Q.931 Call Protocol Validation and Performance Assessment, based on LOTOS

E. Vazquez, J. Vinyes, A. Azcorra, M. Alvarez-Campana
Dep. of Telematic Engineering – Technical University of Madrid
ETS Ing. Telecomunicacion, E-28040 Madrid, Spain

Abstract — This paper reports the validation and performance analysis of a Q.931 signalling software, which was developed in the frame of the project Factory Customer Premises Network (FCPN, Esprit P2198). In FCPN, a top-down stepwise refinement approach was used, supported by the LOTOS Formal Description Technique. This approach was based on a sequence of design steps, where each step consisted of three tasks: production of the target design, assessment of the design, and prototyping or implementation.

This paper addresses the second task, i.e. the assessment of the Q.931 protocol specification. The assessment of the functional requirements is based on testing. The main issues involved in the testing task are discussed, and the test suite derived for the Q.931 specification is described. The non-functional requirements are considered using two different approaches: Firstly, the paper reports real performance measures taken in Q.931 protocol prototype implemented with the LOTOS-to-C compiler TOPO. Secondly, it presents an extension of the LOTOS language called LOTOS-TP, which includes simulation oriented features, and discusses its application to the Q.931 specification. The resulting model can be used to simulate the Q.931 protocol and evaluate its performance in various environments.

Introduction

The work reported in this paper describes the validation and performance analysis of a Q.931 signalling software developed in the frame of the project Factory Customer Premises Network (FCPN, ESPRIT-2198). FCPN addresses the problem of Infrared Mobile terminals carried by users through a factory (see [5]). The protocol architecture on the Infrared link is based on an ISDN approach. Following CCITT guidelines, the Q.931 [2, 3] signalling software has been structured in two functional blocks: Call Protocol and Call Control. This approach allows the portability of the Call Protocol to different target machines, leaving in the Call Control the machine-dependent features.

The Q.931 Call Protocol used in FCPN has been developed using LOTOS [7], a formal description technique standardized by ISO in 1988. The development method used for the Q.931 Call Protocol in FCPN reflects ideas developed in the frame of the project Lotosphere (ESPRIT-2304). Our research team has matured those ideas after their application in some prototype developments, including ESA (European Space Agency) and ESPRIT projects. The case reported in this paper reflects a non trivial application of that methodology to a signalling block capable of handling over one hundred simultaneous calls over multiple D channels. Other authors have also used the same methodology for experiences over Q.931. It is worth citing the paper [4] by Ernberg et al. Although they did not arrive to a working implementation, their work is valuable under the methodological point of view.

This paper addresses one particular task of the design process: the assessment of the functional requirements and of the performance requirements. Other tasks of the complete design process have been presented in other papers (e.g. [1] describes the implementation task). The assessment of the functional requirements is based on testing. Non-functional requirements are assessed with two different approaches: Firstly, using real performance measures taken from a prototype of the Q.931 protocol implemented with the LOTOS-to-C compiler TOPO[8, 9]. Secondly, using a Q.931 simulation model based on an extension of LOTOS called LOTOS-TP, which includes simulation oriented features.

0-7803-1772-6/94/$3.00

Signalling Protocol Validation

The Validation of System Designs with LOTOS

With our methodology, in each design step an *assessment* task is performed to achieve confidence in the correctness of the design and/or detect errors. This task must assess that the design produced in the previous task is consistent with the system requirements. In a general design case, the assessment can be separated into two subtasks:

> Subtask 2.1 assesses the consistency of the design with the requirements considered relevant at the current design step, in order to validate the design decisions made.
>
> Subtask 2.2 assesses the consistency of the current design with all the previous design steps.

The first subtask has to compare an non-formal input –the requirements– with a formal one –the formal description of the design–. Such a comparison can only be done by human interpretation of the non-formal requirements. The interpretation of the requirements must be formalized and compared to the design. With the present state of the art, the only practical approach is based on testing, and has two parts:

1. *Generation of the test suite*, where tests are formalized as LOTOS processes. This part has to be done by hand.
2. *Execution of the test suite* using automatic tools and procedures as much as possible.

Subtask 2.2 must verify the equivalence relations that guarantee the consistence between successive refinements. For example, the verification of the testing equivalence between two system refinements consists in proving that they have the same response to external tests. This task may be accomplished automatically. However, the verification algorithms are very complex and cause state explosion even for simple specifications, so in practice the testing equivalence is verified partially, by computing the responses to a limited number of tests.

As the previous subtask, subtask 2.2 is also divided into two parts, test suite generation and test suite execution. However, in this case one of the specifications can be processed with a test generation tool to generate the test suite in a (semi)automatic way. Then, the resulting test suite is executed on the other specification.

Q.931 Protocol Validation

The design process followed to obtain the Q.931 specification consists of two refinement steps: *service specification* and *protocol specification*. The first step specifies the service that must be provided by the Q.931 protocol according to the (non-formal) definition given by the CCITT Recommendation. The second step specifies a protocol entity that is able to provide that service. These two steps are described in more detail below.

Service Specification: The Q.931 service is defined in terms of the *possible sequences of signalling messages* that may be observed at the user-network interface according to the Q.931 Recommendation. These messages are carried in layer 2 data primitives. Therefore, these are the primitives relevant to our service specification.

As shown in figure 1, the layer 2 service access points are the only visible interface in the service specification. The layer 3 internal structure, for example Q.931 entities, internal interfaces, local resources, etc is irrelevant at this point of the design process.

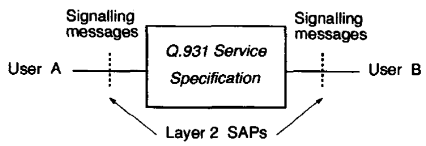

Figure 1: Q.931 Service Specification

The service specification has been tested in several call establishment and release scenarios. The testing method used is based on the semantics of the parallel composition of LOTOS, and is supported by a tool called LOLA[12, 13] (LOtos LAboratory). With this method, each test scenario is itself a certain signalling message sequence specified in LOTOS, which is composed in parallel with the service description in LOTOS to test if the sequence is accepted or not. The tests are automatically executed in batch: an auxiliary tool is used to run all the test cases through LOLA and produce a file summarizing the results.

Protocol Specification: Once the service specification has been tested, the next refinement step is the specification of the protocol. In our case, this specification defines the behavior of a Q.931 protocol entity, and has two visible interfaces: the layer 2 interface, as in the service specification, and the layer 3 interface. Considering the two assessment subtasks presented above, two different test configurations are needed. Firstly, the protocol specification has to be consistent with the service specification. This can be assessed with a test configuration where two Q.931 protocol entities are composed with a network emulation module that interconnects their upper interfaces. The resulting system, shown in figure 2, must have the same observable behavior as the service specification, so it must pass the same tests.

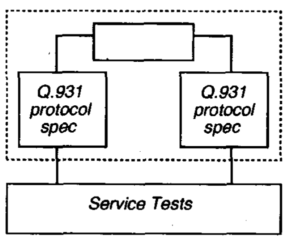

Figure 2: Composition of Q.931 Protocol Specifications

In the second configuration, a Q.931 protocol entity is tested separately, observing the signalling messages exchanged through the lower interface, as in the service specification tests, *and* the service primitives exchanged through the upper interface. This configuration serves to test if the requirements related to the layer 3 service interface, which were not considered in the service specification, are met by the protocol specification.

The Q.931 Test Suite

Generation of the Q.931 Test Suite: The generation of a test suite for a protocol specification such as the Q.931 one, which has complex interfaces consisting of many different messages and service primitives, may be a rather cumbersome task. Consequently, one of the main objectives in the design of the Q.931 test suite was to build it in a modular way, so that complex test sequences could be easily created by combining a relatively small number of basic "test modules". Each test module is a LOTOS process which can be combined with other processes, by using the appropriate LOTOS parallel and sequential composition operators, in order to obtain a meaningful test. This approach can be applied recursively in order to obtain tests of increasing complexity. For example, several instances of a call setup test process can be composed in interleaving to obtain a multi-call test process.

The first step followed to obtain the simplest test processes is to find the relevant states of the protocol and identify the possible transitions among them. The Q.931 protocol specification had 15 relevant states (e.g. call initiated, call received, active, etc).

The second step is to identify the externally observable events associated to the transitions identified in the previous step. In general, each transition of the Q.931 protocol specification involves two events: the transmission or reception of a service primitive in the layer 3 interface, and the transmission or reception of the corresponding signalling message via a service primitive in the layer 2 interface.

Each possible pair of primitives is encapsulated in a LOTOS process, which terminates with a LOTOS *exit*, so that it can be sequentially composed with other processes by using the LOTOS *enabling* operator. The service primitive parameters that may change from one test to another, e.g. call reference, called number, etc, are included as parameters in the process header.

The final test bed is made of different combinations of 47 elementary test processes, giving a total of 75 single call tests, and 48 multi-call tests. The purpose of the single call tests is to study the specification behavior in the different situations that may arise during the setup and release of a single call as thoroughly as possible. The multi-call tests study aspects related with the management of several consecutive or concurrent calls, for example the selection of call references. It assumes that the specification behavior for a single call has been previously tested, so it considers combinations of typical calls only.

Execution of the Q.931 Test Suite: The Q.931 protocol test suite obtained with the procedure summarized above has been executed with a LOTOS-to-C compiler called TOPO. The procedure to execute a test with a compiler such as TOPO is similar to the one used with a state exploration tool such as LOLA: the test takes the form of a LOTOS process which is composed in parallel with the behavior under test. With LOLA, the composition of the behavior and the test can be exhaustively analyzed to determine if the desired conditions are met in all cases. For example, the test result provided by LOLA may prove that the sequence of primitives represented by the test is *always* accepted without blocking.

With TOPO, however, the composition of the behavior and the test is translated to C, compiled and executed. Since the specification may be non-deterministic, the fact that *one* execution ends without blocking does not necessarily guarantee that the specification *never* blocks in the situation represented by the considered test. TOPO, therefore, does not provide the definite test results given by LOLA, but it is much faster, and may even be the only practical alternative when the specification is too complex to be expanded with LOLA. When TOPO is used, each test can be executed several times in order to increase the probability of detecting incorrect behaviors.

Since TOPO is not primarily designed for testing purposes, it is necessary to make some changes in the specification before compiling and executing the tests. The specifications processed by TOPO must have a *noexit* behavior, and cannot have external parameters or gates. To achieve this, the behavior under test is encapsulated in a LOTOS process which is composed in parallel with the tests, hiding the synchronization events between the the two parts.

Furthermore, the behavior part of the resulting specification is structured as a *choice* where each branch consists of an internal event followed by one test process composed in full synchronization with the tested process. In this way, the complete set of test processes and the behavior under test are included in a single file which is compiled only once. The selection of a particular test is done at execution time by means of the *wait annotation* supported by TOPO, which allows the selection of any of the choice branches depending on the value of an integer parameter.

Q.931 PERFORMANCE ASSESSMENT

Prototype Performance Measurement

In order to obtain a first estimation of the Q.931 protocol performance, the Q.931 formal specification has been implemented in a UNIX system with the LOTOS-to-C compiler TOPO. The results of real measures taken in this environment are summarized below.

To obtain the prototype Q.931 implementation, the Q.931 specification (3800 lines) together with the complete test suite described in the previous section (6400 lines) is translated from LOTOS to C with the TOPO compiler. The C files generated by TOPO are compiled, and then linked with the appropriate library functions, giving an executable file of 1073 Kbytes. (If the test suite is not included in the specification, the size of the resulting executable file is 417 Kbytes.)

The resulting Q.931 implementation is executed as a user process in a diskless Sun SPARCstation connected via Ethernet to a file server and running SunOS release 4.0.3. The measures presented below correspond to the CPU time consumed by the Q.931 process during the execution of several test cases selected from the complete test suite. (The CPU times are obtained with the *time* operating system command.)

The selected tests provide a rough estimation of the *maximum number of signalling messages per CPU second* that the Q.931 implementation is able to process, in the following cases:

1. The user side initiates a call which is immediately rejected by the network side, repeating the whole cycle n times.
2. The user side initiates a call and clears it as soon as it has been established, repeating the whole cycle n times.
3. The user side establishes n calls in parallel and then clears them.

Each test case has executed 5 times with n=50 calls. For each test case, table 1 lists the number of signalling messages per call exchanged in each direction (from User to Network and vice versa), the average and maximum CPU times consumed to process the 50 calls, and the average number of messages processed per CPU second. These time values correspond to the "user" time provided by the UNIX *time* command. The "system" time was in all cases less than 5 % of the user time.

Table 1: Q.931 Performance Results

test	mess. per call		CPU sec.		mess.
case	U→N	N→U	Aver.	Max.	per sec.
1	1	1	5.04	5.1	19.84
2	3	4	13.92	14.3	25.14
3	3	4	36.56	37.1	9.57

Performance Simulation with LOTOS

The Q.931 formal specification has been used as the basis to obtain a simulation model suitable for evaluating the Q.931 protocol performance in different implementation environments. This model makes use of the temporal and probabilistic description provided by LOTOS-TP, an extension of the standard LOTOS language designed for performance simulation. LOTOS-TP [10] was developed to support the evaluation of performance of systems that are being designed with LOTOS. This allows the inclusion of non-functional requirements in the design process. LOTOS-TP includes two main extensions with respect to LOTOS: the quantitative description of time, and the probabilistic characterization of behaviors. The additional refinement steps required to obtain the Q.931 performance model in LOTOS-TP from the standard LOTOS specification are supported by TOPOSIM, a simulation tool based on TOPO, the LOTOS-to-C compiler used during the protocol validation phase.

Quantitative Description of Time: The ability to describe the progression of time in a quantitative way, which LOTOS lacked, is essential for most performance evaluation applications. In our case, this is necessary to define the time consumed by the Q.931 protocol while it is performing internal actions, as well as characteristics of the protocol environment, for example, the time required to send a signalling message through the D channel in the user-network interface, the delay from the instant when the user side receives a particular message to the instant when it generates the corresponding response message, etc.

The semantic model followed to include quantitative time in LOTOS-TP is based in the "As Soon As Possible" (ASAP) criterion. With the ASAP semantics, the synchronization among several processes that offer the same action takes place immediately. The time domain is not predefined. It is defined by a set of values (a *sort*) and operations included in the specification.

Probabilistic Description of Behaviors: The ability to describe a system in a probabilistic way allows the specifier to abstract the real behavior of a given system component by expressing it in terms of probabilities. In our case this may serve to represent the user behavior, for example the probability of acceptance or rejection of an incoming call in the called user interface. A probabilistic description is also useful to model real phenomena that involve random values, for example the duration of a call, the time interval between successive calls, etc.

Simulation Facilities: In addition to the two basic features summarized above, LOTOS-TP provides facilities to describe the system parameters that must be measured and the statistics that will be computed during the simulation. Other aspects, such as the duration of the simulation period, the initial transient interval, and the values of variable parameters have not been explicitly considered in the language, but they can be easily defined with commands of the simulation tool.

Simulation Methodology: To study the performance of a protocol with LOTOS-TP, the user has to specify a complete system composed of three parts:

1. The protocol entities.
2. A model of the protocol environment, for example the protocol users and the underlying communication facilities
3. The measurement processes required to obtain the desired performance metrics

The model of the environment represents the non-functional requirements that the protocol design must meet, for example the traffic load generated by the protocol users, the throughput provided by the communication channel used, etc. In general, many of these values will be considered as simulation parameters which will be set when the simulation is executed.

Concerning the measurement processes, there must be one of them for each performance metric considered. Each metric is treated as an stochastic process. During the simulation, the associated measurement process will provide the sequence of values taken by the process and the time intervals between consecutive values. The measurement processes must be included in the specification in such a way that they have access to the parts of the system where the values of the measured stochastic processes are generated. They must obtain the stochastic process values and pass them outside the specification without modifying the behavior of the studied protocols.

Finally, the user must include a header at the beginning of the specification indicating the statistics that must be calculated from the values supplied by the measurement process. The simulation tool creates one observer process that synchronizes in the external gates of the specification and computes the required statistics.

The TOPOSIM Tool: TOPOSIM [11] is a simulation tool based on the LOTOS-to-C compiler TOPO that can be used to evaluate the performance of a system specified in LOTOS-TP. TOPOSIM generates a set of C modules that can be compiled and linked to obtain an executable program. This program recognizes a number of command line options that serve to specify the duration of the simulation period and, optionally, the duration of the initial transient period, as well as the simulation parameter values. Additionally, the simulation program can generate a trace that shows the interactions between the specification and the observer process created by the tool.

The Q.931 Model in LOTOS-TP

Starting from the Q.931 specification in LOTOS, we have specified in LOTOS-TP a system composed of two Q.931 protocol entities, one corresponding to the user side and the other to the network side, communicating over a D channel by means of the service primitives provided by the LAP-D protocol. (See figure 3.) Three steps have been followed to obtain this specification:

1. Specification Changes: The Q.931 specification has to be modified in order to include the LOTOS-TP extensions and to adapt it to the TOPOSIM tool. All Q.931 timers, which were implemented with C annotations (supported by the TOPO compiler) due to the temporal limitations of LOTOS, were correctly specified in LOTOS-TP.

2. Specification of the Q.931 Environment: The Q.931 environment consists of a process representing a user terminal in the user side, a process representing the call service module in the network side, and, finally, a process representing the LAP-D service.

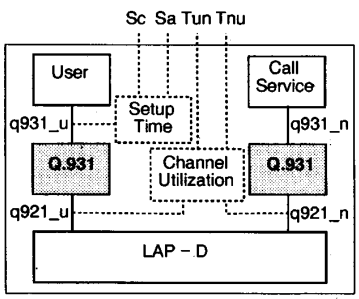

Figure 3: Complete Q.931 Specification

The process that models the LAP-D service is very simple and considers the data transfer primitives only. The transfer delay between the data request and the data indication primitives depends on the signalling channel rate and the length of the different Q.931 messages (including the LAP-D header). The other two processes, user terminal and call service, are more complex, because they must be able to handle the different sequences of layer 3 service primitives that may take place during a call. The approach followed to specify them makes use of the Q.931 validation test suite presented in the previous section. The specification of the user and call service environment processes has been obtained by eliminating the layer 2 service primitives of the independent test cases used to validate the protocol and merging them in a single behavior tree, using alternative operators where necessary, so that any valid layer 3 service primitive sequence is accepted. Once the complete behavior tree has been obtained, the different alternatives of a typical call (successfully established, called user busy, etc) can be weighted with a selection probability by using the facilities of LOTOS-TP. The behavior of these modules for several consecutive or concurrent calls can be obtained by using recursive process instantiation and the parallel composition operators provided by LOTOS.

3. Definition of Measurement Processes: Two measurement processes were included in the Q.931 system (see figure 3). The first one measures the delay between the initial SETUP message and the CONNECT message (gate "Sc"), and the delay between the SETUP and the first end-to-end response message, i.e. ALERTING or CONNECT (gate "Sa"). The second one gives the transmission channel utilization in the user – network direction (gate "Tun"), and in the network – user direction (gate "Tnu").

Simulation Results

Table 2 shows an example of the results obtained from the measurements taken at gate "Sa", assuming a traffic load of 10 Erlang and a average call duration of 180 seconds.

Table 2: Q.931 Simulation Result Example

Setup time (secs)			Channel utiliz.	
Min.	Aver.	Max.	U → N	N → U
0.132	0.429	0.804	0.954%	1.070%

In this example, 60% of the calls were assumed to be successfully completed. The remaining 40% corresponds to four different categories of unsuccessful calls, each of them with its own probability. These probabilities and other experiment details, such as signalling message lengths, signalling message sequences, delays before receiving the user responses, etc, are omitted for brevity, but can be found in [6]. The important point is that these parameters can be easily tuned using the features of LOTOS-TP and TOPOSIM, in order to model different real Q.931 environments accurately.

Conclusions

The work reported in this paper has produced three practical results. Firstly, a Q.931 Call Protocol LOTOS specification has been validated with the testing approach described in the paper. Secondly, we have carried out performance measurements on a prototype implementation of the Q.931 Call Protocol running in a UNIX environment. Finally, we have developed a Q.931 simulation model in LOTOS-TP that can be used to estimate the performance of the Q.931 Call Protocol given the hardware/software characteristics of other execution environments.

Besides these practical results, an important output of the work has been the production of a *more refined LOTOS methodology* and an *evaluation* of it under an industrial point of view. The evaluation of the formal design and assessment tasks is very positive. Expressing the tests in LOTOS and running them with LOTOS tools against the *specification* proved better than the conventional approach of coding tests in either C or a special purpose language and running them against the *implementation*, when the design is much more difficult to change. The performance simulation model was obtained from the LOTOS specification by using the simulation extensions provided by LOTOS-TP. Even though the TOPOSIM tool had some limitations, this approach reduced the effort required to develop and validate the simulator.

References

[1] A. Azcorra, E. Vázquez, M. Alvarez-Campana, and J. Vinyes. "Formal Description Techniques at Work: an ISDN Implementation of Q.931 Using LOTOS". In A. Danthine, et al, editors, *Protocol Specification, Testing and Verification, PSTV XIII*, pp. 175-190, Liege, Belgium, May 1993. IFIP WG6.1, North-Holland.

[2] CCITT. *ISDN User-Network Interface Layer 3 Specification for Basic Call Control*. Fascicle VI.11 Q.931, Blue Book, 1989.

[3] CNET. *ISDN VN2 Signalling Procedures*. Technical Report LAA/RSM/133, Lannion, 1986.

[4] P. Ernberg, T. Hovander, and F. Monfort. "Specification and implementation of an ISDN telephone system using LOTOS". In M. Diaz and R. Groz, editors, *Formal Description Techniques, FORTE V*, pp. 171-186, Perros-Guirec, France, October 1992. IFIP Transactions C2, North-Holland.

[5] FCPN. *Logical link control and signalling protocol definition*. Deliverable 3.3, FCPN, July 1991.

[6] FCPN. *Signalling protocol validation and performance analysis*. Deliverable 11.5, FCPN, October 1991.

[7] ISO. *LOTOS a Formal Description Technique based on the Temporal Ordering of Observational Behavior*. IS 8807, 1989.

[8] J.A. Mañas, T. de Miguel, T. Robles, J. Salvachúa, and G. Huecas. *TOPO: Quick Reference*. Technical Report V3.0, DIT-UPM, Madrid, Spain, April 1991.

[9] J.A. Mañas, T. de Miguel, J. Salvachúa, and A. Azcorra. "Tool Support to Implement LOTOS Formal Specifications". In *Computer Networks and ISDN Systems, 25(7), February 1993.*

[10] C. Miguel. *Técnicas de descripción formal aplicadas a la evaluación de prestaciones de sistemas de comunicación*. PhD thesis, ETS Ing. Telecomunicación, UPM, Madrid, March 1991.

[11] C. Miguel, A. Fernández, J.M. Ortuño, and L. Vidaller. "A LOTOS Based Performance Evaluation Tool". In *Computer Networks and ISDN Systems, 25(7), February 1993.*

[12] S. Pavón and M. Llamas. "The Testing Functionalities of LOLA". In J. Quemada, J.A. Mañas, and E. Vázquez, editors, *Formal Description Techniques, FORTE III*, pp. 559-562, Madrid, Spain, November 1990. IFIP, North-Holland.

[13] J. Quemada, S. Pavón, and A. Fernández. "State Exploration by Transformation with LOLA". In *Workshop on Automatic Verification Methods for Finite State Systems*, Grenoble, June 1989.