講演番号	039

A Semantics-Preserving Master Scheduler for Mixed-Criticality Control in Lingua Franca

Yutaka Matsubara [1] Wenhung Kevin Huang [2] Akihito Iwai [2]

1) Nagoya University
Furo-cho, Chikusa-ku, Nagoya, 464-8603, Japan (E-mail: yutaka@ertl.jp)
2) DENSO CORPORATION
1-1 Showa-cho, Kariya, Aichi, 448-8661, Japan

KEY WORDS: software and its underlying technologies, software-partitioning, operating system, software platform, SDV, Lingua Franca, mixed-criticality scheduling, deterministic reactive systems, controlled degradation[E3]

Mixed-criticality cyber-physical systems must protect high-criticality control functions—such as braking or stabilization loops—without sacrificing deterministic reactive behavior, even when low-criticality workloads for logging, telemetry, or perception post-processing share the same execution platform. In Lingua Franca (LF), a deterministic coordination language whose runtime organizes execution by logical time, safe intervention is possible only at the ready-set boundary: the narrow point after the runtime constructs the ready set of enabled reactions for a logical-time tag but before it commits to a dispatch order. Fig. 1 shows how the Master Scheduler (MS) integrates with the LF runtime on a Linux-based software platform for Software-Defined Vehicles: the runtime exposes the ready set and per-reaction metadata—including criticality level, relative deadline, and recent timing history—to the user-space MS, which can select an eligible reaction, trigger configured degradation of optional low-criticality work, or safely fall back to the default dispatcher. Because every intervention is validated by semantic guards that check ready-set membership and metadata consistency, the approach preserves LF logical-time semantics while making overload handling explicit and auditable without kernel, RTOS, or hypervisor modification.

The design is driven by four requirements: semantics preservation (R1), user-space deployability (R2), controlled degradation under overload (R3), and observability with auditability (R4). These requirements lead to a conservative control plane in which all out-of-contract choices are rejected by runtime guards and ordinary LF execution is restored by explicit safe fallback, ensuring that uncertainty or incomplete information never leads to unsafe behavior. The key design distinction is between dispatch reordering and workload reduction. Reordering only permutes the same set of reactions already present in the ready set; because cumulative execution demand is unchanged, no permutation can reduce total demand within a logical-time step. When that demand exceeds available capacity, no reordering restores feasibility. Controlled degradation, by contrast, selectively removes eligible low-criticality work while preserving the semantics of the remaining execution, thereby reducing demand below the feasibility threshold. This structural difference—not merely an empirical observation—is central in mixed-criticality LF systems where high-criticality and low-criticality reactions are often released synchronously at the same logical-time tag.

Evaluation on a Linux-based containerized environment (Docker on Apple M3, 8 vCPUs, Ubuntu 24.04) shows three main results. In E1, reruns with n = 30 measured runs per condition showed near-zero control-plane overhead: for ready-set sizes N = 16, 32, 64, and 128, observe-only overhead was 0.35%, 0.14%, 0.14%, and −0.50%, while intervene overhead was −0.31%, −0.64%, 0.31%, and −0.07%, with all 95% confidence intervals crossing zero. In E2, high-criticality miss rate was 19.3% for the LF baseline (n = 60), 88.7% for MS + RT single-worker (n = 30), and 11.8% for MS + RT worker-group (n = 30), demonstrating that RT-control granularity is critical: elevating a single worker starves the remaining threads required for LF forward progress, causing severe priority inversion. In E3, ready-set selection alone did not consistently improve high-criticality timing under synchronous overload because selection cannot reduce workload volume, whereas controlled degradation—which removes eligible low-criticality reactions under sustained overload—reduced high-criticality miss rate to near zero in the evaluated scenarios, with zero semantic-guard fallbacks across all 630 runs.

The contribution of this work is therefore a semantics-aware interpretation of mixed-criticality scheduling for LF. The MS is not a hard real-time enforcer and does not replace OS-level isolation, preemption, or resource reservation mechanisms; instead, it acts as a user-space semantic control layer that makes overload handling explainable, reproducible, and compatible with logical-time determinism. The results do not imply ISO 26262 or ASIL-style certification guarantees, but they position the MS as a complementary mechanism that makes scheduling rationale visible for post-run analysis and policy tuning. Across the evaluated setup, the key implication is that overload handling in LF-based mixed-criticality systems is fundamentally a workload-management problem rather than a dispatch-ordering problem. Future work includes broader workload coverage, longer experimental runs, hardware isolation, integration with enforcement-oriented mechanisms, and coordinated control across multiple runtimes.

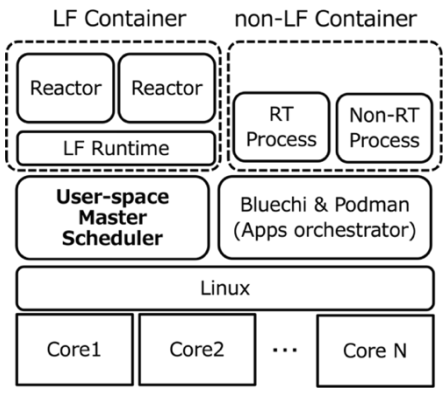

Fig. 1 Master Scheduler integration with LF Runtime including ready-set on Linux-based software platform for Software-Defined Vehicle.