Loosely Time-Triggered Architectures for Cyber-Physical Systems

Albert Benveniste
IRISA/INRIA, Campus de Beaulieu
35042 Rennes cedex, France

***Abstract*—Cyber-Physical Systems require distributed architectures to support safety critical real-time control. Kopetz' Time-Triggered Architectures (TTA) have been proposed as both an architecture and a comprehensive paradigm for systems architecture, for such systems. To relax the strict requirements on synchronization imposed by TTA, Loosely Time-Triggered Architectures (LTTA) have been recently proposed. In LTTA, computation and communication units at all triggered by autonomous, non synchronized, clocks. Communication media act as shared memories between writers and readers and communication is non blocking. In this paper we review the different variants of LTTA and discuss their principles and usage.**[1]

I. Introduction

Embedded electronics for safety critical systems has experienced a drastic move in the last decade, particularly in industrial sectors related to transportation (aeronautics and space, automobile, and trains, trams, or subways).

A. From Federated to Integrated Architectures

In the past, electronics for such systems was limited and mostly relying on analog technology for sensors, actuators, as well as computing devices. So far the paradigm was the following: each different function requires its own set of sensors, actuators, its analog controller, and its dedicated set of wires. This architecture, referred to as *Federated Architecture*, has survived within the first generation of safety critical embedded systems equipped with digital computing systems. Federated Architectures have proved to be safe and robust in that they ensured built-in partitioning between the different functions. Federated Architectures could not be sustained in the late 90's, however, due to the drastic increase in number and complexity of functions and their interdependence. The 80 ECU (Electronic Computing Units) found in some high end automobiles today lead to a dead end. Examples of interdependence are already found in functions such as ESP and engine control in automobiles, not to speak of active safety and X-by-wire functions planed for deployment in future cars. The conclusion is clear: Federated Architectures can no longer be considered as a sustainable paradigm for future systems.

This issue was first identified by the aeronautic sector, where *Integrated Modular Avionics* (IMA) was proposed in the early 90's, see figure 1. In IMA, several functions are hosted in a same computing unit and some functions are distributed

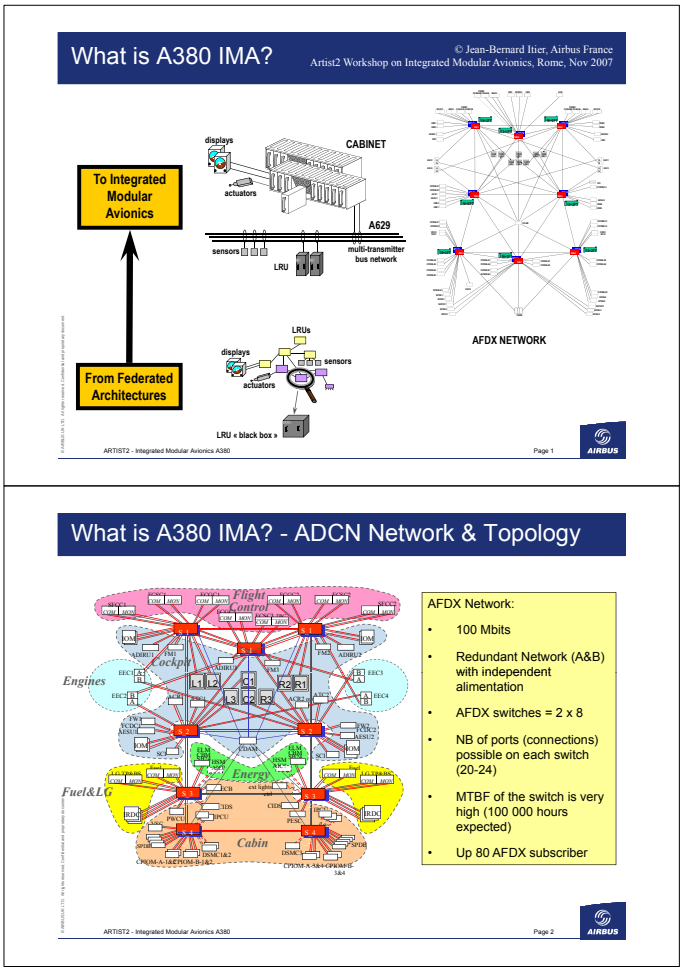

Figure 1. From Federated Architectures to IMA. Courtesy of Airbus-France.

across different computing units. Computing units as well as communication media can be standardised, thus allowing for drastic reduction in computing devices and wiring. Also, the industrial sector can get reorganised: The OEM no longer needs to buy complete systems or sub-systems from a tier one supplier. Instead, it can buy, from different suppliers, the detailed specification of a function, a computing hardware, a communication infrastructure, a middleware enabling easy deployment of functions over the distributed computing architecture, and so on. The same happens in other sectors, see for

[1] This research was supported in part by the European Commission under IST STREP 215543 COMBEST.

instance the AUTOSAR[2] architecture in automotive industry.

While this move from Federated to *Integrated Architectures* [18] opens new possibilities for further increase of embedded electronics in future embedded systems, it raises a number of challenging issues:

- The folding of different functions over shared computing units and the sharing of communication media can cause undesired interferences, thus imparing the wanted partitioning between functions.
- Since system integration involves a mix of hardware, communication infrastructure, middleware, and software (not to speak of the actuators, sensors, and other devices) in a complex way, mismatch and failure to meet overall requirements emerges as a high risk at that very late stage of system development.
- Since the overall system design relies on a layered view of the system, with several levels of abstraction corresponding to different computing or communication paradigms, it is not clear at all how detailed design can indeed match system level specifications.

B. TTA as a vision of System Architecture

To address these problems as a whole, studies regarding *System Architecture* have been developed since the late 80's. Most remarkable is the *Time-Triggered Architecture* (TTA) developed by Hermann Kopetz and his school [15], [16]. TTA builts on a vision of the system in which physical time is seen as a first class citizen and as a help, not an ennemy. The Model of Computation and Communication (MoCC) of TTA is that of *strong synchrony*: the system is equipped with a discrete logical time, that is consistently maintained throughout the overall system. Strong synchrony is achieved by maintaining strictly synchronized physical clocks throughout the distributed architecture, up to a certain maximum accuracy — which in turn specifies the finest granularity of the discrete time in a TT Architecture. Having the precise MoCC of strong synchrony makes the deployment of an application easy, provided that the latter is also based on the same MoCC — fortunately, Simulink/Stateflow and Scade, which are standard tools in use in these industrial sectors, are examples of formalisms obeying the synchronous MoCC. In particular, techniques for generating semantic-preserving implementations of synchronous models on TTA have been studied in [7].

In addition, TTA offers more possibilities to address the above discussed difficulties. Firstly, time can be used as a help in building fault tolerance services with its redundancy management and fault detection and mitigation. Secondly, time is also a help for partitioning, and for integrating components visible through their interfaces: Time Division Multiplexing (TDM) is a well established technique to grant a function access to communication or computing. TDM is also at the very core of task scheduling.

[2]http://www.autosar.org/

C. LTTA: why and what?

However, the TTA approach carries cost and timing penalties that may not be acceptable for some applications. Indeed, jitter with smaller delays is preferred to fixed but longer delays for distributed control applications [3]. Also, TTA is not easily implementable for long wires (such as in systems where control intelligence is widely distributed) or for wireless communications. Finally and most importantly, re-designs are costly, due to the need for a global re-design of Time-Division multiplexing of the different functions or tasks.

Hence, there has been growing interest in less constrained architectures, such as the *Loosely Time-Triggered Architecture* (LTTA) [4]. LTTA is characterized by a communication mechanism, called *Communication by Sampling* (CbS), which assumes that: 1/ writings and readings are performed independently at all nodes connected to the medium, using different local clocks; and 2/ the communication medium behaves like a shared memory. See figure 3 for an illustration. LTT architectures are widely used in embedded systems industries. The authors are personally aware of cases in aeronautics [21], nuclear, automation, and rail industries where the LTTA architecture with limited clock deviations has been used with success. It is indeed the architecture of choice for railway systems, in which tracks are used as the communication medium and computing systems are carried by the trains and work autonomously, see figure 2.

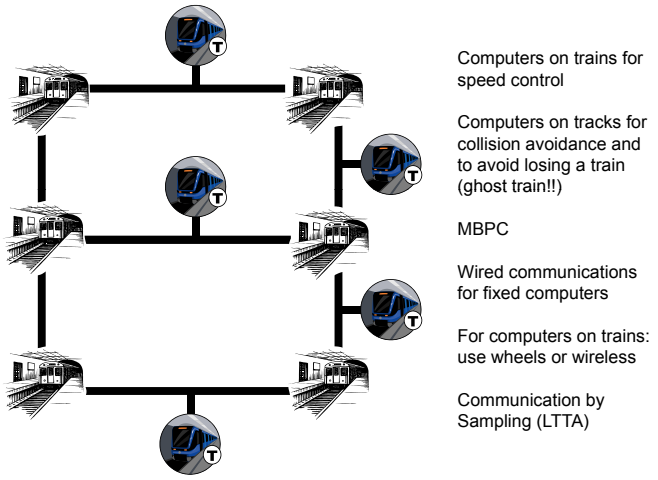

Figure 2. CbS in railway systems.

LTTA is very flexible and efficient as it does not require any clock synchronization, and it is not blocking both for writes and reads. Consequently, risk of failure propagation throughout the distributed computing system is reduced and latency is also reduced albeit at the price of increased jitter, see [3]. However, data can be lost due to overwrites or alternatively duplicated because reader and writer are not synchronized, see [1], [22], [8] for a detailed discussion of the resulting artifacts. If, as in safety critical applications that involve discrete control for operating modes or handling protection, data loss is not permitted, then special techniques must be

developed to preserve the semantics of the specification.

The LTT bus based on CbS was first proposed in [4] and studied for a single writer-reader pair and [19] proposes a variation of LTTA where some master-slave re-synchronization of clocks is performed. More recently, LTT architecture of general topology was studied in [1], [22], using techniques reminiscent from so-called back-pressure [6], [5] and the related elastic circuits [9]. In a different direction, [17] developed an alternative approach where up-sampling is used in combination with "thick" events as a way to preserve semantics. This approach, which is more time-based as compared to [1], [22], was further developed and clarified in [8].

In this paper we discuss the novel issues LTTA raises due to the artifacts caused by CbS communication. We study the essence of LTTA and we put it in the perspective of System Architecture. This paper does not contain any new technical result by itself, but no such an overview existed yet. The paper is organized as follows. In section II we discuss the use of CbS communication and the artifacts it causes, for both low level, hard real-time, continuous feedback control and for discrete systems. Section III is the core of this paper. The two versions of LTTA to be put on top of CbS are presented, namely the back-pressure one and the timed one. Finally, in the conclusion, we cast this work in the perspective of System Architecture and discuss future directions.

II. Use of LTTA and artifacts

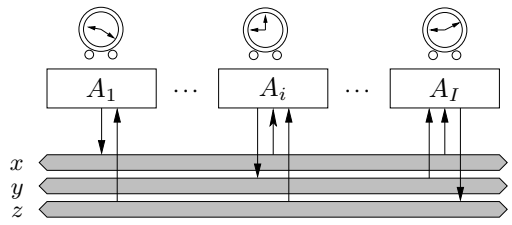

Figure 3. Communication by Sampling (CbS). Three shared memories are depicted as shaded buses, for the three variables x, y, and z. For each variable, there is one writer and zero or more readers.

LTTA relies on *Communication by Sampling*, which is illustrated on figure 3 and formalized as the following set of assumptions:

Assumption 1 (Communication by Sampling):

1) *the communication medium behaves like a collection of shared memories, one for each variable;*
2) *updates of every variable are visible to every node;*
3) *writings and readings are performed independently at all nodes connected to the medium, using different, non synchronized, local clocks.*

Use for continuous feedback control: Because it is a close replica of analog architectures in the digital world, the architecture of figure 3 and assumption 1 is directly suited to low level continuous feedback control. CbS communication causes jitter, the effect of which is analysed, e.g., in [14] using frequency domain techniques. Unfortunately, CbS communication also results in losses and duplications, see figure 4. Therefore, the techniques of [14] only address part of the

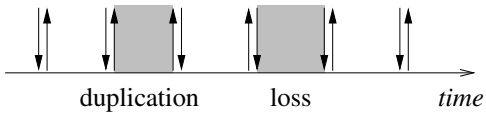

Figure 4. Duplications and losses due to CbS. Downward and upward pointing arrows indicate writings and readings, respectively.

problem. On the other hand, to the best of our knowledge, *robust control design* techniques [11][3] do not encompass duplications and losses in the way uncertainties are captured. Still, CbS communication is a very natural and appealing architecture for distributed control and is indeed widely used in practice. Thus, it is our opinion that a full study of CbS in the context of robust control design remains a challenge for control scientists.

Use for discrete control and supervision: Discrete systems such as finite state machines are generally not robust against duplication or loss of data. Combinational functions are generally not robust, and, even worse, sequential functions (with dynamically changing state) may react to such artifacts by diverging from their nominal behaviour. The issue is illustrated on Figure 5 for the case of combinational functions. We show here the case of A_1 reading two boolean inputs

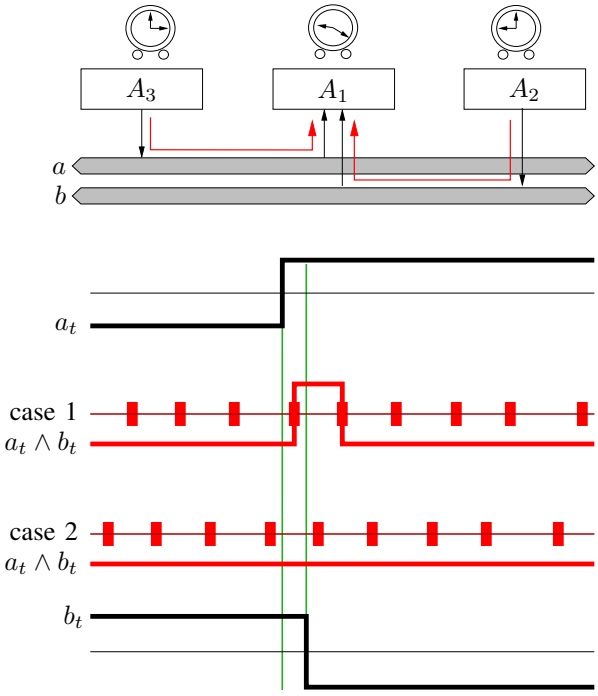

Figure 5. Sensing multiple signals, for distributed clocks subject to independent drifts and jitters. Referring to figure 3, we show the case of A_1 reading two boolean inputs originating from A_2 and A_3, respectively, and computing their conjunction. Cases 1 and 2 correspond to two different outcomes for the local clock of A_1.

[3]See also the Matlab toolbox http://www.mathworks.fr/products/robust/

a_t and b_t originating from A_2 and A_3, respectively, and computing their conjunction $a_t \wedge b_t$. Cases 1 and 2 correspond to two different outcomes for the local clock of A_1. Observe that the result takes three successive values F, T, and F for case 1, whereas case 2 yields the constant value F. The origin of the problem is that events attached to different signals can be separated by arbitrary small time intervals — in Figure 5 the problem comes from the very close jumps for a_t and b_t. Increasing the clock rate cannot prevent this from occurring. More clever protocols are needed to cope with this.

III. Two protocols for LTTA

In this section we present two protocols that were proposed on top of CbS communication in order to ensure that the deployment of synchronous applications over resulting LTT Architectures preserves the semantics. The first protocol is an adaptation of elastic circuits in hardware, and the second one is a softening, time based, adaptation of the original TTA.

A. Back-Pressure LTTA

Elastic circuits were proposed in [10], [12], [9] as a semantic preserving architecture in which Kahn Process Network [13] type of execution is performed using bounded buffers. This is achieved by relying on a mechanism of *back-pressure* [5] by which readings from a buffer by a node is acknowledged to the writer using a reversed virtual buffer. Net $\mathcal{N}_{ji}$ of figure 6 depicts how a link $j \to i$ with a 2-buffer is implemented in an elastic circuit for running a synchronous application with a 1-delay communication.[4] *Back-pressure* places and arcs of this net are dashed, to distinguish them from the corresponding *direct* places and arcs, which are solid — solid and dashed places and arcs both obey the usual net semantics. Only direct places model data communication, back-pressure ones are there to prevent from buffer overflow at the link $j \to i$.

In our study, however, we cannot make direct use of elastic circuits since the activation of nodes in elastic circuits is triggered by tokens, not by autonomous non-synchronized quasi periodic clocks as in LTTA. To adapt to the constraints of LTTA, the authors of [22] have proposed to enhance elastic circuits with a *skipping mechanism* that we present now under the name of *back-pressure* LTT Architecture.

Figure 6 depicts net $\mathcal{N}_{ji}$ associated to each directed link $j \to i$ of the architecture. Note that this net assumes a 2-buffer on each link. Reactions at node i are captured by the net $\mathcal{N}_i$ which is composed of a two-step "read; write" together with a skipping mechanism associated to node i. The following holds:

- This skipping mechanism is triggered by the local clock at node i, see figure 6;
- Transitions with labels r_i and w_i have priority over transition with label $skip_i$.

[4]This is an assumption that we enforce in the sequel. It makes the analysis easier but it is not essential. On the other hand, it is often valid in actual designs.

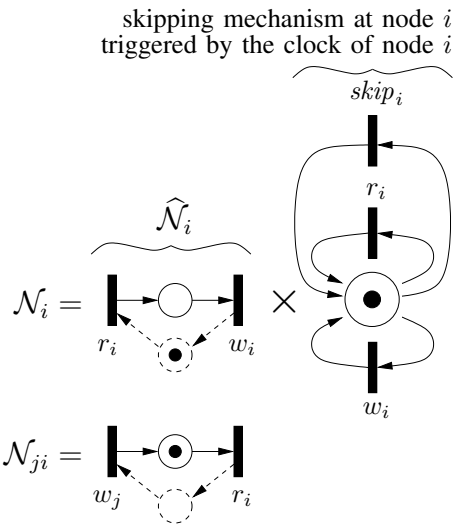

Figure 6. Back-pressure net $\mathcal{N}_{ji}$ (bottom) associated to each directed link $j \to i$ of the architecture and net $\mathcal{N}_i$ (top) showing the mechanism of skipping at node i.

The composition indicated on figure 6 by the symbol $\times$ is by superimposing transitions having same label. Thus, when none of the two "read" or "write" transitions is enabled and clock of node i has a tick, then $skip_i$ is fired, expressing that node i keeps silent at that tick. This skipping mechanism was proposed in [22] in order to avoid computing units getting blocked — blocking is replaced by skipping.

Consider the following two product nets

$$\widehat{\mathcal{N}} = \left(\prod_i \widehat{\mathcal{N}}_i\right) \times \left(\prod_{j\to i} \mathcal{N}_{ji}\right) \quad (1)$$

$$\mathcal{N} = \left(\prod_i \mathcal{N}_i\right) \times \left(\prod_{j\to i} \mathcal{N}_{ji}\right) \quad (2)$$

where the product is obtained by superimposing transitions having identical labels. Net $\widehat{\mathcal{N}}$ yields the *elastic circuit* implementing the original synchronous application according to the Kahn Process Network semantics. Net $\mathcal{N}$ is the *back-pressure net* modeling our back-pressure LTTA, with the skipping mechanism.

Observe that net $\widehat{\mathcal{N}}$ exhibits no conflict and is thus an *event graph* (also sometimes called *marked graph*). With our assumption of single-delay communications, net $\widehat{\mathcal{N}}$ is indeed 1-safe. Net $\widehat{\mathcal{N}}$ just models the Kahn Process Network execution of synchronous application, which is known to preserve synchronous semantics. Mild fairness conditions on the different local clocks ensure that no skipping mechanism can be unfair with respect to reads and writes, which in turn ensures that back-pressure net $\mathcal{N}$ also preserves synchronous semantics. Performance results are provided in [22] for more general architectures (with arbitrary topology and buffer sizes).

The skipping mechanism ensures that computing nodes themselves never get blocked due to the failure of other nodes or communication. However, net $\widehat{\mathcal{N}}$ exhibits blocking read communication between the different computing nodes of the architecture. This means that, *when focusing on the*

effective communication of fresh data at a given node, blocking does still occur in net $\mathcal{N}$. This observation actually motivated considering the alternative, time-based, LTT Architecture that we propose and analyse in the next section.

B. Time-Based LTTA

To overcome the drawbacks of the back-pressure architecture, the following protocol can be considered.

We first give an intuition for it that we call the *wandering philosophers:* n philosophers want to meet every hour and tell each other the beautiful papers they have read in their different libraries. They all have (1) a watch, and (2) a beep. The protocol is as follows: when they have finished chatting they separate and go to their respective libraries. Their plan is to come back within one hour. However, since they are venerable philosophers, they possess venerable watches too, which somehow are imprecise and non synchronized. Now, would they entirely rely only on their watches, they could get frustrated by not meeting any more after a few rounds, because time divergence has accumulated. However, using their equipement in a clever way they can proceed as follows. When the earliest one comes back to the meeting point, then she beeps everyone. So, all other venerable philosophers can walk their way back to the meeting point while keeping thinking. Now, suppose that everyone knows that all watches won't diverge by more than five minutes within one hour. They can then deduce that they won't need to wait for more than a certain amount of minutes before being able to chat again with all the colleagues. And the procedure repeats.

In this protocol, roles of the different devices are the following. The watches ensure that the philosophers won't hurry back to the meeting until their watch indicates they should do so. This will leave enough time for everyone to study enough philosophy. Still, the beep ensures that the return will occur faster in case a philosopher has too slow a watch to rely on it. Observe that every philosopher will return to the meeting point and won't keep stuck at the library, even if she doesn't hear the beep in case she's got old. Philosophers never get blocked in their quest for wisdom.

This idea is now formalized as the *time-based LTTA* [8], whose protocol is defined, for each node i, by the automaton of figure 7. This automaton is triggered by the local clock of node i. It comes equipped with a local counter n and its actions are guarded, either by events collected from CbS medium thanks to assumption 1.2, or by conditions on counter n.

Conditions on the four parameters n_{1a}, n_{1b}, n_{2a}, and n_{2b}, are provided in [8] that ensure the preservation of synchronous semantics at deployment. This is shown by ensuring that broadcast and execution phases nicely alternate globally in the overall architecture, despite clock jitter and drifts and variable transmission delays. Performance results are also provided in [8].

Observe, now, that the time-based LTTA has non-blocking communication. If a node or communication link fails by getting silent, then the nodes reading from that node or that link will just proceed to their local computations using old data provided as backups by CbS communication, in combination with fresh data from live links and nodes. Time-based LTTA is thus fully non-blocking, although the application enters a degraded mode in case of silent failure of a node or link, by using outdated data from its failed input links.

IV. Conclusion

Complex embedded systems design calls for a comprehensive vision of system architecture as well as methodologies for system design.

Regarding methodologies, Platform Based Design (PBD) [2] has been advocated by A. Sangiovanni-Vincentelli as a method to reflect, in a single model, several layers encountered in the course of system design. Typical instances of such layers are the application (top layer) and the computing infrastructure (bottom layer). By using the principle of "meeting-in-the-middle", both layers are reflected in the mid layers using appropriate abstractions, and the joint model can then be used to study deployment.

As for System Architectures, we have emphasized the importance of the contribution of TTA, not only as a distributed execution infrastructure, but also as a comprehensive framework for overall System Architecture. TTA offers the advantage of having a clean MoCC, which is a pre-requisite for PBD to apply smoothly. Of course, as explained in, e.g., [16], TTA cannot be used as the single architectural paradigm in a complex, multi-layered, embedded system. Referring to the case of aircrafts and to figure 1, by being very safe but too constraining, TTA would be suitable for flight control and engine control, but probably not for other sub-systems. This has indeed been acknowledged by H. Kopetz himself with the development of a mixed TTA/standard Ethernet infrastructure [20]. Now, even for the safety critical hard real-time layers where TTA seems appropriate, it may not always be accepted for flexibility reasons as explained in the introduction.[5]

LTTA was proposed as a softening of TTA for the very same layers. In fact, the objective of LTTA is to offer an abstraction that emulates TTA. So far the work done on LTTA [22], [8] has achieved this objective regarding 1/ the preservation of semantics, and 2/ performance. By offering a TTA abstraction, it opens the way to using TTA-based System Architecture whenever wanted. Still, fault tolerance and robustness services are needed as well. These must be developed as part of the LTTA middleware, not on top of it — this is work to be done. Preliminary studies reveal that the back-pressure based LTTA is more flexible but less robust against failures than time-based LTTA. It makes therefore sense to use different versions of LTTA for different parts of the system. This opens the question of the blending of the two architectures, while maintaining its essential properties regarding semantics preserving.

[5] It is for instance not used in the A380.

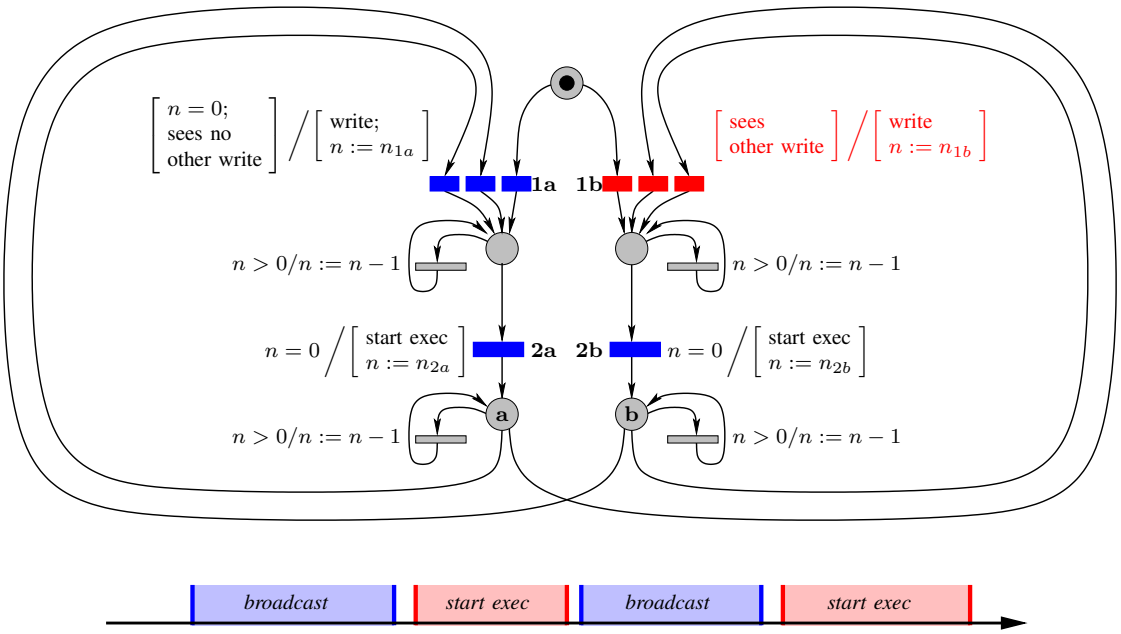

Figure 7. Time-based LTTA following [8]: the protocol sitting at each node i. Counter n is local to this node and the protocol is triggered by the local clock of node i. The red transitions are synchronizing, the other ones let time pass.

REFERENCES

[1] A. Benveniste and P. Caspi and M. di Natale and C. Pinello and A. Sangiovanni-Vincentelli and S. Tripakis, "Loosely Time-Triggered Architectures based on Communication-by-Sampling," in *EMSOFT*, 2007, pp. 231–239.

[2] A. Sangiovanni-Vincentelli, "Quo Vadis, SLD? Reasoning About the Trends and Challenges of System Level Design," *Proc. of the IEEE*, vol. 95, no. 3, pp. 467–506, 2007.

[3] K.-E. Årzén, "Timing analysis and simulation tools for real-time control," in *Formal Modeling and Analysis of Timed Systems*, ser. LNCS, P. Pettersson and W. Yi, Eds. Springer, Sep. 2005, vol. 3829, extended astract in the Proceedings of FORMATS 2005, Uppsala. Invited Talk.

[4] A. Benveniste, P. Caspi, P. L. Guernic, H. Marchand, J.-P. Talpin, and S. Tripakis, "A protocol for loosely time-triggered architectures." in *EMSOFT*, 2002, pp. 252–265.

[5] L. P. Carloni, "The role of back-pressure in implementing latency-insensitive systems," *Electr. Notes Theor. Comput. Sci.*, vol. 146, no. 2, pp. 61–80, 2006.

[6] L. P. Carloni, K. L. McMillan, and A. L. Sangiovanni-Vincentelli, "Theory of latency-insensitive design," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 20, no. 9, pp. 1059–1076, 2001.

[7] P. Caspi, A. Curic, A. Maignan, C. Sofronis, S. Tripakis, and P. Niebert, "From Simulink to SCADE/Lustre to TTA: a layered approach for distributed embedded applications," in *Languages, Compilers, and Tools for Embedded Systems (LCTES'03)*. ACM, 2003.

[8] P. Caspi and A. Benveniste, "Time-robust discrete control over networked loosely time-triggered architectures," in *Proc. of the IEEE Control and Decision Conference*, December 2008.

[9] J. Cortadella and M. Kishinevsky, "Synchronous elastic circuits with early evaluation and token counterflow," in *DAC*, 2007, pp. 416–419.

[10] J. Cortadella, A. Kondratyev, L. Lavagno, and C. P. Sotiriou, "Desynchronization: Synthesis of asynchronous circuits from synchronous specifications," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 1904–1921, 2006.

[11] Feng Lin, *Robust Control Design: An Optimal Control Approach*. Wiley, 2007, iSBN: 978-0-470-03191-9.

[12] K. C. Gorgônio, J. Cortadella, F. Xia, and A. Yakovlev, "Automating synthesis of asynchronous communication mechanisms," *Fundam. Inform.*, vol. 78, no. 1, pp. 75–100, 2007.

[13] G. Kahn, "The semantics of a simple language for parallel programming," in *Information Processing 74, Proceedings of IFIP Congress 74*. Stockholm, Sweden: North-Holland, 1974.

[14] C.-Y. Kao and B. Lincoln, "Simple stability criteria for systems with time-varying delays," *Automatica*, vol. 40, no. 8, pp. 1429–1434, 2004.

[15] H. Kopetz, *Real-Time Systems*. Kluwer Academic Publishers, 1997.

[16] ——, "The complexity challenge in embedded system design," in *ISORC*, 2008, pp. 3–12.

[17] C. Kossentini and P. Caspi, "Approximation, sampling and voting in hybrid computing systems." in *HSCC*, 2006, pp. 363–376.

[18] M. di Natale and A. Sangiovanni-Vincentelli, "Moving from Federated to Integrated Architectures: the role of standards, methods, and tools," *Proc. of the IEEE*, to appear.

[19] J. Romberg and A. Bauer, "Loose synchronization of event-triggered networks for distribution of synchronous programs." in *EMSOFT*, 2004, pp. 193–202.

[20] K. Steinhammer, P. Grillinger, A. Ademaj, and H. Kopetz, "A time-triggered ethernet (TTE) switch," in *DATE*, 2006, pp. 794–799.

[21] P. Traverse, I. Lacaze, and J. Souyris, "Airbus fly-by-wire: A total approach to dependability," in *IFIP World Congress, Toulouse*. IFIP, 2004.

[22] S. Tripakis, C. Pinello, A. Benveniste, A. L. Sangiovanni-Vincentelli, P. Caspi, and M. D. Natale, "Implementing synchronous models on loosely time triggered architectures," *IEEE Trans. Computers*, vol. 57, no. 10, pp. 1300–1314, 2008.