2024 Design, Automation & Test in Europe Conference (DATE 2024)

sLET for distributed aerospace landing system

CHABROL Damien
ASTERIOS TECHNOLOGIES
Massy, France
ORCID: 0000-0001-6794-9593

PHAVORIN Guillaume
ASTERIOS TECHNOLOGIES
Massy, France
ORCID: 0000-0002-1276-9023

JENN Eric
IRT Saint Exupéry
Toulouse, France
ORCID: 0000-0001-9699-3497

***Abstract*— Aerospace systems are more and more distributed, both at equipment level, thanks to the emergence of multicore SoCs, and at system level, where architectures involve an ever-increasing number of nodes interconnected by digital networks. Computations distribution is a means to leverage the available processing power and achieve higher performances. It allows to reduce communication needs, by placing processing closer to actuators and sensors, and makes it possible to comply with various dependability and industrial objectives such as fault tolerance, industrial segregation of responsibilities, etc. This makes integration activities even more critical, due to the interaction complexity between the software components and their deployment on the hardware platform. Therefore, predictability, testability, and ultimately strong determinism are crucial high-level properties needed not only at equipment level but at the whole system scope, which cannot be tackled without changes in the design process. So, new programming and execution paradigms are required. This paper deals with a solution to support the distribution process, based on the sLET paradigm, which is applied to an aerospace landing system over a distributed system architecture.**

Keywords— SW/HW integration, strong determinism, sLET

I. INTRODUCTION

Embedded systems are at the heart of the More Electric Aircraft (MEA) revolution, framed by the European Commission in Flightpath 2050 and targeting greener and cleaner sky. This comes at the cost of rising complexity. Indeed, new upcoming features, like hybrid gas-electric propulsion aircraft and electric taxi, are multi-scale resource-guzzler at system and equipment levels, while subjected to stronger safety, reliability and longevity requirements. As a result, Electric/Electronic architecture must evolve.

Nowadays, using multicore platforms in avionics becomes mandatory. Combined with the improvement of communication networks in terms of throughput and determinism, embedded systems tend to be both more condensed (merging criticality-heterogenous functions on a same processor) and more distributed (to ensure robustness, availability, actuators/sensors closeness). Integration complexity explodes at system and software levels whereas integration is often highlighted as the key pain point of the development process. Indeed, major embedded systems suffer from software system integration issues due to the interaction complexity between the software components and their deployment on the hardware platform [1]. Renew development methods and tools are then indispensable.

Abstraction is the main fundament for designing computing systems: its purpose is to provide high-level design while hiding implementation details. Critical requirements for the system, and in particular temporal ones such as response times, have to be preserved all the way till implementation. Thus, time should be expressed at abstraction level [2]. To account for time in an explicit abstract way, the synchronous Logical Execution Time (sLET) paradigm was introduced in 1990s by the French Atomic Commission as the couple of (1) time-triggered execution and (2) data visibility principle [3]. This provides a high-level hierarchical time abstraction hiding platform details and implementation choices, like scheduling and execution time variability, while offering composability, time determinism and data flow determinism. The sLET paradigm has been successfully applied up to nuclear power plants 1E-classified systems by Framatome since 2020 [4] and more recently by Safran over single and multicore architectures [5]. Proven its worth at equipment level, industrials hope to use it for the entire distributed system. In this paper, we illustrate how to use the sLET paradigm up to a distributed architecture, drawing our inspiration from [6].

This work has been conducted as part of the ARCHEOCS project, with the support of Safran Electronics & Defense and the French National Research Agency (ANR).

II. sLET FOR DISTRIBUTED ARCHITECTURE

sLET is an ideal and mathematically tractable view of time: the platform architecture is abstracted by encompassing physical execution times inside logical time windows. So, the architect does not have to consider the physical time at which computations occur, but the logical time window in which they occur and when input/output streams are available. Thus, sLET has several benefits:

- Hierarchical timing description based on the divide-and-conquer principle during design workflow,
- Early timing verification thanks to a correct-by-construction principle and continuous integration strategy,
- Separation of concerns between the functional design (what) and the implementation choices (how).

Distributed architecture is no exception to the rule: timing properties are independent of the modules allocation on the different nodes, as long as the logical timing constraints expressed at system-level are satisfied by the implementation. Thus, the sLET paradigm can be applied similarly for single-node multicore platforms and multi-node ones. Only the implementation differs.

sLET is then the basis for a whole time-aware development workflow dedicated to highly safety-critical systems. This approach has been applied to a simplified DAL-A certified landing gear control system, previously realized by Safran Landing Systems. It has resulted in the implementation of a workable demonstrator enforcing sLET principles over a distributed architecture. This use-case will serve as an example to illustrate our approach.

A. System sLET design and architecture

Here, we deal only with the braking function, reduced to the COM channel, mapped over two T1040 multicore boards connected by an Ethernet network as depicted in Figure 1.

For readability, we focus on the main cause-effect chain which is to apply a braking command to the wheels while

979-8-3503-4859-0/DATE24/© 2024 EDAA

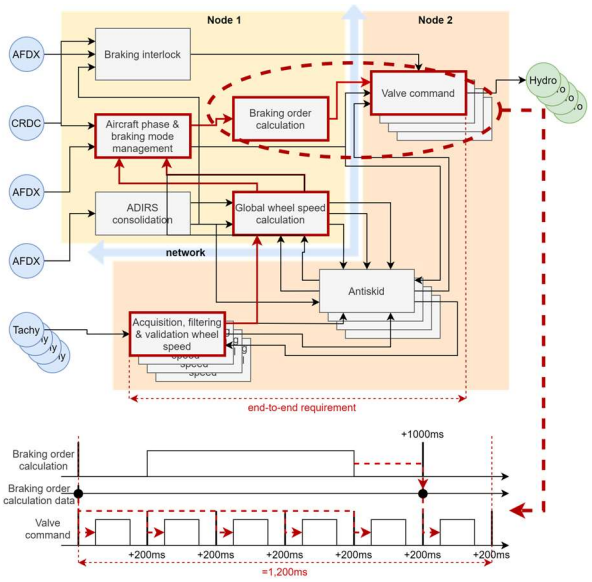

Figure 1: Functional architecture of the landing gear system (TOP) with sLET design of the braking function (BOTTOM).

touching the ground. sLET highlights the functional properties that have to be preserved during integration, like the worst-case response time. Note that sLET distinguishes functions and dataflows temporal properties: a function is triggered on non-necessarily consecutive clock ticks, and can thus be periodic or not, while a dataflow is sampled periodically according to a unique clock. sLET clocks are logical and all derive from the same root clock, thus defining a single timing domain spread over all the system nodes. The mapping between the physical time (e.g. timer ticks) and the logical clock ticks is implementation-dependent and can differ from one node to another.

B. *Network sizing*

For aerospace highly critical systems, the communication network shall ensure deterministic end-to-end latencies. That implies strong intertwined evidence at network and individual node levels, each driving the other. Various combinations of topology and protocol exist to fulfill this requirement. Here we focus on hypotheses coming from the sLET design to help compute a network sizing.

After functions allocation on each node, the distributed dataflows are known. Sample dates and sLET windows specify the instants at which data should become available to consumers. They also determine the latest times at which data can be received. This means that sLET never considers communication latencies to be insignificant. A logical transmission window can be deduced from the previous sLET design according to the dataflow rhythm, as well as the producer and consumer logical timing windows, as depicted in Figure 2. As a result, the producer and consumer sLET windows are never modified, but the durations allowed for functional execution may be shortened to reserve time for data transmission. This synchronization point between functional processing and network transmission is set logically and solved offline after network scheduling.

Knowing the accuracy of the synchronization between nodes, the network access sharing policy can be defined. In the case of the Ethernet network considered for our use case, we consider a TDMA protocol.

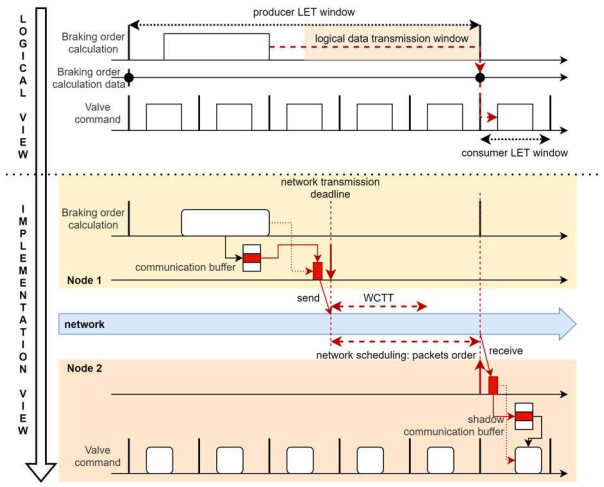

Figure 2: Logical (TOP) and implementation (BOTTOM) views of a distributed sLET communication (plain arrows represent data accesses, dashed arrows depict precedence constraints).

C. *Equipment sLET design and architecture*

At individual node level, the sLET design is extracted from the previous system step. It is then locally enriched in a common development way to fulfill the computed network scheduling, viewed as a contract between the nodes. This refinement can be automated. sLET provides a formalization allowing to specify predecessor/successor constraints between two functional blocks and their dedicated sLET intervals. sLET implementation is built over lock-free/wait-free ring buffers. The distributed sLET implementation manages ring buffer replica on all nodes according to a data-centric consistency model, ensuring that each dataflow is up to date at each associated clock instants, as depicted in Figure 2.

In summary, sLET programs can natively be distributed across multiple machines, making sLET a natural, temporally robust placeholder not just for program execution but also for program communication. By this way, a top-down, iterative and consistent time-aware development process can be built all the way from system level. At deployment level, an adequate implementation allows to ensure sLET properties (strong determinism) at execution thanks to synchronized logical/physical clocks and remote and consistent ring buffers. The braking system use case, briefly presented here, has validated the sLET benefits for an industrial usage.

REFERENCES

[1] Boydston, A., & Feiler, P.H. “Architecture Centric Virtual Integration Process (ACVIP) : A Key Component of the DoD Digital Engineering Strategy”, 2019.

[2] Edward A. Lee, “Computing Needs Time”, Technical Report No. UCB/EECS-2009-30. University of Berkeley, 2009.

[3] F. Siron, D. Potop-Butucaru, R. de Simone, D. Chabrol, and A. Methni, “The synchronous Logical Execution Time paradigm,” 2022.

[4] V. David, C. Aussaguès, S. Louise, P. Hilsenkopf, B. Ortolo, and C. Hessler, “The OASIS based qualified display system,” in Proceedings ANS (NPIC&HMIT 2004), 2004.

[5] A. Methni, E. Ohayon, F. Thurieau. « ASTERIOS Checker : A Verification Tool for Certifying Airborne Software”. 10th European Congress on Embedded Real Time Systems (ERTS 2020), Jan 2020, Toulouse, France

[6] D. Chabrol. “Study, design and realization of a fault-tolerant and predictable synchronous communication protocol on off-the-shelf components”, PhD Thesis, 2006