

A simulation study of adaptive voice communications on IP networks

A. Barberis^a, C. Casetti^a, J.C. De Martin^b, M. Meo^{a,*}

^aDipartimento di Elettronica, Politecnico di Torino, 10129 Torino, Italy

^bIRITI-CNR, Politecnico di Torino, Torino, Italy

Received 6 November 2000; accepted 6 November 2000

Abstract

This paper presents simulation results outlining the behavior of rate-adaptive voice communications over IP networks. In the considered architecture, voice coders adapt their rate to the current state of the network so as to generate only the bandwidth that the network is capable of carrying. An algorithm is proposed for driving the transmission rate of voice sources on the basis of estimations of the network conditions, measured in terms of packet delays and losses.

The effectiveness of the proposed solution is then investigated in various scenarios which comprise: (i) a dedicated network in which the available bandwidth is exclusively shared between adaptive voice connections; (ii) a scenario in which adaptive voice sources compete with other TCP-like sources; and (iii) an uncontrolled network environment. We have compared the performance of the rate-adaptive against the non-adaptive (i.e. fixed-rate) approach for the transport of voice over IP. Using a rate-adaptive approach, more voice communications can be carried while maintaining a good quality of service, even on non-segregated networks. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: IP telephony; Voice coding; Simulation

1. Introduction

Although the value of integrating voice and data transport on the same network has been perceived for many years now, it is only recently, with the huge and still rapidly growing expansion of IP networks, that the issue has seized the attention of individuals, companies and academic institutions worldwide. To accomplish such integration, voice, so far transported predominantly by circuit-switched networks, is digitized, the resulting samples compressed to save bandwidth, then packetized and sent through the network. At the receiver side, voice packets are reordered, decompressed and then played back to the listener.

Several challenging problems need to be solved to achieve the same level of quality offered by the traditional PSTNs in the context of a packet network like the Internet.

Delay: A first major issue is that of delay. In a generic IP network, in fact, delays are neither bounded nor predictable. Interactive, two-way telephony, however, imposes strict limits on the allowable overall mouth-to-ear delay. Delay contributions due to the network — primarily, propagation time and average time spent in router queues — can be

controlled by ad hoc engineering of the network. Other contributions to the overall delay depend, instead, on the specific voice over IP (VoIP) architecture employed: parameters like packet size, speech coding algorithm and playout buffer size all play a role in determining the final delay budget.

Delay variations: Even in the context of carefully engineered networks, however, delay remains variable. The effects of such variability can be controlled by using a buffer, commonly called the *playout buffer*, that allows the receiver to wait for all the packets arriving within an acceptable time limit. A smooth playback of the incoming speech signal is then possible.

Packet losses: A third issue is that, in networks based on the best-effort model, packets can be lost. Generally, packet losses occur when a congested router drops the tail of its queues. Effective ways to replace the missing speech packets at the receiver are, therefore, needed.

Speech compression: Transport of voice over data networks is generally made possible by speech compression techniques. Typical bit rates range from 64 kb/s for companding PCM to 2.4 kb/s for some vocoders. Each speech coding algorithm implements a different trade-off between output speech quality, algorithmic delay, bit rate, computational complexity and robustness to background noise. The optimal speech coder for VoIP depends on the

* Corresponding author.

E-mail addresses: casetti@polito.it (C. Casetti), demartin@polito.it (J.C. De Martin), michela@polito.it (M. Meo).

specific scenario. No speech coding standard has yet been developed with VoIP as the primary application. Usually, fixed-rate speech coders are employed. Such coders generate a constant output bit rate, independently of the network conditions. If the network cannot sustain the traffic, the packets will be delayed and/or dropped.

The overall objective of voice transmission over IP networks is to find an array of technical solutions to these challenges that guarantees the desired level of perceptual quality under most network conditions.

This paper focuses on voice coders whose rate adapts to the current state of the network. If the network is congested, speech is coded at lower bit rates; if the network, instead, is lightly loaded, the speech coder is allowed to operate at higher bit rates. This approach, which may be called adaptive voice over IP (AVoIP), relies on variable bit-rate speech coders to generate only the bandwidth that the network is capable of carrying with a given quality of service (QoS) at any instant in time.

AVoIP has several appealing features. Firstly, an efficient use of network resources is granted since AVoIP applications do not need a rigid partitioning of the link bandwidth (as plain VoIP does); instead, AVoIP can exploit *statistical* rather than *deterministic* QoS guarantees: as such, it is a natural candidate for Intserv/Diffserv networks [1]. Also, by adapting gracefully to network congestion, AVoIP flows leave room for signaling and critical in-band flow management sharing the same network facilities. Furthermore, operators can enjoy a great flexibility: by trading off perceived QoS and monetary cost, they can increase the range of services they are offering, e.g. fixed-high-bit-rate calls, fixed-low-bit-rate calls, adaptive bit-rate calls, etc. Finally, new services can be implemented, especially in environments where both data and voice share a narrowband link. For example, technical support being provided through a modem link could use AVoIP to allow customer–salesperson interaction while patches are uploaded on the customer's workstation.

A contribution of this paper is the effort to quantify the performance of AVoIP in terms of those measures which have a major impact on the perceptual QoS. Typical metrics such as average bit rate and packet loss probability are thus considered, as well as more sophisticated statistics on packet losses.

The remaining part of this paper is organized as follows: Section 2 explains the rationale for AVoIP and describes a rate control algorithm. Section 3 summarizes the issues affecting perceptual QoS and the measurements of AVoIP performance. The simulation setup is described in Section 4 and, finally, the results are shown in Section 5.

2. The case for 'light-weight' congestion control algorithms in IP telephony

The issue of flow and congestion control for real-time

traffic is a delicate one: application-layer coding algorithms and lower-layer rate control schemes should cooperate to respond to critical situations such as network congestion. While a window-based flow control is hardly viable since it would result in stop-and-go source behavior, a 'light-weight', QoS-aware congestion control scheme that drives the source encoding rate has the potential to lessen the impact of unruly real-time traffic.

The main goal of such an algorithm is reducing the load on the network when congestion (i.e. queue build-up at the intermediate nodes) occurs. If a large part of the network traffic is composed of variable-bit-rate regulated voice traffic and the algorithm regulating each source is devised so as to react to specific 'warnings' such as increasing delay or sudden packet loss, then a control strategy may successfully reduce the occurrence of congestion.

The topic of variable-rate-adaptive schemes for multimedia streams has been addressed by several researchers in the past. In a general framework [2,3], Shenker compared the benefits of rigid and adaptive applications, pointing out that the rate-adaptive applications are more robust to network congestion than the other classes of adaptive applications (i.e. the delay-adaptive applications, which are tolerant of occasional delay bound violations, but whose generation rate is independent of the network congestion). In Ref. [4], Cox and Crochiere foreshadow adaptive schemes for variable-bit-rate coders by analyzing the behavior of a buffer control mechanism that has sources react to buffer overflows and underflows. The idea of an end-to-end feedback control is already present in the work by Bially et al. [5], where network-driven voice sources based on embedded speech coding schemes are studied, although the involvement of the intermediate nodes is still required; in Ref. [6], Yin and Hluchyj developed an analytical model for an end-to-end feedback control mechanism for variable bit-rate coders. Bolot and Turletti in Ref. [7] proposed and tested an adaptive video coder based on the state of the network while in Ref. [8], they introduced a polling mechanism that allowed a real-time content transmitter to elicit feedback information from the receivers and use it to control its own rate.

Similar in spirit to our approach, the works by Busse et al. [9] and Sisalem and Schulzrinne [10] employ a sender-based packet loss rate estimation scheme to drive the transmission rate of the real-time source; the latter, in particular, also attempts to have the adaptive real-time source behave fairly towards concurring TCP connections.

Adaptive algorithms can act upon several aspects of a voice connection, i.e. the coding algorithm, the packet length or the playout delay at the receiver. The first two parameters can be chosen by the transmitter while the third can be adjusted by the receiver. The coding algorithm affects the source transmission rate, as well as other aspects of voice communication, while the packet length trades off the transmission overhead and packetization time (and thus, the delay). The playout delay can be controlled at the

receiver by proper regulation of the playout buffer length. The size of the playout buffer can be fixed, determined at call start-up and then kept constant, or allowed to change in the course of the conversation to adapt to variations of the behavior of the network. The main trade-off is between degradation due to delay versus degradation due to packet losses: on the one hand, longer buffers decrease the number of packets discarded because of late arrival (packets are discarded at the receiver if they are ‘too’ late), at the expense of increased delay; on the other hand, shorter buffers can curb the playout delay, but at the expense of increased packet loss rates.

In this paper, we will focus our attention on providing adaptability to IP telephony applications through variable bit-rate coding algorithms. Application-level congestion avoidance must be accomplished by looking at end-to-end metrics, such as packet loss, latency and jitter.

Packet losses are an indication of a severe network congestion: any action taken following this notification may be belated, thus leading to a prolonged congestion. However, preemptive measures may also be undertaken by acting over an increasing-delay notification. Since the delay experienced by packets traveling from a source to a destination is a function of the number of hops, and of the queue length at intermediate nodes, a sudden increase in the delay observed by the receiver (and relayed to the sender, through periodic receiver’s reports) is often followed by packet loss within the next few round-trip times.

2.1. Monitoring of network congestion

Making the source rate — that is, the rate at which the speech compressor operates — depend on the state of the network requires some way of estimating such a state since the IP service model does not offer congestion notification. The focus is on two main measures: the effective capacity of the link, largely determined by the bottlenecks along the connection, and the detection of temporary congestion. Based on such measures, the rate control algorithm will select bit rates compatible with the estimated capacity and respond to short-term clogging of the link.

In order to implement an effective adaptive algorithm, the estimates of quantities related to the ongoing connection and needed by the adaptive algorithm are to be carried out. Such a procedure involves several aspects:

1. Since delay estimates are essentially based upon a correct alignment of both the source’s and the receiver’s clocks, the skew between these clocks should be estimated.
2. While single delay estimates can be made on the reception of every RTP packet, the loss rate estimates should reflect the ratio between the lost and the expected packets, computed over a given time span.
3. Some quantities require averaging in order to compare the current estimates with the expected behavior; more-

over, the average should be weighted so as to smooth out the fluctuations.

4. The information to be exchanged between the source and the receiver (i.e., who computes the estimates, who averages the quantities, etc.) should be determined.

2.2. The adaptive algorithm

The rate control algorithm described in this paper, starting from delay and loss measurements relayed by RTCP reports, tries to regulate the output rate of several voice sources. It does not rely on explicit congestion indications by either the intermediate nodes or the receivers. Rather, it uses the information that can be carried by cyclic RTCP receiver reports to let the source know the state of the ongoing connection. The control algorithm runs at the source itself. An alternative scheme would be one where the algorithm runs at the receiver, where all the raw estimates are available, not only the down-sampled, smoothed versions of the same. Such a receiver-based scheme, however, would require a signaling mechanism to convey the rate increase/decrease commands to the source.

The adaptive algorithm follows the ‘additive increase/multiplicative decrease paradigm’ that is successfully employed by many congestion control algorithms, such as TCP or ABR. The basic idea is that the source coder should reduce its rate when packet delays have been observed to have increased considerably above a ‘high-mark threshold’. Also, it should drastically decrease the rate to, say, half its value, when severe congestion is detected (i.e. the packets are lost). Besides, it should switch to higher rates if no packets are lost and if the delay decreases below a ‘low-mark threshold’.

Clearly, since we are looking at an IP telephony scenario, both parties can, at different times, be considered the source and the destination. In order to simplify the description, we shall decouple these interactions and only consider a communication between a generic source and a receiver.

The algorithm flow chart is reported in Fig. 1 and its pseudo-code is provided in Fig. 2. It is periodically executed by the source, whenever a receiver RTCP report reaches it. It should be pointed out that, if no RTCP is received by the source for longer than 5 s, the source takes it upon itself to reduce the rate by a step; thus, it conservatively keeps its transmission speed from increasing only because a lack of statistics (i.e. no losses, stable delay) is interpreted as ‘favorable’ network conditions.

Throughout the algorithm description, the following parameters are used (when possible the reference values used in simulation are reported between brackets):

- `curr_time`: current time;
- `dec_int`: minimum time between back-to-back decrements (1 s in the simulations);

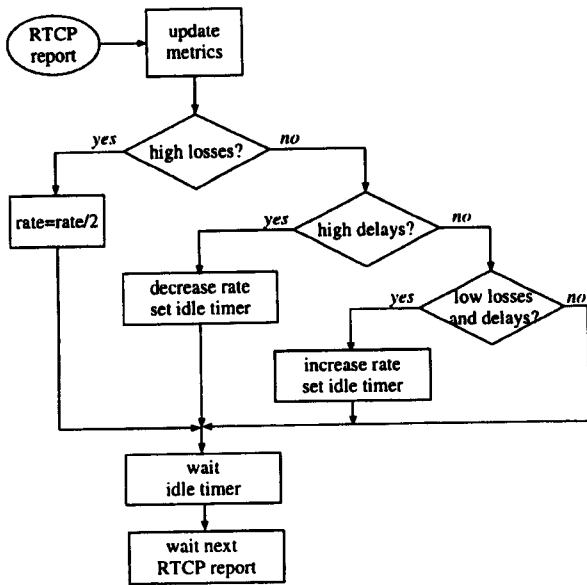


Fig. 1. Flow chart of the adaptive algorithm.

- `inc_int`: minimum time between back-to-back increments (3 s in the simulations);
- `var_time`: time when the last rate decrement/increment occurred;
- `Max_Loss`: loss ratio that triggers the algorithm's rate decrement (3% in the simulations);
- `Min_Loss`: loss ratio that triggers the algorithm's rate increment (7% in the simulations);
- `peer_SLR`: smoothed loss ratio;
- `avg_delay`: average (smoothed) delay;
- `latency_thresh`: minimum relative latency increase that triggers a rate decrement (in the simulations we used `latency_thresh = 0.1`, indicating that the rate should be lowered if the latest delay measure is 10% higher than the mean so far computed);
- `peer_delay`: latest measure of delay.

The possible values for the bit rate vary between 8 and 64 kb/s in steps of 8 kb/s.

The numerical values of the parameters that were used in simulation are often the results of 'sensible' choices, driven by experience and observation, rather than of thorough analysis. Moreover, we have observed that, when parameters change within a reasonable range, there is no dramatic impact on the algorithm behavior (i.e. `dec_int` did not yield significantly different behaviors between 1 and 5 s, although it makes sense to have `dec_int` < `inc_int`, to be consistent with the 'slow increase, fast decrease' paradigm). As is often the case with complex systems, a choice of parameters that optimizes the behavior of the system (i.e. maximizing utilization and/or minimizing losses) is computationally infeasible. We have, however, verified through extensive simulations that the algorithm

```

halve = decrement = increment = FALSE;
/* Compare loss rates */
if (peer_SLR > Max_Loss) then
  if (curr_time > var_time + dec_int) then
    halve = TRUE;
    var_time = curr_time;
  endif
else if (peer_SLR < Min_Loss) then
  if (curr_time > var_time + inc_int) then
    increment = TRUE;
    var_time = curr_time;
  endif
endif
/* Compare delay estimates */
if (peer_delay > avg_delay + avg_delay * latency_thresh) then
  if (curr_time > var_time + dec_int) then
    decrement = TRUE;
    var_time = curr_time;
  endif
else if (peer_delay < avg_delay - avg_delay * latency_thresh) then
  if (curr_time > var_time + inc_int) then
    increment = TRUE;
    var_time = curr_time;
  endif
endif
/* Change rate according to choice (decrement has priority) */
if (halve = TRUE) then
  HalveRate();
else if (decrement = TRUE) then
  DecreaseRate();
else if (increment = TRUE) then
  IncreaseRate();
endif
  
```

Fig. 2. Pseudo-code for the adaptive algorithm.

is stable and behaves in the predicted way, although we cannot guarantee that it yields optimal results. The stability of the algorithm is vouched by its conformance to the 'additive increase, multiplicative decrease' paradigm made popular by the TCP algorithm.

Smoothed quantities are computed through an exponential averaging function: $\bar{x}(n) = \alpha \bar{x}(n-1) + (1-\alpha)x(n)$, where $\bar{x}(n)$ is the averaged quantity and $x(n)$ the n th sample; α is a positive real number smaller than 1 ($\alpha = 0.2$ in the simulations) and determines the speed and accuracy with which the average $\bar{x}(n)$ follows the trend of $x(n)$.

Being a second-order estimate, and thus more delicate to be properly accounted for, the jitter is not used in this algorithm. However, we plan to extend the scope of our adaptive algorithms to take into account the jitter in the future.

3. Perceptual quality of service

Perceptual QoS is a function of all the factors affecting the speech signal from the moment it is produced by the speaker's mouth to the moment it is perceived by the listener's ear. Focusing only on what takes place between the output of the A/D converter at the transmitter and the input of the D/A converter at the receiver several main factors impacting perceptual QoS can be identified.

Firstly, in most cases, the stream of speech samples produced by the A/D converter will be compressed by a *speech coder* [11,12]. Uncompressed telephone-bandwidth speech, indeed, requires bit rates greater than 100 kb/s; 64 kb/s if non-uniform quantization is employed, as in ITU-T standard G.711 (log PCM). Modern speech coders,

instead, can operate at bit rates of 8 kb/s or lower. Such bit-rate savings, however, generally come at the cost of a certain degree of perceptual degradation with respect to the original speech signal. Thus, the first factor affecting the perceptual QoS is the quality generated by the specific speech coder employed in the application; its output quality represents the upper bound to the perceptual QoS achievable by the system, that is, the quality under ideal conditions. Note also that modern speech coders usually operate on segments of speech, called *frames*, and not on individual samples. A delay is, therefore, necessarily introduced between the instant a sample enters the speech coder and the instant when the corresponding synthesized sample exits the decoder. The delay introduced by the speech coder is usually called *algorithmic delay*. As the *overall* delay (algorithmic, transmission, queuing, buffering, decoding, etc.) increases, the perceptual quality decreases, as will be discussed below.

The compressed bit-stream produced by the speech coder is then packetized and sent on the network, where two other important factors come into play: packet losses and potentially unbounded delay.

Packet losses are typical of speech transmission over packet networks, where packets can arrive too late for timely playback, or even not arrive at all. In this case, a whole speech frame (typically 5–25 ms of speech, or more if a packet contains more than one frame) is either unavailable or declared unusable due to bit errors in the perceptually crucial bits of the frame (e.g. the most significant bits of the energy). In both cases, techniques to fill the gap corresponding to the corrupted or missing frame are needed. In the case of high rate waveform coders, such as PCM, sample-based techniques such as repetition, interpolation and extrapolation are used. In more complex coders, instead, the parameters of the previous, correctly received frame are generally used, often by repeating them, with a muting mechanism activated in case of consecutive frame erasures. Note that some speech coders are quite sensitive to frame erasures: under 3% random frame erasures, for instance, the quality of the widely used ITU-T G.729 8-kb/s standard drops by about 0.5 MOS, a very significant drop. For 5% frame erasures losses the quality drops even lower, to less than 3 MOS. In general, the impact of the packet losses varies, depending on the statistics of the frame erasures (random, burst) and the specific speech coder at hand.

As mentioned before, another very significant factor influencing perceptual QoS is delay. Besides the delay introduced by the speech coder at the transmitter, additional delay is introduced at several other points during transmission, some of it unbounded due to the IP best-effort model. The perceptual effects of delay on telephony have been extensively studied in the past. ITU-T recommendation G.114 reports the results of several subjective tests, which led to the conclusion that for most users a delay of less than 150 ms is not objectionable. Delays between 150 and

400 ms, instead, are deemed tolerable, provided that the service provider is aware of the effects that such levels of delay have on quality. Delays above 400 ms are not acceptable for general network planning. Note that, unlike most other speech transmission scenarios, in VoIP delay can vary during the course of the same conversation; the perceptual effects of such variability have not yet been extensively studied.

To guarantee a certain level of perceptual QoS is, therefore, equivalent to guaranteeing that:

1. the chosen speech coder is capable of delivering the desired speech quality;
2. there is enough capacity to transmit reliably the bit rate generated by the speech coder;
3. packet losses are below a certain threshold;
4. the overall mouth-to-ear delay is contained within a given level.

3.1. Perceptual quality and AVoIP

When the compressed bitstream is transmitted over a time-varying channel, estimating and controlling the perceptual quality becomes particularly demanding. In the specific case of VoIP, the IP network can be seen as a lossy channel with time-varying *capacity*, *packet loss rate* and *delay*.

The instantaneous capacity is determined in part by the structural characteristics of the connection, such as its bottlenecks, and in part by the traffic dynamics. In any case, the basic assumption is that transmission rates above the current capacity result in packet losses and/or increased delays, that is, in potentially significant degradation of the perceptual quality.

If a fixed-rate source coder is employed and the capacity drops below the operating bit rate, the perceptual quality will be more or less severely affected with no way of controlling the degradation. In this scenario, countermeasures can only be implemented at the system level, e.g. increasing the capacity of the system and implementing mechanisms that ensure that the capacity will never, or very rarely, drop below a certain safety level.

If, instead, a network-driven variable bit-rate speech coder is used, the time-varying nature of the channel can be, at least to some extent, matched, resulting in greater control of the perceptual QoS and more efficient use of the network resources. More specifically, an adaptive approach makes it possible to choose the optimal bit rate for the current instantaneous channel conditions. Under the assumption that lower bit rates are correlated with lower levels of packet losses and/or delay, there is a threshold below which the decrease in quality erailed by the lower bit rate is more than offset by a lower level of degradation due to losses and delay. In other words, the adaptive approach makes it possible to choose the best trade-off

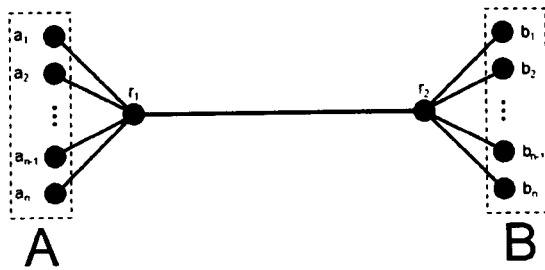


Fig. 3. *Bottle-neck topology.*

between bit-rate, packet losses and delay, and thus to operate, at least in principle, at the instantaneous optimal operating point.

Although most speech coding standards are fixed-rate, several variable bit-rate speech coders are available. A recent example of a variable bit-rate solution is the new GSM adaptive multi-rate (AMR) speech coding standard [13]. In GSM-AMR, one of eight rates ranging from 4.75 to 12.2 kb/s is selected depending on the instantaneous condition of the wireless channel, a scenario with several points of contact with the one considered in this paper. Another example is the new ISO MPEG-4 audio standard which includes a variable bit-rate CELP speech coder operating at bit rates between 3.85 and 12.2 kb/s [14].

As a measure of perceptual quality, the following performance indices are taken into consideration in this paper:

- average and instantaneous bit rate of the voice coder;
- average and instantaneous delay experienced by the packets in traversing the network (it comprises the queuing and transmission delays);
- the distribution, expressed in terms of the probability mass function, of the number of consecutive packets (belonging to the same voice stream) which are successfully received;
- the distribution of the number of consecutive packets which are lost.

The last two measures assess the ease in reconstructing a gap of missing frames in a corrupted voice stream.

4. Simulation setup

All the simulations whose results are presented in this paper were run using a simple bottleneck network topology. More complex topologies, while more realistic, have the drawback of providing results that do not lend themselves to reliable interpretation. Fig. 3 provides a pictorial representation of the topology. The bottleneck link (between routers r_1 and r_2) is supposed to have a limited bandwidth equal to 256 kb/s while the secondary links are assigned a sufficiently high bandwidth so that all traffic traveling from

a source to a destination only experiences congestion on the bottleneck link. Link latencies are 3 ms for the bottleneck link and 1 ms each for the secondary links (resulting in a 10-ms round-trip delay). The buffering capacity of the bottleneck link router is equal to 16 kB and buffer congestion is dealt with using a droptail strategy. Sources are placed at both ends of the bottleneck link, although the results examined by us only refer to the set of sources transmitting in one direction (the presence of reverse-direction sources may, however, affect the reception of RTCP packets carrying receiver reports relating to forward-direction communications). Simulated voice sources generate a continuous traffic flow throughout the simulation.

Our study was targeted at studying three main traffic scenarios:

- *homogeneous traffic*: only VoIP sources are active, thus simulating a segregated network environment where the resources are entirely devoted to voice traffic;
- *interfering FTP traffic*: VoIP sources compete for bandwidth with greedy FTP connections sharing the same network path;
- *interfering Pareto/UDP traffic*: a single VoIP source and interfering traffic modeled as Pareto *On/Off* connections.

The simulation tool is the well-known ns simulator [15], modified to suit our purposes.

5. Simulation results

The performance metrics presented in this section are meant to compare the performance of VoIP communications with and without adaptivity, under the scenarios outlined in Section 4. Therefore, every experiment was replicated several times, using adaptive-rate VoIP sources in one instance and constant-bit-rate (CBR) VoIP sources otherwise. Rate adaptation was achieved by having the sources transmit variable-size packets at a fixed rate (i.e. on rate increases, the source starts sending larger packets without changing the packet inter-transmission time fixed at 125 ms). All CBR sources send 512 bytes-long packets, while the transmission frequency is determined according to the source rate (and is kept constant). In this paper, we shall examine the comparisons between AVoIP and fixed-rate sources transmitting at 8, 16, 24 and 32 kb/s. The respective performances were compared in terms of loss rates and delivery delays; the loss process was further analyzed through the observation of several statistical properties, characterizing the quality of the communication perceived by end users, i.e.:

- *back-to-back lost packets*: average variance and pdf of the number of packets that were consecutively dropped by router — the bottleneck link router — because of the onset of congestion;

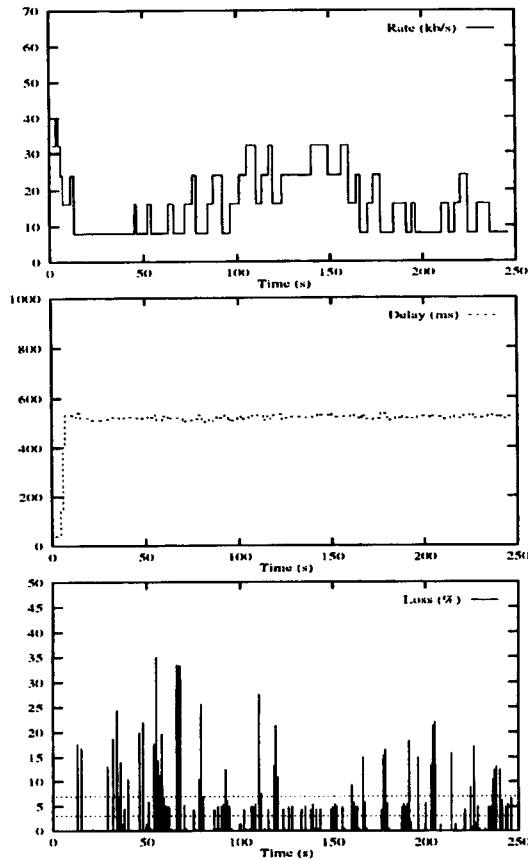


Fig. 4. Sample AVoIP source behavior under homogeneous traffic (20 sources).

- *back-to-back received packets*: average, variance and pdf of the number of packets that were delivered and played back on time, with no packet drops in between.

5.1. Homogeneous traffic

The first set of experiments focuses on several voice sources competing against each other on a bottleneck link, without interfering traffic. This situation might represent a virtual private network of a small VoIP operator, carrying all IP phone calls on the same network infrastructure. Also, it might depict the aggregate behavior of a subset of traffic in a DiffServ environment: following the guidelines set

Table 1
AVoIP and CBR comparison (20 sources)

Source type	Loss (%)	Delay (s)
AVoIP (12.44 kb/s)	20.4	0.52
CBR 32 kb/s	65.0	0.53
CBR 24 kb/s	53.6	0.53
CBR 16 kb/s	31.7	0.51
CBR 8 kb/s	0	0.06

Table 2
AVoIP and CBR comparison (15 sources)

Source type	Loss (%)	Delay (s)
AVoIP (12.59 kb/s)	4.13	0.30
CBR 32 kb/s	53.25	0.53
CBR 24 kb/s	38.11	0.52
CBR 16 kb/s	9.01	0.50
CBR 8 kb/s	0	0.05

forward in Ref. [16], all voice traffic could be marked so as to receive an adequate service level through per-hop-behaviors (PHB) negotiated between the customer and the provider.

Results were obtained for 10, 15 and 20 active sources in each direction.

Fig. 4 provides a first glimpse as to the behavior of the algorithm for a sample AVoIP source over a 250-s time span, when 20 AVoIP sources share the bottleneck link: from top to bottom, the source rate, the end-to-end delay and the loss percentage are shown as a function of time. A straightforward observation is that, aside from the very beginning, the algorithm is mostly operated through loss indications resulting in rate halving while single-step rate reductions due to delay indications rarely take place (an example can be seen approximately 150 s into the experiment). This behavior appears to be peculiar to the considered scenario since single-step reductions are more frequent when non-adaptive interfering traffic is present, as will be shown below.

Further indications can be drawn by comparing the loss and delay metrics when adaptive only and CBR only traffic is simulated: Tables 1, 2 and 3 detail these comparisons for 20, 15 and 10 sources, respectively (the rate between brackets for AVoIP sources refers to the average rate at which the sources have been transmitting during the simulation). Several observations are in order:

- AVoIP sources, while not necessarily providing an optimal performance (i.e. maximal network utilization and no losses) in all the three cases, succeed in finding a suitable ‘operating point’ without any prior knowledge of the available bandwidth (as is often the case); the high loss percentage and high delays experienced when 20 sources share the link are determined by the optimal operating point being placed between 8 and 16 kb/s,

Table 3
AVoIP and CBR comparison (10 sources)

Source type	Loss (%)	Delay (s)
AVoIP (18.81 kb/s)	0.02	0.05
CBR 32 kb/s	29.77	0.52
CBR 24 kb/s	7.18	0.51
CBR 16 kb/s	0	0.05
CBR 8 kb/s	0	0.04

Table 4
Average and variance of the length of packet loss bursts for AVoIP and CBR sources

Source type	Average	Variance
AVoIP	1.70	2.54
CBR 32 kb/s	3.83	22.69
CBR 24 kb/s	3.08	17.94
CBR 16 kb/s	2.14	8.53

Table 5
Average and variance of the length of packet reception bursts for AVoIP and CBR sources

Source type	Average	Variance
AVoIP	6.15	57.68
CBR 32 kb/s	2.01	6.24
CBR 24 kb/s	2.46	16.93
CBR 16 kb/s	4.00	39.00

i.e. between two adjacent ‘states’ that either yield no loss or cause more than 30% of lost traffic; the algorithm thus suffers from the lack of bit-rates between these two values;

- the loss rates of AVoIP sources roughly correspond to the loss rates of CBR sources operating at a rate equal to the average AVoIP rate (this assumption is based on a linear interpolation of CBR sources behavior as summarized by Tables 1–3); again, this result is peculiar to the scenario we are examining, as will be discussed below;
- when large numbers of sources are sharing the link, the delivery delays for AVoIP sources are quite high; beside the lack of dynamic range discussed above, this is a consequence of the algorithm not acting upon *high* delays but, rather, upon *highly varying* delays (Fig. 4 highlights that the delay variation is not high).

Although the speech quality perceived by end users cannot be ascertained in a simulation experiment like the ones we conducted, nonetheless, we tried to overcome this

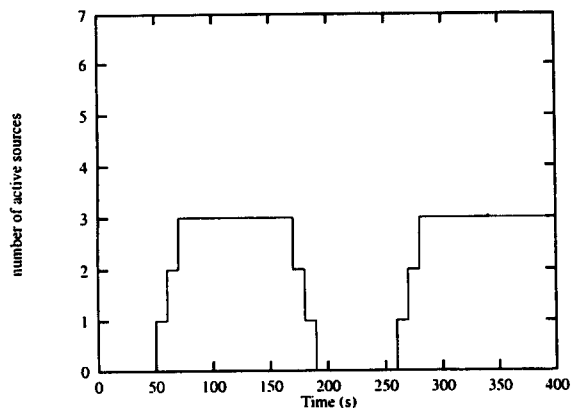


Fig. 5. Number of concurrent FTP connections as a function of time.

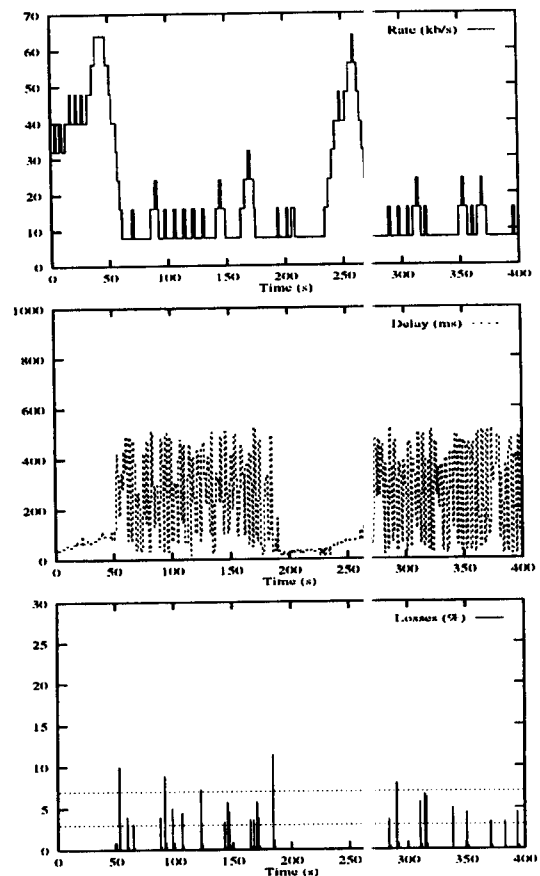


Fig. 6. Sample AVoIP source behavior with interfering FTP sources (10 AVoIP sources + 3 FTP sources).

limitation through a simple statistical analysis of the flow of received packets. First of all, we compared the average and the variance of the length of *loss bursts*, i.e. back-to-back packets that were not received and that, conceivably, led to ‘playback holes’ hard to recover even with the techniques described in Section 3. Similarly, we kept trace of the number of consecutive packets that were correctly received. These quantities are reported in Table 4 and 5 for the case of 20 sources (it should be noted that CBR sources at 8 kb/s

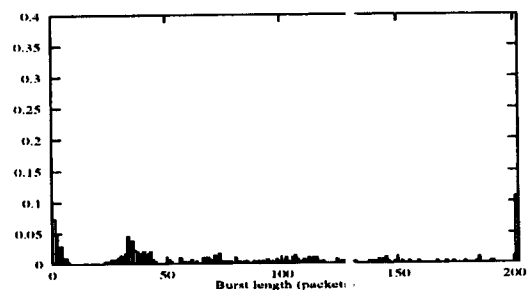


Fig. 7. Distribution of the number of back-to-back received packets by the AVoIP source.

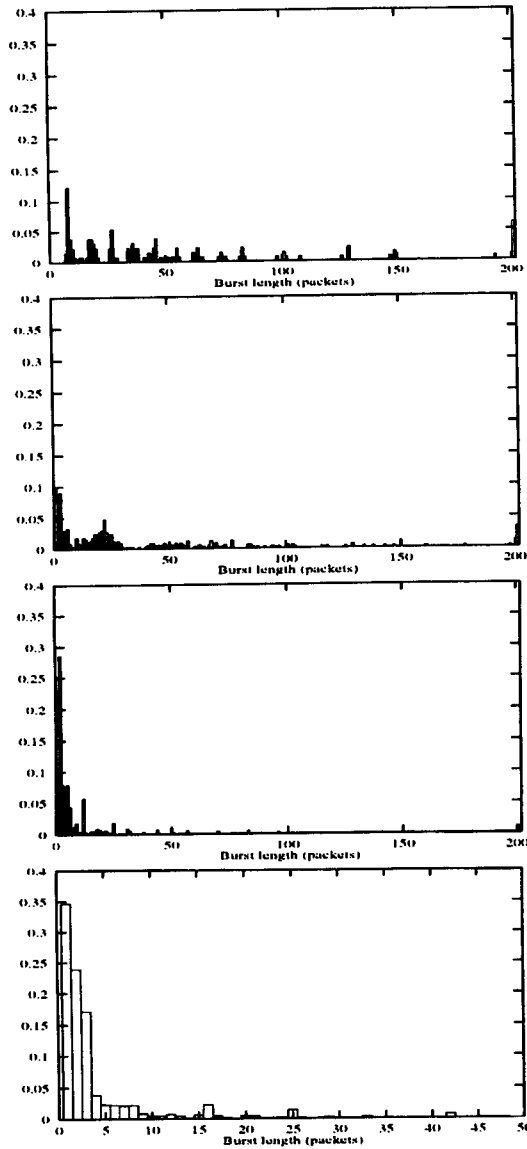


Fig. 8. Distribution of the number of back-to-back received packets by 8, 16, 24 and 32 kb/s CBR sources (from top to bottom).

are missing: since they do not register any losses, their statistics would have been meaningless). These results point out that, on average, AVoIP sources lose one to two packets out of every six to seven received packets. Compared with 16 kb/s CBR sources, which lose, on average, two packets and more out of every four received packets, the perceived speech quality will be superior for AVoIP sources.

5.2. Interfering FTP traffic

The second scenario we have considered features the staggered activation of the three greedy FTP file transfers (they are turned *On* at 10-s intervals starting at times 50 and

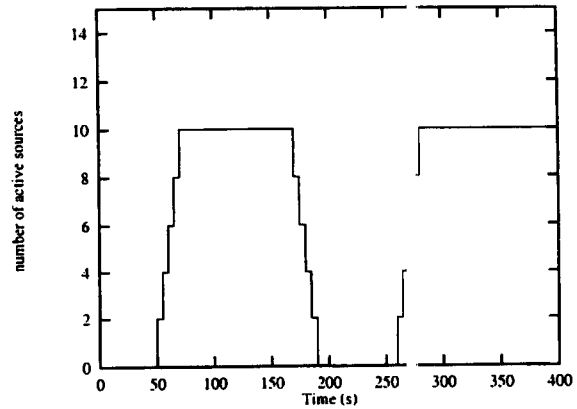


Fig. 9. Number of concurrent Pareto *On/Off* connections as a function of time.

260 s, and are turned *Off* at 10-s intervals starting at time 170 s, as shown in Fig. 5). FTP sources send 1000-bytes-long packets and the TCP connections they are supposed to establish have a maximum window of 20 packets. Ten voice sources are active throughout the simulation.

The behavior of a sample AVoIP source can be observed

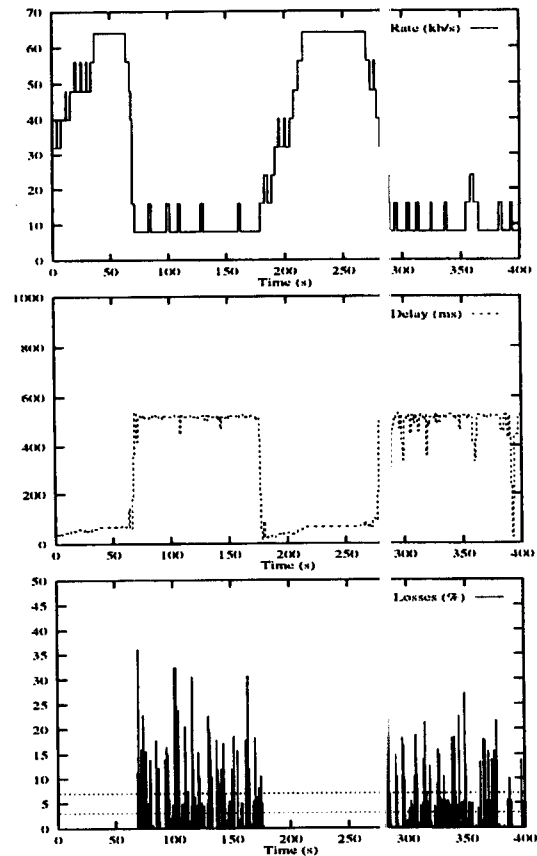


Fig. 10. AVoIP source behavior with interfering UDP sources.

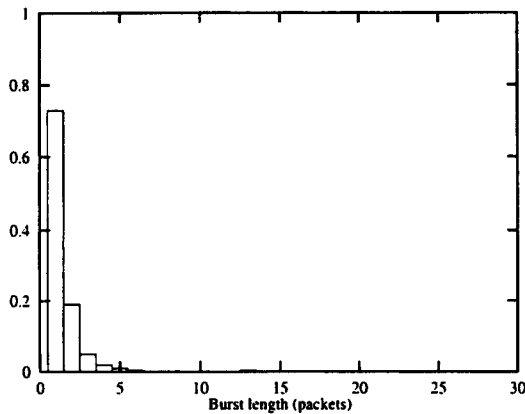


Fig. 11. Distribution of the number of back-to-back lost packets by the AVoIP source.

in Fig. 6 over a 400-s simulation run. By comparing the rate curve with the one shown in Fig. 4, we have an exemplification of how the delivery delay can drive the control algorithm: it provides a smooth transition to lower bit rates (as can be seen in several instances during the simulation) by single-step decreases, as opposed to the steeper transitions observed when only AVoIP sources were present, in a 'stationary' scenario. We can, thus, conclude that our algorithm is all the more effective when the network conditions change (i.e. when the sources switch on and off).

The comparison with the behavior of a CBR source in this scenario allows us to investigate some statistical properties of the packet reception pattern, an observation that can be linked to users' satisfaction with the quality of the received speech. Fig. 7 plots the probability distribution of the number of back-to-back received packets for the AVoIP source and it should be compared with similar plots in Fig. 8, referring to CBR sources at 8, 16, 24 and 32 kb/s (from top to bottom). Note that the bursts longer than 200 packets are cumulatively accounted for at the x -axis value of 200. Of course, longer, uninterrupted streams of received packets can be translated into a better perceptual quality and AVoIP sources (whose measured average rate was 11.6 kb/s) appear to have the edge when compared to CBR sources.

5.3. Interfering Pareto/UDP traffic

The last scenario we have examined features a single voice source and interfering traffic modeled as Pareto *On/Off* sources over UDP connections. As many as 10 such sources are active at the same time (the activation pattern is similar to the one described in the previous section and is shown in Fig. 9). It should be pointed out that the deactivation of a source is potentially disruptive of the LRD effect

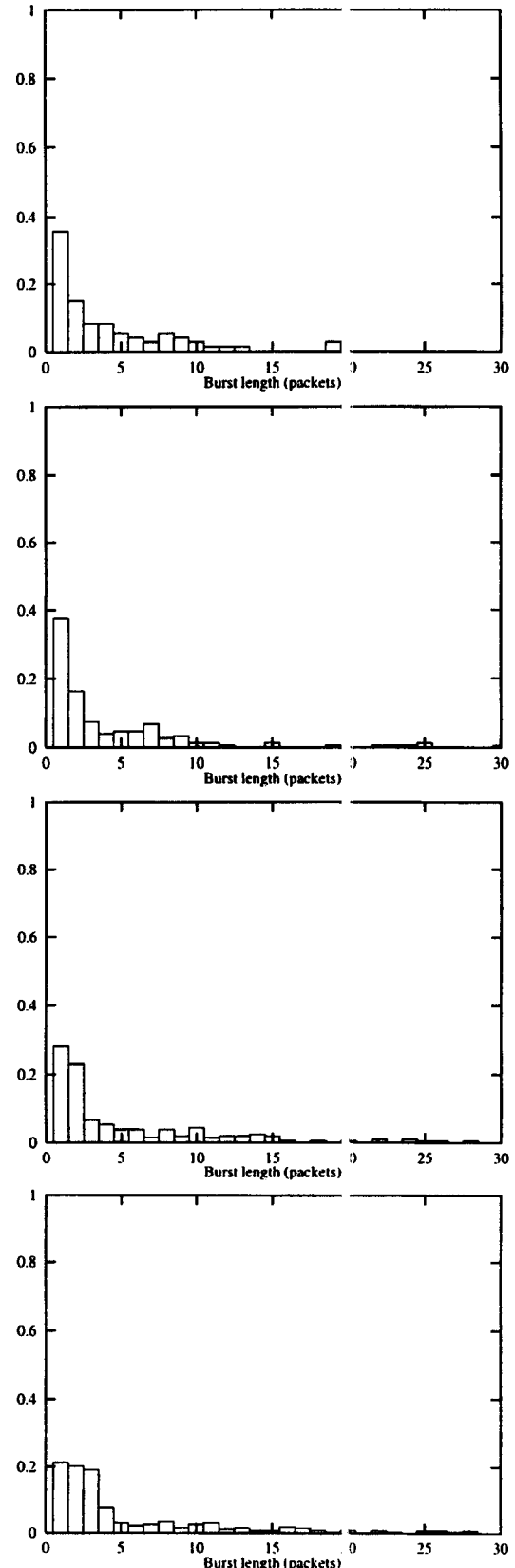


Fig. 12. Distribution of the number of back-to-back lost packets by 8, 16, 24 and 32 kb/s CBR sources.

introduced by the Pareto distribution since the source is switched off even in the midst of an *On* period.

Since UDP traffic is not adaptive (while FTP traffic employs TCP congestion control), we expect this scenario to be more testing than the previous one.

The Pareto sources parameters are:

- *On time*: 2 s;
- *Off time*: 2 s;
- *average nominal rate*: 128 kb/s;
- *packet size*: 1 kB;
- *Pareto α shape parameter*: 0.5.

Interesting observations can be drawn from Fig. 10, where rate, delay and loss dynamics for the voice source are plotted. The intervention of the varying-delay indications, resulting in single-step decrements, are visible not only when the UDP sources are activated, but also when they are deactivated (thin rate peaks around 200 s): the algorithm curbs the aggressiveness of the AVoIP source as the UDP traffic is slowly leaving the buffer at the router. No losses are detected after the first UDP sources are deactivated.

Fig. 11 plots the probability distribution of the length of loss bursts of the AVoIP source and it should be compared with similar plots in Fig. 12, referring to CBR sources at 8, 16, 24 and 32 kb/s (from top to bottom). To put things in perspective, the AVoIP source reached an average of 27.88 kb/s during the simulation. By comparing the distribution of losses, we can conclude that long, disruptive playback gaps (five or more lost packets) were rarely observed, while this event was more frequent for CBR sources, even at rates much lower than the average rate achieved by the AVoIP source.

6. Conclusions and future work

We have presented a study on the effectiveness of using network-driven adaptive voice sources for interactive voice communications over IP networks. In the proposed architectural solution, variable bit-rate voice coders adapt their rate to the time-varying network conditions by means of a control algorithm whose aim is maximizing the utilization of the available bandwidth while reducing and preventing the occurrence of packet losses. Delay and packet loss estimates are employed to drive the algorithm's response to building congestion. Delay jitter does not play a part in the rate adaptation algorithm, but could be used to devise an algorithm that controls the size of the playout buffer, which is usually employed to smooth out delay variations. Such a feature is left for future work.

The performance of the proposed approach is evaluated in various scenarios which comprise a network dedicated to the exclusive use of adaptive voice sources as well as heterogeneous uncontrolled network environments.

The results are also compared against the performance of non-adaptive sources; it is shown that the adaptive approach is more effective, being able to carry more voice communications while maintaining an acceptable QoS, even on non-segregated networks.

Acknowledgements

This work was supported in part by a research contract between CSELT and Politecnico di Torino and in part by the Italian Government through the Ministry for University and Research (MURST).

References

- [1] J. Wroclawski, Specification of the controlled-load network element service, RFC 2211, September 1997.
- [2] S. Shenker, Fundamental design issues for the future Internet, *IEEE Journal on Selected Areas in Communications*, September (1995) 1176–1188.
- [3] L. Breslau, S. Shenker, Best-effort versus reservations: a simple comparative analysis, *ACM Computer Communication Review* 28 (4) (1998) 3–16.
- [4] R.V. Cox, R.E. Crochiere, Multiple user variable rate coding for tasi and packet transmission systems, *IEEE Transactions on Communications* 28 (COM-3) (March 1980) 334–344.
- [5] T. Bially, B. Gold, S. Seneff, A technique for adaptive voice flow control in integrated packet networks, *IEEE Transactions on Communications* COM-28 (1980) 325–333.
- [6] N. Yin, M.G. Hluchyj, A dynamic rate control mechanism for source coded traffic in a fast packet network, *IEEE Journal on Selected Areas in Communications* 9 (7) (September 1991) 1003–1012.
- [7] J.C. Bolot, T. Turletti, A rate control mechanism for packet video in the Internet, *Proceedings of IEEE INFOCOM '94*, Toronto, Canada, June 1994.
- [8] J.C. Bolot, T. Turletti, I. Wakeman, Scalable feedback control for multicast video distribution in the Internet, *Proceedings of SIGCOMM '94*, London, UK, August 1994.
- [9] I. Busse, B. Deffner, H. Schulzrinne, Dynamic QoS control of multimedia applications based on RTP, *Proceedings of the 1st International Workshop on High Speed Networks and Open Distributed Platforms*, St. Petersburg, Russia, June 1995.
- [10] D. Sisalem, H. Schulzrinne, The loss-delay adjustment algorithm: a TCP-friendly adaptation scheme, *Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video (NOSS-DAV)*, Cambridge, England, July 1998.
- [11] A. Gersho, Advances in speech and audio compression, *Proceedings of the IEEE* 82 (6) 900–918 (June 1994).
- [12] W.B. Kleijn, K.K. Paliwal (Eds.), *Speech Coding and Synthesis*, Elsevier, Amsterdam, 1995.
- [13] E. Ekudden, R. Hagen, I. Johansson, J. Svedberg, The adaptive multi-rate speech coder, *Proceedings IEEE Workshop on Speech Coding Proceedings*, Haikko Manor, Porvoo, Finland, June 1999.
- [14] ISO/IEC, Information technology-coding of audiovisual objects, part 3: audio, subpart 3: Celp. 14496-3 FDIS, December 1998.
- [15] ns-2, network simulator (ver. 2), LBL. URL: <http://www-mash.cs.berkeley.edu/ns>.
- [16] S. Blake, D. Black, M. Carlson, E. Davies, J. Wang, W. Weiss, An architecture for differentiated services, RFC 2475, December 1998.