# Towards an unified experimentation framework for protocol engineering

Laurent Dairaine, Ernesto Exposito, Hervé Thalmensy

▶ **To cite this version:**

Laurent Dairaine, Ernesto Exposito, Hervé Thalmensy. Towards an unified experimentation framework for protocol engineering. The IEEE 20th International Conference on Advanced Information Networking and Applications, Workshop on Service Oriented Architectures in Converging Networked Environments (SOCNE06), Apr 2006, Vienne, Austria. pp.555-559. hal-04037001

**HAL Id: hal-04037001**
**https://hal.science/hal-04037001**

Submitted on 20 Mar 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Towards an Unified Experimentation Framework for Protocol Engineering

Laurent Dairaine, Ernesto Exposito, Hervé Thalmensy
ENSICA/LAAS-CNRS
Applied Mathematics and Computer Science Department
1, Place Emile Blouin, 31000 Toulouse, France
Laurent.Dairaine@ensica.fr

## Abstract

*The design and development process of complex systems require an adequate methodology and efficient instrumental support in order to early detect and correct anomalies in the functional and non-functional properties of the solution. In this article, an Unified Experimentation Framework (UEF) providing experimentation facilities at both design and development stages is introduced. This UEF provides a mean to achieve experiment in both simulation mode with UML2 models of the designed protocol and emulation mode using real protocol implementation. A practical use case of the experimentation framework is illustrated in the context of satellite environment.*

## 1. Introduction

Designing and developing communication protocols and real-time systems is a complex and long process. During the initial phase dedicated to the analysis, the modelers try to represent and describe the current state of the studied system thus creating a model of the reality aimed at detecting and understanding the problem to be solved. From this phase, the designers use this model to start proposing and designing solutions to the problem. These solutions will be generally specified using high level languages in a high abstraction level context. Different resulting specifications will need to be evaluated in order to study their efficiency and that they are not disturbing the normal behavior of the system. During this evaluation phase, not only the functional properties of the solutions have to be studied, but also the performance or the Quality of Service (QoS) provided. The following phase consists in the implementation of the solution and its integration in the real system. However, before deploying the solution in the real system, an evaluation process has to be carried out again, in order to verify that both functional and non-functional properties have been respected during the implementation.

In this article, an Unified Experimentation Framework (UEF) for Protocol Conception and Implementation is presented. This framework is intended to provide sufficient support to the designer with the experimentation phases in both design and development phases. The framework is designed to allow an accurate specification of the packet impairments and, thanks to object oriented approach, to allow extension to be achieved in such a way to provide high level modelisation tools for representing arbitrary complex networking scenario. All experimentation models are designed using UML2. As study case, the experimentation framework is used for designing an experimentation environment able to evaluate end-to-end protocols in the context of a satellite environment providing QoS guarantees.

This paper presents the following structure: the first section presents a background providing the basis to this platform and briefly describes the various components of the *x*QoS platform. Section 3 provides details on the UEF, presenting the principle and its logical components. Section 4 proposes a particular study case describing how to use the UEF to build a communication platform able to experiment with QoS GEO satellite link. Finally, section 5 proposes few conclusions.

## 2 Background

Various methods and languages have been proposed to optimize the design and development process[7]. We consider that an effort should be done in order to connect these two worlds. The Unified Development Process methodology (UDP) [2, 5] and the Unified Modeling Language (UML)[6] have been selected in the framework of our *x*QoS platform [3] for being considered as an adequate and efficient methodology and tool for designing and developing complex systems.

In this context, the conception process should be punctuated by many experimentations. This is needed to evaluate both functional and non functional properties of the target protocol. Ad-hoc experimentation environment should be

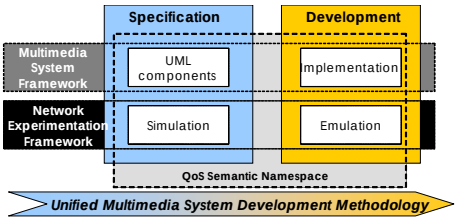


**Figure 1. The *x*QoS System component**

provided to properly achieve this task. The classical ways to achieve these experimentations are currently used: simulation (e.g., event-driven), live testing (not considered in this paper) and emulation (production of realtime controlled communication behavior, see e.g., [8, 1, 11, 10]). Network emulation is used achieve experiments using both real protocol implementation and network models allowing to create a controlled communication environment.

Simulation and emulation experimentation approaches can be considered as complementary because they allow providing a way to tackle different problems during the protocol conception phases. Simulation is used at early stage to evaluate various solutions while emulation is used at implementation stage for achieving experiments with real codes. The *x*QoS platform proposes a design and development platform integrated in a specialized methodology intended to model, specify, test and deploy real-time applications and advanced communication services [3]. Figure 1 illustrates the different components of this platform including a semantic namespace, models for multimedia applications and test-beds for high level specifications (simulator) as well as for solution implementations (emulator).

The semantic namespace (language) defines the QoS of the system from the user and service provider viewpoints. This semantic namespace will be particularly used to validate the non-functional properties of the solution during the specification and implementation evaluation phases. The *x*QoS platform includes design patterns providing structural and behavioral models for legacy and new real-time applications in order to accurately express the requirements to be satisfied by the communication system. These patterns are specified in UML and are intended to be used for designing the solutions. Finally, the platform proposes an Unified Experiment Framework (UEF) integrating two components: a simulator intended to be used to evaluate the solution specifications and an emulator suited to evaluate the corresponding implementations. Both components are suited to evaluate the functional and non-functional properties of the solution. This is achieved using the structural and behavioral description provided by UML for both, the specification and the implementation, as well as the QoS properties described using the *x*QoS language. These QoS properties are evaluated using different scenarios where the channels connecting the distributed components are modified in order to provide imperfect communication and study its consequences in the evaluated solutions.

The rest of this paper focuses on the UEF.

## 3 The Unified Experiment Framework

Experimentation using simulation and emulation are different means to test both functional and non-functional properties of developed protocols. Considering the non-functional properties, the UEF should provide a way to introduce controlled communication impairments. Possible impairments are mainly delay, packet losses and packet modification. Another important issue about the UEF is to provide a mean to produce transparently a controlled behavior in such a way to test and stress the experimental protocol. This leads to end-to-end QoS channel that could focus on both two main objectives. The first one is to produce an *Artificial QoS* in such a way to evaluate the protocol over specific QoS conditions, not imperatively related to any technology. This processing allows the user to test and stress its experimental protocol in a reproducible target QoS conditions, aiming to point out errors or bugs that could be difficult to produce in a non-controlled environment. This can be useful for example at transport level to study the impact of various packet drops in a TCP connection (e.g., the SYN/ACK, etc.) or at application level what happened at if a particular block of ”Intra” picture is delayed. A second objective is to achieve a *realistic QoS* in such a way to reproduce as more accurately as possible, the behavior of specific network architecture. This type of experiment allows the user to evaluate the protocol over an existing network or internetwork without using a real testbed with all related technologies (e.g. a wireless network, a satellite network, and xDSL link, or any interconnection of such technologies).

The UEF is based on the experimentation channel (EChannel) component offering a target QoS to the System under Test (SuT). It is defined as a data path providing to the SuT particular QoS impairments. The EChannel implements the impairment following the requirements described into the previous paragraph. A possible representation of this model is pictured into the following composite structure diagram implementing a SuT instantiated by a source and a receiver, using an emulation channel to communicate between each protocol entity.

The Experiment Channel should provide the final target behavior in terms of QoS for the experimentation. This resulting system may implement a very simple behavior such a constant end-to-end delay or a more complicated scenario such as the behavior of a end-to-end path constituted by a various underlying network technologies. To allow such

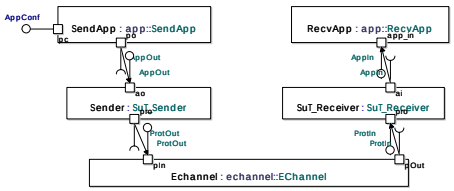


**Figure 2. UEF components**

an arbitrary complexity to be implemented, the EChannel behavior results from the composition of Experimentation Nodes (ENodes). An example of EChannel that models a satellite link is provided in figure 3. Each ENode is an active component that offers the necessary ports to achieve the internal communication (pOut and pIn). The nodes are individually parameterized using the pConf port. The pSpy port may be used to give information about the ongoing processed traffic to an external management module. The InputTap is a special ENode intends to capture the traffic coming from the SuT and to prepare it to be processed by the set of EChannel components (e.g., it adds useful fields such as capture timestamps, length, etc.). At the end of the EChannel, the OutputTap get rid of all these working fields, then providing the necessary transparency to the experiment traffic.

The Experiment Node is designed into two main parts to differentiate the actual impairments to achieve and the controlling process: the experiment processor is really impairing the packets while the experiment model is deciding how the packet must be processed. The former component provides the various impairments actions defined in the top of the section (i.e., delaying, dropping, packet modifying). The experiment model has access to various informations (i.e., length of packet, capture time, internal packet fields, external ports, etc.) and provides an abstract representation aiming at specifying actions to be taken on packets. An experiment channel processing can be defined either statically or can evolve dynamically during the experiment. Classically, a statechart is used to achieve this specification.

Two main types of models are defined, namely passive and active models. Passive models act on packet events according only to external packet properties such as time it reach the experiment node or its length. Example of classical passive models implemented are e.g., delay or bandwidth. Active models can react on a larger set of stimuli. Those stimuli can be time driven or packet driven scenario, model based on value of the data contained into the processed packets, or even any other external signals such as measurement traces realized on real or simulated networks. An example of such function could be the loss of a specific packet according to a particular history of the past processed packets (e.g., a finite state machine associated to the processed flow). The loss signal could also be a particular packet multiplexed into the experiment flow (e.g., to ensure synchronization between the required impairment and the processed packets) or using an external port. The various schema provided by UML2 are particularly adapted to the specification of such models. All these models (both active and passive) can be composed together to obtain a more complex behavior. ENodes are proposed as an extensible library that intends to provide various types of experimentation processing. The end-user can then compose the channel depending on experiment objectives.

The UEF provides two tools: a simulator intended to be used to evaluate the SuT Design solution and an emulator suited to evaluate the real performance of the SuT implementation. The Telelogic UML tool[9] has been selected to provide the specification and simulation functionalities in the *x*QoS platform. As it is demonstrated in the following section, the UEF has been also implemented in order to test real implementation in a same way than they can be tested at design phase with the Telelogic tool.

## 4 Use-case: a GEO QoS Satellite link

### 4.1 The QSat Access Network

The context of this use-case is a QoS GEO satellite link. This link is interesting to experiment due to the particular QoS services it offers and the difficulty to access to a real satellite system offering such service guarantees. The targeted satellite link uses widely spread DVB technologies namely: DVB-S standard on the Forward link (from the Hub Station toward the Satellite Terminal) and DVB-RCS standard on the Return link (from the Satellite Terminal toward the Hub Station) [2]. Due to the high cost of satellite resources, experimentations involving this particular access network[1] will be done on an emulated satellite link. In order to develop this emulation platform, the characteristics of the satellite link are firstly presented.

The main issue in the satellite communication context, and more particularly on the satellite return links, is to make an efficient use of the transmission resources. Recent techniques based on dynamic bandwidth assignment, enable a high efficiency of the return link usage. Emerging protocols, such as DAMA (Demand Allocation Multiple Access) as the MAC access scheme combines efficiency (a high utilization of the return link resources) and QoS Guarantees. This access scheme is targeted for the satellite access network experimentation instance presented in the case-study.

The DVB-RCS system[2] is based on the following principle: A Return Channel Satellite Terminal (RCST) com-

[1]The satellite access network is the only technology which will be emulated into the EuQoS project. All other access networks are implemented using real technologies.

municates with the Network Control Center (NCC) in order to request capacity allocations. The resources allocation plan (i.e. the Terminal Burst Time Plan (TBTP)) is then sending to RCSTs using MPEG-2 Transmission Stream (MPEG-2 TS) cells containing IP packets. All user terminal transmissions are controlled by the NCC. Before sending data, a terminal must join the network by communicating (logon) with the NCC describing its configuration. The satellite terminal can then use the reserved slot in the superframe to send data in its buffers. Dynamic resource control consists in assignment of resources slots to terminals which are conditioned by the availability of resources (capacity). The assignment is done by the MAC Scheduler (in the NCC) which implements a DAMA protocol.

The SLA defined at logon between the Terminal and the Hub, specifies guarantees on different classes of access named capacity assignment to the Return Link of the satellite, such as Constant Rate Assignment (CRA), Rate Based Dynamic Capacity (RBDC), Volume Based Dynamic Capacity (VBDC) which can be guaranteed or not and Free Capacity Assignment (FCA). We will particularly focus on delay associated to CRA and Guaranteed-VBDC. The CRA is statically programmed in the hub and requires no dynamic signalling from the ST. This capacity is guaranteed. If the connection set-up is admitted, the terminal will benefit of its requested bandwidth for the duration of the connection, so that the traffic is not subject to any scheduling/queuing delay. The CRA is a rate capacity fully provided for the duration of the connection without any DAMA request, therefore, the associated network delay is the geostationary delay, $\approx$300ms. The VBDC requires explicit requests from the ST to get a capacity. A ST must signal its request in terms of total number of slots required to empty its local queue. These requests can be supplemented by new requests any time more traffic is queued. The scheduler accumulates all requests from each terminal. The total capacity of the queued requests is decreased as soon as slots are granted. Practically, the terminal must do a capacity request to the NCC based on the volume of data to transmit in its buffers. Then the NCC allocates some time slots. In VBDC, the delay will depends on three parameters: latency to send a capacity request, minimum latency scheduling to receive the TBTP (congestion dependent) and geostationary RTT. A typical delay will then vary between 1400ms and 2100ms or even more in case of severe congestion.

## 4.2 The QSat Experiment Channel

We implement a channel reproducing the delay associated to the satellite link. The Forward link is proposed as a simple EC integrating a constant delay (e.g., 300ms) and throughput shaper ENodes. The limitation on the throughput depends on the contract used on the Return Link.

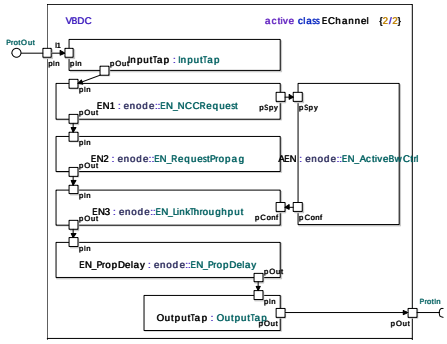


**Figure 3. The VBDC experiment channel**

The QSat-EC presented in Figure 3 implements the return link. A Decision-making ENode has to dynamically compute the bandwidth limitation, delay and packet loss rate values to apply to the set of ENodes which actually impairs the flows, according to the actual QoS Contract as well as information concerning the ongoing network traffic. The decision-making ENode will then initially sets initial impairments and modifies dynamically these values in real-time, in order to produce the delay associated to the QoS signalling process. Then, the signalling protocol is not really implemented, but only its resulting effects on the traffic are taken into account. The figure 3, an Echannel implementing a VBDC traffic class is implemented using four passive and one active Enodes nodes.

Then packets cross the channel, each of constituting Enodes producing a particular aspect of the VBDC delay link behavior. The first ENode models the time interval between two VBDC requests from the Satellite Terminal toward the NCC. It is implemented as a circular delay periodically decreasing between 500 and 0ms. The second ENode corresponds to the delay for the request to reach the NCC, to be processed and to come back to the ST. The value is $\approx$700ms, $2* \approx$300ms for satellite link, and at least 100 ms is required to process the requests but this time can also increase. The third ENode implements the rate at which the Satellite Terminal is actually sending data on the link. This variable throughput depends on realtime traffic measurements achieved by the Active ENode (AEN). The AEN spies the traffic packets and dynamically compute the actual input rate. The computed rate will be applied taking into account the QoS signalization latency. Then, the traffic will be delayed, and if the traffic is too important for the satellite link capacity, losses will be encountered due to buffer overflow. Finally, the fourth ENode corresponds to a fixed delay of 300 ms for the propagation delay.

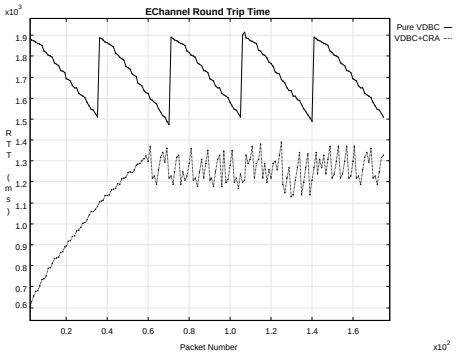


**Figure 4. Delay of the QSAT EC**

### 4.3 Implementation and first measures

The implementation of the EChannel can be achieved either in the UML2 environment, providing simulation capabilities with tool such as Telelogic Tau and in an emulation framework, providing testing capabilities with real code software. The following results has been obtained from the emulation framework implemented into a simple PC under FreeBSD system (more details on the implementation is available in see [4]).

In the studied case, the sender is situated at ST side and generates a constant bit rate of 128kbps measured at the IP layer. The stream is going through the return link (DVB-RCS) and a response is sent for each packets through the Forward link (DVB-S) to measure the RTT delay. The Forward link is supposed to not introduce any congestion. Two scenarios are foreseen, one with a pure guaranteed VBDC and a mix of CRA with guaranteed VBDC. For both scenarios the packets RTT delay is measured. The VBDC case implements a SLA with a guaranty $minVBD = 512kbps$. The bit rate of the input traffic is supposed to be less than the $minVBDC$. A global request for all the traffic buffered is done every 530 ms. This implies the bursty behavior of VBDC that sending at the higher rate as possible the traffic received in this 530 ms interval. This leads to a delay varying a lot depending on when the data actually arrived in the buffer and when the next request is sent. When the capacity is granted, all the traffic in the queue is sent at the rate of 512kbps. The RTT delay of packets is thus periodically decreasing as seen in the Figure 4, because the incoming traffic arriving just before a capacity request is sent is waiting less in the buffer than the one arriving just after a request has been sent.

The Mixed CRA and VBDC case implements a SLA of 64Kbps for the CRA and $minVBDC = 64Kbps$. The bit rate sent is still 128kbps, and now corresponds to twice as much as the available CRA bandwidth. Consequently, during the period corresponding to the transfer, processing and granting of VBDC requests, the packets are buffered in the bandwidth queue. The buffer size increases, implying an increase of the delay (see the dashed line on Figure 4). At the moment, first VBDC requests went through, the bandwidth is set application rate, thus the bit rate entering the buffer is equal to the outgoing traffic. Consequently, the buffer is neither emptying nor increasing in size but get stabilized as well as the delay perceived by packets. Some fluctuations still persist due to the bursty behavior of VBDC requests.

## 5 Concluding Remarks

In this paper, we introduced an UEF allowing to build experiment in both simulation with UML2 models and emulation with real protocol implementation. It is based on the concept of experiment channel that allows the description of arbitrary complex impairment behavior over the data packets conveyed between the evaluated protocol entities. The use of the UEF is provided through the description of a case-study for a satellite system to be integrated into a heterogeneous QoS network.

## References

[1] M. Carson and D. Santay. NISTNet : A Linux-Based Network Emulation Tool. *ACM Computer Communication Review*, 2003.

[2] ETSI/DVB. Interactive channel for satellite distribution systems, 2001.

[3] E. Exposito, R. Malaney, X. Wei, and D. Nghia. Using the xqos platform for designing and developing the qos-seeker system. In *INDIN'05, 3rd International IEEE Conference on Industrial Informatics*, Perth, Australia, 2005.

[4] M. Gineste, H. Thalmensy, L. Dairaine, P. Sénac, and M. Diaz. Active Emulation of a DVB-RCS Satellite Link in an End-to-end QoS-oriented Heterogeneous Network. In *23th AIAA International Communication Satellite Systems (ICSSC)*, page 12, Rome, Italy, 2005.

[5] I. Jacobson, G. Booch, and J. Rumbaugh. *The Unified Software Development Process*. Addison-Wesley, 1998.

[6] OMG. Object management group, 2005.

[7] L. Osterweil. Strategic directions in software quality. *ACM Computing Surveys (CSUR)*, 28(4):738–750, December 1996 1996.

[8] L. Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *ACM Computer Communication Review*, 27(1):31–41, 1997.

[9] Telelogic. Tau uml2 tool, 2005.

[10] M. Zec and M. Mikuc. Operating system support for integrated network emulation in imunes. In *First Workshop on Operating System and Architectural Support for the on demand IT InfraStructure*, Boston, USA, 2004.

[11] P. Zheng and L. M. Ni. Empower: A network emulator for wireline and wireless networks. In *IEEE Infocom*, San Francisco, 2003.