Information Societies Technology (IST) Programme

Future and Emerging Technologies - Open Domain

Project acronym: Advance
Project full title:

Advanced Validation Techniques
for Telecommunication Protocols

Contract no: IST-1999-29082

Deliverable 18

Case studies

PGM protocol
IEEE-1394 root contention protocol

October 2001 - January 2004

Date of preparation: January 2004

Contributors: Paris, France-Telecom, Liège, Grenoble, Edinburgh

Contents

1 Introduction

The Advance project has begun the 1st of October 2000 for a duration of three years. Its goal is to develop a new generation of verification tools for telecommunication protocols. Those new tools should be able to tackle with problems like the resolution of timing constraints, unbounded numbers of data packets, unrestricted architectures and others. Since those problems comprise infinite components, previous verification tools based on the enumeration of all possibilities are unable to resolve those problems.

The project has a strong commitment to produce tools which could have a real impact on actual development methods in the telecommunication industry. It comprises mainly academic teams (from University of Edimburg, University of Liège, University of Paris VI, University of Uppsala, Verimag and Weizmann Institute) but also teams from industrial R & D centers (Ericsson and France Telecom R & D). A significant part of the project is to validate the new tools by case studies upon problems big enough for ensuring that the tools would be useful later on real-size projects.

This report gives a description of the work done during the in two case studies: the PGM protocol and the IEEE1394 route contention protocol.

- The main case study that was considered in the Advance project is the PGM protocol. This case study was submitted by France Telecom R & D. The work around this protocol was performed in the two last years of the project. In the second year of the project, an important work has been done in modelling the protocol at different level of abstractions and in analysing the protocol by taking into account different aspects and sources of complexity (topologies, arbitrary number of components, unbounded number of packets, constraints on various parameters such sizes of buffers and sender/receivers windows, delays, rates of losses, etc). After the end of this second year, tools developed in the project (IF-toolbox, TReX, and LASH) were able to handle all these aspects, except that they were unable to compute automatically the (quite complex) constraints on the various parameters of the protcols which guarantee correction w.r.t. some properties (we have been actually able to perform a parametric analysis of the protocol, but this analysis was not fully automatic). Therefore, this problem was left as a challenge for the third year of the project. After some improvements in TReX and some modelling effort, we were able to perform automatically, for the first time, a quite non trivial parametric analysis of this complex protocol.

  This report includes the relevant parts of the deliverable of the second year concerning PGM, and adds the new results about the study of PGM.

- The IEEE 1394 Root Contention Protocol is another example of complex protocols for which wa have been able to perform an automatic parametric analysis. Actually, the parametric analysis of this protocols requires the manipulation of heterogeneous models manipulating both integer counters and real-valued clocks. In the first year of the project, a parametric analysis of this protocol was performed for the first time using the tool TReX [38]. This was possible since TReX uses representation structures and analysis techniques which can handle such heterogeneous parametric models. The tool LASH was at that time unable to perform such an analysis since only integer counters were available in its input models. In the third year of the project, new representation structures and analysis techniques were implemented in LASH allowing to handle the models in the needed class, and successful experiments on the RCP protocol have been performed using these techniques.

- Last, a criticism that can be made to formal methods is that they don't work directly on complete software but on simplified models of software because they are not able to cope with the complexity of complete systems. We believe that the progresses made recently in the field of formal methods, including those brought out by the Advance project, will enable to treat real source code in the years to come, including all their potential infinite behaviours. In this context, France Telecom R & D has, in the last year of the project, begun to work on the verification of security properties of Java libraries, and has been able to check some real size software. We will describe shortly those new results in this report.

The main case study of the Advance project was around the PGM protocol. The problem was submitted by France Telecom R & D. It consists in validating the PGM protocol, a multicast communication protocol designed by the IETF [1]. France Telecom, as an operator of telecommunication networks, has a natural interest in studying the behaviour and performances of protocols which could be deployed in the future, and are proposed by network equipment builders, like Cisco. This study has been done by France Telecom R & D by designing a SDL[2] model of the protocol. This model enabled France Telecom to simulate the behaviour of the protocol, but it didn't make it possible to prove properties of PGM, because this specification contains infinite parts, which can not be treated by the tools around the SDL language.

The sources of infinite behaviour in PGM, and more generally in telecommunication protocols come from several ways, but three of them seem to us particularly important to tackle because of their ubiquity in telecommunication protocols.

- Telecommunication protocols describe how to send data packets from a source to one or several receivers. The first source of infinity (or, more precisely, of intractable behaviours) comes from the number of those data packets. In PGM, the protocol numbers each data packets, and PGM can send 2^{32} different packets before coming back to the initial numbering. This implies that an exhaustive study of the protocol is out of the bounds of classical tools which enumerate all possibilities. It is necessary to use tools which treat the maximum number of packets as a parameter, and can then derive properties of the protocol whatever value this parameter is.

- A second source for infinite behaviour is that communications protocols are designed to work in different configurations. PGM is a multicast protocol, that is, its configuration is tree-like: from one source, data are transmitted through several levels of nodes to receivers, which can be seen as tree leaves. Again, new methods have to be defined for deriving properties of the protocol which are true whatever the topology is.

- Last, telecommunication protocols make often use of timers. For example, PGM use timers for managing its policy of asking reemissions of lost data packets. And one of the goals of PGM is to ensure that all losses can be recovered. This property depends on the relations between the timers used in the protocol and the delays introduced in the transfer of data packets. Since the timers can take any value, an interesting goal is to derive constraints between those values for ensuring that desired properties are fulfilled. This is what we have called earlier the constraint synthesis problem.

The two first subjects have been studied in the second year of the project, whereas in the last year we looked mainly the constraint synthesis problem, in the context of the PGM protocol, but also of the RCP protocol.

The work pursued in the Advance project has followed the lines described here. A first work was to model the PGM protocol in the language IF used and defined in the project. An SDL description given by France Telecom R & D served as a starting point, together with the IETF informal standards. This work was mainly done by the teams from University of Paris 7 and VERIMAG, with support from France Telecom R & D, and it has influenced the definition of the new version of the IF language. This modeling work is described in the next section by Marc Boyer from Paris 7, with some description by Marius Bozga of VERIMAG of the design of IF 2.0. This work was done in the second year of the project.

Section 3 describes the work which has been accomplished using "classical" verification tools upon PGM. This work was done at University of Paris 7 in the second year of the project. Section 4 concerns the automatic synthesis using TReX of constraints between the parameters of the PGM protocol which guarantee that all messages will be recovered. This work was done by Paris 7 in the third year of the project.

Section 5 describes the use of the LASH tool from University of Liège in the verification of LASH. The tool enables to cope with the parametrization of the number of data packets.

Section 6 is a contribution by the University of Edinburgh showing how to prove that properties can be true for families of different topologies of the PGM protocol when they are true for a simple representent of such a family. This work has been performed in the second year of the project.

[1] Internet Engineering Task Force, *i.e.* the body responsible for standardizing the communication protocols of the Internet.

[2] Specification and Description Language, a standardized language defined by the International Telecommunications Union for describing telecommunication systems and protocols.

The next two parts of the report are dedicated to the constraint synthesis problem for the IEEE RCP. We first recall a study of the RCP protocol, using the TReXtool, which was done during the first year of the project [38] by Paris 7. Then, we present the work performed by Liège in the third year of the project concerning the parametric analysis of this protocol using LASH..

The last section has been written by Yves-Marie Quemener of France Telecom R & D. It is a short note about new studies trying to apply advanced model-checking techniques like those developed in the Advance project to the verification of properties of source code. The conclusion tries to infer some information about the usability of the various tools and techniques, using inputs from the different users, and to give some perspective on the future of formal methods tools.

Yves-Marie Quemener from France Telecom R & D served as editor for this report.

Acknowledgement The researchers from the ADVANCE project who participated to the work on the PGM case study are: Ahmed Bouajjani, Marc Boyer, Petr Matusek, and Mihaela Sighireanu from Paris, Yves-Marie Quemener from France-Telecom, Bernard Boigelot and Louis Latour from Liège, Marius Bozga and Yassine Lakhnech from Grenoble, Javier Esparza and Monika Maidl from Edinburgh.

2 The PGM protocol (year 2)

The Pragmatic General Multicast (PGM) protocol is "a reliable multicast transport protocol for applications that require ordered or unordered, duplicate-free, multicast data delivery from multiple sources to multiple receivers" [84, Abstract, p. 1]. It is an "Experimental Protocol" [84, Status of this Memo, p. 1] whose "central design goal is simplicity of operation with due regard for scalability and network efficiency" designed for "multicast applications with basic reliability requirement" [84, Abstract, p. 1].

One specific aspect of this protocol is that it is "NAK-oriented", *i.e.* the reliability mechanism is not assured by acknowledging the received messages, but by sending a negative acknowledgment (`NAK`) when a message seems to be lost. The idea between this mechanism is to reduce the number of control messages respectively to data messages.

The protocol is fully described in [84].

2.1 Overview of the protocol

A PGM session has one single sender, a set of receivers and a set of network elements. From a topological point of view, it can be seen as a tree, with the sender as the root, the network elements as nodes and the receivers as leaves.

The source "multicasts sequenced data paquets (`ODATA`)" downstream in the tree. When a receiver detect that an `ODATA` has been lost (receiving out of sequence `ODATA` for example), it sends upstream a `NAK` with the sequence number of the `ODATA` paquet send. When the source gets a `NAK`, if the matching data paquet is still in memory, the same data is sent in response, but in a repair data paquet (`RDATA`).

This basic behaviour is very dependent upon the reliability of the `NAK`. Hence, a hop-by-hop reliability for `NAK` is ensured: each element of the tree (leaf or node) will repeat the emission of a given `NAK`, until the reception of a matching `NAK` confirmation (`NCF`) sent by the upper element in the tree (node or root).

Moreover, some source-path-message (`SPM`) are sometimes sent. Their main goal is to maintain some routing information for the protocol (the topolopgy of the PGM tree). Moreover, they carry some informations like the last message sent by the source, which helps the receivers to detect losses.

2.2 The protocol in the ADVANCE project

This PGM protocol is a case study suggested by France Telecom in the ADVANCE project. The PGM protocol had been studied by one of their team, looking at a draft version of the protocol, and modeled in SDL [15].

Hence, there existed three main sources of specification for this protocol: the drafts (the final RFC has been published a few months after the start of the project), a France Telecom specification document [15] and the SDL source code.

2.3 More details on the PGM protocol

The full description of the protocol in [84] is 111 pages long, and even only the main mechanisms need 30 pages to be described.

Only some details useful to understand the studies in this document will be presented in this section.

2.3.1 The transmit window

The transmit window is the set of data packets that have been sent by the source and that are still in memory. The trailing edge (called `TXW_TRAIL`) is the oldest data packet available for repair and the leading edge (called `TXW_LEAD`) is "the sequence number of the most recent data packet the source has transmitted" [84, § 3.3].

2.3.2 Detection of losses

The PGM protocol is NAK-oriented: a receiver does not acknowledge the received packets, but sends a `NAK` to signal the losses. The question is: how can it detect losses? It should have a (partial) knowledge on the transmit window: having a lower bound of `TXW_LEAD` to `NAK` packets not received, and an approximation of `TXW_TRAIL`, to avoid to send `NAK` for messages that are no more in the source buffers.

To achieve it, each `ODATA` and `RDATA` packet will be sent with the current value of `TXW_TRAIL` (and its sequence number is a lower bound of the current `TXW_LEAD`), and each `SPM` will be sent with the current value of `TXW_TRAIL` and `TXW_LEAD`.

The PGM works with the assumption that the underlying network deliver the packets mostly in sequence. Then, if a hole in the sequence of received packets is detected, it is considered as a loss.

Hence, by comparing the sequence number of the last message received with the sequence number of the `ODATA` (or `RDATA`) or with the `TXW_LEAD` of a `SPM`, the receiver can detect losses.

It will also detect unrecoverable losses by looking at the `TXW_TRAIL` [3].

2.3.3 `NAK` and `NCF`

The reliability of the `NAK` mechanism is ensured hop-by-hop: when a receiver sends a `NAK`, its reception is acknowledged by the node by sending a `NCF`, and then the `NAK` is forwarded upstream.

When the source receives a `NAK`, it replies with a `NCF`, and then, if the `NAK` sequence number is in the transmit window, the source repairs the loss with the adequate `RDATA`.

2.3.4 Avoiding `NAK` explosion: filtering and random backoff

One problem with multicast protocol is the explosion of signaling messages at the source: if every receiver acknowledges each received message or `NAK` each loss, with thousands of receivers, the receiver can get a storm of signaling messages.

To avoid `NAK` explosion, the nodes have some mechanisms to "filter" the `NAK`: if a `NAK` is received from downstream or a `NCF` from upstream, all `NAK`s with the same sequence number received a few time after (see [84, § 7.4]) will be dropped.

Moreover, each receiver waits a "random backoff" before sending a `NAK`, and cancels this sending if a matching `NCF` is received while waiting. That is, if the same packet is detected as lost by all the receivers under a given node, if the random backoffs are different enough, only one receiver will send a `NAK`, which will be acknowledged to all the receivers (using multicast), and all (except the one that have sent the first `NAK`) will cancel the `NAK` emission. This is a rough description: the full mechanism is in [84, § 6.3].

[3] In fact, there is no requirement in the [84] to declare a loss as unrecoverable by just looking at this information, but it will be an assumption of our models, that seems reasonable.

After this short overview of PGM, we present what are the properties interesting to verify, and the complexities introduced by the protocol.

2.4 Properties of PGM to be verified

The protocol is unreliable, hence, we can not verify in the general case that all messages will be delivered. In fact, in the specification of the protocol, the only property that is expected from the protocol is "PGM guarantees that a receiver in the group either receives all data packets from transmissions and repairs, or is able to detect unrecoverable data packet loss." [84, Abstract, p. 1]. This property is called "*Losses-signaled*" in this paper.

This property implies that a receiver that will receive all messages from the network and drop all of them, just signaling to the application level that all messages have been lost, such a receiver will be compliant with the specification of the protocol. Of course, such a receiver is of no interest. This could mean that the property *Losses-signaled* itself has no interest. In fact, it only reflects that, because of the unreliability of the communication medium, it is difficult to impose strong properties upon PGM, but only a weak one like *Losses-signaled*.

However, there are some other properties that could be interesting to study, to see the efficiency of the mecanisms of the protocol.

Reliability efficiency. Even if there is no requirement in the protocol of a total reliability, a retransmission mechanism exists. It could be interesting to see the efficiency of this mechanism, *i.e.* which are the parameters of the systems that have an influence upon the reliability of the system, and for example under which assumption are all messages delivered.

In the PGM protocol, some parameters can be controlled by the protocol itself (like the throughput of the source, or the size of the window and the `SPM` rate), and other ones that are imposed by the environment (like the communication delays, the loss rate, etc.). Then, the question could be: in a given environment, what are the parameters that the protocol can change to increase the reliability?

Protocol load. Another important aspect in a protocol is the ratio payload/bandwidth, *i.e.* how many bytes are sent in the network and how many were really needed? To completely adress this issue, a model should take into account the size of the headers and the size of the packets, and such details will be very hard for the tools.

Nevertheless, a simplest way is to count the messages without any useful data: how many signaling messages (`NAK`, `NCF`, `SPM`) and how many useless `ODATA` or `RDATA` (such a packet is useless if it is discarded by the receiver, *i.e.* if a `ODATA`/`RDATA` with the same sequence number has already been received, or if the receiver has already declared this message as definitly lost). Even in signaling messages, there are useless messages, like duplicated `NAK`s.

2.5 Sources of complexity in PGM

For deciding what are the best abstractions to do when modeling PGM, our first work was to list the dimensions of the complexity of the system.

Topology. If we want to consider the fully general case, the topology of the system is a general tree. Hence, it should be possible to derive properties whatever the topology of the system is, as soon as it is a tree. But the verification tools typically used need to consider one given topology. Then, more and more complex trees can be taken into account, like illustrated in Figure 1. We will also see in section 6 how to derive properties for infinitely many topologies.

Loss policy. If no assumption is done upon the loss policy, *i.e.* if the more general case is assumed (every message can be lost), it will be impossible to prove anything about the protocol.

Nevertheless, there are different more or less realistic assumptions that can be done. The simplest one is, obviously, to assume that there is no loss, and then to study what are the properties of PGM. But this assumption is also the less realistic one. The next is "one fixed loss" *i.e.* one fixed paquet is lost. A more complex is "one random loss" (it is more complex for verification

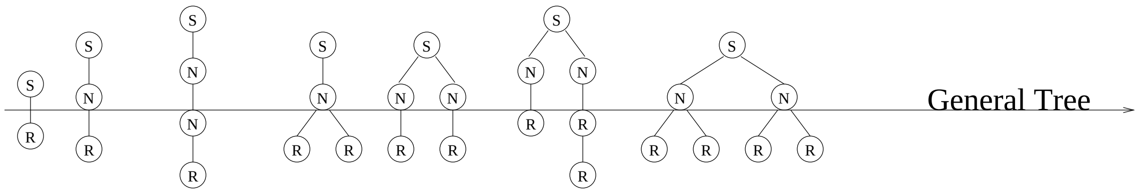


Figure 1: Increasing complexity of topology

tools, since they have to test, for each packet, what happens if this packet is lost or not). Then, more complex assumptions can be considered, like "at most N losses" or "at most N losses and never more than S in sequence" or "at most N over M packets are lost", with a special case when this M is the window size.

But this dimension of complexity is on the number of losses. Another dimension is: what kind of messages can be lost ? The most simple assumption is that only `ODATA` can be lost. Then, the signaling of the loss with `NAK` and `NCF` messages won't be affected by losses, and the protocol can repair the loss with `RDATA` messages. But in a real network, the signaling `NAK` messages and the `RDATA` ones could also be lost. To handle those potential losses, the protocol includes a loop of `NAK` emissions, and the system of `NAK` confirmations with `NCF` messages.

Communication mediums. What is the assumption for the communication between processes? In the worst case, a IP network is an unordered, unreliable channel, with unbounded transmission delay. Nevertheless, the IP network in general works, and assuming that it can be modelled as a FIFO queue is realistic.

For our tools, the most simple case is when processes are communicating through a one-bouded queue (juste one share variable). After that, delays can be added, or bounded queues or even more unbounded ones.

Transmission length. What is the size of the transmission (*i.e.* the number of packets sent during a session)? How many messages are the tools able to handle? For testing all the mechanisms of the protocol, the transmission length should be at least more than 2-windows long (to have some messages in the first window, some in the last, and some neither in one or the other). And a window should contain at least 3 messages, to have a first message, a last and also one that is neither first neither last.

But this is a simplification, because the PGM has not been designed to send only 9 messages but more realistically thousands of data packets.

An abstraction simplifying the work for the verification tools is to assume an infinite transmission (they do not have to take into account any ending).

The most difficult is a parametric one, *i.e.* being able to consider the length as a parameter, and making verification without any assumption on the value. This kind of work is described in section 5.

Traffic shape. Another dimension of complexity is the shape of the traffic: in [84, § 5.1.2], it is assumed that "a source MUST use a token bucket scheme or any other traffic management scheme that yields equivalent behavior" and, "in addition, a source MUST bound the maximum rate at which successive packets may be transmitted using a leaky bucket scheme drained at a maximum transmit rate, or equivalent mechanism."

Even with this restriction, it opens a wide range of traffics' shapes, considering the values of the token bucket size, the token bucket rate and the leaky bucket rate...

Moreover, the protocol is also sensitive to a local absence of data to transmit (because data packets carry signaling informations), and a special mechanism has been defined for this purpose [84, § 5.1.5 Heartbeat SPMs].

The most simple case is when the source emits at constant rate (the token bucket size is one) and never stop to transmit until the end of the communication.

Group membership (joining and leaving). "PGM has no notion of group membership" [84, § 1] and receivers can join or leave the session at any time, without any signaling.

Nevertheless, there is a special mechanism in the receiver to handle the arrival in a session. When a receiver joins, it had not received the messages sent before its arrival. Then, which messages can be considered as lost, and requested to be retransmitted ? A mechanism is presented in [84, § 6.1]. There is also an option of the protocol (`OPT_JOIN`, [84, § 9.4]) that defines a more "clever" algorithm to deal with joining.

Another difficulty in modeling the arrival of receivers is that this implies that the topology of the protocol is dynamic.

The model can be simplified, assuming that all receivers are in the group from the very beginning (and then, not implement the mechanism of initial data reception), or consider one late join at a given moment, or consider random join and leave.

Figure 2: Joining complexity

Removing mechanisms. Another solution to decrease the complexity of the system is to remove some of the mechanisms, that are designed to improve the mean performances of the protocol, but are not very relevant in a verification context, like the complex `NAK` generation mechanisms.

More roughly, mechanisms like the heartbeat `SPM` [84, § 5.1.5] or the filtering in nodes of `NAK`s [84, § 7.4] and `RDATA` [84, § 7.8] can also be removed.

Implementation choices. The specification of the protocol leaves the programmer with a lot of freedom about how to implement a lot of mechanisms, like the rate of ambient `SPM` [4], or the progress policy of the sender window[5].

An undeterministic policy could be modeled, that simulates the best as well as the worst policy. This solution has two major drawbacks: first, it can be hard to verify since it increases the number of potential behaviours; second, the performances of the protocols will be very bad (as an example, the transmit window policy that drops a message from memory as soon as it had been sent will never allow to recover any loss).

2.6 Simplifications and assumptions done in the project

The presentation of the complexity dimensions was done during the sixth meeting of the ADVANCE project (Paris, December 2001).

During this meeting, after a discussion with the France Telecom team about the realism of different options, it had been decided not to take into account the joining/leaving of receiver, and to focus on the three following points: parametric verification regarding the number of sent packets; deriving properties which are true whatever the topology is; and deriving temporal constraints. But if those tasks needed advanced verification tools, it can be shown that classical verification tools can already bring interesting results.

3 Verification of the PGM using classical tools (year 2)

By classical tools, we mean simulation and model-checking tools which work by exhaustively enumerating all the possible behaviours of the studied system. Such tools are not able to fulfill the more complicated verification tasks we had in mind but can nevertheless bring interesting results.

[4] The ambient `SPM` rate must be "at least sufficient to maintain current source path state in PGM network elements." [84, § 5.1.4]

[5] Even if an appendix in the RCF ([84, Appendix F]) is devoted to this mechanism, it is given "for illustrative purpose only".

3.1 Tools we used

3.1.1 The IF langage

A common langage is used by the different tools of the project: IF [35, 62]. Nevertheless, there were two versions of the langage (1.0 and 2.0). The version 2.0 has been designed during the ADVANCE project, and gives some very interesting new features for modeling distributed systems. But, during the first years of the project, most of the tools are only accepting IF 1.0. Then, to be able to check the same model with differents tools, the models had to be written in IF 1.0.

Roughly speaking, the IF underlying theoretical model is the synchronisation of timed automata with counters, communicating trough queues.

3.1.2 The IF toolset

In the IF toolset, we find a tool named `generator`, that generates all the state space of the system (if finite). The output is a graph with labelled transitions in the ".aut" format, that can be handled by the CADP tools [37].

3.1.3 The CADP toolset

To verify off-line the properties or to minimize and compare graphs, the CADP toolset [37] have been used.

3.2 Modeling the PGM protocol

The full PGM protocol is out of the scope of existing verification tools: a complete modelization implies to be able to handle thousands of processes, dynamic creation and destruction of processes, dynamic routing, dozens of counters and timers per process, and transmissions of length greater than 2^{32}...

Nevertheless, some interesting aspects of the protocol should be studied, doing the right assumptions and abstractions, simplifying some complexities we have already discussed.

A first model was produced by the VERIMAG team, as a translation of the SDL code given by FRANCE TELECOM, using the IF 2.0 langage.

This code had some drawbacks for our use: it was written in IF 2.0 and made an intensive use of lists and external procedures (*i.e.* procedures written in `C` langage), that can not be handled by the verification tools (and are very ressource consuming for the IF `generator`).

Hence, there was a need to define other models for the project.

The two first works were not exactly new models, but common material for the members of the project (in order to share the same knowledge of the protocol): the IF 2.0 file result of the translation has been commented, and an "automaton oriented" description of the [84] has been written. All files have been uploaded on the WEB server [6, Case Studies].

The first code written was the "very simple pgm" with the following features:

Topology: 1 source – 1 network node – 1 receiver

Policy of loss: 1 random `ODATA` lost

Communication medium: 1-bounded buffers (without delay), one per type of message, reliable

Transmission length: parameter of the code

Traffic shape: none (untimed model, but limited by the size of the communication buffers[6])

Group membership: the receiver is part of the group since the first time

Removed mechanisms: no heartbeat `SPM`, no filtering in nodes, no repeat of `NAK` (useless since `NAK` communication is reliable), untimed model

[6]The production rate is limited by the consuming rate and buffer size. Since the communication is reliable, the sender can not write in a buffer if the data has not been transmitted.

Implementation choices: undeterministic transmit window policy, and ambient rate driven by the number of send messages (one `SPM` is send each time `AMB_RATE` `ODATA` have been sent, with `AMB_RATE` a parameter of the code)

The architecture (*i.e.* topology and communication buffers) of this "very simple pgm" is illustrated in Figure 3.

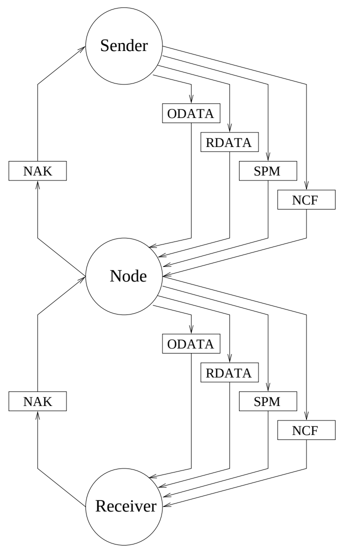


Figure 3: Topology and communication buffers of the "very simple pgm" model

This was a first code, the "starting point" for the project. The aim was to use it as a first input for the tools, and to see if more complex models could be taken into account by the tools, or if some more simplifications and abstractions had to be done.

3.3 Influence of modeling PGM upon the IF language

As the entry point of the Advance verification toolset, the common language IF needs enough expressive power to allow for a sound and direct mapping from complex SDL specifications. Therefore, the PGM protocol, and in particular its SDL specification written by France Telecom, has had a strong influence on the design of the common language IF-2.0.

In addition to the fact that PGM is the major Advance case study, we found the PGM specification extremely interesting mainly for two reasons.

First, this specification was developped by people working outside the Advance project and moreover outside the formal verification area. Therefore, the emphasis was on *modelling* i.e, remaining closer as possible to the IETF description, and not on verification. That is, no verification tool or method had been a priori targeted and no simplifications were made.

Second, given its complexity, as well as the large number of SDL concepts used, the PGM specification represents a typical medium-sized case study in the telecommunication area. We recall here that the whole description is about 3400 lines of textual SDL/PR code, and uses features such as abstract data types, dynamic creation and destruction, asynchronous communication, procedures, timers, etc.

Therefore, the IF-2.0 language includes (but is not limited to) most of the widely used concepts of the SDL language, which are also present in the PGM specification (see above). Nevertheless, the IF-2.0 language offers, in addition to SDL, a much more powerful timed-model based on urgencies, parameterization at different levels (i.e, data, timing or configurations), structured control on transitions, hierachical states, etc.

All these features combined have been made possible the complete rewriting of the whole SDL specification in IF-2.0. The IF-2.0 description is about 1300 lines of IF code, which is 2.5 times smaller than the SDL one. Moreover, some aspects of the specification get improved, for example, the multicast tree structure become parameterized. Furthermore, the rewriting process pointed out several inconsistencies in the original SDL specification such as wrong initializations, dead code, useless communications, etc.

That is, the existence of the PGM case study guarantees for the continuation of the project that the modelling language we decided to use is able to cope with realistic telecommunication applications.

3.4 First experiment: *Losses-signaled*

The first experiment was simply to verify the *Losses-signaled* property on the very simple model.

Since this property is very basic, it was more a test for our model and the performances of the verification tools that a real test of the property. Actually, an interesting behavior has been found, that was unforeseen, even if, once the computer has given a counter-example, the mistake appears obvious. But is it not the case for most of the bugs?

The context of the experiment was the very simple protocol, described in Section 3.2, with a few messages.

The results were that the property *Losses-signaled* was verified for all messages, except for those of the last window.

The experiments are presented in the Table 1, looking at the lines with configuration “SNR 1p” and “Closing SPM” value “false” in the rows of *Losses-signaled*.

By scanning the counter examples, the reason why the messages in the last window can be neither delivered neither signaled as lost, can be explained as follows: if the first message of the last window is lost, and if the transmit window policy makes the window advance by trying to keep as much messages in memory than is allowed, and closing it once the transmission is ended, all `ODATA` messages will be sent with a `TXW_TRAIL` value meaning that the lost message can be recovered. But the `NAK` sent by the receiver can be received too late, and discarded. Then, the receiver is not able to detect that the message has been lost and waits indefinitely (the mechanism of loss detection is presented in Section 2.3.2).

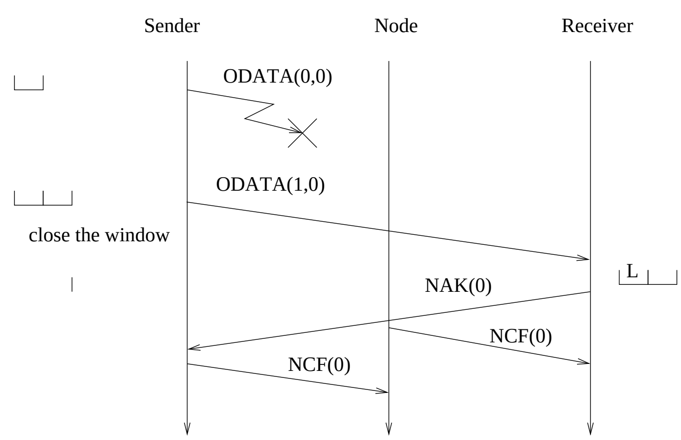


Figure 4: Example of last window frozen

This behavior is illustrated in Figure 4, considering a very simple communication of 2 data packets, with 2 as window size.

This highlights the fact that there is no special mechanism in the protocol to identify the closing of a session.

Nevertheless, in the protocol, there is an “heartbeat SPM” mechanism, designed to send some `SPM` when no `ODATA` or `RDATA` are sent. In our model, a “closing SPM” option has been added: if this option is set, when all the `ODATA` packets have been sent, a `SPM` is sent (with the value of the transmit window). This option is illustrated in Figure 5, considering the same communication as in Figure 4.

Then, if this option is set, the property *Losses-signaled* is verified for all messages, as presented in the result Table 1 in lines with “Closing SPM” value “true” (for “SNR 1p” configurations).

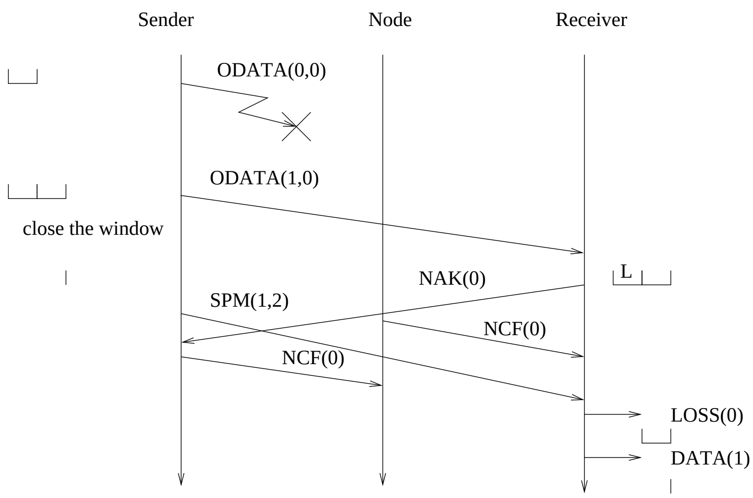
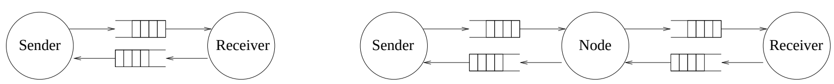


Figure 5: Example of effect of the “closing SPM”

3.5 Second experiment: removing nodes

The second experiment was designed to handle a problem about the architecture of the protocol.

What is the role of a node in the model? In the real protocol, it is used for routing and filtering, but, as our models do not consider routing problems, if we do not care about the filtering aspect, can we remove the nodes? This would simplify the verification work without loss of generality if we could show that removing nodes have no effect.



(a) The SR config

(b) The SNR config

Figure 6: Architecture of the two configurations compared

Hence, two topologies of the protocol have been considered: one with one source, one node and one receiver (designed by SNR), and one with only one source and one receiver (designed by SR), communicating through unbounded FIFO queues[7], like illustrated on Figure 6.

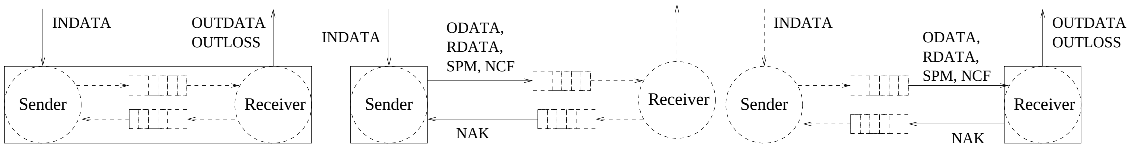
If we are to make comparisons, the main question is what is the comparison criterion?

Three point of views have been defined for the problem:

1. the *blackbox* point of view: looking the system from the application point of view, hiding all actions except for the inputs and outputs of the session (Figure 7(a));
2. the *source* point of view: looking only at the inputs and outputs of the sender process (Figure 7(b));
3. the *receiver* point of view: looking only at the inputs and outputs of the receiver process (Figure 7(c)).

Then, the comparisons will be to test the “τ^*-bisimulation” relation ([51], the `-iequ` option of the `aldebaran` tool [37]) between the behaviours graphs, with only a given set of visible actions (different for each point of view).

[7]We could not compare the two systems using the 1-bounded queues, like with the model used in section 3.4, since it is obvious that the configuration SNR with 1-bounded buffers could perhaps be equivalent to a SR configuration with 2-bounded buffers, but never to a SR with 1-bouded buffers. And, with the tools used, it was easier to consider unbounded FIFO than 2-bouded FIFO.



(a) BlackBox (b) Source (c) Receiver

Figure 7: The different points of view

Some configurations have been tested, as presented in the Table 1, looking at the lines "SNR q" and "SR q". The rows "Blackbox Equiv.", "Source Equiv." and "Receiver Equiv." give the results for this test of equivalence.

A first result, that confirms the previous study, is that the effect of the "Closing SPM" option is the same for these models than if the system was communicating through 1-place buffers.

But the main results are that, in all the tested cases, both systems are equivalent from the application and sender point of view, but not from the receiver point of view.

The results for the blackbox (application) and sender point of view were expected, but not the one for the receiver.

To find the differences between both configurations for the receiver point of view, we have tried to hide more and more actions, randomly, until getting some new results.

The interestings results, presented in the Table 2 are that:

- adding the `NCF` messages to the list of hidden actions, the receiver points of view are equivalent for all tested configurations (except the case with 7 messages sent: the tool runs out of memory);
- with only `NAK` and `NCF` messages as visible actions, the receiver points of view are equivalent for all tested configurations (with the same restriction on memory).

Looking at the counter-examples and those new results, the explanation of the non-equivalence comes from the atomicity of `NCF` reply (the `NAK`/`NCF` mechanism is described in Section 2.3.3): when the receiver is plugged directly to the sender, a `NCF` is immediatly followed by a `RDATA` (if the matching messages is still in transmit window), because the behaviour of the source is to retransmit the `NCF` and just after the matching `RDATA` (in our model); whereas when the receiver is plugged to the node, it will receive a `NCF` when the node has received the `NAK`, but the `RDATA` is not sent immediately after, as the node will then transmit the `NAK` to the source, and the `RDATA` will be sent later.

3.6 Parametric analysis: condition for recovering all losses

The PGM protocol is unreliable, and it can not be guaranteed that all messages will be recovered. Nevertheless, the "efficiency" of the protocol can be studied, *i.e.* looking at the performances under "reasonable" assumptions.

In this section, the question is: is there a relation between the parameters of the system that implies that the system will recover all losses, under the assumption that only `ODATA` messages can be lost?

The tool TReX was designed to automatically derive this kind of constraints, and we will see in section 4 the results of this tool, but the first study was done "manually", guessing the relation first and then verifying it on some configurations with the IF `generator` tool.

3.6.1 Condition for recovering all losses

What is the mechanism that allows to recover a loss? First (obvious) step, a message must be lost. Second, the receiver must get a message that signals that a previous message has been lost: it can be a `ODATA`, a `RDATA` or a `SPM` sent after the loss. Third, the `NAK` signaling the loss must be received

# of mess	Window Size	AMB rate	Closing SPM	Conf.	Trans.	States	*Losses-signaled* 0	1	2	3	4	5	6	BlackBox Equiv.	Source Equiv.	Receiver Equiv.
5	1	2	**ff**	SNR 1p	17,642	7,443	**tt**	**tt**	**tt**	**tt**	**ff**	-	-	-	-	-
				SNR q	36,944	20,802	**tt**	**tt**	**tt**	**tt**	**ff**	-	-	**tt**	**tt**	**ff**
				SR q	5,501	3,189	**tt**	**tt**	**tt**	**tt**	**ff**	-	-			
5	1	2	**tt**	SNR 1p	19,821	8,338	**tt**	**tt**	**tt**	**tt**	**tt**	-	-	-	-	-
				SNR q	49,141	27,849	**tt**	**tt**	**tt**	**tt**	**tt**	-	-	**tt**	**tt**	**ff**
				SR q	6,555	3,812	**tt**	**tt**	**tt**	**tt**	**tt**	-	-			
5	3	2	**ff**	SNR 1p	1,089,462	349,270	**tt**	**tt**	**ff**	**ff**	**ff**	-	-	-	-	-
				SNR q	2,002,554	1,018,918	**tt**	**tt**	**ff**	**ff**	**ff**	-	-	**tt**	**tt**	**ff**
				SR q	181,185	89,933	**tt**	**tt**	**ff**	**ff**	**ff**	-	-			
5	3	2	**tt**	SNR 1p	1,139,092	367,235	**tt**	**tt**	**ff**	**ff**	**ff**	-	-	-	-	-
				SNR q	2,594,217	1,345,093	**tt**	**tt**	**ff**	**ff**	**ff**	-	-	**tt**	**tt**	**ff**
				SR q	214,434	108,176	**tt**	**tt**	**ff**	**ff**	**ff**	-	-			
5	∞	∞	**ff**	SNR 1p	567,709	189,912	**ff**	**ff**	**ff**	**ff**	**ff**	-	-	-	-	-
				SNR q	328,118	168,336	**ff**	**ff**	**ff**	**ff**	**ff**	-	-	**tt**	**tt**	**ff**
				SR q	55,883	28,176	**ff**	**ff**	**ff**	**ff**	**ff**	-	-			
5	∞	∞	**tt**	SNR 1p	643,660	217,471	**ff**	**ff**	**ff**	**ff**	**ff**	-	-	-	-	-
				SNR q	442,742	233,676	**ff**	**ff**	**ff**	**ff**	**ff**	-	-	**tt**	**tt**	**ff**
				SR q	69,970	35,163	**ff**	**ff**	**ff**	**ff**	**ff**	-	-			
7	2	2	**ff**	SNR 1q	1,704,342	556,170	**tt**	**tt**	**tt**	**tt**	**tt**	**ff**	**ff**	-	-	-
				SNR q	21,182,556	11,073,929	†	†	†	†	†	†	†	†	†	†
				SR q	1,016,579	527,475	**tt**	**tt**	**tt**	**tt**	**tt**	**ff**	**ff**			
7	2	2	**tt**	SNR 1q	1,735,130	567,283	**tt**	**tt**	**tt**	**tt**	**tt**	**tt**	**tt**	-	-	-
				SNR q	26,333,252	13,944,848	†	†	†	†	†	†	†	†	†	†
				SR q	1,182,067	620,581	**tt**	**tt**	**tt**	**tt**	**tt**	**tt**	**tt**			
7	2	∞	**ff**	SNR 1p	152,298	58,718	**tt**	**tt**	**tt**	**tt**	**tt**	**ff**	**ff**	-	-	-
				SNR q	994,660	531,116	**tt**	**tt**	**tt**	**tt**	**tt**	**ff**	**ff**	**tt**	**tt**	**ff**
				SR q	95,214	51,551	**tt**	**tt**	**tt**	**tt**	**tt**	**ff**	**ff**			
7	2	∞	**tt**	SNR 1p	170,734	66,051	**tt**	**tt**	**tt**	**tt**	**tt**	**tt**	**tt**	-	-	-
				SNR q	1,287,932	689,024	**tt**	**tt**	**tt**	**tt**	**tt**	**tt**	**tt**	**tt**	**tt**	**ff**
				SR q	112,146	61,401	**tt**	**tt**	**tt**	**tt**	**tt**	**tt**	**tt**			

†= out of memory ; SNR 1p= Configuration with topology Sender-Node-Receiver, communicating through 1-place buffers ; SNR q= Configuration with topology Sender-Node-Receiver, communicating through FIFO queues ; SR q= Configuration with topology Sender-Receiver, communicating through FIFO queues

Table 1: Experiment results

# of mess	Window Size	AMB rate	Closing SPM	Without NCF	Only NAK and NCF
5	1	2	**ff**	**tt**	**tt**
5	1	2	**tt**	**tt**	**tt**
5	3	2	**ff**	**tt**	**tt**
5	3	2	**tt**	**tt**	**tt**
5	∞	∞	**ff**	**tt**	**tt**
5	∞	∞	**tt**	**tt**	**tt**
7	2	2	**ff**	†	†
7	2	2	**tt**	†	†
6	2	∞	**ff**	**tt**	**tt**
6	2	∞	**tt**	**tt**	**tt**

†= out of memory

Table 2: Equivalence between SNR and SR configurations with queues, from the receiver point of view

before the corresponding data have been dropped from the sender memory. Then, the matching `RDATA` is sent and must be received.

Hence, in the property are involved the following parameters:

- the loss assumption on `ODATA`, `RDATA`, `SPM` and `NAK`;
- the transit delay of messages;
- the rate of `ODATA` (sender traffic policy) and `SPM` (ambiant and heartbeat `SPM`); and,
- the transmit window policy.

Notice that the two first points are driven by the environment (we could make some assumption while modeling, but the implementation can not control them) and the two last are parameters of the protocol (that can be changed, up to the computer limits).

The most general relation is out of the scope of this study: it seems a first interesting step to consider one given "reasonable" configuration and to find the associated relation.

3.6.2 The considered model

The model tested is based on the SNR configuration of the section 3.5:

Topology: 1 source – 1 network node – 1 receiver

Policy of loss: N random `ODATA` lost (N is a parameter in the code)

Communication medium: FIFO queues with fixed delay

Transmission length: parameter of the code

Traffic shape: `ODATA` are sent at constant rate, one `ODATA` every `PROD_RATE` time unit

Group membership: the receiver is part of the group since the first time

Removed mechanisms: no heartbeat `SPM`, no filtering in nodes, no repeat of `NAK` (useless since `NAK` communication is reliable), no backoff before sending a `NAK`

Implementation choices:

Transmit window policy: the sender tries to keep the maximum number of packets in memory, *i.e.* to keep "Window size" messages, minus one (keep free to receive data from the application). When the transmission is ended, drop one packet every `PROD_RATE` time unit.

ambient `SPM`: the ambient rate is driven by the advance of the window. One `SPM` is sent each time `AMB_RATE` packets have been dropped out of the transmit window, (with `AMB_RATE` a parameter of the code).

closing `SPM`: the "closing SPM" option is always turned on.

Note on "rate". There are two opposite views on the notion of rate. Formally, a rate is a number of actions per time unit. Nevertheless, while modeling or programming, it is easier to manipulate the waiting time between two actions. Since this waiting time is a way to encode the rate, the variables used in the model uses the suffix "`RATE`" instead of "`INTER_ACTION_WAITING_TIME`". But one consequence of this convention is that, to increase the production rate of `ODATA`, the variable "`PROD_RATE`" has to be decreased.

3.6.3 Guessing the relation

As presented in section 3.6.1, all losses are recovered if the `NAK` is received before the end of the lifetime of the data in the transmit window.

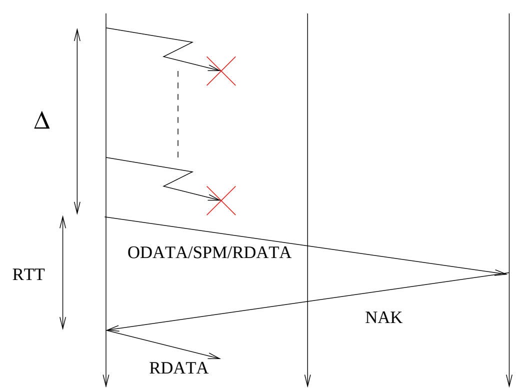


Figure 8: Pattern of loss recovering

The Figure 8 gives the pattern of a loss recovering in our model. The delay between the instant a message (that will be lost) is sent and the instant that a message (that will not be lost) is sent is called Δ. The delay between the sending of this message and the reception of the matching `NAK` is the Round Trip Time (RTT), assuming that a `NAK` is sent as soon as a loss is detected (it is the case in our model, but in a more realistic one, we should add the random backoff).

A message is recovered if this $\Delta + RTT$ is lesser than the lifetime (Lf) of the data.

What are the values of those three variables?

The value of Δ depends on the kind of the first non-lost message: `ODATA`, `RDATA` or `SPM`, *i.e.* $\Delta = \min(\Delta_{\texttt{ODATA}}, \Delta_{\texttt{RDATA}}, \Delta_{\texttt{SPM}})$.

We are not able to give an upper bound of $\Delta_{\texttt{RDATA}}$ (this could be ∞ if no message has been lost before).

For $\Delta_{\texttt{ODATA}}$, considering that no more that `NLOSS` messages can be lost, its maximum bound is $\texttt{NLOSS} * \texttt{PROD\_RATE}$.

Notice that this maximum bound is no more true at the end of the transmission: it is false for the `NLOSS` last messages (that can all be lost and not followed by any `ODATA`).

For $\Delta_{\texttt{SPM}}$, the upper bound is simply $\texttt{AMB\_RATE} * \texttt{PROD\_RATE}$, except for the first window (as the ambiant `SPM` are only related to the advance of the window, and there is no regular advance of the window in the first window).

The lifetime of a data is also simple: $Lf = (\text{Window size} - 1) * \texttt{PROD\_RATE}$.

Then, the condition on parameters to recover all losses should be $Lf \geq \Delta + RTT$. Nevertheless, it is partially false. There is a problem of interleaving of actions at the date when a data is dropped: a data is not accessible during all its lifetime. Because in our model, the dropping action takes no time, the data is "first" ready and "after" lost. The condition is then $Lf > \Delta + RTT \iff$:

$$\text{Window Size} - 1 - \min\left(\texttt{NLOSS}, \texttt{AMB\_RATE}\right) > \frac{RTT}{\texttt{PROD\_RATE}} \tag{1}$$

Keep in mind that this relation is false on the first window (the evaluation of the $\Delta_{\texttt{SPM}}$ assumes a "constant" rate of `SPM`, and in our model, the `SPM` messages are sent only in case of window advance) and on the `NLOSS` last messages (the evaluation of the $\Delta_{\texttt{ODATA}}$ assumes that a `ODATA` will be sent in the futur, and it is false at the end of the transmission).

3.6.4 Testing the relation on some configurations

The relation on parameters has been tested on some configurations: different series of experiments have been made, looking at the evolution of the losses while increasing or decreasing each of the parameters involved in the relation.

The results are presented in Tables 3, 4 and 5.

The Tables 3 and 4 are devoted to the study of the recovering based on the $\Delta_{\texttt{ODATA}}$ (assuming $\Delta_{\texttt{SPM}} = \infty$) and Table 5 studies the rate of ambiant `SPM`.

Considering recovery based on `ODATA` In Tables 3 and 4, the row $|w| - 1 - l$ is the value Window Size $- 1 -$ `NLOSS`, and the row $\frac{RTT}{Prod}$ is $\frac{RTT}{\texttt{PROD\_RATE}}$ *i.e.* the left and right part of the Equation 1 (because `AMB_RATE` $= \infty$ in all the configurations presented there).

The first three configurations tested just increase the size of the transmission. It shows that the last message is always lost. Basically, in all configurations in Tables 3 and 4, the `NLOSS` last messages can always be definitely lost (where `NLOSS` is the maximum number of messages that can be lost, given in the first row).

The second, third and fourth series of experiments (Table 3) just increase the production rate (they decrease the time between two emissions of `ODATA`, then increase the number of `ODATA` per time unit).

In the second series, the window size is 3 and the maximum number of losses is 1: the expression Window Size $- 1 -$ `NLOSS` is 1. Then, for all configurations where $\frac{RTT}{\texttt{PROD\_RATE}} < 1$, the messages (exept the last) are all delivered, never lost. When the threshold 1 is overtaken, all messages can be definitively lost.

The third series is the same as the second, except that both the window size and the number of losses have been incremented by one. As the difference is the same as in previous series, we have the same threshold behaviour with 1.

The fourth series assume 2 losses and a window of size 5, *i.e.* the difference Window Size $- 1 -$ `NLOSS` is 2. Then, the threshold behaviour exists at value 2.

The fifth, sixth, and seventh series are in Table 4. They just decrease the window size, to show that a wide window allows to recover all losses and a small one does not. The relation given in Equation 1 is always verified.

The eighth experiment, the last in Table 4 increases the number of losses. For the 0, 1 and 2 losses, only the last messages can be definitly lost. With 3 losses, the Equation 1 is no more verified, and all messages can be lost.

Considering recovery based on `SPM` The Table 5 presents some experiment to test the effect of the ambiant `SPM` rate, without the recovering mechanims driven by `ODATA`. To do so, in all configurations tested, the value of expression Window Size $- 1 -$ `NLOSS` is 0 (and then can never be greater than the ratio RTT/Prod).

The first series increases the `SPM` ambiant rate. In the first line, with no `SPM` (`AMB_RATE` $= \infty$), all messages can be definitively lost. On the opposite, with `AMB_RATE` $= 1$, all messages not in the first window will be delivered. The Equation 1 is confirmed.

An interesting point is that, even if the Equation 1 is not satisfied (with `AMB_RATE` $= 2$ on the second line for example), some messages are always recovered (one out of two). In fact, the message sent just before the `SPM` can always be recovered, because the sum $\Delta_{SMP} + RTT$ is small enough for this message.

# of losses	# of mess.	AMB rate	Window Size	RTT	Prod. rate	States	Trans.	Mess. always delivered						$\lvert w\rvert - 1 - l$	$\frac{RTT}{Prod}$
								0	1	2	3	4	5		
Increase the size of the transmission															
1	3	∞	3	8	9	709	750	**tt**	**tt**	**ff**	-	-	-	1	0.89
1	4	∞	3	8	9	1,363	1,446	**tt**	**tt**	**tt**	**ff**	-	-	1	0.89
1	5	∞	3	8	9	2,107	2,240	**tt**	**tt**	**tt**	**tt**	**ff**	-	1	0.89
Evolution of the production rate															
1	6	∞	3	8	15	2,716	2,896	**tt**	**tt**	**tt**	**tt**	**tt**	**ff**	1	0.53
1	6	∞	3	8	10	5,027	5,935	**tt**	**tt**	**tt**	**tt**	**tt**	**ff**	1	0.80
1	6	∞	3	8	9	2,935	3,122	**tt**	**tt**	**tt**	**tt**	**tt**	**ff**	1	0.89
1	6	∞	3	8	8	8,671	10,535	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	1	1.00
1	6	∞	3	8	7	2,858	3,053	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	1	1.14
1	6	∞	3	8	5	3,324	3,656	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	1	1.60
1	6	∞	3	8	3	9,550	11,531	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	1	2.67
Evolution of the production rate (with wider window but same diff,)															
2	6	∞	4	8	15	9,577	10,932	**tt**	**tt**	**tt**	**tt**	**ff**	**ff**	1	0.53
2	6	∞	4	8	10	19,891	24,762	**tt**	**tt**	**tt**	**tt**	**ff**	**ff**	1	0.80
2	6	∞	4	8	9	10,632	12,069	**tt**	**tt**	**tt**	**tt**	**ff**	**ff**	1	0.89
2	6	∞	4	8	8	60,187	78,667	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	1	1.00
2	6	∞	4	8	7	11,211	12,796	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	1	1.14
2	6	∞	4	8	5	19,173	23,198	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	1	1.60
2	6	∞	4	8	3	43,028	54,838	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	1	2.67
Evolution of the production rate (with wider window and bigger diff,)															
2	6	∞	5	8	15	8,153	9,281	**tt**	**tt**	**tt**	**tt**	**ff**	**ff**	2	0.53
2	6	∞	5	8	10	15,543	19,173	**tt**	**tt**	**tt**	**tt**	**ff**	**ff**	2	0.80
2	6	∞	5	8	5	18,494	22,726	**tt**	**tt**	**tt**	**tt**	**ff**	**ff**	2	1.60
2	6	∞	5	8	4	198,971	270,047	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	2	2.00
2	6	∞	5	8	3	45,027	57,698	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	2	2.67
2	6	∞	5	8	2	732,344	1,015,141	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	2	4.00

Table 3: Are all losses recovered? Effects of the production rate

# of losses	# of mess.	AMB rate	Window Size	RTT	Prod. rate	States	Trans.	Mess. always delivered						$\|w\|-1-l$	$\frac{RTT}{Prod}$
								0	1	2	3	4	5		
Evolution of the window size															
1	6	∞	5	8	9	2.118	2.235	**tt**	**tt**	**tt**	**tt**	**tt**	**ff**	3	0.89
1	6	∞	4	8	9	2.558	2.712	**tt**	**tt**	**tt**	**tt**	**tt**	**ff**	2	0.89
1	6	∞	3	8	9	2.935	3.122	**tt**	**tt**	**tt**	**tt**	**tt**	**ff**	1	0.89
1	6	∞	2	8	9	3.238	3.404	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	0	0.89
1	6	∞	1	8	9	845	915	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	-1	0.89
Evolution of the window size (with faster source)															
1	6	∞	5	8	8	5.662	6.788	**tt**	**tt**	**tt**	**tt**	**tt**	**ff**	3	1
1	6	∞	4	8	8	7.698	9.310	**tt**	**tt**	**tt**	**tt**	**tt**	**ff**	2	1
1	6	∞	3	8	8	8.671	10.535	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	1	1
1	6	∞	2	8	8	5.194	6.054	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	0	1
1	6	∞	1	8	8	845	915	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	-1	1
Evolution of the window size (with more losses)															
2	6	∞	6	8	20	6.375	7.267	**tt**	**tt**	**tt**	**tt**	**ff**	**ff**	3	0.4
2	6	∞	5	8	20	8.153	9.281	**tt**	**tt**	**tt**	**tt**	**ff**	**ff**	2	0.4
2	6	∞	4	8	20	9.577	10.932	**tt**	**tt**	**tt**	**tt**	**ff**	**ff**	1	0.4
2	6	∞	3	8	20	8.796	9.687	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	0	0.4
2	6	∞	2	8	20	9.678	10.222	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	-1	0.4
2	6	∞	1	8	20	1.989	2.169	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	-2	0.4
Evolution of the number of losses															
0	6	∞	4	8	20	344	344	**tt**	**tt**	**tt**	**tt**	**tt**	**tt**	3	0.4
1	6	∞	4	8	20	2.397	2.546	**tt**	**tt**	**tt**	**tt**	**tt**	**ff**	2	0.4
2	6	∞	4	8	20	9.577	10.932	**tt**	**tt**	**tt**	**tt**	**ff**	**ff**	1	0.4
3	6	∞	4	8	20	22.924	27.228	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	0	0.4
4	6	∞	4	8	20	33.014	39.636	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	-1	0.4
5	6	∞	4	8	20	35.800	43.088	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	-2	0.4

Table 4: Are all losses recovered? Effects of the number of losses and window size

# of losses	# of mess.	AMB rate	Window Size	RTT	Prod. rate	States	Trans.	Mess. always delivered											$\lvert w\rvert -$ $1-l$	$\frac{RTT}{Prod}$
								0	1	2	3	4	5	6	7	8	9	10		
Evolution of the rate of ambiant SPM																				
3	7	∞	4	8	6	234,090	308,894	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	-	-	-	-	-7	1.33
3	7	2	4	8	6	753,936	989,649	**ff**	**ff**	**ff**	**tt**	**ff**	**tt**	**ff**	-	-	-	-	1	1.33
3	10	2	4	8	6	753,936	989,649	**ff**	**ff**	**ff**	**tt**	**ff**	**tt**	**ff**	**tt**	**ff**	**tt**	**ff**	1	1.33
3	7	1	4	8	6	753,936	989,649	**ff**	**tt**	**tt**	**tt**	**tt**	**tt**	**tt**	-	-	-	-	2	1.33
Evolution of the production rate																				
3	7	2	4	8	10	202,502	253,482	**ff**	**tt**	**tt**	**tt**	**tt**	**tt**	**tt**	-	-	-	-	1	0.80
3	7	2	4	8	9	116,142	136,320	**ff**	**ff**	**ff**	**tt**	**ff**	**tt**	**ff**	-	-	-	-	1	0.89
3	7	2	4	8	8	855,154	1,125,118	**ff**	**ff**	**ff**	**tt**	**ff**	**tt**	**ff**	-	-	-	-	1	1.00
3	7	2	4	8	7	117,731	138,085	**ff**	**ff**	**ff**	**tt**	**ff**	**tt**	**ff**	-	-	-	-	1	1.14
3	7	2	4	8	6	753,936	989,649	**ff**	**ff**	**ff**	**tt**	**ff**	**tt**	**ff**	-	-	-	-	1	1.33
3	7	2	4	8	5	347,583	437,608	**ff**	**ff**	**ff**	**tt**	**ff**	**tt**	**ff**	-	-	-	-	1	1.60
3	7	2	4	8	4	10,538,131	14,372,868	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	-	-	-	-	1	2.00
3	7	2	4	8	3	2,202,437	2,879,437	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	**ff**	-	-	-	-	1	2.67

Table 5: Are all losses recovered? Effects of the ambiant rate of `SPM`

The second series increases the production rate. We have the same kind of behaviour that for the first series: when the relation is satisfied, all messages are received, and when increasing the data rate, more and more messages can be definitively lost.

Considerations on time and space. It can be noticed that, in some series, the size of the state space increases globally, but not locally. It is particularly visible on second, third and fourth series of Table 3.

This global increase seems "normal": there are more and more messages in the system at a time as the production rate increases, which implies that the system is more "complex".

Nevertheless, there are some local decreases: in the second series, when `PROD_RATE` comes from 10 to 9, from 8 to 7, in the second series when `PROD_RATE` comes from 10 to 9, from 8 to 7. Each time, from an even to an odd value.

Our explanation is that this is linked to the fact that the RTT is even (in fact, the delay of each queue is 2): if the production delay (`PROD_RATE`) is also even, there are more events which can interleave at even dates than if the production delay is odd.

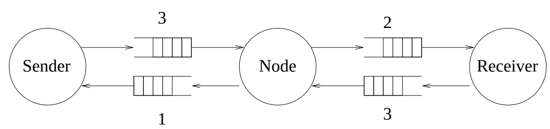


Figure 9: The delays of the experiments given in Table 6

To confirm this hypothesis, the same series of experiments as in Table 3 have been made, changing just the transit delays. They are given in Figure 9[8]. The total gives a RTT value of 9. The results are in Table 6.

It can be clearly seen that the divisibility of the RTT by the production rate has a great influence on the size of the state space.

In the first series of experiments (in which the size of the transmission is increased), with RTT=`PROD_RATE` =9 the state space is about twice bigger than with RTT=8 and `PROD_RATE` =9.

In the second series of experiment, we clearly have a smaller state space with RTT=9 (whereas increasing the delay should increase the complexity of the system), except for production rates 9, 7, 5 and 3. It can be explained by the fact that, for RTT=8, 8 and 7 are coprime, and also for 8 and 5: the non-interleaving gain is the same for both systems, and a system with a greater RTT is more complex. On the opposite, when the RTT and the production rate are not coprime (RTT=9 and `PROD_RATE` =9 or 3), the situation is far worse.

We can do the same kind of comparisons with the third and forth series, and we can also compare the timed and untimed versions.

We can compare the SNR q configuration of the "removing node" experiment (Section 3.5, with only 1 loss) and this timed version.

We can see that, except for a high rate of `ODATA`s, the state space is bigger in untimed systems that in the timed one. It can be explained by the fact that the timing constraints reduce the interleaving of actions between processes: for example, the production rate with the upper bound of queues limits the number of data in the queues, and therefore the reactions of the node and receiver.

3.7 Conclusion

Before the general conclusions about the whole case study, it is interesting to make a few points about this study done with "classical" tools. The first point is that, even with tools which don't take into account infinite behaviours, it is possible to verify interesting properties, as we just showed. More specifically, we studied the following points:

[8]Downstream 3 from sender to node, 2 from node to receiver; upstream: 3 from receiver to node and 1 from node to sender.

# of losses	# of mess.	AMB rate	Window Size	RTT	Prod. rate	States	Trans.
Increase the size of the transmission							
1	3	∞	3	9	9	1,199	1,291
1	4	∞	3	9	9	2,677	2,929
1	5	∞	3	9	9	4,329	4,753
Evolution of the production rate							
1	6	∞	3	9	15	2,473	2,581
1	6	∞	3	9	10	2,957	3,202
1	6	∞	3	9	9	6,149	6,753
1	6	∞	3	9	8	3,372	3,854
1	6	∞	3	9	7	3,356	3,646
1	6	∞	3	9	5	10,793	13,657
1	6	∞	3	9	3	34,968	44,658
Evolution of the production rate (with wider window but same diff,)							
2	6	∞	4	9	15	6,937	7,411
2	6	∞	4	9	10	9,008	10,061
2	6	∞	4	9	9	30,659	35,122
2	6	∞	4	9	8	12,197	14,722
2	6	∞	4	9	7	14,918	17,542
2	6	∞	4	9	5	29,751	36,675
2	6	∞	4	9	3	128,329	166,095
Evolution of the production rate (with wider window and bigger diff,)							
2	6	∞	5	9	15	5,867	6,233
2	6	∞	5	9	10	7,331	8,111
2	6	∞	5	9	5	22,162	26,936
2	6	∞	5	9	4	17,078	20,725
2	6	∞	5	9	3	125,125	159,150
2	6	∞	5	9	2	37,777	45,744

Table 6: Same experiments that in Table 3 with RTT=9

# of losses	# of mess.	AMB rate	Window Size	Prod. rate	RTT=8		RTT=9	
					States	Trans.	States	Trans.
1	5	2	3	9	4,737	5,119	16,666	20,719
1	5	2	3	8	16,666	20,719	4,611	5,008
1	5	2	3	5	5,514	6,182	11,785	14,781
1	5	2	3	4	85,391	116,517	22,207	26,61
1	5	2	3	3	18,383	22,568	150,571	205,007
1	5	2	3	2	1,406,428	1,935,306	218,920	276,457
1	5	2	3	untimed	1,345,093 / 2,594,217			
1	7	∞	2	9	4,394	4,619	6,800	7,880
1	7	∞	2	9	6,800	7,880	4,569	4,824
1	7	∞	2	9	5,133	5,561	12,427	15,497
1	7	∞	2	9	43,420	58,878	14,158	16,451
1	7	∞	2	9	13,311	15,621	85,807	116,624
1	7	∞	2	9	1,094,280	1,531,442	85,807	116,624
1	7	∞	2	untimed	689,024 / 1,287,932			

Table 7: Comparing timed and untimed systems

- First, we showed that the minimal property that losses get signalled at the end of the session is not necessarily respected. It will be necessary for this property to be respected that a "closing SPM" mechanism is implemented.
- Then, we looked at the role of the nodes elements in the protocol. If this role is obvious as relay for the messages, it wasn't clear if they could be suppressed in the study of the protocol: do they have more effect than simple relays? In fact, we showed that nodes have to be taken into account when studying the protocol: they introduce a decoupling between the `NCF` answer and the `RDATA` messages that follow them which implies a more complex behaviour than if the nodes are suppressed.
- Last, we tried to guess what should be the relations between the various parameters of the protocol for ensuring the recovery of all losses. The exhaustive simulation tool enabled to verify this relation on a whole set of different configurations.

But this study also reveals the well-known limitations of existing tools. That is, the studies were done with finite-stated models, whereas the nature of the protocol implies that infinite behaviour should be studied. This would be the goal of the following studies. First, in section 5, we will study the protocol with the number of transmitted packets being a parameter, whereas the previous study uses at most something like 10 messages. Second, in section 6, we will study the protocol with any kind of topology, whereas the previous study uses a fixed topology of one sender, one node, one receiver. Last, in section 4, we will study how it is possible to derive automatically a relation impliying that there is no loss, whereas the previous study needs to first "guess" this relation and then verify it for a few cases.

Another point that should be discussed is the usability of the tools in the telecommunication industry.

As we have seen in 3.3, the version of the IF language designed in the Advance project is powerful and expressive enough to have simple and complete descriptions of the PGM protocol. It can be argued that those descriptions are better than those done in the SDL language, in part because of the cleaner syntax of the IF language. A graphical syntax could be a nice add-on to this language. Also, the lack of a "loop" construct in the automaton framework has been resented as limiting. Nevertheless, it is possible to call external C code, which alleviates this limitation.

The associated tools enable to realize precise studies and verifications of the behaviour of specifications. Some limitations have been encountered during this first study. The more stringent are the following ones:

- Some properties to be verified had been described using the μ-calculus in the `evaluator` tool. A simpler language for expressing properties could be investigated.
- The outputs of the verification tools are difficult to handle. A "debugging" mode would be interesting, with step-by-step reachability and labelling of the transitions.

4 Parameter Constraint Synthesis for the PGM using TReX (year 3)

4.1 A first study with TReX

The aim of the work done with TREX is to synthesize automatically the constraint obtained as condition for the recover of all losses in section 3. However, we are not able to apply directly TREX on the model proposed in section 3 for two reasons:

- it uses features of IF which are not yet supported in TREX, mainly communications queues with delays and arrays;
- it contains too much variables that should be considered as infinite domain variables (more than 20 counters and more than 20 clocks).

We have worked on two models that simplify the model proposed in 3:

The first model has been obtained by applying the following abstractions:

- only one loss is considered;
- the communication media becomes one place buffers with a maximal delay for transmission of packets; and,
- all integer variables are abstracted on enumerated variables.

The resulting model contains 32 enumerated variables, 11 clocks, and 2 real parameters. In order to apply TReX on this model, we modified TReX to allow “succ” and “pred” operations on enumerated types. However, this abstraction does not allow us to synthesize the constraint because the exploration did not terminate.

The second model is obtained using the same abstraction on the number of losses and the communication media, but now we keep the integer variables and consider them as counters. The idea is to test if the presence of counters allows TReX to apply extrapolation techniques in order to speed up the termination. We obtained a model with 2 finite variables, 3 clocks, 15 counters and 8 parameters.

The analysis of this model did not terminate, but it gives us hints on the improvements that should be done in the exploration algorithm of TReX.

This first study done during the second year of the project was a first attempt at the constraint synthesis problem of the PGM protocol. The rest of this section describes the solutions we obtained during the last year of the project. A necessary first step was to make a new simplified model of the PGM protocol, which we describe in the next section.

4.2 PGM Modelling

4.2.1 Overview of the protocol

A “session” of the PGM protocol (a given data transfer from a source to a group of receivers) builds a tree: the source is the root of the tree, the receivers are the leaves, and the other network elements are intermediary nodes. This tree may change during the session by the dynamic join/leave of receivers. In the normal course of data transfer, a source multicasts sequenced (numbered) data packets (`ODATA`) belonging to its sliding window along a path of the distribution tree to the receivers. When a receiver detects missing data packets from the expected sequence, it unicasts repeatedly to the last network element of the path negative acknowledgements (`NAK`s) containing the sequence number of missing data. Network elements forward `NAK`s hop-by-hop to the source using the reverse path, and confirm each hop by multicasting a `NAK` confirmation (`NCF`) in response to the child from which the `NAK` was received. Receivers and network elements stop sending `NAK` at the reception of a corresponding `NCF`. Finally, the source itself receives and confirms the `NAK` by multicasting an `NCF` to the group. If the data missing is still in the window, repairs (`RDATA`) may be provided by the source in response to the `NAK`. The advance of the sending window is done only by time passing.

To avoid `NAK` implosion, PGM specifies procedures for `NAK` elimination within network elements in order to propagate just one copy of a given `NAK` along the reverse of the distribution path.

To avoid `NAK` implosion, PGM specifies procedures for `NAK` elimination within network elements in order to propagate just one copy of a given `NAK` along the reverse of the distribution path.

The basic data transfer operation is augmented by `SPM`s (Source Path Messages) packets from a source, periodically interleaved with `ODATA`. `SPM`s have two functions. First, they carry routing informations used to maintain up-to-date PGM neighbour information and a fixed distribution tree. Second, they complement the role of data packets when there is no more data to send by holding the state of the sender window. In this way, the receiver may detect data losses and send further `NAK`s.

4.2.2 Abstract model

Analysing the full PGM protocol is beyond limits of current verification tools because of dynamic topology, a lot of processes, counters and clocks. In order to study the main property — full reliability — we built a model which abstracts from dynamic joining/leaving a node, multiple senders, and `SPM`s. In [32, 33, 49] are discussed several solution to reduce the complexity of the PGM w.r.t. main property.

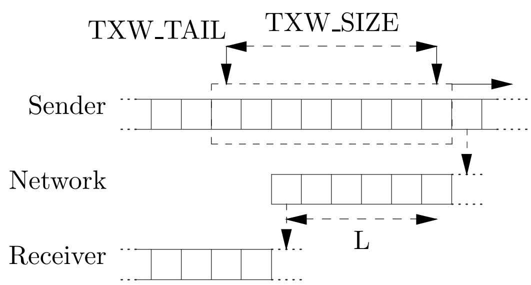


Figure 10: Global view of the protocol running in the sender and one of its receivers.

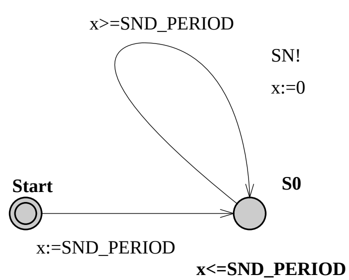


Figure 11: Sender extended timed automaton.

Our abstract model is based on a global view of the protocol running in the sender and one of its receivers, as presented on Figure 10. The intermediate nodes between the sender and the receiver are abstracted into a unreliable, unbounded FIFO queue. Only data packets (ODATA) are transmitted between the sender and the receiver, the other packets (SPM, NAK, NCF, RDATA) are abstracted also.

The sender fills the queue with ODATA packets at a rate equal to 1 / SND_PERIOD, where SND_PERIOD is a parameter. When an ODATA packet is sent, the sender advance its window by one. If the oldest packet in the window is numbered TXW_TRAIL and the size of the window is constant and equal to TXW_SIZE (a parameter), then the last packet sent has the number $\texttt{TXW\_TRAIL} + \texttt{TXW\_SIZE} - 1$. Note that we do not consider the initial phase of the protocol, when the sender window is not yet full of packets.

The network receives data from the sender and when its buffer is not empty, it delivers data to the receiver at a rate equal to 1 / CH_PERIOD. Depending on the relation between the reception rate and the deliver rate, the number of packets travelling on the network may be 0, 1, or more. We consider that the initial value of the number of packets in transit in the network is BUFFER_LENGTH, an integer parameter. If L is the current length of the queue, the packets stored in this queue belong to sequence numbers $[\texttt{TXW\_TRAIL} + \texttt{TXW\_SIZE} - L \,..\, \texttt{TXW\_TRAIL} + \texttt{TXW\_SIZE} - 1]$. The network may nondeterministically[9] loss packets by chucks of NLOSS packets, where NLOSS is another parameter. The lost packets should be in queue, i.e., $L \geq \texttt{NLOSS} + 1$. Moreover, in order to simplify the model, we consider that the losses are done at the head of the queue, just before a packet delivery. In this way, the receiver detects the loss immediately, because the packet received is numbered by $n + \texttt{NLOSS}$, where n is the expected sequence number. For the transmission of packets from the receiver to the sender, we consider that the transmission delay is CH_DELAY, another parameter.

The receiver accepts data from the network at the rate equal to 1 / CH_PERIOD. The sequence number of packet received is $\texttt{TXW\_TRAIL} + \texttt{TXW\_SIZE} - L' - 1$, where L' is the length of the buffer after the reception of the packet. If no loss, $L' = L - 1$. If a loss appears, $L' = L - \texttt{NLOSS} - 1$

[9]A more realistic model is to consider probabilistic loss. Although the theory of analysis by model-checking of probabilistic lossy channels has done important progress, there is neither tool nor theory which relate this domain with extended automata.

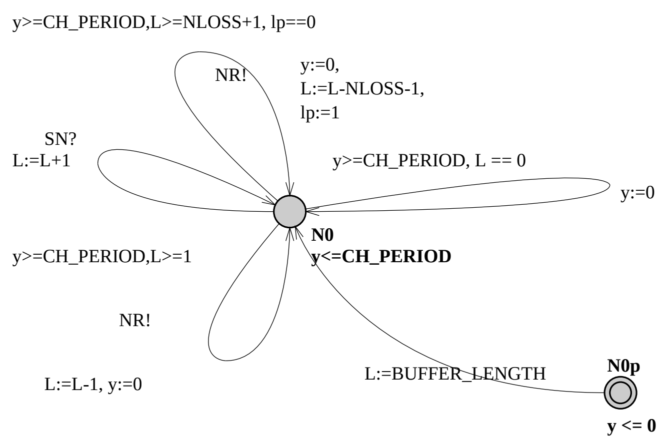


Figure 12: Network extended timed automaton.

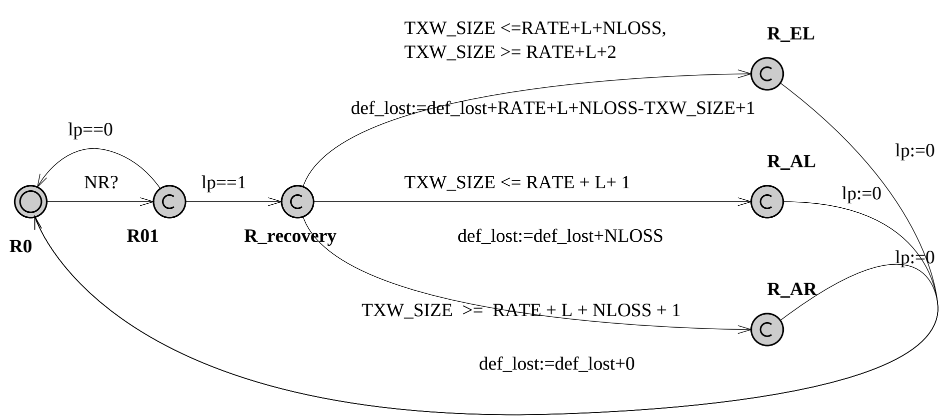


Figure 13: Receiver extended timed automaton.

and the packets lost have sequence numbers belonging to the interval $[\texttt{TXW\_TRAIL} + \texttt{TXW\_SIZE} - L \mathinner{..} \texttt{TXW\_TRAIL} + \texttt{TXW\_SIZE} - L' - 2]$. The receiver tries to recover the lost packets by sending the request for recover to the sender. But due to the abstraction to a global view of the system, the receiver may compute effectively which is the number of definitively lost packets, i.e., packets which cannot be recovered because they are no more in the sender window. Indeed, during the transmission of the request for recovery, the sender advances its window (increments TXW_TRAIL) by $\lceil \texttt{CH\_DELAY}/\texttt{SND\_PERIOD} \rceil$ packets, ratio denoted by RATE in the following. So only the packets having the sequence number greater or equal to $\texttt{TXW\_TRAIL} + \texttt{RATE}$ may be recovered.

Three cases are then possible, corresponding to three states of the receiver (R_AR, R_AL, resp. R_EL), following the position of $\texttt{TXW\_TRAIL} + \texttt{RATE}$ with respect to the interval of the lost packets $[\texttt{TXW\_TRAIL} + \texttt{TXW\_SIZE} - L \mathinner{..} \texttt{TXW\_TRAIL} + \texttt{TXW\_SIZE} - L' - 2]$:

$\forall$**R** All lost packets may be recovered if

$$\begin{aligned} \texttt{TXW\_TRAIL} + \texttt{RATE} &\leq \texttt{TXW\_TRAIL} + \texttt{TXW\_SIZE} - L & (2)\\ \texttt{TXW\_SIZE} &\geq \texttt{RATE} + L & (3)\\ \texttt{TXW\_SIZE} &\geq \texttt{RATE} + L' + \texttt{NLOSS} + 1 & (4) \end{aligned}$$

$\forall$**L** None of the NLOSS lost packets may be recovered if

$$\begin{aligned} \texttt{TXW\_TRAIL} + \texttt{RATE} &> \texttt{TXW\_TRAIL} + \texttt{TXW\_SIZE} - L' - 2 & (5)\\ \texttt{TXW\_SIZE} &\leq \texttt{RATE} + L' + 1 & (6) \end{aligned}$$

$\exists$**R** Some of the lost packets, i.e., $\texttt{RATE} + L' + \texttt{NLOSS} - \texttt{TXW\_SIZE} + 1$, may be recovered if

$$\begin{aligned} \texttt{TXW\_TRAIL} + \texttt{TXW\_SIZE} - L &< \texttt{TXW\_TRAIL} + \texttt{RATE} \leq \texttt{TXW\_TRAIL} + \texttt{TXW\_SIZE} - L' - 2 & (7)\\ \texttt{RATE} + L' + 2 &\leq \texttt{TXW\_SIZE} \leq \texttt{RATE} + L - 1 & (8)\\ \texttt{RATE} + L' + 2 &\leq \texttt{TXW\_SIZE} \leq \texttt{RATE} + L' + \texttt{NLOSS} & (9) \end{aligned}$$

In the relations above, L' should be replaced by $L - \texttt{NLOSS} - 1$ to obtain a constraint on the length of the buffer before loss.

This abstracted model has been implemented using extended timed automata as shown on Figures 11, 12, and 13. These automata communicate by rendez-vous on gates SN and NR, and by shared variables L and lp (boolean to mark that a lost is produced). The states labelled by C are urgent states, i.e., states where the time does not advance.

4.3 Full reliability constraint

The abstract model proposed in the previous section gives in part the constraint to be satisfied by the parameters in order to obtain full reliability. The problem is that in equations (2) and followings L' (the current value of L) is a variable.

To obtain a relation depending only on time and the parameters of the model, we should compute L like a function:

$$L = f(t, \texttt{BUFFER\_LENGTH}, \texttt{SND\_PERIOD}, \texttt{CH\_PERIOD}, \texttt{NLOSS}) \quad (10)$$

Remark. We suppose in the following that variables SND_PERIOD, CH_PERIOD, and NLOSS are parameters, i.e., symbolic constants whose values are fixed from the beginning and do not vary. This hypothesis simplify the analysis and it is at the base of the automatic analysis with TReX.

However, in the analysis below, the theoretical results are valid either if the parameters vary in time while preserving the constraints separating the cases below.

In computing L, we distinguish four cases:

Case 1: $\texttt{SND\_PERIOD} > \texttt{CH\_PERIOD}$, i.e., the rate of arrivals in queue is less than the one of departures. Then, L decreases to 0, decrease which may be accelerated by losses. The extreme values of L are:

$$0 \leq L \leq \texttt{BUFFER\_LENGTH} \quad (11)$$

where BUFFER_LENGTH is the initial value of the queue.

Case 2: $\mathtt{SND\_PERIOD} = \mathtt{CH\_PERIOD}$, i.e., arrivals in queue are at the same speed as "normal" departures by message deliver. The size of the queue is constant, and equal to $\mathtt{BUFFER\_LENGTH} - [\mathtt{BUFFER\_LENGTH}/\mathtt{NLOSS}]\,\mathtt{NLOSS}$, because losses may be done while the buffer has enough data. The extreme values of L are:

$$\mathtt{BUFFER\_LENGTH} - \left[\frac{\mathtt{BUFFER\_LENGTH}}{\mathtt{NLOSS}}\right]\mathtt{NLOSS} \leq \quad L \quad \leq \mathtt{BUFFER\_LENGTH} \tag{12}$$

Case 3: $\mathtt{NLOSS} \leq \mathtt{CH\_PERIOD}/\mathtt{SND\_PERIOD} - 1$, i.e., arrivals in queue are faster than the sum of departures by delivery and loss. If we consider that a loss may appear at each delivery, we obtain a minimum value for L,

$$L = \mathtt{BUFFER\_LENGTH} + \left[\frac{t}{\mathtt{CH\_PERIOD}}\right]\left[\frac{\mathtt{CH\_PERIOD}}{\mathtt{SND\_PERIOD}} - 1 - \mathtt{NLOSS}\right]$$

where t is the time elapsed from the beginning of analysis. Then, the maximal value of L is not defined, and we obtain that:

$$\mathtt{BUFFER\_LENGTH} \leq \quad L \quad < \infty \tag{13}$$

Case 4: $\mathtt{NLOSS} > \mathtt{CH\_PERIOD}/\mathtt{SND\_PERIOD} - 1 > 0$, i.e., arrivals in queue are faster than "normal" departures, but not enough to fill the losses between two delivery. An exact description of this case is done by the following relation:

$$L = \mathtt{BUFFER\_LENGTH} + \left[\frac{t}{\mathtt{CH\_PERIOD}}\right]\left[\frac{\mathtt{CH\_PERIOD}}{\mathtt{SND\_PERIOD}} - 1\right] - k\mathtt{NLOSS}$$

where k is the number of losses done from the beginning of analysis. If we consider that a loss may appear each time the buffer has enough data, the minimum value of L is 0. Otherwise, the length of the queue may increase over any limit. Then, for this last case:

$$0 \leq \quad L \quad < \infty \tag{14}$$

This analysis leads to important conclusions.

Property 1: The full recovery of data lost may be done all the time only in the first case above, since in the other cases there is no maximum value for L. The constraint that should be satisfied by the parameters of the protocol is:

$$\mathtt{SND\_PERIOD} \geq \mathtt{CH\_PERIOD} \ \wedge \ \mathtt{TXW\_SIZE} \geq \mathtt{RATE} + \mathtt{BUFFER\_LENGTH} \tag{15}$$

Partial recovery of losses is always possible, for any relation between arrivals, departures, and losses rates, when

$$\mathtt{TXW\_SIZE} > \mathtt{RATE} + \mathtt{BUFFER\_LENGTH} - \mathtt{NLOSS} \tag{16}$$

None of the losses may be recovered in any case if

$$\mathtt{TXW\_SIZE} \leq \mathtt{RATE} - \mathtt{NLOSS} \tag{17}$$

The proof of this property is easy to obtain from equations 2–7, the fact that $L' = L - \mathtt{NLOSS} - 1$ and the equations 11–14.

Remark: The constraints between the parameters and the law of evolution of L are non-linear relations on reals and integers. Indeed, the exact computation of L variable leads to non-linear expressions between integer variables. Also, the law of evolution of L changes at a border given by a non-linear expression between reals and integers. This observation does not change if the model for losses is changed to a periodical policy, since we will need to divide time parameters by the period of losses.

Case 1: `SND_PERIOD` > `CH_PERIOD`						
Experiment	`SND_PERIOD`	`CH_PERIOD`	*#param*	*Time*	*Mem*	*#acc*
Partially inst.	2	1	4	00:00:37		1
	3	1		00:00:35		2
	4	1		00:00:59		3
	5	1		00:01:04		3
	3	2		00:01:24		3
	4	2		00:00:36		1
	5	2		00:06:19		6
	4	3		00:01:36		3
	5	3		00:01:23		3
	5	4		00:01:35		3
Fully param.	-	-	6	>02:00:22	>200	> 6

Table 8: Experiments for one time loss, case 1.

The last remark implies that the analysis of this model should be done by instantiating some of parameters in order to avoid non-linear constraint, and/or by applying acceleration and/or approximation techniques.

We will now show how we deal with the analysis by model-checking of the abstract model of PGM. This analysis will consist in automatically generating the relations necessary for reaching one of the states representing either complete recovery or partial recovery or definitive loss of the packets in the abstract model of the receiver process. For reducing non-linearities, we consider first that there is only one chunk of `NLOSS` packets lost.

4.4 Analysis of a model with one chunk of lost packets

We consider here the case where only one chunk of `NLOSS` packets is lost. This means that in the receiver's automaton, we remove the transitions going to state `R0` after a loss (from states `R_AL`, `R_EL`, and `R_AR`). Since we obtain the constraints satisfied by the parameters at the first loss, only partial results are obtained for some of four cases presented in the previous section, as will be explained in the remainder of this section.

The model obtained is a system of three automata communicating via synchronisation gates and extended with one finite variable (`lp`), two clocks (`x` and `y`), two counters[10] (`def_lost` and `L`), and six (integer or real) parameters (`SND_PERIOD`, `CH_DELAY`, `TXW_SIZE`, `RATE`, `BUFFER_LENGTH`, `NLOSS`).

All experiments reported below have been done on a Pentium III at 1GHz and 512MB.

TReX has been tuned to extrapolate only loops of depth less than 6 (this value has been chosen experimentally). Moreover, when an extrapolation of a loop generated non-linear constraints between parameters, the extrapolation had been cancelled (i.e., the reachability continues in an exact manner for the loop with the risk of not terminating it).

Case 1: `SND_PERIOD` > `CH_PERIOD`.
The fact that we deal with only one loss is not important since the number of losses that may be done is bounded by the initial size of the buffer.

We did two kind of experiments: (1) fully parameterized by adding the constraint `SND_PERIOD` > `CH_PERIOD` to the initial constraint on parameters, and (2) partially parameterized by instantiating parameters `SND_PERIOD` and `CH_PERIOD`.

Memory consumption is weak (less than 3MB) for partially instantiated experiments.

The symbolic graphs obtained for this case contain the three states representing the complete or partial recovery of losses, and the total loss of all packets (`R_AL`, `R_EL`, and `R_AR`). Those states are reachedfor the following constraints (we give the stronger one involving the parameters):

- `R_AR`:

[10]Only variable `L` counts in the model with only one loss.

Case 2: SND_PERIOD = CH_PERIOD						
Experiment	SND_PERIOD	CH_PERIOD	*#param*	*Time*	*Mem*	*#acc*
Fully param.	-	-	5	00:00:06		0

Table 9: Experiments for one time loss, case 2.

```
 txw_size >=  rate + buffer_length and
 buffer_length >= nloss + 1
```

which is exactly the constraint given in equation 15.

- R_EL:

```
txw_size >= rate + buffer_length - nloss - n3 - 1  and
txw_size <= rate + buffer_length - n3 - 3  and
buffer_length >= nloss + n3 - 3  and
buffer_length >= n3 - 2  and
n3 >= 0
```

which is a generalisation of equation 16 and its limit (for n3=0) is exactly this equation.

- R_AL:

```
txw_size <= rate - nloss + buffer_length - n3 - 1  and
buffer_length >= nloss + n3 - 2  and
buffer_length - n3 - 1 <= 0 and
n3 >= 0
```

which is a generalisation of equation 17 and its limit (for n3=buffer_length-1) is exactly this equation.

Case 2: SND_PERIOD = CH_PERIOD.
This case can be dealt fully parameterized with TReX.

The constraints obtained are the same than for the previous case:

- R_AR:

```
 txw_size >=  rate + buffer_length
```

- R_EL:

```
txw_size >= rate + buffer_length - nloss + 1  and
txw_size <= rate + buffer_length - 1
```

- R_AL:

```
txw_size <= rate + buffer_length - nloss
```

Case 3: NLOSS $\leq$ CH_PERIOD/SND_PERIOD $-$ 1.
For the last two cases, we introduce a new parameter, q, for CH_PERIOD/SND_PERIOD, and we consider that q ≥ 2.

The experiments done for those cases instantiate CH_PERIOD as follows:

- Inst. (1): by a constant. SND_PERIOD is also instantiated.
- Inst. (2): by the expression $k *$ q where k is a constant value representing SND_PERIOD.
- Par.: by the expression SND_PERIOD $*$ q.

The constraints obtained are similar to those for the first case. Since we consider the case of one time loss, we cannot obtain that it is not possible to always recover losses.

Case 3: `NLOSS` $\leq$ `CH_PERIOD/SND_PERIOD` $- 1$						
Experiment	`SND_PERIOD`	`CH_PERIOD`	*#param*	*Time*	*Mem*	*#acc*
Inst. (1)	2	4	4	00:00:09		1
	3	6		00:00:09		1
	2	6		00:00:09		1
	3	9		00:00:09		1
	4	12		00:00:09		1
	2	8		00:00:09		1
Inst. (2)	2	2q	5	00:00:27		2
	3	3q		00:00:27		2
	4	4q		00:00:27		2
Par.	d	dq	6	00:05:04		2

Table 10: Experiments for one time loss, case 3.

Case 4: `NLOSS` $>$ `CH_PERIOD/SND_PERIOD` $- 1$						
Experiment	`SND_PERIOD`	`CH_PERIOD`	*#param*	*Time*	*Mem*	*#acc*
Inst. (1)	2	4	4	00:00:11		1
	3	6		00:00:11		1
	2	6		00:00:08		1
	3	9		00:00:08		1
	4	12		00:00:08		1
	2	8		00:00:08		1
Inst. (2)	2	2q	5	00:00:26		2
	3	3q		00:00:26		2
	4	4q		00:00:26		2
Par.	d	dq	6	00:05:10		2

Table 11: Experiments for one time loss, case 4.

Case 4: `NLOSS` $>$ `CH_PERIOD/SND_PERIOD` $- 1 > 0$.
The experiments and the results obtained are similar to the third case.

4.5 General analysis with several chunks of lost packets

One interesting point for the analysis of the general model is to show (Property 1) that there is no way to always recover losses in the third and fourth cases. This can be obtained by searching in the symbolic graphs obtained for the model with several losses the paths where `def_lost` is always equal to 0. The problem is to be able to generate this graph. The theoretical analysis done shows that the expression computed for L is complex, so its automatic computation may fail.

Another interesting point is to compute the number of definitively lost packets (`def_lost`) w.r.t. the number of losses. But for this, we should be able to deal a class of assignments for counters which is out of the class allowed by the DBM data structures. For this reason, we need another data structure for counters which allows more general assignments and have efficient extrapolation algorithms. We have not been able to complete those more general analyses within the existing state of the TReXtool.

5 Verification with infinitely many packets using LASH (year 2)

5.1 Introduction

The *Liège Automata-based Symbolic Handler (LASH)* is a tool for building and manipulating finite-state representations of sets, and for exploring the state-space of infinite-state systems. This document describes analysis experiments of a simple IF model of the PGM protocol using LASH.

5.2 Protocol Model

5.2.1 Restrictions

The state-space exploration technique implemented in LASH is not able to handle the general IF specification of PGM as provided by Verimag and France Telecom. The following restrictions have been applied in our model:

- All processes are instantiated statically. Indeed, LASH does not handle dynamic features, because it relies on *meta-transitions* which can currently only be computed on static structures.
- The integer data type is the only one to be handled. This is due to the symbolic representation system used in the current version of the state-space exploration module of LASH. As a consequence, clocks, timers and signals have been abstracted out of our model.

5.2.2 Structure

The model described in this section is written in IF 1.0. There are four static processes: *Init*, *Source*, *Network Element*[11] and *Receiver*, communicating with each other by means of synchronizations. The processes *Source*, *Network Element* and *Receiver* represent respectively a transport-layer originator of PGM data packets, a network-layer entity and a transport-layer consumer of PGM data packets. The process *Init* is used to initialize the system. No additional source, network element or receiver is instantiated during execution.

Because of this static structure, the `SPM` messages have been abstracted out. These messages carry the source-assigned information regarding the edges of the source's transmit window (`TXW_TRAIL` and `TXW_LEAD`), and the information regarding the path followed by the messages from the source to the receiver. The former element is merely a duplication of the same information embodied in the `ODATA` and `RDATA` messages, and the latter one is constant in our model.

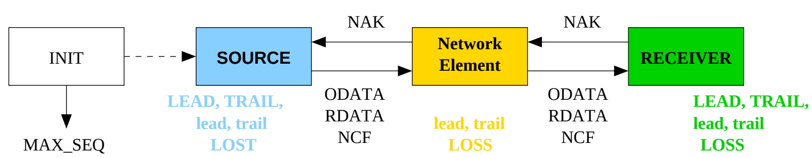


Figure 14: Abstracted PGM model.

5.2.3 Behavior

The Source sends `ODATA` messages to the Network Element, which forwards them directly to the Receiver. If the Receiver identifies a missing message with a sequence number still belonging to the transmit window, it sends a `NAK` to the Network Element, until it gets back a `NCF`. Upon reception of a `NAK`, the Network Element sends a `NCF` to the Receiver, establishes a repair state for the `NAK` if it does not already exists, and sends `NAK` messages to the Source until it eventually gets back a `NCF`. Finally, if the missing message sequence number is still within the transmit window, the Source sends a `RDATA` to the Network Element, which forwards it to the Receiver.

5.2.4 Transmit and Receive Windows

Any message is associated with a sequence number ranging from 0 to a bound called `TXW_MAX_SEQ`. This bound can either be a specific value, or a parameter.

[11] That is, a node of the network.

The *Transmit Window* maintained by the Source is delimited by two sequence numbers, `TXW_TRAIL` and `TXW_LEAD`. Any message with a sequence number that belongs to this interval can be "repaired", i.e., sent as an `RDATA` in the case of a message loss. `TXW_LEAD` is incremented each time a `ODATA` is sent, and `TXW_TRAIL` is incremented non-deterministically. The maximum window size is fixed by the variable `TXW_MAX_RTE` (in our model). When `TXW_LEAD` or `TXW_TRAIL` reaches `TXW_MAX_SEQ`, it is reset to 0.

The Network Element does not maintain any window (any message `ODATA` received is thus forwarded directly), and only `RDATA` corresponding to the repair state (if any) is forwarded.

Finally, the Receiver maintains a *Receive Window*, delimited by the variables `RXW_TRAIL` and `RXW_LEAD`. When an `ODATA` is received, `RXW_LEAD` is updated according to the value of the *lead* field of the message. `RXW_TRAIL` is incremented non-deterministically but, if its value corresponds to an older message than the oldest message available for repair from the source (as indicated by the *trail* field of the `RDATA` and `ODATA` messages), `RXW_TRAIL` is set equal to *trail*. As in the case of the Transmit Window, when `RXW_LEAD` or `RXW_TRAIL` reaches TXW_MAX_SEQ, it is reset to 0.

5.2.5 Message Transmission

The messages are transmitted from one process to another by means of *synchronizations*. One gate is used per message (`ODATA`, `RDATA`, `NAK`, `NCF`) and per direction, hence a total of eight gates are used for the communication. In addition, one gate is used internally in order to generate an initial non-deterministic value for the parameter `TXW_MAX_SEQ`. The synchronization mechanism acts like a queue of zero size (i.e., a rendez-vous), preserving the order of messages.

5.2.6 Loss Modeling

In our model, at most one `ODATA` is lost non-deterministically between the Source and the Network Element. The sequence number of the lost message corresponds to the value of the `LOSS_SEQ_NB` variable.

The Receiver detects a loss as soon as the sequence number of a received `ODATA` is not contiguous to the sequence number of the last accepted `ODATA`. This is done by comparing the *lead* field of the message with `RXW_LEAD`. Once a loss is detected, the sequence number is stored in a local variable (`LOSS`). This variable is set to -1 as long as no loss is pending. If the trailing edge of the Receive Window moves beyond the sequence number of the loss, or if an `RDATA` corresponding to the loss is received, data recovery for that sequence number terminates by resetting `LOSS` to -1.

The Network Element creates a repair state as soon as it receives a `NAK` from the receiver. This repair state is modelized by a variable that holds the value of the loss sequence number. As soon as a `RDATA` corresponding to the repair state is received by the Network Element, the repair state is removed by resetting `LOSS` to -1.

5.3 Analysis

5.3.1 Goals

Our verification experiment pursues two goals. The first one is merely technical and can be viewed as a debugging tool. The goal is to get knowledge of the model behaviour by analyzing the reachable values of the auxiliary variables (mostly required to ensure a correct resetting of the windows boundaries), and to correct undesired behaviours. This analysis also encompasses the performance impact of the meta-transitions, which are required in order to explore infinite state space with finite resources. Since the meta-transition computation algorithms are costly both in time and memory, a trade-off is often required for the number of meta-transitions that must belong to the model.

The second verification goal concerns the analysis of the state space and, specifically, the relationships between the variables and the global and local properties of the system. Specifically, our main goal was to study closely the relationships between the `LEAD` and `TRAIL` values of the windows.

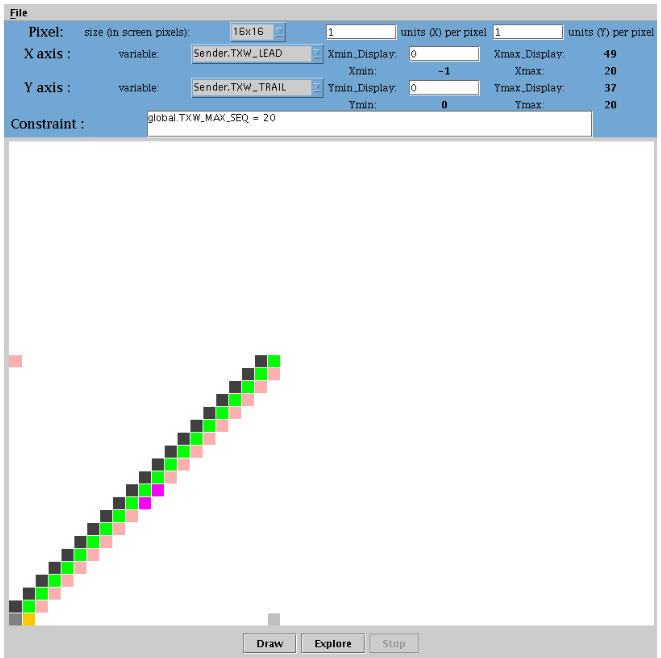


Figure 15: Transmit Window.

5.3.2 Tools

The LASH toolset provides two means for analysing the state space of the system. The first method consists in inserting some assertion within the IF code of the model, which is then either validated or disproved by the state-space exploration. A second method, more interactive, relies on a prototype tool for visualizing the reachable values of a given pair of variables. In this document, we will mainly use the latter.

5.3.3 Experiments

The PGM model described in 5.2 has been fitted with a Transmit Window size (`TXW_MAX_RTE`) of 2, and with a loss sequence number set to 10. Two sets of experiments were conducted on the resulting model. We started by setting explicitly a maximum sequence number `TXW_MAX_SEQ` for messages. In a second phase, we let `TXW_MAX_SEQ` be a free parameter of the model (which, in this case, is infinite-state).

Specific `TXW_MAX_SEQ`. We chose the value 20 for `TXW_MAX_SEQ` because it is large compared to the Transmit Window size (in our model, 2), but still manageable.

Figure 15 displays the reachable values of the variables `TXW_LEAD` (x-axis) and `TXW_TRAIL` (y-axis) of the Source, each color corresponding to a different exploration step. The lower left corner of the graph corresponds to the pair (0,0), and the top left to the pair (20, 20). Figure 16 displays the evolution of this graph during the state space exploration.

Three interesting observations can be inferred from this picture. First, the resetting of the variables has been successfully implemented and amounts to a modulo operation. Second, although the transmit window size is 2, each value of one variable corresponds to three possible values for the other one. Finally, the effect of a meta-transition is noticeable. Indeed, all the pairs of identical values are reached after a single exploration step. The cyclic path represented by this meta-transition corresponds to the following behaviour of the system : The source sends an `ODATA`, which is forwarded by the Network Element to Receiver, and the `TXW_TRAIL` is then incremented.

Let focus now on the Receive Window. Figure 17 displays the variables `RXW_LEAD` (x-axis) and `RXW_TRAIL` (y-axis) of the Receiver.

An interesting observation is the breadth of the diagonal strip, which indicates that for each

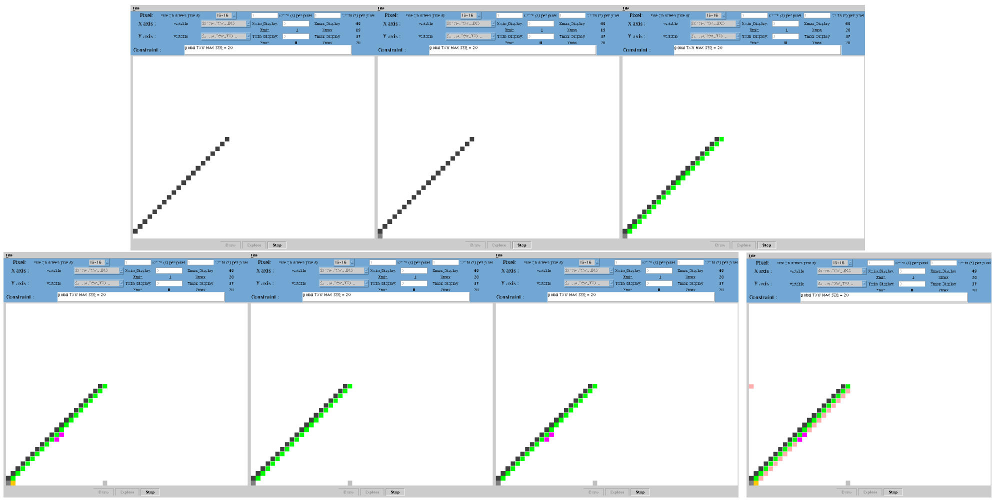


Figure 16: Transmit Window: Sequence during state-space exploration.

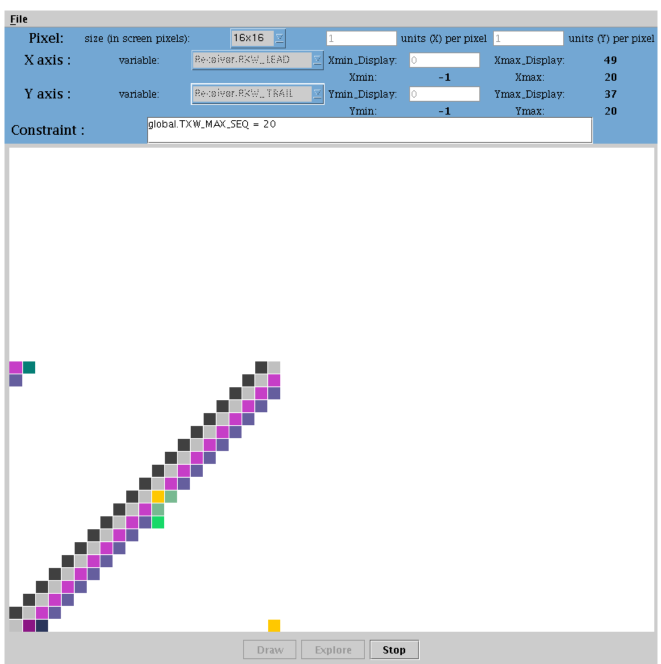


Figure 17: Receive Window.

Figure 18: Receive Window: Sequence during state-space exploration.

value of one variable the other one may take four different values. An exception to this rule is the extra value noticeable at (11,8), corresponding to the possible message loss. Indeed, when the receiver gets an `ODATA`, it updates first its variable `RXW_LEAD` and then updates `RXW_TRAIL`. So, when a loss appears, the gap between the updated value of `RXW_LEAD` and the "old" `RXW_TRAIL` is maximum.

As an example of the possibilities offered by LASH, we display in Figure 19 the values of the reachable bounds of the Transmit Window in the particular case of the extreme value (11,8) reached by the Receive Window. (This displayed set of values is simply obtained by applying elementary set operations on the sets previously generated during the state space exploration.) The advantage of this approach is that the computation of the reachable state space of the model has only to be performed once. Visualizing interesting features of this set, such as the one discussed here, is then fast (a few seconds) and easy.

`TXW_MAX_SEQ` as a parameter. In those series of tests, we do not fix the value of `TXW_MAX_SEQ`, which is then chosen non-deterministically among all the positive integers prior to running the model. Obviously, since the model becomes infinite-state, its exploration with a finite amount of resources is only possible if it contains suitable meta-transitions.

By taking the same value (10) for `LOSS_SEQ_NB` and `TXW_MAX_RTE`, we obtain the Figures 20 and 21, corresponding respectively to Figures 15 and 17.

This parametrized model generalizes the models in which a specific value of `TXW_MAX_SEQ` is specified. Notice that any pair of values appearing in the Figures 15 and 17 appears also in the Figures 20 and 21. This provides an efficient mean to test for several values of a parameter. It is sufficient to explore the parametrized system and construct its set of reachable states, and in a second phase, constrain the parameter to be equal to the desired values.

5.4 Analysis Costs

Table 12 summarizes the analysis costs for the model described in this document. (In the first part of the table, the value of `TXW_MAX_SEQ` is explicitly specified in the analyzed model; the last line

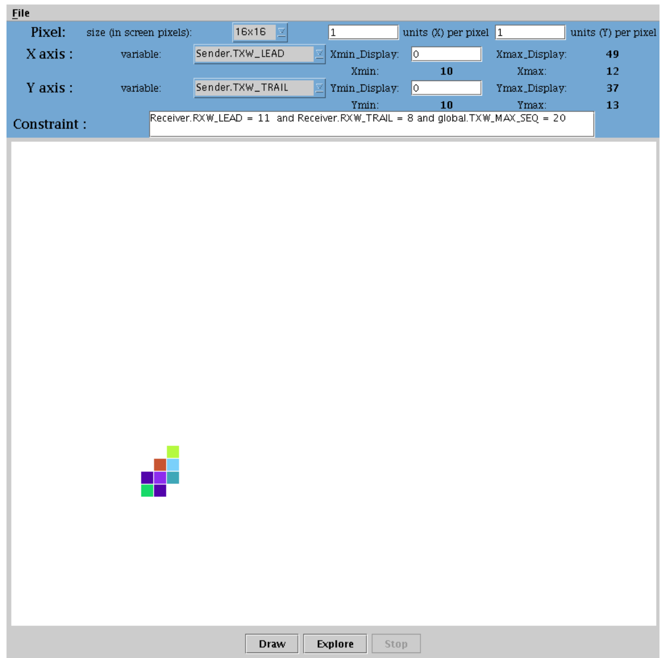


Figure 19: Constrained Transmit Window.

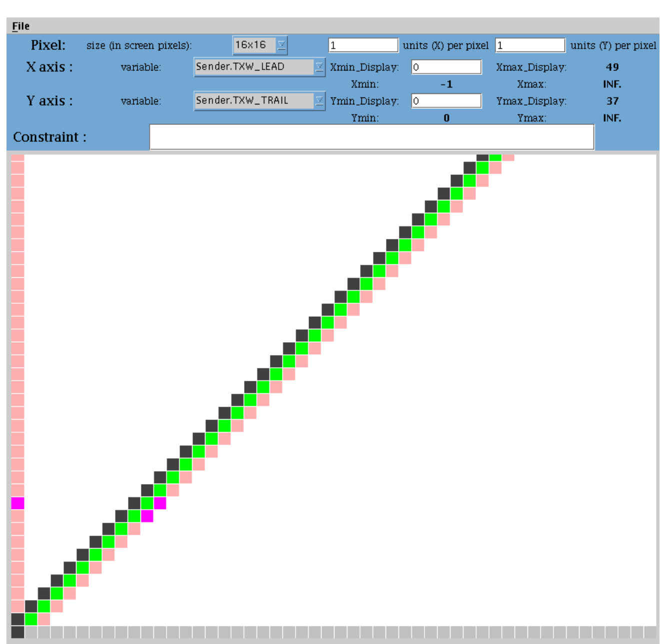


Figure 20: Transmit Window in parametrized case.

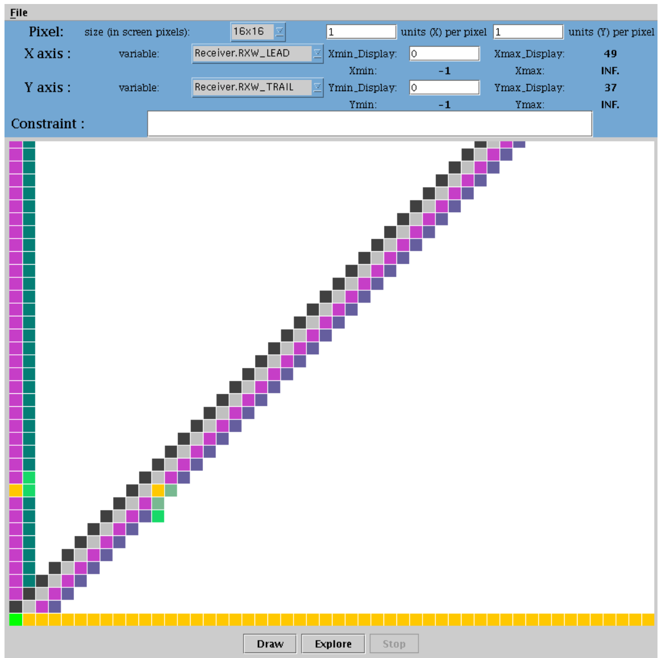


Figure 21: Receive Window in parametrized case.

TXW_MAX_SEQ	Nb_steps	Nb. NDD States	Nb. States	Max. Mem.	Time
20	62	100 K	26 K	125 Mb	7h42
200	62	160 K	249 K	283 Mb	10h19
2000	62	219 K	2,479 K	467 Mb	17h09
20000	62	294 K	24,781 K	719 Mb	35h59
ND	76	225 K	INF.	279 Mb	23h52

Table 12: Analysis costs

corresponds to the parametrized case.) For each value of `TXW_MAX_SEQ`, the table gives the number of exploration steps before reaching a fixpoint, the number of internal states of the finite-state representation of the set of reachable states, and the number of elements in this set. The time and memory costs are also given.

On a logarithmic scale, the number of reachable states follow linearly the value of `TXW_MAX_SEQ`. This behaviour is explained by noticing that the windows bounds remain close to the current sequence number of messages, which ranges from 0 to `TXW_MAX_SEQ`. The values of the other auxiliary variables do not depend on `TXW_MAX_SEQ`. An interesting feature is that the number of exploration steps remain constant when `TXW_MAX_SEQ` grows. This indicates that the meta-transitions manage to capture the repeated incrementation of ODATA sequence numbers. The size of the internal representation is logarithmic in the value of TXW_MAX_SEQ. This indicates that the internal representation handles adequately this type of system behaviour captured by the meta-transitions.

The behaviour of the parametrized system is worth noticing. Although the result of its state-space exploration contains sufficient information for building exactly the set of reachable states of every model in which the value of `TXW_MAX_SEQ` is fixed, the resources needed by the computation are similar to those required by a fixed value of 2000.

5.5 Property Validation

After this global analysis of the system behaviour, we can now write down a property which sets precisely the range of values that the Receive Window size is thought to take, and check that this property holds in all model executions. The property that has been chosen is given below. Intuitively, there are two possible situations, whether a loss is pending or not. For each of these situations, there are three cases: a nominal one (the diagonal strip on the figures), a case in which only `RXW_LEAD` has been reset, and a case in which only `RXW_TRAIL` has been reset.

```
(Sender.LOST = 0 and
  ( ( Receiver.loop_LEAD = 0 and Receiver.loop_TRAIL = 0 and
      Receiver.RXW_LEAD - Receiver.RXW_TRAIL <= 2 and
      Receiver.RXW_LEAD - Receiver.RXW_TRAIL >= -1)
    or
    ( Receiver.loop_LEAD = 1 and
      Receiver.RXW_LEAD + global.TXW_MAX_SEQ - Receiver.RXW_TRAIL <= 2 and
      Receiver.RXW_LEAD + global.TXW_MAX_SEQ - Receiver.RXW_TRAIL >= -1)
    or
    ( Receiver.loop_TRAIL = 1 and
      Receiver.RXW_LEAD - global.TXW_MAX_SEQ - Receiver.RXW_TRAIL <= 2 and
      Receiver.RXW_LEAD - global.TXW_MAX_SEQ - Receiver.RXW_TRAIL >= -1)))
or
(Sender.LOST = 1 and
  ( ( Receiver.loop_LEAD = 0 and Receiver.loop_TRAIL = 0 and
      Receiver.RXW_LEAD - Receiver.RXW_TRAIL <= 3 and
      Receiver.RXW_LEAD - Receiver.RXW_TRAIL >= -1)
    or
    ( Receiver.loop_LEAD = 1 and
      Receiver.RXW_LEAD + global.TXW_MAX_SEQ - Receiver.RXW_TRAIL <= 3 and
      Receiver.RXW_LEAD + global.TXW_MAX_SEQ - Receiver.RXW_TRAIL >= -1)
    or
    ( Receiver.loop_TRAIL = 1 and
      Receiver.RXW_LEAD - global.TXW_MAX_SEQ - Receiver.RXW_TRAIL <= 3 and
      Receiver.RXW_LEAD - global.TXW_MAX_SEQ - Receiver.RXW_TRAIL >= -1)))))
```

This property is validated by generating the set of values satisfying its negation, and then checking whether the intersection of this set by the projection of the computed set of reachable values over the relevant variables is empty. Once the reachable part of the state space is computed, the projection, intersection and test emptiness operations take only a few seconds.

5.6 Conclusions

We have conducted successfully experiments in which abstracted versions of a PGM model have been analyzed, both in the case of a fixed bound on the message sequence numbers, and with a parametrized model. The methodology described in this document indicates the essential feedback provided by our prototype visualization tool.

Finally, the performance analysis indicates that running LASH is still very time consuming. The positive aspect is that the state-space exploration needs to be achieved only once, since checking a property on a computed representation of the reachable state-space can be done with little additional cost.

6 Verification with infinitely many topologies (year 2)

We considered an abstract version of the PGM protocol without time. The PGM protocol consists of a sender, a receiver and a network element process. It is parameterized and can be instantiated in form of any tree that has the sender at the root, receivers at leaves and network elements at all internal nodes. So there is an infinite set $\mathcal{N}(PGM)$ of instantiations of the PGM protocol. Verifying a property for a parameterized protocol means showing that any instantiation satisfies the property.

We addressed the problem of dealing with instantiations of arbitrary shape by reducing the instantiations that have to be considered. More precisely, we identified subsets $\mathcal{N}'(PGM) \subseteq \mathcal{N}(PGM)$ such that all instantiation in $\mathcal{N}'(PGM)$ satisfies the property if and only if all instantiations in $\mathcal{N}(PGM)$ satisfy the property. Thereby, we restricted ourselves on properties that speak only of the sender and the receivers.

6.1 Reduction to simple instantiations for the PGM protocol

We started by proving that it suffices to consider the set of instantiations $\mathcal{N}^1(PGM)$ that contains all trees of depth at most 3, i.e. all instantiations with only one layer of network elements. The proof idea is to show that any $T \in \mathcal{N}(PGM)$ can be simulated (with respect to the sender and the receivers) by some $T' \in \mathcal{N}^1(PGM)$.

Theorem 1 ([67]) *Let ϕ be a property speaking only about the sender and receivers. $\mathcal{N}(PGM) \models \phi$ if and only if $\mathcal{N}^1(PGM) \models \phi$.*

While instantiations in $\mathcal{N}^1$ are easier to handle than general instantiations, $\mathcal{N}^1$ is still infinite because there can be any number of receivers. In a second step, we observed that if one is only interested in the behaviour of a bounded number of receivers, and if the number of messages of the protocol is finite, it is possible to restrict the number of receivers that have to be considered. We showed that if the number of data packets sent in the PGM protocol is bounded, then there is a particular element $T \in \mathcal{N}^1$ for which it suffices to check the property. This works because the network element and the receiver of the PGM protocol can iterate to send a particular message any number of times. (Such behaviour can occur if the agent does not receive messages any more, i.e. all messages sent to it are lost.)

Theorem 2 ([67]) *Let ϕ be a property speaking only about the sender and k receivers, and let the PGM protocol be restricted to a finite set of data packets. Then there is an instantiation $T \in \mathcal{N}^1$ such that $T \models \phi$ if and only if $\mathcal{N}^1(PGM) \models \phi$.*

6.2 Generalizing the approach

In the next step, we wanted to formulate the approach we took for verification of the PGM protocol in an abstract way. Thereby be made clear under which conditions such reductions are possible.

In [49], we defined an abstract framework for multicast protocols with instantiations in form of trees.

For a multicast protocol P, let $\mathcal{N}(P)$ be the set of all instantiations, and let $\mathcal{N}^1(P)$ be the instantiations with only one layer of internal nodes. We say that a protocol P is collapsable if it suffices to consider instantiations in $\mathcal{N}^1(P)$ in order to prove properties about $\mathcal{N}(P)$ that only speak about the sender and receivers.

The main part of [49] consists of defining conditions under which a multicast protocol is collapsable. The main restriction concerns the network elements. In the PGM example, network elements can forward incoming messages, i.e. multicast messages from above to successors, and send messages from below to the predecessor. Thereby, downward messages can be filtered, i.e. during a multicast, it depends on the history of a channel whether a message is sent down to it. Furthermore, a network element can 'generate' a message, i.e. send a message m' when receiving another message m (or no message), depending on its internal state. We defined this type of functionality of network elements in the abstract setting and gave restriction on the filtering and generating of messages that suffice to make a protocol collapsable.

Theorem 3 ([49]) *For a collapsable multicast protocol and a property speaking only about the sender and receivers, the following holds: $\mathcal{N}(P) \models \phi$ if and only if $\mathcal{N}^1(P) \models \phi$.*

Our aim was to obtain clear, simple conditions under which reduction is possible. This lead us to a formulation of collapsable multicast protocol that is only satisfied by a slight variant of the PGM protocol. But we felt that it is interesting to consider a variant of the PGM protocol for which collapsing is simple to prove.

We also formulated the reduction to only one instance of the protocol in the abstract setting, and obtained the following:

Theorem 4 ([49]) *Let ϕ be a property speaking only about the sender and k receivers, and let P be a protocol with a finite set of messages. Then there is an instantiation $T \in \mathcal{N}^1(P)$ such that $T \models \phi$ implies that $\mathcal{N}^1(P) \models \phi$.*

If the receiver and the network element agent can iterate sending messages, then the other direction holds as well.

6.3 Conclusion and future work

This section presents an important result about multicast protocols in general. This result has not been obtained by using an automatic tool, but by a mathematical reasoning. We have studied conditions which enables to derive properties for all kinds of instantiations of multicast protocols from the simplest one of the instantiations: the protocol reduced to one sender, one node, and one receiver. Unfortunately, those conditions are verified only for a restricted version of PGM. Nevertheless, this first study shows the way for generalizing the results we obtain previously to all sets of multicast topologies.

At the moment, the abstract formulation of collapsable multicast protocols is motivated by the PGM protocol. We plan to investigate in a more systematic manner under which reductions described here are possible for multicast protocols.

7 Parametric analysis of the RCP with TReX (year 1)

7.1 Introduction

Due to the progress done in the area of classical timed model checking, tools like KRONOS [88] and UPPAAL [73] are routinely used for the verification of industrial case studies. Now, important research effort is concentrated on the extension of this classical framework with other kind of infinite-domain variables (e.g., counters, queues) or with *parameters* (i.e., uninstantiated constants). In this way, the models verified are closer to the real implementations and it is possible to synthesize the constraints on parameters such that some properties are verified. Indeed, in the classical framework, the synthesis of constraints on parameters is done (e.g., [81]) by giving (manually) concrete values to parameters and then, extrapolate the results obtained. This manual process is very time consuming and error prone. Therefore, tool support for deriving the constraints *automatically* is very important. There are currently several tools that can do analysis of parametrized timed systems: HYTECH [57], LPMC [65], TReX [31], and a prototype extension of UPPAAL [61]. Now, these tools has to be confronted to realistic case studies in order to guide the research effort.

The IEEE 1394 root contention protocol [1] is a such case study. It is a real-time leader election protocol for two processes, used by the the physical level of the IEEE 1394 bus in order to determine the topology of the network. In this protocol, timing parameters like transmission delay and waiting intervals play an essential role. For some values of parameters, the protocol is correct and for others it fails either by electing two leaders or by infinite running. We are interested in synthesizing automatically the constraints that ensure the correctness of the protocol. This is interesting since it has been shown by experiments [81] that the values for waiting intervals given in the standard [1] do not support the increase of the transmission delay, i.e., the length of the cable.

In order to synthesize the constraints, we use the TReX [31] tool on the different models of the protocol. TReX is a tool for automatic analysis of automata-based models equipped with variables belonging to different infinite/finite domains and with parameters. These models are, at the present time, timed automata extended with counters and parameters and communicating through unbounded lossy FIFO channels and shared variables. The techniques used in TReX are based on *symbolic reachability analysis*. Symbolic representation structures are used to represent infinite sets of configurations for the variables. Forward/backward exploration procedures are used to generate a finite abstraction of the input model representing the graph of reachable symbolic configurations. The termination is not guaranteed, but efficient *extrapolation techniques* [3, 13] are used to help it. These techniques are based on computing the (exact) effect of iterating control loops detected automatically during the exploration.

For timed automata, the symbolic representation structure used by TReX is an extension of classical DBM [47], Constrained Parametric DBM [13]. They allow to deal in a uniform way with counter/clock automata, parametric/non-parametric models, and systems generating linear or nonlinear arithmetical constraints between parameters.

We begin by giving a short presentation of the TReX tool, where we explain mainly the techniques used by this particular case study. Then, Section 7.3 describes informally the IEEE 1394 root contention protocol and gives its specification. Section 7.4 presents five models considered

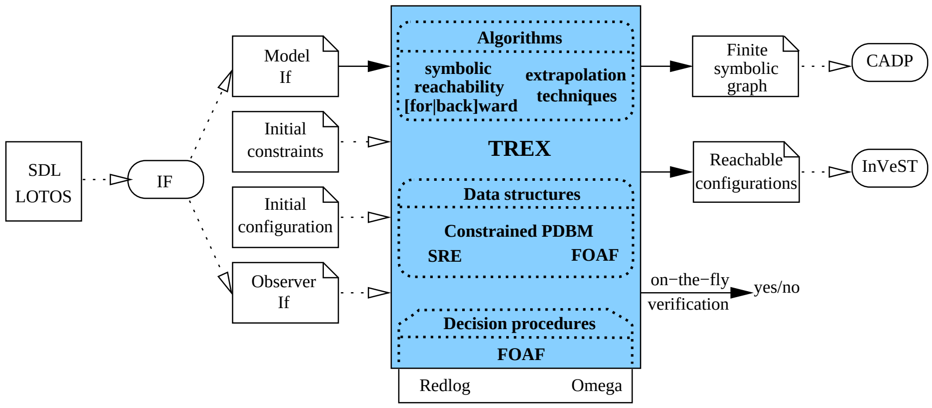


Figure 22: Overview of the TREX architecture and environment.

for analysis with TREX and reports, for each model, the experimental results. Also, we compare, when possible, our results with those reported in [61] or obtained with HYTECH. Finally, we give our concluding remarks.

7.2 Symbolic Reachability Analysis with TReX

Figure 22 shows the overall environment and architecture of TREX.

The input model of TREX is given by a textual IF [34] file. This file can be obtained by translation (using IF tools) of a graphical SDL specification, or of a textual LOTOS one. Also, the IF file can be used (after instantiation of parameters) to do finite model-checking with the CADP toolbox [50] or KRONOS. The input model is the only mandatory input of TREX. Additionally, the user can specify initial constraints on the parameters occurring in the model. These constraints play the role of an invariant: they are true in each reachable configuration explored by TREX. The reachability analysis can begin either from the initial state of the model or from a symbolic initial configuration (i.e., control state and symbolic valuation of variables) specified by the user.

TREX can check on-the-fly safety properties on the transitions of the input model. The property should be given as an *observer* written in IF. The observer is an extended automaton sharing transition labels with the input model. TREX does on-the-fly the product of the observer with the input model and synchronizes transitions having the same labels. If the synchronization is not possible, it means that the input model does not satisfy the property. Moreover, the sequence of transitions from the initial state of the input model to the state where the synchronization fails represents a diagnosis (counterexample). Indeed, the symbolic configuration where the synchronization fails gives a necessary (super-approximation) set of constraints ϕ for which the property is not satisfied. This set can be reduced by verifying the same property with an initial constraint including the negation of a subset of ϕ. The minimal set of constraints needed to satisfy the property can be obtained by iterating this procedure. We use intensively this technique for the synthesis of constraints during the verification of safety properties for the root contention protocol.

Also, TREX can generate the (finite) graph of reachable configurations. It is a finite abstraction of the analyzed model, which can be used for finite-state model checking.

The core of TREX are a forward/backward exploration algorithm and the symbolic representation structures for infinite sets of valuations for variables. The exploration algorithm is generic and can be used for any kind of symbolic representation providing a data structure, basic operations ($\cup$, $\cap$, $\subseteq$, ...), symbolic successor/predecessor operations, and an extrapolation procedure.

In the current version, TREX provides three packages for symbolic representation of configurations: SRE (Simple Regular Expressions) [5] package for lossy FIFO channels, Constrained Parametric DBMs [13] for parameterized counter and timed automata, and FOAF (First Order Arithmetical Formulas) for linear and nonlinear constraints on parameters. Since the models used for this case study are parametric timed automata, we present shortly the last two data structures above.

Parametric DBMs (PDBMs) are extension of DBMs [47], which are used (e.g., in KRONOS) in order to represent clock valuation sets (zones). A PDBM is a square matrix, which elements are pairs of the form (comparison operator, arithmetical term on parameters). In Constrained PDBMs, the PDBM is associated with a first order formula on the parameters occurring in the matrix. All operations on DBMs (emptiness, intersection, and inclusion tests) can be adapted for Constrained PDBMs [13]. These operations use a canonical form for Constrained PDBM, which is computed by a *symbolic* Floyd-Warshall algorithm. Since this algorithm is very time consuming, TREX does the hashing of all new computations. In practice, this optimization gives good results, reducing by 2 to 10 times the number of computations of the canonical form.

The FOAF package provides a compact representation (including simplifications according to the arithmetical rules) for terms and formulas used in Constrained PDBMs. It provides methods to give for each term/formula its kind (linear or non-linear). This is particularly useful to know which decision procedure can be applied for the test of satisfiability of formulas. Moreover, FOAF provides the Fourier-Motzkin procedure [48] for elimination of quantifiers over real variables. In order to test the satisfaction of formulas, TREX uses also external decision procedures like: OMEGA [72] for linear constraints on integers, and REDUCE [56] for linear and non-linear constraints on reals.

TREX has been used to analyze several nontrivial protocols in their parametric versions, such as the Bounded Retransmission Protocol (BRP) [44]. This particular example requires the full power of TREX since it is a parametric heterogeneous model involving clocks, counters, and lossy channels. Moreover, the constraints manipulated in this model are nonlinear (contain products between variables). As far as we know, TREX is the only existing tool which allows to deal fully automatically with such a complex model.

7.3 Root Contention Protocol

The IEEE's Microcomputer Standards Committee started to work in 1986 on the unification of several serial buses such as VME, MULTIBUS II, and FUTURE BUS. The goal was to provide easy to use, low cost, high speed communications. This effort led to a new serial bus protocol defined in the IEEE 1394 Standard [1], which supports transfer rates of 100, 200, and 400Mps. Although the standard has been finalized, the architecture is still being refined and adapted to provide additional data rates of 800, 1,600, and 3,200Mbps. Part of this ongoing work is reflected in the proposals IEEE 1394a [83] and 1394b.

The IEEE 1394 standard specifies a serial bus that supports peer-to-peer, isochronous and asynchronous data transfers among up to 64 devices. Also, the bus is hot-plug-and-play since its reconfiguration occurs automatically whenever a new device is added. The IEEE 1394 architecture involves *nodes* (addressable entities that run their own part of the protocol) connected by serial *cables* or backplane. On each node, the protocol consists of four layers: the physical layer, the link layer, the transaction layer, and the serial bus management layer. We restrict our attention to the protocol of the physical layer executed on nodes linked by cables.

The physical layer includes electrical signaling, mechanical interface, data transmission, *bus configuration* and arbitration. At this level, each node has several ports which may be connected by cables to other nodes. The configuration task of the physical level consists of tacking the relatively flat physical topology of the bus and turning it into a logical tree structure with a root node. Configuration consists of three phases: (1) the bus initialization, (2) the tree identification, where it is checked whether the bus topology is a tree and, if so, a root is elected, and (3) the self identification where each node selects a unique physical identifier and sends it to other nodes.

The root contention protocol occurs at the end of the second phase. In this second phase, each node tries to assign to its connected ports a parent or a child flag. Leaf nodes (i.e., nodes having only one connected port) start by outputting a `Parent_Notify` signal to their nearest neighbor node (with certain propagation delay). When a node receives a `Parent_Notify` signal on one of its ports, it marks that port as bound to a child and outputs a `Child_Notify` signal on the same port. Upon detecting this state, the leaf node marks this port as a parent port and removes the signaling, thereby confirming that the leaf node has accepted the child designation. A node with several connected ports waits until only one port remains unmarked and then it signals `Parent_Notify` on this port. The port on which it receives a `Child_Notify` is marked as parent. The node with all its connected ports marked child is chosen as the root of the tree. In the final stage of the tree identification phase, it is possible that two neighboring nodes have marked all but one port

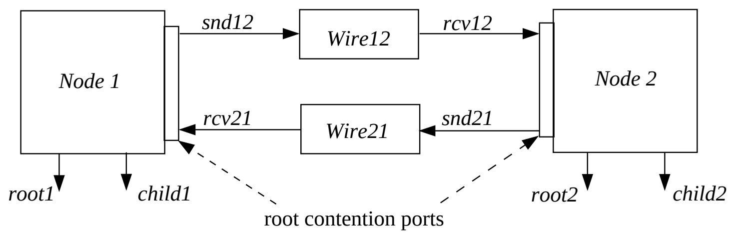


Figure 23: General architecture of the implementation models..

as being child and signal `Parent_Notify` to each other. In this case, the *root contention protocol* is initiated in order to elect one of the two nodes as the root of the tree.

If a node receives a `Parent_Notify` signal on a port, while sending a `Parent_Notify` signal on that port, it knows, locally, that it is in root contention. Then, it removes the `Parent_Notify` signal and leaves the line in the `Idle` state. At the same time, it starts a timer and picks a random bit. If the random bit is one, the node will wait for a long time (`Root_Cont_Slow`), else it will wait for a short time (`Root_Cont_Fast`).

When the waiting timer expires, the node samples its contention port. If it sees `Idle`, it starts sending `Parent_Notify` anew and looks to see a `Child_Notify` signal in return for acknowledgement. When the timer expires and the node sees `Parent_Notify` on its port, it sends a `Child_Notify` signal back as acknowledgement. Now, it knows that it has to take the root role. However, the root contention may occur again if the nodes pick the same random bits (at the same moment) or they pick different random bits within a time equal with the difference between `Root_Cont_Slow` and `Root_Cont_Fast`. In this case, both nodes detect renewed root contention and the whole process is repeated until one of them becomes root.

[1] specifies that 1590 $ns \leq$ `Root_Cont_Slow` $\leq 1670\ ns$ and 760 $ns \leq$ `Root_Cont_Fast` $\leq 850\ ns$ for a cable length of maximum 4.5 m (so the communication delay is also a bounded parameter). Since the ongoing work on the standard converges to changes of the transfer speed and cable length, it is interesting to have the relation between the parameters of the protocol such that its properties are satisfied.

The following three properties are required for this protocol:

- *Safety1*: A unique root is elected.
- *Safety2*: In a round of the protocol, a root is elected only on the request made during this round. The rounds are delimited by `Idle` signals.
- *Liveness*: The protocol terminates under the assumption that the probability to pick different bits is equal to one.

The first and third property are natural. The second property is stated in informal notes to the IEEE 1394a Working Group (see [81] for more details). It says that the the decision of a node to become root have to be made on the basis of the `Parent_Notify` signal sent after the detection of the contention, and not on an old signal. This means that the `Idle` state signaled by a node after the detection of the root contention have to arrive to the node's pair before the expiration of its waiting interval.

In the remainder of the paper, we present the models we consider as input of TREX and the results obtained for the verification of the properties above.

7.4 Models and Experimental Results

The concrete architecture of all models considered consists in two (symmetrical) automata modeling the contention nodes and two automata implementing the cable, one wire for each direction (see Figure 23).

Depending on the semantics chosen for the communication between the nodes, the models proposed for the root contention protocol fall into two classes. The first class [85] considers that the communication is done by message passing, the messages being `Parent_Notify` and `Child_Notify`.

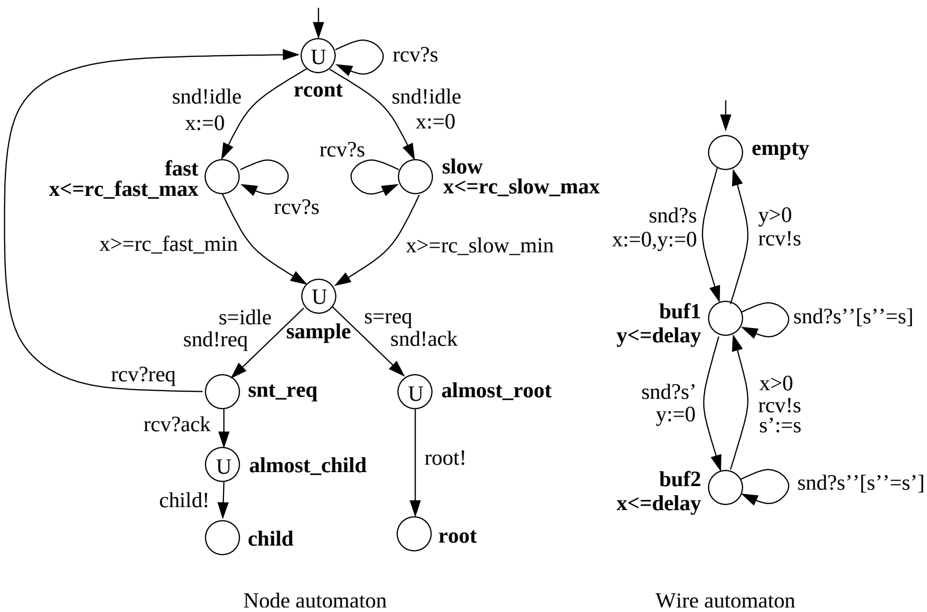


Figure 24: Generic node and wire automata for Model I1.

Also, the messages can be overwritten and lost. It is pointed out in [81] that this abstraction is inappropriate, since IEEE 1394 standard specifies that, at this phase, the communication is done by signaling states, not by transmitting packets. Indeed, the whole tree identification process occurs in a matter of microseconds. The second class of models [81] implements the communications by signals persisting at the input port of the receiving node, until the sending node changes its output port signal. So, besides the `Parent_Notify` and `Child_Notify` signals, the wire can be left undriven (`Idle` signal). We take this last model for communications.

All models are parametrized by five parameters: the maximum delay for communication and signal processing (*delay*), the minimum and maximum values for the long waiting interval (*rc_slow_min* resp. *rc_slow_max*), and the minimum and maximum values for the short waiting interval (*rc_fast_min* resp. *rc_fast_max*). The initial constraint on these parameters, ϕ_0, says that all the timers, including the communication delay cannot be 0, and that the values of the faster interval of waiting are less than the values of the slower interval:

$$\phi_0 \equiv delay > 0 \wedge 0 < rc\_fast\_min \leq rc\_fast\_max < rc\_slow\_min \leq rc\_slow\_max$$

In the remainder of this paper, we use the following shorter notations for the signals : `req` for `Parent_Notify` and `ack` for `Child_Notify`. All experiments have been performed on a Pentium III at 597MHz and 256MB of memory.

7.4.1 Model I1: synchronous detection of the initial root contention

This model is inspired by the one proposed in [81] (see Figure 24). There are four timed, input/output automata with urgent states (marked by U). We suppose that the reader is familiar with the model of timed automata. Input/output automata have transitions labelled by a tuple build from: the name of the port (e.g., `snd` and `rcv`), the operation on this port (`?` for input, `!` for output), and the signals sent (e.g., `req`, `idle` or empty). The output operation is not blocking, but the input operation is blocking until the input is available. We reduce the number of control states of the automata proposed in [81] by using enumerated variables in order to stock signals received. Also, we use guarded input operations in the generic wire automaton with the following semantics: the input is taken only if the guard (boolean expression between square brackets) is satisfied.

The generic automata of Figure 24 are substituted with the signals specific to each node. The whole system is then described by the following CCS-like expression:

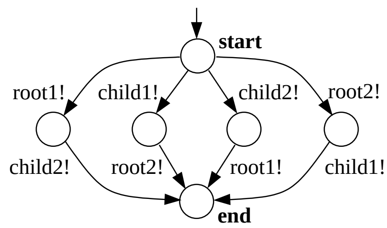


Figure 25: Observer for property *Safety1*.

$$\begin{array}{l} \quad\quad\quad\quad Node1\ [root1/\mathrm{root}, child1/\mathrm{child}, snd12/\mathrm{snd}, rcv21/\mathrm{rcv}] \\ ||\ \ Wire12\ [snd12/\mathrm{snd}, rcv12/\mathrm{rcv}]\ \ ||\ \ Wire21\ [snd21/\mathrm{snd}, rcv21/\mathrm{rcv}]\ \ || \\ \quad\quad\quad\quad Node2\ [root2/\mathrm{root}, child2/\mathrm{child}, snd21/\mathrm{snd}, rcv12/\mathrm{rcv}] \end{array}$$

The model supposes that initially, the nodes detect simultaneously the root contention. However, this is not true for the subsequent rounds. The modeling of wires as two place buffers is correct if we suppose that the wires do not get more inputs than the nodes can handles, i.e., than the receiving part can distinguish. So, until a new signal arrives, the old one continue to be driven across the wire. From experimental results, [81] shows that the modeling of wires as two place buffers is correct if $delay < rc\_fast\_min$, which is the case in IEEE 1394.

Since the input model of TReX allows only communications by shared variables and lossy FIFO channels, we modeled the input/output communications by shared variables. This gives us a model having four processes, five parameters, six clocks, and six enumerated variables. The same model in HyTech input format have four processes, five parameters, and six clocks.

Verification of Safety1: The observer used to verify the property *Safety1* is given on Figure 25. First, we apply TReX on the full parameterized version of the model, with ϕ_0 as initial constraint. We obtain that the property is not satisfied. The performances are reported in Table 13. Note that HyTech fails to give this diagnosis by forward reachability because of lack of memory. This is due to the fact that, in forward analysis, HyTech does not check the property on-the-fly, but tries to build the reachability graph. Using backward reachability, HyTech succeeds to give a diagnosis, but not to synthesize a symbolic constraint like TReX.

The symbolic configuration where synchronization fails include the following set of constraints between parameters:

$$\begin{array}{l} \phi_0 \ \wedge\ \overbrace{rc\_fast\_min < delay}^{\tilde{\phi_1}} \ \wedge\ \overbrace{rc\_fast\_min < 2 * delay}^{\tilde{\phi'_1}} \ \wedge\ \overbrace{rc\_slow\_min - rc\_fast\_max < 2 * delay}^{\tilde{\phi_2}} \ \wedge \\ rc\_slow\_min - rc\_fast\_max + 2 * rc\_fast\_min < 2 * delay\ \wedge \\ rc\_slow\_max - rc\_fast\_max < delay\ \wedge\ 2 * rc\_fast\_min < delay\ \wedge \\ rc\_slow\_min - rc\_fast\_max < delay\ \wedge\ rc\_slow\_min - rc\_fast\_max + rc\_fast\_min < delay\ \wedge \\ delay < rc\_slow\_max\ \wedge\ delay < rc\_fast\_max\ \wedge\ delay < rc\_slow\_min - rc\_fast\_min\ \wedge \\ 2 * delay < rc\_slow\_min\ \wedge\ 2 * delay < rc\_fast\_max\ \wedge\ rc\_fast\_max + rc\_fast\_min < rc\_slow\_min \end{array}$$

The constraints $\tilde{\phi_1}$, $\tilde{\phi'_1}$, and $\tilde{\phi_2}$, are interesting since their strict negation has been found by experiments in [81] to be sufficient conditions to satisfy properties *Safety1* and *Safety2*.

Then, we introduce the strict part of the negation of each constraint above in the initial constraint and we model-check the safety property. It results that only the constraint $\phi_1 \equiv delay < rc\_fast\_min$ is needed to satisfy this property. This result has been obtained only with HyTech (see Table 13). The partial graph obtained with TReX satisfies the property.

Since the parameterized extension of Uppaal [61] is not yet available, we only put in Table 13 the results reported in [61] (and obtained on a similar configuration). For this, we checked the property for some instantiations of parameters. The case we consider is where each guard or invariant for *delay* is strict, both *rc_fast_max* and *rc_slow_max* are infinite, and both *delay* and *rc_slow_min* are equal to *rc_fast_min*. We have so a model with only one parameter. The safety property is verified. The results we obtain are compared with those reported by [61] in Table 13. The difference of performances in this case is in our disfavor. This can be explained by the more

Table 13: Experimental results for Model I1 and property *Safety1*.

model	*property*	*no. param.*	*constr.*	*result*	TREX		HYTECH		*P*UPPAAL	
I1	*Safety1*	5	ϕ_0	false	3:09:53	45MB	0:27:46	out of mem.	not given	
		5	$\phi_0 \wedge \phi_1$	true	> 67h	> 192MB	0:00:06	5MB	not given	
		1	$rc\_fast\_min > 0$	true	0:03:09	15MB	0:00:01	5MB	0:00:08	11MB

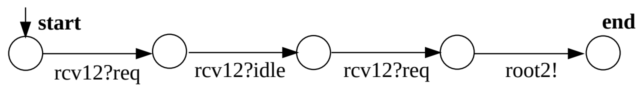


Figure 26: Observer for property *Safety2* and the Node 2.

complex model we considered (synchronizations done by shared variables) and the use of the heavy decision procedures for non-linear constraints of REDUCE.

Verification of Safety2: This property cannot be easily verified on this model, since for the initial round no request is sent. In order to verify it, we will introduce the Model I2 (see section 7.4.2).

Verification of Liveness1: This cannot be done with this model, since we cannot model the probability to choose the random bits. This property will be considered for the Model I3 (see section 7.4.3).

7.4.2 Model I2: asynchronous detection of the initial root contention

We relax the hypothesis of Model I1 concerning the initial detection of the root contention, since it cannot be satisfied in a distributed system. In fact, before detection of contention, each node sends initially a *req* signal on the (unique) port which is not yet marked as child port, and then waits for an acknowledgement or a request. Suppose that *Node1* (resp. *Node2*) sends *req* at the (absolute) time t_1 (resp. t_2). Then, $-delta \leq t_1 - t_2 \leq delta$. Indeed, if $t_1 - t_2 > delta$, *Node1* cannot send a *req* because it has already received a *req*, the delay of communications being bound by *delta*.

Using this observation, we modify the generic automaton of nodes for Model I1 as follows. The **rcont** state is no more initial, and we add a new initial state named **init**. A node may wait in the **init** state no more than *delta* (i.e., the invariant of state **init** is $\mathbf{x} \leq \mathbf{delta}$), then it sends a request signal and moves to **snt_req** state. This model is particularly useful for the verification of the property *Safety2*.

Verification of Safety2: The observer used to verify the property *Safety2* for the Node 2 is given on Figure 26. Like for the first property, we apply TREX on the full parameterized version of the model, with ϕ_0 as initial constraint. We obtain that the property is not satisfied. The symbolic configuration where synchronization fails include the following set of constraints between parameters:

$$\phi_0 \ \wedge \ \overbrace{rc\_fast\_min < delay}^{\tilde{\phi}_1} \ \wedge \ delay < rc\_fast\_max$$

This corresponds to the constraint synthesized for the property *Safety1*. If we add to ϕ_0 the strict negation of the constraint $\tilde{\phi}_1$, ϕ_1, we obtain that the property is false, and the following constraint is synthesized:

$$\phi_0 \ \wedge \ \overbrace{delay < rc\_fast\_min}^{\phi_1} \ \wedge \ \overbrace{rc\_fast\_min < 2 * delay}^{\tilde{\phi}'_1} \ \wedge \ 2 * delay < rc\_fast\_max$$

Table 14: Experimental results for Model I2 and property Safety2.

model	*property*	*no. param.*	*constr.*	*result*	TREX		HYTECH	
I2	*Safety2*	5	ϕ_0	false	0:00:50	14MB	0:00:50	29MB
		5	$\phi_0 \wedge \phi_1$	false	0:05:57	9MB	0:00:08	5MB
		5	$\phi_0 \wedge \phi_1'$	false	0:01:41	17MB	0:00:08	8MB
		5	$\phi_0 \wedge \phi_1' \wedge \phi_2$	false	0:03:13	20MB	0:00:07	4MB

The strict negation of $\tilde{\phi}_1'$ has been found manually in [81] to be a sufficient condition to satisfy property *Safety2*. Then, we introduce the strict part of the negation of this constraint in the initial constraint and we model-check the safety property.

It follows that the property is not satisfied, and we obtain the constraint $\tilde{\phi}_2 \equiv rc\_slow\_min - rc\_fast\_max < 2 * delay$ and a diagnosis showing that the observer fails because the renewal of the root contention. This is a liveness problem, and we show for Model I3 that $\phi_2 \equiv 2 * delay < rc\_slow\_min - rc\_fast\_max$ is the condition to solve in one pass the root contention if different bits are picked.

The experimental results are in Table 14. For PUPPAAL, no results are reported in [61], for this particular case.

7.4.3 Model I3: centralized, boolean fairness for Model I1

The models above do not include the hypothesis that the probability to pick different bits is equal to one. Since the input model of TREX does not include probabilistic aspects, we simplify this hypothesis by adding to the Model I1 a centralized generator of bits which always gives different values for any two successive calls. Therefore, it is essential that root contention is resolved within the same round if both nodes pick different random bits. Since the probability to pick different random bits is strictly greater than zero in each pass, the nodes will eventually pick different bits and thus elect a root, with probability one.

We add to the Model I1 a boolean variable shared by the two nodes, which implement the fair generator. Also, we add labels `retry1!` resp. `retry2!` to the transitions from the state `snt_req` to the state `rcont` in order to observe the renewal of the protocol. We use the same observer as for the property *Safety1*, but the set of observed labels is now updated with the `retry1!` and `retry2!`. This means that the observer does not accept synchronization with any retry transition.

By applying TREX on the full parameterized version of the model, with ϕ_0 as initial constraint, we obtain that the property is not satisfied. The set of constraints synthesized is:

$$\phi_0 \ \wedge \ \overbrace{rc\_fast\_min < delay}^{\tilde{\phi}_1} \ \wedge \ \overbrace{rc\_fast\_min < 2 * delay}^{\tilde{\phi}_1'} \ \wedge$$
$$rc\_slow\_max < 2 * delay \ \wedge \ rc\_slow\_max - rc\_fast\_min < delay \ \wedge \ delay < rc\_fast\_max$$

The constraints synthesized are those needed for the satisfaction of the safety properties. So, we repeat the experiment by tacking $\phi_0 \wedge \phi_1$ resp. $\phi_0 \wedge \phi_1'$ as initial constraints. In both cases, we obtain that the property is not satisfied. The following constraint appears in the fail state of both counterexamples:

$$\tilde{\phi}_2 \equiv rc\_slow\_min - rc\_fast\_max < 2 * delay$$

By tacking now the strict negation of this constraint, we obtain that the property is satisfied. This has been also proved by experiments in [81]. The performances are given in Table 15.

7.4.4 Model A1: abstraction of communications

The timed, input/output automaton A1 of Figure 27 taken from [85, 81] abstracts the communications between the nodes and records the status of each of the two nodes. They prove that if the parameters satisfy the additional constraint:

$$\overbrace{2 * delay < rc\_fast\_min}^{\phi_1'} \quad \wedge \quad \overbrace{2 * delay < rc\_slow\_min - rc\_fast\_max}^{\phi_2}$$

Table 15: Experimental results for Model I3.

model	*property*	*no. param.*	*constr.*	*result*	TREX		HYTECH	
I3	*Liveness*	5	ϕ_0	false	1:05:58	28MB	0:00:04	3MB
		5	$\phi_0 \wedge \phi_1$	false	0:02:54	9MB	0:00:03	2MB
		5	$\phi_0 \wedge \phi_1'$	false	0:02:51	9MB	0:00:03	2MB
		5	$\phi_0 \wedge \phi_1' \wedge \phi_2$	true	0:02:09	8MB	0:00:01	4MB

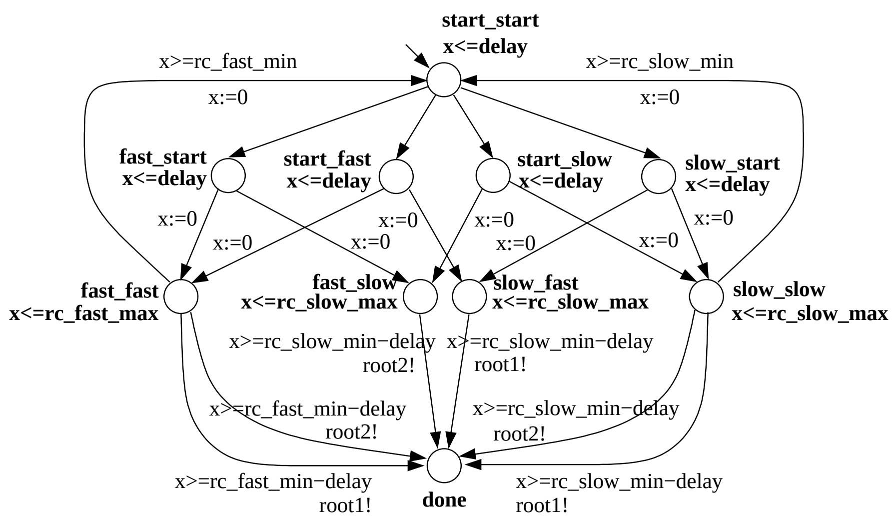


Figure 27: Model A1: abstraction of communications.

this abstraction is correct, in the sense that all execution paths of I1 are included in A1. For this model, we compute the symbolic graph using TREX and HYTECH. The two symbolic graphs obtained are equivalent. The performances are given in Table 16. Then, we consider A1 as being an observer for the Model I1 in the presence of the initial constraint $\phi_0 \wedge \phi_1' \wedge \phi_2$. This property is verified, so we are able to verify mechanically the results given in [85, 81].

7.5 Conclusion

This paper reports about the fully automatic verification of the IEEE 1394 root contention protocol using TREX and HYTECH tools. This is an industrial protocol in which timing parameters play an essential role.

We consider the models proposed in [81] and improve them in order to verify easily the safety and liveness properties of the protocol. This verification allows us to *synthesize automatically* the constraints derived by experimental results in [81]. Although in some cases the performances of TREX are not so good as those obtained with the prototype extension of UPPAAL [61], we are able to deal with more complex properties and to directly synthesize the constraints. Compared with HYTECH, TREX is slower, but it provides symbolic diagnosis while a safety property is checked. This is very useful for constraint synthesis.

The relatively bad timing performances obtained motivate us to connect TREX to a tool for

Table 16: Experimental results for Model A1.

model	*property*	*no. param.*	*constr.*	*result*	TREX		HYTECH	
A1	true	5	ϕ_0	false	0:00:02	0.1MB	0:00:01	0.1MB
I1	A1	5	$\phi_0 \wedge \phi_1' \wedge \phi_2$	false	7:24:30	190MB	0:31:17	out of mem.

linear constraints solving and to improve its input model with communication by synchronization on actions.

8 Parametric analysis of the RCP with LASH (year 3)

8.1 Introduction

During the third year of the project, the Lige group has produced results that extend the symbolic state-space exploration techniques previously developed for analyzing infinite-state systems to a specific class of models with heterogeneous data types. More precisely, the *hybrid acceleration* method that has been developed tackles the analysis of systems modeled by finite-state machines extended with unbounded integer variables and unbounded real clocks. It is thus well suited for checking properties of communication protocols in which integer variables model e.g., message sequence numbers, and real clocks are used for describing the timed aspects. Unlike other approaches, hybrid acceleration is not limited to computing reachable sets that can be expressed as finite unions of convex sets, or to analyzing models with a bounded state-space exploration depth.

Like in the previous section, such a reachability analysis will produce constraints on the parameters enabling to reach given states representing given properties of the system. This is what we call here parameter synthesis.

There are currently two ways of performing hybrid acceleration, both aimed at computing the reflexive and transitive closure of a part of the transition relation of the model under study. The first method [22] relies on studying the arithmetical properties of the transformation labeling a control cycle. If this transformation satisfies some conditions, then it is shown that its closure can be characterized mathematically in a way that leads to a simple computation algorithm. The second method [26] consists in representing the transition relation that is to be accelerated as a finite-state transducer, and of observing regularities that appear when this transducer is repeatedly applied to the representation of a set of initial states. In some situations, such regularities can be captured in a finite-state structure that can be proven equal to the transitive and reflexive closure of the transducer.

Due to the recent nature of those results, their implementations are still prototypal and are not yet part of the regular release of the LASH tool developed in Lige [64]. Our prototypes are limited to handling small models (i.e., with few control locations and few variables), lack some optimizations in their core algorithms, and have a limited front end. The experiments described in these notes are aimed at assessing the potential of hybrid acceleration to check realistic protocol models, and at better pinpointing the parts of the implementations that would benefit the most from theoretical improvements and practical optimizations. Since the models of the PGM protocol developed in the context of the project are too complex for being dealt with by our prototypes, we have opted for studying a simple model of the IEEE 1394 Root Contention Protocol developed by the Paris group.

The goals of the experiments that have been carried out included

- Computing an exact representation of the reachable state space of a model of RCP in which time is neither abstracted away, nor discretized. Such a computation makes it possible to check properties such as the reachability of selected control locations, the absence of deadlocks, as well as all the safety properties that are finite-state representable (in the present setting, this includes all the properties that can be expressed in first-order arithmetic without multiplication [25]).

- Computing a symbolic representation of the set of parameter values for which some safety properties are validated. This can be done by turning some parameters into variables with a value that is nondeterministically initialized with its full range of possible values, and not modified during the runs of the model.

- Evaluating the adequacy of the proposed acceleration techniques to the constructs that occur in real-life models, and suggesting directions for improvements.

8.2 Accelerating Hybrid Transformations

Hybrid automata [7] are natural models for systems with both discrete and continuous features. A hybrid automaton takes the form of a finite-state graph (modeling the control part of the system) extended with a fixed number of unbounded real variables (modeling the data part). A configuration of a hybrid automaton is thus composed of a control location and a value for each variable. The configuration of a model can change in two ways:

- By staying at the current control location for some amount of time (*"time-step"*). The value of the variables then evolves according to *evolution laws*, provided that some *invariant condition* stays satisfied. Each control location is associated with an evolution law and an invariant condition;
- By following an edge in the control graph from the current control location (*"transition-step"*). The value of the variables is then modified according to a *transformation* labeling the edge. A *guard* can also restrict the possibility of following such an edge.

For the present work, we considered *linear hybrid automata* [7], for which evolution laws, invariant conditions, transformations and guards are restricted to be expressed in linear arithmetic. In traditional approaches, the reachability of such models is studied by a symbolic state-space exploration procedure, starting from a symbolic representation of the set of reachable configurations, and performing repeatedly time-step and transition-step operations until a fixpoint is reached.

Unfortunately, there are systems for which the set of reachable configurations has an unbounded depth, and thus cannot be computed in finite time using this approach. For analyzing such systems, one has either to introduce approximations, or to use *acceleration techniques* aimed at computing at once the effect of the transitive and reflexive closure of the transition relation (or a part of it). We have developed two strategies for applying acceleration to systems modeled by hybrid automata.

8.2.1 Arithmetic-based Approach

This approach is focused on the acceleration of control loops, i.e., given a cycle in the control graph, one characterizes the data transformation undergone by the variables while following this cycle, and attempts to compute in one step its repeated effect. If successful, the computed closure of the transformation is added to the transition relation of the system, which speeds up its state-space exploration. During this exploration, the sets of reachable configurations that are manipulated are represented by *Real Vector Automata* [20], which are finite-state machines recognizing real vectors encoded as infinite words. RVA are expressive enough to represent all the sets that can be defined in first-order arithmetic without multiplication [25].

In order to be able to compute the repeated effect of loops, one needs

- A mathematical characterization of the data transformations labeling arbitrary control paths of linear hybrid automata (taking into account time-steps as well as transition-steps),
- A decision procedure for checking whether the closure of a transformation preserves the representable nature of sets, and
- An image computation algorithm for applying the closure of a transformation to represented sets.

Solutions to these problems are provided in [22], in which data transformations are described by *Linear Hybrid Transformations*, and a sufficient *periodicity criterion* is developed for computing the closure of such transformations.

8.2.2 Transducer-based Approach

This method attempts to accelerate control structures that are more complex than loops. As with the arithmetic-based approach, the configurations of the model are encoded as infinite words, and sets of such configurations are represented by RVA. With a suitable encoding scheme, it can be shown that the relation transition of a linear hybrid automaton can be represented by a finite-state input/output RVA, i.e., an infinite-word transducer.

In the transducer-based approach [26], one starts from a set of initial configurations represented by an RVA, and apply repeatedly to this RVA the transformation represented by a given transducer. After each step, the computed RVA is determinized and reduced to a canonical form. On thus obtains a sequence of RVA, among which specific equivalence relations are constructed in order to detect similar structures between successive automata. In some situations, these similarities can be extrapolated into a finite-state structure, which is "guessed" to be equal to the closure of the studied transformations. Procedures have been developed for then checking that the guess is exact, i.e., corresponds precisely to the expected result.

8.3 The Root Contention Protocol

The Root Contention Protocol (RCP) considered in these notes is a subpart of the IEEE 1394 standard, which defines a communication protocol suited for a high-speed serial bus connecting computer peripherals. The RCP is a leader election protocol, designed for arbitrating conflicts between bus nodes during the identification and organization of the elements connected to the bus [80]

The behavior of RCP relies on timers. When contention occurs between two nodes, each of them chooses a random bit of information, which determines the time it will wait before trying again to become the leader. Several parameters characterize the time constants inherent to the protocol : the physical transmit time *delay*, the minimum and maximum waiting delays *minfast*, *maxfast* for the choice "0", and the minimum and maximum waiting delays *minslow*, *maxslow* for the choice "1". The protocol specifications restrict these parameters to satisfy the following constraints [80] :

$$\begin{aligned} delay &> 0 \\ minfast &> 0 \\ maxfast &\geq minfast \\ minslow &> maxfast \\ maxslow &\geq minslow \\ 2delay &\leq minfast \\ 2delay &\leq minslow - maxfast \end{aligned}$$

For our experiments, we started from the intermediate model of RCP given in [80], for which a IF model had been developed by the Paris team. A linear hybrid automaton depicting this model is given in Figure 28. In order to reduce the size of the control graph, this automaton was reduced by using a symmetry relation on pairs of similar states, and by merging together states with the same evolution laws and invariants linked by guardless and transformationless edges.

8.4 Experimental Results

8.4.1 Parameter Synthesis without Acceleration

The first experiment consisted of computing an exact representation of the set of reachable configurations of the model at each control location, starting from an initial set containing all permitted values of the parameters. (In other words, this procedure can be seen as synthesizing for each control location the set of values of the parameters that make this location reachable). It is worth mentioning that the computation does not depend on the property that is to be verified on the model. Actually, once computed, the representation of the reachable set makes it possible to decide easily and efficiently any safety property of the protocol that can be expressed in the first-order additive theory of integer and real variables.

Using our simplified model, we were able to compute a symbolic representation of its set of reachable configurations. The RVA representing the set of reachable variable values in the location *start-start* has 2872 states. A fixed point was reached in 1564 seconds without using acceleration. The total amount of memory used was 76 MB.

It is interesting to compare these results with those of other experiments in which the timed aspects of RCP were modeled by using discrete counters to track the progress of time. With this

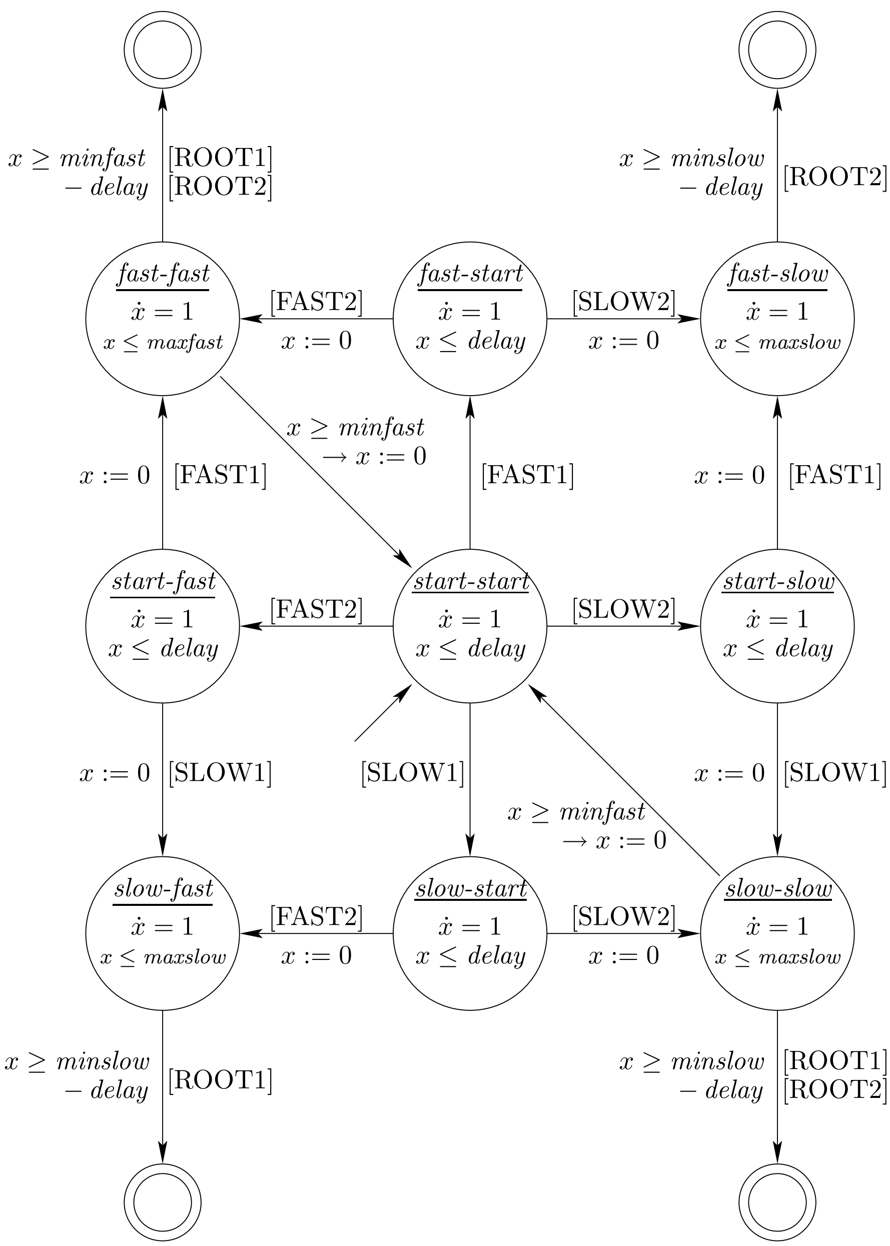


Figure 28: Intermediate model of RCP.

approximation, using finite-word automata to represent sets of configurations, a representation of the reachable set of states was computed in 1087 s, using 15 MB of memory. This shows that using infinite-word rather than finite-word automata for representing the sets that have to be manipulated is a viable approach, the slight additional costs of which are not significant enough to outweight the advantage of avoiding unnecessary approximations.

8.4.2 Parameter Synthesis with Acceleration

Our second experiment was aimed at exploiting the acceleration techniques developed for hybrid automata. We modified our model by adding an extra variable t keeping track of global time, i.e., initialized at 0, evolving at rate 1 in all locations, and not being modified by transitions. We then attempted to accelerate the simple control cycles of the model, so as to obtain an exact representation of its set of reachable configurations.

Unfortunately, it turned out that the hybrid transformations labeling the simple cycles of the model did not satisfy the criterion for being acceleratable. This can be explained as follows. Consider the loop that starts from location *start-start*, and visits *start-fast* and *fast-fast* before returning to *start-start*. Following this loop once, the current value of t (the global time) is incremented by an amount that ranges from the value of *minfast* to *delay*+*maxfast*. Thus, following the loop k times, with $k > 0$, turns the value of t into the interval $[k.\mathit{minfast}, k.(\mathit{delay} + \mathit{maxfast})]$. Since the relation between the current and future values of t and the iteration number k cannot be expressed in additive arithmetic (it requires to multiply the value of parameters by k), this relation cannot be represented by an RVA, and acceleration is impossible.

In order to alleviate this problem, which restricts the possibility of performing parameter synthesis, we had to manually set the values of the parameters in our model. Using the values $\mathit{delay} = 1$, $\mathit{minslow} = 6$, $\mathit{maxslow} = 7$, $\mathit{minfast} = 2$, $\mathit{maxfast} = 3$, we were able to compute an exact representation of the reachable state space in 2.430 s, using 0.727 MB of memory. The RVA representing the set of reachable data values at location *start-start* has 41 states.

Finally, it was attempted to apply transducer-based acceleration to our last model. Due to the limitations of our current prototype implementation, we were not able to achieve this goal with a reasonable amount of resources.

8.5 Conclusions

The reachability analysis of hybrid systems using automata-based representations of sets is an interesting technique that makes it possible to combine a dense representation of time with discrete features. Using this approach in practice incurs a slight significant cost, which could probably be further lowered by optimizing and fine-tuning our current implementation. Performing acceleration together with parameter synthesis is still problematic, especially when one needs to iterate transformations in which the value of a parameter is added to a variable. Future work will be to try to generalize all the theoretical results and practical realizations developed for systems with unbounded integer variables to models that also include real clocks.

9 Perspectives on the verification of real software systems

A criticism which is often made to the formal methods of software analysis similar to those studied during the Advance project is that they are not able to cope with the complexity of real software systems, and can only be applied to simplified models of software. This criticism can certainly be made to the works of the Advance project.

A typical answer is that such limitations are inherent to the expressivity of software, and also that analysing simplified models can be very useful for catching in the earlier stages of development the design errors which are very expensive to correct later. Those answers are correct, but it would nonetheless be very gratifying to be able to apply techniques similar to those of the Advance project to real software. One conclusion that can be made after the three years of the Advance project is that the progress made during the last years implies that it is reasonable to think that the next step, for years to come, of the studies in formal methods will consist in trying to apply those techniques to source code systems.

In the last months of the Advance project, the France Telecom R & D team has pursued some research and experiments in this line of work. This research was done in the framework of the PhD thesis of Laurent Gottely who should be submitted in the first semester of 2004. This PhD thesis work was not initially considered as a part of the Advance project, but we estimate that the obtained results are sufficiently significant and connected to the overall goal and results of the Advance project for being presented in this framework. In the exploitation and dissemination to come of the results of those works, we will acknowledge the support of the European Union's IST program.

The problem we have examined is the security of Java libraries, in particular the way that Java libraries use a security manager for ensuring that critical methods are executed in a correct way.

The security manager is a device of the Java language which implements security policies with respect to the use of critical methods. Before using a critical method, for example the opening of a socket connection, if a security manager is activated, a check must be made that the use of the critical method is done in a correct way. This check consists in the call of a check method of the security manager, dedicated to the control of the critical method.

Our goal was to design a tool which will meet the following three goals:

- The first goal is that the tool should check that the security manager is well used, that is for example that all calls to a critical method are preceded by a call to the check method, and that if the result of the check method is negative, the critical method is not called.
- The second goal is that the tool should work upon libraries, that is it should be able to prove that the security manager is well executed in a given library, whatever context in which this library is called.
- The third goal is that the tool should work upon Java bytecode, in order to ensure that the complete Java language is treated. Also, this makes the tool independent of the future evolutions of the Java language and ensures that some simplifications of Java features are already done prior to the analysis.

For ensuring those goals, we had to define absractions that enable to consider Java objects and every possible context of library method calls. First, k-limitation is used to take care of the recursive data structures. That is, when recursive data structures, like chained lists for example, grow beyond a given parameter k, they are considered equal. Secondly, due to the open-world nature of the analysis, the references are abstracted based on the knowledge of their allocation site. It allows to maintain a relatively precise abstract equality even between references supplied by the environement. Those abstractions are automatic, and enable us to produce a finite model of the potentially infinite behaviour of a given Java library.

Then, the security properties that we want to check are expressed in term of automata relating the sequencing of the executions of bytecode instructions. This enables to translate our problem to a classical model checking problem, like the one described in section 3. For solving this problem, we have used tools which have been improved in the Advance project, around the manipulation of transition systems.

After the development of a prototype, we have been able to check some properties of real libraries, in particular several libraries from the official Java distribution by Sun Microsystems. We have discovered in one of those libraries a case where a critical method can be called whereas the corresponding check method gives a negative result, because of the interplay with the exception mechanism. This point is not necessarily a security breach, but could be indicative of one.

We plan to pursue this line of research for the years to come.

10 Conclusion

At the end of this report, we can draw some conclusions of the case studies that were done, mainly around the PGM protocol, in the framework of the Advance project.

The first conclusion we can draw is that the new version of the IF language which has been defined in the Advance project enables to model accurately complex telecommunication protocols. The PGM example has been completely modelled, from an initial SDL version. This new model has some advantages upon the previous version, mainly that it is easy to describe a parametrized architecture whereas it was more complicated in SDL.

The main interest that comes from the existence of IF is that a whole set of tools are being developed, which will take as entry points specifications described in the IF language. To have a common language implies that it is not necessary to have several developments for using different tools, using languages specific to each tool.

The first tools we used are well-known verification tools based on an exhaustive exploration of all possible behaviours of the specification under study. This study enabled us to establish some facts about the PGM protocol: the `SPM` messages ensure that the lost data would be detected; the presence of network elements have an influence about the sequencing of `NCF` messages; and, we tested on a few examples a relationship guaranteeing that data losses are recovered by the protocol.

But those studies are limited to finite-state versions of the PGM protocol. The breakthrough of the Advance project is to define tools that can take into account the infiniteness of PGM. We have shown how LASH enable to verify that the window policy is respected whatever is the number of data packets. A mathematical reasoning has defined under which conditions the properties proved on a simple architecture are also true for all possible tree-like architectures. TReX was used for deriving automatically relations between the parameters of PGM that guarantee that losses are necessarily recovered. Similar results have been obtained upon a hybrid model (mixing counter and real-time clocks) of the RCP protocol, first with the TReXtool in the first year of the project, and also, with the LASH tool in the third year. It is important to note that those studies bring innovative results which were not reachable a few years ago.

However, it has been necessary to rewrite parts of the PGM to study it with the tools. Because of the complexity implied by the study of infinite behaviours, tools have treated simplified versions of PGM. Those versions are quite accurate and involve the main features of the more complete ones, but the necessity to produce them could have discouraged users less acquainted with the tools. Nevetheless, it is clear for us (France Telecom R & D) that the Advance project has made impressive achievements, which is are visible in particular in the work on case studies. For instance, the parameterized analysis (synthesis of constraints guaranteeing that given properties are satisfied) of protocols such as the PGM or the IEEE RCP show the power and the relevance of the techniques and tools developed in ADVANCE. Indeed, these results cannot be achieved by other existing tools. More generally, the Advance tools are able to check properties of infinite state systems. This makes possible, in the years to come, to envision that those kind of tools will be used for checking properties of real source code, which will have a tremendous impact on the diffusion of formal methods.

References

[1] *IEEE Standard for a High Performance Serial Bus*. Number Std 1394-1995. IEEE, August 1996.

[2] M. Abadi and L. Lamport. An old-fashioned recipe for real time. In *Proceedings of REX Workshop, Real-Time: Theory in Practice*, volume 600, pages 1–27, 1993.

[3] P. Abdulla, A. Annichini, and A. Bouajjani. Symbolic verification of lossy channel systems: Application to the bounded retransmission protocol. In *Proceedings of 5th TACAS*, volume 1579, 1999.

[4] P. Abdulla, A. Bouajjani, B. Jonsson, and M. Nilsson. Handling global conditions in parametrized system verification. In *Proceedings of 11th CAV*, volume 1633, 1999.

[5] P.A. Abdulla, A. Bouajjani, and B. Jonsson. On-the-fly analysis of systems with unbounded, lossy, FIFO channels. In *Proceedings of the 10th CAV*, volume 1427, pages 305–317, 1998.

[6] Advance project home page. On the WWW at `liafa.jussieu.fr/haberm/ADVANCE/main.html`.

[7] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 6 February 1995.

[8] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[9] R. Alur and T.A. Henzinger. Finitary fairness. In *Proceedings of the 9th Annual Symposium on Logic in Computer Science*, pages 52–61. IEEE Computer Society Press, 1994.

[10] R. Alur and T.A. Henzinger. Local liveness for compositional modeling of fair reactive systems. In P. Wolper, editor, *Proceedings of the 7th CAV*, volume 939, pages 166–179, 1995.

[11] R. Alur, T.A. Henzinger, and M.Y. Vardi. Parametric real-time reasoning. In *ACM Symposium on Theory of Computing*, pages 592–601, 1993.

[12] G.R. Andrews. *Concurrent Programming: Principles and Practice*. Benjamin/Cummings Publishing Company, 1991.

[13] A. Annichini, E. Asarin, and A. Bouajjani. Symbolic techniques for parametric reasoning about counter and clock systems. In E.A. Emerson and A.P. Sistla, editors, *Proceedings of the 12th CAV*, volume 1855, pages 419–434, July 2000.

[14] Ahmed Bouajjani Aurore Annichini and Mihaela Sighireanu. Trex: a tool for reachability analysis of complex systems. In *Proceedings of CAV'2001*, volume 2102 of *LNCS*. Springer-Verlag.

[15] Gérard Babonneau. Pgm reliable transport protocol specification. Technical Report BD/CNET/DSM/RDC//98/NLM, CNET, 1999.

[16] G. Bandini, R. Lutje Spelberg, R. de Rooij, and W. Toetenel. Parametric verification of the ieee-1394a root contention protocol using lpmc. Submitted for publication. http://tvs.twi.tudelft.nl/, July 2000.

[17] K. Baukus, Y. Lakhnech, and K. Stahl. Verifying universal properties of parameterized networks. In *Proceedings of FTRTFT*, 2000.

[18] S. Bensalem, Y. Lakhnech, and S. Owre. InVeSt: A tool for the verification of invariants. In *Proceedings of the 10th CAV*, volume 1427, 1998.

[19] B. Boigelot. *Symbolic Methods for Exploring Infinite State Spaces*. PhD thesis, University of Liège, 1998.

[20] B. Boigelot, L. Bronne, and S. Rassart. An improved reachability analysis method for strongly linear hybrid systems. In *Proceedings of the 9th International Conference on Computer-Aided Verification*, number 1254 in Lecture Notes in Computer Science, pages 167–177, Haifa, Israel, June 1997. Springer-Verlag.

[21] B. Boigelot, P. Godefroid, B. Willems, and P. Wolper. The power of qdds. In *Proceedings of SAS'97*, volume 1302, 1997.

[22] B. Boigelot, F. Herbreteau, and S. Jodogne. Hybrid acceleration using real vector automata. In *Proceedings 15th International Conference on Computer Aided Verification (CAV)*, volume 2725 of *Lecture Notes in Computer Science*, pages 193–205, Boulder, USA, 2003. Springer-Verlag.

[23] B. Boigelot, S. Rassart, and P. Wolper. On the expressiveness of real and integer arithmetic automata. In *Proceedings of ICALP'98*, 1998.

[24] B. Boigelot and P. Wolper. Symbolic verification with periodic sets. In *Proceedings of the 6th CAV*, volume 818, 1994.

[25] Bernard Boigelot, Sébastien Jodogne, and Pierre Wolper. On the use of weak automata for deciding linear arithmetic with integer and real variables. In *Proc. International Joint Conference on Automated Reasoning (IJCAR)*, volume 2083 of *Lecture Notes in Computer Science*, pages 611–625, Siena, Italy, June 2001. Springer-Verlag.

[26] Bernard Boigelot, Axel Legay, and Pierre Wolper. Omega-regular model checking. In *Proc. TACAS*, 2004. To appear.

[27] A. Bouajjani, A. Collomb-Annichini, Y. Lackneck, and M. Sighireanu. Analysing fair parametric extended automata analysis. In Patrick Cousot, editor, *Proceedings of Static Analysis Symposium*, volume 2126, pages 335–355, July 2001.

[28] A. Bouajjani and P. Habermehl. Symbolic reachability analysis of fifo-channel systems with nonregular sets of configurations. In *ICALP'97*, volume 1256, 1997.

[29] A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular model checking. In E.A. Emerson and A.P. Sistla, editors, *Proceedings of the 12th CAV*, volume 1855, pages 403–418, July 2000.

[30] A. Bouajjani, S. Tripakis, and S. Yovine. On-the-fly symbolic model checking for real-time systems. In *Proceedings of IEEE RTSS'97 (San Francisco, USA)*, December 1997.

[31] Ahmed Bouajjani, Aurore Collomb-Annichini, and Mihaela Sighireanu. Trex: A tool for reachability analysis of complex systems. In G. Berry, H. Common, and A. Finkel, editors, *Proceedings of Computer Aided Verification*, volume 2102, pages 368–372, June 2001.

[32] M. Boyer. On modeling and verifying the pgm protocol. Technical report, LIAFA, 2002.

[33] Marc Boyer and Mihaela Sighireanu. Synthesis and verification of constraints in the pgm protocol. In Stefania Gnesi, editor, *Proceedings of International FME Symposium*, Lecture Notes in Computer Science, pages 264–281. Springer-Verlag, 2003.

[34] M. Bozga, J.-C. Fernandez, L. Girvu, S. Graf, J.-P. Krimm, and L. Mounier. If: A validation environment for times asynchronous systems. In E.A. Emerson and A.P. Sistla, editors, *Proceedings of the 12th CAV*, volume 1855, pages 543–547, July 2000.

[35] M. Bozga, J.-Cl. Fernandez, S. Ghirvu, S. Graf, J.-P. Krimm, and L. Mounier. If: a validation environment for timed asynchronous systems. In *Proceedings of CAV'2000*, LNCS. Springer-Verlag.

[36] T. Bultan. *Automated symbolic analysis of reactive systems*. PhD thesis, University of Maryland, 1998.

[37] Cadp home page. On the WWW at `www.inrialpes.fr/vasy/cadp`.

[38] A Collomb-Annichini and M. Sighireanu. Parameterized Reachability Analysis of the IEEE 1394 Root Contention Protocol using TReX. In *Proc. Workshop on Real-Time Tools (RT-TOOLS'01)*, Aalborg, August 2001. Aalborg University.

[39] M. Colon and H. Sipma. Synthesis of linear ranking functions. In *Proceedings of TACAS'01*, 2001.

[40] P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In *Proceedings of the Second International Symposium on Programming*, pages 106–130. Dunod, Paris, France, 1976.

[41] P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Conference Record of the Sixth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 269–282, San Antonio, Texas, 1979. ACM Press, New York, NY.

[42] P. Cousot and N. Halbwacs. Automatic discovery of linear restraints among variables of a program. In *Conference Record of the Fifth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 84–97, Tucson, Arizona, 1978. ACM Press, New York, NY.

[43] D. Dams, R. Gerth, and O. Grumberg. A heuristic for the automatic generation of ranking functions. In *Proceedings of WAVe*, pages 1–8, 2000.

[44] P.R. D'Argenio, J.-P. Katoen, T.C. Ruys, and J. Tretmans. The bounded retransmission protocol must be on time! In *Proceedings of 3rd Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 1217, pages 416–432, 1997.

[45] G. Delzanno and J.F. Raskin L. VanBegin. Csts: Compact data structures for parameterized verification. *Software Tools for Technology Transfer*, 2001.

[46] M. Devilliers, W. Griffioen, J. Romijn, and F. Vaandrager. Verification of a leader election protocol — formal methods applied to IEEE 1394. Technical Report CSI-R9728, Computing Science Institute, University of Nijmegen, December 1997.

[47] D. Dill. Timing assumptions and verification of finite-state concurrent systems. In J. Sifakis, editor, *Proceedings of the 1st CAV*, volume 407, pages 197–212, 1989.

[48] B.C. Eaves and U.G. Rothblum. Dines-Fourier-Motzkin quantifier elimination and an application of corresponding transfer principles over ordered fields. *Mathematical Programming*, 53:307–321, 1992.

[49] Javier Esparza and Monika Maidl. Simple representative instantiations for multicast protocols. In H. Garavel and J. Hatcliff, editors, *Proceedings of TACAS'03*, LNCS. Springer-Verlag.

[50] J.-C. Fernandez, H. Garavel, A. Kerbrat, R. Mateescu, L. Mounier, and M. Sighireanu. Cadp (cæsar/aldebaran development package): A protocol validation and verification toolbox. In R. Alur and T.A. Henzinger, editors, *Proceedings of the 8th CAV*, volume 1102, pages 437–440, August 1996.

[51] Jean-Claude Fernandez and Laurent Mounier. Verifying bisimulation on the fly. Technical report, LGI-IMAG, 1990. Distributed with the CADP package.

[52] R.W. Floyd. Assigning meanings to programs. In *In. Proc. Symp. on Appl. Math. 19*, pages 19–32. American Mathematical Society, 1967.

[53] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns — Elements of Reusable Object-Oriented Software*. Professional Computing Series. Addison-Wesley, 1994.

[54] P. Ganty. Algorithmes et structures de données efficaces pour la manipulation de contraintes sur les intervalles. Master's thesis, Université Libre de Bruxelles, 2002.

[55] W. Harvey and P.J. Stuckey. A unit two variable per inequality integer constraint solver for constraint logic programming. In *Proceedings of ACSC'97*, 1997.

[56] A.C. Hearn. *REDUCE — User's and Contributed Packages Manual*. Codemist Ltd., February 1999. version 3.7.

[57] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. Hytech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1(1):110–122, 1997.

[58] T.A. Henzinger, O. Kupferman, and S.K. Rajamani. Fair simulation. In A. Mazurkiewicz and J. Winkowski, editors, *Proceedings of the Conference on Concurrency Theory*, volume 1243, pages 273–287, 1997.

[59] T.A. Henzinger and S.K. Rajamani. Fair bisimulation. In S. Graf and M. Schwartzbach, editors, *Proceedings of the 6th Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 1785, pages 299–314, 2000.

[60] G.J. Holzmann. *Design and Validation of Computer Protocols*. Software Series. Prentice Hall, 1991.

[61] T. Hune, J. Romijn, M. Stoelinga, and F. Vaandrager. Linear parametric model checking of timed automata. In *Proceedings of TACAS'01*, 2001.

[62] If home page. On the WWW at `www-verimag.imag.fr/async/IF/index.html`.

[63] R. Jain. *FDDI Handbook: High-Speed Networking using Fiber and Other Media*. Addison-Wesley, 1994.

[64] The Liège Automata-based Symbolic Handler (LASH). Available at `http://www.montefiore.ulg.ac.be/~boigelot/research/lash/`.

[65] R.F. Lutje Spelberg, W.J. Toetenel, and M. Ammerlaan. Partion refinement in real-time model checking. In A.P. Ravn and H. Rischel, editors, *Proceedings of 5th FTRTFT*, volume 1486, pages 143–157, 1998.

[66] S. Luttik. Description and formal specification of the link layer of ieee-p1394. In *Proceedings of FICS'97*, pages 43–56, 1997. Also available as Report SEN-R9706, CWI, Amsterdam.

[67] Monika Maidl. Simple representative instantiations for the pgm protocol. Unpublished.

[68] Z. Manna and A. Pnueli. Completing the temporal picture. *Theoretical Computer Science*, 83(1):97–130, 1991.

[69] K.L. McMillan. *The SMV system*. Cadence Berkeley Labs, 1999.

[70] A. Miné. The octagon abstract domain. In *AST 2001 in WCRE 2001*, IEEE, pages 310–319. IEEE CS Press, 2001.

[71] J. Moller, J. Lichtenberg, H.R. Anderson, and H. Hulgaard. Difference decision diagrams. Technical Report IT-TR-1999-023, Technical University of Denmark, 1999.

[72] Omega Team. *The Omega Library*, November 1996. version 1.1.0.

[73] P. Pettersson and K.G. Larsen. UPPAAL2k. *Bulletin of the European Association for Theoretical Computer Science*, 70:40–44, February 2000.

[74] A. Pnueli and E. Shahar. Liveness and acceleration in parametrized verification. In *Proceedings of the 12th CAV*, volume 1855, 2000.

[75] V. Pratt. Two easy theories whose combination is hard. Technical report, MIT, Cambridge, September 1977.

[76] J. Romijn. A timed verification of the ieee 1394 leader election protocol. In *Proceedings of FMICS'99*, pages 2–29, May 1999. Appeared in Formal Methods in System Design 2000.

[77] C. Shankland and M. van der Zwaag. The tree identify protocol of IEEE 1394 in μ-CRL. Technical Report SEN-R9831, CWI, Amsterdam, November 1998.

[78] R. Shostak. Deciding linear inequalities by computing loop residues. *Journal of the ACM*, 28(4):769–779, October 1981.

[79] M. Sighireanu and R. Mateescu. Verification of the link layer protocol of the ieee-1394 serial bus ("firewire"): an experiment with e-lotos. *Software Tools for Technology Transfer*, 2(1):68–88, 1998.

[80] David P. L. Simons and Mariëlle Stoelinga. Mechanical verification of the IEEE 1394a root contention protocol using uppaal2k. *International Journal on Software Tools for Technology Transfer*, 3(4):469–485, 2001.

[81] D.P.L. Simons and M.I.A. Stoelinga. Mechanical verification of the ieee 1394a root contention protocol using uppaal2k. Technical Report CSI-R0009, Computing Science Institute Nijmegen, May 2000. Conditionally accepted to STTT.

[82] Normed simulations. W. griffioen and f. vaandrager. In *Proceedings of CAV'98*, volume 1427, pages 332–334, 1998.

[83] IEEE Computer Society. Draft standard for a high performance serial bus (supplement). Technical Report P1394a Draft 2.0, IEEE, March 1998.

[84] Tony Speakman, Dino Farinacci, Jon Crowcroft, Jim Gemmell, Steven Lin, Dan Leschiner, Michael Luby, Alex Tweedly, Nidhi Bhaskar, Richard Edmonstone, Todd Montgomery, Luigi Rizzo, Rajitha Sumanasekera, and Lorenzo Vicisano. Pgm reliable transport protocol specification. RFC 3208, IETF, 2001.

[85] M. Stoelinga and F. Vaandrager. Root contention in IEEE 1394. In *Proceedings of 5th AMAST Workshop on Real-Time and Probabilistic Systems, ARTS'99*, volume 1601, 1999.

[86] A. Valmari. State space generation with induction. In *Proceedings of the Scandinavian Conference on Artificial Intelligence*, pages 99–115, June 1989.

[87] P. Wolper and B. Boigelot. On the construction of automata from linear arithmetic constraints. In S. Graf and M. Schwartzbach, editors, *Proceedings of the 6th Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 1785, 2000.

[88] S. Yovine. Kronos: A verification tool for real-time systems. *Springer International Journal of Software Tools for Technology Transfer*, 1(1/2), October 1997.