QoS EVALUATION OF IP TELEPHONY SERVICES: A SPECIFICATION LANGUAGE BASED SIMULATION SOFTWARE TOOL

Marco Roccetti[a], Alessandro Aldini[b], Marco Bernardo[b], Roberto Gorrieri[a]

[a]Dipartimento di Scienze dell'Informazione, Università degli Studi di Bologna
Mura Anteo Zamboni 7, 40127 Bologna, Italy

[b]Centro per l'Applicazione delle Scienze e Tecnologie dell'Informazione
Università di Urbino, Piazza della Repubblica 13, 61029 Urbino, Italy

Abstract: This article reports on our experience in using the EMPA$_{gr}$/ TwoTowers technology, we have developed in the past six years, in order to provide support to the design of new protocols for audio communications over IP, to the verification of correctness related properties, and to the assessment of the QoS of different design alternatives as well as the performance comparison with other existing protocols. The novelty of our approach amounts to the use of an executable specification language extended with performance modeling capabilities (called EMPA$_{gr}$) to precisely describe protocols at a suitable level of abstraction. Our approach based on formal techniques may be advantageous with respect to conventional simulators routinely used by network engineers because, besides allowing the quality and the efficiency of the specified protocols to be assessed, it provides a compositional support to system modeling at a high level of abstraction and is capable of verifying functional properties of systems.

Key Words: IP Telephony, Quality of Service, Formal Specification Languages, Simulation Software Tools, Performance Evaluation and Modelling

1. INTRODUCTION

Internet based audio applications such as IP telephony, Internet multicast conferencing, IP phone directories, and voice Web browsing, are emerging as some of the most promising services (e.g., [1]). Key to success of such sophisticated

applications are the Quality of Service (QoS) requirements that should be guaranteed in spite of some unforseeable environment constraints typical of wide area networks, such as variable queueing delays and packet loss during network transmission. For instance, two of the main factors that affect the QoS for real time audio applications over the Internet are the quality of codec and the latency caused by the network access, the operating system, and the sound card delays. Even if improved coding technologies have allowed understandable speech to be transmitted, latency still represents a major problem. On the one hand, studies indicate that users can tolerate latencies of up to 300 msec, while it is known that unwanted delay variation (known as jitter) with peaks on the order of 500/1000 ms for congested Internet links can easily occur. On the other hand, distortions of the perceived speech may also be caused by packet loss (audio packet loss rates over 10% may have a tremendous impact on speech recognition). The scenario described above compelled computer scientists and engineers to develop techniques for ensuring that real time audio applications are implemented correctly despite of their strict functional and performance requirements. With this in view, in this article we survey the EMPA$_{gr}$/TwoTowers technology, we have developed in order to provide support to the design of new protocols for audio communications over IP, to the verification of correctness related properties, and to the assessment of the QoS of different design alternatives as well as the performance comparison with other existing protocols.

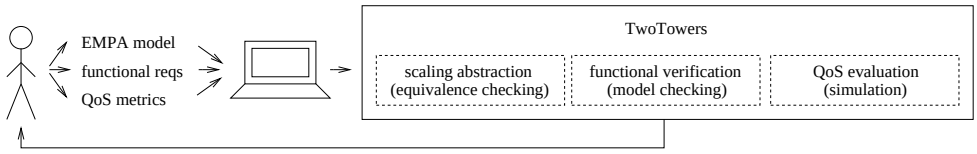

Figure 1: The EMPA$_{gr}$/TwoTowers simulation environment

As shown in Figure 1, when using the software tool TwoTowers [3], a network engineer is required to provide [4]: (*i*) a description of the audio protocol to be designed and of the operational scenario in which it will be deployed under the form of a specification in the executable modeling language EMPA$_{gr}$, (*ii*) a specification of the functional requirements to be met by the audio protocol described by means of an EMPA$_{gr}$ companion language based on temporal logic, and (*iii*) a representation of the QoS metrics of interest for the audio protocol expressed through a further EMPA$_{gr}$ companion language based on reward theory [8]. TwoTowers automatically processes the EMPA$_{gr}$ specification in order to derive a state transition model on which (*i*) the specified functional requirements are checked, and (*ii*) a simulation can be conducted to compute the QoS metrics of interest. The novelty and the advantages of the adopted approach with respect to a standard simulation environment, can

be summarized as follows.

From the modeling viewpoint, the specification language EMPA$_{gr}$ [4, 7] embeds the capability of specifying in a rigorous way the functional behavior of the audio protocol and of describing the duration of each protocol activity. This is done in a modular way by describing in a top down fashion each system component and by assembling them to form a complete and detailed system model.

From the analysis viewpoint, starting from the EMPA$_{gr}$ based system model, formal verification techniques (like model checking) can be used in a completely automatic way to investigate both functional properties and performance behavior of the audio protocol.

The rest of this paper is organized as follows. In Section 2 we present an overview of the architecture of TwoTowers. In Section 3 we show how EMPA$_{gr}$ may be exploited to support compositional modeling. Section 4 explains how the level of simulation abstraction can be adjusted by inserting or removing implementation details in such a way that the intended behavior of each protocol component is preserved. In Section 5 we report on how TwoTowers permits the specification of functional requirements by means of a temporal logic [10]. In Section 6 we concentrate on the evaluation of the QoS metrics of interest conducted by simulating the specified model of the protocol. In Section 7 we focus on the researches that have been developed by exploiting TwoTowers. Section 8 concludes the paper.

2. TwoTowers: ARCHITECTURE AND FUNCTIONALITIES

TwoTowers is divided into a graphical user interface, a compiler, a functional analyzer, and a performance analyzer (see Figure 2).

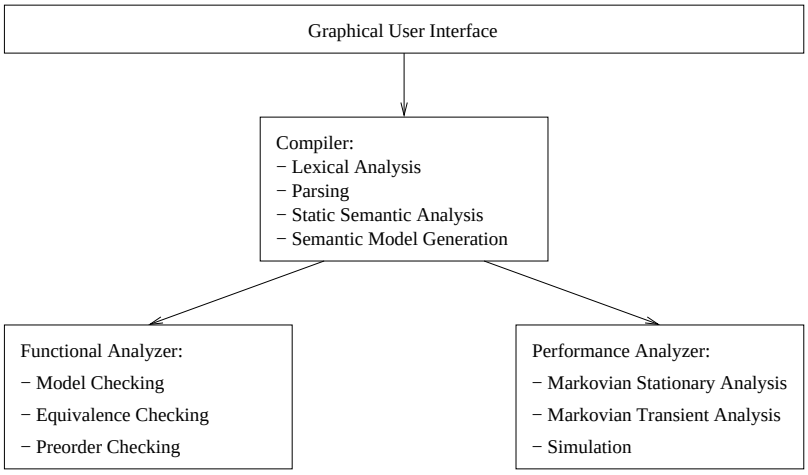

Figure 2: Architecture of TwoTowers

The graphical user interface allows the user to edit the specifications of the audio protocols in EMPA$_{gr}$, compile them, and run the various analysis routines. Additionally, it permits to edit the specifications of functional requirements and QoS metrics for audio based services.

The compiler is in charge of parsing EMPA$_{gr}$ specifications (see Section 3) and pinpointing lexical, syntactical and static semantical errors. If a specification is correct, the compiler can produce the semantic model on which further analyses are based. Such a semantic model is a labeled state transition graph where states are algebraic terms derived from the EMPA$_{gr}$ specification of the protocol and transitions are labeled with actions occurring in the EMPA$_{gr}$ specification.

The functional analyzer takes care of verifying that certain functional requirements are satisfied by the semantic model derived from a correct EMPA$_{gr}$ specification. This is achieved by interfacing TwoTowers with the Concurrency Workbench tool [11], thereby providing support for equivalence checking (see Section 4) and model checking (see Section 5).

Finally, the performance analyzer computes QoS metrics (see Section 6) on the semantic model derived from a correct EMPA$_{gr}$ specification. This is achieved either by performing a simulation of an EMPA$_{gr}$ specification according to the method of independent replications [17], or by numerically analyzing the continuous/discrete time Markov chain derived from the EMPA$_{gr}$ specification [16].

3. COMPOSITIONAL MODELING

As mentioned in the previous Section, TwoTowers accepts specifications of audio protocols written in the executable modeling language EMPA$_{gr}$. This language allows the user to express audio protocol configurations as modular algebraic specifications rather than programs expressed in imperative programming languages such as e.g. C++.

Specifically, the EMPA$_{gr}$ modeling paradigm is based on actions, whereby the functional and performance aspects of system activities are modeled, and a set of algebraic operators – such as sequential composition, alternative composition, and parallel composition – which permit the integration of system activities in order to formally build a description of the overall system. In particular, the sequential composition and the alternative composition operators are typically used to order and serialize different activities within a single protocol component. For instance, the buffer at the receiving site of an audio protocol specification may be simply described by the term

$$Buffer = <consume>.Buffer,$$

which repeatedly accepts new audio packets through the action *consume*, while

the producer component of the sending site that puts packets into the buffer may be specified by the term

$$Producer = <produce>.Producer + <wait>.Producer,$$

which either creates a packet through the action *produce* or stays idle through the action *wait*. The parallel composition operator is usually employed to describe the general architecture constituted by the individual components executed in parallel and the way in which they interoperate. In particular, by "$Producer \parallel_S Buffer$" we express a link between the *Producer* and the *Buffer* by specifying (in the set S) that the action *consume* is connected to the action *produce*, meaning that the two components synchronize on such activities. This modeling style permits to dynamically compose single EMPA_{gr} specifications of different audio protocol components into arbitrary configurations.

Then, each component can be embedded with additional information that allow both functional and nonfunctional behaviors to be described. For instance, a first extended variant of the audio protocol specification may include the input buffer size:

$$\begin{aligned} Buffer(buffer\_size, current\_level) = \text{if } (current\_level = buffer\_size) \text{ then} \\ <consume>.Buffer(buffer\_size, current\_level - 1) \\ \text{else } (\ <accept>.Buffer(buffer\_size, current\_level + 1) + \\ <consume>.Buffer(buffer\_size, current\_level - 1)\ ) \end{aligned} .$$

At any time the *Buffer* is ready (i) to accept new incoming packets if not full (we change the synchronization set S of $Producer \parallel_S Buffer$ so that now the synchronizing actions are *accept* and *produce*), and (ii) to free a buffer cell by consuming a received packet via the action *consume*. Additionally, the actions of the *Producer* can be parametrized in order to numerically estimate the idleness periods and the packet generation phases:

$$Producer = <produce_{pack\_gen\_rate}>.Producer + <wait_{delay}>.Producer,$$

where *pack_gen_rate* and *delay* are the parameters of two probability distribution functions which express the packet generation rate and the idleness period duration, respectively. The choice between the two actions is probabilistically guided by such probability distributions.

At this point, if we want to detail the system by modeling the network, we may add a term

$$\begin{aligned} Channel = <send>.P \\ P = <transmit_{trans\_delay}>.<deliver>.Channel + <lose_{loss\_rate}>.Channel, \end{aligned}$$

which is put in parallel between the *Producer* and the *Buffer* by simply changing the communication interface in such a way that now the *Producer* interacts with the *Channel* by synchronizing the actions *produce* with the action *send*,

while the *Channel* and the *Buffer* communicate by synchronizing on the actions *deliver* and *accept* (see the graphical representation of Figure 3).

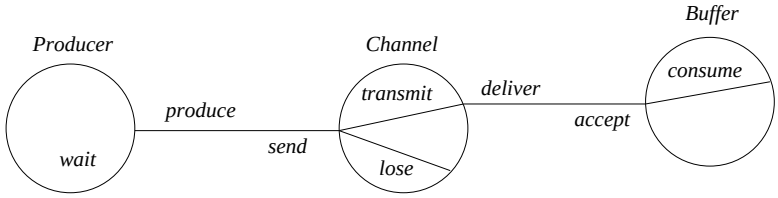

Figure 3: Composing audio communication scenario

The above scenario can be both equipped step by step with more detailed information (e.g., by specifying an acknowledgement mechanism in the *Buffer* and a full duplex *Channel*), and enriched with other components (e.g., by adding a receiver component which consumes the buffered packets at a given rate and plays out them). Therefore, through a stepwise refinement modeling process, we can pass from the above simplified scenario to the complex and detailed algebraic specification of a real protocol for the transmission of soft real time audio over the Internet (see, e.g., [2]).

As emphasized by the example above, unlike traditional simulation environments, the EMPA$_{gr}$ specification paradigm provides the network engineers with a flexible and intuitive modeling framework where alternative configurations can be set up on-the-fly under different network dynamics. This is due to the higher level of abstraction at which the overall system can be specified and the modularity provided by the specification language.

4. ABSTRACTING SIMULATION

The provision of simulation performance for large system models at acceptable costs in terms of memory and time requirements is typically achieved by tuning the level of simulation abstraction [9]. On the contrary, more accurate simulation results may be obtained by increasing the level of detail of the model. To deal with such a trade-off between accuracy of the model and accuracy of the simulation results, several simulator environments provide support for scaling the level of abstraction. However, to the best of our knowledge none of them supplies methods and tools to verify that changing the level of simulation abstraction preserves the functional consistency of the developed models. To this aim, TwoTowers provides a technique that may be exploited during the stepwise refinement modeling process to verify the correctness of each refinement step as follows. Starting from a very simple and abstract model which surely captures the functional requirements, the idea is to iteratively refine and substitute it with a more detailed one until the point is reached at

which performance information about system activities are available and may be attached to the model.

The verification of the functional consistency of the different models iteratively obtained by scaling the level of abstraction is accomplished within TwoTowers with the equivalence checking technique [12]. In essence, such a technique consists of formally and automatically verifying that the state transition graphs underlying two consecutive EMPA$_{gr}$ specifications in the stepwise refinement process have matching observable behaviors. More precisely, given two equivalent states s_1 and s_2 in the two graphs, whenever s_1 can execute an action a thus reaching state s_1', then s_2 can perform the same action a (possibly preceded and followed by a sequence of more detailed actions not visible from the outside) thus reaching a state s_2' equivalent to s_1', and vice versa [12]. Thus, applying equivalence checking to EMPA$_{gr}$ specifications reduces to verifying that the behavior of the more detailed specification is the same as the behavior of the more abstract specification up to the newly introduced details.

As an example, consider a simple audio streaming mechanism like the one presented in Section 3. The simplest abstract EMPA$_{gr}$ model of such an audio mechanism may consist of a buffer with a fixed capacity into which the server puts audio frames (action *produce*) at a certain rate and from which the receiver withdraws audio frames (action *consume*) with a given rate. On the one hand, if we knew the rates at which audio frames are put into/withdrawn from the buffer, such performance information could be plugged into the model thus allowing simulation. On the other hand, suppose that such performance information are not known at that generic abstraction level, but more detailed performance information are available about each specific activity of the streaming mechanism such as the audio frame generation process at the server, the frame reception process at the client, the audio playout strategy, and the acknowledgement transmission mechanism. In such a case, we need a more accurate model that takes into account the available performance information. In order to make sure that the more detailed EMPA$_{gr}$ model of the mechanism is a correct refinement of the more abstract one, equivalence checking is applied to the state transition graphs of the more abstract EMPA$_{gr}$ model and of a variant of the more detailed EMPA$_{gr}$ model. Such a variant is obtained by hiding all the actions occurring in the more detailed EMPA$_{gr}$ model (e.g. *transmit*) except for those originally expressed also in the more abstract EMPA$_{gr}$ model, i.e. *produce* and *consume*. If the equivalence checking outcome is positive, the more detailed EMPA$_{gr}$ model of the mechanism is a correct refinement of the more abstract one as the behavior of actions *produce* and *consume* is preserved.

5. VERIFICATION SUPPORT

An important advantage of TwoTowers is the possibility of exploiting well known formal verification techniques to investigate functional properties of the protocols. Indeed, with respect to conventional simulation environments, TwoTowers provides verification of the satisfaction of audio protocol correctness properties even before proceeding with the calculation of QoS metrics. Prominent examples of functional verification are checking that an audio packet exchange mechanism does not lead to deadlock, controlling that certain activities are carried out according to a given order, or ensuring that certain resources are used in a mutually exclusive way.

The technique adopted in TwoTowers to provide such a verification support is called model checking [10]. With model checking we refer to the activity of expressing audio protocol functional requirements by means of a set of formulas in some temporal logic and verifying that a given EMPA_{gr} specification of the audio protocol is a model for those formulas, i.e. it possesses the properties formally described by those formulas.

As an example, in order for the audio streaming mechanism of Section 3 to be correctly designed, it is required that no audio frames are enqueued in the receiving buffer if they are not produced by the sender. This kind of functional requirement can be captured by means of a formula expressed in the temporal logic implemented in TwoTowers that establishes the order in which actions *produce* and *accept* must be executed. At this point, the model checker available in the functional analyzer of TwoTowers can be employed to automatically verify whether the EMPA_{gr} model of the audio streaming mechanism satisfies this functional requirement.

6. QoS EVALUATION

After formally verifying that a given EMPA_{gr} model of an audio protocol satisfies functional properties of interest, TwoTowers allows the network engineer to describe the QoS metrics that are relevant for that protocol. Typical considered metrics for audio over IP are the following [2]:

- the quality of the perceived audio, which is affected by the type of used codec and the experienced packet loss rate;
- the interactivity degree of the audio communication, which is influenced by the average packet audio playout delay and the jitter.

The QoS metrics like those mentioned above can be defined and evaluated during simulation runs within TwoTowers as detailed in the following. Since EMPA$_{gr}$ models are composed of actions, the simulation of an EMPA$_{gr}$ model consists of executing its constituent actions.

First, all the values for probability distributions involved in the simulation are sampled through pseudo random number generators. For instance, let us consider the transmission delay that is an important factor affecting the average packet audio playout delay. Suppose, as seen in Section 3, that the action denoting the transmission of a packet is $transmit_{trans\_delay}$, where the duration of the related activity is described by parameter *trans_delay* associated with a particular probability distribution (e.g. an exponential distribution). Therefore, each time such an action occurs, we sample a value for the transmission delay (according to the associated probability distribution) to compute the time to be elapsed to complete the related activity. In alternative, instead of being randomly sampled, such a value can be read from a suitable file in case of trace driven simulation. This latter modality is used to capture trace of live traffic to be injected into the simulation environment in order to evaluate the simulated system responses.

Then, during a simulation run, QoS metric related samples are collected by observing the occurrences of those actions which are relevant for the metrics of interest. More precisely, a reward is associated with each relevant action and an accumulator measures the amount of accumulated reward, i.e. the accumulator is incremented by the reward value each time that action is performed during simulation. As a consequence, each QoS metric is defined as an arithmetic expression whose constituents are the actions occurring in the EMPA$_{gr}$ model. The evaluation of such an expression at the end of a simulation run consists of replacing each action occurring in the expression with the corresponding accumulated reward value. This leads to the computation of the average, the variance, or the probability distribution of the specified performance metric, together with the related confidence intervals. For instance, by measuring during the simulation the time elapsed between the *produce* event and the *consume* event related to each packet, and by evaluating via the reward technique above the expression $lose/(lose + consume)$, we may study the trade off between the packet losses and the average packet audio playout delay.

7. IP TELEPHONY RESEARCHES USING TwoTowers

The types of research we have developed in the field of IP telephony by exploiting the EMPA$_{gr}$/TwoTowers technology fall in the two following categories:

- QoS prediction of an audio mechanism to be developed.
- QoS comparison of several existing audio mechanisms.

7.1. QoS Prediction

Many Internet audio tools were designed and developed that provide an audio quality which is comparable to that of the circuit-switched telephone system, at a typical sampling rate of 8 kHz. Those IP audio tools, such as NeVot and rat, operate by periodically sampling talkspurts generated at the sending host, packetizing them, and transmitting the obtained packets to the receiving site using datagram based connections (e.g. UDP). Typically, those packet audio applications embody adaptive playout mechanisms that are used to buffer talkspurt packets at the receiving site and to delay their playout time in order to compensate for variable network delays.

Recently, a new playout mechanism [15] has been designed that, unlike others, dynamically adapts the talkspurt playout delays to the network traffic conditions assuming neither the existence of an external mechanism for maintaining an accurate clock synchronization between the sender and the receiver, nor a specific distribution of the end-to-end transmission delays experienced by the audio packets. This playout mechanism is the basis of an Internet audio tool, called BoAT, developed at the University of Bologna (see `http://radiolab.csr.unibo.it/BoAT/src`). Succinctly, the technique for dynamically adjusting the talkspurt playout delay is based on obtaining, in periodic intervals, an estimation of the upper bound for the packet transmission delays experienced during an audio communication. Such an upper bound is periodically computed using round trip time values obtained from packet exchanges of a three way handshake protocol performed between the sender and the receiver. At the end of such a protocol, the receiver is provided with the sender's estimate of an upper bound for the transmission delay that can be used in order to dynamically adjust the talkspurt playout delay and the receiver buffer size. A correct estimate of such an Upper Bound for the audio packets Transmission Delay (denoted as UBTD) represents a very important parameter of the audio control mechanism since it directly influences the final talkspurt playout delay value perceived by the receiver. The proposed audio control mechanism guarantees that the talkspurt playout delay may be dynamically set from one talkspurt to the next, without causing gaps inside the talkspurt themselves, provided that intervening silence periods of sufficiently long duration are exploited for the adjustment.

In order to develop such a mechanism without incurring in the costs due to the late discovery of errors and inefficiencies, part of the design process has relied on simulative modeling and experimentation with TwoTowers. In essence, the construction of the EMPA_{gr} model of the audio mechanism has been part of the design process of the mechanism itself. The QoS figures obtained from the EMPA_{gr} model have turned out to be useful to predict the

adequacy of the policy adopted in the design of the audio playout delay control mechanism.

exp.	initial synch	90% confid. inter.	periodic synch	90% confid. inter.
1	76.527	[76.329, 76.724]	96.702	[96.335, 97.070]
2	73.829	[73.684, 73.975]	96.712	[96.490, 96.934]
3	71.786	[71.606, 71.967]	96.453	[96.183, 96.723]
4	71.598	[71.406, 71.790]	96.583	[96.243, 96.922]
5	78.935	[78.775, 79.094]	96.600	[96.296, 96.904]
6	70.829	[70.626, 71.032]	96.612	[96.282, 96.942]
7	72.368	[72.159, 72.577]	96.481	[96.183, 96.779]
8	78.268	[78.073, 78.462]	96.594	[96.233, 96.954]
9	74.422	[74.214, 74.629]	96.480	[96.154, 96.806]
10	74.518	[74.290, 74.745]	96.745	[96.461, 97.028]

Table 1: Prediction of the synchronization effect in BoAT: percentage of played out packets

We now show two significant examples of using TwoTowers during the development of BoAT. As mentioned above the playout mechanism embedded in BoAT exploits an internal synchronization protocol needed to keep synchronized the system clocks of the peers of the audio communication, while governing the playout process at the receiver. To demonstrate the necessity of having such a synchronization protocol embodied in BoAT, during the development phase we defined two EMPA_{gr} specifications modeling the behavior of BoAT with and without the internal clock synchronization protocol. In both cases we conducted 10 experiments, each consisting of 10 simulation runs concerning a period of 30 sec. In particular, in such experiments packet transmission delays are assumed to follow a Gaussian distribution. The results are reported in Table 1 (where also 90% confidence intervals are shown) and illustrate the percentage of played out packets for both EMPA_{gr} specifications. Those results correctly predicted that performing the synchronization only once at the beginning of the audio communication results in an audio packet loss around 20-25%, while the audio packet loss rate falls down to 4-5% if the clock synchronization protocol is run with a frequency of one second [6].

The second example is related with the UBTD tuning process during the design of BoAT. It is well known that two are the most important metrics that influence the user perception of audio data: 1) the percentage of audio packets that arrive at the destination too late to be played out (and can be so considered lost), and 2) the amount of total delay that audio packets have to experience before they are played out at the destination. Thus, in order

to evaluate the QoS of an Internet audio mechanism the playout delay vs. the loss rate tradeoff metric must be studied. To this end, we have conducted simulation runs where packet transmission delays, instead of being sampled, are computed by using experimentally obtained delay measurements of three 5 min long audio conversations between two different Internet sites (trace driven simulation).

In Figure 4 we report the results obtained to investigate the influence of the UBTD value on the QoS metric mentioned above. As seen from the figure, it has been possible to predict that the playout mechanism embodied in BoAT has the potential to produce intelligible audio at the receiver if a reduction by a factor of up to 50% of the UBTD value is carried out, at the price of a loss rate increase [2].

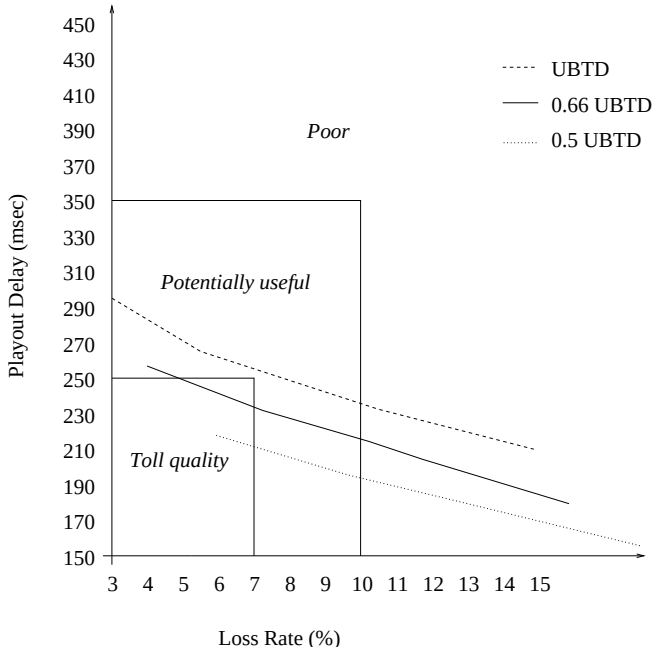

Figure 4: UBTD tuning in BoAT

7.2. QoS Comparison

Originally, each playout mechanism underlying the Internet audio tools mentioned in the previous section was devised and experimented under complementary network conditions (e.g., different sets of experimental traces) so that a precise comparison among them was not possible. By resorting to TwoTowers we performed a precise simulative comparison of each audio playout mechanism under the same network conditions, obtaining consistency with experimental results when available.

In particular, we have considered traditional QoS audio metrics in order to compare three different audio playout mechanisms via simulation using both experimentally obtained delay measurement of several 5 min long audio conversations between two different Internet sites and two 1 min long audio conversations randomly generated according to Gaussian distributed delays and exponentially distributed delays, respectively. Specifically, we have contrasted the audio playout mechanisms embedded in BoAT with the two audio playout control mechanisms proposed in [13] and [14], respectively. As an example, in Figure 5 we report on one of the trace driven simulation mentioned above. For the purpose of this example, audio mechanism #3 denotes the audio playout mechanism of BoAT, while audio mechanisms #1 and #2 denote the audio playout control mechanisms presented in [13] and [14], respectively. The figure shows the playout delay/loss rate tradeoff metric in a trace driven scenario and illustrates that mechanism #3 outperforms the others in this particular case. The full comparison of the three playout control mechanisms conducted in [2] has revealed that neither of them outperforms the others in general, as their performance depends on the network traffic conditions.

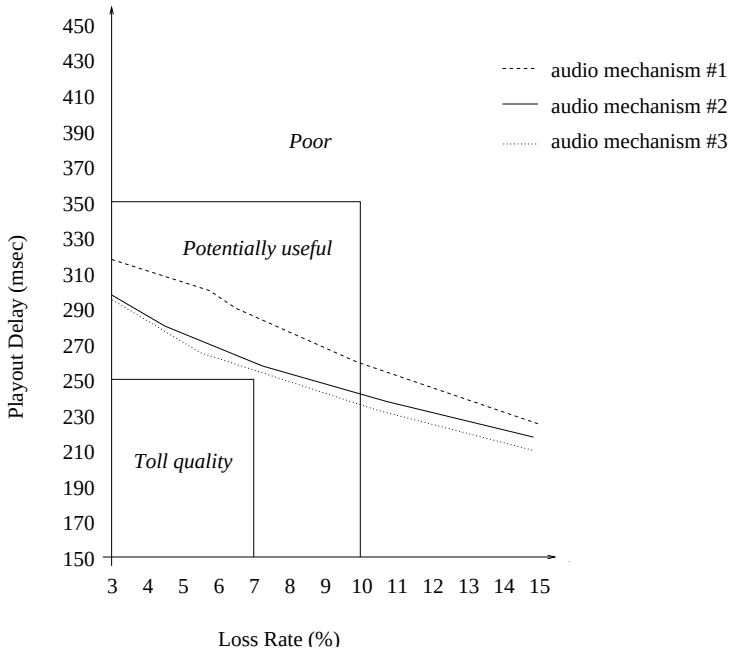

Figure 5: Comparing the QoS of different IP audio mechanisms

8. CONCLUSIONS

In this article we have surveyed the main features of the EMPA$_{gr}$/TwoTowers technology used to design, verify, simulate and implement IP audio mech-

anisms. The prominent advantages of our technology with respect to existing simulators are the modularity of the adopted specification language, the strategy for scaling the simulation abstraction level, and the support for the verification of functional requirements. Our experience in using the EMPA$_{gr}$/TwoTowers technology for developing a new audio mechanism for the Internet has reinforced our idea that the combination of simulation and formal description and verification techniques may be a successful approach in providing QoS supporting for voice communication in the next generation Internet.

Future developments of our simulation environment should address the implementation of an easier-to-use architectural description language based on EMPA$_{gr}$ [5], as well as the exploitation of the compositional structure of EMPA$_{gr}$ simulation models in order to achieve parallel and distributed simulation.

Acknowledgements This research has been partially funded by Italian M.I.U.R. and C.N.R., and by European FP5 RTD project TAPAS (IST-2001-34069).

References

[1] A. Aldini, A. Amoroso, M. Roccetti (2003). A Secure Protocol for Voice-Operated E-Commerce Systems over IP Networks, *International Journal of Pure and Applied Mathematics*, to appear.

[2] A. Aldini, M. Bernardo, R. Gorrieri, M. Roccetti (2001). Comparing the QoS of Internet Audio Mechanisms via Formal Methods, *ACM Transactions on Modeling and Computer Simulation*, **11**, 1-42.

[3] M. Bernardo (2002). *TwoTowers 2.0 User Manual*, (`http://www.sti.uniurb.it/bernardo/twotowers/`).

[4] M. Bernardo (1999). Theory and Application of Extended Markovian Process Algebra, *Ph.D. Dissertation*, University of Bologna (Italy), (`http://www.sti.uniurb.it/bernardo/publications/theses.html`).

[5] M. Bernardo, L. Donatiello, P. Ciancarini (2002). Stochastic Process Algebra: From an Algebraic Formalism to an Architectural Description Language, In: *Performance Evaluation of Complex Systems: Techniques and Tools*, M.C. Calzarossa and S. Tucci Editors, Lecture Notes in Computer Science, **2459**, 236-260.

[6] M. Bernardo, R. Gorrieri, M. Roccetti (1999). Formal Performance Modelling and Evaluation of an Adaptive Mechanism for Packetised Audio over the Internet, *Formal Aspects of Computing*, **10**, 313-337.

[7] M. Bravetti, M. Bernardo (2000). Compositional Asymmetric Cooperations for Process Algebras with Probabilities, Priorities, and Time, In: *Proc. of 1st Int. Workshop on Models for Time Critical Systems*, Electronic Notes in Theoretical Computer Science, **39**:3, State College (PA).

[8] M. Bravetti, M. Bernardo (2003). Performance Measure Sensitive Congruences for Markovian Process Algebras, *Theoretical Computer Science*, **290**, 117-160.

[9] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, H. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, H. Yu (2000). Advances in Network Simulation, *IEEE Computer*, May 2000, 59-67.

[10] E.M. Clarke, O. Grumberg, D.A. Peled (1999). *Model Checking*, MIT Press, Cambridge (MA).

[11] W.R. Cleaveland, J. Parrow, B. Steffen (1993). The Concurrency Workbench: A Semantics-Based Tool for the Verification of Concurrent Systems, *ACM Trans. on Programming Languages and Systems*, **15**, 36-72.

[12] R. Milner (1989). *Communication and Concurrency*, NJ, Prentice Hall: Englewood Cliffs.

[13] S.B. Moon, J. Kurose, D. Towsley (1998). Packet Audio Playout Delay Adjustment: Performance Bounds and Algorithms, *ACM Multimedia Systems*, **6**, 17-28.

[14] R. Ramjee, J. Kurose, D. Towsley, H. Schulzrinne (1994). Adaptive Playout mechanisms for Packetized Audio Applications in Wide-Area Networks, In: *Proc. of INFOCOM '94*, Montreal (Canada).

[15] M. Roccetti, V. Ghini, G. Pau, P. Salomoni, M.E. Bonfigli (2001). Design and Experimental Evaluation of an Adaptive Playout Delay Control Mechanism for Packetized Audio for Use over the Internet, *Multimedia Tools and Applications*, **14**, 23-53.

[16] W.J. Stewart (1994). *Introduction to the Numerical Solution of Markov Chains*, Princeton, Princeton University Press.

[17] P.D. Welch (1983), The Statistical Analysis of Simulation Results, In: *"Computer Performance Modeling Handbook"*, S.S. Lavenberg Editor, Academic Press, 267-329.