Formal Validation of a Deterministic MAC Protocol

Karen Godary-Dejean, David Andreu

Submitted to MVDES'10 Special Issue

Formal validation of a deterministic MAC protocol

KAREN GODARY-DEJEAN and DAVID ANDREU
LIRMM, University Montpellier 2

This article deals with the formal validation of a medium access protocol. This protocol has been designed to meet the specific requirements of an implantable network-based neuroprosthese. This article presents the modeling of STIMAP with Time Petri Nets (TPN), and the verification of the deterministic medium access it provides, using timed model checking. Doing so, we show that existent formal methods and tools are not perfectly suitable for the validation of real system, especially when some hardware parameters has to be considered. This article then presents how these difficulties have been managed and gives the validation results for STIMAP, providing constraints on the protocol parameters that must be respected to guaranty its determinism.

Categories and Subject Descriptors: C.2.2 [**Computer-communication networks**]: Network Protocols—Protocol verification; C.2.5 [**Computer-communication networks**]: Local and Wide-Area Networks—Access schemes, Buses; D.2.4 [**Software engineering**]: Software/Program Verification—Validation

General Terms: Reliability, Verification

Additional Key Words and Phrases: Formal validation, MAC determinist protocol, Model checking, Modeling, Timed Petri nets

1. INTRODUCTION

In order to improve the daily life living of para- and quadriplegic patients, Functional Electrical Stimulation (FES) is a palliative solution. FES has been successfully used to face functional deficiencies, in well-known applications such as: pacemaker, deep brain stimulation, pain control or hearing restoration. Implanted FES is also studied for movement rehabilitation such as foot droop for hemiplegic patients or even more complex movements, as well as for restoration of bladder function. Nevertheless, effective solutions for implanted FES are actually mainly based on centralized architectures. Since it is not conceivable to address all possible functional deficiencies in only one surgical operation, the architecture must be extensible. That means that once implanted the evolution of the neuroprosthese should be possible, i.e. it must be possible to add new sites of stimulation and/or measurement of electrical activities on nerves and/or muscles. This is not

Author's address: K. Godary-Dejean and D. Andreu, LIRMM, UMR 5506 - CC 477, 161 rue Ada, 34095 Montpellier Cedex 5 FRANCE.

possible with existing implantable neuroprostheses. Moreover, centralized implants lead to complex surgery, high risk of failure during and after surgery, and global infection problems, which can be limited with a distributed architecture. So, we designed and developed a new architecture for such implantable FES system based on technological advances in networks and micro-electronics domains. This new architecture relies on a network of small distributed units in charge of activating and monitoring the peripheral nervous system [Andreu et al. 2009]. Thus, the network constitutes the backbone of our neuroprosthese and it must be managed in a reliable and deterministic way in such critical embedded system.

The medium access mechanism used in this architecture must be deeply studied since it plays a major role regarding the communication between artificial devices that control natural organs. To fulfill the context specific requirements and constraints, we designed a new MAC protocol, presented section 2. Taking early into account validation preoccupations, the behavior of this MAC protocol, called STIMAP (Sliding Time Interval based Medium Access Protocol), has been modeled using Time Petri nets (TPN). Both TPN-based simulation and experimentations contributed to the validation of STIMAP but they did not provide exhaustive guarantees of determinism.

Some particularities of STIMAP could not be validated without applying a validation methodology based on formal methods. As a consequence, we went further in the validation process, applying a model checking based validation methodology, briefly mentioned section 3. It allows verifying properties through an exhaustive analysis of the whole states space (all possible states) of a formal model of the system. Therefore, formal approach gives more confident validation results thanks to the exhaustive verification of properties on the STIMAP protocol. We applied it to the validation of the Time Petri net (TPN) based STIMAP model: that constitutes the focus of this paper. Sections 4, 5 and 6 expose respectively the modeling and the validation results of STIMAP. They show difficulties we encountered in the identification of time parameters that guaranty the respect of communication time constraints, and in the integration of such parameters in the validation process. This article is concluded by our ongoing works on formal analysis methods and tools.

2. THE STIMAP MAC PROTOCOL

In order to explain the expected characteristics of the MAC protocol to be formally validated, we first briefly introduce our network-based FES architecture. Then the STIMAP protocol which has been designed to manage the medium access within this architecture is presented.

2.1 Communication in the FES architecture

The whole FES architecture relies on an implanted part and some external devices like for instance the patient's remote controller. The implanted architecture is constituted by a global controller and several distributed stimulation units (DSU) and measurement units (DMU) which are connected together via an intrabody asynchronous communication network. The global controller is in charge of coordinating the activities of distributed units. The link between the intrabody architecture and extrabody device(s), for data and energy transmission, is ensured by means of an

transcutaneous inductive link. In this architecture the global controller also ensures the "gateway" between those two worlds.

Each implanted device embeds a 3-layer protocol stack corresponding to the reduced OSI model:

—Physical layer: the network is based on an implantable bus we developed as there is not yet available wireless technology for intra-body communication with deeply implanted devices taking into account both data and energy transmission. Our bus is based on a medical 2-wire cable (35N LT metal implantable 2-wire cable), transmitting data and energy [Souquet et al. 2007]. Whatever is the medium, the bus topology of the network is essential. Indeed, for the approach to remain valid for both wired and wireless technologies (implanted and external devices), we assume that the medium fundamentally corresponds to one shared domain of not detectable collision.

—MAC layer: on the implanted network, two types of logical addresses are required, allowing unicast, multicast and broadcast communications: individual addresses (one for each DSU) and group addresses (for different groups of DSUs, and all DSUs). These addressing modes are necessary since the controller can communicate for example with a single DSU to program it, or with a group of DSUs to start a stimulation, or with all DSU to stop any stimulation. The notion of group is significant in our context since at a given instant of time the movement control only concerns a subset of muscles (simultaneous stimulation of agonistic and antagonistic muscles for example) and thus a subset of DSUs (those associated to these muscles). This implies that it is possible to dynamically impose to a DSU to subscribe/unsubscribe to one or several groups. The possibility to dynamically unsubscribe a DSU from a group is also important, particularly for medium sharing efficiency purposes. Consequently, basic functionalities of the MAC layer are to filter incoming packets, since at the physical layer we systematically broadcast frames, and to manage subcription/unsubscription to groups. But the most important functionality is to ensure a reliable, deterministic and efficient medium sharing. It must ensure that solely one node of the network uses the medium at any instant of time, to avoid any collision. The risk of collision must be avoided to be sure that no error notification neither any request can be lost, even if for reliability purpose acknowledgements can be used. Two types of acknowledgements are provided: frame reception (MAC layer) and request execution (Application layer) acknowledgements.

—Application layer: payloads of the application protocol correspond to DSU configuration, download/upload of micro-programs (stimulation profiles to be executed on a DSU), remote control by means of start/stop requests and stimulation parameter modifications (modulation of the stimulus magnitude, duration and frequency), and also acknowledgements and errors notified by DSUs.

The context is thus characterized by:

—The topology is fixed: no mobility neither dynamic node insertion.

—The application needs two kinds of logical addresses: unicast and multicast (broadcast being a particular multicast address). The set of nodes (group of

DSU for instance) that communicates at the different phases of the stimulation is dynamically set. Moreover it must be possible to exclude or suspend members of the communicating group, at any time.

—The application obviously requires a reliable and efficient communication medium. For reactivity, and security purposes the global controller of the application (master unit of the architecture) must have the possibility to quickly react, i.e. the possibility to quickly emit its requests without waiting for a long time before acceding to the medium.

—For reliability, as the used technology does not allow collision detection, the MAC protocol must ensure that collisions can not occur.

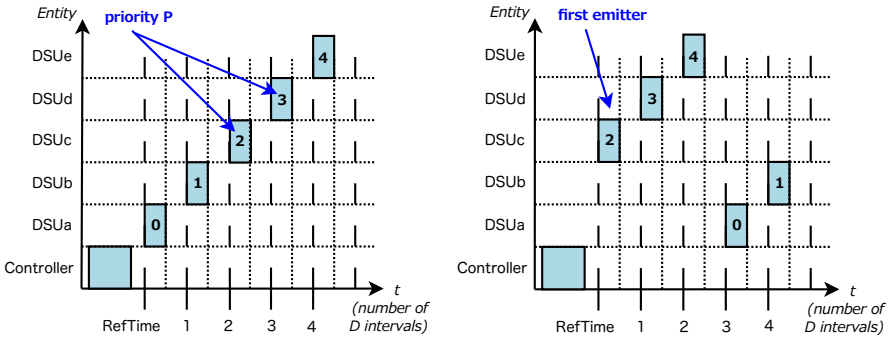
—Finally, the context imposes the solution to be simple as it must be embedded in small implantable electronic devices. Moreover this solution must be based on independent communication devices, to facilitate the system evolutivity.

2.2 The STIMAP medium access

In the given context, contention-based solutions are not adequate. Schedule-based solutions, considering that election and TDMA based methods belongs to this family, are possible solutions as they favor collision-free and deterministic medium access. However, TDMA solutions are often static and periodic ones, i.e. requiring regular time synchronization and sometimes regular timing information sending. This increases energy consumption and decreases efficiency in case of passives nodes (nothing to emit). In our architecture, the topology is that of a centralized controller and stimulation units distributed over a bus. The controller, being the master, is the central point of the system, initiating all the communications with the distributed units, i.e. the slaves. The method is simple, as one slave transmits a frame only if the master has demanded or authorized it. The master/slave approach is appropriated, nevertheless the limited efficiency due to the protocol overhead must be improved, taking into account the multicast possibility. Indeed, this model of cooperation is not really efficient when dealing with a group of slaves as it requires to poll slaves, i.e. to individually communicate with each one. Time sharing (TDMA) can be a good solution to avoid polling, if clock drift problem and time-slots losses can be faced. However, a "static" TDMA is not adequate since the controller needs to communicate with a subset of slaves only when stimulation is performed, and moreover this subset of slaves can be dynamically defined.

So, the combination of time sharing and master/slave relation are the basis of the STIMAP protocol we propose. In short, we defined a method, adequate to our decentralized architecture, which is simple to implement, which allows TDMA to be contextually used, providing dynamic time-slots assignement and limiting time-slots losses.

STIMAP uses a method based on the master-slave model which has been modified for efficiency considerations. It obviously allows basic individual master-slave and slave-master communications, but also offers a way to manage the communications with a group of slaves without polling all the members of the group. The master manages the access of the slaves to the medium by means of a "Speaking Right", similar to a token, it allocates to the members of the network. We distinguish Individual Speaking Right (ISR, individual token) and TDMA-based Group



(a) First emitter $BP = 0$ (b) First emitter $BP = 2$

Fig. 1. GSR based communication scenarios

Speaking Right (GSR, group token). The ISR based mode is trivial: the DSU which receives an ISR has the authorization to emit one packet (practically, there is no payload requiring to send more than one packet). On the contrary, the GSR based mode is more complex and will be explained in the following section.

2.3 Group medium access principles: the GSR based mode

In order to avoid polling of slaves, we define a TDMA-based group medium access sharing. TDMA is only contextually used, i.e. only when the master has to communicate with a group of slaves. This section outlines the principles of the STIMAP medium access mechanism, assuming the existence of a global time. Indeed, the TDMA sequence begins with the emission of a master frame on which all the DSUs are synchronized. The influence of the clock drift is limited as the TDMA sequence is not a long period. The GSR frame reception can be seen as a logical synchronization; every slave time-slot is positioned relatively to this frame.

When dealing with a group of slaves (DSU), each group member (each DSU) knows the size of the group and its position in that group as this position is defined in term of priority: the lower is the position the higher is the priority. A DSU can be member of different groups and can have different positions in these groups. When the master allocates a GSR to a given group, it sends only one frame, which can be compared to a beacon. This frame is physically broadcasted, but only dedicated to the members of the indicated group. In this frame, it also indicates which member must begin the communication. Indeed, it is not necessarily the member of highest priority that begins to send its packet.

Example. Figure 1 represents two basic GSR communications for a group of 5 DSUs (group size $GS = 5$). In the first scenario of Figure 1(a), the DSUa with priority 0 begins to emit after the reception of the GSR frame sent by the controller. The others DSUs then emit one after the other on their turn, according to their priority. In the second scenario of Figure 1(b), all the members of the group emit one after the other but the first DSU to emit is the one with priority 2 (DSUc).

2.4 Time-slot positioning mechanism

Knowing its position in the group, the group size and the member who must begin the communication, each member determines when it will have the right to emit its packet, i.e. the position of its time-slot. Each member can speak during the time-slot duration, a given time interval D that has been automatically computed by the master or explicitly specified in an initial phase. The positioning, in terms of member's position in the communication round, is given by the variable Pos_P, which corresponds to the position of the member of priority P:

$$Pos_P = P - BP + \alpha_P \times GS \tag{1}$$

where:

—P is the priority number of the slave,

—BP is the priority number of the DSU which begins the communication,

—GS is the group size,

—and $\alpha_p = 1$ if $P < BP$ else $\alpha_p = 0$.

And then, from a time point of view, the time-slot position of the DSU with priority P is given by:

$$TimeSlotPosition_P = RefTime + Pos_P \times D \tag{2}$$

where:

—D is the time-slot duration,

—and $RefTime$ corresponds to the reference instant of time for every node. The reference time mechanism is described section 2.6.

Example: scenario of Figure 1(a). This scenario could be used to illustrate the time slot positioning mechanism. It represents a GSR communication of 5 DSUs ($GS = 5$), beginning with the DSUa emission. DSUa's priority is $Pa = 0$, then we have $BP = Pa = 0$.

If we are interested in the calculation of the time slot position of DSUb ($Pb = 1$), we have $Pb > BP$ then $\alpha_{Pb} = 0$. Then we can calculate the time slot position of DSUb using equations 1 and 2:

$$\begin{aligned} Pos_{Pb} &= 1 - 0 + 0 \times 5 = 1 \\ TimeSlotPosition_{Pb} &= RefTime + 1 \times D \end{aligned}$$

In the same way, we can calculate the time slot position of DSUd ($Pd = 3$):

$$\begin{aligned} Pos_{Pd} &= 3 - 0 + 0 \times 5 = 3 \\ TimeSlotPosition_{Pd} &= RefTime + 3 \times D \end{aligned}$$

Then the DSUb must emits its frame one D interval after the $RefTime$ instant, as we can shown in Figure 1(a). Similarly, DSUd must wait 3 D intervals before emitting.

Example: scenario of Figure 1(b). This scenario illustrates the time slot positioning when the first emitter is not the one with the smallest priority. Indeed, the communication in this case is begun by DSUc, then we have $BP = Pc = 2$.

As in the preceding example, we want to calculate the time slot position of DSUb and DSUd, with priority $Pb = 1$ and $Pd = 3$. The calculation of their time slot positions with equations 1 and 2 becomes:

$$\begin{aligned} \alpha_{Pb} &= 1 \quad \text{since} \quad Pb < BP \\ Pos_{Pb} &= 1 - 2 + 1 \times 5 \\ Pos_{Pb} &= 4 \\ TimeSlotPosition_{Pb} &= RefTime + 4 \times D \end{aligned}$$

$$\begin{aligned} \alpha_{Pd} &= 0 \quad \text{since} \quad Pd > BP \\ Pos_{Pd} &= 3 - 2 + 0 \times 5 \\ Pos_{Pd} &= 1 \\ TimeSlotPosition_{Pd} &= RefTime + D \end{aligned}$$

Then the DSUb must wait 4 D interval after the reference time instant, whereas DSUd emits its frame 1 D interval after $RefTime$.

2.5 Sliding time interval mechanism

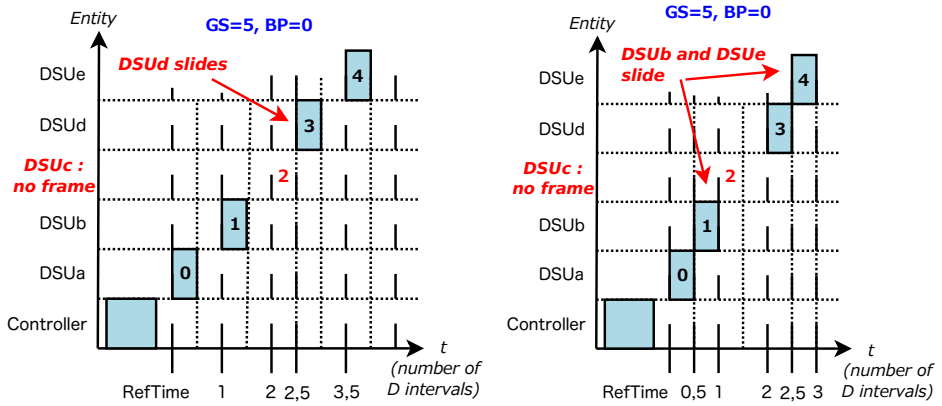
The time-slot attributed to a slave is in fact constituted by two half time-intervals: the first half-interval is dedicated to the slave communication, and the second one is reserved for a potential reaction of the master. For instance, if the slave notifies a significant error the master could have to immediately react, to stop the stimulation for example. So, to be sure that the master will have access to the medium without any collision risk, this second half-interval is reserved. This contributes to the reactivity of the distributed stimulation architecture and is very important in our context.

However, if a slave has nothing to transmit, its half-interval is free and the half-interval reserved for the master is then unused and wasted. To avoid that, i.e. to reach better performances, the MAC method integrates a sliding time interval mechanism. When waiting for its time-slot, every slave listens to the medium: if the previous member of the group did not emit a packet then it brings backward its own time-slot by a half time-interval. In other words, it recovers the half time-interval that was reserved for the master but that will never be used.

In fact, the master can dynamically configure the sliding strategy. Three possibilities, named sliding rules, which must be exploited in a coherent way by the master, are proposed:

—*No sliding*: the master wants to always be able to react. It inhibits the sliding mechanism.

—*Sliding in case of "unused time interval"*: it corresponds to the case previously exposed. For example, this rule is selected when the master asks a DSU group to notify their potential error detection. So, if a DSU does not have any error to notify, it does not emit a frame and the next group member can recover the second half time-interval because the extern master will not have to react. An



(a) Unused time interval sliding rule (b) Used time interval sliding rule

Fig. 2. GSR communication with sliding

example is given on Figure 2(a) showing that DSUd begins to emit its frame after 2.5 time-intervals (after the reference time) instead of 3 time-intervals in the normal case (Figure 1(a)).

—*Sliding in case of "used time interval"*: it corresponds to a situation in which the master must not respond if the DSU emits a frame. For example, this rule is selected when the master performs a test of presence on a DSU group. If a DSU responds, the second half time-interval can be recovered since the master will not has to react. An example is given on Figure 2(b) showing that DSUb and DSUe begin to emit their frames after 1/2 time-interval instead of 1 time-interval (Figure 1(a)).

With this sliding mechanism, the time-slot positioning becomes more complex. It can be represented by the following equation:

$$TimeSlotPosition_P = RefTime + Pos_P \times \frac{D}{2} + \sum_{k=1}^{Pos_P} \delta_k \times \frac{D}{2}, \quad \forall Pos_P > 0 \quad (3)$$

where Pos_P is computed according to equation 1, and δ_k depends on the selected sliding rule:

—in case of "no sliding" rule, $\delta_k = 1 \ \ \forall i$,

—in case of "unused time interval" rule, $\delta_k = 1$ if the previous member (ie. the member at position i-1) sent a frame, else $\delta_k = 0$,

—in case of "used time interval" rule, $\delta_k = 0$ if the previous member sent a frame, else $\delta_k = 1$.

This equation is of cause valid $\forall Pos_P > 0$, as for the first emitter (with $Pos_P = 0$) there is no sliding possibility.

Example: scenario of Figure 2(a). We can illustrate the sliding mechanism calculating the time slot position of the DSUb and DSUd. This scenario represents

a GSR communication with the unused time interval sliding rule, when the DSUc does not emit its frame.

Remember that in section 2.4 we have calculated $Pos_{Pb} = 1$ and $Pos_{Pd} = 3$, which do not change regardless the sliding rule value if GS and BP are still the same ($GS = 5$ and $BP = 0$). Then equation 3 gives:

$$\begin{aligned} TimeSlotPosition_{Pb} &= RefTime + 1 \times \frac{D}{2} + \delta_1 \times \frac{D}{2} \\ TimeSlotPosition_{Pd} &= RefTime + 3 \times \frac{D}{2} + \delta_1 \times \frac{D}{2} + \delta_2 \times \frac{D}{2} + \delta_3 \times \frac{D}{2} \end{aligned}$$

With the "unused time interval" rule we have in this scenario:

— $\delta_1 = 1$ because the DSUa at position 0 has emit a frame;
— $\delta_2 = 1$ because the DSUb at position 1 has emit a frame;
— $\delta_3 = 0$ because DSUc at position 2 does not emit.

Then:

$$\begin{aligned} TimeSlotPosition_{Pb} &= RefTime + D \\ TimeSlotPosition_{Pd} &= RefTime + 5 \times \frac{D}{2} \end{aligned}$$

As shown Figure 2(a), and confirmed by these calculations, the time slot position value for DSUb does not change compared with the scenario of Figure 1. However, DSUc does not emit its frame, thus the following DSUd has slided and its time slot position is different than in the normal scenario: 2,5 D interval instead of 3.

Example: scenario of Figure 2(b). We finally illustrate the sliding mechanism with the used time interval rule, calculating the DSUb and DSUd time slot positions:

—we still have $Pos_{Pb} = 1$ and $Pos_{Pd} = 3$;
—with this new sliding rule, $\delta_1 = \delta_2 = 0$ because the preceding DSUa and DSUb emit their frames, and $\delta_3 = 1$ because DSUc does not emit;
—then we have:

$$\begin{aligned} TimeSlotPosition_{Pb} &= RefTime + Pos_{Pb} \times \frac{D}{2} + \delta_1 \times \frac{D}{2} \\ &= RefTime + \frac{D}{2} \\ TimeSlotPosition_{Pd} &= RefTime + Pos_{Pd} \times \frac{D}{2} + \delta_1 \times \frac{D}{2} + \delta_2 \times \frac{D}{2} + \delta_3 \times \frac{D}{2} \\ &= RefTime + 4 \times \frac{D}{2} \end{aligned}$$

DSUb slides and emits its frame 0,5 D interval after the reference time instant. On the contrary, DSUd does not slide, it emits 2 D intervals after $RefTime$.

2.6 Reference time positioning mechanism

The preceding mechanisms guaranty a deterministic access to the medium provided that the DSUs are synchronized on the master frame. This is the case in an ideal

world where we do not consider hardware effects of the physical architecture: all the DSUs therefore receive the master frame at the same time and are synchronized. However in our context, we can not consider the propagation time as negligible: if implanted, a wireless network is indeed a small range one but with potentially important propagation time variations, due to absorption coefficient differences within the body. Thus, we have implemented in STIMAP a synchronization mechanism: the reference time mechanism, to synchronize the DSUs at the beginning of the TDMA sequence.

2.6.1 *Basic principles.* The goal of the reference time mechanism is to provide a common reference time for all the DSUs to begin the TDMA-based GSR communication. This communication begins with the emission of a master frame, similar to a beacon, holding a GSR token. As soon as it receives this frame, each node determines its reference time, imposing that it must start the TDMA sequence a constant duration, chosen as a half-interval $\frac{D}{2}$, after the GSR frame has been sent. The idea of the reference time mechanism is based on subtracting the transmission time from controller to slave, estimated to $\frac{RTT_{DSUi}}{2}$, to the previously mentioned common constant. The difference between the instant of time at which the DSUs receives the master frame will then be balanced by the subtraction of the different propagation times.

Let $RefTimeWait_{DSUi}$ be the duration the DSUi has to wait after the GSR reception, $RefTime_{DSUi}$ be the reference time instant for DSUi, and $BeaconTT_{DSUi}$ the transmission time of the beacon frame between the controller and the DSUi. The reference time mechanism is then defined by:

$$RefTimeWait_{DSUi} = \frac{D}{2} - \frac{RTT_{DSUi}}{2} \tag{4}$$

$$RefTime_{DSUi} = BeaconTT_{DSUi} + RefTimeWait_{DSUi} \tag{5}$$

equation 5 et 5 XX PROBLEME

As the $RefTimeWait_{DSUi}$ is a waiting duration, it of course can not has a negative value. The D parameter has to respect the following reference time constraint:

$$D \geq \max RTT_{DSUi} \tag{6}$$

where $\max RTT_{DSUi}$ corresponds to the worst RTT of all slaves.

Example. Figure 3 shows how this mechanism works illustrating it with two DSUs:

—DSU2 receives the master frame first, and begins to wait $RefTimeWait_{DSU2} = \frac{D}{2} - \frac{RTT_{DSU2}}{2}$;

—then DSU1 receives the master frame, and waits $\frac{D}{2} - \frac{RTT_{DSU1}}{2}$.

Supposing that the transmission time of the beacon from controller to DSUi is equal to $\frac{RTT_{DSUi}}{2}$, the reference time is then the same for both DSUs: $RefTime_{DSU1} = \frac{RTT_{DSU1}}{2} + \frac{D}{2} - \frac{RTT_{DSU1}}{2} = \frac{D}{2} = \frac{RTT_{DSU2}}{2} + \frac{D}{2} - \frac{RTT_{DSU2}}{2} = RefTime_{DSU2}$.

This reference time mechanism theoretically guaranties a global reference time for all DSUs even if the propagation times are not the same for all the network members.

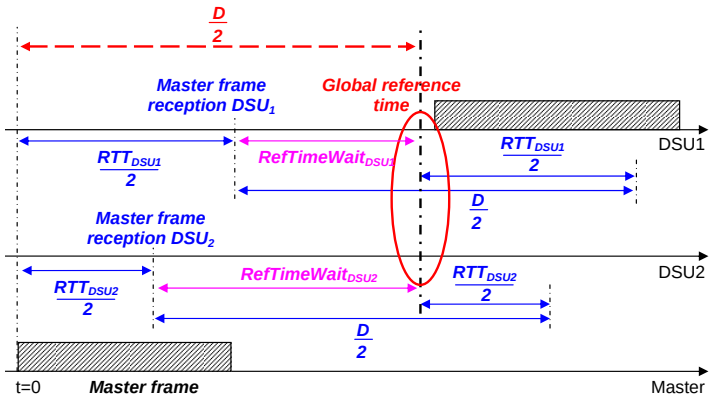


Fig. 3. Basic principle of the reference time mechanism

However this supposes that transmission times between the controller and slaves are equal in both ways (controller to slave and slave to controller), which could be an unrealistic hypothesis.

2.6.2 *Transmission time estimation, for symmetric and asymmetric hardware architecture.* The problem of the reference time mechanism should be based on the real transmission time of the GSR beacon from the controller to each slave. Let us introduce its corresponding variable T^R_{DSUi}. This variable (parameter of the model) must be known to configure the protocol before the running phase. In the previously given reference time definition, this parameter was estimated to $\frac{RTT_{DSUi}}{2}$. Indeed, the RTT parameter can easily be measured on the architecture, counting on the master side the time elapsed between the frame emission to the slave and the reception of the associated answer. In STIMAP, the RTT measurement is performed during the initialization phase, using a DPI mode: the duration of an individual communication is measured for each DSUi and considered as its RTT parameter value (RTT_{DSUi}).

As previously noticed, the hypothesis $T^R_{DSUi} = \frac{RTT_{DSUi}}{2}$ implies a strong constraint on the hardware: it supposes that the network cards have the same performances in reception than in emission. Supposing on the contrary that $T^R_{DSUi} \neq \frac{RTT_{DSUi}}{2}$, the reference time instant could differ depending on the DSUs and then the theoretical equation becomes:

$$RefTime_{DSUi} = T^R_{DSUi} + RefTimeWait_{DSUi} \quad (7)$$

$$= T^R_{DSUi} + \frac{D}{2} - \frac{RTT_{DSUi}}{2} \quad (8)$$

Establishing that the RTT values can potentially be different for the all the DSUs, we can consider two hardware architecture types: the symmetric and the asymmetric ones. In a symmetric hardware architecture, the hardware couplers take the same time to emit and to receive a frame. Then the RTT durations are not necessarily the same for the different DSUs ($RTT_{DSUi} \neq RTT_{DSUj}$) but for each DSUi the transmission time of a frame is the same whatever the direction of the transmission: $T^R_{DSUi} = \frac{RTT_{DSUi}}{2}$. In asymmetric architectures, $T^R_{DSUi} \neq \frac{RTT_{DSUi}}{2}$

and the reference time instants are not the same for each DSUi. Therefore the asymmetric case has to be taken into account in the validation process since this desynchronization impacts the protocol determinism.

However it is difficult to know the real duration of the one-way transmission of a frame in an asynchronous network; this requires specific material and experimentations, and it cannot be performed on our implanted network. Nevertheless, the aim of this study is to take into account the asymmetric case in a formal validation process to identify the limitations of the initial hypothesis.

3. VALIDATION METHODOLOGY

The validation of a system can be performed with different methods. Simulation of a model of the system, as well as test of the system behavior on a prototype, are well-known and effective methods. Simulation has the advantages to allow the execution of big systems, whereas the design of a big system prototype is difficult in a practical and an economical point of view. For communication protocols validation, the simulation tools (OPNET or NS2 [Garrido et al. 2008] for example) offer the possibility to introduce dedicated problems to simulate the desired environment behavior. The system can then be validate in this simulated environment. Similarly, environment changes can be produced to validate the system prototype in a specific context. Fault injection campaigns [Hsueh et al. 1997] are a very common example of this method. Moreover, experimentation on prototype allows to test the implementation process, whereas simulation validate only a system model. But the simulation method, as well as the system test on prototype, are not exhaustive ones. For a given simulation run, not all the possible system states are considered. As the behavior is simulated in a random way, it is not possible to be sure that all the possible behaviors has been explored, even in a numerous simulation campaign. This problem is the same for the system test: the system is tested with a input data scenario, which is not exhaustive. It is not feasible to test all the possible input data values, as there is an infinite number of such scenarios. Furthermore, some specific values can not be tested as they can result from faults impossible to recreate.

Therefore, all the more in some critical contexts, more formal methods are required to complete the validation process. Formal methods are based on mathematical concepts and provide more confidence on the validation results are they are exhaustive ones. This exhaustive validation is done by model checking ([Sifakis 1992; Clarke and Emerson 1982; Berard et al. 2001]): verification of properties on the state space of the system model, i.e. all the possible behaviors. Furthermore, analysis of a model of the system allows detecting errors at an early step of the design process, before the implementation step. Then this validation method is the most appropriate one in our critical context, as this formal approach leads to a more accurate validation of the system reliability.

The methodology we used is a classic one from now for formal validation of communicating systems ([A. David 2000], [Godary et al. 2007], [T. Stauner 1997]). This methodology can be resumed in four main parts. The first step of the validation process is the modeling of the system. It is then necessary to choose a formal language which fits with the system to model and the properties to verify. The

second step consists in abstracting the model to allow the analysis process. Indeed, combinatory explosion is a well-known problem of exhaustive analysis methods. The system model must then be reduced, keeping on only the relevant information. The desired properties must also be modeled, since they can be too complex to be expressed directly in temporal logic. Next, the properties can be verified, and, at last, this validation results must be analyzed to conclude on the system reliability.

In our context, we used model checking on Petri Nets (PN) based system models. PN are a formalism allowing the expression of parallelism, synchronization, resource sharing and concurrency in a simple and natural way. They are then well-adapted for the modeling of distributed and communicating systems. At last, PN are associated to a mathematical formalism from which structural and behavioral analysis can be performed, including the model-checking analysis. Since we deal with non-autonomous systems, i.e. systems that interact with their environment via inputs (signals, sensors, etc.) and outputs (signals, actuators, etc.), we use extensions of PN that permit the description not only of the evolution of the model state but also when this occurs. Thus, we use the temporal extension introduced by [Merlin 1974]: the Time Petri Nets (TPN). TPN allow modeling dense time as intervals associated to transitions. In this article, the validation results have been obtained with the TINA toolbox[1] [Berthomieu et al. 2004]. The system is modeled and simulated using TINA, which also builds the analysis state graph [Berthomieu and Diaz 1991]. Finally, the selt model checker (a part of the TINA toolbox) is used for properties verification.

Time Petri Nets (TPN)

Validating some specific properties, as collision absence, we can be sure that it could never append in the system provided that the system implementation is faithful to the system model. This problem is resolved in our context as the STIMAP model has been implemented on a FPGA based prototype using an automatic VHDL code generator named HILECOP [Andreu et al. 2008]. HILECOP allows the automatic translation of Petri Nets into VHDL components. This guaranties that the properties verified on the system model remain true on the implemented system.

In the specific context of STIMAP, the validation process has two main goals. First, it must of course verify the behavior of all the protocol mechanisms verifying all the necessary properties, as for example the verification of collision absence. But this validation process has a second purpose: improving the efficiency of the protocol, without loosing the reliability. Indeed, one parameter (named D) is the basis of the temporal behavior of STIMAP. Fixing D as a too small value definitively leads to communication problems, as collisions. But fixing a too large D value implies an increase of the communication duration for each DSU and then badly influences the protocol performances. Thus, this paper provides validation results as constraints on the STIMAP parameters which must be respected to verify the desired properties.

[1] www.laas.fr/tina

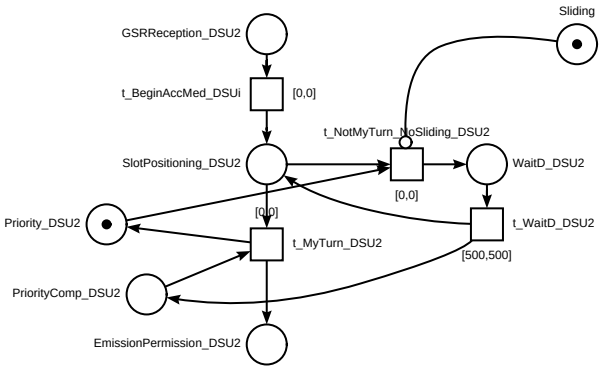


Fig. 4. Basic DSU model: time slot positioning mechanism

4. STIMAP MODEL

The whole system comprises several DSUs communicating with one master. But the combinatory explosion of the model checking method does not permit to analyze a complex model. Since the goal of the validation process is to validate the STIMAP mechanisms concepts, the analysis can be done on a limited number of DSUs. So, the global model we consider in the following only represents three DSUs, the master and the medium of communication. Moreover, models of all these components are abstracted ones: parts of the models that support the mechanisms to be studied have been kept in details whereas parts that do not have any influence have been aggregated. It is for instance the case of the model of the application layer as it does not impact the MAC behavior. We particularly focus on the GSR mode so we represent the emission and the reception of the GSR frame, mentioning to all DSUs if the sliding is activated (`Sliding` place), and in such case the selected sliding rule (`Sliding_Rule` place, see Figure 5).

4.1 DSU model: time slot positioning mechanism

First we only consider the model of the basic medium access strategy of STIMAP: the time slot positioning mechanism in case of group communication (GSR mode) without the sliding time interval mechanism. In this simplified version of the MAC model, given Figure 4, we suppose that the GSR beacon is received at the same time for all DSUs; that corresponds to the symmetric hardware architecture case. In that case, the reference time positioning mechanism is not represented on the model.

This model of the MAC behavior is the same for all the DSU, except for the priority (each DSU having a different priority in the group). The priority mechanism, which is the basis of the TDMA principle, is represented by means of places `priority_DSUi` and `priorityComp_DSUi`. The first place is the priority of the DSU, which practically represents the number of slots the DSU has to wait before emitting. The second place is the complementary one: it represents the number of slots already waited. The DSU which is represented has a priority 1 (the first priority being equal to 0): it has to wait only one D interval before emitting.

The reception of the GSR frame is modeled by the generation of a token in

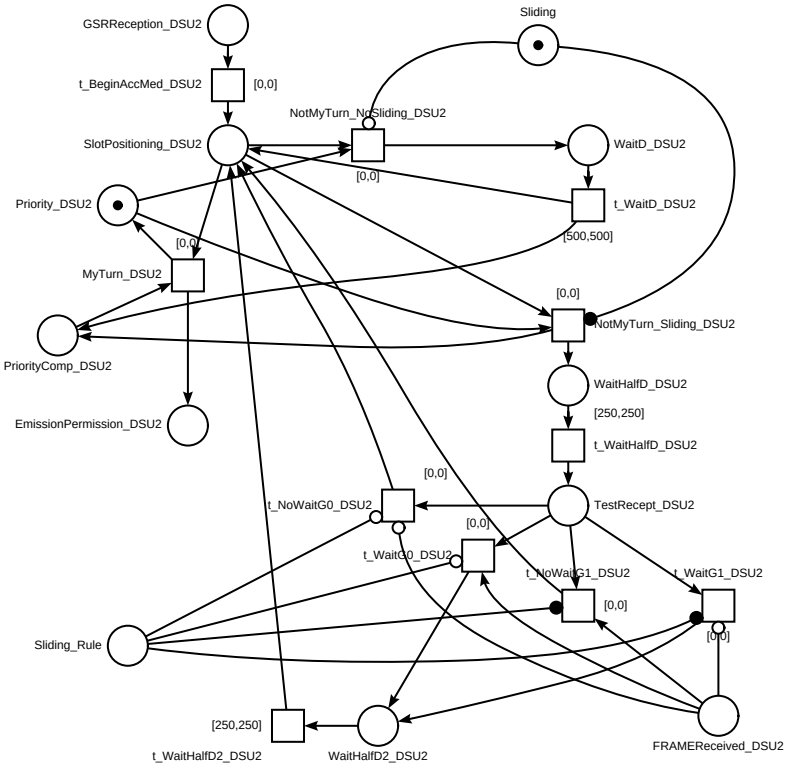


Fig. 5. DSU time slot positioning model with the sliding mechanism

the place `GSRReception_DSUi`. Then the medium access mechanism starts: the transition `t_BeginAccMed_DSUi` is fired and the place `SlotPositioning_DSUi` is marked. In case of no sliding, the `Sliding` place is not marked and then transition `t_NotMyTurn_NoSliding_DSUi` could be fired (inhibitor edge). The fire of the transitions `t_NotMyTurn_NoSliding_DSUi` or `t_MyTurn_DSUi` depends on the priority of the DSU and on the number of already waited slots. If there is at least one token in the `priorityComp_DSUi` place, transition `t_NotMyTurn_NoSliding_DSUi` is fired and the DSU waits a D interval (here $D = 500$ time units). On the contrary, if the DSU has already waited all the expected slots, the DSU can access to the medium: transition `t_MyTurn_DSUi` is fired and the place `t_EmissionPermission_DSUi` is marked, representing the permission to emit. The behavior of the frame emission will be described section 4.3.

4.2 DSU model: sliding time interval mechanism

We now expose the model of the time slot positioning with the sliding mechanism (Figure 5). In case of sliding (place `Sliding` marked), the DSU slot D is shared into two half-intervals. If its not its turn to emit, the DSU has to wait: transition `NotMyTurn_Sliding_DSU2` is fired. After waiting the first half-interval (transition `t_Wait_HalfD_DSUi`), the DSU verifies (in the `TestRecept_DSUi` place) if it can slide or not, depending on the sliding rule and the reception (or not) of

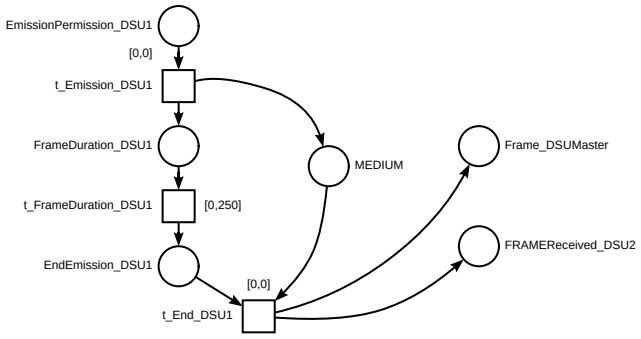


Fig. 6. Model of the medium

the preceding DSU frame. The reception of a frame is modeled by a token in place `FRAMEReceived_DSU2`. In our context we suppose that it is not possible to detect a frame which is being emitted before the end of its emission. Then the MAC level considers there is no frame on the medium until a complete frame is received. Supposing that the sliding rule is "used time interval" (`Sliding_Rule` is marked):

—if the DSU has received the preceding frame, it slides: `t_NoWaitG1_DSUi` is fired and the DSU returns in the `SlotPositioning_DSUi` place.

—if not, `t_WaitG1_DSUi` is fired and the DSU waits one more half-interval.

4.3 Medium model

The medium is modeled in a simple way, using only one place to represent its occupation. This representation is well-adapted to represent a shared medium where all the frames are sent in broadcast (like on wired bus as non-switched Ethernet). For wireless medium, it depends on the broadcast range: we suppose here that all nodes are reachable. Modeling the medium in a global way is a convenient over-approximation: if there is no collision detected on the global medium model, we can be sure that collision will not actually occur.

Figure 6 shows the medium model and its relation with DSU1: when DSU1 emits on the medium (transition `t_Emission_DSU1`), the place `MEDIUM` becomes marked representing that the medium is occupied. At the end of the frame emission (transition `t_End_DSU1`), this token is consumed and a token is generated at the MAC level of the receivers (the DSU2 and the Master for instance), representing the complete reception of a frame. This model supposes that all the DSUs (and the master) receive the emitted frame at the same time. This is an over-approximation of the reality, since all nodes do not necessarily receive the frame simultaneously (in the model they are all as slow as the slowest).

4.4 DSU model: Reference time positioning mechanism

In the symmetric architecture case, the reference time is supposed to provide a global synchronization time to all DSUs (see section 2.6). To model that, we just have to integrate a waiting duration after the beacon reception (GSR master frame). On the model the transition `t_BeginMedAcc_DSUi` becomes the transition

`t_RefTime_DSUi` to which is associated a fixed time interval equal to $\frac{D}{2}$.

In the asymmetric architecture case, the reference time mechanism has to be entirely modeled, and it is quite difficult to do so using Petri nets. Indeed, the reference time is not the same for all DSUs, as it depends on the value of the T^E_{DSUi} and T^R_{DSUi} parameters. The basic idea to model the reference time is to separately integrate T^R_{DSUi} and TE_{DSUi} durations: the first one is taken into account in the $RefTimeWait_{DSUi}$ duration whereas the latter is taken into account in the $FrameDuration_{DSUi}$ duration. Of course the $RefTimeWait_{DSUi}$ calculation must be done in an off-line step, since in the Tina tool the fired interval of transitions can only be integer values (calculation are not possible). But this solution provides a model for fixed values of T^E_{DSUi} and T^R_{DSUi}. Since we want to study all the possible cases of the asymmetry, these values must not be constant ones: we want to represent them as random values taken in a given interval. Representing T^R_{DSUi} as a random value in a given interval, it is so impossible to separately represent the T^R_{DSUi} and $RefTimeWait_{DSUi}$ parameters. Indeed, the calculation of the $RefTimeWait_{DSUi}$ value depends on the exact value of T^R_{DSUi} and should then be dynamically determined taking into account the exact firing date of the transition to which T^R_{DSUi} has been associated. But the dynamic calculation of a transition fired interval is impossible. We resolve this problem representing the reference time instant itself as a random duration in a time interval defined by $[\min RefTime, \max RefTime]$ and associated to transition `t_RefTime_DSUi`. This interval is calculated off-line and represents all the possible reference time values for the given T^R_{DSUi} possible values. However this solution does not resolve the second mentioned problem: the T^E_{DSUi} value is also dependent on the exacte value of T^R_{DSUi} (since $T^E_{DSUi} + T^R_{DSUi} = RTT_{DSUi}$). This problem will be discussed in section 6.

5. STIMAP FORMAL VALIDATION FOR A SYMMETRIC ARCHITECTURE

This section concerns the STIMAP validation process in a symmetric architecture case: the transmission duration of a frame (included the emission and reception time) is the same between the controller and the DSU than between the DSU and the controller. It then can be estimated with the RTT measurement (see section 2.6.2). The reference time mechanism then provides a global reference time which is the same for all DSUs and permits the DSUs to be synchronized .

The validation process concerns of course different properties, like the time-slot positioning and the respect of the emission order of the GSR mode. This article focuses on the "no collision" property. This property is verified if there is no more than one token in the place `MEDIUM` of the medium model (figure 6). The result of this verification is dependent on the relation between the `D` parameter and the `FrameDuration_DSUi` and `AnswerDuration_DSUi` ones. `FrameDuration_DSUi` is the duration of the DSUi frame, whereas `AnswerDuration_DSUi` is the duration of the master answer frame (when it reacts and so answers in the dedicated DSUi slot), including frame treatment duration at the master application level. The validation has been done for each possible scenario: no sliding, sliding in case of used or unused interval, with or without DSU frame emission. This work is the continuation of the one presented in [Godary et al. 2007], which presents the validation results of a

Table I. STIMAP constraints in a symmetric architecture case

Sliding rule	**No-collision Constraints**
No sliding	$D > FrameDuration_{DSUi} \quad \forall i$
Sliding : unused	$\begin{cases} \frac{D}{2} > FrameDuration_{DSUi} \quad \forall i \\ D > FrameDuration_{DSUi} + AnswerDuration_{DSUi} \quad \forall i \end{cases}$
Sliding : used	$\begin{cases} \frac{D}{2} > FrameDuration_{DSUi} \quad \forall i \\ \frac{D}{2} > AnswerDuration_{DSUi} \quad \forall i \end{cases}$

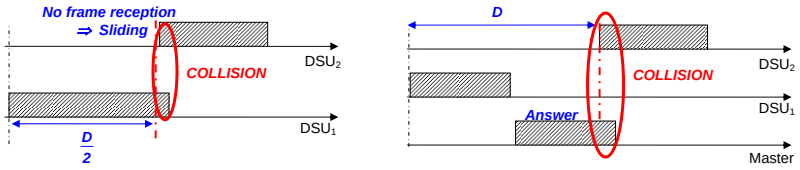


Fig. 7. Collision examples with sliding in case of unused interval

very simple model: the time slot positioning without sliding, assuming a symmetric architecture. This first work permitted to verify basic concepts of the TDMA-based GSR communication. We want now to study and validate the full mechanism of STIMAP.

Setting the `D` parameter value, we find using dichotomy the maximal values for the parameters `FrameDuration_DSUi` and `AnswerDuration_DSUi` for which no collision can occur. We thus extract some basic constraints that the parametering of STIMAP has to respect to ensure a collision-free medium access in a symmetric hardware architecture case. These constraints confirm the theoretical behavior of STIMAP. They are resumed in Table I and explained in the following.

No sliding. In this case, the only condition is that the DSU must have finished to emit at the end of the D interval: $FrameDuration_{DSUi}$ must be shorter than D, for all DSUs.

Sliding in case of unused time interval. In this case, the master can answer to the DSU frame, then a DSU must not slide if there is a DSU emission in its preceding slot. For this sliding rule, two constraints have to be respected. First, as illustrated at the left of Figure 7, if the duration of the DSU1 frame is longer than $\frac{D}{2}$, the next node (DSU2) and the master do not received this frame before the end of the first half-interval. Then they suppose that no frame has been emitted: the master does not answer and DSU2 slides and begins to emit its frame, provoking a collision with that of DSU1. The second condition concerns the master answer: if a DSU emits a frame and the master answers to it (see the right of Figure 7), the next DSU does not slide and waits the end of the D interval. So, the master frame must be finished before the end of the D interval. Then the sum of the DSU frame duration and the answer duration must be lower than the D parameter. If the first DSU does not emit its frame, the master do not answer, the next DSU slides, there is no additional condition.

Sliding in case of used time interval. In this case, the master answers if there is a passive DSU. In this sliding rule, two constraints must be respected: first if the DSU1 emits a too long frame (greater than $\frac{D}{2}$), the master supposes there is no

frame and begins to answer, provoking a collision with the DSU1 frame. Second, if the DSU1 does not emit its frame, the master answers in the second half interval: this frame must be finished before the DSU2 begins to emit in its time-slot, then the answer duration must be lower than $\frac{D}{2}$.

Concluding on these validation results, we can say that the behavior of the medium access protocol STIMAP is valid (i.e. it verifies all the desired properties) provided that the D parameter respects the conditions resumed table I. However this validation has been done assuming the hypothesis that all the DSUs have the same global reference time (symmetric architecture case). Thus the previous validation results are dependent on the reference time values of each DSU. So we have to study this mechanism and extend the validation process to the asymmetric architecture case. This validation is presented in the next section.

6. STIMAP FORMAL VALIDATION FOR AN ASYMMETRIC ARCHITECTURE

An experimental series of measurements with a platform based on Ethernet and RF technology [Andreu et al. 2008], shows that some hardware architectures can imply important variations between the different RTT durations, but most of all these experimental measurements prove that it is possible to have significant differences between the transmission duration from the master to a DSU (named T^R_{DSUi}) and the one from this DSU to the master (named T^E_{DSUi}). This means that the system is not a perfect one and that the hypothesis on the synchronization of the DSUs must be reconsidered: a system is neither a perfect nor a symmetric one.

We have seen that the reference time mechanism in a symmetric hardware provides a global reference time on which all the DSUs are synchronized. An asymmetric hardware architecture, on the contrary, provokes a gap between the reference times of the different DSUs: they are not synchronized anymore. We then have to study the effect of the asymmetric hardware on the medium access mechanism, principally the implied collision risks.

6.1 Values of the model parameters

On the STIMAP model, we fix the STIMAP model parameters, respecting the basic constraints of the protocol verified in the preceding section, and adding the asymmetric hypothesis.

We have to use the two parameters T^R_{DSUi} and T^E_{DSUi} in the STIMAP model to represent an asymmetric system. But as said section 4.4, we can not represent dynamic calculations on the fired interval of the transitions.

For the reference time parameter, we can represent it as a random duration in a $[RefTime^{MIN}, RefTime^{MAX}]$ interval for all DSUs. We then have to calculate these two bounds, depending on the asymmetric hypotheses. Indeed, experimental results shows that the asymmetric is always in the same way for a given technology. For example, Ethernet hardware couplers are always faster to emits than to receive, whereas the situation is inverse for a RF technology. The validation process of STIMAP for an asymmetric hardware architecture is then decomposed in two steps: on the fast emission hypothesis, i.e. when the DSU is faster in emission than in reception ($T^E_{DSUi} < T^R_{DSUi} \ \ \forall i$), and with the opposite slow emission hypothesis.

But we also have to consider the relationship between the three parameters

T^E_{DSUi}, T^R_{DSUi} and RTT_{DSUi}. We want to explore all the possibilities of the parameters values. First, the worst asymmetric solution is when T^E_{DSUi} and T^R_{DSUi} are the most different, for example, considering the $TE < TR$ hypothesis, if $TE = 0$ and $TR = RTT$. We can then supposed that $T^E_{DSUi} \in [0, \frac{RTT_{DSUi}}{2}]$ and $T^R_{DSUi} \in [\frac{RTT_{DSUi}}{2}, RTT_{DSUi}]$. However, we also have to respect that $TE + TR = RTT$. Then in the preceding given intervals, the TE and TR values are not independent one from the other. This constraint can not be express on the model and have to be verified analyzing the verification results.

Then we have to fix the D parameter value. We want to fix it as the best value, which is the most inferior one to increase the protocol efficiency, but respecting all the protocol constraints: the ones of table I, and also the reference time constraint of equation (6): $D \geq RTT_{DSUi} \quad \forall i$. The D value will depend on the direction of the asymmetry, as we will see in the two next sections.

We also suppose that $FrameDuration_{DSUi} = T^E_{DSUi}$. In fact, this first parameter represents the medium occupation duration when DSUi emits a frame, whereas T^E_{DSUi} also includes the reception duration of the master. Therefore the medium occupation duration is an over-approximation of the reality. Supposing that, our validation results will be good even if the reception duration of the master is null.

6.1.1 *Parameters values for a fast emission architecture: $T^E_{DSUi} < T^R_{DSUi}$.* The hypothesis that $T^E_{DSUi} < T^R_{DSUi}$ can be traduced in two intervals for those parameters values:

$$T^E_{DSUi} \in [0, \frac{RTT_{DSUi}}{2}] \quad \text{and} \quad T^R_{DSUi} \in [\frac{RTT_{DSUi}}{2}, RTT_{DSUi}]. \tag{9}$$

In this hypothesis, as $FrameDuration_{DSUi} = T^E_{DSUi}$ and $T^E_{DSUi} \in [0, \frac{RTT_{DSUi}}{2}]$, then we can fix $D = \max RTT_{DSUi}$.

Then, knowing D and considering the reference time theoretical equation 2.6.2 we have:

$$\begin{aligned} RefTime_{DSUi} &= T^R_{DSUi} + \frac{D}{2} - \frac{RTT_{DSUi}}{2} \\ &= T^R_{DSUi} + \frac{D}{2} - \frac{T^E_{DSUi} + T^R_{DSUi}}{2} \end{aligned}$$

We can then explain the `RefTime_DSUi` parameter as:

$$RefTime_{DSUi} = \frac{D}{2} + \frac{T^R_{DSUi} - T^E_{DSUi}}{2} \tag{10}$$

Based on these conclusions, we can calculate the maximal and minimal values of the `RefTime_DSUi` parameter.

Table II. Parameters values for asymmetric architectures

$T^E_{DSUi} < T^R_{DSUi}$	$T^E_{DSUi} > T^R_{DSUi}$
$D = \max RTT_{DSUi}$	$D = 2 \times \max T^E_{DSUi} = 2 \times \max RTT_{DSUi}$
$FrameDuration_{DSUi} = T^E_{DSUi}$	$FrameDuration_{DSUi} = T^E_{DSUi}$
$T^E_{DSUi} \in [0, \frac{RTT_{DSUi}}{2}]$	$T^E_{DSUi} \in [\frac{RTT_{DSUi}}{2}, RTT_{DSUi}]$
$T^R_{DSUi} \in [\frac{RTT_{DSUi}}{2}, RTT_{DSUi}]$	$T^R_{DSUi} \in [0, \frac{RTT_{DSUi}}{2}]$
$RefTime \in [\frac{\max RTT_{DSUi}}{2}, \max RTT_{DSUi}]$	$RefTime \in [\frac{\max RTT_{DSUi}}{2}, \max RTT_{DSUi}]$

$$\begin{aligned}
\min RefTime_{DSUi} &= \frac{D}{2} + \min\left(\frac{T^R_{DSUi} - T^E_{DSUi}}{2}\right) \\
&= \frac{D}{2} + \frac{\min T^R_{DSUi} - \max T^E_{DSUi}}{2} \\
&= \frac{D}{2} + \frac{\frac{RTT_{DSUi}}{2} - \frac{RTT_{DSUi}}{2}}{2} \\
\min RefTime_{DSUi} &= \frac{D}{2} = \frac{\max RTT_{DSUi}}{2}
\end{aligned}$$

$$\begin{aligned}
\max RefTime_{DSUi} &= \frac{D}{2} + \max\left(\frac{T^R_{DSUi} - T^E_{DSUi}}{2}\right) \\
&= \frac{D}{2} + \frac{\max T^R_{DSUi} - \min T^E_{DSUi}}{2} \\
&= \frac{D}{2} + \frac{\max RTT_{DSUi} - 0}{2} \\
\max RefTime_{DSUi} &= \max RTT_{DSUi}
\end{aligned}$$

All these parameters values are resumed table II.

6.1.2 *Parameters values for a slow emission architecture:* $T^E_{DSUi} > T^R_{DSUi}$. In this hypothesis, T^E_{DSUi} is always bigger than T^R_{DSUi}. We can traduce it with:

$$T^E_{DSUi} \in [\frac{RTT_{DSUi}}{2}, RTT_{DSUi}] \quad \text{and} \quad T^R_{DSUi} \in [0, \frac{RTT_{DSUi}}{2}]. \tag{11}$$

In that case, the most restrictive constraint for the D value is: $D > 2 \times FrameDuration_{DSUi}$. Then we fix $D = 2 \times \max RTT_{DSUi}$, and the reference time interval becomes: $RefTime_{DSUi} \in [\frac{\max RTT_{DSUi}}{2}, \max RTT_{DSUi}]$.

6.2 Analysis of one collision situation

6.2.1 *Collision scenario.* Verifying the STIMAP model for asymmetric architecture, with the fixed parameters of table II, we detect that collisions are possible. In this section we will precisely analyze a situation which lead to a collision, with sliding in case of unused interval. The execution of this scenario is shown Figure 8, and its detailed behavior is:

—$t = 300\mu s$: the DSU2's reference time is $300\mu s$; as DSU2 is not the first DSU of the GSR sequence, it waits a half time-interval $\frac{D}{2}$ before emitting.

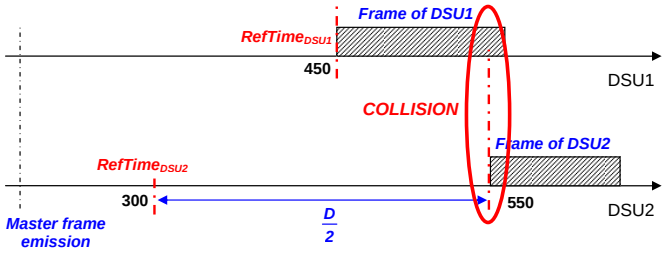


Fig. 8. Collision example in an asymmetric architecture case, with "used time interval" sliding rule

—$t = 450\mu s$: the DSU1's reference time is $450\mu s$; as DSU1 is the first DSU to emit, it begins to emit its frame.

—$t = 550\mu s$: DSU2 has waiting its first $\frac{D}{2}$ interval, it has to decided if it must slid or not; if the frame of DSU1 is longer than $= 100\mu s$, this frame emission is not finished and DSU2 considers that it has not received a frame. With the "unused time interval" sliding rule, DSU2 then slides and begins to emit its frame, ***provoking a COLLISION***.

This scenario seems realistic, but we have to study the specific values of T^E_{DSui} and T^R_{DSui} parameters to verify if they respect their constraint. First, if $450 = RefTime_{DSU1} = \frac{D}{2} + \frac{T^R_{DSU1} - T^E_{DSU1}}{2} = 250 + \frac{T^R_{DSU1} - T^E_{DSU1}}{2}$ then we have:

$$T^R_{DSU1} - T^E_{DSU1} = 400$$

In the same time, we must have:

$$T^R_{DSU1} + T^E_{DSU1} \leq \max RTT_{DSU1} = 500$$

So if the reference time value if DSU1 is $450\mu s$, we must have a T^E_{DSU1} value inferior to $50\mu s$. But the collision occurs only if $T^E_{DSU1} > 100\mu s$.

Then this collision situation is not a real one, it corresponds to one of the over-estimated states of the system.

6.2.2 *Generalization.* Figure 8 shows that **to generate a collision** with the sliding mechanism in case of unused interval, the hardware values must respect the following equations:

$$\begin{aligned}
RefTime_{DSU1} + FrameDuration_{DSU1} &> RefTime_{DSU2} + \frac{D}{2} \\
RefTime_{DSU1} + T^E_{DSU1} &> RefTime_{DSU2} + \frac{D}{2} \\
\frac{D}{2} + \frac{T^R_{DSU1} - T^E_{DSU1}}{2} + T^E_{DSU1} &> \frac{D}{2} + \frac{T^R_{DSU2} - T^E_{DSU2}}{2} + \frac{D}{2} \\
\frac{T^R_{DSU1} + T^E_{DSU1}}{2} &> \frac{T^R_{DSU2} - T^E_{DSU2}}{2} + \frac{D}{2}
\end{aligned}$$

In summary a collision can occur if the hardware parameters respect:

$$(T^R_{DSU1} + T^E_{DSU1}) - (T^R_{DSU2} - T^E_{DSU2}) > D \tag{12}$$

Calculating the maximal value of the left term with our hypothesis, we have: $\max\left((T^R_{DSU1} + T^E_{DSU1}) - (T^R_{DSU1} - T^E_{DSU1})\right) = \max\left(T^R_{DSU1} + T^E_{DSU1}\right) - \min\left(T^R_{DSU1} - T^E_{DSU1}\right) = \max(RTT_{DSU1}) - 0$.

Then, as in the fast emission hypothesis $D = \max RTT_{DSUi}$, the left term could never be superior to D, and then ***a collision could never occurred for an asymmetric architecture with sliding in case of unused interval.***

6.3 Other validation results and STIMAP constraints summary

The preceding section analyses the results of the no-collision verification with sliding in case of unused interval, with the fast emission hypothesis. This analysis shows that in fact, the collision situations detected with the model checking validation correspond to some unrealistic states of the system, provided that the D parameter respects $D > \max RTT_{DSUi}$. Similar analyses of all the no-collision verification results (for each sliding rules, for both fast and slow emission hypotheses) shows that collisions are never possible, but for the slow emission hardware architecture the maximal D value must be $D > 2 \times \max RTT_{DSUi}$.

Table I finally reminds all the constraints for the STIMAP parameters, for both symmetric and asymmetric hardware. But all these constraints can be resumed in the following one, which includes all the others:

$$D > 2 \times \max RTT_{DSUi} \tag{13}$$

7. CONCLUSION

This article deals with the validation of STIMAP, a deterministic medium access protocol. This protocol has been designed to meet the specific requirements of a FES application. Thus the protocol must fit with real-time and reliability constraints, as well as embedded ones. It must be reactive, determinist, reliable, with a simple and light implementation. Thus, STIMAP is based on a TDMA group communication with a sliding mechanism to improve the efficiency of the classic TDMA approach. This approach is mixed with a master/slave approach to initiate a TDMA communication for one group of nodes. In such a critical context, we propose to complete classical validation methods as simulation and prototype experimentation with a formal validation process, which provides more confident validation results.

This article presents the Time Petri Nets (TPN) STIMAP model and its validation, focusing on the no-collision property verification. The STIMAP validation is proved with an analysis of model checking results. The TPN formalism has been chosen to guarantee the implementation process: the validated model (in fact, an abstracted one) is directly implemented, automatically generating VHDL code for a FPGA target with the HILECOP tool. Moreover, the TPN formalism fits well with our modeling needs. However, the timed model checking for TPN is not as developed as we need. First, the verification process often implies the research of an a priori unknown duration value. It is then necessary to make several verification runs to find the suitable value. But the parameterized model checking is not yet mature, especially for the TPN formalism, so it can not be used. Second, the STIMAP validation is confronted to an even more complex modeling and verifica-

tion problem: the fired interval of one transition is a calculation which depends of the fired date of another transition. This fired date is dependant on the hardware used architecture, and the formal validation have to study all the possible cases to guarantee the protocol behavior. But this dynamical construction of the state space graph of a model is not resolved for now. This paper then presents a validation of an over-approximation of the real possible states of the system, then a precise analysis of the results to conclude for the real states. So, the whole validation process allows extracting constraints which must be verified to guarantee the STIMAP behavior. We conclude saying that, provided that these constraints are respected, the STIMAP protocol has a correct behavior for whatever hardware architectures.

This study also shows that the model checking is not still fully well adapted to the modeling and the validation of real systems. For example, the STIMAP validation has been done without changing the basic mechanisms of the protocol, as the synchronization one. Parameterized model checking, as well as (dynamic) calculation of transitions fired intervals, could help to find the most suitable constraint for the reference time value. Therefore, an interesting continuation of this work should be the study and the improvement of the interface between the theoretical formal methods, and their associated tools, with the actual needs of the systems validation. Especially in specific contexts as embedded ones, where real-time constraints and hardware parameters have to be considered during the validation process. This work has already begun with the development of a validation tool: LPT (Little Parametric Tool) [Godary 2008], which allows the parameterized verification of the maximal execution time between two transitions using automatic dichotomy. This work should be deepened to provide a useful and suitable formal validation tool. In the same idea, we work on the interfacing between the HILECOP tool and this new validation tool, in order to closely link the system design process and the formal validation one.

REFERENCES

A. David, Y. W. 2000. Modelling and analysis of a commercial field bus protocol. In *Proc. of 12th Euromicro Conference on Real-Time Systems (ECRTS'00)*. Stockholm, Sweden.

Andreu, D., Guiraud, D., and Souquet, G. 2009. A distributed implantable architecture for activating the peripheral nervous system. *Journal of Neural Engineering 16,* 6.

Andreu, D., Souquet, G., and Gil, T. 2008. Petri net based rapid prototyping of digital complex system. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI08)*. Montpellier, France.

Berard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., and Schnoebelen, P. 2001. *Systems and software verification: model-checking techniques and tools*. Springer-Verlag New York, Inc., New York, NY, USA.

Berthomieu, B. and Diaz, M. 1991. Modeling and verification of time dependent systems using time petri nets. *IEEE Transactions on Software Engineering 17,* 3, 259–273.

Berthomieu, B., Ribet, P.-O., and Vernadat, F. 2004. The tool tina – construction of abstract state spaces for petri nets and time petri nets. *International Journal of Production Research 42,* 14.

Clarke, E. M. and Emerson, E. A. 1982. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs, Workshop*. Springer-Verlag, London, UK, 52–71.

Garrido, P. P., Malumbres, M. P., and Calafate, C. T. 2008. ns-2 vs. opnet: a comparative study of the ieee 802.11e technology on manet environments. In *Simutools '08: Proceedings*

of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, 1–10.

Godary, K. 2008. Lpt : Little parametric tool, outil pour la validation d'une borne temporelle paramtre. In *6ime Confrence Internationale Francophone dAutomatique (CIFA'08)*. Bucarest, Roumanie.

Godary, K., Andreu, D., and Souquet, G. 2007. Sliding time interval based mac protocol and its temporal validation. In *Proceedings of the 7th IFAC International Conference On FieldBuses & Networks in Industrial & Embedded Ssytems*. 119–126.

Hsueh, M.-C., Tsai, T. K., and Iyer, R. K. 1997. Fault injection techniques and tools. *Computer 30,* 4, 75–82.

Merlin, P. 1974. A study of the recoverability of computing systems. Ph.D. thesis, Department of Information and Computer Science, University of California, Irvine, CA.

Sifakis, J. 1992. A unified approach for studying the properties of transition systems. *Theoretical Computer Science 18,* 3, 227–258.

Souquet, G., Andreu, D., and Guiraud, D. 2007. Intrabody network for advanced and efficient functional electrical stimulation. In *9th International Workshop on Functional Electrial Stimulation (FES'07)*. Vienne, Austria.

T. Stauner, O. Mller, M. F. 1997. Using hytech to verify an automotive control system. In *Proc. Hybrid and Real-Time Systems (HART'97)*. Grenoble, France.

Received ; revised ; accepted ;