Analyzing a space-protocol: from specification, simulation to experimentation

Marc Andreu, Michèle Haziza, Claude Jard, Jean-Marc Jézéquel

Analyzing a Space-protocol: from Specification, Simulation to Experimentation [1]

Marc Andreu[a], Michèle Haziza[a], Claude Jard[b] and Jean-Marc Jézéquel[b]

[a]Matra-Marconi-Space, ZI du Palays, F-31077 Toulouse, FRANCE

[b]Irisa/CNRS, Campus de Beaulieu, F-35042 Rennes, FRANCE

Abstract

The space industry demands that software components, like the others, draw near the "zero-default" quality. It is now acknowledged that fulfilling this goal for complex programs like distributed applications or communication protocols, requires the use of formal specifications and of computer-aided verification tools. This paper describes an evaluation led by the aerospace company Matra-Marconi-Space in cooperation with Irisa (an academic research center) on the interest and suitability of formal methods and related technologies in this context. This evaluation involves an actual Matra-Marconi-Space specific space protocol (SDM+), the use of formal methods based on the FDT Estelle, and an experimentation tool called Echidna (made at Irisa) to simulate and prototype protocols on real distributed systems. We describe here this evaluation process along with the main conclusions we drew on it.

Keyword Codes: I.6.4; C.2.2; F.4.3
Keywords: Model Validation and Analysis; Network Protocols; Formal Languages

1 Introduction

For particular systems, software failures can have catastrophic consequences. In the space area for example, a simple bug can endanger the success of the whole mission, even induces a satellite crash.

Consequently, the space industry demands that software components, like the others, draw near the "zero-default" quality. It is now acknowledged that fulfilling this goal for

[1]This work was partly funded by Matra-Marconi-Space via a research contract with Irisa

complex programs like distributed applications or communication protocols, requires the use of formal specifications and of computer-aided verification tools.

Finally, it is not surprising that MATRA-MARCONI-SPACE, which spends a lot of time to design specific protocols for space applications, has become particularly interested in the so-called formal description techniques, and wanted to evaluate formal methods and related technologies on its own problems.

Contacts were established with IRISA, which proposed its know-how in the ESTELLE [2] technology, and was interested in the industrial evaluation of its prototyping tool, named ECHIDNA [6, 5]. So a contract was made allowing MATRA-MARCONI-SPACE to launch such an evaluation on one of its proprietary space protocol called SDM+.

This paper describes this evaluation process, involving both methodology and technology. Section 2 presents the evaluation context: the SDM+ protocol, the methods and the tools used. In section 3, we describe the first stage of our approach: the formalization and modeling of the protocol, as well as its validation through intensive simulations. The second stage, described in section 4, refers to the use of parallel computers to prototype and experiment the protocol. In section 5, we present the main points made through the evaluation process, and we conclude giving some prospects.

2 The context of the evaluation

2.1 The SDM^+ space-protocol

SDM^+ is a system of selective dissemination of satellite information protected via a ground feedback path (using TRANSPAC, the X25 french public network): some front-end receivers (called *active front-end receivers* and denoted *FE* in the following) can ask a front-end emitter (denoted *FR* in the following) to retransmit some badly or non-transmitted messages. Massive repetition requests are avoided through the use of a pilot FR (denoted *PFR* in the following) directly connected to the FE and anticipating normal FR repetition requests, which are delayed. A parameter sets the maximum number of authorized retransmissions, thus giving some kind of security level.

Satellite broadcast is done via the SDM network developed by MATRA-MARCONI-SPACE and operated by POLYCOM. Figure 1 shows the general architecture of the system.

2.2 Modeling, simulating and experimenting

Once a model of a protocol has been produced, it can be validated through different methods. Validation tools vary widely in their forms and their abilities, but they always requires as input a formal description of the distributed algorithm (or protocol). They output data on properties of the algorithm under consideration that can be viewed with some confidence level.

The designer may attack his/her algorithm by three complementary techniques. One can think about formal checking of properties, for example by analyzing the state space of

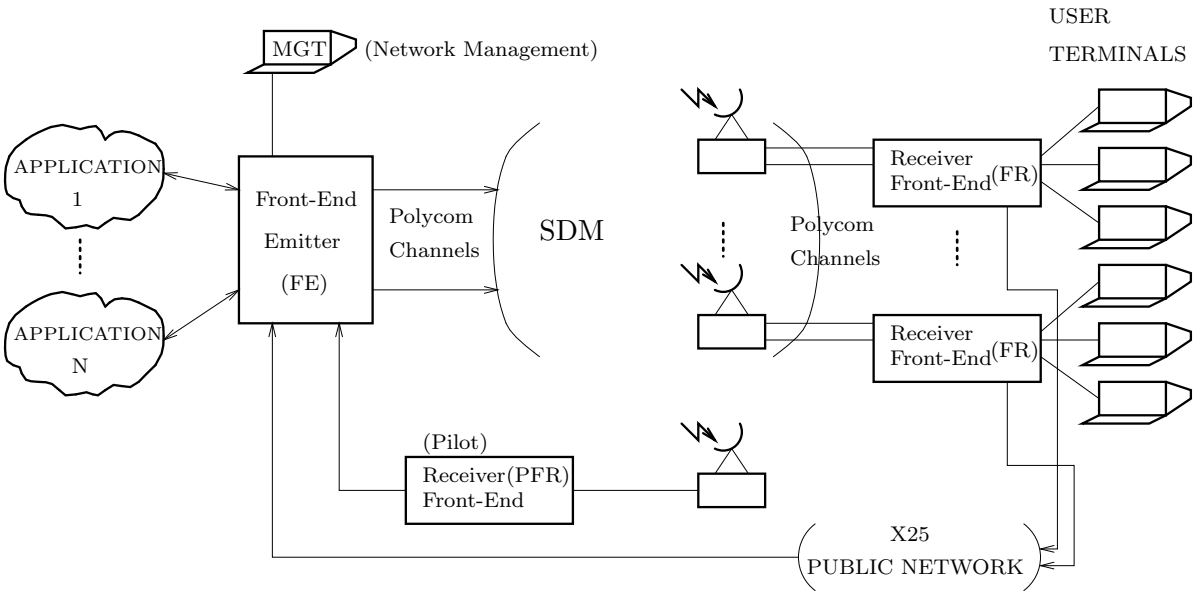

Figure 1: The SDM^+ protocol architecture

the protocol. This gives a definite answer about validity, but existing methods can only easily be applied to analyzing simplified models of the considered protocol. This forces the distributed algorithm to be described at a high abstraction level, so its formal verification lets widely open the problem of property preservation during its refinement course.

Another mean is *protocol simulation*. This kind of simulation is usually centralized and permits to observe and control numerous aspects. It can deal with more refined models of the algorithm and can efficiently detect errors on a subset of the possible algorithm behaviors. The main difficulty is to formally describe and simulate the execution environment. This is generally very simplified, because it wouldn't be realistic (nor interesting) to take into account all the parameters of a real system, as for example, the exact influence of message size on transmission delays, or the action durations (which are not computable without execution).

A usual way is also to develop a prototype implementation, along with special purpose programs for *observation and testing*. Here, the execution environment is obviously the real one. But beyond the large effort of development implied, the protocol behavior is hardly observable and largely depends on implementation conditions. Indeed, it may closely depend on implementation choices (e.g. non-determinism resolution), and so it is difficult to generalize the possible behaviors from the observation.

It appears that these approaches are more complementary than in competition, and that an advised designer would try to use all of them. To help him, IRISA proposes an intermediate technique, (called *experimentation*) between random simulation and prototype implementation that consists in experimenting on parallel computers.

Experimentation aims at observing the algorithm execution in a real environment (i.e. on a real distributed system) while allowing the larger subset of possible behaviors to occur. Some environmental parameters are produced by the parallel computer and the

system observation provides information on the efficiency and quality of the protocol. If this distributed system is general enough in providing a controlled environment, we will be able to transpose some aspects of the experimental behaviors onto any other system, more or less modulus the experimentation machine. Various experiments running on the Intel iPSC hypercube have confirmed this point of view [1].

Another advantage of experimentation on parallel machines is that we can fully use the power of such machines, to make it feasible the validation of large distributed algorithms made of a few thousand processes. However special techniques must be used to perform distributed observation and measure acquisition.

This concept is provided by the ECHIDNA tool, developed by IRISA and designed for the experimentation of distributed softwares.

In order to promote the technique, this tool has been applied to the SDM protocol developed at MATRA-MARCONI-SPACE. This "reverse engineering" approach allowed us to conduct the study while bearing in mind two objectives.

The first one is a blind study to check the model and tool validity (do they permit to trace the already identified errors?). In this case, ECHIDNA is used as an ESTELLE specification tool then as a simulation/validation tool. These functionalities are not specific to ECHIDNA (other tools are available and even more performing to that end) but are an indispensable step in the development and validation process of a model.

The second objective was to study ECHIDNA as a tool for experimenting with parallel computers (for which it has been designed). During the first phase, we tried to qualify the model in use as well as its distributed operation; in the following phases, we studied the SDM protocol itself.

2.3 The ECHIDNA tool

Within the ECHIDNA project, the following tools have been built to provide some kind of continuity between simulation and implementation on parallel machines:

- an ESTELLE compiler —restricted to the so-called static subset of the language. The ESTELLE language [2, 3] can be rapidly described as being based on a model of communicating state machines extended with the Pascal programming language. It includes the whole of the Pascal language, but encapsulates it in elements which make ESTELLE a genuine language for the expression of parallel behaviors. Three main characteristics should be noted:
  - A specification is made up of several modules. ESTELLE makes it possible to specify accurately the interfaces between each of them.
  - The behavior description of each module is precise enough to ensure that the behavior of the specification is unambiguous, or that any ambiguities in it (non-determinism) are explicit.
  - The concept of typing (type definition and instance creation) from the Pascal language is extended to parallel objects such as modules, channels, etc. Some of them are parameterizable.

- distributed runtime kernels for Intel Hypercubes *iPSC/1* and *iPSC/2*, for Transputers based FPS-T40 and Telmat T-nodes, for networks of Sun workstations (with TCP/IP) and for PCs.
- a set of software tools to observe the behaviors of the distributed algorithm under experimentation (global time builder, non intrusive trace, auto-recording of events, etc.)
- facilities to interface an ESTELLE program with its environment
- a windowed interactive source level debugger, available on SUN workstations.

ECHIDNA has been already used to compare various versions of protocol prototypes, in terms of efficiency, resource management, etc., and to measure real time related performances. It can also be used to discover unexpected situations, like macroscopic effects of subtle perturbations.

It can come into play at different phases of protocol development like description/specification/checking with the interactive simulation and graphic interface, and during the experimentation and observation phase with embedded functions permitting to obtain traces. With an additional instrumentation, it is possible to validate some kind of properties (by building trace checkers) and to have an idea —at least qualitatively— of the system performances (test automation and curve plotting).

It is necessary to keep the following points in mind concerning this experiment and especially when it comes to interpreting the simulation results. We study a model of the protocol and not the protocol itself; consequently, the results must be considered as qualitative indications (curve shapes). Lastly, the conclusions may depend on the experimental set-up: target computer, simplified representation of the environment, etc. The model has to be qualified when necessary. Some limitations are also imposed by the chosen method of formal description: ESTELLE is hardly adapted to the expression of real time constraints and to broadcasting communication.

3 Protocol modeling and validation

3.1 Modeling SDM+ in ESTELLE

We restricted our study to the internal protocol of data transfer. We won't be dealing with the network management aspects (FE-MGT dialogue) nor with flow control on the emitting application function of the retransmission flow as performed by FE. This leaves us with a real size protocol whose complexity is still manageable in such a project aimed at evaluating a method and a tool.

From the SDM documents and some already described state machines (work carried out by a trainee who managed to sketch an initial model of the ESTELLE protocol), it was very easy to grasp the overall behavior of the system in a few thousands of ESTELLE lines, see figure 2.

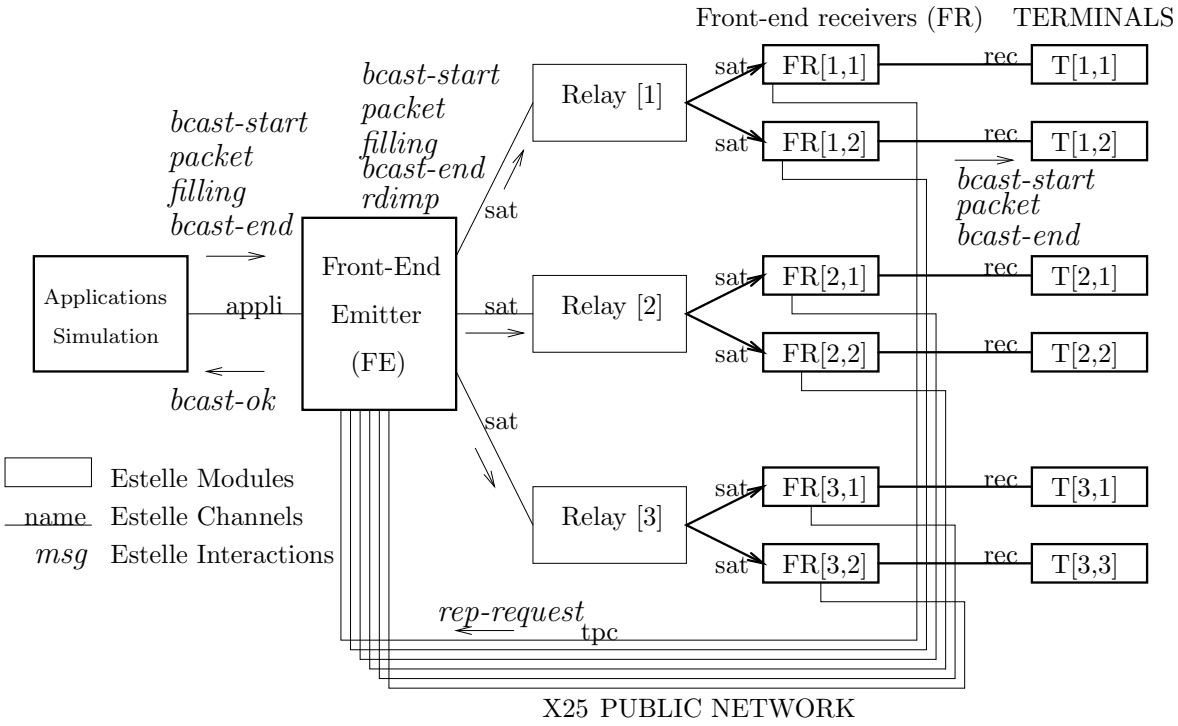

Figure 2: The ESTELLE model of the SDM^+ protocol

The model was then improved by interviewing the designers. We have to point out that the description of a protocol with abstract machines using a formal language such as ESTELLE implies the accurate, rigorous and comprehensive understanding of the protocol for whatever configuration may occur during implementation. Before starting simulation and validation procedures, it was possible to identify the protocol, to visualize its global structure (split into modules), to display its operating modes in a simple and natural way (automata) and to pinpoint possible faults or unexpected but likely situations. It was possible to have a systematic preliminary approach by checking that all messages in the system are processed in each state of the machines.

Modeling a protocol raises the problem inherent to the level of abstraction necessary for the study. One has to strike a compromise between simplification which gives more flexibility (rapidity of modifications) and the adequation with the real system which leads to more accuracy and a better readability of results. A formal description method permits to modify the level of abstraction while designing the model.

We have used this facility to smooth the rough initial model while new functionalities are being displayed and controlled. For example, in a first phase, the selective broadcasting aspect was discarded. The FE sent as many messages as there were receiving stations. When the model was expanded, relays were introduced to represent the features of the FE-FRs satellite link and the ensuing disturbances. The relays play a twofold part: error entry and pseudo-broadcasting. The ESTELLE channels are point-to-point which excludes the broadcast of a single message to several modules simultaneously. In order to avoid the artificial overload of FE with the increasing number of receivers, relay modules take over

to send the messages to FRs. So, whatever their number may be, FE only sends three messages: one to each relay (see figure 2 for the ESTELLE architecture of the protocol).

3.2 Using simulation

During the development of this procedure, we used the interactive simulation functionality available on ECHIDNA. This function was practical and efficient based on the principle one window one module. It allows the user to direct the simulation running by choosing adequate transitions while visualizing the different states, variables and messages. It can also launch simulations within a given time limit (number of steps) according to different strategies (synchronous, asynchronous, sequential). It is able to return to the previous stage in order to study a scenario more precisely and to keep track of the case history of the different transitions. To go further in this phase, the technique of probes or observers as provided by VEDA [4] for example would have been necessary.

However, using interactive and intensive simulations, we found easily some real errors existing in the SDM+ protocol:

- the consecutive loss of the indication of broadcasting interruption or of following broadcast disturbs largely the system.
- the reception of the following broadcast while FR retrieves errors is not specified in the system.
- a wide gap between the FRs is to detriment of the slower ones: the same request can only be treated once by the FE.

4 Prototyping on parallel computer

4.1 Principles

Our target machine for experimentation was the Intel/iPSC2 hypercube, located at IRISA. This is a parallel computer whose architecture is of hypercube topology design, composed with 64 Intel 80386 processors with 4Mbytes of memory each. It provides a really parallel communication network.. It was then possible to study the global behavior of our distributed algorithm in a real and efficient parallel environment, as the relative speeds of communication links and of processors approximate the real situation of the protocol. In our case we had simple control means at our disposal through tests already implemented on the SDM protocol: the reverse engineering principle permits to check whether the implementation on the hypercube is relevant.

In our terminology, an experiment is a run of our protocol ESTELLE model on the parallel machine with a given set of parameters. The results of an experience are used to display a point on a graph representing the variation of some relevant performance feature relatively to a parameter.

The experiments were automated by using shell script files. A first script rewrites the ESTELLE source program by changing the value of a given parameter of the specification (one of the constants defined at the beginning of the program), then compiles the obtained file, allocates the number of necessary processors for hypercube operation and launches the simulation and retrieves traces in the output file as often as there are values to be attributed to the studied parameter.

Then a second script file is loaded to analyze the output file previously obtained in order to plot the curve representing a performance parameter function of the studied parameter. A few very simple UNIX commands (awk, gnu-plot, ...) facilitate all these operations.

For each variable whose influence was studied, two simulation results have been chosen as representative parameters: the proportion of lost messages and the time-lag between first and last message sent by the application module. Figures 3 to 6 show a sample of the kind of results we obtained.

4.2 Checking the consistency of the model

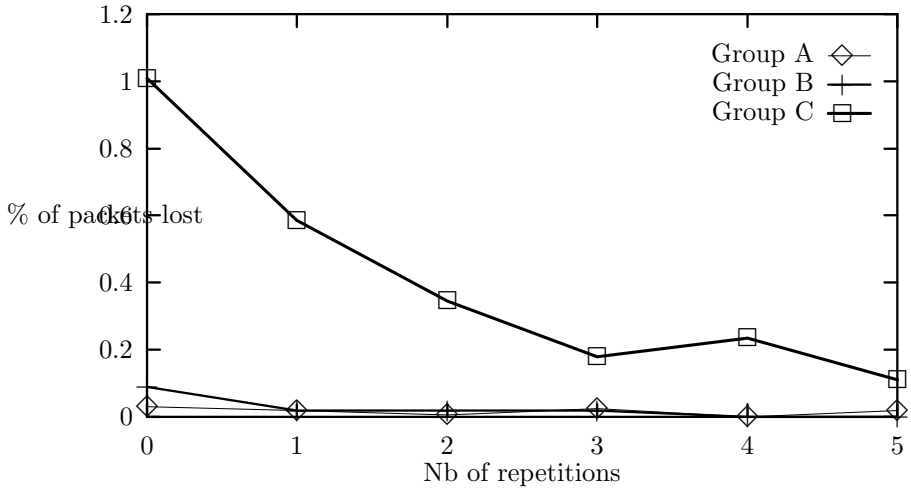

Figure 3: Performance results: lost messages related to the number of repetitions

We started by model qualifications. Some curves were plotted to check the consistency of results when referring to the influence of the selected parameters. They were reassuring because compliant with the expected behavior. This is the case with the results on the rate of lost messages function of the number of requests for authorized retransmissions (see figure 3), and on the rate of losses function of error rates on the link (see figure 4).

For the first case, it drops then levels off a threshold from which an additional request won't improve the performance. Some errors may occur during broadcasting or retransmission so that several repeats may be necessary for one message. However on the one hand, a message is unlikely to be distorted during several successive repetitions. On the

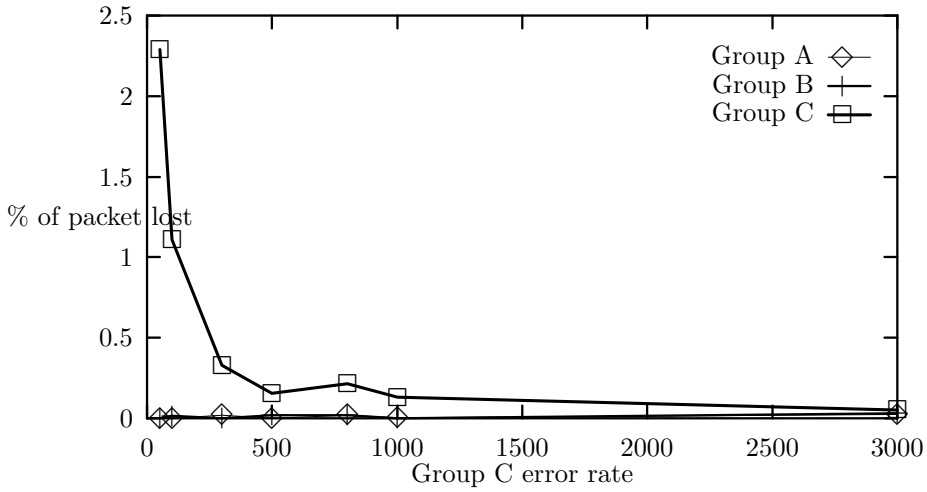

Figure 4: Performance results: lost messages related to the error rate of the line

other hand, after a certain time lapse, the initial messages of the emitting buffer are replaced by the new ones and the FE cannot transmit then any longer.

For the second case, the study was conducted by modifying the rate of errors in one single group, which permits to study the independence of the three groups. The resulting curve shows, as was expected, that the number of packets lost decreases when increasing the mean number of packets between two streams of errors. It seems to indicate that a very poor reception in one of the receiver groups does not disturb the behavior of the others.

4.3 Getting qualitative results

After checking the consistency of the model behavior, we can draw several qualitative conclusions about other experimental resulting curves.

Concerning the influence of the number of broadcast packets, the two corresponding curves are exponential; which may be accounted for by the increasing load due to repetitions (see figure 5). In fact, the higher the number of broadcast messages, the more errors can occur. This brings about an increase in the number of repeats, i.e. the number of circulating messages, of errors, ... This "avalanche" phenomenon may be a problem but according to the exponential curve coefficient, the values are plotted in the nearly linear portion of the curve: this should be checked on the system with true scale tests.

The second point is the influence of the number of receivers (see figure 6). The study was conducted by gradually increasing the number of receivers in each of the three groups. According to the results, both the simulation time and the rate of errors are linear, which may entail an overload of the FE. Let us note that in the case of our model in which the PFR is not represented, the situation can illustrate the two different cases where the PFR does not play its monitoring role any longer (failure) or the PFR is in another group of receivers (not altered by the same disturbances).

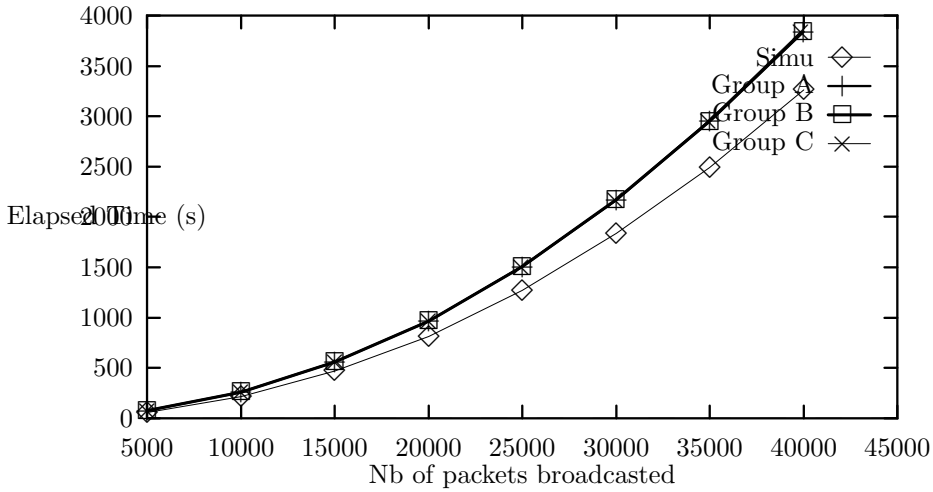

Figure 5: Performance results: influence of the number of broadcast packets

We have also studied the influence of the security level on simulation time (not shown here). The simulation time does not or hardly depend on the number of authorized repetitions. Since the rate of errors decreases when the number of repetitions rises and then levels off, it is not of any interest to limit this parameter unless the threshold is reached.

5 Evaluation balance

5.1 Using formal methods

We have explained how the ESTELLE formal description language made it possible to structure a protocol specification (modules), and to define it precisely in a clear and legible way (automata).

It facilitates the variation in the level of abstraction according to the study in progress and the aim of modeling. The same model can be used for specification, validation or experimentation purposes.

Most problems related to the system operation were raised during the ESTELLE description phase and we were able to identify quite rapidly and in blind mode some of the errors already identified by the designers. This first stage seems to be quite useful to get a comprehensive and accurate reference document easy to consult (ESTELLE basic knowledge can be acquired very quickly from Pascal or any other structured language and from notions on finite state automata).

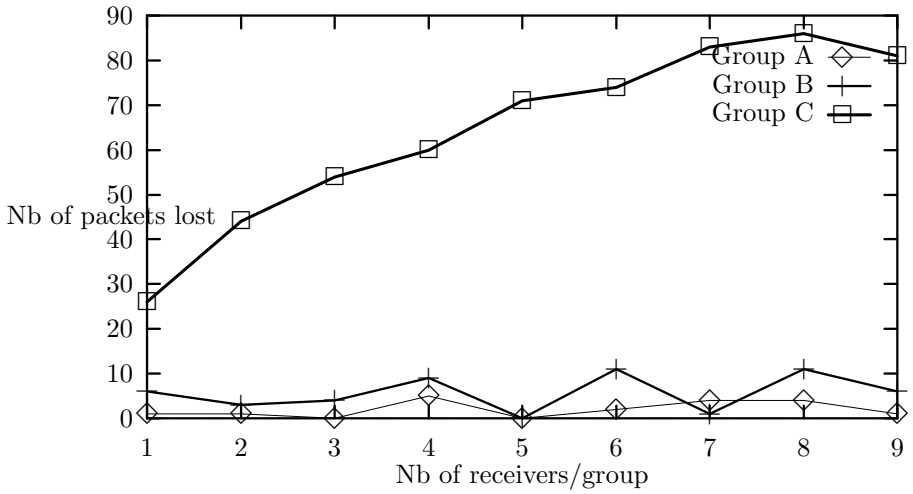

Figure 6: Performance results: influence of the number of receivers

5.2 Using ECHIDNA

Concerning the ECHIDNA tool, we found the development of the specification via interactive simulation and trace observation from distributed implementation is a valuable facility. It revealed itself as a good means of communication between designers, developers, suppliers and clients. This can be attributed to:

- the use of the ESTELLE language (see above),

- the interactive simulation function which permits to understand how a given protocol actually works,

- the trace recording function facilitating observation.

We found however some limitations, particularly linked to the problem of time. Delay clauses of ESTELLE implemented carefully on the top of physical clocks, would have simplified the definition of experiments.

All the potentialities of such a tool have not been used in the framework of the study, but it may be interesting here to review them:

- all the exchanged messages can be buffered, which will permit to re-execute a distributed trace for analysis.

- there exists a simple ESTELLE-C interface using the ESTELLE so called *primitive* functions, written in C language and stored in libraries. This is useful to go beyond the limited scope of ESTELLE.

- quasi-automated implementation on the target system: the interface with the target system, i.e. approximately 10 percent, should be written since the generated code is of the rather high level of the ESTELLE virtual machine. The difference in efficiency is compensated by saving time and the insurance that the system matches the validated model.

6 Conclusions

The experiment of using an FDT at Matra-Marconi-Space was very valuable and conclusions are positive at different levels. For the designers, it offered a structured and standardized development method, an instant identification of problems, hints for building of test scenarios, starting from the observations made during simulations and outlining erratic situations or efficacy limitations, and a know-how transfer to any person interested in the protocol. For developers, it provided a detailed and non ambiguous specification. For Matra-Marconi-Space the key point is the use of homogeneous design methods that permit to increase confidence by a thorough study before the prototyping or implementation phase and make possible testing the system during the development phase.

References

[1] M. Adam, Ph. Ingels, and M. Raynal. *Algorithmes distribués synchrones et systèmes répartis asynchrones : concepts, mises en œuvre et expérimentations.* Rapport de Recherche RR-0862, INRIA, Centre IRISA, Rennes, July 1988. 27 p.

[2] ISO 9074. *Estelle: a Formal Description Technique based on an Extented State Transition Model.* ISO TC97/SC21/WG6.1, 1989.

[3] ISO 9074. *Proposed draft addendum to ISO 9074:1989 — Estelle tutorial.* ISO 9074:1989 ISO/IECJTC1/SC21/F60.

[4] C. Jard, R. Groz, and J.F. Monin. Development of VEDA: a prototyping tool for distributed algorithms. In *IEEE Trans. on Software Engin.*, pages 339–352, March 1988.

[5] C. Jard and J.-M. Jézéquel. ECHIDNA, an Estelle-compiler to prototype protocols on distributed computers. *Concurrency Practice and Experience*, 4(5):377–397, August 1992.

[6] C. Jard and J.-M. Jézéquel. A multi-processor Estelle to *C* compiler to experiment distributed algorithms on parallel machines. In *Proc. of the* 9^{th} *IFIP International Workshop on Protocol Specification, Testing and Verification, University of Twente, The Netherlands*, North Holland, 1989.