A Real-Time Protocol for the Internet Based on the Least Mean Square Algorithm

Nestor Becerra Yoma, *Member, IEEE*, Juan Hood, and Carlos Busso

***Abstract*—Generally, real-time applications based on User Datagram Protocol (UDP) protocol generate large volumes of data and are not sensitive to network congestion. In contrast, Transmission Control Protocol (TCP) traffic is considered "well-behaved" because it prevents the network from congestion by means of closed-loop control of packet-loss and round-trip-time. The integration of both sorts of traffic is a complex problem, and depends on solutions such as admission control that have not been deployed in the Internet yet. Moreover, the problem of quality-of-service (QoS) and resource allocation is extremely relevant from the point of view of convergence of streaming media and data transmission on the Internet. In this paper an adaptive real-time protocol based on the Least Mean Square (LMS) algorithm is proposed to estimate the application UDP bandwidth in order to reduce the quadratic error between the packet loss and a target. Moreover, the LMS algorithm is also applied to make sure that the reduction in the average bandwidth allocated to each TCP process will not be higher than a given percentage of the average bandwidth allocated before the beginning of the UDP application.**

***Index Terms*—Internet, real-time applications, QoS, TCP, UDP.**

I. Introduction

REAL-TIME applications usually require constant bit rates, low delay and jitter, and are implemented with the User Datagram Protocol (UDP) protocol, which in turn provides open-loop congestion control to adapt the transmission rate. As a consequence, these applications can be very aggressive in terms of the use of network resources in opposition to Transmission Control Protocol (TCP) traffic, which is considered "well-behaved" because it prevents the network from overload by means of closed-loop control of packet-loss and round-trip-time. In addition, although the IPv4 suite of protocols do not provide quality-of-service (QoS), priority to UDP packets could cause even higher degradation to TCP traffic [10]. To address this problem, several real-time protocols have been proposed to make UDP applications behave like TCP traffic to keep the network stable. These TCP-friendly protocols can be classified in window-based or rate-based schemes [20]. The protocols implemented with the window-based algorithms [2]–[4], [8], [9], [18], [19], [22] use a congestion window at the sender or at the receiver, as in TCP, whose size is increased in the absence of congestion indications and decreased when congestion occurs. On the other hand, algorithms that employ the rate-based approach adapt their transmission rate considering some network feedback mechanisms that detect the congestion level. The rate-based protocols can be subdivided into simple additive increase, multiplicative decrease (AIMD) mechanisms and model-based congestion control. The AIMD schemes [7], [11]–[13] mimic the TCP congestion control, which in turn results in a transmission rate that shows the typical short term sawtooth behavior that is extremely discontinuous. As a consequence, the AIMD mechanism is not very suitable for continuous media streams. Model-based congestion control [6], [15], [17], [23]–[25] uses a model for the TCP transmission rate and adapt the sending rate to the average long-term throughput of TCP. This produces much smoother rate changes that are better suited for the kind of traffic mentioned above. Model-based congestion control schemes generally compute the transmission rate with the estimation of packet-loss (*PL*) and round-trip-time (*RTT*) using the same expressions derived for the TCP protocols. For instance, the TCP throughput bandwidth, which corresponds to the packet-rate offered by a TCP application, can be approximated by [5]

$$B_{TCP} = \frac{1}{RTT} \cdot \sqrt{\frac{3}{4 \cdot PL}}. \tag{1}$$

However, the TCP estimation of the throughput bandwidth is derived assuming specific congestion control schemes to reliably transmit packets between hosts. As a consequence, this transmission rate could be considered too conservative due to the fact that real-time applications attempt to sustain a constant bit rate with low delay and jitter, although tolerate *PL*.

As mentioned above, integrating UDP and TCP traffic presents several problems because the former generally demands high and constant bandwidth, while the latter adapts the transmission rate according to the network conditions. One possibility that has been investigated is to use admission control for both sort of traffic. A large number of admission control schemes have been proposed in the literature [10] but none of these is currently being employed on the Internet.

This paper proposes a real-time protocol that uses more bandwidth than the TCP-friendly protocols mentioned above, but does not require any network admission control mechanism to protect the TCP traffic from unacceptable degradation due to the increase of the bandwidth required by UDP applications. The protocol addressed here adapts the bandwidth allocated to a real-time application with the Least Mean Square (LMS)

Manuscript received June 27, 2002; revised November 21, 2002. This work was supported by Conicyt-Chile. The associate editor coordinating the review of this paper and approving it for publication was Dr. Anna Hac.

N. Becerra Yoma is with the Electrical Engineering Department, University of Chile, Santiago, Chile (e-mail: nbecerra@ing.uchile.cl).

J. Hood is with Codelco, Rancagua, Chile (e-mail: jhood@codelco.cl).

C. Busso is with the University of Southern California, Los Angeles, CA 90007 USA (e-mail: busso@usc.edu).

Digital Object Identifier 10.1109/TMM.2003.819582

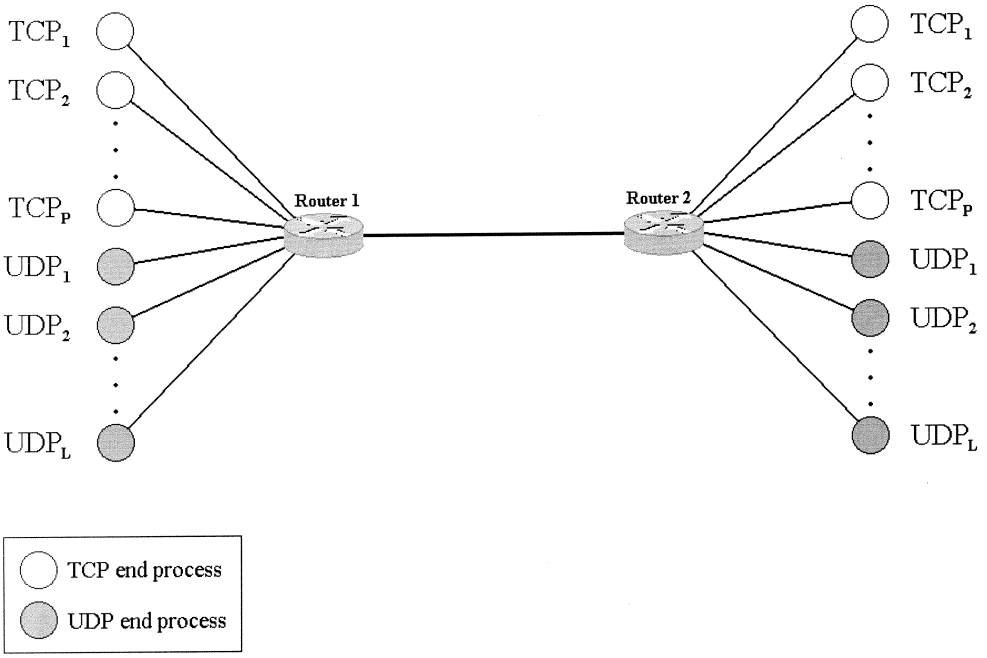

Fig. 1. Two-router problem.

algorithm that aims at reducing the error between *PL* and a desired response. This strategy also aims at providing an estimation of the perceptive quality of the signal at the receiver, because this quality is strongly related to *PL* and the transmission rate. Moreover, the LMS technique is also applied to prevent the UDP application from unacceptably degrading the TCP processes by making sure that the average bandwidth allocated to each TCP process will not be higher than a given percentage of the average bandwidth allocated before the start of the real-time process. Finally, instead of exponentially or linearly modifying the packet-rate according to *PL* as done by TCP protocols [14], [16], the mechanism presented in this paper uses a form of the gradient algorithm to estimate the transmission rate. The scheme proposed here has not been found in the literature and offers an interesting strategy to overcome the limitation of the Internet to allocate resources in both TCP and UDP traffics and to optimize the QoS of real-time applications.

II. Real-Time Protocol

The protocol was designed to address the problem of UDP and TCP applications between two routers (Fig. 1). The two-router topology, which has been used by several authors [6], [11]–[13], is initially employed here just as a model to illustrate the proposed algorithm. It is worth mentioning that all the experiments presented in this paper correspond to a real Internet connection that includes several routers and sources of concurrent traffic. In Fig. 1, there are a finite number of TCP sources of traffic when one or more UDP applications are introduced. The problem is how to estimate the packet-rate of the UDP traffic with the following constraints.

1) The protocol should provide a closed-loop mechanism to protect TCP traffic from unacceptable degradation.
2) The UDP packet-rate should be higher than the one provided by TCP-friendly protocols, if allowed by the network conditions.
3) Due to the fact that a real-time application attempts to sustain a constant transmission speed, the adaptation of the UDP packet-rate according to network conditions should be smoother than the one provided by TCP-friendly protocols.
4) No acknowledgment should be used because this mechanism contributes to overload the network. Instead of that, control packets will be employed because represent a significant improvement on this issue. Each control packet can carry statistics about several (e.g., 100) transmitted data packets.

The proposed transmission protocol is shown in Fig. 2. The algorithm attempts to estimate the application UDP bandwidth in order to reduce the quadratic error between the packet loss and the desired response that would be imposed by the coding scheme. Moreover, the protocol discussed here keeps the application from unacceptably degrading the TCP processes by making sure that the average bandwidth allocated to each TCP process will be higher than a given percentage of the average bandwidth allocated before the beginning of the UDP application.

A. LMS Algorithm and the Error on Packet-Loss

The main idea of this adaptation is to estimate B_{UDP}, the bandwidth in packets/second of the UDP real-time process, in order to achieve a packet-loss target. The LMS algorithm [21] uses the quadratic error at instant t, $e(t)$, as an estimation of the mean-square error (MSE):

$$MSE(t) \approx e^2(t) = [Target\_PL - PL(t)]^2 \qquad (2)$$

where *Target_PL* is the desired response of packet-loss. The partial derivative of $MSE(t)$ with respect to $B_{UDP}(t)$ is given by

$$\frac{\partial MSE}{\partial B_{UDP}} \approx 2 \cdot e(t) \cdot \frac{\partial e(t)}{\partial B_{UDP}} = -2 \cdot [Target\_PL - PL(t)] \cdot \left. \frac{\partial PL}{\partial B_{UDP}} \right|_t . \quad (3)$$

With this estimate of $\partial MSE/\partial B_{UDP}$, the LMS steepest-descent adaptive algorithm [21] is written as

$$B_{UDP}^{PL}(n+1) = B_{UDP}(n) - A_{PL} \cdot \frac{\partial MSE}{\partial B_{UDP}} = B_{UDP}(n) + 2 \cdot A_{PL} \cdot [Target\_PL - PL(n)] \cdot \left. \frac{\partial PL}{\partial B_{UDP}} \right|_n \quad (4)$$

where $n = 0, 1, 2, \ldots$ is a nonnegative integer that denotes discrete sequences with $t = n \cdot \Delta T$; ΔT is the time interval between two consecutive estimations of B_{UDP}; A_{PL} is the adaptation rate, a constant that regulates the step size of a steepest-descent algorithm as the LMS employed here; and, $\partial PL/\partial B_{UDP}$ is estimated in the discrete domain with

$$\left. \frac{\partial PL}{\partial B_{UDP}} \right|_n \approx \frac{PL(n) - PL(n-1)}{B_{UDP}(n) - B_{UDP}(n-1)} = \left. \frac{\Delta PL}{\Delta B_{UDP}} \right|_n . \quad (5)$$

However, $\partial PL/\partial B_{UDP}$ denotes the variation of *PL* with respect to B_{UDP} when all the other parameters (network resources and concurrent traffic) that affect *PL* are kept constant. This is certainly not possible in a real application and $\Delta PL/\Delta B_{UDP}$ is in fact just an approximation of $\partial PL/\partial B_{UDP}$. In order to reduce the effect of the concurrent traffic in the convergence of the algorithm, a lower and upper threshold are imposed to $\Delta PL/\Delta B_{UDP}$. Actually, $\partial PL/\partial B_{UDP}$ should be greater or equal than 0 but, due to the random behavior of the concurrent traffic, the observed $\Delta PL/\Delta B_{UDP}$ might be negative. Considering that $B_{UDP}(n)$ is expressed in terms of packets/second and is an integer, $|B_{UDP}(n+1) - B_{UDP}(n)|$ needs to be higher than 0.5 to observe an adaptation on B_{UDP}. If the quadratic error on *PL* is low, $|B_{UDP}(n+1) - B_{UDP}(n)|$ might be lower than 0.5. In this sense, the lower threshold $LowerTh(\Delta PL/\Delta B_{UDP})$ could be estimated so the adaptation on B_{UDP} is guaranteed when the error on *PL* is higher or equal than a percentage k (e.g., 10%) of *Target_PL*:

$$LowerTh\left(\frac{\Delta PL}{\Delta B_{UDP}}\right) = \left| \frac{1}{4 \cdot A_{PL} \cdot k \cdot Target\_PL} \right| . \quad (6)$$

Also due to the random behavior of the concurrent traffic, the observed $\Delta PL/\Delta B_{UDP}$ might be too high and the estimation of $B_{UDP}(n+1)$ according to (4) might present unbearable fluctuations. This can be avoided by setting a higher bound to $\Delta PL/\Delta B_{UDP}$ so $|B_{UDP}(n+1) - B_{UDP}(n)| \leq \alpha \cdot B_{UDP}(n)$, which in turn leads to

$$HigherTh\left(\frac{\Delta PL}{\Delta B_{UDP}}\right) = \left| \frac{\alpha \cdot B_{UDP}(n)}{2 \cdot A_{PL} \cdot [Target\_PL - PL(n)]} \right| . \quad (7)$$

Equation (4) estimates the UDP bandwidth in order to satisfy the condition $PL(n) = Target\_PL$. Nevertheless, the application packet rate will also depend on the restriction discussed in the following section.

B. Controlling the Degradation of TCP Traffic

According to Fig. 2, at the beginning packets are sent at a low rate in order to evaluate *PL*(0) and *RTT*(0), which correspond to an approximated estimation of packet-loss and round-trip-time, respectively, at time $n = 0$, when the UDP application asks to start the transmission. Due to the fact that the packet-rate was low, *PL*(0) and *RTT*(0) corresponds approximately to the estimation of packet-loss and round-trip-time, respectively, without the real-time process. Then $B_{TCP}(0)$, the average packet-rate of TCP applications at time $n = 0$ (see Fig. 2), is computed with (1), which corresponds to the TCP flow throughput in packets/second. Considering that the routers in Fig. 1 do not distinguish one application from another, and so *PL*(0) and *RTT*(0) are approximately the same for all the UDP and TCP processes, $B_{TCP}(0)$ is approximately the same in all the TCP applications.

The protocol discussed here also keeps the application from unacceptably degrading the TCP processes. It is worth mentioning that any coding scheme imposes a higher bound for *PL*, *MaxPL*, beyond of what the perceptive quality of the video, audio or speech signal is too low. When $PL(0) > MaxPL$ the application stops the connection. If $PL(0) \leq MaxPL$, the protocol initially transmits at the application packet-rate target, B_{Target}, and then adapts $B_{UDP}(n)$ to minimize the quadratic error between $PL(n)$ and *Target_PL*. However, $B_{UDP}(n)$ will also be adapted if

$$B_{TCP}(n) < (1 - \beta) \cdot B_{TCP}(0) \quad (8)$$

where β is the percentage of bandwidth taken from TCP processes at $n = 0$, $B_{TCP}(n)$ is computed with (1), and $PL(n)$ and $RTT(n)$ are computed every ΔT seconds by means of control packets sent by the receiver. This bandwidth is not necessarily allocated only to the UDP application and may also correspond to other TCP processes started at $n > 0$. Consequently, if condition (8) is satisfied, $B_{UDP}(n)$ is also adapted employing the LMS algorithm as in Section II-A:

$$B_{UDP}^{B_{TCP}}(n+1) = B_{UDP}(n) + 2 \cdot A_{B_{TCP}} \cdot [(1 - \beta) \cdot B_{TCP}(0) - B_{TCP}(n)] \cdot \left. \frac{\partial B_{TCP}}{\partial B_{UDP}} \right|_n \quad (9)$$

where $A_{B_{TCP}}$ is the adaptation rate, and $\partial B_{TCP}/\partial B_{UDP}$ is estimated with $\Delta B_{TCP}/\Delta B_{UDP}$ as in (5). Notice that condition (8) is valid when the real-time application starts to transmit. In order to update $B_{TCP}(0)$ in (8), the protocol can be reinitialized every 10 or 20 min to follow the dynamics of the concurrent traffic.

The discussion in Section II-A about the concurrent traffic and the estimation of the partial derivative is also applicable to

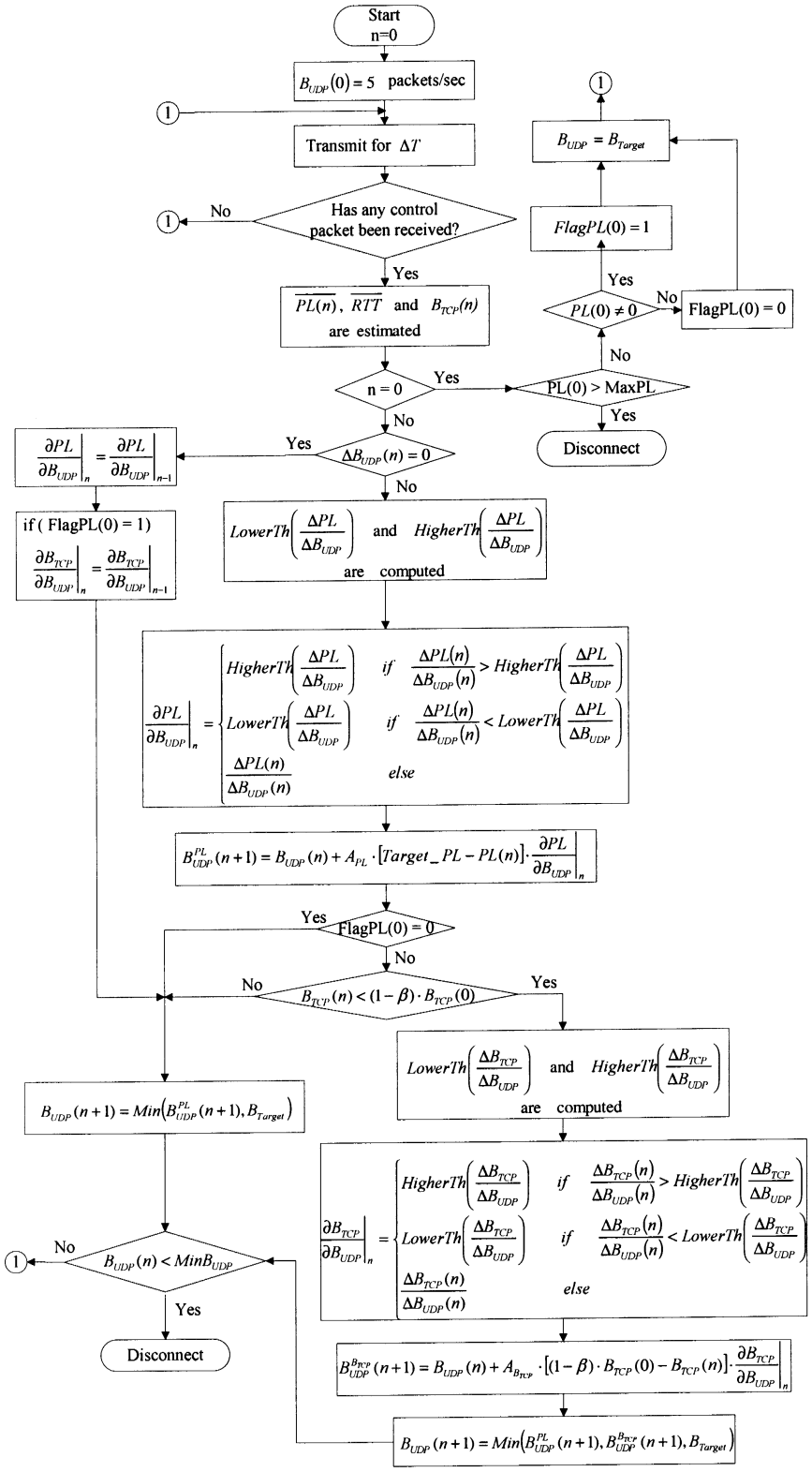

Fig. 2. Real-time protocol.

the problem of adapting B_{UDP} to satisfy (8). As a consequence, as in (6) and the following thresholds are defined:

$$HigherTh\left(\frac{\Delta B_{TCP}}{\Delta B_{UDP}}\right) = \frac{-1}{4 \cdot A_{B_{TCP}} \cdot k \cdot (1-\beta) \cdot B_{TCP}(0)} \quad (10)$$

$$LowerTh\left(\frac{\Delta B_{TCP}}{\Delta B_{UDP}}\right) = \frac{-\alpha \cdot B_{UDP}(n)}{2 \cdot A_{B_{TCP}} \cdot [(1-\beta) \cdot B_{TCP}(0) - B_{TCP}(n)]} \quad (11)$$

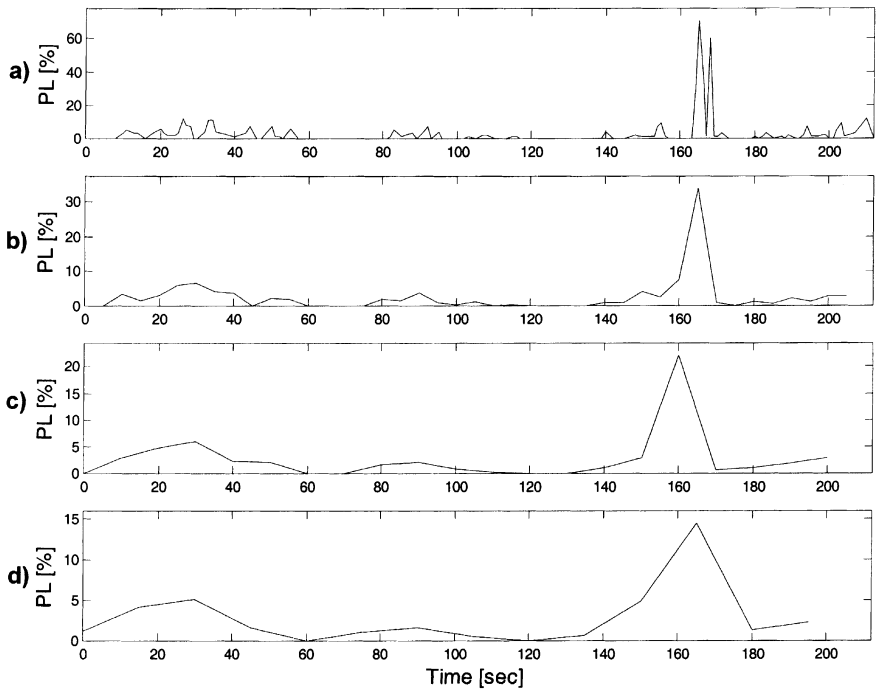

Fig. 3. Packet-loss measured by transmitting UDP packets at constant rate between the hosts at UCh and UNM. The *PL* is averaged within windows with lengths equal to (a) 1 s; (b) 5 s; (c) 10 s; and (d) 15 s.

as in (7), if $(1-\beta)\cdot B_{TCP}(0) - B_{TCP}(n) > 0$. Notice that $(\partial B_{TCP}/\partial B_{UDP}) \leq 0$ by definition.

According to (8) the minimum bandwidth allocated to each TCP process would decrease geometrically with the number of UDP applications in average. If r real-time applications sequentially starts to transmit, condition (8) applied to the UDP application r is given by

$$B_{TCP_r}(n) < (1-\beta)^r \cdot B_{TCP_1}(0) \tag{12}$$

where $B_{TCP_r}(n)$ is the observed average TCP transmission rate estimated at the application r with (1), and $B_{TCP_1}(0)$ is the average packet-rate of TCP applications when the first UDP process asks to start the transmission. This behavior is between a conservative TCP-friendly scheme and an open-loop UDP application that uses as much bandwidth as necessary regardless of the other processes competing for the same network resources.

A new TCP application, when one or more real-time processes based on the algorithm proposed here are already transmitting, will increase the network load and all the TCP applications will reduce their average transmission rate. Similarly, the existing UDP will reduce their transmission rate because the increase in the network load will result in the increase of *PL*.

With the estimations according to (4) and (9), $B_{UDP}(n+1)$ is computed as follows. If condition (8) is not satisfied

$$B_{UDP}(n+1) = \min\left[B_{UDP}^{PL}(n+1), B_{Target}\right]. \tag{13}$$

Otherwise, $B_{UDP}(n+1)$ is adapted with

$$B_{UDP}(n+1) = \min\left[B_{UDP}^{PL}(n+1), B_{UDP}^{B_{TCP}}(n+1), B_{Target}\right]. \tag{14}$$

C. Treating Exceptions

It is worth mentioning that if $PL(0) = 0$, there is no information about the TCP transmission rate at $n = 0$, $B_{TCP}(0)$, and (9) cannot be employed to estimate $B_{UDP}(n+1)$. On the other hand, $(\Delta PL/\Delta B_{UDP})|_n$ and $(\Delta B_{TCP}/\Delta B_{UDP})|_n$ are estimated if $\Delta B_{UDP}(n) \neq 0$. In contrast, if $\Delta B_{UDP}(n) = 0$, $(\Delta PL/\Delta B_{UDP})|_n$ and $(\Delta B_{TCP}/\Delta B_{UDP})|_n$ are made equal to $(\Delta PL/\Delta B_{UDP})|_{n-1}$ and $(\Delta B_{TCP}/\Delta B_{UDP})|_{n-1}$, respectively. Finally, a lower bound for $B_{UDP}(n+1)$, $MinB_{UDP}$, is also introduced to take into consideration the fact that speech/audio/video coding schemes provide the lowest operating bit rate. Beyond this threshold, the coder cannot operate so the protocol stops the connection.

D. Choosing the Parameters Employed by the Protocol

The time interval ΔT is very important for the convergence of the algorithms described by (4) and (9). Fig. 3 shows *PL* averaged within windows with four different lengths. Packet-loss was measured by transmitting UDP packets at constant rate between the two hosts employed in this research as described in Section III. According to Fig. 3, a 10 or 15 s window gives a reasonably smooth representation of the dynamics of *PL*, so in the experiments presented here ΔT was equal to 15 s.

The adaptation rates A_{PL} and $A_{B_{TCP}}$ in (4) and (9), respectively, are intrinsic to adapting algorithms based on the gradient method. The protocol behavior is analyzed with several adaptation rates in Figs. 4 and 5. Nevertheless, higher and lower bounds are introduced according to (6), (7), (10), and (11) to guarantee the convergence of the algorithms and avoid oscillations. Some versions of the LMS algorithm re-estimate the adaptation rate in order to accelerate the convergence and achieve a

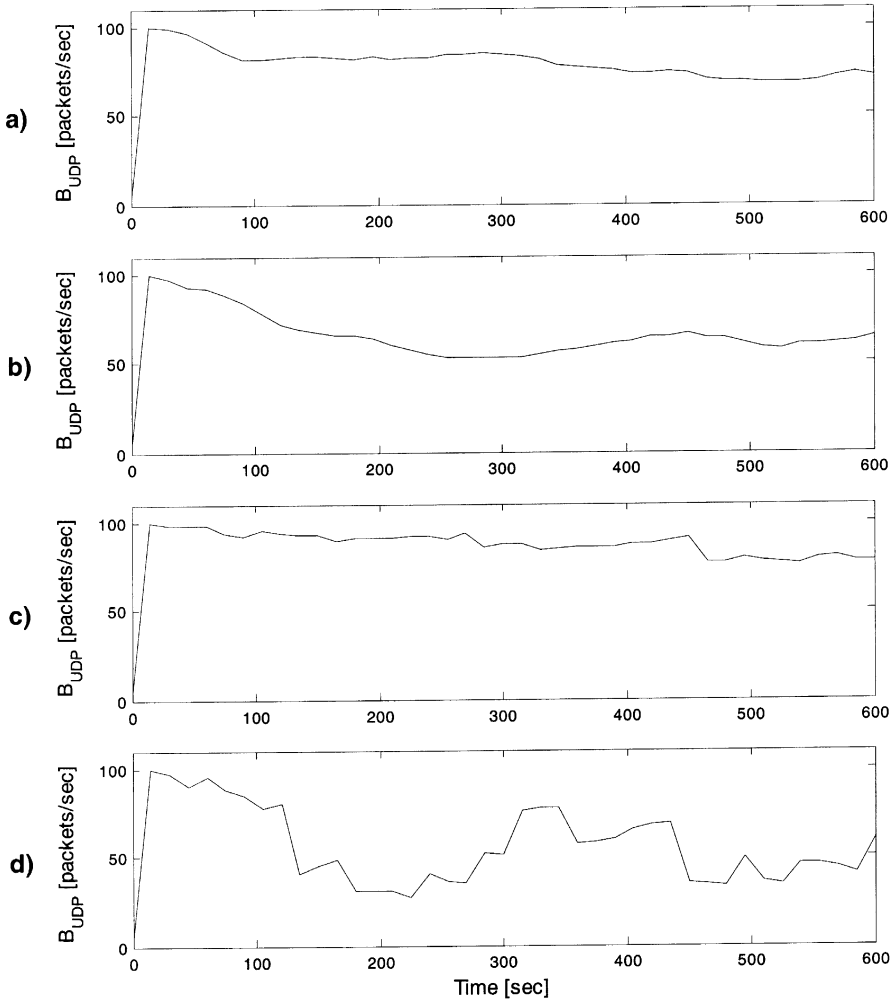

Fig. 4. UDP application bit rate, $B_{UDP}(n)$, versus time with the adaptation rates A_{PL} in (4) and $A_{B_{TCP}}$ in (9) equal to (a) 0.1; (b) 0.5; (c) 1; and (d) 5. The packet-loss target was made equal to 5%.

lower MSE. This technique was not applied here and is not considered essential to the proposal of this article, although it could be addressed as a future work. The constant k and α in (6), (7), (10), and (11), respectively, and β in (8) were chosen just as examples and do not deserve further discussions in the context of the contributions presented by this paper.

The protocol proposed in this paper attempts to return the UDP application bit rate, $B_{UDP}(n)$, given a *Target_PL* employed in (4). However, a suitable *Target_PL* is a function of the network resources and the concurrent traffic. For instance, a low packet-loss target may not be possible in some cases, and the protocol will reduce $B_{UDP}(n)$ until reaching $MinB_{UDP}$ and then finishing the connection. The authors believe that the problem of optimizing *Target_PL* should be seen as equivalent to finding the optimum pair (*Target_PL*, $B_{UDP}(n)$) from the application point of view. Every speech/audio and video coding scheme provides a perceptual quality measure as a result of (*Target_PL*, $B_{UDP}(n)$). Consequently, the estimation of *Target_PL* strongly depends on the application and is also out of the scope of this paper in order to preserve the generality of the protocol proposed here. This is also valid for *MaxPL*, $MinB_{UDP}$ and B_{Target}.

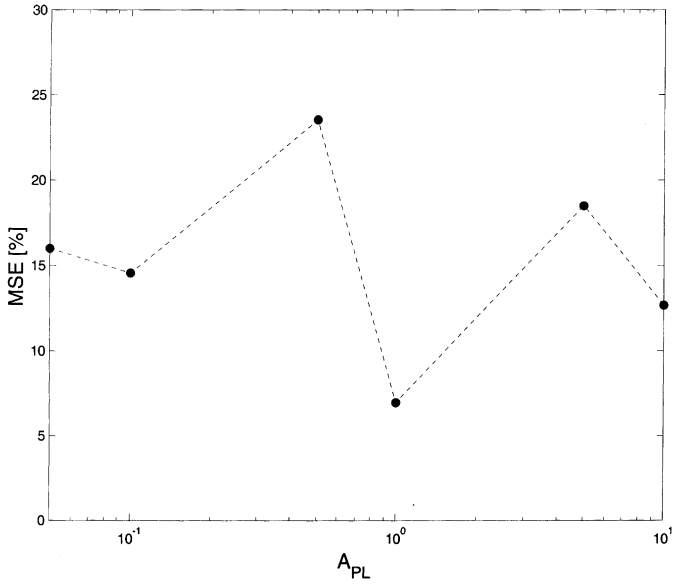

Fig. 5. MSE (in percentage) of the packet-loss target, $([Target\_PL - PL(n)]^2/Target\_PL^2) \times 100\%$, versus the adaptation rate A_{PL}.

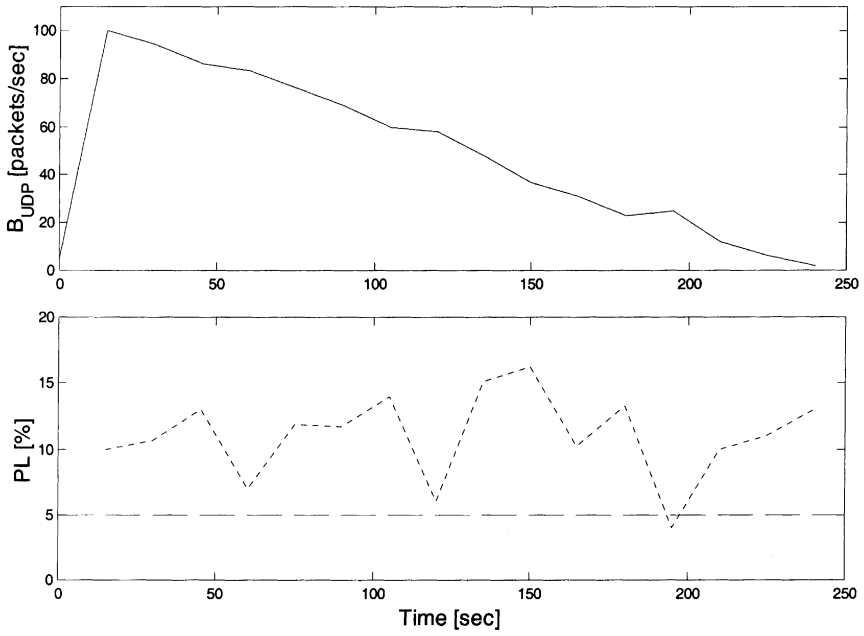

Fig. 6. UDP application bit rate, $B_{UDP}(n)$, versus time with high concurrent traffic. The adaptation rates A_{PL} in (4) and $A_{B_{TCP}}$ in (9) were equal to 1. The packet-loss target was made equal to 5%.

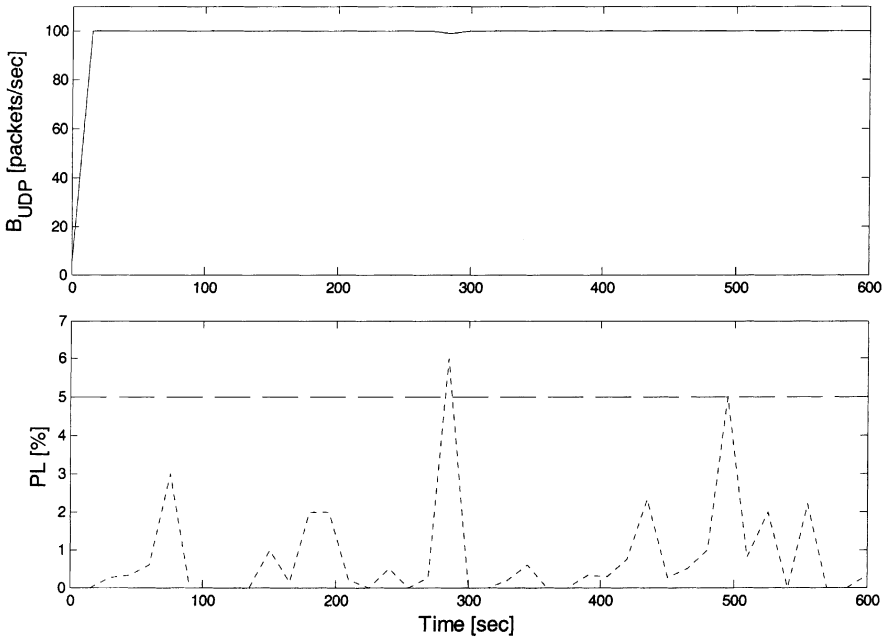

Fig. 7. UDP application bit rate, $B_{UDP}\,(n)$, versus time with low concurrent traffic. The adaptation rates A_{PL} in (4) and $A_{B_{TCP}}$ in (9) were equal to 1. The packet-loss target was made equal to 5%.

III. Experiments

The protocol proposed here was tested in a real Internet connection with a client process at the University of Chile (UCh) in Santiago, sending UDP packets to a server process at the University of New Mexico (UNM) in Albuquerque. The packet-rate, $B_{UDP}(n+1)$, from client to server was adapted with the protocol in Fig. 2, which in turn makes use of the information provided by control packets sent back by the server to the client. The following configuration was employed in the experiments reported here: $\beta = 30\%$; $MaxPL = 30\%$; $B_{Target} = 100$ packets/s; $MinB_{UDP} = 5$ packets/s; k in (6) and (10) was equal to 0.1; α in (7) and (11) was equal to 0.5; the adaptation rate A_{PL} in (4) was made equal to the adaptation rate $A_{B_{TCP}}$ in (9); and finally, $\Delta T = 15$ s. This setting is just an example of operating conditions and does not affect the generality of the protocol proposed here. Results are shown in Figs. 4–10.

As seen in Fig. 4, the protocol was able to adapt the UDP application bit rate, $B_{UDP}\,(n)$, with several adaptation rates A_{PL} in (4) and $A_{B_{TCP}}$ in (9). The fluctuations of $B_{UDP}(n)$ are certainly due to variations in the concurrent traffic. This result suggests: first, the algorithm is able to converge although the partial derivatives were roughly approximated with the

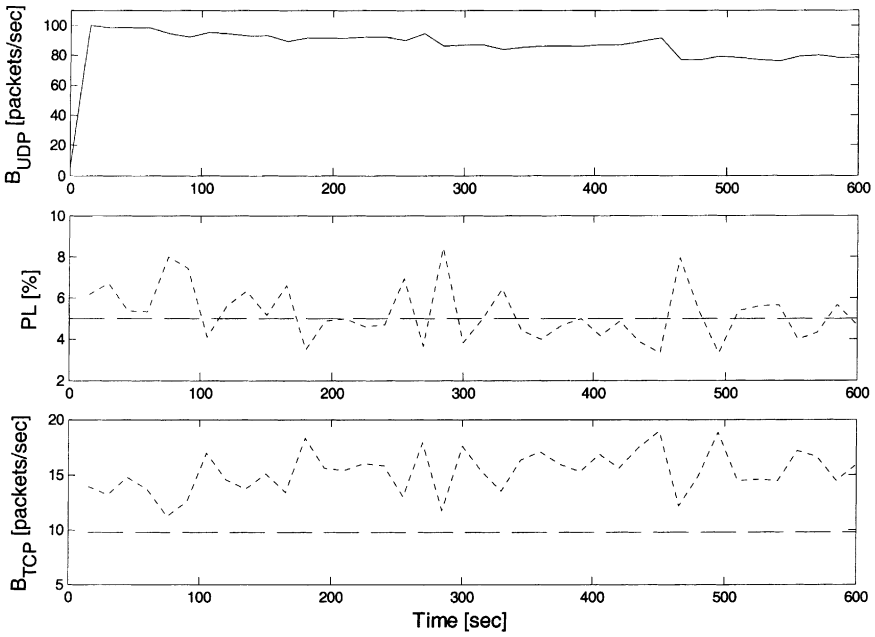

Fig. 8. Experiment with one UDP process employing the real-time protocol proposed in this paper. From top to bottom: UDP application packet-rate (B_{UDP}) versus time; *PL* versus time and the desired response $Target\_PL = 5\%$ according to (4); and, the throughput TCP bandwidth (B_{TCP}), according to (1) versus time and the threshold $(1-\beta)\cdot B_{TCP}(0)$ as in (8).

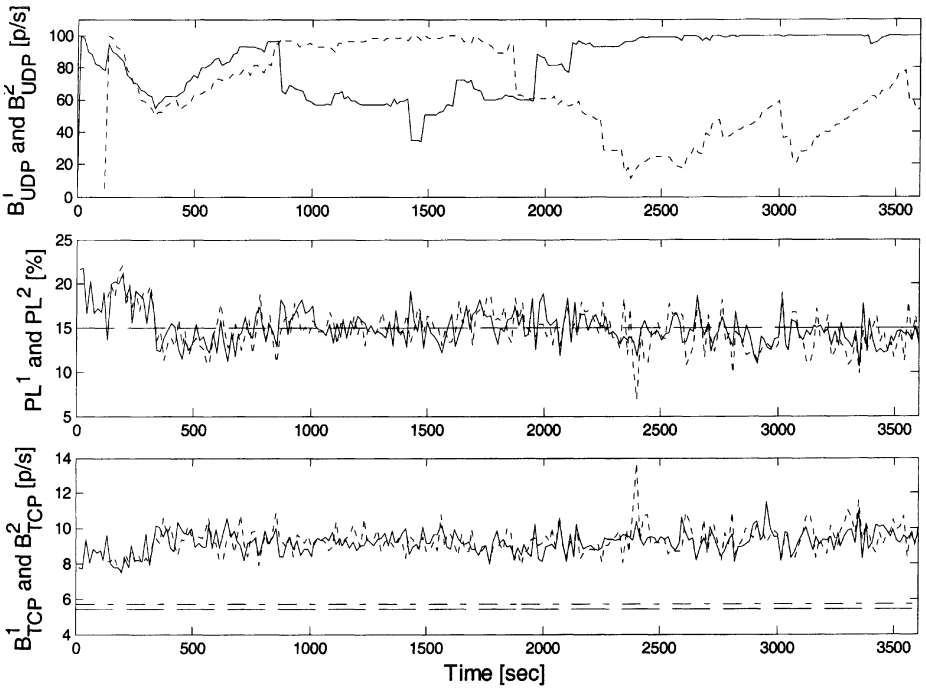

Fig. 9. Experiment with two UDP process employing the real-time protocol proposed in this paper. Parameters in $\mathrm{process}_1$ and $\mathrm{process}_2$ are shown with (———) and (– – – – – –), respectively. From top to bottom: $\mathrm{process}_1$ and $\mathrm{process}_2$ packet-rates, B^1_{UDP} and B^2_{UDP}, respectively, versus time; packet-losses, PL^1 and PL^2, versus time and the desired response $Target\_PL = 15\%$ according to (4); and, throughput TCP bandwidths, B^1_{TCP} and B^2_{TCP}, versus time and the thresholds $(1-\beta)\cdot B_{TCP}(0)$ as in (8) in $\mathrm{process}_1$ (higher) and $\mathrm{process}_2$ (lower).

ordinary derivatives; second, there is a reasonable wide range of suboptimal values defined for the adaptation rates. It is worth highlighting that the adequate use of thresholds, to impose bounds on the estimation of partial derivatives, guarantees the convergence of the algorithm.

According to Fig. 5, the MSE on packet-loss does not decrease when the adaptation rate A_{PL} is reduced, as should be expected according to [21]. This should be a result of the random behavior of the concurrent traffic, which in turn could not be kept constant due to the nature of the experiment based on a real Internet connection.

Fig. 6 shows a situation when the chosen *Target_PL* was too low due to the intensity of the concurrent traffic. In this case, the protocol reduced $B_{UDP}(n)$ until reaching $MinB_{UDP}$ and the

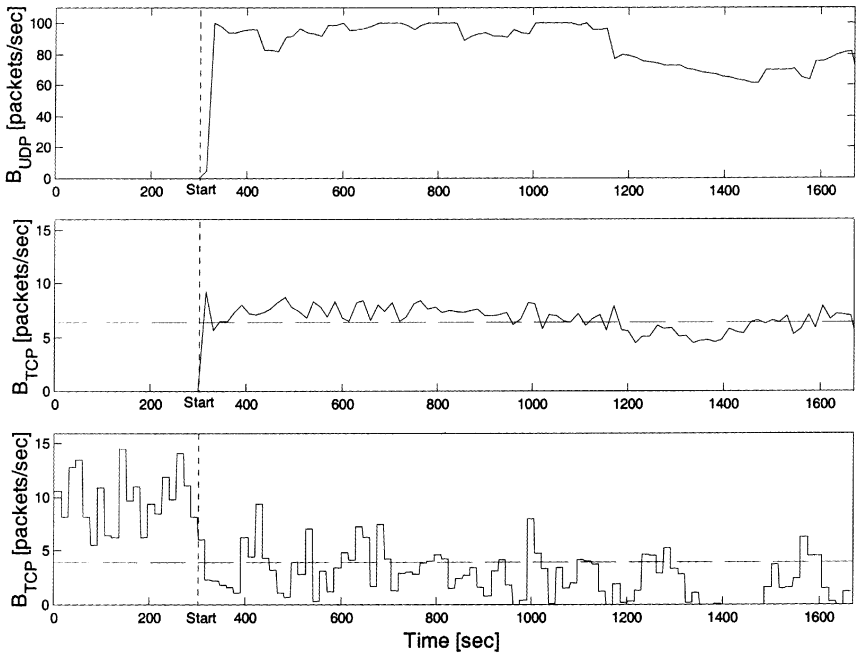

Fig. 10. Effect of starting one real-time application on a previously existing TCP process. From top to bottom: real-time UDP process packet-rate vs. time; the throughput TCP bandwidth (B_{TCP}) according to (1) vs. time and the threshold $(1-\beta)\cdot B_{TCP}(0)$ as in (8); and, the real TCP transmission rate versus time measured at the receiver, and the threshold equal to $(1-\beta)$ times the TCP packet rate when the UDP application start the transmission. The packet-loss target was made equal to 5%.

connection was ended. The opposite situation is shown in Fig. 7 where the concurrent traffic was low compared to *Target_PL* and the protocol kept $B_{UDP}(n)$ almost constant and equal to the packet-rate target $B_{Target} = 100$ packets/s.

The evolution of $PL(n)$ and $B_{TCP}(n)$ in a stable application is shown in Fig. 8. As can be seen, $PL(n)$ oscillates around *Target_PL*. In contrast, $B_{TCP}(n)$ was always above $(1-\beta)\cdot B_{TCP}(0)$, (9) was not required, and $B_{UDP}(n)$ was adapted only with (4). As mentioned above, the variation of $B_{UDP}(n)$ must be due to fluctuations of the concurrent applications. Notice that an ordinary TCP-friendly real-time protocol would transmit, in the best case, at a packet-rate similar to $B_{TCP}(n)$. It is worth highlighting that $B_{UDP}(n)$ is much higher than $B_{TCP}(n)$, although the degradation of the concurrent TCP applications was controlled. The bandwidth allocated to each TCP process would have been reduced in a maximum of $\beta = 30\%$ when compared with $B_{TCP}(0)$, which would correspond to the average bandwidth allocated to each TCP processes at $n = 0$, when the UDP process starts to transmit. This is certainly true in the two-router problem shown in Fig. 1, where all the TCP applications observe approximately the same $PL(n)$ and $RTT(n)$. However, in the experiments reported here, a real Internet connection was employed with a more complex environment than the one in Fig. 1. The TCP applications at the end router (at UCh in this case) do not measure the same *PL* and *RTT* because the packets have different destinations. As a consequence, the UDP process does not have an uniform effect on the TCP traffic: the reduction in the allocated bandwidth to TCP processes would be the highest and equal to β, in the worst case, when the TCP and UDP packets share the same path and destination; and the lowest reduction in the allocated bandwidth to TCP applications would correspond to the situation when the TCP and UDP packets share only one end router.

According to Fig. 9, when two UDP applications compete for the same network resources between UCh and UNM, it is also noticeable that $B^1_{UDP}(n)$ and $B^2_{UDP}(n)$ may not present the same trend. This is due to the facts that $PL^1(n)$ and $PL^2(n)$ can be different as a result of the short estimation interval that may provide low statistical significance, and that the replacement of the partial derivative with the ordinary one can be an inaccurate approximation in some cases. However, despite these limitations, the algorithm clearly converges, and $PL^1(n)$ and $PL^2(n)$ are kept around *Target_PL* while $B^1_{UDP}(n)$ and $B^2_{UDP}(n)$ are adapted, which in turn confirms the effectiveness of the close-loop mechanism in the protocol proposed here.

The effect of starting one real-time application on a previously existing TCP process is shown in Fig. 10. The transmission of UDP and TCP packets takes place between the hosts in UCh and UNM. As can be seen in Fig. 10, when the UDP process (top) starts the transmission the real TCP transmission rate measured at the receiver (bottom) is significantly reduced. After that, the proposed protocol adapts the UDP packet-rate in order to keep the estimated TCP transmission rate (middle), computed with (1) and employed by the protocol in condition (8), above $(1-\beta)\cdot B_{TCP}(0)$. Note that the estimated and the real TCP transmission rate are not necessarily the same due to the approximations required by (1), to the computation of *PL* and *RTT*, which in turn is very different in TCP and in the real-time protocol presented here, and to the fact that the real TCP packet-rate was observed in the receiver. When

compared with the real TCP transmission rate (bottom), the UDP packet-rate (top) is much smoother which satisfies one of the constraints mentioned in Section II. Finally, as can be seen in Fig. 10 (bottom), it is worth mentioning that the computation of *PL*(0) and RTT(0) could be done within a time interval longer than ΔT in order to have a more representative estimation of $B_{TCP}(0)$. Nevertheless, the authors consider that this modification is not relevant in the context of the contributions presented here.

Note that the estimation of $B_{UDP}(n+1)$ according to (4) and (9) computes $\Delta B_{UDP}(n+1)$ that reduces the error on *PL* and satisfies the constraints (8). This is consistent with the nature of speech/audio/video transmission in real-time, which is certainly very sensitive to discontinuities in the allocated bandwidth and should be preserved from abrupt transitions.

The full-duplex transmission problem involves two real-time protocols transmitting in opposite directions. If both applications adapt $B_{UDP}(n+1)$ independently, the hosts may end up transmitting at different rates, since $PL(n)$ and $RTT(n)$ are not necessarily the same at both sides. This problem could easily be overcome by setting the transmission rate as the lowest $B_{UDP}(n+1)$ estimated in both directions.

IV. Conclusions

A real-time protocol is proposed based on the assumption that the UDP packet rate should be adapted to reduce the error between packet-loss and the target, and to prevent the application from using more bandwidth than a given percentage of the bandwidth allocated to TCP processes. Both constraints are implemented with the LMS algorithm and the method proved to be effective in a real Internet connection. One of the main limitations of the approach proposed here is the estimation of the partial derivative, needed by the LMS algorithm, with the ordinary derivative. Nevertheless, the proper use of thresholds, to impose bounds on the estimation of partial derivatives, guarantees the convergence of the algorithm. On the other hand, the scheme can be seen as a compromise between the conservative TCP-friendly protocols and the ordinary UDP open-loop scheme, and has not been found in the specialized literature. When compared with the Fast Least Squares (FLS) algorithm, which is an efficient implementation of the Recursive Least Squares (RLS) technique, the LMS algorithm requires five times less computational load [1]. Nevertheless, the adaptive algorithm computes the UDP transmission rate every ΔT that is equal to 15 seconds in this paper, so the discussion on computational complexity is not relevant. However, the RLS family of algorithms can provide a faster convergence than the classic LMS, and the applicability of those algorithms are proposed as a future work.

The approach covered in this paper does not need an admission control mechanism, and can be considered an interesting contribution to the problems of integrating TCP and UDP traffic, and QoS allocation, due to its generality, effectiveness and simplicity. Finally, a well-known technique in adaptive signal processing is applied to communications network problem, which in turn may start a new research field.

Acknowledgment

The authors would like to thank Dr. B. Pellom from the University of Colorado, Boulder, for having proofread this manuscript. The authors also wish to thank Dr. F. McInnes from the University of Edinburgh, U.K., for the discussions on multimedia applications on packet networks and stochastic modeling while he was visiting the Department of Electrical Engineering at University of Chile, and Prof. R. Jordan and J. Salas, from the University of New Mexico, Albuquerque, for having provided the remote host employed in the experiments reported here.

References

[1] M. G. Bellanger, "Computational complexity and accuracy issues in fast least squares algorithms for adaptive filtering," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 3, June 1988, pp. 2635–2639.

[2] S. Bhattacharyya, D. Towsley, and J. Kurose, "A novel loss indication filtering approach for multicast congestion control," *J. Comput. Commun., Special Issue on Multicast*, 2000.

[3] J. Byers, M. Frumin, G. Horn, M. Luby, M. Mitzenmacher, A. Roetter, and W. Shaver, "Flid-dl: Congestion control for layered multicast," in *Proc. Second Int. Workshop on Networked Group Communication (NGC 2000)*, Palo Alto, CA, Nov. 2000.

[4] S. Kasera, S. Bhattacharyya, M. Keaton, D. Kiwior, J. Kurose, D. Towsley, and S. Zabele, "Scalable fair reliable mulitcast using active services," *IEEE Network Mag., Special Issue on Multicast*, vol. 14, no. 1, pp. 48–57, Jan./Feb. 2000.

[5] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," in Proc. ACM SIGCOMM'98,, 1998.

[6] J. Padhye, D. Towsley, J. Kurose, and R. Koodli, "A model based TCP-friendly rate control protocol," in *Proc. Int. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Basking Ridge, NJ, June 1999.

[7] R. Rejaie, M. Handley, and D. Estrin, "RAP: An end-to-end rate-based congestion control mechanism for real time streams in the internet," in Proc. INFOCOMM'99, 1999.

[8] I. Rhee, N. Balaguru, and G. Rouskas, "Mtcp: Scalable tcp-like congestion control for reliable multicast," *Proc. IEEE INFOCOM*, vol. 3, pp. 1265–1273, Mar. 1999.

[9] L. Rizzo, "pgmcc: A tcp-friendly single-rate multicast congestion control scheme," *ACM SIGCOMM*, Aug. 2000.

[10] J. W. Roberts, "Traffic theory and the internet," *IEEE Commun. Mag.*, vol. 39, no. 1, pp. 94–99, Jan. 2001.

[11] D. Sisalem, H. Schulzrinne, and F. Emanuel. (1997) The Direct Adjustment Algorithm: A TCP-Friendly Adaptation Scheme. GMD-FOKUS. [Online]. Available: http://www.fokus.gmd.de/usr/sisalem

[12] D. Sisalem and A. Wolisz, "LDA+ TCP-friendly adaptation: A measurement and comparison study," presented at the The 10th Int. Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'2000), Chapel Hill, NC, June 25–28, 2000.

[13] ——, "MLDA: A TCP-friendly congestion control framework for heterogeneous multicast environments," presented at the Eighth Int. Workshop of Quality of Service (IWQoS 2000), Pittsburgh, PA, June 2000.

[14] W. Stallings, *High-Speed Networks. TCP/IP and ATM Design Principles*. Englewood Cliffs, NJ: Prentice-Hall, 1998.

[15] W. Tan and A. Zakhor, "Error control for video multicast using hierarchical fec," presented at the Int. Conf. Image Processing, Oct. 1999.

[16] A. S. Tanenbaum, *Computer Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[17] T. Turletti, S. Parisis, and J. Bolot, "Experiments With a Layered Transmission Scheme Over the Internet," INRIA, France, RR-3296, 1997.

[18] L. Vicisano, J. Crowcroft, and L. Rizzo, "Tcp-like congestion control for layered multicast data transfer," *Proc. IEEE INFOCOM*, vol. 3, pp. 996–1003, Mar. 1998.

[19] H. A. Wang and M. Schwartz, "Achieving bounded fairness for multicast and tcp traffic in the internet," presented at the ACM SIGCOMM, 1998.

[20] J. Widmer, R. Denda, and M. Mauve, "A survey on TCP-friendly congestion control," *IEEE Network*, vol. 15, pp. 28–37, May–June 2001.

[21] B. Widrow and S. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.

[22] K. Yano and S. McCanne, "A window-based congestion control for reliable multicast based on tcp dynamics," presented at the ACM Multimedia, Los Angeles, CA, Oct. 2000.
[23] Q. Zhang, W. Zhu, and Y. Zhang, "Network-adaptive rate control with Tcp-friendly protocol for multiple video objects," in IEEE Int. Conf. Multimedia and Expo (ICME) 2000, New York, July 2000.
[24] ——, "Resource allocation for multimedia streaming over the internet," *IEEE Trans. Multimedia*, vol. 3, pp. 339–355, Sept. 2001.
[25] W. Zhu, Q. Zhang, and Y. Zhang, "Network-adaptive rate control with unequal loss protection for scalable video over internet," in *Proc. ISCAS, 2001 IEEE Int. Symp. Circuits and Systems*, vol. 5, 2001, pp. 109–112.

Néstor Becerra Yoma (M'03) was born in Santiago, Chile, in 1964. He received the B.Sc. and M.Sc. degrees from UNICAMP (Campinas State University), Sao Paulo, Brazil, and the Ph.D. degree from the University of Edinburgh, U.K., all in electrical engineering, in 1986, 1993, and 1998, respectively.

In 1998 and 1999, he was a Postdoctoral Researcher at UNICAMP and a full-time Professor at McKenzie University, Sao Paulo. Since 2000, he is a Professor with the Department of Electrical Engineering, University of Chile, Santiago, where he is currently lecturing on telecommunications and speech processing, and working on robust speech recognition/speaker verification, dialog systems, and voice over IP. At the University of Chile, he has set up the Laboratory of Speech Processing and Transmission to study speech technology applications on the Internet and telephone line. His research interests include speech processing, real time Internet protocols; QoS, and usability evaluation of interfaces. He is the author of the stochastic weighted Viterbi algorithm.

Dr. Becerra Yoma is a member of the International Speech Communication Association.

Juan Hood was born in Santiago, Chile, in 1975. He received the B.Sc. degree in electrical engineering with the highest distinction from the University of Chile, Santiago, in 2002.

In 2001 and 2002, he was a Research Assistant in the Laboratory of Speech Processing and Transmission, Department of Electrical Engineering, University of Chile, where he worked on real-time protocols. His research interests include communications networks, quality-of-service, wireless communications, systems, and control. Since March 2003, he has been a Project Engineer at Codelco, the Chilean state copper company, Rancagua.

Carlos Busso was born in Santiago, Chile, in 1977. He received the M.Sc. degree from the Department of Electrical Engineering, University of Chile, in 2003, and is currently pursuing the Ph.D. degree at the University of Southern California, Los Angeles.

He was a Research Assistant at the Laboratory of Speech Processing and Transmission, Department of Electrical Engineering, University of Chile, from 2000 to 2003. He is working on speech coding, voice over IP, low bit rate coding distortion in speech recognition and real-time protocols for the Internet. His research interests include digital signal processing, speech processing and communications networks.