TTEthernet-based architecture simulation with Ptolemy II

Guillaume Brau, Claire Pagetti

TTEthernet-based architecture simulation with Ptolemy II

Guillaume Brau
ONERA, Toulouse, France
Guillaume.Brau@onera.fr

Claire Pagetti
ONERA, Toulouse, France
Claire.Pagetti@onera.fr

ABSTRACT

This paper focuses on the performance evaluation of real-time embedded architectures. Recent embedded systems mostly rely on large distributed platforms composed of sensors, calculators and actuators communicating through a shared network. In such systems, each component comes with its own mechanisms to guarantee systems integrity. In order to accurately observe behaviors and evaluate architecture properties, simulation is an efficient method if it takes into account all these mechanisms. In this work, we illustrate a modelling and simulation strategy, based on the Ptolemy II framework, to study temporal phenomena of an embedded platform. We apply the methodology on an avionics case study executed on a distributed architecture communicating via a Time-Triggered Ethernet (TTEthernet) network.

1. INTRODUCTION

In order to ensure real-time behavior of distributed systems, both computation and communication have to be temporally predictable, in the sense that worst-case times must be computable. In order to achieve this goal, the platforms rely on specific real-time mechanisms. For example, in the civil aeronautics domain [1], IMA (Integrated Modular Avionics) architectures are composed of ARINC653 modules using an AFDX (Avionics Full-DupleX switched Ethernet) network. In aerospace and military domains [2], calculators provide a fixed-priority scheduling policy in order to share a common processor between tasks. In that case, communications are supported by SpaceWire or Mil-1553 B buses.

Numerous works are devoted to the analysis of temporal behaviors of distributed systems. Several techniques have been developed to compute worst-case response times on calculators, worst-case traversal times on networks and worst-case latency times to combine them. In addition to these worst case approaches, simulation is an efficient method to accurately observe a representative set of the behaviors.

1.1 Contributions and objectives

In this work, in order to evaluate an embedded architecture, we have defined a library of components extending a basic modelling and simulation framework. We illustrate the procedure onto an avionics case study allocated to a real-time architecture composed of fixed-priority scheduled calculators which communicate through a TTEthernet network mixing Time-Triggered and Event-Triggered traffics [3].

The modelling and simulation framework, Ptolemy II [1] [4], enables to model multi-domains systems such as electrics, physics, or software. Thanks to an actor oriented implementation based on Java, Ptolemy proposes a graphic framework intended to create and simulate models. By using basic actors provided with the framework, we are able to precisely model specific components of any embedded platform. Then, by combining them appropriately, we build a global representative model of the studied platform that we, finally, simulate. Through a specific actor called the *director*, the model is processed by an associated *model of computation* – for our work, we mainly use *Discrete Events* (DE) and *Finite-State Machines* (FSM) paradigms. In the discrete case, the model of computation manages specific signals, called *events*, transmitted from actor to actor. Therefore, during the simulation, we can use a various range of *sinks*, set on different points of the model, to observe events properties : the value and the timestamp. We can observe several temporal properties : (1) from delays between input and output ports, we can focus on the response time of calculators. We can (2) measure the elapsed time at commutator level to highlight a specific commutator traversal time, or (3) observe events between several successive commutators in order to quantify the end-to-end latency of a specific path. Furthermore, we can obtain the distribution of relevant measurements.

In this paper, we only illustrate one property : the end-to-end latency along a functional chain [5]. A functional chain $C : \xrightarrow{a_0} \tau_1 \xrightarrow{a_1} \dots \tau_n \xrightarrow{a_n}$ is a succession of treatments realized by a set of tasks possibly mapped onto different modules (*i.e.* calculators), communicating through the shared network. Each task τ_i receives a stimulus a_{i-1} and produces a response a_i sent to the next one in order to compute the final result a_n. The latency $\mathcal{L}_\mathcal{C}$, associated to the functional chain, is the series of execution times and traversal times required to provide the output a_n as the result of the input a_0. We simulate the case study and act on influencing parameters in order to observe upper and lower

[1] available at `http://ptolemy.eecs.berkeley.edu/`

latency bounds. Finally, we try to find an optimal configuration (*i.e.* we seek to minimize latency bounds) of the TTEthernet-based architecture.

1.2 Related work

The (real-time) embedded systems performance evaluation includes a large amount of modelling and simulation granularity. About embedded architectures study, [6] deals with Integrated Modular Avionics. It details four levels of modelling dedicated for different performance evaluation needs, simulated using SES/Workbench. [7] targets *operational architecture* evaluation and describes a case study issued from automotive. It simulates the response time along a logical chain of a kind of architecture composed of Electronic Control Units communicating through CAN and VAN buses. Presently, there are few published papers which deal with the evaluation of TTEthernet technologies – and even less referring to the evaluation of complete TTEthernet-based architectures. We can cite two complementary approaches which aim to evaluate network traversal times. The first, detailed in [8], is based on an extension of the OMNeT++ INET framework intended to model and then simulate time-triggered switches and end-systems. Regarding hardware measurements, [9] illustrates a mixed hardware-software solution using a Real-Time Linux system.

The remainder of the paper is structured as follows. Section 2 details the case study and its allocation on the platform. A reminder of fixed-priority scheduling and TTEthernet technologies is given. Section 3 gives an overview of basic components modelling while the section 4 deals with simulations and latency experiments. We finally conclude in the section 5.

2. CASE STUDY

The avionics case study is a part of an aircraft's Flight Management System (FMS). This system interacts with the crew and deals with static and dynamic information about the flight plan (*i.e.* the predefined path between departure and arrival points) : location, distance and estimated time of arrival.

2.1 System description

The system, composed of four main tasks, is depicted in Figure 1. The *Keyboard and cursor control Unit (KU)* reads data sent by the pilot (or copilot) through keyboards while the *Multi Functional Display (MFD)* refreshes displays consecutively. From the KU task, crew requests are forwarded to the *Flight Manager (FM)* task which computes the response about the flight plan and returns it to the MFD. For this, it requests static data to the *Navigation Data Base (NDB)* task and also relies on dynamic data from sensors. The communication flow from *KU* to *MFD* via *FM* and *NDB* forms the functional chain C :

$$\xrightarrow{req} KU \xrightarrow{wpId} FM \xrightarrow{query} NDB \xrightarrow{answer} FM \xrightarrow{wpInfo} MFD \xrightarrow{disp}$$

$\mathcal{L}_C$ is the latency related to this functional chain.

2.2 Allocation

The platform is composed of a set of calculators, hosting an online preemptive Fixed-Priority (FP) scheduler, and a Time-Triggered Ethernet (TTE) network.

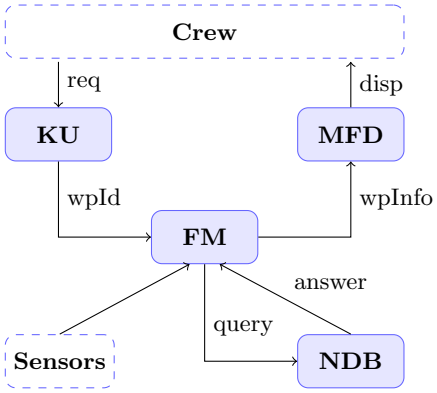

Figure 1: Case study

Calculators.

Tasks are statically mapped onto calculators. The task set may be described by $\Sigma = \{\tau_1, \ldots, \tau_m\}$ where a task τ_i is specified by a quadruplet (O_i, T_i, C_i, p_i) : O_i is the offset, T_i the period, C_i the execution time and p_i the priority. The FP scheduler executes the tasks according to the priorities. We assume for any task τ_i :

- $C_i \in [BCET_i, WCET_i]$, the execution time is bounded between the *Best-Case Execution Time* (BCET) and the *Worst-Case Execution Time* (WCET) ;
- the inputs are consumed at the beginning of the task and the outputs produced no later than the end ; a task may produce several outputs during its execution.

The table 1 summarizes the allocation on three modules. *AutoTest* tasks are modules background tasks.

module	task	O (ms)	T (ms)	$WCET$ (ms)	p
M_1	KU	0	50	25	1
M_1	MFD	25	50	25	2
M_2	FM	0	60	30	1
M_2	$FMAutoTest$	0	120	15	2
M_3	NDB	0	100	20	1
M_3	$NDBAutoTest$	0	200	50	2

Table 1: Calculators parameters

Communication network.

In this paper, we focus on a real-time Ethernet solution, called TTEthernet (for *Time-Triggered Ethernet*), provided by TTTech. This protocol is based on a time-triggered implementation where the network entities (*i.e.* end-systems and commutators) are synchronized [3, 10]. TTEthernet supports two classes of messages :

- Time-Triggered (TT) messages occur at known points of time and are treated with highest priority. Predefined slots of time are allocated for each source of TT messages in order to ensure small transmission delays.
- Event-Triggered (ET) messages could occur at any point of time and compete for network access. Event-triggered traffics have a lower priority than TT messages. In our case, ET messages are AFDX-like [11].

Both Time-Triggered and Event-Triggered traffics must be deterministic. For this, each data flow source has a limited bandwidth. TT link allocations consider T the period and n, the number of frames allowed per slot. AFDX "virtual links" are described by a *bag* which is the minimum time required between two frames sending. In both case, s_{max} is the maximal frame size. The allocation is depicted in the table 2.

link	source	destination(s)	variables(s)	T (ms)	n	s_{max} (bits)
TT_7	NDB	FM	$answer$	120	2	4000
link	source	destination(s)	variables(s)	bag (ms)		s_{max} (bits)
VL_1	KU	FM	$wpId$	32		600
VL_2	FM	MFD	$wpInfo$	8		5000
VL_3	FM	NDB	$query$	16		1000

Table 2: Communication links parameters

3. MODELLING AND SIMULATION

FP scheduling.

A fixed-priority scheduling can be simulated using a specific Ptolemy director. The TM director (for *Timed Multitasking*) [12] enables to compute concurrent components. The figure 2 illustrates the modelling of the module 2 scheduler. *FM_Clock* and *FM_AutoTest_Clock* define periods of FM and $FMAutoTest$ tasks. To each task is associated a priority and an execution time. Then, scheduler outputs give "tics" of the module execution.

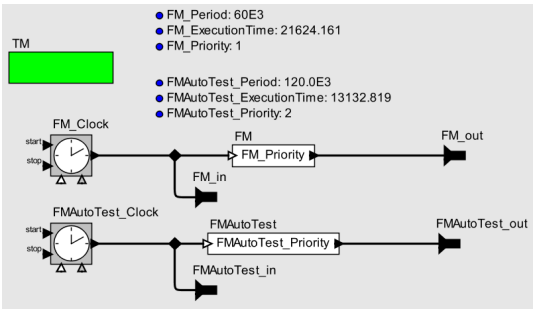

Figure 2: Fixed-priority scheduler modelling

TTEthernet commutators.

Classically, an Ethernet commutator can be modelled with three elements.

- an input queue for the *store-and-forward* mechanism ;
- a switch component subject to an intrinsic delay to simulate the commutation ;
- a set of output queues, associated to each physical output port, which simulate ports transmission delay.

In a specific TTEthernet commutator, an additional component, we call *TTEController*, ensures message scheduling. The model of this component is depicted in Figure 3.

The controller takes care of *in* events, which deal with network frames, and the *out_size* input, linked with the current size of the "physical" output queue. First, the *BooleanSwitch*

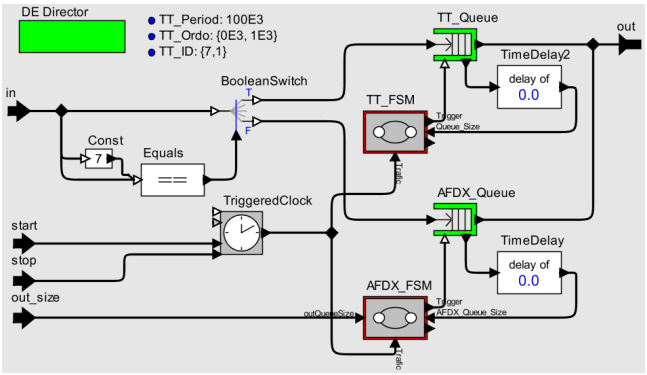

Figure 3: Modelling of a TTEController

determines if the input event is TT-representative (*i.e.* its value equals 7 in our case) or AFDX-representative (*i.e.* all others). According to the quality of service, input events are directed to the *TT_Queue* or the *AFDX_Queue*. The queues are managed thanks to the *Triggered-Clock* which implements the static scheduling. This is the aim of the finite-state machines, *AFDX_FSM* and *TTE_FSM*, to identify the appropriate slots of time.

4. EXPERIMENTS

We have acted on simulation parameters to evaluate the latency fluctuation. Figure 4 graphically translates a sequence simulated with Ptolemy. We can see functional task executions and communications between modules. In this example, the resulting latency is $\mathcal{L}_{\mathcal{C}}$=199,07 ms.

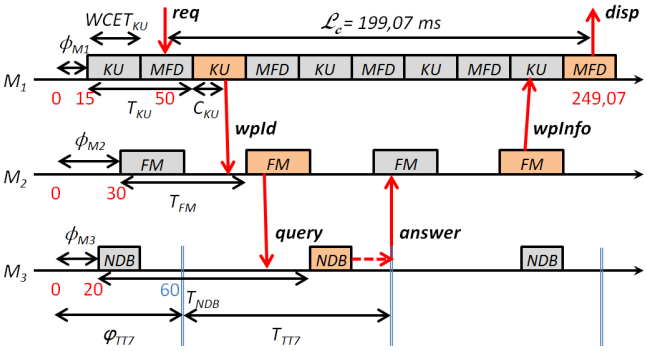

Figure 4: Simulated sequence

We identify several latency influencing parameters :

- the phases of modules ϕ_{M_1}, ϕ_{M_2} and ϕ_{M_3} ;
- the phase of the time-triggered link φ_{TT_7} ;
- the execution times $C_{task} = \{C_{KU}, C_{FM}, C_{NDB}\}$;
- the network load that impacts on traversal times.

Thanks to the exact modelling of the network, we are able to simulate and measure traversal times. Depending on link, traversal times are about a few hundred of microseconds. This relatively small value is due to our restricted case study where the network load is not important enough to influence

traversal times. Thus, it does not impact on the functional chain latency.

For the experiment, we associate to other pending parameters a limited random value in order to simulate a series of scenarios :

- $\phi_{M_1} \in [0, T_{KU}[$, $\phi_{M_2} \in [0, T_{FM}[$, $\phi_{M_3} \in [0, T_{NDB}[$;
- $\varphi_{TT_7} \in [0, T_{NDB}[$;
- $C_{task} \in]0, WCET_{task}]$.

By iterating simulated sequences, we observe a latency fluctuation. In the case where all parameters are randomly allocated, we show that simulated *best-case latency time* and *wort-case latency time* are $BCLT = 76,347ms$ and $WCLT = 497,859ms$.

In order to find an "optimal" synchronization between modules and links , we generate a finite set of random scenarios to approach the lower latency bound. Thereafter, we stop the simulation and quote the required configuration. Figure 5 details simulation results with $\phi_1 = 15ms$, $\phi_2 = 35ms$, $\phi_3 = 30ms$ and $\varphi_7 = 50ms$. By maintaining this synchronization and depending on execution times, the latency is bounded between 125,272 ms and 349,581 ms.

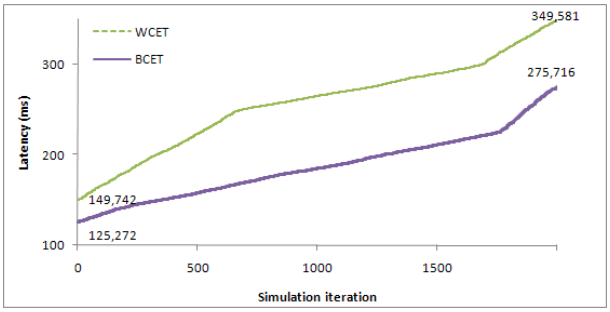

Figure 5: Latency with an optimal parametrization

The improvement of the synchronization between components has reduced the $WCLT$ of about 150 ms (with a slight deterioration of the $BCLT$). These experiments show that the time-triggered approach makes the distributed system "more deterministic". Indeed, synchronization capabilities of the TTEthernet-based architecture enables to act on components phases, making the architecture more stable and predictable.

5. CONCLUSION

In this paper, we have presented a methodology to model and simulate an embedded architecture. We have illustrated the procedure on FP scheduled calculators and a TTEthernet network for communications.

In future work, we will experience the network load influence. For this, we need to increase the case study architecture size. On the one hand, it will enable to evaluate the network behavior and on the other hand it will experience the scalability of our procedure. Moreover, it will be important to compare time-triggered (TTEthernet) and event-triggered (AFDX) approaches and its influence on the functional chain latency.

Last but not least, it would be interesting to confront our results with a formal verification method or/and real measurements.

6. REFERENCES

[1] C. Fraboul, F. Frances, and J.-L. Scharbarg. Embedded networks in civilian aircraft avionics systems. In Bogdan Wilamowski and J. David Irwin, editors, *The industrial electronics handbook: industrial communications systems*, chapter 24, pages 450–480. CRC Press LLC, 2 edition, february 2011.

[2] A. Mifdaoui, F. Frances, and C. Fraboul. Full duplex switched ethernet for next generation "1553b"-based applications. In *Real Time and Embedded Technology and Applications Symposium, 2007. RTAS '07. 13th IEEE*, pages 45 –56, april 2007.

[3] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer. The time-triggered ethernet (tte) design. *8th IEEE International Symposium on Object-oriented Real-time distributed Computing (ISORC), Seattle, Washington*, May. 2005.

[4] J. Eker, J.W. Janneck, E.A. Lee, Jie Liu, Xiaojun Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Yuhong Xiong. Taming heterogeneity - the ptolemy approach. *Proceedings of the IEEE*, 91(1):127 – 144, jan 2003.

[5] M. Lauer, J. Ermont, C. Pagetti, and F. Boniol. Analyzing end-to-end functional delays on an ima platform. In *Proceedings of the 4th international conference on Leveraging applications of formal methods, verification, and validation - Volume Part I*, ISoLA'10, pages 243–257, Berlin, Heidelberg, 2010. Springer-Verlag.

[6] C. Fraboul and F. Martin. Modeling and simulation of integrated modular avionics. In *Parallel and Distributed Processing, 1998. PDP '98. Proceedings of the Sixth Euromicro Workshop on*, pages 102 –110, jan 1998.

[7] P. Castelpietra, Ye-Qiong Song, F. Simonot-Lion, and M. Attia. Analysis and simulation methods for performance evaluation of a multiple networked embedded architecture. *Industrial Electronics, IEEE Transactions on*, 49(6):1251 – 1264, dec 2002.

[8] T. Steinbach, H.D. Kenfack, F. Korf, and T.C. Schmidt. An extension of the omnet++ inet framework for simulating real-time ethernet with high accuracy. In *SIMUTools 2011 – 4th International OMNeT++ Workshop*, New York, USA, March 21-25 2011. ACM DL.

[9] Florian Bartols, Till Steinbach, Franz Korf, and Thomas C. Schmidt. Performance analysis of time-triggered ether-networks using off-the-shelf-components. In *Proceedings of the 2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, ISORCW '11, pages 49–56, 2011.

[10] W. Steiner, G. Bauer, B. Hall, M. Paulitsch, and S. Varadarajan. Ttethernet dataflow concept. In *Network Computing and Applications, 2009. NCA 2009. Eighth IEEE International Symposium on*, pages 319 –322, july 2009.

[11] ARINC 664. *Aircraft Data Network, Part7: Determinist Networks*. 2003.

[12] Jie Liu and E.A. Lee. Timed multitasking for real-time embedded software. *Control Systems, IEEE*, 23(1):65 – 75, feb 2003.