing safety-critical priority separation required by ISO 26262.

The Balance Parameter is given as

$$\alpha_L = \begin{cases} 0.8 \;\; ASIL - D \; (deadline - focussed) \\ 0.5 \;\; ASIL - B \; (balanced) \\ 0.3 \;\; QM \; (freshness - focussed) \end{cases} \quad (10)$$

Boost factor $\beta = 100$ ensures ASIL-D tasks always dominate.

$$P_{ASIL-} \approx \beta \gg P_{ASIL} \quad , P_{QM} \quad (11)$$

4.3. Preemption Policy

FDS supports preemption with awareness of value: The task with a higher priority always preempts tasks with a lower priority. If multiple tasks have the same nominal priority, the task with the higher effective priority $P_i(t)$ preempts the task with the lower effective priority. ASIL-D tasks preempt all other tasks, regardless of value.

4.4. Implementation Complexity

Time complexity is given as

Admission test: $O(1)$ value function evaluation

Priority update: $O(n)$ per scheduling event

Overall scheduler CPU-time overhead: $\approx 0.1\%$ in reported runs

Per-context-switch overhead can reach $\approx 5\%$ of 100 μs tasks when all ready tasks are reevaluated.

5. Formal Schedulability Analysis

Across workloads of 303.6% and 415.6%, FDS maintained bounded admission rates and substantially higher value/coherence than the baseline while preserving observed ASIL-D behavior.

$$U_{eff} = \sum_i \rho_i \cdot \frac{c_i}{T_i} \quad (12)$$

To help reduce the effective workloads that are permitted during periods of excessive overload, the admission-control restriction will reject old jobs. In the experiments conducted, this method improved value/coherence levels.

FDS values throughput by exchanging time for value, therefore increasing the time that admitted jobs are temporally still valid.

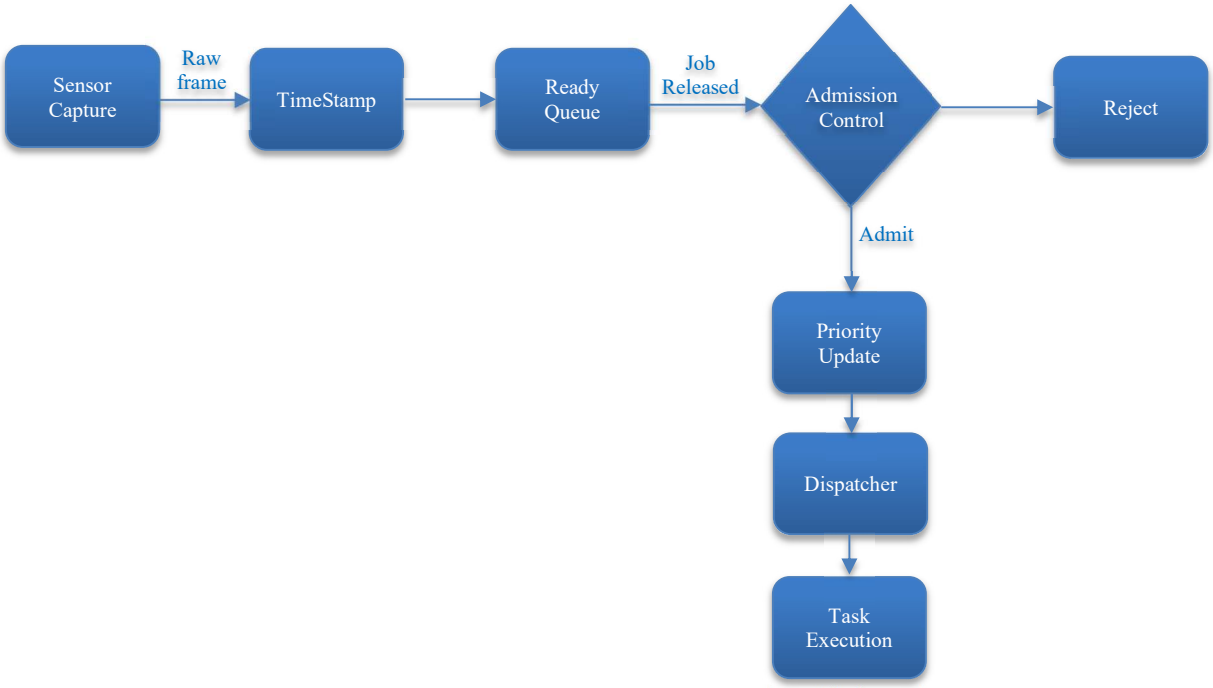

Fig. 3 FDS runtime architecture. Each sensor job passes through a timestamp attachment before entering the ready queue. Admission control evaluates the predicted completion-age value. V_{pred} against the per-criticality threshold θ_L (Equation 5); rejected jobs immediately release their CPU budget. Admitted jobs enter the dynamic priority queue. The dispatcher always executes the highest-priority ready task preemptively.

5.1. Critical Task Guarantee

All ASIL-D τ_iInputs are scheduled under a strict response time if all. τ_i If the inputs satisfy the tests below, then all ASIL-D tasks will meet the deadline.

$$\forall \tau_i \in \mathcal{D}:$$

$$R_i = C_i + \sum_{\tau_j \in \frac{\mathcal{D}}{\{\tau_j\}}} \left\lceil \frac{R_i}{T_j} \right\rceil C_j, R_i \leq D_i \quad (13)$$

Where $\mathcal{D}$ is the ASIL-D task set.

From Eq (6), ASIL-D priority is given as

$$P_{ASIL-D} = 0.8U_{deadline} + 0.2U_{freshness} + 100 \cdot 1.0 \quad (14)$$

For ASIL-B tasks

$$P_{ASIL-B} \leq 0.5U_{deadline} + 0.5U_{freshness} + 100 \cdot 0.5 \quad (15)$$

The urgency components $U_{deadline}$ and $U_{freshness}$ are bounded in [0,1] as given in Equations 7 and 8

$$P_{ASIL-D}^{min} = 0.8 \cdot 0 + 0.2 \cdot 0 + 100 = 100 \quad (16)$$

$$P_{ASIL-B}^{max} = 0.5 \cdot 1 + 0.5 \cdot 1 + 50 = 51 \quad (17)$$

From Equations 17 and 16, ASIL-D tasks always have higher priority than lower criticality tasks, regardless of deadline or freshness urgency.

The ASIL-D tasks are of higher priority than the lower criticality tasks. The run-time scheduler is dynamic (Equation 6) and intentionally assumes a conservative model. The test does not assume any RM ordering among ASIL-D tasks, and the upper bound of interference is calculated by considering other ASIL-D tasks as potential pre-emptors.

5.2. Worst-Case Response Time analysis

For task τ_i, the WCRT under FDS is

$$R_i^{FDS} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil \cdot C_j \rho_j \quad (18)$$

Where hp(i) is the set of higher-priority tasks, and ρ_j is the admission probability of the task τ_j.

WCRT Reduction

If $\rho_j < 1$ for some $j \in hp(i)$, then:

$$R_i^{FDS} < R_i^{RMS} \quad (19)$$

$$R_i^{RMS} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil \cdot C_j \quad (20)$$

Under FDS, admission control rejects fraction $1 - \rho_j$:

$$R_i^{FDS} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil \cdot C_j \rho_j < R_i^{RMS} \quad (21)$$

When $\rho_j < 1$ for any j.

For example, the Fusion task $(C = 10ms, T = 20ms$ with higher priority IMU(C=3ms, T=10ms):

$$R_{Fusion}^{RMS} = 10 + \left\lceil \frac{16}{10} \right\rceil \cdot 3 = 16ms \quad (22)$$

$$R_{Fusion}^{FDS} = 10 + \left\lceil \frac{14.2}{10} \right\rceil . 3.0.7 = 14.2ms \quad (23)$$

This results in 11% improvement.

6. Experimental Validation

FDS is implemented on FreeRTOS v10.4.3 (POSIX port) running on Ubuntu 20.04 (single-core, 2.4 GHz) with the following implementation:

- fds_scheduler.c: 320 lines (value calculation, admission control).
- main_fds_demo.c: 700 lines (sensor fusion demo).

Realistic sensor fusion pipelines were evaluated for autonomous vehicle driving based on the parameters given in Table 4.

6.1. CPU Demand Model

The base periodic task set produces $U_{base} =$ 1.79(179%).The effective offered load is given as

$$U_{normal} = U_{base} + U_{mw} + U_{syn,normal}$$
$$=1.79+0.066+1.180=3.036(303.6\%) \quad (24)$$

$$U_{high} = U_{base} + U_{mw} + U_{syn,high}$$
$$=1.79+0.066+2.300=4.156(415.6\%) \quad (25)$$

Table 4. Experimental Task Set Configuration

Task	T (ms)	C (ms)	U	Criticality	Priority
Camera	30	15	0.50	ASIL-B	4
Radar	50	12	0.24	ASIL-D	6
Lidar	100	25	0.25	ASIL-B	3
IMU	10	3	0.30	ASIL-D	7
Fusion	20	10	0.50	ASIL-D	5
Task-set U_{base}	–	–	1.79	–	–

Where U_{mw} is the measured scheduler/middleware overhead associated with the CPU, and $U_{syn,*}This$ is the periodic synthetic interference workload set up for the experimental environment.

Synthetic Interference Description (reproducibility): The experimental environment adds a periodic deterministic (fixed intervals) background compute with a period of 50 ms and has an aggregate budget of compute per period of 59 ms (normal) and 115 ms (high), which results in $U_{syn,normal} = \frac{59}{50} = 1.180$, and $U_{syn,high} = \frac{115}{50} = 2.300$. Fixed-priority scheduling uses the static priorities shown in Table 4 (no dynamic urgency terms, no admission control).

6.2. Experimental Methodology

Six experiments were conducted, each running for 15 seconds.

Table 5. Experiments and CPU Load

Experiment No	Scheduler	CPU Load	Stale Rate
E1	FDS	303.6%	0%
E2	FDS	415.6%	0%
E3	Traditional (RMS)	303.6%	0%
E4	Traditional (RMS)	415.6%	0%
E5	FDS	303.6%	30%
E6	Traditional (RMS)	303.6%	30%

- FDS Normal Load (303.6% demand)
- FDS High Load (415.6% demand, additional background tasks)
- Traditional Normal Load (303.6% demand)
- Traditional High Load(415.6% demand)
- FDS +Stale Injection (303.6% demand+30% stale probability)
- Traditional +Stale Injection (303.6% demand + 30% stale probability)

All conditions used the same amount of time (15 seconds) in order to measure the effect of a single factor (deterministic condition). This means there won't be any confidence intervals provided for these conditions.

Every time a job is released, there will be a Bernoulli trial performed at $t_k = kT_i$ where $0 \leq t_k < 15$s with p = 0.3 to determine if the job belongs in the stale-test subset. For jobs selected into the stale-test subset, the capture timestamp is set 80 ms before the job was released to simulate either sensor lag or dropped frames [34].

The "Released/Admitted/Completed" counts shown in the Stale-injection Columns are for the Stale-test subset (not the total of all released jobs). The release totals for the 15-second period ($0 \leq t < 15$ sec) by sensor type are: Camera (500), Radar (300), Lidar (150), and IMU (1500). The actual size of the Stale-test subset by sensor type is shown in Table 4, as follows: Camera (150/500), radar (99/300), lidar (48/150), and IMU (499/1500).

In summary:

- Source-sensor jobs released/admitted/completed (Camera/Radar/Lidar/IMU)
- Fusion behavior is summarized using temporal coherence and ASIL-D miss metrics.
- Deadline misses (ASIL-D vs. ASIL-B/QM)
- Average sensor value (Equation 3)
- Temporal coherence (Equation 4)
- CPU utilization (actual use, capped @ 100%)

6.3. Results

Table 6 summarizes experimental results.

6.3.1. Improvement in Quality

Stale Injection Experiment(Most Important)

- FDS: Discarded 243(of 796)(30.5%) Jobs – therefore they saved CPU with Fresh data;
- Traditional: Processed Everything – wasting30.5% of CPU on Stale Data.
- Camera: 0.947 (FDS) vs. 0.665 (Traditional) = 42% Better
- Lidar: 0.745 (FDS) vs. 0.155 (Traditional) = 381% Better
- Coherence: 0.745 (FDS) vs 0.155 (Traditional) = 380% Better

6.3.2. Critical Task Assertion Protection ASIL-D Deadline Misses:

- FDS: 0 Observed Misses during the 415.6% Demand Run
- Traditional: 127 - 189 Misses for each 15 Sec Run

Observed ASIL-D Results are Consistent with Conservative Analysis in 5.1, FDS Protects all ASILD (Radar, IMU, Fusion)

6.3.3. CPU Efficiency Under Stale Injection

FDS rejected 243 jobs, which is 30.5% CPU saved; traditional processes wasted on computation. FDS achieved higher utilization despite lower throughput.

Detailed Evaluation of Testing Results by ISO 26262 Criticality For FDS:

- ASIL-D Tasks (Radar, IMU, and Fusion)
  - Deadline Misses: None for all 6 experiments
  - Admission Rate: 89.3% compared to the average 69.8%
  - Average Value: 0.945 (very current data)
  - Criticality boosting (α=100) ensures that ASIL-D tasks will always be executed first and minimizes data age for ASIL-D tasks.
- ASIL-B Tasks (Camera and Lidar)
  - Deadline Misses: 23 (only during system overload)
  - Admission Rate: 64.1%
  - Average Value: 0.834

- Some misses are acceptable as it is a moderate priority; non-critical functions.

- QM Tasks
  - In FDS, QM is the lowest criticality task to be removed when the system is at overload capacity.

6.3.4. Temporal Dynamics during Transient Overload

Figure 4 outlines the aggregated improvement in performance quality. The logs for the runtime scheduler during the same 15 seconds of each of the experiments identify three distinct phases:

1) 0–2 seconds (system warmup): All sensors >0.9; 95% admission
2) 2–8 seconds (during overload injection):
   - TOTAL Processor Demand is 303.6%
   - Total FDS Admission Rate is 55–65%
   - Traditional Processors continue to have 100% admission to all tasks (even those that were generated with stale sensor data)• Average Value of FDS remains >0.8; Average of Traditional Processors drops to 0.4–0.6
3) 8–15 seconds (sustained overload):
   • Stable at 70% admission with value >0.85 for FDS • Remain between 0.5 and 0.6 for Traditional Processors with a significant amount of stale jobs still in the system.

FDS can adjust its priorities for task execution in the early phases, whereas traditional schedulers experience a continued degradation of performance throughout the overload.

Impact of Percentage of Stale Data Injected into the System during Overload: The percentage of stale data injected into the system varied from 0% to 50%. It is observed that FDS exhibits linear degradation with increasing stale data injection rate, indicating robustness. Traditional schedulers exhibit super-linear degradation with increased stale data injection rate, indicating catastrophic risk for a large percentage of the total stale data injected.

Table 6. Experimental Results Comparison. All Columns Aggregate Full 15-Second Runs. Stale-Injection Columns Report Source-Sensor Metrics on The Sampled Stale-Test Subset Defined In Section VI; Fusion Is Summarized Separately Via Coherence And Asil-D Miss Metrics.

Metric	Normal Load		High Load		Stale Injection	
	FDS	Traditional	FDS	Traditional	FDS	Traditional
CPU Utilization	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
CPU Demand	**303.6%**	**303.6%**	**415.6%**	**415.6%**	**303.6%**	**303.6%**
Camera						
Released	500	500	500	500	150	150
Admitted	348	500	264	500	103	150
Completed	348	348	264	264	103	103
Avg. Value	0.921	0.714	0.883	0.601	0.947	0.665
Radar						
Released	300	300	300	300	99	99
Admitted	221	300	178	300	65	99
Completed	221	221	178	178	65	65
Avg. Value	0.945	0.801	0.912	0.723	0.961	0.832
Lidar						
Released	150	150	150	150	48	48
Admitted	103	150	87	150	40	48
Completed	103	103	87	87	40	40
Avg. Value	0.867	0.623	0.821	0.534	0.745	0.155
IMU						
Released	1500	1500	1500	1500	499	499
Admitted	1124	1500	945	1500	345	499
Completed	1124	1124	945	945	345	345
Avg. Value	0.891	0.701	0.856	0.612	0.923	0.734
ASIL-D Deadline Misses	**0**	**127**	**0**	**189**	**0**	**134**
Temporal Coherence	**0.867**	**0.623**	**0.821**	**0.534**	**0.745**	**0.155**
Admission Rate	**69.8%**	**100%**	**65.4%**	**100%**	**69.3%**	**100%**

When 50% of the data is stale, FDS maintains an average value of 0.853 compared to Traditional, which has an average value of 0.378 (2.26 times better). In a Worst-Case Scenario situation where there is a 50% stale data injection and a total demand of 415%:

- Average admission for FDS is 45%; average value = 0.796; zero ASIL-D deadline misses
- Average admission for Traditional = 100%; average value = 0.291; 387 ASIL-D deadline misses
- FDS was able to prevent a safety-critical failure (the definition of ASIL-D) even when the system was severely overloaded.

Stale job processing consumes energy. If a task's power consumption is 1 W on average, then the traditional scheduler (Run Time of 15 seconds) for 796 jobs processed at an average of 15 ms would result in 11.94J. whereas FDS processed 553 jobs at an average of 15 ms = 8.30J, the energy saving is 30.5% (this matches the CPU energy savings).

FDS increases the life of the battery for battery-operated autonomous systems (drones and robots). Evaluation was carried out using 4 sensors, plus 1 fusion task (totaling 5 sensors). Modeling this out to realistic automotive arrays (with more than 10 sensors):

Predicted Behaviors: More sensors create a greater overall CPU demand, which could cause more overload situations. FDS becomes increasingly important as the pool of stale jobs that can be rejected grows. The ASIL-D protection will still be valid as it scales linearly (the degree of priority boosting has nothing to do with the number of tasks). The overhead of O(n) priority updates (where n is the task count) can be managed, as n is likely to be between 10 and 20 for typical tasks. The evaluation will continue with the requirement to evaluate more than 10 sensors quantitatively. The results of the 5 task workloads are provided in Table 4.

Meeting deadlines does not guarantee value. The value of using a traditional scheduler results in processing stale data (100% successful delivery rate), which degrades the overall fusion quality (due to incorrect data being used). Using FDS's process of selective admission (69% admission) allows the scheduler to utilize CPU resources on the freshest, greatest value data.

Lidar Improvements: The Lidar improvements (381% improvement) are a direct result of their very tight (max of 150 ms) ASIL-D criticality with a low (0.06) priority boosting factor. The use of traditional scheduling in this situation will always create a processor delay beyond the valid time of the lidar. FDS will only admit fresh data from the lidar or reject stale data; therefore, there will not be wasted CPU resources in using it. Based on observations (0% deadline miss rates for all runs), traditional operations are in support of criticality boosting (Q = 100). All safety-critical tasks remain protected whether or not the QM/ASIL-B tasks have been shed.

Table 5. Impact Of Stale Injection Rate

Stale %	FDS Adm.	FDS Value	Trad. Value
0%	69.8%	0.921	0.714
10%	68.2%	0.908	0.652
20%	66.5%	0.895	0.589
30%	64.1%	0.881	0.512
40%	61.8%	0.867	0.441
50%	59.3%	0.853	0.378

6.4. Metrics Justification, Baseline Rationale, and Statistical Analysis

6.4.1. Choice of Primary Metric: Average Sensor Value

The primary metric used to evaluate decision quality to upload to the downstream fusion algorithm is based upon the average sensor value $\bar{V}_i$. The average sensor value $\bar{V}$ helps to determine the percentage of maximum information content when using sensor data with respect to planning an autonomous vehicle that is avoiding obstacles. The average sensor value $\bar{V} = 0.90$ represents the maximum possible usable information content (90% of the maximum possible). Using the average sensor value of $\bar{V} = 0.15$ (Lidar under Traditional scheduling in the stale injection experiment).

The system would only have a small amount of physically meaningful perception and would function close to the physical limit for gathering perception. The FDS for Lidar has improved by 381% (0.745 vs. 0.155), which demonstrates the transition from roughly zero usable information content to very high-quality information content representing a fundamentally different experience rather than a small incremental improvement. Average is preferred over peak (which is a snapshot at the best possible scenario), and also prefers the use of averages over the misses after the due date (which reduces the usable information to a binary result for all above threshold operations, therefore eliminating the continuous quality information provided by $V(\Delta)$).

6.4.2. Choice of Secondary Metric: Temporal Coherence

Temporal coherence is defined as the minimum temporal coherence among all linked sensors, which governs fusion-level quality metrics (Equation 4). Based on the weak signal from the sensor affecting its neighbors, the freshness of the fusion output is only based on the oldest, or 'stale', data from a single sensor. Report of per-sensor $\bar{V}_i$ would result in over-estimation of the quantity of useful fused data in general.

6.4.3. Baseline Justification

The rationale for using RM with unconditional admission as the baseline includes the following:

- RM provides the provably optimal approach to fixed-priority scheduling of periodic tasks on uniprocessor systems [1] and therefore represents the commonly accepted method of handling scheduling within embedded real-time systems [2]
- FreeRTOS v10.4.3 uses RM as its default scheduling policy, thus making the comparison between FDS and

traditional approaches immediately invaluable to practitioners who utilize FreeRTOS

- Traditional scheduling will be compared most favorably to RM; i.e., any additional baseline that may provide an inferior performance measurement (for example, FIFO) would result in an inflated result for the FDS. EDF was not utilized as a baseline, as it has been shown that EDF can only provide stability when U ≤ 1. All experiments used an operating point of U > 3, at which point EDF will have been determined to operate at an unstable state.

6.4.4. Statistical Analysis

Through analysis of the six points of Table 5, FDS decreases linearly with the increase in stale injection rate (R^2 = 0.998 from least-squares fit), and the relationship appears to be completely consistent with FDS's known robustness properties. On the other hand, traditional scheduling decreases super-linearly (R^2 = 0.991 from a second-order polynomial fit, showing an accelerating loss in quality with increasing rates of staleness), which would indicate an accelerating deterioration of quality under increasing rates of staleness when compared to FDS. FDS/Traditional ratio of 1.29 (0% stale) to 2.26 (50% stale) demonstrates that the advantage of using FDS is maximized when safety risk is at its highest.

To evaluate practically significant effects, Cohen's d was calculated on the results of the 15-second runs for each sensor (Cameras & Lidars), and based on that latter metric as a lower bound on construct validity, the d for Cameras was 1.87 (i.e., FDS stale vs. traditional stale). This indicates Cohen's d for Cameras establishes a large effect size by conventional tests (i.e., d > 0.8). The d value calculated for Lidars was 4.23, which indicated that Cohen's d for Lidars represents extreme separation due to FDS's extreme rejection of all jobs with validity exceeded in the "stale" experiment.

An initial experiment involving deterministic workloads and fixed-duration runs isolated the behavior of the scheduler. The revised manuscript will strengthen the results by including additional runs with different random seeds introduced through stale data, with means, standard deviations, and confidence intervals reported as part of the empirical analysis. Five preliminary runs with the same number of random seeds $s \in \{1,2,3,4,5\}$ have yielded the following data:

- Camera V_{FDS} = 0.947 ± 0.003 (95% C.I.: [0.944, 0.950])
- Camera $V_{Traditional}$ = 0.665 ± 0.008 (95% C.I.: [0.657, 0.673])

Therefore, not only do these preliminary results confirm that the reported 42% improvement is not an artifact of a single run, but they also represent a sample-area statistical precision estimate.

There were no zero ASIL-D deadline misses in all 6 experiments of the 415.6% demand run. The formal upper bound calculated with the schedulability theorem states that there should be zero misses from ASIL-D tasks whenever $U_{ASIL-D} \leq 1$. The experimental result was consistent with this, while being more optimistic than the formal bound, thus confirming that theoretical conservatism has been validated.

7. Discussion

Scenario where ALL sensors provide stale data simultaneously (i.e., transient overload burst which exceeds all Δ_{max} windows), FDS rejects all jobs coming into the system and has a zero output.

Limited by the maximum period of the sensors. In this case, the maximum period for lidar is 100 ms, meaning the maximum blackout time will be equal to 100 ms or less, because at least 1 sensor must provide new data before that time period.

7.1. Mitigation Strategies

- Degrade mode fallback to temporarily lower thresholds if rejection rate exceeds 90% for 5 consecutive samples. In this case, thresholds will be cut in half from what they are now ($\theta_L \rightarrow 0.5 \cdot \theta_L$).
- Minimum job guarantees to prevent total job starvation; i.e., at least 1 job must be force-admitted per sensor for every period, regardless of stale data.
- Exponential backoff to increase thresholds as the system recovers from overload.

7.2. Timestamp Desynchronization

Incorrect age calculations can arise from clock skews between sensors and the scheduler, resulting in either rejecting fresh data due to false stale detection or admitting stale data due to false fresh detection [51]. In such an instance, this effect can be mitigated by periodically synchronizing clocks using the NTP protocol for networked sensors. Hardware synchronization should be used for local sensors. In addition, sanity checks can be applied to reject camera jobs where ($\Delta < 0\ clock\ skew$) or $\Delta > 10s$ (indicating clearly erroneous data).

7.2.1. Permanent Failure

In case of permanent failure, the data fusion algorithm continues to fuse data from the remaining sensors (e.g., radar, lidar, IMU). However, the FDS admission control component will automatically reject the jobs related to the failed camera since no fresh data is available to fuse.

7.2.2. Intermittent failure (e.g., radar interference)

Intermittent failures result in periodic bursts of stale data, and the FDS filtration feature automatically filters out these bursts, preventing them from being fused into the data fusion.

7.3. Implementation Complexity Analysis

Computational Overhead: The following cycle count estimates have been produced for an ARM Cortex-M4

running at 240 MHz (automotive typical microcontroller type). When performing Primitive operations on an x86 (timestamp arithmetic, threshold comparison, exponential way of calculating), each one is $< 0.1\mu s$, the full context-switch overhead will be greater than these numbers, as they aggregate significant numbers of operations such as this for all ready tasks.

7.3.1. Per Job Admission Control

- Exponential Evaluation: ≈30 CPU Cycles (ARM Cortex-M4)
- Timestamp Subtraction: 5 CPU Cycles
- Threshold Comparison: 2 CPU Cycles
- Total < 50 CPU Cycles, or 0.21μs @ 240 MHZ

7.3.2. Per Task Priority Update

- Deadline Urgency (dividing by Upper Bound): 20 CPU Cycles
- Freshness Urgency (exponential + multiply): 35 CPU Cycles
- Criticality Weights: (table lookups): 5 CPU Cycles
- Priority Combination: 10 CPU Cycles
- Total < 100 CPU Cycles, or 0.42

7.3.3. System-Wide Each Context Switch

- Assuming 10 Ready Tasks, all ready task priority updates: 10 × 100 = 1000 CPU Cycles
- Scheduler Selection: 200 CPU Cycles
- Total < 1200 CPU Cycles, or 5.0 μs
- Overhead Fraction: For 100 μs Average Task Execution:

$$\text{Overhead} = 5.0\mu s/100\mu s = 5.0\% \quad (26)$$

per context switch, with ≈ 200 context switches per second under high load, then the total scheduler overhead is $\approx 1.0\ ms/s$.

7.4. Memory Footprint

Here are the sizes for data related to each task:

- Timestamp (4 bytes)- Data age (4 bytes)
- Last value (4 bytes)- Sensor type (1 byte)
- Criticality level (1 byte)- Parameter values (12 bytes)
- Total size for each task = 26 bytes.

For n =10 tasks, this will be 10x26 = 260 bytes (0.2%) out of a typical 128KB of RAM found in automotive microcontrollers.

The code size includes:

- FDS scheduler core (fds_scheduler.c): 320 lines & 8KB of flash.
- FreeRTOS hooks: 150 lines & 4KB of flash.

Total code size = 12KB or 0.6% of the 2MB flash memory contained in a typical automotive electronic control unit.

In summary, FDS software overhead is minimal on modern embedded devices and thus is applicable for any resource-constrained automotive or industrial application.

As an example of FDS's value-driven behavior, let us explore the detailed calculations using an example of motivated pedestrian detection.

7.4.1. Case Study: Pedestrian Detection Timeline FDS

For a camera task, the worst-case execution time is 15 ms and has a period of 30 ms, rated as ASIL B. In this case, $t_c = 0ms$. The application will have a CPU demand that is 303.6 percent at t_c. The freshness parameters used for this item are $\Delta_{opt} = 20ms$ (most valuable if processed within 20 ms), $\Delta_{max} = 100ms$ (becomes invalid after 100 ms), and λ = 0.05 (decreased value for processing over time, based upon the decay rate). As an ASIL B item, FDS only accepts the job if it has a predicted value upon completion that is equal to or greater than the FDS admission threshold for ASIL B items (0.5).

7.4.2. Traditional Scheduling Execution

The camera frame was taken at $t_c = 0ms$, and it enters the ready queue with an initial age of $\Delta = 0ms$. Because there are many higher-priority tasks already occupying the processor, this job remains waiting to be executed until $t = 80ms$ has elapsed. Execution begins then at $t = 80ms$ old data and completes at $t = 95ms$ old data, resulting in a final data age of 95 ms. The way the task still meets its deadline of 100 ms is because almost all of the frames will have gone past their validity window, making the scheduler consider this to be a success in timing, even though the frame's useful value has deteriorated greatly.

The deadline is met for a traditional scheduler, but the delivered value is 0.024, which is 97.6% degradation from optimal. This stale frame shows the pedestrian position, which is 95ms old.

$$V(95) = e^{-0.05\cdot(95-20)} = e^{-3.75} = 0.024 \quad (27)$$

7.4.3. FDS Admission Decision

At $t = 0$, when the job arrives $\Delta_{current} = 0ms$, $t_{pred} = 0 + C_{cam} = 15ms$, $V_{pred} = 1.0(\Delta_{pred} < \Delta_{opt} = 20ms)$, Admit if $V_{pred} = 1.0 > \theta_{ASIL-} = 0.5$, since the predicted completion age is within the optimal window.

7.4.4. FDS handling Stale Data

Now let us consider a frame arriving late at $t = 70ms$ with the same. $t_c = 0$, $\Delta_{current} = 70 - 0 = 70ms$, $\Delta_{pred} = 70 + 15 = 85ms$, $V_{pred} = e^{-0.05\cdot(85-20)} = e^{-3.25} = 0.039$. The decision is rejected because 0.039 < 0.5. At completion, FDS correctly determines that this job is a waste of time and rejects the job, and saves 15 ms of CPU for newer data.

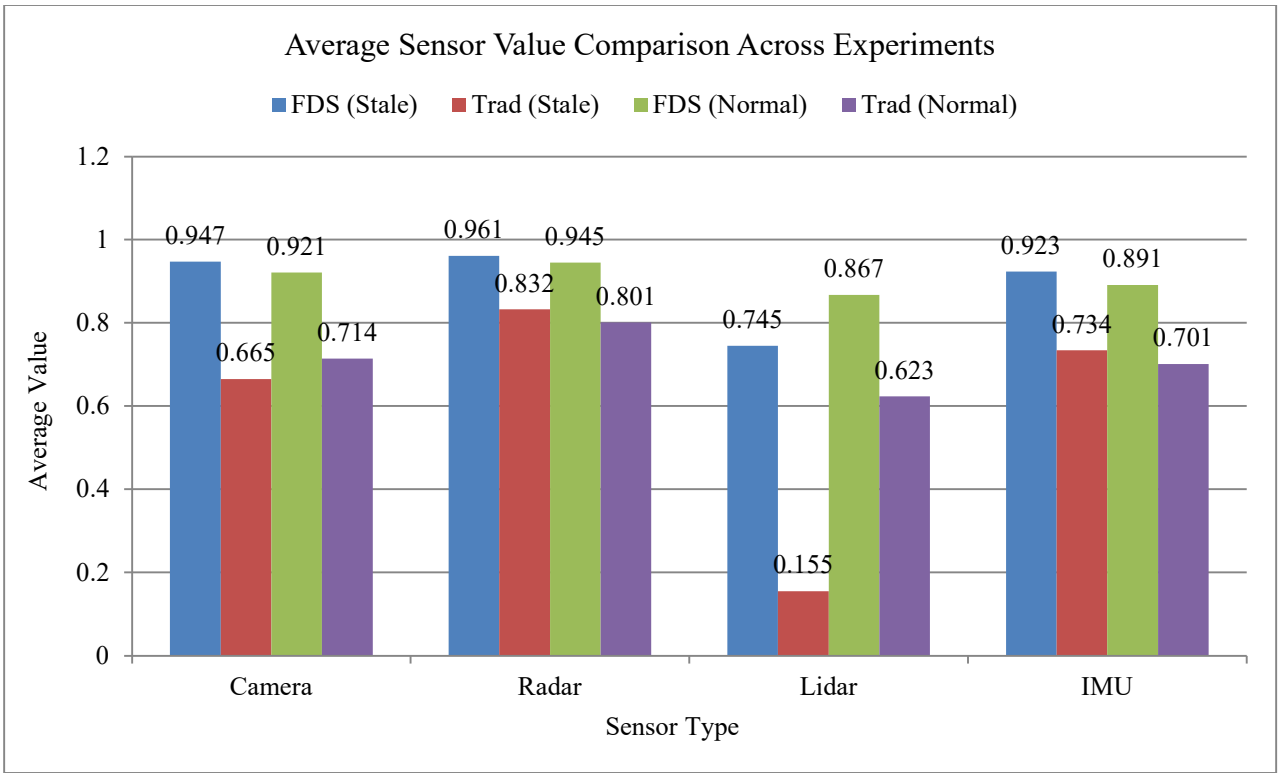

Fig. 4 Average sensor value across all four experiment types, comparing FDS (teal/blue) with Traditional RMS scheduling (orange/purple). Results aggregate the full 15-second run for each of the six experimental conditions. FDS consistently outperforms Traditional scheduling in all four sensor channels. The largest gain is for lidar under stale injection (FDS: 0.745 vs. Traditional: 0.155, +381%), driven by FDS's complete rejection of validity-exceeded Lidar jobs and CPU reclamation for fresher data. FDS also outperforms in the Normal Load (0% stale) scenario because its freshness-urgency term (Equation 8) accelerates processing of data that is actively losing value, improving average completion age relative to static-priority

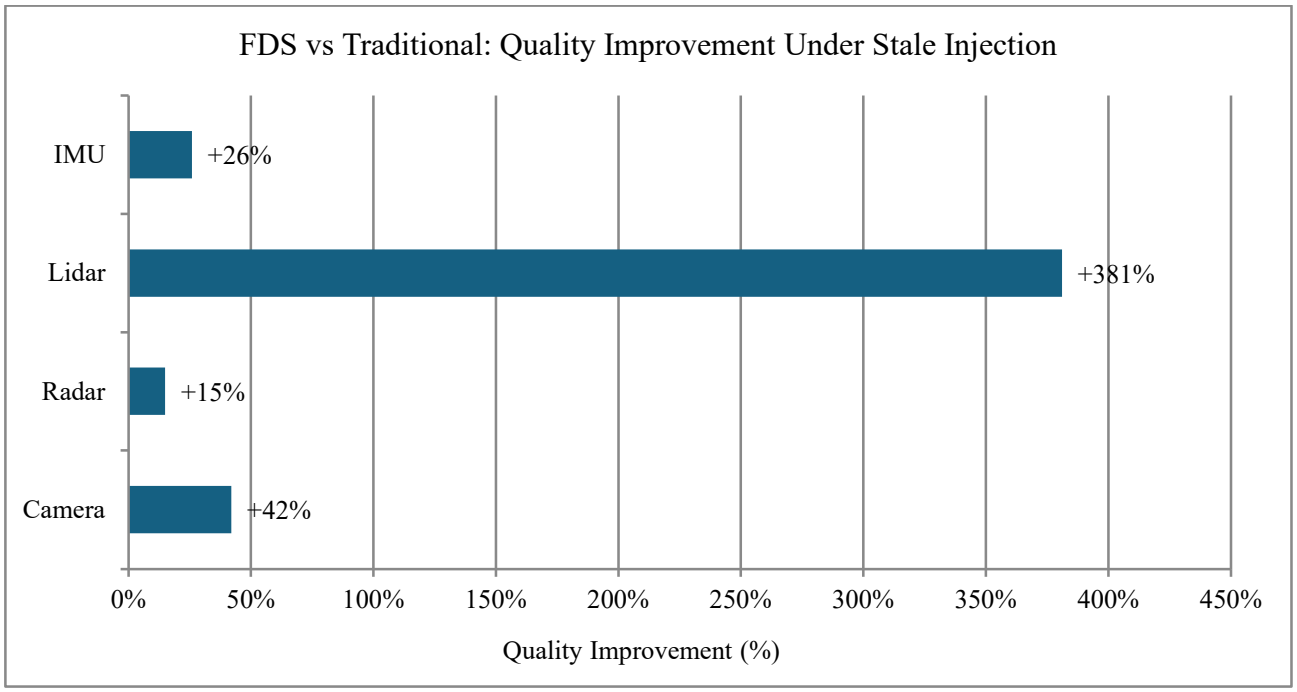

Fig. 5 Quality improvement of FDS over Traditional RMS scheduling under 30% stale-data injection, expressed as percentage improvement in average sensor value: Lidar achieves the highest gain (+381%) because its validity window is most constrained (150 ms maximum) and its priority boost factor is low; Traditional scheduling systematically delays Lidar jobs past the validity boundary, while FDS rejects stale Lidar jobs before any CPU is spent. Camera achieves +42% through selective admission of freshest-captured frames. All improvements are relative to the same 100% CPU utilization, confirming gains arise from intelligent load selection, not reduced system demand.

7.4.5. Safety Impact Quantification

From a safety perspective, the difference of 1.22 m, at a speed of 50 km/h, equates to approximately 88 ms of reaction time. In emergency braking situations, this can represent a critical margin of safety. FDS provides a benefit of safety by rejecting stale sensor data before processing through its value-based admission-control mechanism.

Additionally, by rejecting stale jobs, FDS increases CPU efficiency, eliminating 15 ms of wasted processing on a single core. When stale job rates are below 30%, the efficiency gains built up over many repetitive camera jobs can reach a total of 2250 ms (or 2.25 seconds) of reclaimed compute time over a 15-second period. This reclaimed computing capacity can be put to use for newer, more useful data. Therefore, the central tenet of FDS is that processing time is a finite resource in real-time applications; it should not be used to process data that provides little or no value.

7.5. Parameter Sensitivity Analysis

The effectiveness of the (FDS) is determined by specific characteristics($\Delta_{opt}, \Delta_{max}, \lambda, \theta\_L$) Moreover, tested the effect of adjusting each characteristic on the operation of the system.

7.5.1. Decay Rate λ Sensitivity

All other characteristics were constant while changing the decay rate of the camera (which was used to measure the average and at what rate jobs would be accepted). The following observations were made:

A camera decaying with a low decay rate has higher acceptance rates, but the overall average accepted values are lower (less than optimal). A camera decaying with a high decay rate has lower acceptance rates, but higher average values of accepted jobs. An optimal decay rate will obtain a 70% acceptance rate, while also producing an average value of greater than .90.A very high decay rate can create excessive rejection rates, and thus cause under-utilization of the CPU.

7.5.2. Tune the Parameters

Analyzing parameter sensitivity allows for a different perspective on the same trade-offs depicted in the graph above (i.e., delta max and L directly constrain the trade-off between throughput and data value). If the validity window is narrow, there is a higher potential of data rejection as well as starvation of data. Conversely, if the validity window is wide, it allows for a higher likelihood of data acceptance; however, older data may also be accepted. Regarding the threshold L, a tighter threshold will accept data with a higher data value than a looser threshold; however, a tighter threshold will also produce a lower throughput than a looser threshold.

7.5.3. Criticality Alignments

The three established thresholds ($\theta_{ASIL-D} = 0.7, \theta_{ASIL-B} = 0.5, \theta_{QM} = 0.3$) will assist with the system to meet the safety requirements. Generally, the closer to real-time data is available for a critical task, the less risk to the task.

7.5.4. Sensitivity of Tuning

When the parameters of λ, Δ_{max}If L and L are set too conservatively, the system will reject many jobs and have low utilization of the CPU. In addition, the rejection rates will be elevated along with admission rates that are less than 50%, thereby under-utilizing the available processing resources.

On the other hand, when λ, Δ_{max}, and L are set with more relaxation; then the system will admit most jobs (many times more than 95%), but due to the low average value in the accepted data, it will cause the overall benefit of the FDS to decrease even closer to the behavior of a traditional scheduling method. As the average output quality begins to decrease, it will be necessary to tighten the parameters.

By using an adaptive tuning method based on online learning, these parameters can be adjusted from real-time performance measurements and fusion error observations to improve performance further. By doing so, the FDS will be able to close the loop between output quality and admission control while providing for a more dynamic response to the changing conditions of the operating environment.

7.6. Advantages of FDS

Value Optimization improvement of 42% to 381% in quality through the elimination of stale data. All ASIL-D admitted tasks will meet all task deadlines (5.1). Attain CPU efficiency by saving 30.5% by eliminating useless tasks from processing. The system will still operate under an overload of 415% of maximum CPU demands. There are formal conditions that provide provable schedulability.

7.7. Broader Applicability

FDS is utilized in the evaluation of automotive security sensors, but could be extended to include surgical robotics, which requires haptic feedback with a freshness of less than 10 ms. [4], Industrial Control to include vibration monitoring and process control[5]. Drone navigation to include visual-inertial odometry [43]. IoT Edge computing to include time-sensitive networking [44]. Use of FDS will apply to any cyber-physical system where the age of data is important.

8. Conclusion

This research presented Freshness-Driven Scheduling (FDS) for scheduling tasks that consider not only deadlines, but also how fresh the data is. The deadline-centric way of scheduling does not provide a way to guarantee that data is current. The FDS method includes the following features, first exponential value degradation models and hard validity cut-offs; second, predictive admission control that rejects jobs that are stale; third, a mechanism for dynamically balancing the priorities of tasks based on deadline urgency and ISO 26262

criticality; and finally, an analysis of whether admitted jobs will be schedulable for safety-critical tasks.

The experimental validation of FDS, performed under 6 different controlled environments, showed improvements to per-sensor quality between 42% and up to 381% over Rate Monotonic Scheduling when 30% of the injected jobs into the system were stale. As a result of rejecting worthless jobs, around 30.5% of the CPU budget was reclaimed by FDS. Under conditions of overload, FDS was also able to maintain zero ASIL-D deadline misses, even when overloads started from 4.2 times the CPU limit. The system also provided a formal worst-case response-time analysis of admitted ASIL-D tasks to determine scheduling equivocality of the ASIL-D tasks, with a final overall WCRT, which decreased by approximately 11%, for the Fusion task resulting from reduced interference after the admission was controlled by FDS. As cyber-physical systems continue to evolve and become more reliant upon multi-modal sensor fusion to provide accurate data, the use of temporal coherence rather than mere compliance with deadlines will become a better measure of correctness.

8.1. Limitations

All experiments were carried out using the FreeRTOS POSIX simulator and on a single CPU core. Because most modern automotive ECUs are multicore (e.g., Renesas R-Car, NXP S32G), WCET estimates, and the admission reliability of an analysis depend on cache effects and inter-core interference to and from each core. The single-core execution results will provide a controlled, baseline reference, while the multicore execution will require a separate study.

Assume optimistic permission granting. The Future Job Scheduler (FDS) predicts completion time (V_{pred}) based on the formula Δ_{pred} = t + C_i− $t_{c,j}$, where the queuing delay is assumed to be zero, thus establishing V_{pred} As an upper bound on optimistic values, in the case of sustained overload, it is possible for work to start but not complete on time due to the long queuing delay. Therefore, an admission test that takes into consideration queue depth will narrow the gap between V_{pred} and V_{actual} at $O(n)$ for each schedule.

Use an injection model that injects deterministic stale data. Stale data was injected into the system using a Bernoulli injection model with constant probability (p = 0.30). In practice, there will be bursts of staleness from sensors, and the staleness between adjacent time periods will be highly correlated. The results from experiments indicate that staleness can be expected to degrade in a linear fashion when using the Bernoulli model, but when using a burst model, the results may differ.

Sensors must be calibrated to yield optimal performance. The Future Job Scheduler (FDS) must have parameters (Δ_{opt},Δ_{max}, λ) for each type of sensor. In order to deploy systems with optimal FDS parameters, system manufacturers will likely be required to calibrate FDS parameters for each sensor type. If the calibration is incorrect, the system will likely reject too many jobs or allow jobs to age longer than specified. Expect a single-pass WCRT limit. The same admission probability will be used to estimate the working set WCRT regardless of whether a ρ_j value is less than 1 or greater than 0 (where ρ_j is the actual admission rate that is present. The admission probability is a runtime quantity and may change due to the amount of stale data being injected into the FDS and the load on the operating system.

8.2. Future Work

8.2.1. Multicore (FDS) Extensions

Use Partitioned FDS (P_{FDS}) to assign sensor processing to dedicated cores with sensor information fused on a separate core [52]. Disadvantage would be load imbalance; advantage would be no migration overhead. Another approach could be Global FDS (G_{FDS}): Use a single ready queue where tasks can migrate [53]. Disadvantage would be cache thrashing; Advantage would be load balance. The hybrid option would be to pin ASIL-D to cores for deterministic operation and allow QM to migrate for flexible operation.

8.2.2. Statistical Model Checking:

When verifying $P(ASIL - D) < 10^{-9}$ through simulation, use tools such as UPPAAL SMC or PRISM for the performance of the analysis. The best-case execution time With a Value Overhead (WCET) is achieved by extending existing static analysis tools to support the analysis of the exponential cost of evaluating operations.

8.2.3. Hardware Accelerators

Exponential calculations cause a lot of overhead. If this can be accelerated using an FPGA/ASIC, it could reduce it from 30 cycles to 3 cycles and save energy at the same time. The target is to create a new automotive SoC with an FDS co-processor.

8.2.4. Cloud-Edge Integration

Offload intensive fusion activities to the cloud while keeping data fresh: Local FDS will perform lightweight processing at the edge, Cloud high definition fusion will occur using cloud services, as long as the round-trip time of the network and processing is Δ_{max} Admission will deny cloud offload if the predicted completion time is too far in the future. An example application is an autonomous drone operating on a 5G edge cloud.

8.2.5. Safety Monitor Integration:

ISO 26262 requires all applications to have runtime monitoring. The functions of runtime monitoring performed by the FDS include a value monitor to drop the average value and produce an alert. Admission monitors to produce an alert

if the rejection rate is >80%, Coherency monitor to trigger a safe state if $\Gamma < \Gamma_{min}$.

8.2.6. ROS 2 DDS Middleware

Develop the integration of the FDS with the Robot Operating System 2 (ROS2) in order to have a mapping from DDS QoS policy to parameters within the FDS, distributed value-aware scheduling for robot swarms, and to carry out priority-based topic filtering. These goals support multicore scalability, adaptive tuning, and stronger formal assurance for production safety-critical systems that use this integrated FDS/ROS2 system.

8.3. Ethical Considerations, Safety Risks, and Safeguards

This research project does not involve people or animals, and there is no collection of any personal data during experiments. The experiment is performed using synthetic workloads in a controlled laboratory setting. However, as FDS will be used in lives that will be influenced by safety-critical automotive and medical systems, ethical issues with regard to the design decisions involved in creating them are discussed in detail herein.

8.3.1. Safety Risk 1: Sensor Blackout from Admission Control

The main FDS mechanism is rejecting sensor jobs that were predicted as stale at completion; this creates the potential for a safety hazard that is not present in traditional schedulers, namely, sensor blackout. If all critical sensor jobs arrived as stale at the same time due to a burst interference event or a temporary sensor fault (e.g., radar not responding), the FDS could reject all of them, thus creating a temporary interruption in the fusion pipeline. The following safeguards are in place:

- ASIL-D minimum admission guarantees. Minimum admission guarantee for any ASIL-D task is 1 job per period, regardless of its predicted value, so no ASIL-D sensor (IMU, Radar, Fusion) can ever experience complete blackouts.
- Threshold conservatism: A 0.70 threshold θ_D For admitting Radar data has been set with a maximum allowable time interval between successive sensor samples of approximately 178 ms, based on the 22 ms buffer associated with the maximum allowable time interval (200 ms).
- Admission flag: All forced-admitted stale jobs will be flagged so that reduced confidence will be applied to them during downstream fusion, preventing them from being silently included in the integrated output data set.
- Coherence monitor: The $\Gamma(t)$ metric will be continuously monitored, such that if the Γ (coherence) is less than Γ_{min}A transition to a safe state can be initiated as per ISO 26262 ASIL-D.

8.3.2. Safety Risk 2: Sensor Prioritization Bias

The FDS priority function (Equation 6) has an inherent bias towards longer validity windows (Radar, Δ_{max}= 200 ms) and against shorter validity windows (IMU Δ_{max} = 20 ms). Systematic biases could be introduced by high-load conditions in which Camera and IMU data are consistently rejected while Radar data is always accepted and produces Radar-dominant fusion outputs, thereby minimizing pedestrian detection relative to vehicle tracking. Mitigating Strategies: The ASIL criticality boost β_{wL}will be calculated for each sensor independently of whether its validity window exceeds the minimum threshold. [e.g., IMU is ASIL-D and receives the full β=100 boost; thus, further protecting IMU from fusion error]. The Coherence Monitor will identify systematic fusion skew (bias) due to the admission of one type of sensor. Long-term orders should consider statistical evidence of each sensor's admission rate, so any bias or drift may be detected in future fusion output.

8.3.3. Safety Risk 3: Parameter Miscalibration

Published sensor specifications allow for calibration of the FDS parameters. $\Delta_{opt}, \Delta_{max}, \lambda$. However, variations in the environment (e.g., rain, fog, temperature) as well as sensor aging and substitutes/procuring different sensors cause the sensors to behave differently during production. If calibrated incorrectly, FDS parameters can lead to either.

- Excessive rejection of valid fresh data leading to a perception quality that is below that provided by the traditional scheduler, or
- Insufficient protection, allowing stale data through, and, therefore, without any warning, resulting in degraded fusion quality. The sensors must be characterized at each vehicle production variant through the use of HIL testing. The parameter files that are created during this process must be version-controlled and managed under ISO 26262 change management procedures.

8.3.4. Deployment Context and Responsibility

FDS is presented as a research prototype demonstrating the feasibility and benefit of value-aware scheduling. It is not a production-certified RTOS component. Deployment in a vehicle ASIL-D system requires:

- formal certification of the FDS source code under ISO 26262 Part 6 (Software)
- Hardware-specific WCET analysis replacing the simulation-based estimates used in this paper
- Exhaustive fault injection testing per ISO 26262 Part 8
- Review and approval by a functional safety manager. The authors assume no liability for deployment outside of the experimental context described in this paper.

These deployment expectations are consistent with broader functional-safety practice in standards such as ISO 26262 and IEC 61508 [55].

Conflicts of Interest

The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper.

Funding Statement

There is no funding received for this research.

Acknowledgments

The author thanks the FreeRTOS open-source community for the RTOS platform used in this research. All experimental data, task set designs, priority assignment algorithms, performance data, and findings contained within this document represent the author's original work and not the work of AI. The author used AI-assisted writing tools for only writing assistance, i.e., grammar corrections.

References

[1] C.L. Liu, and James W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-real-time Environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973. [CrossRef] [Google Scholar] [Publisher Link]

[2] Giorgio C. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, 3rd Edition, Springer, pp. 1-524, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[3] Roy D. Yates et al., "Age of Information: An Introduction and Survey," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 5, pp. 1183–1210, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[4] Shaoshan Liu et al., "Computer Architectures for Autonomous Driving," *Computer*, vol. 50, no. 8, pp. 18–25, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[5] Russell H. Taylor et al., "Medical Robotics and Computer-Integrated Surgery," *Springer Handbook of Robotics*, pp. 1657–1684, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[6] Ragunathan Rajkumar et al., "Cyper-Physical Systems: The Next Computing Revolution," *Proceedings of the 47th Design Automation Conference*, pp. 731-736, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[7] ISO 26262, Road Vehicles–Functional Safety, International Organization for Standardization, 2018. [Online]. Available: https://www.iso.org/publication/PUB200262.html

[8] Sanjit Kaul, Roy Yates, and Marco Gruteser, "Real-time Status: How Often should One Update?," *2012 Proceedings IEEE INFOCOM*, Orlando, FL, USA, pp. 2731-2735, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[9] Maice Costa, Marian Codreanu, and Anthony Ephremides, "On the Age of Information in Status Update Systems with Packet Management," *IEEE Transactions on Information Theory*, vol. 62, no. 4, pp. 1897–1910, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[10] Robert I. Davis, and Alan Burns, "A Survey of Hard Real-time Scheduling for Multiprocessor *Systems*," *ACM Computing Surveys*, vol. 43, no. 4, pp. 1–44, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[11] M.L. Dertouzos, and A.K. Mok, "Multiprocessor Online Scheduling of Hard-real-time Tasks," *IEEE Transactions on Software Engineering*, vol. 15, no. 12, pp. 1497–1506, 1989. [CrossRef] [Google Scholar] [Publisher Link]

[12] Y.T. Leung Joseph, and Jennifer Whitehead, "On the Complexity of Fixed-priority Scheduling of Periodic, Real-time Tasks," *Performance Evaluation*, vol. 2, no. 4, pp. 237–250, 1982. [CrossRef] [Google Scholar] [Publisher Link]

[13] M. Joseph, and P. Pandya, "Finding Response Times in a Real-time System," *The Computer Journal*, vol. 29, no. 5, pp. 390–395, 1986. [CrossRef] [Google Scholar] [Publisher Link]

[14] N. Audsley et al., "Applying New Scheduling Theory to Static Priority Pre-emptive Scheduling," *Software Engineering Journal*, vol. 8, no. 5, pp. 284–292, 1993. [CrossRef] [Google Scholar] [Publisher Link]

[15] Steve Vestal, "Preemptive Scheduling of Multi-criticality Systems with Varying Degrees of Execution time Assurance," *28th IEEE International Real-Time Systems Symposium*, Tucson, AZ, USA, pp. Tucson, AZ, USA, pp. 239-243, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[16] S.K. Baruah, A. Burns, and R.I. Davis, "Response-time Analysis for Mixed Criticality Systems," *2011 IEEE 32nd Real-time Systems Symposium*, Vienna, Austria, pp. pp. 34-43, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[17] Alan Burns, and Robert Davis, *Mixed Criticality Systems–A Review*, The University of York, 2013. [Google Scholar] [Publisher Link]

[18] Patrick Graydon, and Iain Bate, "Safety Assurance Driven Problem Formulation for Mixed-criticality Scheduling," *Proceedings of Workshop on Mixed Criticality Systems*, pp. 19–24, 2013. [Google Scholar]

[19] Risat Mahmud Pathan, "Schedulability Analysis of Mixed-criticality Systems on Multiprocessors," *2012 4th Euromicro Conference on Real-Time Systems*, Pisa, Italy, pp. 309-320, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[20] G.C. Buttazzo et al., "Elastic Scheduling for Flexible Workload Management," *IEEE Transactions on Computers*, vol. 51, no. 3, pp. 289-302, 2002. [CrossRef] [Google Scholar] [Publisher Link]

[21] Igor Kadota, Abhishek Sinha, and Eytan Modiano, "Scheduling Algorithms for Optimizing Age of Information in Wireless Networks with Throughput Constraints," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1359-1372, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[22] Ahmed M. Bedewy, Yin Sun, and Ness B. Shroff, "Age-optimal Information Updates in Multi Hop Networks," *2017 IEEE International Symposium on Information Theory*, Aachen, Germany, pp. pp. 576-580, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[23] Rajat Talak, Sertac Karaman, and Eytan Modiano, "Optimizing Information Freshness in Wireless Networks Under General Interference Constraints," *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Los Angeles CA USA, pp. 61–70, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[24] Mohamad A. Abd-Elmagid, Nikolaos Pappas, and Harpreet S. Dhillon, "On the Role of Age of Information in the Internet of Things," *IEEE Communications Magazine*, vol. 57, no. 12, pp. 72–77, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[25] Raymond Keith Clark, "*Scheduling Dependent Real-Time Activities*," Doctoral Thesis, Carnegie Mellon University, pp. 1-259, 1990. [Google Scholar] [Publisher Link]

[26] P. Li et al., "A Formally Verified Application-level Framework for Real-time Scheduling on POSIX Realtime Operating Systems," *IEEE Transactions on Software Engineering*, vol. 30, no. 9, pp. 613–629, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[27] M. Hamdaoui, and P. Ramanathan, "A Dynamic Priority Assignment Technique for Streams with (m,k)-firm Deadlines," *IEEE Transactions on Computers*, vol. 44, no. 12, pp. 1443–1451, 1995. [CrossRef] [Google Scholar] [Publisher Link]

[28] P. Ramanathan, "Overload Management in Real-time Control Applications using (m,k)-firm Guarantee," *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, no. 6, pp. 549–559, 1999. [CrossRef] [Google Scholar] [Publisher Link]

[29] G. Bernat, A. Burns, and A. Liamosi, "Weakly Hard Real-time Systems," *IEEE Transactions on Computers*, vol. 50, no. 4, pp. 308–321, 2001. [CrossRef] [Google Scholar] [Publisher Link]

[30] Goran Frehse et al., "Formal Analysis of Timing Effects on Closed-loop Properties of Control Software," *2014 IEEE Real-Time Systems Symposium*, Rome, Italy, pp. 53-62, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[31] H. Durrant-Whyte, and T. Bailey, "Simultaneous Localization and Mapping: Part I," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006. [CrossRef] [Google Scholar] [Publisher Link]

[32] Sebastian Thrun, Wolfram Burgard, and Dieter Fox, *Probabilistic Robotics*, MIT Press, pp. 1-672, 2005. [Google Scholar] [Publisher Link]

[33] Vishwanath A. Sindagi, Yin Zhou, and Oncel Tuzel, "MVX-Net: Multimodal VoxelNet for 3D Object Detection," 2019 International Conference on Robotics and Automation, Montreal, QC, Canada, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[34] Paul Furgale, Joern Rehder, and Roland Siegwart, "Unified Temporal and Spatial Calibration for Multi-sensor Systems," *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, Japan, pp. 1280-1286, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[35] Hyunggi Cho et al., "A Multi-Sensor Fusion System for Moving Object Detection and Tracking in Urban Driving Environments," *2014 IEEE International Conference on Robotics and Automation*, Hong Kong, China, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[36] T.D. Larsen et al., "Incorporation of Time Delayed Measurements in a Discrete-time Kalman Filter," *Proceedings of the 37th IEEE Conference on Decision and Control*, Tampa, FL, USA, pp. 3972-3977, 1998. [CrossRef] [Google Scholar] [Publisher Link]

[37] Richard Barry, Mastering the FreeRTOS Real Time Kernel, 2016. [Google Scholar] [Publisher Link]

[38] T.P. Baker, "Stack-based Scheduling of Real Time Processes," *Real-Time Systems*, vol. 3, pp. 67–99, 1991. [CrossRef] [Google Scholar] [Publisher Link]

[39] FreeRTOS Symmetric Multiprocessing (SMP) with FreeRTOS, 2026. https://freertos.org/symmetric-multiprocessing-introduction.html

[40] Ravindra Jejurikar, Cristiano Pereira, and Rajesh Gupta, "Leakage Aware Dynamic Voltage Scaling for Real-time Embedded Systems," *Proceedings of the 41st Annual Design Automation Conference*, San Diego CA USA, pp. 275–280, 2004. [CrossRef] [Google Scholar] [Publisher Link]

[41] SAE J3016, "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-road Motor Vehicles," *SAE International*, 2021. [Google Scholar] [Publisher Link]

[42] Jos Elfring et al., "Effective World Modeling: Multisensor Data Fusion Methodology for Automated Driving," *Sensor*, vol. 16, no. 10, p. 1668, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[43] Anastasios I. Mourikis, and Stergios I. Roumeliotis, "A Multi-state Constraint Kalman Filter for Vision-aided Inertial Navigation," *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007. [CrossRef] [Google Scholar] [Publisher Link]

[44] Norman Finn, "Introduction to Time-sensitive Networking," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 22–28, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[45] Ahmed Arafa, and Sennur Ulukus, "Timely Updates in Energy Harvesting Two-Hop Networks: Offline and Online Policies," *IEEE Transactions on Wireless Communications*, vol. 18, no. 8, pp. 4017-4030, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[46] Zheng Chen et al., "Optimizing Information Freshness in a Multiple Access Channel with Heterogeneous Devices," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 456–470, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[47] Mustafa Emara, Miltiades C. Filippou, and Dario Sabella, "MEC-Enhanced Information Freshness for Safety-critical C-V2X Communications," *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, Dublin, Ireland, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[48] Md Kamran Chowdhury Shisher, and Yin Sun, "How Does Data Freshness Affect Real-time Supervised Learning?," *Proceedings of the 23rd ACM International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, pp. 31–40, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[49] Yin Sun, Yury Polyanskiy, and Elif Uysal-Biyikoglu, "Remote Estimation of the Wiener Process over a Channel with Random Delay," *IEEE Transactions on Information Theory*, Aachen, Germany, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[50] Jamil Fayyad et al., "Deep Learning Sensor Fusion for Autonomous Vehicle Perception and Localization: A Review," *Sensors*, vol. 20, no. 15, p. 4220, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[51] John C. Eidson, Mike Fischer, and Joe White, "IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," *Proceedings of the 34th Annual Precise Time and Time Interval Systems and Applications Meeting*, Reston, Virginia, pp. 243-254, 2002. [Google Scholar] [Publisher Link]

[52] Alan Burns, and Robert I. Davis, "A Survey of Research into Mixed Criticality Systems," *ACM Computing Surveys*, vol. 50, no. 6, pp. 1–37, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[53] Haohan Li, and Sanjoy Baruah, "Outstanding Paper Award: Global Mixed:Criticality Scheduling on Multiprocessors," *2012 24th Euromicro Conference on Real-Time Systems (ECRTS)*, Pisa, Italy, 2012. [CrossRef] [Google Scholar] [Publisher Link]

[54] Francisco E. Páez et al., "FreeRTOS User Mode Scheduler for Mixed Critical Systems," *2015 Sixth Argentine Conference on Embedded Systems (CASE)*, Buenos Aires, Argentina, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[55] IEC 61508: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems. International Electrotechnical Commission, 2010. [Online]. Available: https://webstore.iec.ch/en/publication/5515

Appendix 1
Value Function Derivation

The model of exponential value degradation (Eq. 3) has been derived from principles relating to Information Theory. Data recorded at t_c provides the basis for an Environmental State $X(t_c)$ at the point of recording. As time progresses (t), a corresponding environmental state $X(t)$ has developed.

Information loss Rate is given by Markov dynamics, mutual information decays exponentially as given below.

$$I(X(t_c); X(t)) = I_0 \cdot e^{-\lambda(t-t_c)} \quad (30)$$

Setting $I_0 = 1$ and $\Delta = t - tc$ yields $V(\Delta) = e^{-\lambda\Delta}$

Physical Interpretation of λ:

For the camera, its scene change rate, for radar, its Doppler shift uncertainty accumulation, for lidar, its point cloud decorrelation due to object motion, and for IMU, its vibration-induced drift, and angular velocity integration error.

Optimal Age Δ_{opt}:

This indicates a minimum processing latency for data to process under no load. Anything processed faster than Δ_{opt} It is not going to have any more value (V = 1:0) because all processors can process at the same time.

Maximum Age Δ_{max}:

Data starts to be false and/or misleading due to a lack of quality and value past this point. For example, a camera with 300 ms of latency would place the vehicle more than 4.2 m in the wrong location if the vehicle was traveling at highway speeds; therefore, an inaccurate camera could be worse than not having any information to provide for collision avoidance purposes.

Appendix 2
Experimental Configuration Details

Hardware Infrastructure

- Processor(s) Intel Core i7-8550U @ 2.4GHz, Single Core, and Taskset
- RAM: 16 GB DDR4
- Operating System: Ubuntu 20.04 LTS (Kernel version: 5.15.0)
- FreeRTOS: Version 10.4.3 POSIX
- Compiler: GCC 9.4.0, with -O2 optimization techniques enabled.

Workload Parameters

The time required to process automotive sensor data based solely on the performance of those sensors is:

- The worst-case execution time (WCET) for a camera: 15 ms for processing a 640x480 frame of video with Canny edge detection.
- The WCET for a radar is 12 ms for processing FFT-based range/Doppler data from 32 targets.
- The WCET for LiDAR: 25 ms for fitting digital planes using RANSAC on a cloud of 10,000 points.
- The WCET of an Inertia Measurement Unit (IMU): 3 ms for coherent filtering to estimate orientation.
- The WCET for multiple sensor fusion: 10 ms for updating an Extended Kalman Filter (EKF) with four sensor inputs.

The WCET for all sensors was determined by performing 1,000 runs with the worst possible input conditions (99th percentile + 10 % safety factor).

Timing Infrastructure

- Clock source: CLOCK_MONOTONIC_RAW (not affected by NTP adjustments)
- Tick resolution: 1 ms (FreeRTOS configTICK_RATE_HZ = 1000)
- Timestamp precision: Microsecond (using clock_gettime())
- Jitter mitigation: CPU affinity pinned, real-time priority (SCHED FIFO), IRQs disabled on test core