Packetized Audio for Industrial Applications: A Simulation Study

Marco Roccetti Marco Bernardo Roberto Gorrieri

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

Abstract

A mechanism for transmitting packetized audio over wide area, shared networks is presented and its performance is studied via simulative analysis. The proposed mechanism does not need "in advance" allocation of either network bandwidth or computing resources, but maintains satisfiable values of the tradeoff between the audio packet playout delay and the packet loss due to late arrivals. Because of this, the proposed mechanism may be profitably integrated in industrial environments with other hard real time applications for which, instead, resource allocation is desirable. The performance of the proposed mechanism (i.e. the percentage of audio packets that arrive on time for being played out at the destination during an audio communication) is analyzed via simulation by a stochastically timed process algebra based software tool termed TwoTowers.

Keywords: packet audio, simulation tools, performance evaluation, process algebras.

1 INTRODUCTION

In industrial environments such as power plants, automated factories, and railways (as well as in other relevant scenarios such as geographically scattered electronic auction bidding systems and teleconferencing systems for cooperative medicine), digital audio and video based computer applications have emerged as useful tools for the support of many important activities [8, 11]. In the above cited contexts, digital audio and video play, at least, the following important roles: i) real-time communication (the use of digital audio and video may permit to human workers to cooperate more profitably), ii) monitoring and supervision (audio recorders and video cameras provide audio and visual record for use for security purposes), iii) sensors for plant control and management (audio and video processing are increasingly being used in the automated activities of the plant), iv) educational and training activities (audio and video documentation may be profitably used for on-line documentation of the human operators). In the above mentioned settings, several different functions have to be integrated in the same computer network such as: traditional applications transporting non real time data, control applications with hard real time requirements, and continuous media such as digital audio and video. In addition, the problem of the integration of the above mentioned different functions and applications may be notably exacerbated by the need to provide those functions to systems that can typically extend across wide geographical distances (through inter- or intranetworks) and incorporate and master a possibly large set of physical and logical resources. Hence, in an industrial environment it may seem desirable to dedicate a relevant portion of total network bandwidth, computing resources, and buffer space to the traffic with hard real time requirements, and to allow audio and video applications (typically characterized by soft real time deadlines) to use the remainder of the network bandwidth and computing resources. Thus, bandwidth, computing resources, and buffer space for very critical connections (e.g. hard real time control and some critical video applications) may be allocated in advance (for example by using, in an IP based framework, new protocol suites such as the Resource Reservation Protocol RSVP [17]), while the remainder of the resources may be used on demand and shared "fairly" among competing users and applications.

In this framework, the purpose of this paper is twofold. First, we briefly introduce in Sect. 2 an efficient mechanism for transmitting packetized audio that does not need "in advance" allocation of network bandwidth, computing resources, and buffer space, while maintaining satisfiable values of the tradeoff between the audio packet playout delay and the packet loss due to late arrivals [14]. Owing to that, our mechanism may be transparently and profitably integrated (for example over an IP based platform) with other hard real time applications for which, instead, resource allocation is needed in order to satisfy their deadlines. Second, we show in Sect. 3 how the performance of such a proposed mechanism may be predicted by using a software tool (termed TwoTowers) which is based on the stochastically timed process algebra EMPA [1, 2]. The main motivation behind the use of the above mentioned algebraic formalism in order to describe and analyze the audio mechanism is that it has permitted an easier and faster modelling activity, thanks to its compositionality.

2 PACKETIZED AUDIO: A BACKGROUND

Packetized audio applications have currently turned out into tools that many Internet (and intranet) users try to use frequently. For example, the audio conversations of many international conferences and workshops are now usually transmitted over the Mbone, an experimental overlay network of the Internet [9]. Typically, an audio segment may be considered as being constituted of "talkspurt" periods, during which the audio activity is carried out, and "silence" periods, during which no audio packets are generated. In order for the receiving site to reconstruct the audio conversation, the audio packets of a talkspurt must be played out in the order they were emitted at the sending site. If the delay between the arrival of subsequent packets is constant (i.e. there is no jitter), a receiving site may simply play out the arriving audio packets as soon as they are received. However, this is only rarely the case, since jitter free, ordered, on-time packet delivery almost never occurs in the today's packet switched networks. Those variations in the arrivals of subsequent packets strongly depend on the traffic conditions of the underlying network. Packet loss rates (due to the effective loss and damage of packets as well as late arrivals) have been measured over the Internet as varying between 15% and 40% [10]. In addition, extensive experiments with wide-area network testbeds have shown that the delays between consecutive packets may also be as much as 1.5 seconds, thus impairing real time interactive human conversations. Hence, the most used approach adopted by all the most popular audio tools (e.g. NeVot [15], vat [7], the INRIA audio tool [5]) consists in adapting the audio applications to the jitter present on the network. In order to compensate for these variable delays, a "smoothing" playout buffer is used at the receiver. Received audio packets are first queued into the buffer, and the playout of a packet of a given talkspurt is delayed for some quantity of time beyond the reception of the first packet of that talkspurt. Needless to say a crucial tradeoff exists between the length of the imposed playout delay and the amount of lost packets due to their late arrival: the longer the additional playout 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. Typical acceptable values for the end-to-end delay between packet audio generation at the sending site and its playout time at the receiver are in the range of 300-400 msec, instead a percentage of no more than 5 - 10% of packet loss is considered quite tolerable in human conversations [4].

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. In order to implement this playout control policy, all the above cited audio applications make use of the two following assumptions: i) 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); ii) the delays experienced by audio packets on the network follow a Gauss 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 $p_i = t_i + \hat{d}_i + 4 \cdot \hat{v}_i$, where: t_i is the time instant in which the audio packet i is generated at the sending site; $\hat{d}_i$ is the average value of the 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), and finally $\hat{v}_i$ is the variation of the playout delay. Note that the estimation of both the average delay and the average delay variation are carried out using the well known *stochastic gradient algorithm* [12]. Instead, each subsequent packet (say j) of the talkspurt (different from the first one i) is played out by using the following playout time $p_j = p_i + t_j - t_i$.

Extensive experiments have been carried out that have shown that the above mentioned playout delay control mechanism may be adequate to obtain acceptable values for the tradeoff between the average playout delay and the packet loss due to late packet arrivals. However, in some circumstances, the above mechanism may suffer from the following problems, especially whether it is deployed over wide area networks [10]. First, the external software-based mechanisms (e.g. the Internet based NTP protocol) used to maintain the system clocks at both the sending and the receiver synchronized are not so widespread (especially in an industrial framework), and, in addition, they may turn out to be too much inaccurate to cope with the real time nature of the audio generation/playout process. Second, the widely adopted assumption that the packet transmission delays follow a Gauss distribution seems to be a plausible conjecture only for those limited time intervals in which the overall load of the underlying network is quite light. Indeed, recent experimental studies carried out over the Internet have indicated the presence of frequent and conspicously large end-to-end delay "spikes" for periodically generated packets (as is the case with audio packets) [4]. Finally, even in the cases when the delays may be adequately modeled with a Gauss distribution, the computational effort and complexity which are necessary for elaborating the equations provided above may be too large, and thus impact negatively on the real time playout process (especially when the receiving host is under heavy load).

3 PACKETIZED AUDIO: A NOVEL MECHANISM

Recently, a new adaptive algorithm for the control of the playout delay has been proposed that ameliorates all the negative effects of the audio mechanisms described above, while maintaining satisfiable values of the tradeoff between the average playout delay and the packet loss due to late arrivals [14]. This mechanism does not assume a Gauss distribution for the end-to-end transmission delays of the audio packets and, in addition, provides: i) an "internal" and accurate technique that maintains tight time synchronization between the system clocks of both the sending and the receiving hosts, ii) a method for adaptively estimating the audio packet playout time (on a per-talkspurt basis) with an associated minimal computational complexity; iii) an exact and simple method for dimensioning the playout buffer depending on the traffic conditions on the underlying network.

A simplified description of the mechanism is as follows (more details may be found in the already cited paper [14]). When the sender transmits the first packet of an audio talkspurt, the sender timestamps that packet with the value (say C) of the reading of its own clock. When this first packet arrives at the receiver, the receiver also sets its clock to the C value and immediately schedules the presentation of that first packet. Subsequent audio packets belonging to the same talkspurt are also timestamped at the sender with the value of the reading of the sender's clock at the time instant when the packet is transmitted. When those subsequent packets arrive at the receiving site, their attached timestamp is compared with the value of the reading of the receiver's clock. If the timestamp attached to the packet is equal to the value of the clock of the receiver, that packet is immediately played out. If the timestamp attached to the packet is greater than the value of the clock of the receiver, that packet is buffered and its playout time is scheduled after a time interval equal to the positive difference between the value of the timestamp and the actual value of the receiver's clock. Finally, if the timestamp attached to the packet is less than the value of the clock of the receiver, the packet is simply discarded since it is too late for presentation. However, due to the fluctuant delays in real transmissions, the value of the clocks of the sender and the receiver at a given time instant may differ of the following quantity $T_S - T_R = \Delta$, where T_S and T_R are, respectively, the readings of the local clocks at the sender and at the receiver, and Δ is a non negative quantity ranging between 0, a theoretical lower bound, and Δ_{max}, a theoretical upper bound on the transmission delays introduced by the network between the sender and the receiver.

Hence, a crucial issue of the mechanism is an accurate dimensioning of the playout buffer [13]. Both buffer underflow and overflow may result in discontinuity in the playout process. The worst case scenario for buffer underflow (corresponding to the case when packets arrive too late for presentation) is clearly when the first packet arrives after a minimum delay (e.g. 0) while a subsequent packet arrives with maximum delay (e.g. Δ_{max}). It is possible to show [14] that in this case this subsequent packet (transmitted by the sender when its clock shows a time value equal to X, and consequently timestamped with the value X) may arrive at the receiver too late for playout, precisely when the receiver's clock shows the value given by $X + \Delta_{max}$. This consideration suggests that a practical and secure method for preventing from buffer underflow is that the receiver delays the setting of its local clock of an additional quantity equal to Δ_{max}, when the first packet of the talkspurt is received. With this simple modification, the policy guarantees that all the audio packets of the talkspurts that will suffer from a transmission delay not greater than Δ_{max} will be on-time for playout.

However, the above mentioned technique introduces another problem: that of playout buffer overflow. The worst case scenario for buffer overflow occurs in the following circumstance: the first packet of a talkspurt suffers from the maximum delay (i.e. Δ_{max}), instead a subsequent audio packet experiences mininum delay (e.g. 0). It is possible to show [14] that in such a case the subsequent audio packet (transmitted at time X) may arrive at the receiving site when the receiver's clocks is equal to $X - 2 \cdot \Delta_{max}$. In conclusion, this example dictates that the playout buffer dimension may never be less than the maximum number of bytes that may arrive in an interval of $2 \cdot \Delta_{max}$.

However, from the discussion above, two problems have been left unresolved: 1) the estimation of the maximum experienced delay Δ_{max} (or the evaluation of an uper bound on it); 2) the adaptation of the proposed playout control mechanism in order to compensate for the highly fluctuant end-to-end delays experienced over wide area, packet switched networks.

In the mechanism proposed in [14], a technique has been devised that is used to estimate an upper bound for the maximum delay transmission. This technique exploits the so called Round Trip Time (RTT) and is based on a three-way handshake protocol. It works as follows. Prior to the beginning of the first talkspurt in an audio conversation, a "probe" packet is sent from the sender to the receiver timestamped with the clock value of the sender (say C). At the reception of this probe packet, the receiver sets its own clock with the value of the the timestamp attached to the probe packet, and sends immediately back to the sender a "response" packet with attached the same timestamp C. Upon the receiving of this response packet, the sender computes the value of the RTT by subtracting from the current value of its local clock the value of the time stamp C. At that moment, the difference between the two clocks, respectively at the sender and at the receiver, is equal to an unknown quantity (say t_0) which may range from a theoretical lower bound of 0 (that is all the RTT value has been consumed on the way back from the receiver to the sender), and a theoretical upper bound of RTT (that is all the RTT has been consumed on the way in during the transmission of the probe packet). Unfortunately, as already mentioned, t_0 is unknown and a scarce approximation of this value (e.g. $t_0 = RTT/2$) might result in both playout buffer underflow problems and packet loss due to late arrivals. Based on these considerations, in the proposed mechanism [14], the sender, after having received the response packet from the receiver and calculated the RTT value, sends to the receiver a final "installation" packet, with attached the previously calculated RTT value. Upon receiving this installation packet, the receiver sets the time of its local clock, by subtracting from the value shown at its clock at the arrival of the "installation" packet the value of the transmitted RTT. Hence, at that precise moment, the difference between the two clocks at the receiver and at the sender is equal to a value Δ given by $\Delta = T_S - T_R = t_0 + RTT$, where Δ ranges in the interval $[RTT, 2 \cdot RTT]$ depending on the unknown value of t_0, which, in turn, may range in the interval $[0, RTT]$. In order for the proposed policy to adaptively adjust to the highly fluctuant end-to-end delays experienced over wide area, packet switched networks (like the Internet), the above mentioned "synchronization" technique 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 about 1 second).

Hence, each time a new value for RRT is computed by the sender, it may be used by the receiver for adaptively setting the value of its local clock and the playout buffer dimension, as mentioned above. This method guarantees that both the introduced additional playout time and the buffer dimension are always proportioned to the traffic conditions.

4 USING TWOTOWERS TO PREDICT THE PERFORMANCE OF THE MECHANISM

The adaptive mechanism [14] introduced in the previous section has been formally modeled with the stochastically timed process algebra EMPA [2] in order to predict in advance its performance.

Stochastically timed process algebras constitute an emerging field in the area of the integration of formal methods and performance evaluation. They are algebraic languages endowed with a small set of powerful operators (such as sequential composition, alternative composition, parallel composition, and recursion) as well as a family of actions which permit to express both the type and the duration of the activities executed by the systems being modeled. Because of their algebraic structure, they naturally support compositional modeling thus becoming an attractive alternative to well known formalisms such as stochastically timed Petri nets.

EMPA is a stochastically timed process algebra characterized by a considerable expressive power. be-

exp.	estimate	90% confidence interval	estimate	90% confidence interval
1	76.527147	[76.329880, 76.724415]	96.702746	[96.335040, 97.070452]
2	73.829673	[73.684253, 73.975093]	96.712566	[96.490167, 96.934966]
3	71.786939	[71.606600, 71.967279]	96.453736	[96.183720, 96.723751]
4	71.598548	[71.406887, 71.790209]	96.583136	[96.243839, 96.922433]
5	78.935170	[78.775455, 79.094885]	96.600180	[96.296200, 96.904160]
6	70.829697	[70.626774, 71.032620]	96.612087	[96.282160, 96.942014]
7	72.368819	[72.159914, 72.577724]	96.481827	[96.183756, 96.779897]
8	78.268155	[78.073999, 78.462310]	96.594055	[96.233550, 96.954561]
9	74.422198	[74.214966, 74.629431]	96.480821	[96.154645, 96.806997]
10	74.518115	[74.290436, 74.745794]	96.745196	[96.461490, 97.028903]

Table 1: Simulation results

cause of the fact that not only exponentially timed actions can be expressed, but also prioritized weighted immediate actions as well as actions whose duration is unspecified. Every process term is equipped with three semantic models used for analysis purposes: an integrated semantic model (state transition graph with action labeled edges), a functional semantic model (state transition graph with action type labeled edges), and a performance semantic model (Markov chain). Moreover, algebraic terms can be compared by means of a suitable notion of equivalence accounting for both functional and performance aspects.

EMPA is the language adopted in TwoTowers [1], a software tool for the functional and performance analysis of concurrent systems which implements the integrated approach proposed in [2]. TwoTowers is composed of a tool driver (including routines for parsing EMPA specifications and performing lexical, syntactic, and static semantic checks on the specifications), an integrated kernel, a functional kernel, and a performance kernel.

The purpose of the integrated kernel is to perform those analyses that require both functional and performance information and therefore cannot be performed by factoring out information to be passed to either the functional or performance kernels. The integrated kernel contains the routines to generate the integrated semantic model of correct EMPA specifications, as well as a routine to check two specifications for equivalence and a facility for simulating specifications in order to derive performance measures.

The functional kernel generates the functional semantic model of correct EMPA specifications. The analysis of the models as well as the terms themselves is then executed by a version of the Concurrency Workbench of North Carolina (CWB-NC) [6] that was retargeted for EMPA. The functional kernel therefore comprises all the analysis capabilities of the CWB-NC including interactive simulation of systems, reachability analysis to check safety properties, model checking in the μ-calculus or CTL, equivalence checking, and preorder checking.

Finally, the performance kernel generates the performance semantic model of correct EMPA specifications. The analysis of the Markov chains is then carried out by means of the Markov Chain Analyzer (MarCA) [16]: transient and stationary probability distributions are calculated, then performance indexes are derived using rewards.

The adaptive mechanism of [14] has been modeled with EMPA both in the case of initial synchronization only of the sender's and receiver's clock and in the case of periodic synchronization taking place every second (algebraic specifications can be found in [3]). Afterwards, we have conducted with the integrated kernel of TwoTowers a simulative analysis of the performance of the two scenarios represented by the two specifications, in order to compute the percentage of packets that arrive at the destination on time for being played out. More precisely, we have carried out 10 experiments, each consisting of 20 simulation runs concerning a period of 30 seconds. The results are in Table 1, where also 90% confidence intervals are shown (the second and third columns refer to initial synchronization, whereas the fourth and the fifth columns refer to periodic synchronization). As it can be seen, about 25% of packets are discarded in case of initial synchronization only, while in the periodic synchronization case the percentage of discarded packets falls down 4%.

We conclude by pointing out that a prototype implementation of the playout control mechanism has been carried out, using the Unix socket interface and the datagram based UDP protocol. Using such a prototype implementation, an initial extensive experimentation has been conducted in order to test the real performance of the mechanism. In particular, several experiments of audio conversations have been carried out during daytime using a SUN workstation SPARC 5 situated at the Laboratory of Computer Science of Cesena (a remote site of the Department of Computer Science of the University of Bologna (Italy)) and a SUN workstation SPARC 10 situated at the CERN Institute in Geneve (Switzerland). During the experiments, a percentage of no more than 4-5% of loss audio packets was observed due to late arrivals. In order to test the efficacy of the proposed method, also contemporary experiments were conducted on the same testbed without using any playout control mechanism. In those experiments, a percentage of lost audio packets (due to late arrivals) was experienced ranging from about 10% to almost 40% depending on the traffic conditions. As it can be noted, the results obtained from the simulative analysis of the algebraic model of the mechanism correctly predicted the actual performance of the mechanism itself measured on the field. This highlights the importance of using formal description techniques in the initial phase of the design of an algorithm, in order to timely detect errors or inefficiency which may result in cost increases if discovered too late.

Acknowledgements

This research has been partially funded by Italian MURST, Italian CNR under the grant Progetto Strategico "Modelli e Metodi per la Matematica e l'Ingegneria", and Italian CNR through the grant n. 98.00387.CT12. Finally, the first author would like to thank Alessandro Cantelli (University of Bologna) for his technical assistance during the audio experiments discussed in the paper.

References

[1] M. Bernardo, R. Cleaveland, S. Sims, W. Stewart, 1998, "TwoTowers: A Tool Integrating Functional and Performance Analysis of Concurrent Systems", to appear in Proc. of FORTE/PTSV'98, Paris (France)

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

[3] M. Bernardo, R. Gorrieri, M. Roccetti, 1998, "Formal Performance Modelling and Evaluation of an Adaptive Mechanism for Packetized Audio over the Internet", to appear in *Formal Aspect of Computing*

[4] J. Bolot, H. Crepin, A. Vega Garcia, 1995, "Analysis of Audio Packet Loss on the Internet", in Proc. of *Network and Operating System Support for Digital Audio and Video*, pp. 163-174, Durham (NC)

[5] J. Bolot, A. Vega Garcia, 1996, "Control Mechanism for Packet audio in the Internet", in Proc. of *IEEE SIGCOMM '96*, San Francisco (CA)

[6] W.R. Cleaveland, S. Sims, 1996, "The NCSU Concurrency Workbench", in Proc. of the *8th Int. Conf. on Computer Aided Verification (CAV '96)*, LNCS 1102:394-397, New Brunswick (NJ)

[7] V. Jacobson, S. McCanne, *vat*, ftp://ftp.ee.lbl.gov/conferencing/vat/

[8] H.C. Lauer, *et al.*, 1997, "Digital Audio and Video in Industrial Systems", Technical Report of the Mitsubishi Electric Research Labs., Cambridge (MA)

[9] M. Macedonia, D. Brutzmann, 1994, "Mbone Provides Audio and Video across the Internet", *IEEE Computer Magazine* 21:30-35

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

[11] F. Panzieri, M. Roccetti, 1997, "Synchronization Support and Group-Membership Services for Reliable Distributed Multimedia Applications", *ACM Multimedia Systems* 5:1-22

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

[13] P.V. Rangan, S.S. Kumar, S. Rajan, 1996, "Continuity and Synchronization in MPEG", *IEEE Journal on Selected Areas in Communications* 14:52-60

[14] M. Roccetti, V. Ghini, P. Salomoni, V. Pau, M.E. Bonfigli, 1998, "Design, Development and Experimentation of an Adaptive Mechanism for Reliable Packetized Audio for Use over the Internet", submitted for publication

[15] H. Schulzrinne, 1992, "Voice Communication across the Internet: a Network Voice Terminal", Technical Report, Dept. of ECE and CS, Univ. of Massachusetts, Amherst (MA)

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

[17] L. Zhang, 1993, "RSVP: a New Resource Reservation Protocol", *IEEE Network Magazine* 7:8-18