30

A New Approach for Protocols Performance Evaluation using Annotated Estelle Specifications*

M. Hendaz, S. Budkowski
Institut National des Télécommunications
9, Rue Charles Fourier 91011 Evry cedex, France.
{hendaz, stan}@hugo.int-evry.fr

Abstract

The traditional approach consisting in using separately FDTs for specification, verification, and validation, and stochastic models for performance evaluation is no longer of great interest in protocols engineering research. A new trend is now growing towards the merging of the two formalisms to provide a complete environment for communication systems development. In this paper we present an approach in which performance analysis is conducted on Estelle specifications with annotated quantitative concepts. The potential power of our approach is then demonstrated by an XTP protocol performance analysis.

Keywords

Formal Description Techniques (FDTs), Estelle, Performance evaluation, Simulation, Protocol Engineering, Xpress Transfer Protocol (XTP).

1 INTRODUCTION

With the traditional approach, the user has to construct two models: one for correctness study (using FDTs) and the other, based on stochastic models, for performance evaluation. He also has to care of the equivalence between the two descriptions since some of the logical properties are often lost when writing the performance model. Although some studies have addressed automatic generation of performance models from FDTs specifications [Dem92], the approach still suffers from the same classical problems.

Recent studies have attempted to combine FDTs with the performance evaluation field to provide complete environment of protocol engineering. This task can be achieved only by "feeding" the classical FDTs or their design environment with the required performance notions.

The paper is organised as follows. First the different performance approaches related to FDTs are reviewed with a special focus on Estelle [ISO89] (Section 2). Section 3 presents a description of some Estelle performance parameters, and Section 4 discusses the performance evaluation of the XTP protocol based directly on its annotated Estelle specification.

* This work was partially supported by CNET #93PE7404 and by CE within the framework of the COPERNICUS #62 project (High Speed Communication System Supporting Multimedia Applications).

G. V. Bochmann et al. (eds.), *Formal Description Techniques VIII*

2 FDTs PERFORMANCE APPROACHES

FDTs suffer from the lack of quantitative features (e.g., time evolution, statistical parametrisation of nondeterminism) necessary to conduct any performance prediction. Among the several studies that have tried to address this problem, two approaches could be distinguished: the FDT's extensions and FDT's annotations. In this paper we will restrict our considerations only for FDT - Estelle.

2.1 Languages Extension Approach

This approach consists in enhancing FDTs specifications by introducing new concepts in the underlying formalism. Performance enhanced FDTs are now available for any of the standardised FDTs [BB91][Wol91]. Beside the fact that such enhanced FDTs should be resubmitted for standardisation process, the main drawback of this approach consist in the necessity to recompile the specification each time a user wish to modify the performance parameters values included in the specification text.

Estelle extensions were first considered in [BV88] where the concepts which principally deal with resources, nondeterminism, and transmission delays have been introduced.

Resources: resource variables could be declared in the specification and allocated during transitions execution for a specified amount of time. This is achieved with the **hold** clause whose simplified form: **hold for** <exec_time> is used to specify transitions execution times.

Nondeterminism: weight clause, **weight** (<real value expression>), could be assigned to transitions in order to define their execution probabilities for selection when simultaneously enabled.These execution probabilities will be dynamically calculated attributing to the each ready-to-fire transitions, at a given moment, the value corresponding to its specified weight divided by the sum of the weights of all the ready-to-fire transitions.

Transmission Times: The declaration of interaction points could be extended to express transmission delays: <interaction point> <properties> **":"** <interaction point type> where <properties> can be of the forms: **delay** <delay> or **fifo delay** <delay>.

2.2 Annotation Approach

Instead of modifying standard FDTs it is possible to annotate the specifications with performance concepts. These annotations can be introduced in static or dynamic manner:

- by means of directives embedded within special comments (also called "qualified" comments) inserted in the FDTs descriptions and kept after compilation (static annotations) or,
- by means of a sequence of commands (macro files) executed during the performance analysis session to set up given values (dynamic annotations).

The main drawback of the first approach is the same as for the case of the language extension i.e., consist in the necessity to recompile the specification each time a user wish to modify the performance parameters values included in the specification text by means of special comments.

This approach has been used in the PEW tool [Whe92][KW92] by offering many performance features such as channel reliabilities (**$R**<*reliability percentage*>), queueing discipline (**$QR**, **$QP**), propagation time (**$D** = <*delay*>), and the implementation of several probability distributions for delays (**$E** Exponential, **$P** Poisson, **$G** Geometric). To address the time evolution process the PEW uses only delay clauses.

The second new approach may be implemented by extending the existing simulators. In our case the Estelle Edb simulator/debugger (a part of the Estelle Development Toolset (EDT) package [Bud92] [1] was extended [HS95].

Thus a number of specific predefined functions has been implemented to access specific object in the simulation environment. These predefined functions can be used within macros or scripts files executed during a performance simulation session. They support a number of performance features that could aid in performance evaluation of communication protocols:

Transition Execution and System Management Times: Functions were defined to specify the time that would take a transition to execute and the time that would take a "system" module (module attributed *systemactivity* or *systemprocess*) to accomplish its system management phase. The first time type could therefore be useful to represent any temporal events or tasks in protocol functionalities like data transmission, propagation, delivery, or other internal data treatments. It also enable to clearly separate two different concepts, that is:

- instantaneous (i.e., not delayed) transitions which could perform timed (or not) actions, and
- delayed transitions executing timed (or not) actions.

The right way of using a delay clause is when specifying delayed actions that might be performed unless some events occur. A delay transition may also be specified in such a way that it could never be disabled in order to represent time distributed actions like the arrival process of service messages.

Since most of protocol temporal events are typically not uniformly distributed, we implemented various distributions that could be used to describe execution time evolution.

The following Edb commands

$exec_time_max (<transition_access>) := <real_value>
$exec_time_min (<transition_access>) := <real_value>
$density (<transition_access>) := <density>
$mean (<transition_access>) := <real_value>

have been added to the Edb environment to define an execution time interval with a specific distribution for a transition identified by its <transition_access> (its name or associated number).

Another time concept , namely system management time, was defined so that it could aid in specifying the speed of a system evaluation phase. This time would be interpreted as the amount of time during which a system module would be passively open to external interaction with the other systems before checking enabling conditions of its transitions and choosing hence the ones that will be actually fired. This could represent either the system load or the host speed and communication time (through a network) in the case of distributed Estelle specifications. Similar predefined functions to those for transitions execution time were implemented:

$sm_time_max (<module_access>) := <real_value>
$sm_time_min (<module_access>) := <real_value>
$density (<module_access>) := <density>
$mean (<module_access>) := <real_value>

The user can also specify the time distribution related to the time interval specified in the delay clause by using **$delay_density** (<transition_access>).

The specification of transitions selection probabilities during execution was integrated by means of **$weight** (<transition_access>) predefined function.

---

[1] EDT includes also an Estelle compiler Ec (translator + C-code generator) and test environment generator Utdg (Universal Test Drivers Generator).

The Simulation Algorithm Revisited: The simulation algorithm presented in [DB87] was enhanced to take into account the introduction of the above new temporal concepts and especially time distributions. The principal modification in the algorithm deals with how to determine the time when the next simulated event should occur. In fact the original algorithm makes an arbitrary choice of this time and then of a subsystem in which an event can complete. Since events could now have different time distributions, the new algorithm begins by determining the event to be executed and then the completion time.

Medium Automatic Generation: It is always useful when executing an Estelle specification to automatically provide the required execution environment. For a communication protocol this environment may represent the user of the protocol service and the provider of the service used by the protocol (upper and lower layers). EDT already supports the automatic generation of the behaviour of protocol users (UTDG), and we then integrated for performance evaluation the automatic generation of used service (medium-type modules). These modules could be of different type depending on the user choice : reliable, with loss, with duplication, and with reordering. Thus, it is possible to specify channel losses, packet-switching networks reordering, and useless retransmissions by messages duplication (with given probabilities).

Edb monitoring environment: During a specification execution, Edb users may control, observe and collect information by means of simulation commands. These commands could be either simple or composed in maco-commands and are used to describe the execution environment and scenarios of a simulation. In fact, users could initialize specification variables, attribute execution times to transitions and management times to system modules. Furthermore, these commands could be included in so-called observers, to detect global properties and gather required information during the simulation. The observers are executed after each transition execution.

3 PERFORMANCE PARAMETERS

The performance measures which could be computed during an Estelle specification execution may be divided into two groups :

- general parameters that are directly related to Estelle constructs, and
- specific ones that depend on the specification under study.

For the first type, interaction points and transitions are good candidates for performance analysis. In fact, observation of the contents of the queues associated to the interaction points may help to detect overloaded channels and thus detect the bottleneck parts of the system. This may help users to correct either specification bugs: an input transition that is never enabled (provided or from clause) or selected (if it belongs to a module whose a parent has always a transition to execute), or temporal attributes which could be wrongly estimated (when the relative speed between the transition outputting messages and the one consuming them is quite large). The following Edb functions were defined to fulfil these requirements:

- **$qsize**(<interaction_point>): returns the number of interactions present in the given queue,
- **$qstat**(<interaction_point>): returns the queue statistics including minimum, maximum and mean number of interactions that transited by the indicated interaction point.

These functions could also be relative to a given interaction type.

Furthermore, **$qsize** function used as an argument of display command could also be compared to enable congestion detection (using an observer commands) and, for example, to drop any further message until the situation becomes normal. Users may also define a maximum size for each input queue that should not be exceeded. This could be achieved by

means of **$qmaxsize**(<interaction_point>) function. This way, the congestion test will be internally performed without having to specify it into observers. Other functions were defined to consult queue contents without dequeueing messages:

- **$qi**(<interaction_point>, <queue_position>) : returns the message name present in the input queue at the given position,
- **$inq**(<interaction_point>, <interaction_type>) : boolean function that returns true if the given message is in the input queue.

Measures related to Estelle transitions could involve two points:

- the total number of transition fired during a simulation session; it may be used to compute other user performance figures. This parameter is obtained for each transition instance by using the predefined function **$nbuse**(<trans_access>),
- the dead time associated to delay transitions which correspond to the amount of time a system is idle before executing a delay transition. This situation happens for example when a retransmission timer value is over-estimated. This information may thus be useful for tuning such temporal values, and is accessed with the **$deadtime**(<trans_access>) function where <trans_access> should identify a delay transition. This function could also be called without argument to return the total dead time.

Concerning the second type of parameters that would be measured during an execution, Edb allows to access specification objects either by an Estelle source name or, in case such a name does not exist (e.g. queue contents), by a predefined function call. Among the first type objects that could be involved in performance computations might be:

- Pascal variables of the "var" section of a module declaration part,
- exported variables of a module instance.

The values of these variables may be stored in scratch pad variables defined by the user during a simulation for his own usage (accessed by their identifiers always preceded by "#"). These scratch pad variables are then used within observer actions to determine the user intended performance results. Two special predefined scratch pad variables were defined (as Edb internal objects) to evaluate two communication protocols concepts, namely, throughput (**#throughput**) and response time (**#resptime**).

4 PERFORMANCE EVALUATION OF XTP

To test the performance enhanced version of Edb, we chose to analyse XTP mechanisms and to evaluate some implementation strategies provided by the protocol. The choice of XTP protocol was quite straightforward since we have an Estelle specification of XTP and a QNAP2 evaluation of the protocol was done. It helped us later to compare the performance results. In our XTP analysis, we have focused on flow control mechanism by analysing and comparing the four strategies proposed in the QNAP2 analysis [Bud94].

4.1 XTP flow control analysis

The XTP flow control mechanism, based on sliding window, is performed by means of two bits in XTP packets, namely SREQ and DREQ. When SREQ is used the receiver replies at once by sending his status in a CNTL packet. Upon the reception of the latter, the XTP sender updates his window variables. The XTP sender controls this procedure by using a timer that is armed whenever the SREQ bit is set. This timer called WTIMER is thus useful to recover from

an XTP packet loss by making both XTP connected entities enter a synchronising handshake that would end if correctly by leaving the global system in a coherent state.

The dependency between the throughput and the control traffic is quite obvious since the bandwidth is consumed by the transmission of control informations instead of user data. This control traffic could be minimized by a wise setting of the SREQ bit and a limitation of handshake as no data exchange is allowed in this phase. The data retransmission could also affect the throughput, but in our analysis we assumed a null packet loss. On the other hand, the control traffic is necessary to manage the sliding window to avoid getting blocked and consequently stopping the data flow. Therefore there is trade-off between control traffic and window management. Thus, the following four SREQ strategies were compared to identify the influence of the different factors presented above:

strategy S1: the SREQ bit is set when the upper bound of the window is reached, so in the last allowed DATA packet.
strategy S2: each DATA packet has the bit SREQ set.
strategy S3: after setting the SREQ bit in the first packet, we repeat this upon the reception of each CNTL packet.
strategy S4: the SREQ setting is performed when the each DATA response is received and the transmission time of the remaining window data is less than the round trip time.

4.2 Parametrisation of the XTP specification

The used XTP Estelle specification corresponds to the English 3.6 revision of the protocol [PEI92], into which we integrated the different SREQ strategies and the dynamic adjustment of WTIMER.

In order to compare our results to those of the XTP queueing model, we conducted the simulation under the same conditions summarised hereafter:

- Application messages have a fixed length of 4 Ko.
- The packet data field has also the fixed length of 4 Ko (no segmentation).
- The construction time of data packet is 0.7 ms.
- The construction time of control packet is 0.25 ms.
- The application submits messages for transmission on average every 0.7 ms.
- The XTP sender buffer is equal to the receiver buffer plus 2 packets.
- The FDDI communication network could have two bandwidth values: 40 and 50 Mbits/s
- The sliding window length varies from 2 to 50 packets.

The XTP specification was annotated using a predefined macro in which these quantitative parameters were assigned to the specified objects, and the required measures were gathered during the simulation.

4.3 Results

Concerning performance measures we focused only on the throughput parameter versus the variation of the sliding window length and the bandwidth offered by the FDDI network. The following curves related to throughput measure were obtained. They are quite identically to those obtained in [Bud94] when the XTP queueing model was used. The strategies S3 and S4 seemed to be better than the others since they involve the window and the round trip time (RTT) values in the SREQ setting. S2 strategy suffers from a too heavy exchange of control information and S1 strategy is affected from the considerable time the sliding window remains blocked especially in the case of small underlying banwidth.

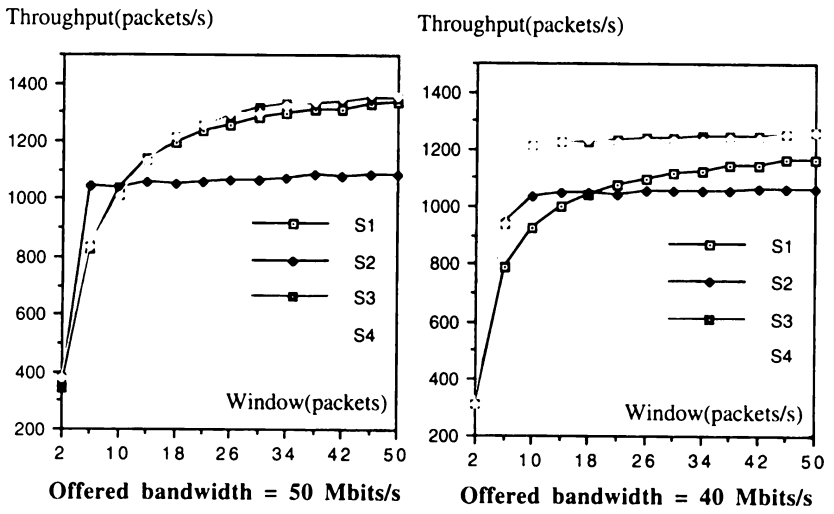

figure 1 XTP Throughput for Estelle simulation

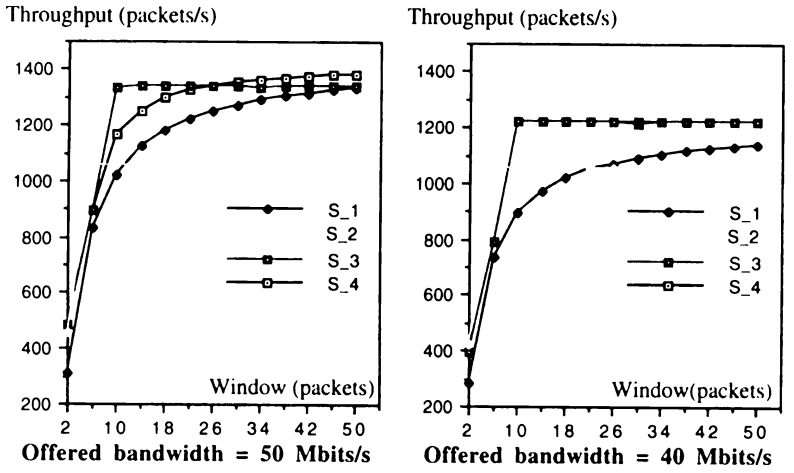

figure 2 XTP Throughput for QNAP2 simulation

5 CONCLUSION

We have presented a dynamic annotation approach for performance evaluation of communication protocols specified in a standard Estelle. The advantage of such an approach is that the same specification is used for validation, verification, and at last for performance study. Thus, the efforts of building two models have been spared and users are sure of the correctness

of the evaluated specifications. We showed that using this approach it is possible to have the same results as by using specialised performance tools like QNAP2. However, the performance evaluation based on FDTs specifications is time consuming since it uses simulation and not analytical solvers. On the other hand, use of such solvers could be only considered for a small subset of FDTs and with only "good" time distributions. The main difficulty of using our approach is find an appropriate mapping between specified objects in Estelle and performance parameters to be measured.

6 REFERENCES

[BB91] F. Bause, P. Buchhols "Protocol Analysis using a timed version of SDL", FORTE III, IFIP North-Holland 1991.

[Bud92] Budkowski S., Estelle Development Toolset (EDT), Computer Networks and ISDN Systems", vol.25, N°.1, 1992.

[Bud94] S. Budkowski, et al. Modelling and analysing the Xpress Transfer Protocol (XTP), The OSI95 Transport Service with Multimedia Support (A. Danthine-Ed.) Springer 1994.

[BV88] G.V. Bochmann, J. Vaucher "Adding Performance aspects to specifications languages", PSTV VIII, IFIP North-Holland 1988.

[Dem92] P. Dembinski, "Queueing Network Model for Estelle", FORTE'92, 83-97, North-Holland, 1992.

[DB87] P. Dembinski, S. Budkowski, "Simulating Estelle specifications with time parameters", PSTV 87, 265-279, North-Holland, 1987.

[HS95] M. Hendaz, S. Budkowski, "Extension d'un Simulateur Estelle pour l'Evaluation de Performance", Rennes, CFIP'95.

[ISO89] Estelle - A Formal Description Technique based on Extended State Transition Model, ISO IS 9074, 1989.

[KW93] G. Wheeler, P.S. Kritzinger, Semi-Markovian Analysis of Protocol Performance, PSTV XIII, Liège 1993.

[PEI92] XTP Protocol Definition Revision 3.6, PEI, Santa Barbara, CA, 1992.

[Whe92] G. Wheeler, "Protocol Engineering from Formal Specifications", Ph.D. Thesis, Dept. of Computer Science. University of Cape Town, 1992.

[Wol91] A. Wolisz, "Timed Interacting Systems for the Performance Enhanced Specification of Communication Protocols", Proc. of SICON'91, Singapore, September 3-6, 1991.

7 BIOGRAPHY

Mohammed Hendaz graduated form the Ecole Natioanle Supérieure d'Inforamtique et Mathématiques Appliquées de Grenoble in 1992. He is currently preparing his Ph.D. thesis on Performance Evaluation of Estelle Specifications in the Software and Networks Department at the Institut National des Télécommunications.

Stanislaw Budkowski received the M.S degree in electronics and the Ph.D. degree in computer science from the Warsaw Technical University, Warsaw, Poland, in 1961 and 1968, respectively. In 1961 he joined the faculty of the Department of Electronics and the Institute of Computer Science of the Warsaw Technical University where he was Associate Professor from 1971. In 1982 he joined l'Agence Informatique (Paris), in 1985 Bull S.A. (Paris) and in 1991 Institut National des Télécommunications where he is actually Professor and Chaireman of the Software and Networks Department. His current work concerns development of methods and software tools supporting communication protocol engieneering.