Assessment of ESTELLE and EDT through Real Case Studies

S. Haddad[a], M. Taghelit[ab] and B. Zouari[a]

[a]Université Pierre et Marie Curie, MASI UA-CNRS
4, Place Jussieu, 75252 Paris Cedex 05, France

[b]Université de Tours, Faculté des Sciences et Techniques
Parc de Grandmont, 37200 Tours, France

Abstract

This paper presents the evaluation experiments of two distributed algorithms using the ESTELLE language and the simulation tool, EDT. Evaluation is carried out by series of simulations in order to determine the most accurate parameter values of the studied algorithms. The interest is to highlight the process of modelling, simulating and evaluating that is common to these experiments in spite of the algorithms diversity. We discuss the difficulties that we have met during this process. Then, we suggest improvements of ESTELLE and its simulation tools to overcome these difficulties.

Keyword Codes: I.6.3; I.6.4; I.6.7

Keywords: Simulation and Modeling, Applications; Model Validation and Analysis; Simulation Support Systems;

1. INTRODUCTION

The evaluation of distributed algorithms is an important step in their analysis. During the evaluation of algorithms, different complexity indices are measured such as service time, memory occupation, etc. These measures mainly depend on three factors:

- environment hypotheses (e.g. reliability degree of the medium),
- heuristic choices (e.g. first fit versus best fit memory allocation policies),
- determination of the algorithm parameters (e.g. timer values in protocols).

Such measures may be obtained either by analytic methods (e.g. markovian analysis) or by simulation techniques. For the latter method, one needs a specification model and a related tool which supports a simulation mode. Formal Description Techniques are an adequate framework for the evaluation of communication protocols and distributed algorithms. Among these techniques, ESTELLE is often chosen to specify such algorithms since it provides numerous facilities like for instance: dynamic creation of entities and links, expression of time constraints and communication by messages. So, numerous analysis tools based on ESTELLE have been developed [1-7]. Generally, these tools use simulation techniques (exhaustive or not) that provide qualitative results (partial or not).

The aim of this paper is to discuss what should be improved in ESTELLE and its simulation tools in the light of two experiments in algorithm evaluation. The two experiments have been driven with the tool EDB [4].

The first experiment concerns the verification and the evaluation of an industrial protocol developed by Matra-Communication for the management of a High-Speed network [9].

The second experiment concerns the tuning of an optimized algorithm for the management of a communication chain between two entities [8]. With the help of the tool, we have tried to obtain the best performances for this algorithm given different hypotheses on the medium's reliability.
The three steps of the experiments, namely modelling, simulation and evaluation have outlined the following problems:

- in ESTELLE the genericity of an algorithm is limited to processes since most of the links must be explicitly specified between process instances and not between generic entities (i.e. modules).
- the specification of the expected behaviour of a medium - message loss and/or sequencing, minimum/maximum transit time,… - requires an additional modelling prone to errors and which confuses the model.
- as the modelling of message loss is done with transitions, a useful feature of a simulation tool should be the specification of a probability law for transition firings. This feature is missing in EDB.
- a complementary feature would be the specification of an execution time in order to model the transit time of a message. To overcome these problems some PASCAL procedures have been added to our "medium protocol". Once again the specification is confused.
- when one wants to compare the quantitative results of different simulations, it is essential to have a common scale of time: obviously it can be neither the number of transition firings nor the CPU (or elapsed) time of the execution. EDB does not provide such a scale.
- an advanced feature (absent in EDB) of a tool would be the iteration of simulations with different values of parameter with possibly: a strategy to obtain an optimum value, value series for the parameters and expressions for the outputs (e.g including operators as average).

The results of our work are the foundation for the development of a software tool based on an ESTELLE-like language. This is to be done in the scope of the framework of the European Eureka-Irena project (EU 389: Industrial Requirements Engineering Based On Nets For Value Added Applications).

In Section 1 we briefly recall the main features of ESTELLE and EDT. In Section 2, we present the principles of the two experimented distributed algorithms. Section 3 describes the modelling, the simulation and the evaluation process of the algorithms using ESTELLE. In Section 4 we discuss the difficulties raised by the modelling and the evaluation of similar protocols and we suggest the corresponding improvements.

2. ESTELLE AND EDT

ESTELLE is a formal description technique (FDT) standardized by ISO [10, 11]. It is well-adapted to describe communication protocols and especially those following the OSI functionnal architecture. ESTELLE holds numerous features related to communication protocols as process hierarchy, message communications through FIFO queues, time constraints and synchronous/asynchronous parallelism. In ESTELLE, concurrent processes are called *modules*. An ESTELLE module represents a *generic entity* from which numerous instances can be created. It is defined by the association of a *module header* and a *module body*. The module header defines exported variables and external interaction points whereas the module body describes its behaviour by an extended *automaton*.

The first step to model a system using ESTELLE is to define its architecture. It consists in determining the main components making up the system and how they are interconnected. Hence, each main component may be divided into other components by refinement.

This refinement is obtained by defining "child" modules of a given module according to the hierarchy concept. This induces the parent/child principle which provides execution priority of a parent module over its children modules.

The second step consists in describing the behaviour of the various modules. For every module, one defines a set of reachable states and a set of transitions. A transition is made up of two parts: a condition part which describes the events that must occur to fire the transition and an action part which specifies a sequence of Pascal statements and ESTELLE primitives that are executed when firing the transition. For a given transition, the firing constraints are of four kinds:

- the process must be in a given state or in a given set of states,
- the process must receive a given message,
- the boolean expression including local variables and/or message parameters must be verified,
- the firing of the transition must be delayed a certain interval of time.

The action part may contain Pascal statements (only a subset of Pascal is used) for internal treatment and ESTELLE primitives for message emission or configuration change (creation/destruction of module instances or connexions).

As there are no capabilities to directly specify a variable transmission time or unreliable communication, the user has to explicitly model such behaviours by defining modules ensuring these tasks. In ESTELLE, the "delay" clause is useful to achieve such modelings. Indeed, the non-determinism induced by the two parameters of a delay clause allows the modelling of most of the predictable environment events.

Numerous ESTELLE tools have been developed since its definition [12, 13]. In this paper, we discuss the modelling capabilities of simulation tools through the EDT one [14]. EDT represents a toolset composed of :

- an ESTELLE compiler, which checks an ESTELLE specification and translates it into C source code,
- an ESTELLE Simulator/Debugger, called EDB [4], which allows verification of an ESTELLE specification.

In accordance with the "atomic transition" principle of the ESTELLE semantics, the unit of execution in EDB is a simple transition. All sorts of non-determinism inherent to an ESTELLE description are resolved during the simulation either in an interactive manner or automatically by a random selection. EDB simulations can be achieved using the observation approach [15]. Observation is the ability to get informations related to the specification objects during a simulation session. It is a useful way to obtain significant results in order to evaluate a given protocol. Observation may be automatically achieved by "observers". The observers are programs written in the simulation language so as to provide simulation informations. Moreover, one can define a simulation scenario in order to verify predefined properties.

Numerous options are available in order to hold a simulation session. The most important ones are:

- definition of a given run-time for the simulation,
- definition of a maximum number of fired transitions,
- control of the simulation stop by an observer (for example when a given global state is reached).

3. EXPERIMENTS

We have carried out experiments with two communication protocols which have the common feature to be fault-tolerant. The two protocols have to manage and to maintain communication between entities. In the first protocol [9], the entities are computers in a *meshed ring network*

In the second one [8], the entities are processes making up a *communication chain* through any network. Through these experiments, we have encountered similar modelling and evaluation problems. In the following, we briefly present the functionalities of the two protocols and outline the options made in the modelling and evaluation stages. These two protocols have been chosen for their complexity and for the several general mechanisms they use.

3.1. The Meshed Ring Protocol

This protocol has been studied in the framework of an industrial project with Matra-Communication [9]. Our contribution was the verification and the evaluation of the protocol.

The main goal of this protocol is to establish a logical ring configuration on a network whose topology is a meshed ring.

Every host has three physical links: two bidirectional *main* links and a bidirectional *redundant* one. The main links are used to establish the optimum logical configuration. The redundant link may be used to re-establish a logical ring in case of failures.

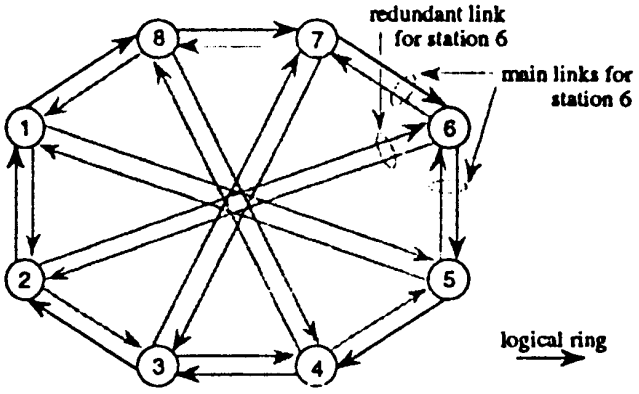


Figure 1: Meshed Ring topology

The main principles of the protocol are:

ELECTION principle: A logical ring must have only one master host whereas the others are slaves.

A logical ring is established when every host knows its status (master or slave) and every slave host knows the master. The election is achieved by exchanging messages and is based on the host identities. Failures may lead to situations where several logical rings coexist.

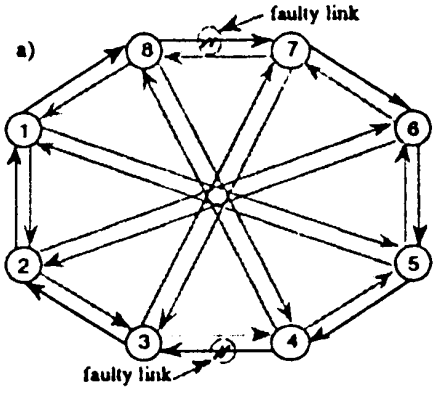
RECONFIGURATION principle: Every host attempts to belong to the largest logical ring.

A reconfiguration is the change of a logical ring configuration. It may be involved by:

- link or host failures.
- link or host restorations.

A typical reconfiguration may consist in splitting a logical ring or merging two logical rings. This last case is illustrated by the following example.

Assume that, initially, a logical ring (black arrows) is established between all the hosts. Assume now that two links fail simultaneously. In the example, only one way of each link is perturbated. The logical ring is broken and the hosts have to re-establish another ring. Each host maintains a table of its configuration links, and according to it can send messages on only one output link. This table is dynamically updated in case of link or host failures.



In the first step, and according to the *ELECTION* principle, all the hosts send messages to elect a master. Because of the two faulty links, two masters are elected then two logical rings are established. The first one is composed with hosts 1, 2, 3 and 8 and the second one with hosts 4, 5, 6 and 7.

Hosts exchange messages including the master identity of the ring they belong to. So, a host knows that the different rings can be merged. This is done in the next step.

In the second step, according to the *RECONFIGURATION* principle, the hosts work to belong to the largest logical ring. In our example, the merging of the two logical rings is achieved by using the redundant link between hosts 1 and 5. This link has been chosen arbitrarily. i.e. any other link could be used.

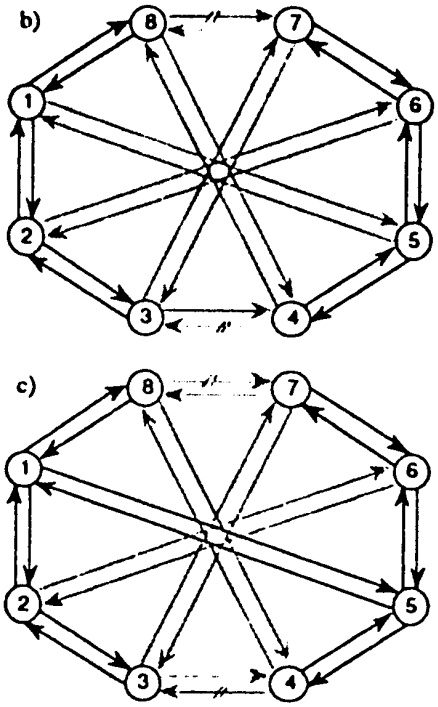


Figure 2: Logical ring re-establishment

In order to ensure the logical ring establishment or reconfiguration, the protocol uses timers. The performance as well as the functioning of the protocol strongly depend on the values of these timers. For instance, in case of multiple reconfiguration schemes, *the management of conflicts is resolved by means of timers.* Therefore, timer values play an important role not only at the quantitative level but also at the qualitative one.

This protocol has been also modelled with ESTELLE and numerous simulations have been achieved using the EDB tool. We have encountered several problems when modelling and simulating this protocol. During the protocol simulations, we have discovered numerous design anomalies which are mainly due to bad choices of timer values. The evaluation of this protocol has allowed us to adjust the timer values and thus to get a correct and an efficient functioning.

3.2. The Communication Chain Protocol

Most distributed systems are designed according to the customer-server model. The general structure of such a model can be viewed as a *chain of cooperating entities.* This model is frequently encountered in numerous distributed applications as file transfer and remote procedure call. An example is shown in Figure 3. Entities El and E6 communicate through inner entities E2, E3, E4 and E5. The communication between host A and host B is managed by entities E2 and E3. Entities E4 and E5 manage the communication between host B and host C.

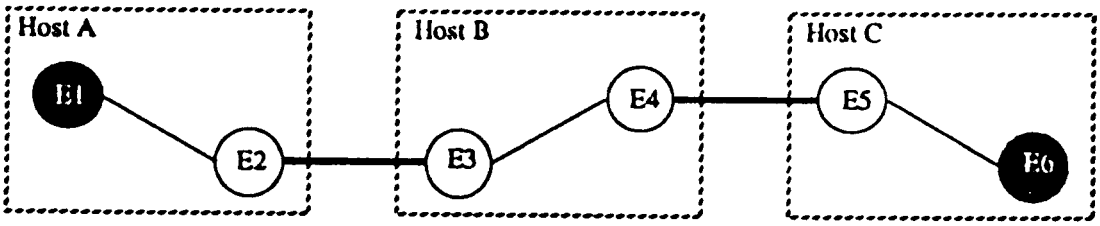


Figure 3: Chain of cooperating entities

The presented protocol aims at the building and the maintenance of a chain of communicating entities in an unreliable environment. The recovery of faulty entities is achieved according to the concept of *dynamic regeneration* introduced in [16]. Dynamic regeneration consists in the dynamic creation of a new entity instance on failure detection of an existing one. It can be regarded as an alternative to the classical fault tolerance techniques based on redundancy. The details on the conditions of applying dynamic regeneration are out of the scope of this paper and can be found in [16].

Protocol targets

- Building of the chain

The building of the chain consists in generating a finite set of entities such that each inner one has a predecessor and a successor. Unlike the other systems where the elements are static, the entities are dynamically generated. On the other hand, an initialization step is required. This step consists in generating the first entity which is on user's initiative.

- Maintenance of the chain in an unreliable environment

If one or several entities fail, the chain breaks and no communication between the initial and ending entities is possible. To keep end-to-end communication, it is necessary to replace the faulty entities. This is carried out by the dynamic regeneration of the faulty entities. Thc treatment of a failure is composed of three steps: the detection, the regeneration of a new entity and its insertion in the chain

When a failure occurs, in the example entity M_i fails, it is detected by the predecessor of the faulty entity. Then, entity M_{i-1} detects the failure of M_i and it regenerates a new entity M_i'. When entity M_i' is created, it inserts itself in the chain, i.e. it connects itself to the predecessor and the successor of faulty entity M_i.

The protocol is carried out by the algorithm whose principles are presented below.

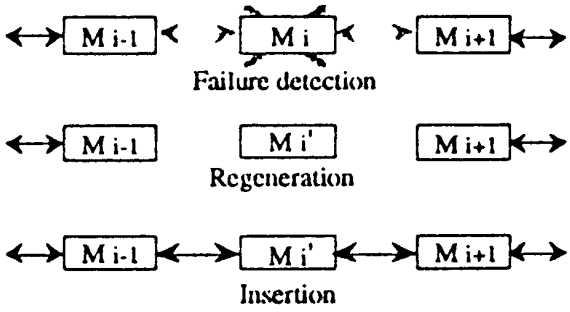


Figure 4: Recovery by dynamic regeneration

Algorithm principles

We want to build and maintain a chain of N entities, where each entity communicates with the adjacent entities. Our algorithm relies on four principles :

ACTIVATION principle: Each non-ending entity carries on the building of the chain.

Each entity with no successor and which is not the N^{th} one generates a successor. This principle allows to carry on the building of the chain if it is not complete and to rebuild it if entities fail.

KNOWLEDGE principle: Each entity knows all its successors.

When a entity fails, it is regenerated according to the *ACTIVATION* principle. Then, this new entity must be attached to the predecessor (the one who generated it) of the faulty entity and to its successor. The regenerating entity must know its two immediate successors so as to give to the regenerated entity the necessary informations to insert itself in the chain.

However, it is not enough in case of multiple failures of adjacent entities. In this case, each regenerated entity has to regenerate a successor until a valid successor has been found.

So, to restore the chain whatever the number of faulty adjacent entities, each entity has to know all its successors.

If the regenerated entity is the initial one, it cannot receive informations on its succcessors from its predecessor since the last one does not exist. In order to avoid the building of a new chain, the initial entity must keep in stable memory information on its successors. This information will be communicated to it, if it is regenerated.

PURGE principle: Each non-initial entity with no predecessor destroys itself after a finite time.

When the predecessor of an entity fails, this one must be eliminated if no new entity replaces the predecessor after a finite time. Hence, "parasite" sub-chains are progressively removed.

SUSPICION principle: The validity of the chain is periodically tested.

Periodically, each entity checks its successor. This mechanism allows to detect faulty entities.

More precisely, the detection is based on an acknowledgement mechanism. Each entity periodically transmits a control message to its successor so as to verify that it is still valid. If no answer has been given after a time t, the successor is considered faulty. This mechanism can bring false detections. whose rate is inversely proportional to t.

3.3. Simulation targets

The analysis of one protocol is an important stage so as to fulfill the designer requirements. These requirements may be of two kinds, qualitative and quantitative. Qualitative analysis consists in verifying whether the protocol ensures the desired functionalities. Quantitative analysis aims at evaluating its parameters to make it high-performance.

Our requirements amount to the following points.

- The well functioning of the protocols: according to different situations, the system must reach a stable state. In the first protocol, a stable state is a state where no more reconfiguration is possible according to the configuration of all the links. For the second protocol, a stable state is the existence of a unique complete chain where each entity holds coherent information.
- The tuning of parameters: it consists in determining the best values of the parameters that have influence on the protocol behaviour. For the meshed ring protocol, the main goal is to minimize reconfiguration time in case of link and host failures. In the case of the chain protocol, the main goal is to minimize *bad detections.* A bad detection is diagnosing that one entity is faulty while it is valid.

For both protocols, and according to the previous main points, we met with the same difficulties and. generally, we applied similar solutions. In the following sections, we mainly outline the modelling and the tuning problems in the case of the chain protocol.

4. Modelling, Simulation and Evaluation

To specify any protocol, ESTELLE imposes the description of the environment in which it executes. The ESTELLE modelling of the protocols distinguishes the modelling of the environment and the modelling of the protocols. The modelling of the environment introduces specific mechanisms which are not directly held by ESTELLE. In our experiments, such mechanisms consist in simulating failure events and the message transit time. The failure events are caused either by entity failures or by message losts. The message transit time varies within a given interval of values. The modelling of the protocols consists in describing the behaviour of each communicating entity.

4.1. Modelling

The environment modelling consists in describing a medium, failure events and a global clock. The medium classically models the transport functionalities (routing, variable transit time) and message losts events. The failure events concern entity failures. The global clock is used as a reference to control the simulation times. We assume that the internal treatment times associated with entities are equal to zero.

4.1.1. Genericity

In ESTELLE, a process is an instance of a generic entity called module. All the instances of a given module inherit the same communication and behaviour characteristics. In particular, they have *the same interaction points defined each with a given role.* However, this leads to some drawbacks when considering the semantics of interaction point connections. Indeed, two connected interaction points must be declared with the same channel but with *opposite roles.* Therefore, it is not allowed to inter-connect instances of a same module with only one interaction point. Such a situation may arise in numerous applications. For example, to create a complete meshed network where all the processes are instances of a same module, we are led to define two interaction points because we cannot connect the instances through the same interaction point.

This difficulty has been also encountered in our experiments. For the meshed ring protocol, the redundant link between stations is modelled with two connections through two interaction points (see Figure 5).

This difficulty is due to a genericity concept which is "restricted" in ESTELLE. In fact, the definition of generic entities and generic interaction points is a basic feature whereas connections are only achieved between instances.

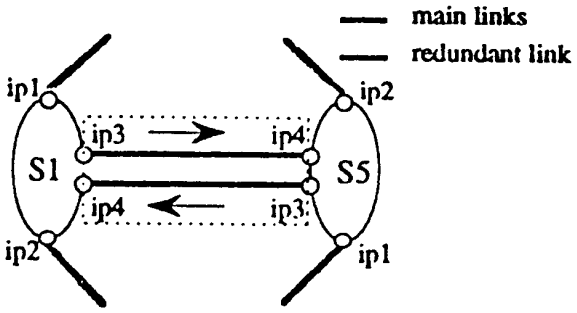


Figure 5: Partial graphical specification of the meshed ring protocol

4.1.2. Absolute time

To achieve *meaningful comparisons between different simulation results,* we need to capture the specification *absolute* time in order to control simulations. Indeed, the common criterion to different simulations for consistent comparison is, generally, the specification time. The EDB tool allows to set a simulation time related only to the *machine run-time* which is independent from the specification time.

For our experiments, to capture the specification absolute time, we introduced a *CLOCK transition.* The *CLOCK* transition is used as a time reference. This transition has a DELAY clause and is the only one defined in the root module. Therefore, it has the largest hierarchical priority and is always enabled. In Figure 6, the TClock parameter represents a clock tip. It is worth noting that this time reference is the same as that of the ESTELLE DELAY clauses.

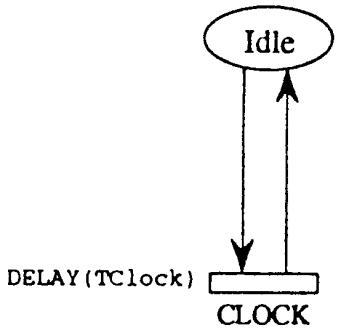


Figure 6: Clock function of the root module

4.1.3. Medium

The role of the medium is mainly the transport of messages between entities. In real applications, a medium is characterised by a message transit time, a rate of message loss and/or out of sequence messages. An ESTELLE specification only induces a communication mechanism through FIFO queues. Hence, the previous features can not be directly specified.

In our experiments, we needed to simulate failures of the medium in which messages are lost. In order to simulate message transit time and since we cannot associate a different execution time with each transition (especially with sending transitions), the following scheme *is* used.

Figure 7 describes in a simplified style the medium modelling. The *START _TRANSPORT* transition periodically invokes the beginning of the message transport. The periodicity is characterized by a *DELAY* clause whose parameters define the time range of the message transit time. The messages from a transmitter instance to a receiver instance are delivered or cancelled according to a loss rate (transition *TRANSPORT).* The *END_TRANSPORT* transition is executed when all the messages have been treated.

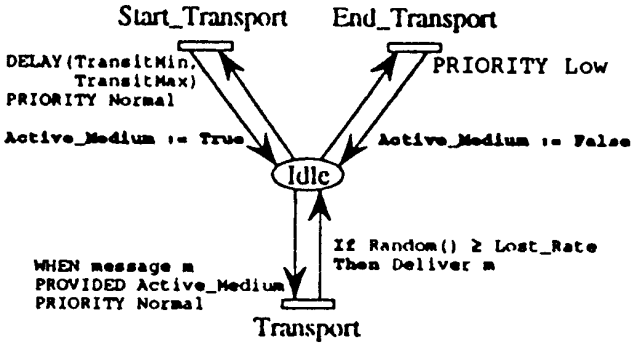


Figure 7: Medium functions

These transitions are defined within a module which has hierarchical priority over those of the communicating entities. The modelling of the module failures is achieved in a similar manner.

4.2. Simulation

Some events related to the system global states can not be controlled by the only modelling. For instance, in an unreliable environment one may verify if the system reaches stable state. In this case, he must be able to stop the occurrence of new failures for a while. The invocation of such events is generally made from an appropriate global state.

4.2.1. Temporary events

In order to verify qualitative properties in our experiments, we have been led to simulate temporary events. For the meshed ring protocol we needed to verify if the establishment of a logical ring is always possible. Therefore, the following scenario has been achieved:

- execution of the system considering host failures and message loss,
- execution of the system without considering host failures and message loss,
- verification of the establishment of the logical ring after a while.

In the example of the Section 2.1, the two link failures broke the logical ring (Figure 2.a). From this moment on and without new failures, two new logical rings are established (Figure 2.b). According to the *RECONFIGURATION* principle, the two rings are merged (Figure 2.c). This state corresponds to a desired configuration.

To control such a system evolution, we handle two boolean "lost_message" and "failure". The "lost_message" variable indicates whether the medium applies an unreliable transmission whereas the "failure" variable indicates whether a link is faulty. Initially, these variables are set to "true". After some simulation time, we change their values to "false" and we observe whether a stable state is reached. Hence, during a simulation session we can directly act (by means of EDB commands) on the medium functioning.

4.2.2. Observers

The EDB tool used for our experiments allows observations during simulations. Observation is the ability to get information related to the specification objects from a simulation session [15]. Observation may be automatically achieved by "observers". The observers are programs written in the simulation language so as to provide simulation information. In particular, observation allows the capture of specific global states. In the example described in Section 2.1, observers have been used to determine the appropriate moment (corresponding to a specific global state) so as to act on the medium functioning.

4.3. Evaluation

In this section we present the simulation results of the communication chain protocol. The different simulations have shown that once a reliable environment is set (it may be set when the simulation is in progress), the system reaches its stable state (the building of a unique chain) in a finite time. In the following, we focus on the more important purpose namely the protocol evaluation. It consists in the tuning of parameters to make the protocol optimum. We test the protocol in extreme situations so as to quickly obtain significant results. These situations are characterized by large message loss and host failure rates.

The protocol evaluation is mainly achieved by means of event accounting. Indeed, for a given simulation, the following information is useful:

- the total number of failure detections,
- the number of bad failure detections,
- the total number of message emission events,
- the total number of message retransmission events.

In order to handle such information, we define "observers" which count the corresponding event occurrences. In general, these events correspond to the execution of particular transitions. In this case, the observers simply provide statistical information on the execution occurrences of the associated transitions.

Three parameters are relevant for the chain protocol: the values of the timers TPRED and TSUCC and the number of message retransmissions (NReem). According to the *SUSPICION* principle, each entity checks its successor every TSUCC time units. On the other hand, TPRED represents the maximum time at the end of which an entity diagnoses a failure or a disconnection from its predecessor if no interaction is made. As long as one entity's message is not acknowledged. it may be sent again until NReem times.

Bad choices of parameter values may involve unfavourable results such as a great number of bad detections and useless messsage retransmissions. Indeed, this leads to useless dynamic regeneration of new entities, which causes system overloading and thus delays the reaching of a stable state.

The optimization concerns the minimisation of bad detections. Indeed, the bad detection reflects a bad functioning of the protocol. It is caused by bad choices of the timer parameters (TPRED and TSUCC) and the number of message retransmissions (NReem). This leads to useless reconfigurations. We proceed in a first stage to the evaluation of the timer parameters. and in a second stage to the evaluation of the message retransmission number.

4.3.1. The tuning or timer parameters

In order to evaluate the TPRED timer value, two opposite constraints must be considered:

- On the one hand, we have to maximize TPRED so as to avoid that an entity assumes it has no more predecessor. Before destroying itself, a given entity must wait a sufficient time to take into account its predecessor normal events even if they are slow.
- On the other hand, we have to minimize TPRED so as to avoid that an entity, which is disconnected from the main chain, uses resources needlessly and for a long time.

The tuning of the TSUCC timer is simpler. Since its role is to determine the periodicity of checking its successor, it has no direct effects on the bad detections. The choice of its value depends on the wished quickness of failure detections.

We study the protocol behaviour according to the values of the ratio TPRED/TSUCC. In the following the simulation hypothesis:

Remark:

The *CLOCK* transition used in the *CHAIN* module provides the basic time unit (btu).

simulation time	= 20 000 btu	
transit time	= [1, 2] btu	(between two adjacent entities)
message lost rate	= 1%	
host failure rate	= 0.02 %	
TSUCC	= 100 btu	

Figure 8 presents a set of curves, each providing the number of bad detections according to the values of TPRED/TSUCC. A given retransmission number is used for each curve.We remark that the number of bad detections decreases when the ratio TPRED/TSUCC decreases. and remains constant from a certain value on. For the two first curves (0 and 1 retransmission), the number of bad detections weakly increases. This is due to the fact that a short simulation time does not exclude atypical simulations. The common conclusion we can make about these curves is: from a certain value. it is useless to wait longer for an entity since it can not reduce the number of bad detections. Finally. for the values 0, 1, 2, 3 and 4 of the retransmission number we respectively obtain the optimum values 1.5, 2.0, 2.5, 2.5 and 3 of the ratio TPRED/TSUCC.

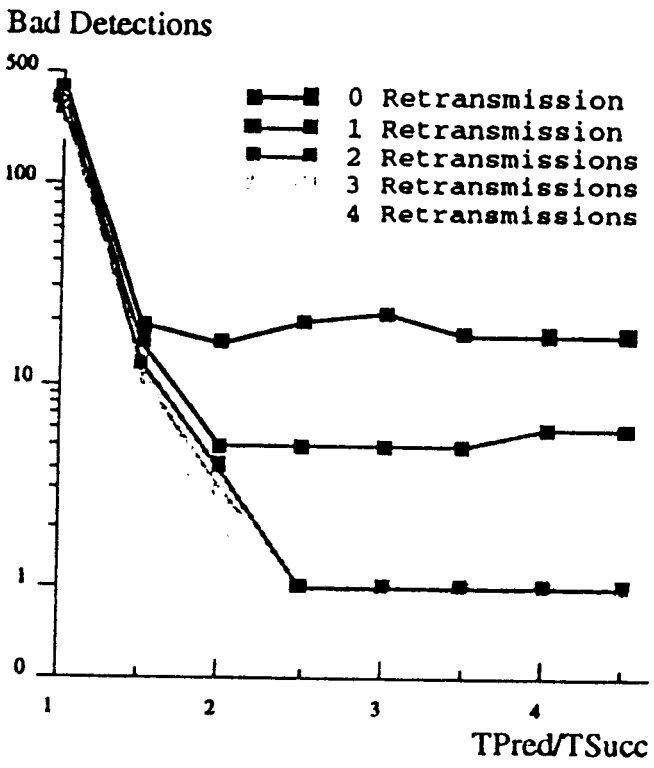


Figure 8: Number of bad detections according to the values of TPRED/TSUCC

4.3.2. The tuning or the message retransmission number

The message retransmission allows to overcome the problem of lost messages. Consequently, it prevents hosts from declaring too quickly their successor or predecessor failed. Similarly to the TPRED parameter, a compromise must be found for the two following constraints :

- On the one hand. if we have few retransmissions, the number of bad detections increases. The extreme case is if we have no retransmissions at all, every lost message leads to a bad detection.
- On the other hand, if we have a lot of retransmissions, we obtain few bad detections but the detection of real failures may be delayed and the medium may be overloaded.

Therefore, we have to determine the number of retransmissions that provides a reasonable value of the cost of bad detections and the cost of retransmitted messages.

In our simulations, we vary the host failure rate and the number of retransmissions and we compute the number of bad detections. We associate the value 300 btu with the TPRED timer.

Figure 9 presents a set of curves, each providing the relationship between the cost of bad detections and the cost of retransmitted messages.

The achieved simulations show that when the failure rate increases, the cost of bad detections decreases. This is quite normal since the cost we consider is the ratio of the bad detections to all the detections (bad detection and real detections).

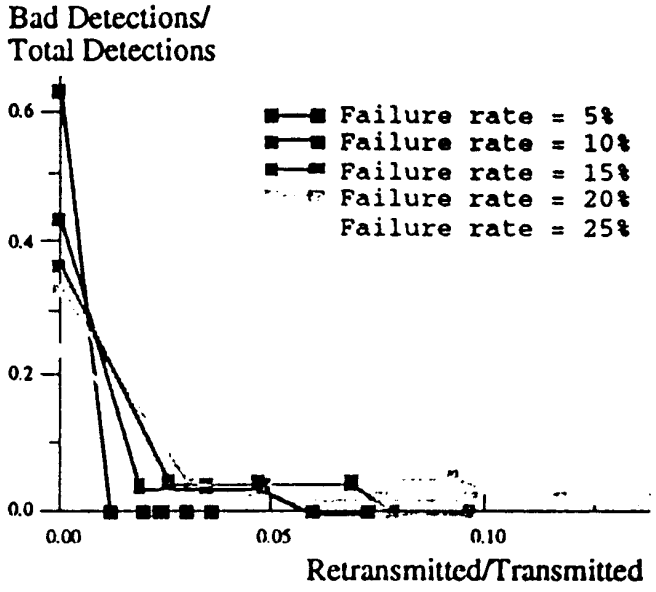


Figure 9: relationship between the cost of bad detection and the cost of retransmitted messages

Remark:

To have an idea on our simulation run-times, we give the values related to the TPRED evaluation:

- with TPRED/TSUCC equal to 1 and 0 retransmission, we have obtained 1082 minutes (cpu time). (which corresponds to 164236 total fired transitions).
- with TPRED/TSUCC equal to 4 and 4 retransmissions, we have obtained 253 minutes (cpu time). (which corresponds to 162307 total fired transitions).

The simulations have been executed on a Sparc machine (Sun 4).

It is worth noting that these values are altered by the used observers. Therefore, they are not very indicative of the protocol complexity.

5. HOW TO IMPROVE ESTELLE AND EDB

Following these two experiments, we can give some critical notes about the simulation technique we used and about simulation tools in general. Along these experiments, we had similar evaluation requirements and we have encountered similar modelling problems. These requirements correspond to an effective need in real applications. Our criticism consists in stating the encountered difficulties in the modelling and evaluation stages, and in giving suggestions to enhance the adequate specification languages and verification tools.

5.1. Language Requirements

The specification of real systems has shown the necessity to introduce new features. The most important ones have been defined in the SSL language [17]. SSL is an ESTELLE-like language which is the formalism used for the protocol verification software developed in the Eureka-Irena project. It emphasizes the features of genericity for conciseness of the protocol specification and formal semantics in order to use verification algorithms with Coloured Petri Nets [18]. The general definition of this language is out of the scope of this paper.

- **Genericity**

In SSL, *the genericity is applied on processes and links.* It means that we can define a set of processes, called *group* in SSL, with a *similar behaviour and similar connections.* Unlike ESTELLE, the SSL connections are defined between groups (i.e. generic entities) and not between instances. This leads to the definition of *generic links.* Here is an example of an architecture description in SSL and in ESTELLE.

Figure 10 describes an example of architecture which is composed of 4 computers and 12 terminals. Each computer is connected to 3 terminals. Computers communicate according to a ring based protocol.

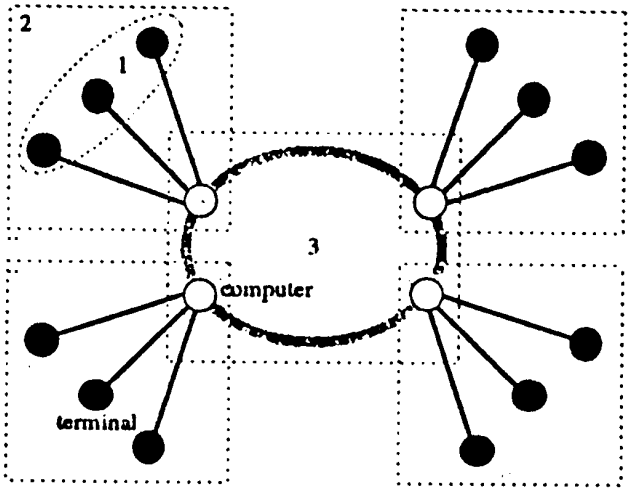


Figure 10: Example of architecture

This architecture is described in SSL as follows:

(Firstly two groups, terminal and computer, are defined and contain one instance each)

```
terminals[3];
```

Duplication to 3 of the terminal group instance (step 1 in Figure 10).

```
ASSOCIATE(computer,terminal,l_ptop)[4];
```

Point to point connections between the computer and all the terminal group instances. Then duplication to 4 of the resulting structure (step 2).

```
ASSOCIATE(computer,computer,l_ring);
```

Ring establishment by connecting the computer instances (step 3).

The same architecture may be described in ESTELLE as follows:

```
ALL i: 1..4 DO IN IT computer[i] WITH computer_body;
ALL i: 1..4 DO
      ALL   j: 1..3
        DO INIT terminal[i,j] with terminal_body;
ALL i: 1..4 DO
      ALL   j: 1..3
        DO CONNECT terminal[i,j].ip TO computer[i].ip[j];
ALL i: 1..4 00
        CONNECT sites[i].succ TO sites[(i mod 4)+1].pred ;
```

- **Medium specification and access**

In SSL, the link has some attributes such as its topology (since it defines a set of links), its alphabet, and its discipline. In its timed version SSL includes the specification of the transit time by means of standard probability law. As a link of SSL is a set of concrete links, the language provides concise mechanisms such as an undeterministic wait of messages or a broadcast primitive. Here is the continued example where a broadcast of the `status_msg` message is achieved by a computer instance on the reception of any `fail_msg` message:

SSL description

```
WHEN    fail_msg ON l_ptop FROM ANY
      BEGIN OUTPUT status_msg ON l_ptop TO ALL END;
```

ESTELLE description

```
ANY i:1..3 DO WHEN ip[i].fail_msg
      BEGIN FOR j:=1 TO 3 DO OUTPUT ip[j].status_msg; END;
```

- **Object approach**

The process and the processes group are a major part of the language. A process is similar to an object and a group is similar to a class. It means that any group defines a type of the language. This type is not equivalent to an interval or an enumeration type for it forbids the use of constants. The fundamental principle of SSL is that a process knows only processes with which it interacts. *Thus processes are never explicitly designated.* When necessary, names are transmitted inside messages and consequently the only name that a process initially knows is its own name using a predefined variable (a very close approach to the one of the object language). Here is the continued example where a computer instance send the `ack_msg` message to the sender of the `req_msg` message:

SSL description

```
/* declaration of the term_ID variable as a groupe terminal type */
VAR  term_Id: terminal;

WHEN req_msg ON l_ptop FROM term_ID
BEGIN OUTPUT Ack ON l_ptop TO term_ID END;
```

When a computer instance receives the `req_msg` message from any terminal instance, the identity of this latter is stored in the `term_ID` variable. The computer instance uses this variable to reply to the sender.

ESTELLE description

```
ANY i:1..3 DO WHEN ip[i].req_msg
      BEGIN OUTPUT ip[i].ack_msg END;
```

5.2. Tools requirements

- **Probability events**

The probability events correspond to real hypotheses in protocol modelling. For instance, host failure and message loss events are submitted to given probability laws. In ESTELLE, only non-deterministic executions may be associated to transitions. There is no way to specify the execution of a given transition with a given probability. In our case, we have been led to write a Pascal procedure associated with a transition so as to model probability events. For instance, the delivery or the loss of a message by the medium is achieved according to a "random" function used in a Pascal procedure.

The ESTELLE simulation tools interpret the non-determinism in the execution of the conflicting transitions by a random choice. If additional information (execution probabilities) is associated with transitions, the probability events can be easily modelled. Such information may be handled at the simulation tool level and need not be considered at the specification level. Moreover, this kind of information would allow the development of performance analysis tools associated with ESTELLE. Some research works have already been interested in a similar approach (EVEDA : variant of VEDA [19]).

- **Message transit time**

Usually, in the modelling of communication protocols, one neglects internal treatment time of processes but takes into account the message transit time. Moreover, the message transit time is not fixed and may vary according to a given probability law. This is essential for quantitative analysis of communication protocols.

ESTELLE does not make any assumption on the execution time of the transitions. Therefore, there is no way to specify an execution time only for emission transitions. ESTELLE implementations provide particular interpretations of transition execution time. For instance, the EDB tool allows to associate a time interval to *all* the transitions of a given ESTELLE module instance. Non-determinism is used to choose a time value within the given interval. This does not resolve our problem since the specified execution time can not be associated only with emission transitions. In our case, to fulfill the hypothesis of a variable transit time, we have created a special protocol (inside the *MEDIUM)* to ensure this functionality.

The suggestion we made about the message transit time is similar to that about probability events. A tool may handle additional information associated with *every* transition to specify its execution time. In this case, the specification of such an information associated only with emission transitions allows the modelling of this functionality. In the same way, this information may be handled at the simulation tool level.

The two previous suggestions (probability events and message execution time) can be used together to model all the medium functionalities.

- **Capture capability or the system time**

The system time may be defined as the time with which the system (described by a protocol) evolves. In ESTELLE, the delay values are assumed to be dynamically changed by a "time process" which is independent of a specification. This means that the system progresses with respect to the delay values and eventually with respect to the execution time of transitions (the latter is considered implementation dependent from the standard point of view [11]). In our case, the execution of transitions is immediate. Consequently, time progresses only when delayed transitions are executed (i.e. when only delayed transitions are enabled in whole the system).

For simulation purposes, it has seemed essential to us to capture the system time (see Section 4). As it has been mentioned, the tuning of parameters is possible only if comparisons are done on simulations having the same system time. Nevertheless, the EDB tool allows to fix a simulation time related to the machine run-time.

It is worth offering simulation commands which allow the capture of the system time. Hence, a user can fix a time for multiple simulations.

- **Parametrization or simulations and automated computation of results**

The evaluation of a protocol involves numerous simulations. These simulations differ about the value of one or more parameters. Hence, it is useful to get the ability of automatically iterating simulations. It would be interesting to support simulation session including: value series for the parameters, expressions for outputs (with operators like average or maximum), etc. An advanced feature could be the specification of a method to obtain an optimum value (e.g. dichotomy research, interpolation method,...). The simulator must manage the appropriate simulations and output the corresponding results.

In the case of our protocol evaluation, with a fixed value of NReem (retransmission number), we are interested in the variation of bad detections according to the values of TPRED/TSUCC.

Such facilities are not available on the used tool, so we have been led to achieve "by hand" the required simulations.

6. CONCLUSION

We have presented in this paper two algorithm evaluation experiments using the ESTELLE language and the tool EDB. These experiments have pointed out the ability of modelling distributed algorithms with ESTELLE and the capabilities of simulation with EDB.

However we have also reported some difficulties encountered during these experiments. On the one hand, these difficulties result from some aspects of the language ESTELLE: the restricted definition of the genericity (which excludes genericity of links) and the difficulty to express attributes of the media (transit time, sequencing,.. .). On the other hand, the tool EDB should be improved to handle: optimization of the observers, time associated to the execution of the model, iteration of simulations with statistical operators and research of optimum.

In the EUREKA project IRENA, we are developing a software tool for protocol verification based on an ESTELLE-like language, SSL, which includes all the improvemcnts shown here and some other ones in order to combine verification by simulation and by formal methods based on Coloured Petri Nets.

REFERENCES

[1] B. Algayres and al..; "VESAR : un outil pour la spécification et la vérification formelle de protocoles"; Colloque Francophone sur l'Ingénierie des Protocoles (CAP'91). Ed. Hermes. pp 253-276

[2] M. Ayache and al..; "EWS: An integrated workstation for the design and the automatic generation of distributed software"; Formal Description Techniques (FORTE'88), Ed. North Holland

[3] J-P. Courtiat; "lntroducing a rendez-vous mechanism in Estelle: Estelle*"; The Formal Description Technique Estelle - Results of the SEDOS project, North-Holland, 1988

[4] EDB. ESTELLE Development TooI; User reference manual BULL/MARBEN Mai 1990

[5] J.C. Fernandez, J.L. Richier, J. Voiron; "Verification of protocol specifications using CESAR system"; Protocol Specification, Testing and Verification, V; IFIP 1986

[6] E. Lallet and al..; "Un outil de génération automatique de l'environnement d'exécution de spécifications ESTELLE"; Francophone Symposium on Protocol Engineering (CFIP'91). Pau France, Ed. Hermes

[7] B. Zouari; "PETRISTELLE: transformation and analysis of ESTELLE specifications"; Francophone Symposium on Protocol Engineering (CFIP'91), 17-19 Sept. 1991, Pau France, Ed. Hennes, pp 89-106

[8] M. Taghelit, B. Zouari; "An Algorithm providing Fault-Tolerance for Layered Distributed Systems: Specification and Testing Using ESTELLE'; ISMM Int. Work. on Parallel Computing, 1991,Trani Italy

[9] S. Haddad, M. Taghelit, B. Zouari; "Modelling, Validation and Simulation of MATRA's Protocol: management of a high-speed netwoork"; MASI-MATRA report (/91-3/SH); June 1991

[10] S. Budkowski, P.Dembinski; "An introduction to ESTELLE: A specification language for distributed systems"; Computer network and ISDN Systems journal, vol. 14, Nb 1, January 1988

[11] ISO-IS 9074; "Estelle: a formal description technique based on extended state transition model"; 1989 (E)

[12] G. v Bochman; "Usage of Protocol Development tools: the result of a survey"; Protocol Specification. Testing and Verification VII, Zurich; CH. H. Rudin (red.), North-Holland, 1987

[13] A.A.F. Loureiro, S.T. Chanson, S.T. Vuong, "FDT Tools For Protocol Development", 5th International Conference on Formal Description Techniques, Lannion, France, Octobre 1992, Tutorials 38-78 .

[14] EDT S. Budkowski, "Estelle development toolset (EDT)", Computer Networks and ISDN Systems 25 (1992) 63-82. North Holland.

[15] R. Groz; "Unrestricted Verification of Protocol Propcrties on Simulation Using an Observer Approach"; Protocol Specification Testing and Verification VII; IFIP 1987, Ed. Sarikaya Bochman (North Holland)

[16] M. Taghelit; "Method and Algorithm for Fault-Tolerant Distributed Systems: Application to Layered Distributed Systems"; Thesis, University Pierre et Marie Curie, December 1991

[17] B. Zouari; "Specification and Verification Methods for Communication Protocol.s"; Thesis of University Paris VI; December 1992.

[18] G. Chiola, C. Dutheillet, G. Franceschinis, S. Haddad, "On Well-formed coloured Nets and their Symbolic Reachability Graph", 11th Int. Conf. on Application and Theory of Petri Nets, Paris, France, Juin 1990.

[19] C. Jard, R. Groz, J .F. Monin; '"VEDA: a software simulator for the validation of protocool specifications"; COMNET' 85, B udapest, October 85

[20] R.E. Miller; "Protocol verification: the first ten years, the next ten years"; PSTV X -IFIP, Ontario June 90

[21] H. Rudin; "Protocol Engineering: a critical assessment"; PSTV VIII; IFIP 1988