A Simulative Analysis of Internet Audio Mechanisms Using Formal Methods [*]

Alessandro Aldini, Marco Bernardo, Roberto Gorrieri, Marco Roccetti

Università di Bologna, Dipartimento di Scienze dell'Informazione
Mura Anteo Zamboni 7, 40127 Bologna, Italy
E-mail: {aldini, bernardo, gorrieri, roccetti}@cs.unibo.it

Abstract

Stochastically timed process algebras based software tools are becoming an important technology to be used for carrying out both functional and performance analysis of several and different case studies (e.g. distributed algorithms, communication protocols, ATM switches, ...). TwoTowers is one of those tools which has been recently enhanced with routines for accepting and analyzing – via discrete event simulation – systems specified with *value passing EMPA*, a stochastically timed process algebra where durations can follow any probability distribution. We show the usefulness of these extensions by a case study concerning two realistic mechanisms for audio transmission over the Internet that have been modeled in value passing EMPA and then simulated by TwoTowers under various (experimentally obtained on the field or randomly generated) traffic conditions. The discrete event simulation of the two value passing EMPA specifications has revealed that neither of the two mechanisms outperforms the other in general, as their performance depends on the traffic conditions.

Keywords: Packetized Voice over the Internet, Simulation in Telecommunications, Formal Methods and Simulation, Multimedia Applications, Computer Aided Analysis and Verification

1 Introduction

Stochastically timed process algebras [5, 9, 11] are algebraic languages equipped with a small set of powerful operators (among which a parallel composition operator) which allow parallel and concurrent computing and communication systems to be compositionally modeled starting from their components [14]. Since system activities are formalized through actions which comprise both the type and the time duration of those activities, it is possible to verify functional properties and to evaluate performance measures of systems. The common feature of most of these languages is that action durations are exponentially distributed, hence enjoying the so-called memoryless property: the residual duration of an action under execution is independent of the time already spent. The semantics for the language can thus be defined in the classical interleaving style and the resulting performance model turns out to be a Markov chain, that can be analytically solved when the state space is finite hence allowing performance measures to be effectively derived. For such a reason, those languages are called Markovian process algebras. EMPA is a Markovian process algebra, rich enough to include probability, priority, nondeterminism, and stochastic time. The developed theory as well as the related integrated approach described in [5] have been mechanized, resulting in the tool TwoTowers [4] that is being profitably used in the functional and performance analyses of several case studies, ranging from distributed algorithms (e.g., mutual exclusion protocols [3]) to communication systems (e.g., ATM switches [1]). However, the limitation to exponentially distributed durations prevents the applicability of the EMPA/TwoTowers technology to many other interesting systems.

In this paper we show that the EMPA/TwoTowers technology can be extended in such a way that the drawback mentioned above is overcome. First of all we would like to point out that the limitation to exponentially distributed durations is mitigated by the fact that phase type distributions, which approximate many frequently occurring distributions, can be indirectly represented in EMPA as combinations of expo-

[*]Correspondence to: Prof. Marco Roccetti, Dipartimento di Scienze dell'Informazione, Università di Bologna, Via Mura Anteo Zamboni 7, 40127 Bologna, Italy, E-mail: roccetti@cs.unibo.it

nential distributions at the expense of a state space growth. Secondly, some stochastically timed process algebras have appeared in the literature which allow for generally distributed durations. The treatment of general distributions in these stochastically timed process algebras is not trivial: they either rely on suitable operators which represent the random setting of a timer and the expiration of a timer [8], or are based on non interleaving semantics [5].

The introduction in EMPA of a technique termed *value passing* allows us to express general distributions without either resorting to timer related operators in the syntax or even leaving the interleaving framework. In the resulting language, called $\mathrm{EMPA}_{\mathrm{vp}}$, processes can exchange values through action synchronization. This means that $\mathrm{EMPA}_{\mathrm{vp}}$ is able to deal with data. Thus, in $\mathrm{EMPA}_{\mathrm{vp}}$ probability distributions can be represented as value passing based expressions thereby allowing generally distributed delays to be modelled, the occurrence times of timed events can be sampled from expressions representing probability distributions and be stored into appropriate lists, and suitable clock variables can be employed in order to detect when it is time to execute a timed event. In other words, we have found out that adding value passing to a process algebra with interleaving semantics allows us to model systems with activities whose duration follows an arbitrary distribution. So, from a modeling standpoint, it is not strictly necessary to resort to a stochastically timed process algebra where actions can be directly given arbitrarily distributed durations.

The price to pay when dealing with generally distributed durations with $\mathrm{EMPA}_{\mathrm{vp}}$ is that only a simulative analysis of performance can be conducted, because an exact analysis would be meaningless in that carried out on Markov chains which lack information about the value of the duration of timed events. TwoTowers has thus been extended with routines for the simulative analysis of performance, which is conducted by statistically inferring performance measures from several independent executions of the specification under consideration. Since the semantics for $\mathrm{EMPA}_{\mathrm{vp}}$ is defined in an operational style and the simulation does not require the construction of the state space a priori, this kind of analysis can in general be employed to study systems with a huge or even infinite state space. We have tested our extension of the EMPA/TwoTowers technology on an interesting case study: measuring the quality of service (QoS) of two soft real time applications concerning audio transmission over the Internet.

The value of integrating voice and data networks onto a common platform is well known. Even if the IP community has not considered the provision of QoS guarantees with the same intensity as the ATM community – the current Internet service model offers a flat, classless, best-effort service to its users – however the feasibility and the expected QoS of audio applications over IP networks have to be carefully considered, if we wish those applications be successful. Many are the factors that affect the QoS for real time audio applications over the Internet (such as, e.g., codec, access, operating system, sound card delays) but probably the most important metric that influences the user perception of audio is represented by the average packet audio playout delay vs. the packet loss rate. This is precisely the metric we will use to compare the QoS of two such protocols: [16] and [18]. These have been formally modeled in $\mathrm{EMPA}_{\mathrm{vp}}$, then analyzed with TwoTowers via simulation using both experimentally obtained delay measurements of three 10 min long audio conversations between two different Internet sites (trace driven simulation) and two 30 sec long audio conversations randomly generated according to Gaussian distributed delays and exponentially distributed delays, respectively. Note that originally each mechanism has been devised and experimented for complementary network conditions, so that a precise comparison of the two mechanisms was not easy. We have here simulated both mechanisms under the same network conditions, obtaining full agreement with the experimental results, when available, but also allowing us to compare precisely the two mechanisms.

We observe that using the $\mathrm{EMPA}_{\mathrm{vp}}$ based simulator implemented in TwoTowers instead of conventional simulators routinely used by network engineers may be advantageous for several reasons. Building an $\mathrm{EMPA}_{\mathrm{vp}}$ specification of a protocol (and in particular of nontrivial protocols like those considered in this paper) should be easier than developing a simulator for the protocol in the standard way because $\mathrm{EMPA}_{\mathrm{vp}}$ allows one to work at a higher level of abstraction in a compositional framework. Moreover, well known formal verification techniques can be applied to an $\mathrm{EMPA}_{\mathrm{vp}}$ specification of a protocol to detect functional properties, whereas this is not the case for conventional simulation programs. As an example, we have verified that the two considered audio mechanisms are deadlock free.

The remainder of this paper is organized as follows. Sect. 2 recalls the main features of the two audio mechanisms that represent our case study. The results of their simulative analysis are reported in Sect. 3, where the effects that playout delays and loss rates (as well as buffer dimensions) may have on the QoS are precisely shown. We observe that this paper is the natural continuation of [17, 6] where the EMPA/TwoTowers technology was used for the first time for modelling and evaluating the audio mechanism proposed in [18].

2 Packetized Voice over the Internet

Internet audio services operate in a bandwidth-, delay- and packet loss- constrained environment. Recently, several codecs have been designed that work well in the presence of the scarce network bandwidth constraint [12]. For example, the ITU codecs G.729 and G.723.1 [13] have been designed for transmitting audio data at bit rates ranging from 8 Kbps to 5.3 Kbps. However, available network bandwidth is not the only requirement to meet for quality audio. In fact, each step in the audio data flow pipeline (from coding to transport, to reception, to decoding) adds delay to the overall transmission. While some delays are relatively fixed (such as coding and decoding), others depend on network conditions. For example, very high average packet delay and packet delay variance (i.e. jitter) on the order of 500/1000 msec may be experienced over many congestioned Internet links. As a consequence, packet loss percentages, due to the effective loss and damage of packets as well as too late arrivals, may vary between 15% and 40%. User studies indicate that telephony users find round trip delays greater than about 300 msec more like an half duplex connection than a real time conversation. However, user tolerance of delays may vary significantly from application to application. The most critical users may require delay of less than 200 msec, but more tolerant users may be satisfied with delays of 300-500 msec. Finally, many experimental tests have revealed that random independent packet loss rates of up to 10% have little impact on speech recognition.

The most used approach in order to ameliorate the effect of jitter and improve the quality of the audio service over the Internet is to adapt the applications to the jitter present on the network. Hence, a *smoothing* playout buffer is used at the receiver in order to compensate for variable network delays. Received audio packets are queued into the buffer and the playout of each packet of a given talkspurt is delayed for some quantity of time beyond the reception of the first packet of that talkspurt. In this way, dynamic playout buffers can hide, at the receiver, packet delay variance at the cost of additional delay; however, packets delayed past the point at which they are supposed to be played out (the playout point) are effectively lost. As a consequence, a crucial tradeoff exists between the length of the imposed additional quantity of delay (and the depth of the receiver buffer) and the amount of lost packets due to their late arrival: the longer the additional delay, the more likely it is that a packet will arrive before its scheduled playout deadline. However, too long playout delays may in turn seriously compromise the quality of the conversation over the network.

In the remainder of this section, we present two different playout delay control mechanisms that were designed to dynamically adjust the audio packets playout delay (and the receiver buffer depth) of packet audio applications, for compensating the highly variable Internet packet delays. For the sake of brevity, only the relevant features of these two nontrivial mechanisms are reviewed. The main aim of the paper, in fact, is to use TwoTowers extended with the features described in the previous section in order to compare the performance provided by the two different audio mechanisms.

2.1 Internet Audio Mechanism #1

In this section we briefly describe the adaptive playout delay adjustment algorithm on which several Internet audio tools, such as rat, NeVot, FreePhone, are based [10, 19, 16]. As mentioned above, a receiving site in an audio application buffers packets and delays their playout time. Such a playout delay is usually adaptively adjusted from one talkspurt to the next one. In order to implement this playout control policy, the playout adjustment mechanism proposed in [16] (denoted in the following as mechanism #1) makes use of the two following assumptions:

- an external mechanism exists that keeps synchronized the two system clocks at both the sending and the receiving sites (for example, the Internet based Network Time Protocol NTP);
- the delays experienced by audio packets on the network follow a Gaussian distribution.

Based on these assumptions, the playout control mechanism works by calculating the playout time p_i for the first packet i of a given talkspurt as

$$p_i = t_i + \hat{d}_i + k \cdot \hat{v}_i$$

where: t_i is the time at which the audio packet i is generated at the sending site, $\hat{d}_i$ is the average value of the playout delay (i.e. the average value of the time interval between the generation of previous audio packets at the sender and the time instants in which those packets have been played out at the receiver),

$k \in \mathbf{N}_{]0,4]}$, and finally $\hat{v_i}$ is the average variation of the playout delay. From an intuitive standpoint, the reported formula is used to set the playout time to be far enough beyond the average delay estimate, so that only a small fraction of the arriving packets should be lost due to late arrivals.

The playout point for any subsequent packet j of that talkspurt is computed as an offset from the point in time when the first packet i in the talkspurt was played out:

$$p_j = p_i + t_j - t_i$$

The estimation of both the average delay and the average delay variation are carried out using the well known *stochastic gradient algorithm* [16]:

$$\begin{aligned} \hat{d}_i &= a \cdot \hat{d}_{i-1} + (1-a) \cdot n_i \\ \hat{v}_i &= a \cdot \hat{v}_{i-1} + (1-a) \cdot |\hat{d}_i - n_i| \end{aligned}$$

where constant a (usually equal to 0.998) is a weight that characterizes the memory properties of the estimation, while n_i is the total delay introduced by the network (difference between the time at which packet i is received at the receiving site and t_i).

In addition, the playout adjustment mechanism #1 is equipped with a *delay spike* detection and management mechanism [15]. Several studies, in fact, have indicated the presence of spikes in end-to-end Internet delays. A spike is a sudden, large increase in the end-to-end network delay, followed by a series of audio packets arriving almost simultaneously. In cases where a delay spike spans multiple talkspurts, it is important to quickly react to the delay spike. The method proposed in [16] to react to spikes is as follows. Two different modes of operation are used depending on whether a transmission delay spike has been detected or not (the *spike detected mode* and the *normal mode*, respectively). For every packet that arrives at the receiver, the mechanism checks the current mode and, if necessary, switches it. If a packet arrives with a delay that is larger than some multiple of the current playout delay, the algorithm switches to the spike detected mode. The end of a spike is detected in a similar way: if the delay of a newly arrived packet is less than some multiple of the playout delay before the current spike, the normal mode is resumed. In addition, during the spike detected mode, the mechanism uses the delay of the first packet of the talkspurt as the estimated playout delay for each packet in the talkspurt. Instead, in the normal mode the mechanism operates by calculating the playout delay as described at the beginning of this section.

2.2 Internet Audio Mechanism #2

The purpose of this section is to summarize the most characterizing features of a playout delay control mechanism (#2) that is suitable for adjusting the talkspurt playout delays of full-duplex, voice-based audio communications across the Internet [18]. The mechanism foregoes the traditional assumptions of externally synchronized systems clocks as well as the use of a specific jitter delay distribution experienced by the audio packets across the Internet, and has been designed to dynamically adjust the talkspurt playout delays to the network traffic conditions at both the sender's and the receiver's sites. Succinctly, the technique for dynamically adjusting the talkspurt playout delay is based on obtaining, in periodic intervals (about 1 second), 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 caller and the callee of the audio communication. Prior to the beginning of the first talkspurt in an audio conversation, the caller initiates the packet protocol exchange activity by sending to the callee a packet timestamped with the time value shown by its own clock. All the technical details concerning this 3-way handshake protocol between the caller and the callee fall outside the scope of the paper, rather, it is more important to notice that at the end of such one protocol exchange, the callee is provided with the caller's estimate of the round trip time value (RTT) experienced during a communication between the caller and the callee. At that moment, the callee may offset its own clock by using a time value based both on the computed RTT and the time information piggybacked onto the first packet emitted by the sender, at the beginning of the protocol exchange. Due to this setting of the callee's clock, the time difference between the two clocks at the caller and at the callee, at the same time instant T, is equal to a value given by:

$$\Delta = T_{Caller} - T_{Callee} = t_0 + RTT$$

where RTT is the computed round trip time value and t_0 is the transmission delay experienced by the first packet emitted by the caller at the beginning of the protocol exchange. It is not difficult to show that Δ is an unknown quantity that may range in the interval $[RTT, 2 \cdot RTT]$ depending on the unknown value of t_0 that, in turn, may range in the interval $[0, RTT]$.

Based on the value of the time difference imposed by the above mentioned protocol between the two system clocks at the caller and at the callee, the following audio packet playout/buffering strategy may be performed at the callee's site. Each audio packet emitted by the caller is timestamped by the caller with the time value shown by the caller's clock at the moment of its generation. When such an audio packet (emitted by the caller) arrives at the callee (i.e. it is delivered to the application level of the receiving host), its generation timestamp t is compared with the value T of the receiver's clock, and a decision is made according to the following rules.

- If $t < T$, the packet is discarded having arrived too late (w.r.t. its playout time) to be buffered.
- If $t > T + \Delta$, the packet is discarded having arrived too far in advance of its playout time to be buffered.
- Finally, if $T < t < T + \Delta$, the packet is arrived in time for being played out and it is placed in the first empty location in the playout buffer at the callee's site.

Using the same rate r adopted for the sampling of the original audio signal at the caller, the playout process at the callee's site fetches audio packets from the buffer and sends them to the audio device for playout, as discussed in the following. When the callee's playout clock shows a value equal to T, the playout process searches the buffer for the audio packet with timestamp T. If such a packet is found, it is fetched from the buffer and sent to the audio device for immediate playout, while the buffer location where the audio packet has been found is marked as empty. If the packet timestamped with T is not present in the buffer, then the playout process replaces the corresponding audio sample with a silence period of length r.

In essence, with the strategy summarized above, a maximum transmission delay equal to Δ is left to the audio packets to arrive from the caller at the callee in time for playout. Consequently, a playout buffering space proportional to Δ is required at the callee for packets with early arrivals. Finally, in order for the proposed scheme to adaptively adjust to the fluctuating network delays experienced over the Internet, the clock synchronization protocol, mentioned at the beginning of this section, is first carried out prior to the beginning of the first talkspurt of the audio conversation, and then periodically repeated throughout the entire conversation. The adopted period is (as already said) about 1 second in order to prevent the two clocks (possibly equipped with different clock rates) from drifting apart. Hence, each time a new RTT value is computed by the caller, it may be used by both the callee and the caller for dynamically setting: i) their local clocks, ii) the playout buffer dimensions, iii) the corresponding values in their playout/buffering policies. This method guarantees that both the introduced additional buffering delays and the buffer dimensions are always proportioned to the traffic conditions, and the talkspurt playout delay may be dynamically updated from one talkspurt to the next provided that intervening silent intervals of sufficiently long duration are exploited for performing those adjustments [18]. A possible problem with the mentioned policy is related to the possible high value for the obtained RTT that may be caused by the fact that either the probe packet or the response packet (during the first two phases of the synchronization operation) has suffered from a very high delay spike. Due to that, a very high value for the playout delay would be introduced, thus impairing the interactivity of the audio conversation. This problem has been solved in [18] by adopting a policy which was inspired by the delay spike detection and management mechanism proposed in [15].

3 Comparing the two Mechanisms with TwoTowers

The two adaptive playout delay adjustment mechanisms informally described in the previous section have been formally modeled with EMPA$_{\mathrm{vp}}$ in order to compare their performance with TwoTowers. Due to lack of space, the EMPA$_{\mathrm{vp}}$ specifications of the two audio mechanisms can be found in [2].

Before presenting the outcome of the comparison, we would like to point out that developing formal descriptions for the two audio mechanisms has allowed us to formally verify properties concerned with the correctness of the mechanisms themselves. As a simple example, we have proved that the mechanism #1 and the three-way handshaking mechanism of the synchronization phase of the mechanism #2 are deadlock free.

As far as the QoS comparison is concerned, because of the presence of nonexponential delays, the analysis has been conducted via simulation. We have evaluated the two mechanisms using both experimentally obtained delay measurements of three 10 min long audio conversations between two different Internet sites (trace driven simulation) and two 30 sec long audio conversations randomly generated according to Gaussian distributed delays and exponentially distributed

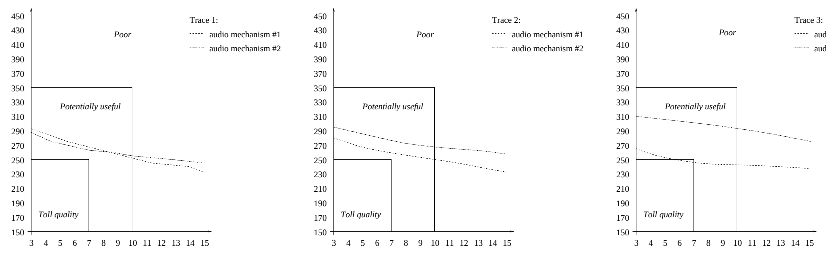

Figure 1: Average playout delay (msec) vs. loss rate (%): trace driven simulation with playout delay spike mechanism deactivated

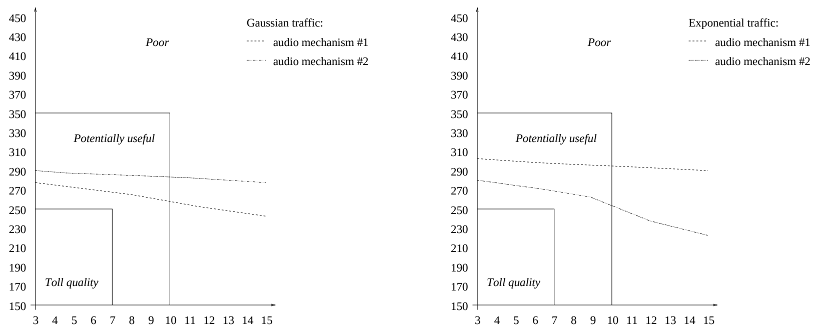

Figure 2: Average playout delay (msec) vs. loss rate (%): randomly generated traffic with playout delay spike mechanism deactivated

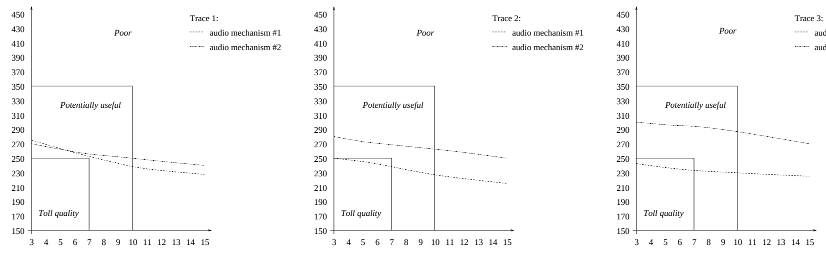

Figure 3: Average playout delay (msec) vs. loss rate (%): trace driven simulation with playout delay spike mechanism activated

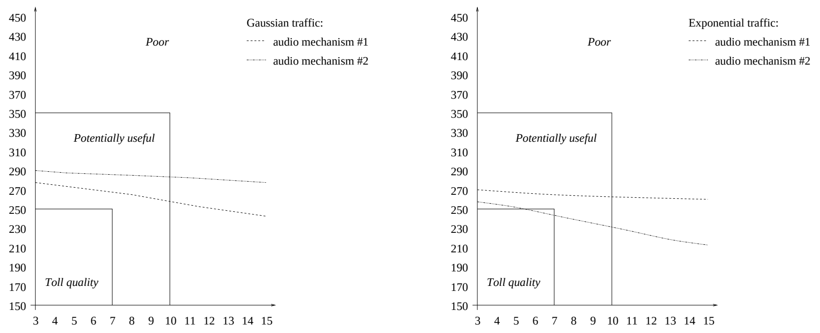

Figure 4: Average playout delay (msec) vs. loss rate (%): randomly generated traffic with playout delay spike mechanism activated

delays, respectively. The two randomly generated traffics are intended as limiting scenarios while the three traces represent intermediate, more realistic scenarios. Fig. 1, 2, 3, and 4 show a mapping from average playout delay and loss rate to QoS for the five above mentioned scenarios, respectively, where Gaussian delays have expected value 100 msec and standard deviation 7 msec, exponential delays have expected value 100 msec (and standard deviation 10 msec), and mechanism #1 has been simulated with $k = 4$. The former pair of figures show the result when the playout delay spike mechanism is deactivated for both audio mechanisms, while the latter pair of figures show the improvement of the QoS for both audio mechanisms when activating the playout delay spike mechanism [16, 18]. In order to provide the reader with an understanding of the effect that various playout delays and loss rates (as well as buffer dimensions) have on the quality of the perceived audio, we have reported in the figures an approximate and intuitive representation of three different ranges of the quality of the perceived audio adopted from [18]: toll quality for delays of less than 200-250 msec and low loss rates, potentially useful for delays of about 300-350 msec and higher loss rates, and poor for delays larger than 350 msec and very high loss rates. We observe that the two audio mechanisms, although based on quite different design principles, are both reasonable since they basically provide comparable QoS. The audio mechanism #1 outperforms the audio mechanism #2 in the case of Gaussian traffic, while the reverse is true in the case of exponential traffic. We wish to point out that the worse performance of the audio mechanism #2 in the second and third trace driven simulations is probably due to the fact that the related traces exhibit a regular delay pattern with few spikes which is quite close to a Gaussian traffic. Using TwoTowers has thus been very helpful to detect that neither of the two audio mechanisms is better than the other one (w.r.t. the considered metric) in general, because their performance depends on the traffic conditions. We conclude by pointing out that, to conduct a comprehensive analysis, other metrics (such as those discussed in [16, 18]) should be taken into account. This is left for future work.

References

[1] A. Aldini, M. Bernardo, R. Gorrieri, *"An Algebraic Model for Evaluating the Performance of an ATM Switch with Explicit Rate Marking"*, to appear in Proc. of the *7th Int. Workshop on Process Algebras and Performance Modeling (PAPM '99)*, Zaragoza (Spain), 1999

[2] A. Aldini, M. Bernardo, R. Gorrieri, M. Roccetti, *"Comparing the QoS of Internet Audio Mechanisms via Formal Methods"*, Research Report n. UBLCS-99-04, Laboratory for Computer Science, University of Bologna (Italy), 1999

[3] M. Bernardo, *"Theory and Application of Extended Markovian Process Algebra"*, Ph.D. Thesis, University of Bologna (Italy), 1999 (`http://www.cs.unibo.it/~bernardo/`)

[4] M. Bernardo, W.R. Cleaveland, S.T. Sims, W.J. Stewart, *"TwoTowers: A Tool Integrating Functional and Performance Analysis of Concurrent Systems"*, in Proc. of the *IFIP Joint Int. Conf. on Formal Description Techniques for Distributed Systems and Communication Protocols and Protocol Specification, Testing and Verification (FORTE/PSTV '98)*, Kluwer, 457-467, Paris (France), 1998

[5] M. Bernardo, L. Donatiello, R. Gorrieri, *"A Formal Approach to the Integration of Performance Aspects in the Modeling and Analysis of Concurrent Systems"*, Information and Computation 144:83-154, 1998

[6] M. Bernardo, R. Gorrieri, M. Roccetti, *"Formal Performance Modelling and Evaluation of an Adaptive Mechanism for Packetised Audio over the Internet"*, Formal Aspects of Computing 10(4):313-337, 1998

[7] M. Bravetti, M. Bernardo, R. Gorrieri, *"Towards Performance Evaluation with General Distributions in Process Algebras"*, in Proc. of the *9th Int. Conf. on Concurrency Theory (CONCUR '98)*, LNCS 1466:405-422, Nice (France), 1998

[8] P.R. D'Argenio, J.-P. Katoen, E. Brinksma, *"An Algebraic Approach to the Specification of Stochastic Systems"*, in Proc. of the *IFIP Working Conf. on Programming Concepts, Methods and C alculi (PROCOMET '98)*, Chapman & Hall, pp. 126-147, Shelter Islands (NY), 1998

[9] N. Götz, U. Herzog, M. Rettelbach, *"Multiprocessor and Distributed System Design: The Integration of Functional Specification and Performance Analysis Using Stochastic Process Algebras"*, in Proc. of the *16th Int. Symp. on*

Computer Performance Modelling, Measurement and Evaluation (PERFORMANCE '93), LNCS 729:121-146, Rome (Italy), 1993

[10] V. Hardman, M.A. Sasse, I. Kouvelas, *"Successful Multi-Party Audio Communication over the Internet"*, Comm. of the ACM 41:74-80, 1998

[11] J. Hillston, *"A Compositional Approach to Performance Modelling"*, Cambridge University Press, 1996

[12] T.J. Kostas, M.S. Borella, I. Sidhu, G.M. Schuster, J. Grabie c, J. Mahler, *"Real-Time Voice over Packet-Switched Networks"*, in IEEE Network 12:18-27, 1998

[13] ITU-T Recommendation G.729 - G.723.1, 1996

[14] R. Milner, *"Communication and Concurrency"*, Prentice Hall, 1989

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

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

[17] M. Roccetti, M. Bernardo, R. Gorrieri, *"Packetized Audio for Industrial Applications: a Simulation Study"*, in Proc. of the *10th European Simulation Symposium (ESS'98)*, SCS, 495-500, Nottingham (UK), 1998

[18] M. Roccetti, V. Ghini, G. Pau, P. Salomoni, M.E. Bonfigli, *"Design and Experimental Evaluation of an Adaptive Playout Delay Control Mechanism for Packetized Audio for Use over the Internet"*, Multimedia Tools and Applications, an International Journal, accepted for publication, 1999

[19] H. Schulzrinne, *"Voice Communication across the Internet: a Network Voice Terminal"*, Tech. Rep., University of Massachusetts, Amherst (MA), 1992

Biography

Alessandro Aldini received the Laurea degree (with honors) in Computer Science from the University of Bologna (Italy) in 1998. He is currently a PhD student in Computer Science at the Computer Science Department of the University of Bologna. His research interests are in the field of: theory of concurrency, formal description techniques and tools for concurrent and distributed computing systems, stochastic processes, performance evaluation and simulation.

Marco Bernardo received the Laurea Degree in Computer Science and the Ph.D. in Computer Science from the University of Bologna (Italy) in 1994 and 1999, respectively. He is currently a Research Assistant at the Department of Computer Science of the University of Bologna. His research interests are in the field of formal methods, concurrency theory, and performance evaluation. He has developed the software tool TwoTowers for the functional and performance modeling and analysis of concurrent systems.

Roberto Gorrieri received the Laurea degree in Computer Science in 1986 and the PhD in Computer Science in 1991, both from the University of Pisa, Italy. Since November 1992, he is Associate Professor of Computer Science at the University of Bologna, Italy. Roberto Gorrieri is currently a member of the Board of the European Association for Theoretical Computer Science (EATCS) and chairman of IFIP WG 1.7 on Theoretical Foundations of Security. His research interests include: theory of concurrent and distributed systems, formal methods for security, real time and performance evaluation.

Marco Roccetti received the Italian Laurea degree in Electronic Engineering from the University of Bologna (Italy) in the academic year 1987/88. Since November 1998, he is an Associate Professor of Computer Science at the Department of Computer Science of the University of Bologna. From 1990 through October 1998, he was with the Department of Computer Science as a Research Associate. His research interests include: design, implementation and evaluation of multimedia computing and communication systems, and performance analysis and simulation of distributed and parallel computing systems. Marco Roccetti is a member of the Society for Computer Simulation International.