Dynamic criticality management with ARTEMIS

Olivier CROS
LIGM / Université Paris-Est
Bat Copernic - 5, bd Descartes
77454 Champs sur Marne, France
olivier.cros0@gmail.com

Geoffrey EHRMANN
LACSC
37 quai de Grenelle
75015 Paris, France
geoffrey.ehrmann@gmail.com

Laurent GEORGE
LIGM / Université Paris-Est, ESIEE
2 Boulevard Blaise Pascal
93162 Noisy-le-Grand, France
lgeorge@ieee.org

***Abstract*—In this work, we propose to detail the mixed-criticality integration inside our network simulator ARTEMIS. The objective here is to propose a solution to manage and simulate concrete criticality level changes inside network infrastructures, in order to focus on a network topology reconfiguration w.r.t to critical and non-critical messages evolutions. Through a transmission time computation model based on a probabilistic approach, we propose a solution to generate flowsets integrating mixed-criticality, in order to simulate the scheduling of these flowsets through different topologies.**

I. Introduction

A. About real-time simulation

In strongly constrained industrial domains like spacecraft, public transports or aircraft, reliability and performances are two fundamental objectives which imply defining strict timeliness constraints to prevent system failures. It seems obvious for everyone to imagine the huge human disaster represented by an aircraft network crash at landing, not to mention the financial impact of such events.

As a matter of fact, new protocols and architectures in real-time networks must be certified before being deployed on industrial structures. These protocols have to be analyzed, verified and tested to be proved reliable and safe to be implemented. But operating the tests directly on real physical systems can appear to be very costly. As a conclusion, these real tests should done when most of the protocol has already been validated. That is why, in order to prepare and run performances and reliability tests replacing some of the physical tests, we need to define simulation tools.

For the real-time network simulation, we propose a dedicated network simulator. This simulator is called Another Real-Time Engine for Message-Issued Simulation (ARTEMIS).

B. Related work

ARTEMIS has already been presented in [1], [2] as an open-source user-oriented real-time network simulation tool. Its architecture was similar to real-time multicores and multiprocessors schedulers architectures like Cheddar [3] (a modular framework for schedulability analysis), SchedMCore [4] (toolbox for multicore simulation), and SimSo [5] (an open-source tool designed for multiprocessor context).

There also exists different network simulators, which are more oriented to industrial context. We can mention NS [6] for global network simulation or OmNET++ [7] for dimensioning and performances purposes. Concerning Real-Time (RT) networks architectures, there also exists different simulators to observe and manage specific network architectures : CANoe [8] for Controller Area Network (CAN) or the work presented in [9] for Avionics Full DupleX switched ethernet (AFDX).

ARTEMIS is a RT network simulation tool, providing schedulability analysis for network topologies. Based on a generic component-oriented model (see [1]), the purpose of ARTEMIS is to propose the integration of mixed criticality constraints inside network topologies. This integration was partially detailed in [2] but the mixed criticality management model presented was rather incomplete. In this work, we propose to detail a more dynamic and configurable mixed criticality model integration inside ARTEMIS, and we detail the technical solutions we have done to represent and manage mixed criticality inside a simulation environment.

C. Contributions

The architecture of ARTEMIS (described in [10]), is organized around a set of external modules (grapher, generators) based on a scheduling simulation core. Based on the work presented in [10], we integrated in ARTEMIS core two main models for mixed criticality management. First, we designed centralized and decentralized criticality management to store and share the criticality level among all the nodes of a topology. Then, in order for the core to act independently and to simulate criticality change events scenarios (not just depending on user actions), we designed a new message generation model inside the core. That is what we detail below in III and IV.

The integration of these new protocols implied to change a part of the generation and scheduling model of ARTEMIS, dedicated to the criticality management inside each node. The modular architecture of ARTEMIS allowed us to design a dedicated part for criticality management, without requiring to modify the input or output data formalization. We also reinforced the design and conception fundamentals by improving the Graphical User Interface (GUI) for a better user experience. We added new functionalities for web-oriented and distributed context, to make ARTEMIS a sharable tool designed to be installed on public web servers. We detail this in II.

We propose to test different potential schedulability analysis results and to evaluate the impact of mixed criticality integration inside different network topologies. This is showed through different simulations, detailed in V. We now describe the different improvements inside ARTEMIS global architecture.

II. Web-oriented architecture

A. Global architecture

ARTEMIS' Graphical User Interface (GUI) is the link between users and simulator's kernel. It has to interpret messages between users and kernel in order to make possible the communication between these two entities. To build a simulation, users must configure a network through the GUI, then the interface sends data to the kernel as XML files. The kernel runs the simulation and returns XML files containing simulation results. Once XML files are parsed, GUI displays results as XML logs or graphs.

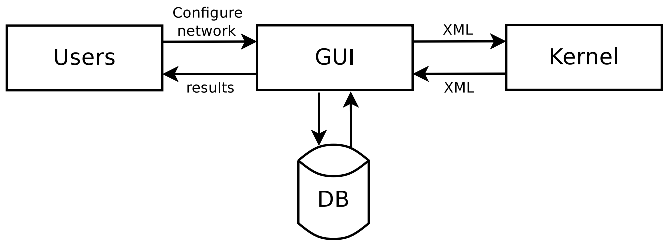

Fig. 1. ARTEMIS architecture

To configure a simulation, users have first to create a topology by using the topology generator or create it manually. Then users have to define all the components of this network, namely the nodes and the links. A node is defined by a name, a scheduling policy, an automatic generated network address and a transmission rate.

Once the topology is created, users can create messages which will define the network behaviour. Users can create a new message manually by configuring the message path, the Worst-Case Transmission Time (WCTT), the period and the offset. The messages creation is more detailed in [2].

Each parameter related to a simulation is saved in a MySQL database. When users click on "Run", GUI generates the XML files after getting data from the database. There are 4 XML files to run a simulation:

- "network.xml" that contains the network topology. It contains a list of nodes with all their attributes (ID and name), and the other nodes to which they are linked to.
- "config.xml" contains the whole configuration of the network, namely the simulation time, the latency and the mixed-criticality management model.
- "graphconfig.xml" contains the name of the graph and the parameters related to graph management.
- "messages.xml" contains the messages to be sent. It lists all the messages and their attributes that we defined previously.

These files are sent to the kernel, which will perform the simulation before returning XML files with graph results, GUI displays these results as a scheduling graph.

B. Web distribution

ARTEMIS is a web-oriented tool. This choice has been made in order to make it easy to install and to use. Using web interface makes it independent from any operating system, which allows us to spread the tool to a large public. The main purpose of the GUI is to be as intuitive as possible. As a matter of fact, ARTEMIS is designed for everyone, which includes non-developers or students who need to be guided, so interface has been designed to be ergonomic, fluid and clear for users; web programming allows a fluid and clear utilization of the tool. Thanks to AJAX architecture, the system answers quickly to users commands, which make the navigation comfortable, and the CSS language enables us to make a clean and sleek visual.

C. Exporting results

In order to improve user experience of the tool, ARTEMIS now integrates a simulation manager. It adds to ARTEMIS a bunch of new functions to manage simulations. Each user can now create its own simulations, export or import them to different platforms in order to increase the reusability of the different simulation configurations built.

The export function produces a ZIP archive containing the input XML files required to build the simulation. Every archive exported by ARTEMIS can also be imported. Importing a simulation triggers the creation of a new simulation and the automatic configuration of it, by using data from the selected ZIP archive. All informations are saved in the database. Then the simulation is ready to be run.

These functions are essential for ARTEMIS. It makes the tool portable and user-oriented by allowing sharing and communication between users and simulation contexts. These new functions allows a user to create dedicated topologies of variable sizes and to propose different messages sets configurations in order to operate benchmarking and performances comparisons on different contexts.

The simulation identification in ARTEMIS is based on session identification and unique identifier association for each simulation configuration : each simulation is unique, and belongs to a specific user. It allows us to improve the portability of ARTEMIS architecture, specially in contexts designed for multi-user utilization, which were the fundamental goal of the web architecture of ARTEMIS.

III. Flowset generator

In order to be able to simulate concrete network scheduling scenarios through ARTEMIS, we define a flowset generator connected to the kernel. Currently, RT simulators propose tasksets generators based on the UUnifast algorithm [11] to build the different tasksets needed for scheduling analyses.

In our work, we adapted current taskset generation algorithms to network context. The purpose of ARTEMIS is to propose scheduling scenarios integrating mixed-criticality in real-time networks. We adapted the current models to generate flowsets mixing messages of different criticality levels. That is the point we propose to detail in the following section.

A. *UUnifast for network context*

Basically, UUnifast [11] is a taskset generation algorithm. Its purpose is to generate a set of n periodic or sporadic tasks, associated to a global load l. Each task of the generated set is characterized by two properties : a Worst-Case Execution Time (WCET), and a period (or minimum inter-arrival time, in the case of a sporadic task). For each task τ_i, we note C_i its WCET and T_i its period. We define the maximum duration T_{max} of the period, based on the duration of the simulation. The generation process is based on 4 different steps :

- First, we generate a random value r_i, based on a uniform law $\mathcal{U}$, with the following expression :

  $$r_i = \mathcal{U}(log(T_{min}), log(T_{max} + T_g))$$

  We assume that all the generated values of r_i are in the interval $[log(T_{min}), log(T_g + T_{max})]$. More details on the computation of r_i were given in [11].
- Then, we compute the period T_i of the task τ_i. This value of T_i is indexed on r_i, according to the expression :

  $$T_i = \left\lfloor \frac{e^{r_i}}{T_g} \right\rfloor * T_g$$

  This bounds the generated period with the value T_{max}. This expression is based on the time granularity of the system, noted as T_g.
- Next, based on a uniform law $\mathcal{U}$ of support 0 and 1, we generate a random utilization u_i for the task. This utilization represents the individual load of the task. It is computed by the cumulative activation of jobs from the task during the time interval $[0; T_{max}]$.
- Finally, we compute the WCET C_i of the task, by computing $C_i = u_i * T_i$.

At the end of the generation of each task, we compare the value of the targetted load l and the value of cumulative utilizations $u = \sum_{k=1}^{n}(u_i)$. If we have $u = l$, the taskset is characterized as correct. In the other case, we discard the taskset and generate a new one.

The problem represented by this method concerns the discarding process. As a matter of fact, this discarding process tends to increase the number of generation loops to run and, as a result, to increase the generation time needed by the algorithm. Inside ARTEMIS, we propose to modify this process in order to reduce the number of discarded tasksets.

Discarding a taskset comes from the point that the cumulated utilization u tends to exceed or lower the value of l. The computation of each value of u_i is based on a uniform law $\mathcal{U}(0, 1)$. As a matter of fact, the generation process tends to generate flowsets with cumulated utilizations which are out of bounds, implying to discard the generated flowset.

The solution we propose was to bound the uniform law $\mathcal{U}$ in order for the global generated utilization u to be centered around the value of l.

As we target a global load value of l for a flowset of size n, we generate utilizations which have an average load equal to $\frac{l}{n}$. Thus, the law $\mathcal{U}$ is characterized by a specific variance v which can be modified to adjust the results of the generating algorithm. The higher the value of v, the more heterogeneous the generated flowset in terms of utilizations, but the more we tend to increase the number of discarded tasksets. We can vary the value of v depending on the accuracy and heterogeneity we target in the generated taskset. For basic simulations, v is set between 0.05 and 0.06 in ARTEMIS core.

B. *Load computation*

In multicore and multiprocessor RT scheduling analyzers, the generation of a taskset is based on a targeted global utilization represented by the taskset. Depending on the network architecture, a generated taskset utilization can exceed 1 but the utilization on each node is less than 1 .

In network context, this global utilization has been replaced by the global load l. One naive approach would be to establish a strict parallel between the global utilization of a taskset and the global load l. In fact, as there is not message transmission or paths computation in processor context, the individual utilization $\frac{C_i}{T_i}$ of a task τ_i represents its direct impact on the system in terms of utilization. On the opposite, the individual load $\frac{C_i}{T_i}$ of a flow v_i in a network is not its direct impact : as each message from the flow will be transmitted once in each switch of its path, the real impact of one flow in terms of traffic depends on its path.

C. *Mixed-criticality integration*

The presented flowset generator generates flows of different criticality levels. It implies defining two different constraints: first, we have to clearly define a protocol to decide which message belongs to which criticality level. Then, for each message, we have to precise the WCTT of the message for each criticality level it belongs to. We detail these two steps below.

In order to decide which message belongs to which criticality level, we first based our work on the assumption made in [12] for the WCTT of a message sent in different criticality modes. If we suppose k different criticality levels $\gamma_1, ..., \gamma_{k-1}, \gamma_k$, we assume for n flows that :

$$\forall i \in [1; n], \forall q, r \in [1; k], q < r \implies C_i^{\gamma_q} \leq C_i^{\gamma_r} \qquad (1)$$

This hypothesis corresponds to the case where increasing the criticality level of a flow leads to send more information. If the maximum criticality level of a flow v_i is γ_q, all the WCTT of v_i will be lower or equal to $C_i^{\gamma_q}$. Given this hypothesis on the different criticality levels of a network, we defined a criticality rate C_{rate} in the network. For each flow generation, we compute a probability p_{rate} included in $[0; 1]$. Once computed, for each criticality level γ_q, we check if $p_{rate} < (C_{rate})^{q-1}$. If that is the case, we generate a dedicated value for $C_i^{\gamma_q}$. If not, we set $C_i^{\gamma_q} = -1$. This hierarchical structure is convenient to mixed-criticality models as it was presented in [13].

IV. Mixed-criticality models

A. Transmission time computation models

Integrating mixed-criticality in ARTEMIS means that the different virtualized topologies created through the tool must be able to manage criticality levels and criticality level switches. This implies to be able to trigger specific events occuring in a criticality level switch. According to previous works on mixed-criticality in networks [10], we assume that two different events can trigger a criticality switch in ARTEMIS :

- Either the user statically designed a criticality switch at a specific time. In that case, the user specifies the level to switch to and the time at which it occurs. It is called the static model.
- Either a message exceeds its WCTT at a specific level according to a configurable law. If we suppose a flow v_i composed of two WCTT C_i^{LO}, C_i^{HI} for two Low (LO) and High (HI) criticality levels, this event corresponds to a time where a message from v_i exceeded the WCTT C_i^{LO}. In that case, necessarily, it implies that v_i has to be considered as occuring a criticality switch to HI

The first model was introduced in ARTEMIS and has already been discussed in [2] . In order for ARTEMIS core to be able to manage the second case (low-critical level WCTT exceeding), it means that the messages generator model has to be able to generate messages exceeding their low-critical WCTT. We propose to detail here the modifications we add to integrate inside ARTEMIS message generator in order to take into account this new generation model.

Inside ARTEMIS core, we defined several $\gamma_1, ..., \gamma_{k-1}, \gamma_k$ criticality levels (minimum 1). Each flow v_i is designed with a specific WCTT $C_i^{\gamma_j}$ for each criticality level j. In the case where the flow does not belong to any criticality level except the lowest one γ_1 (non-critical level), we note $\forall j > 1, C_i^{\gamma_j} = -1$. As a matter of fact, each flow v_i is defined with a set of WCTT noted as $C_i^{\gamma_1}, ..., C_i^{\gamma_{k-1}}, C_i^{\gamma_k}$.

In order to generate potential criticality switch triggering events, generating a message from a flow v_i implies to generate not only a specific message transmission time between the message Best Transmission Time (BTT) and $C_i^{\gamma_1}$, but a message transmission time which is included between the message BTT and its highest WCTT (attached to the highest criticality level the flow v_i belongs to). In order to integrate this mixed criticality model, we integrated inside ARTEMIS different probabilistic models to generate messages of different transmission times, each transmission time associated to a specific criticality level.

We have to keep in mind that basically, ARTEMIS has been designed for worst case analysis. Hence, for each generated transmission time of a message, we round it to the closest highest corresponding WCTT w.r.t. a criticality level. This model allows us to keep a worst-case evaluation of delays in the end-to-end transmission delay computation of the different flows in the network.

1) Linear model: The linear model proposes to generate a transmission time which value is based on a linear probability law. The generated time is computed from two bounds : the flow BTT (noted as B_i) and the message highest WCTT, belonging to a criticality level γ_j (notes as $C_i^{\gamma_j}$). The probability of generating a specific transmission time t is estimated as follows :

$$\mathcal{P}(t) = \begin{cases} 0 \quad \text{if } \ t \leq B_i \\ \frac{T_g}{C_i^{\gamma_j} - B_i} \quad \text{if } \ B_i \leq t \leq C_i^{\gamma_j} \\ 0 \quad \text{if } \ t > C_i^{\gamma_j} \end{cases}$$

2) Strict model: The strict model is based on the assumption that a message transmission time is necessarily equal to one of its WCTT. As a matter of fact, the strict model consists in picking one transmission time among all potential values in $C_i^1, ..., C_i^{j-1}, C_i^j.$,

If we suppose that the flow v_i belongs to $\gamma_1, ..., \gamma_{j-1}, \gamma_j$ criticality levels, we can express its probability model as :

$$\mathcal{P}(t) = \begin{cases} \frac{1}{j} \quad \text{if } \ t = C_i^u, u \in [1; j] \\ 0 \quad \text{if } \ not \end{cases}$$

3) Gaussian-based models: In this model, we define a uniform law $\mathcal{U}$ which is used as a base to compute each transmission time of each message. This uniform law is defined by two parameters : its center c and its deviation d. We note the expression as $\mathcal{U}(c, d)$. In order to define different transmission time computation models, we can adjust both values of c and d. Their role is described as follows :

- The lower the value c, the higher the probability for the transmission time of a message from flow v_i to be equal or close to its BTT. On the contrary, the higher the value of c, the higher the probability to have a generated transmission time close to the highest WCTT of v_i.
- The deviation is used to define the probability of a generated transmission time to be far from c. The higher the deviation, the more heterogeneous the successive generated transmission times.

A complete basic uniform law can generate WCTT which are beyond the bounds B_i and $C_i^{\gamma_j}$. To avoid this, we integrate bounds inside the generator, implying to regenerate a transmission time if the previous one was not between the bounds. This allows us to propose reliable generation models which will not generate out of bounds transmission times or non coherent flowsets.

B. Mixed-criticality switches management

In order to be compliant with Mixed-Criticality (MC) management models proposed in [10], we integrated protocols to manage MC inside switched Ethernet networks. Based on our previous work [10], we integrated first the centralized approach that relies on a global clock synchronization. The purpose of this centralized protocol is to guarantee, through a reliable multicast (implemented in ARTEMIS core), the consistancy of the criticality level of the network in all the nodes at any time. We integrated the centralized MC management protocol in ARTEMIS taking into account the network clock accuracy provided by a clock synchronization protocol.

The centralized protocol implies to switch the criticality level to high levels in nodes even if they do not transmit high-critical flows. This induces a loss of non-critical traffic transmissions. In order to answer to this problem, we also integrated an alternative protocol inside ARTEMIS, based on a distributed and independant MC management protocol among nodes, called the decentralized protocol. This approach does not require a global clock synchronizatio protocol.

In ARTEMIS, we integrated the potential to manage these two modes. The first mode (centralized) was the fundamental one and has already been discussed in [2]. The decentralized MC management imposed to split the criticality management from the global core time management.

We integrated inside ARTEMIS CoreScheduler a new module responsible for criticality management : the CriticalityManager. This module allows us to manage a criticality level table (for centralized protocol) and an independant criticality level value for each node (for decentralized protocol). The CoreScheduler is, among all, responsible for critical switches events and criticality table integration. These concepts were detailed in [2].

The CriticalityManager's role is to store all the different criticality switches and WCTT overuns in the network, in order to associate them with corresponding criticality level switches, either locally in a node (decentralized approach) or in the global topology (centralized approach). Its architecture is detailed in figure

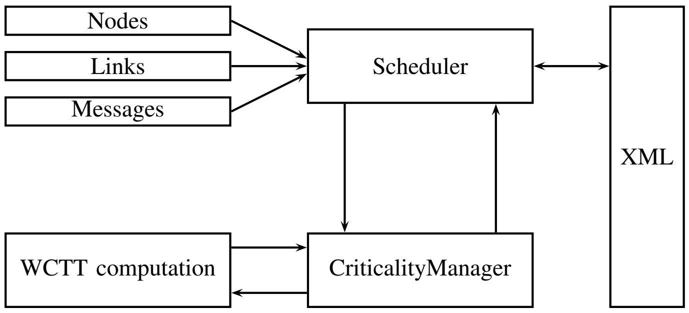

Fig. 2. CriticalityManager architecture in ARTEMIS core

At any time, each node controls if the CriticalityManager triggered a specific criticality switch for this node at a given time. As a matter of fact, the CriticalityManager is a criticality switch engine allowing the criticality level of each node to stay consistant and reliable at any moment of the simulation.

V. Simulation Results

In order to illustrate the dynamic detection of mixed criticality switches, we defined a simple topology composed of 4 switches and a set of end-systems,. We defined also a set of flows. In the different simulation environments detailed below, we defined a topology and a set of flows as described in figure 3.

The different flows parameters are detailed in the table below. We ran the simulations for a dual (LO, HI) criticality level network. We ran a first set of simulations to compare the different linear, strict and gaussian models detailed in III.

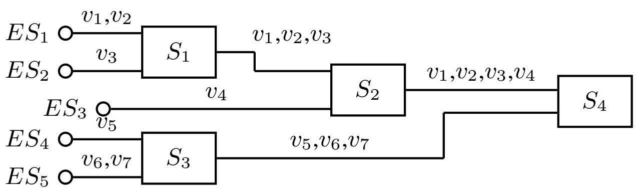

Fig. 3. Simulation environment (topology and flows)

v_i	T_i	C_i^{LO}	C_i^{HI}
v_1	80	5	8
v_2	50	4	-
v_3	80	4	8
v_4	60	3	-
v_5	70	4	7
v_6	80	5	-
v_7	50	3	-

First, we ran a simulation in dynamic centralized mode (see figure 4). Given this model, we can observe that a WCTT overun in LO mode was detected for flow v_3 is ES_2 at $t = 82\mu$s. It means that, at this time, the criticality level switches from LO to HI. As a matter of fact, the system detected that the message from ES_2 exceeded its LO-WCTT and sent a criticality switch event to the CriticalityManager. Figure 4 also shows that the transmission time generator is able to generate different transmission times (corresponding to different WCTT) for the same flow.

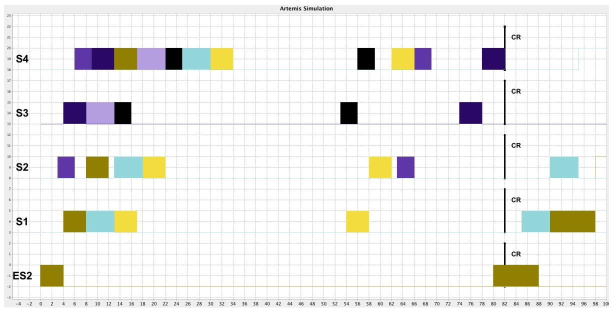

Fig. 4. Simulation in ARTEMIS with dynamic centralized mode

This first simulation is an implementation and proof of concept of MC switches management inside ARTEMIS core. This simulation shows the reliability of the centralized model and that, when a criticality switches happens, the consistancy of the criticality level in all nodes is maintained by the CriticalityManager.

In order to illustrate the decentralized MC management mode, we ran another simulation with the same parameters, but with decentralized protocol. We obtained the results showed in figure 5. We observe in this figure that each node in the path of v_1 detects its LO-WCTT overun. These detections occurs at different times on the different S_1, S_2, S_4 switches, respectively at $t = 1\mu$s, $t = 9\mu$s and $t = 17\mu$s, indexed on the simulation parameters.

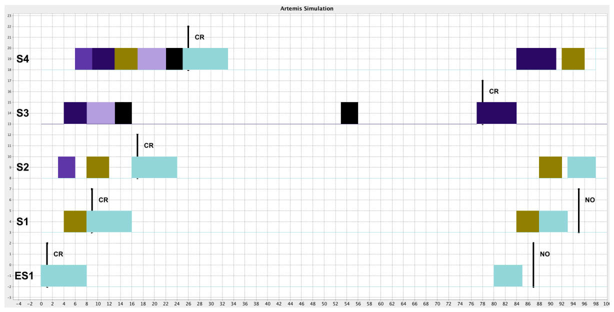

Fig. 5. Simulation in ARTEMIS with dynamic decentralized mode

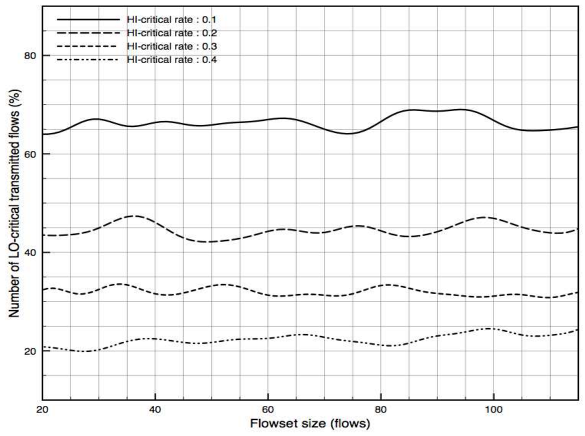

Fig. 6. LO-critical messages transmitted during HI mode

The delay to wait before switching back to LO mode is automatically set to the longest period of all flows in the topology. If no message in HI mode is received by a node during taht period, a node switch back to LO mode. We observe that the node S_1 switches back to LO-mode at $t = 96\mu$s (80μs after the end of transmission of the last HI-critical message).

As it was predicted, the decentralized network allows nodes which are not currently transmitting HI traffic to stay in LO mode. This allows us to transmit a higher amount of LO-critical traffic. In order to illustrate the number of LO critical messages which can be transmitted during both centralized and decentralized approaches, we ran a set of simulations computing the number of correctly transmitted LO messages depending on the amount of HI messages in the network.

We generated 40 different random flowsets, for a number of flowset ranging from 20 to 115. This allows to cover a various set of possible network traffic modelizations. In order to illustrate the impact of HI-critical flows, we generated these simulations for different LO to HI ratios (equal to the number of LO flows devided by the number of HI flows) ranging from 0.1 to 0.4. The results shown in figure 6 shows that, during HI-critical phases, we can assure the transmission of LO-critical messages from 20 % to 70 %, which represents a clear gain in terms of Quality Of Service (QoS).

As a conclusion, we can observe that the criticality level switches impacts the network behavior in terms of QoS, but both presented protocols assure the reliability of HI-critical messages transmission

VI. Conclusion and Perspectives

In this paper, we showed the new mixed criticality management models integrated inside the last version of ARTEMIS. Integrating different criticality protocols allows us to integrate non-critical traffic management in the mixed-criticality simulations. The integration of centralized and distributed mixed criticality switch models allows us to propose a wide range of simulation contexts. This integration makes ARTEMIS specifically designed for reliability and performances tests and practices in the domain of real-time networks simulation. ARTEMIS comes with a taskset generation tool supporting mixed criticality in the context of switched Ethernet Networks.

As a further work, we will propose an on-line downloading platform for ARTEMIS.

References

[1] L. G. X. L. Olivier Cros, Frédéric Fauberteau, “Simulating real-time and embedded networks scheduling scenarios with artemis,” in *WATERS’14*, 2014.

[2] L. G. Olivier Cros, “Mixed-criticality management of networked real-time systems with artemis simulator,” in *WATERS’15*, 2015.

[3] L. N. L. M. F. Singhoff, J. Legrand, “Cheddar: a flexible real time scheduling framework,” in *The Special Interest Group on Ada (ACM’s SIGAda) 2004*, 2004.

[4] M. C. Mesonero, “Environnement de développement d’applications multipériodiques sur plateforme multicœur. la boîte à outil schedmcore,” 2012.

[5] M. Chéramy, P.-E. Hladik, and A.-M. Déplanche, “Simso: A simulation tool to evaluate real-time multiprocessor scheduling algorithms,” in *Proc. of the 5th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems*, ser. WATERS, 2014.

[6] D. Mahrenholz and S. Ivanov, “Real-time network emulation with ns-2,” in *Eighth IEEE International Symposium on Distributed Simulation and Real-Time Applications, 2004. DS-RT 2004.*, 2004.

[7] A. Varga, *OMNeT++ user guide*, 2014.

[8] “Automotive can network response time analysis with variable jitter,” in *Mechatronics 2004 : 9th Mechatronics Forum International Conference*, 2004, pp. 785–794.

[9] T. M. Rodrigo Coelho, Mark Szczepanski and G. Fohler, “A web based monitoring tool for afdx networks,” in *WATERS’15*, 2015.

[10] X. L. Olivier Cros, Laurent George, “A protocol for mixed-criticality management in switched ethernet networks,” in *Workshop on Mixed-Criticality in Real Time Systems Symposium, WMC-RTSS’16*, 2016.

[11] R. I. D. Paul Emberson, Roger Stafford, “Techniques for the synthesis of multiprocessor tasksets,” in *Workshop on Analyzing Tools and methodologies for Embedded and Real-Time Systems(WATERS’10)*, 2010.

[12] S. Vestal, “Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance.” in *Real Time Systems Symposium(RTSS’07)*, 2007, pp. 239–243.

[13] A. Burns and R. Davis, *Mixed criticality systems: A review*. Department of Computer Science, University of York, 2013, vol. Tech. Rep.