An UDP-based real time protocol for the Internet

Nestor Becerra Yoma, Juan Hood Llanos, Carlos Busso Recabarren*
University of Chile, Santiago, Chile

Abstract—**Generally, real-time applications based on UDP protocol generate large volumes of data and are not sensitive to network congestion. In contrast, 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 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 a real-time protocol is proposed to employ the reduction in bandwidth allocated to TCP applications and packet-loss as desired responses to adapt the UDP packet-rate. The method is tested in a real Internet connection and seems to offer a good compromise between conservative TCP-friendly protocols, and the ordinary UDP open-loop scheme.**

I. Introduction

Real-time applications usually require constant bit rates, low delay and jitter, and are implemented with the 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 TCP traffic, which is considered "well-behaved" because it prevents the network from congestion by means of closed-loop control of packet-loss and round-trip-time. In addition, although the IPV4 suit of protocols do not provide QoS, priority to UDP packets could cause even higher degradation to TCP traffic (Roberts, 2001). To address this problem, several real-time protocols (Padhye et al., 1999)(Rejaie et al., 1999)(Sisalem et al.,1997) (Sisalem & Wolisz, 2000) (Sisalem & Wolisz, 2000-2) (Zhang et al., 2000) have been proposed to make UDP applications behave like TCP traffic and to help keep the network stable. These TCP-friendly protocols 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. 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 (Roberts, 2001) but none of these is currently being employed on the Internet (Roberts, 2002).

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 estimates the bandwidth allocated to a real-time application as a percentage of the current TCP traffic, and also employs *PL* as a criteria to bound the UDP bit rate or deny access to a new real-time application. Moreover, instead of exponentially or linearly modifying the packet-rate according to *PL* as done by TCP protocols (Tanenbaum, 1996) (Stallings, 1998), 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.

II. Observing QoS Parameters of Integrated UDP and TCP Traffic

In order to measure QoS parameters (*PL* and *RTT*), UDP packets were sent from a host at University of Chile (UCh), in Santiago, to a host at the University of New

* Nestor Becerra Yoma, Juan Hood Llanos and Carlos Busso Recabarren are with the Electrical Engineering Department, University of Chile, Av. Tupper 2007, P.O.Box 412-3, Santiago, CHILE. E-mails: {nbecerra, jhood}@cec.uchile.cl, Tel. +56-2-678 4205, Fax: +56-2-695 3881

Mexico (UNM), in Albuquerque, USA. The process at UCh, client, simulated a real-time application by generating UDP traffic with variable packet-rate (B_{UDP}) and constant packet-size. The process at UNM, server, received the packets, computed statistics related to *PL* that were sent back to the host at UCh with control packets. These control packets were also used to estimate *RTT*. This mechanism is equivalent to the RTP protocol (Stallings, 1998). The access to Internet from the host at UCh had a maximum bandwidth of 3Mbps.

In the experiment the client process started to send UDP packets at rate of 10 packets/sec and increased the transmission speed in 10 packets/sec every 20 sec until 100 packets/sec. Results are presented in Fig. 1 where B_{UDP}, *PL*, *RTT*, and the throughput TCP bandwidth (B_{TCP}) are shown. B_{TCP} corresponds to the packet-rate offered by a TCP application estimated with *RTT* and *PL* with (Padhye et al., 1998):

$$B_{TCP} = \frac{1}{RTT} \cdot \sqrt{\frac{3}{4 \cdot PL}} \qquad (1)$$

As can be seen in Fig. 1, when the UDP packet rate (B_{UDP}) increased, only a small variation in *PL* and *RTT* was observed. This should be due to the fact that the concurrent TCP traffic adapted its transmission, while the UDP process increased the demand for bandwidth. As a consequence, in this scenario *PL* would detect high congestion only when the UDP application took over all the bandwidth, and the TCP traffic was reduced to zero. Also in Fig. 1, it is possible to observe that B_{UDP} easily exceeds B_{TCP}, which suggests that a TCP-friendly protocol provides a too conservative strategy to estimate the real-time application transmission rate.

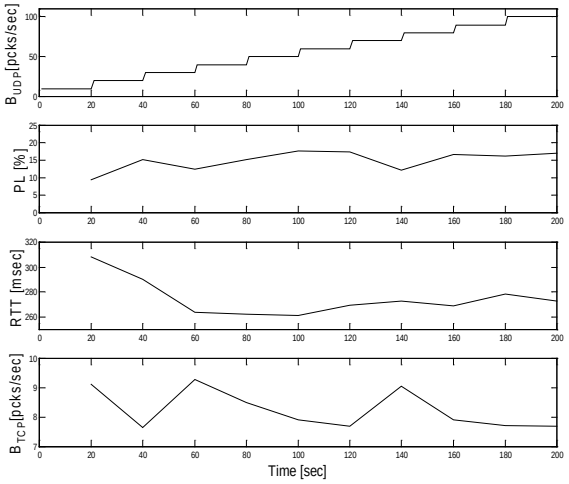

Figure 1: Experiment with one UDP process. From top to bottom: UDP application packet-rate (B_{UDP}) vs. time; packet-loss (*PL*) vs. time; round-trip-time (*RTT*) vs. time; and, the throughput TCP bandwidth (B_{TCP}), according to (1), vs. time.

III. A Real-time Protocol to Combine UDP and TCP Traffic

Based on the results discussed in the pervious section, a protocol was designed to address the problem of UDP and TCP applications between two routers (Fig. 2). In this scenario there are a finite number of TCP sources for traffic when one or more UDP application are introduced. The problem is how to estimate the packet-rate of the UDP traffic with the following constraints:

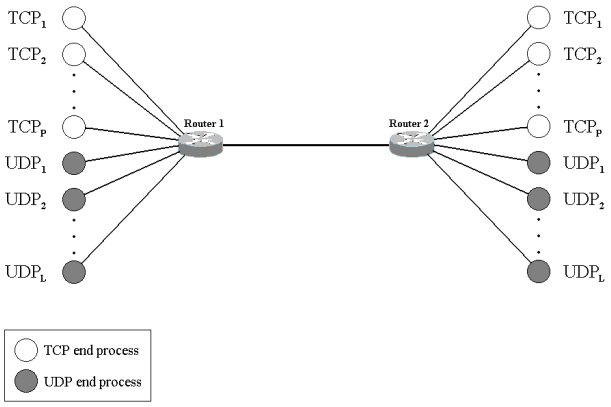

Figure 2: The two-router problem.

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 acknowledgement should be used. This mechanism contributes to congestion of the network.

The proposed transmission protocol is shown in Fig. 3. 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 t=0, when the real-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 application. Then $B_{TCP_p}(0)$, the packet-rate of TCP application p at time t=0, is computed with (1), which corresponds to the TCP flow throughput in packets/sec. Considering that the routers in Fig. 2 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, $B_{TCP_p}(0)$ is the same in all the TCP applications, and can be denoted just with $B_{TCP}(0)$. Therefore, it is reasonable to suppose that the total available bandwidth in terms of packets/second at $t = 0$, if there is not any other UDP application, is given by:

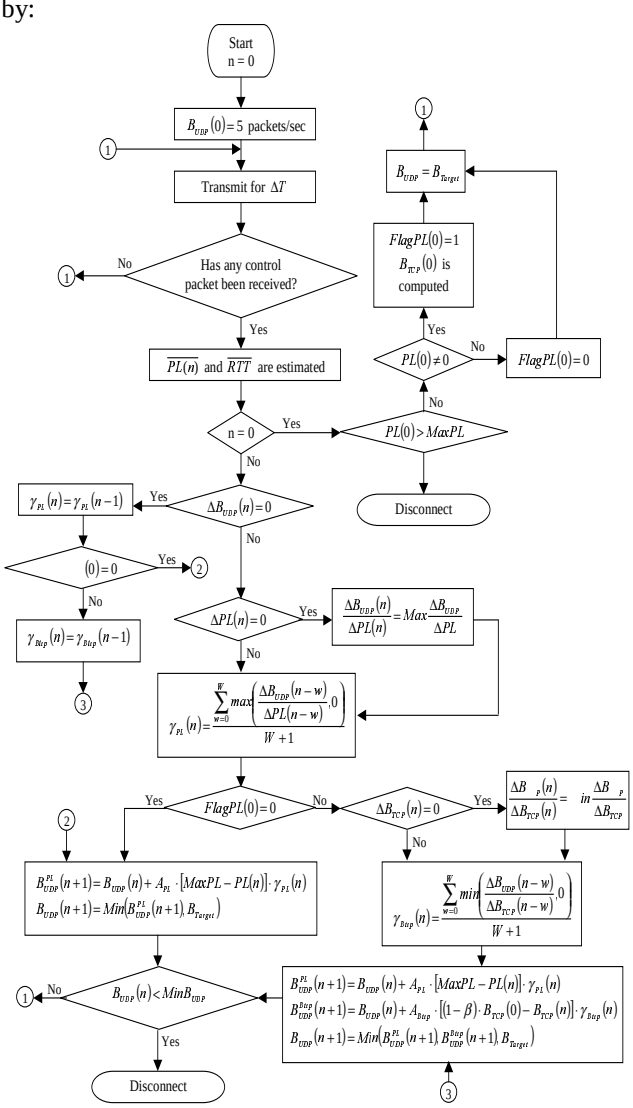

Figure 3: The real-time protocol.

$$B_T(0) = TotalB_{TCP}(0) = \sum_{p=1}^{P} B_{TCP_p}(0) = P \cdot B_{TCP}(0) \qquad (2)$$

where the $TotalB_{TCP}$(t) is total bandwidth allocated to TCP processes at time t, and P is the number of TCP applications. Assuming that the TCP processes in Fig. 2 are using all the network capacity at $t = 0$, and that the network resources do not change, the total available bandwidth remains constant, and:

$$B_T(t) = B_T(0) = TotalB_{TCP}(0) \qquad (3)$$

The protocol discussed here attempts to share a percentage of $TotalB_{TCP}(0)$ if PL is lower than an upper bound $MaxPL$. When $PL(0) > MaxPL$ the application stops the connection. If $PL(0) \leq MaxPL$, the protocol initially transmits at the target application packet-rate, B_{Target}, and then adapts $B_{UDP}(t)$ to get a percentage, β, of $TotalB_{TCP}(0)$ and to satisfy PL(t)=MaxPL. If there was no other UDP application, β corresponds to a fraction of $B_T(0)$, the total available bandwidth, according to (2), and $B_{TCP}(t) = (1-\beta)\cdot B_{TCP}(0)$, with t>0. Notice that $B_{TCP}(t)$, the average bandwidth allocated to each TCP process, would decrease geometrically with the number of UDP applications. This behavior seems to be a reasonable compromise 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.

As mentioned above, $B_{UDP}(t)$ is adapted every ΔT sec in order to satisfy the following restrictions:

$$PL(n) = MaxPL \qquad (4)$$

$$B_{TCP}(n) = (1-\beta)\cdot B_{TCP}(0) \qquad (5)$$

where n=0, 1,2,... is a non-negative integer that denotes discrete sequences with $t = n \cdot \Delta T$; $\beta < 1$ is the percentage of bandwidth taken from the TCP processes; $B_{TCP}(n)$ is estimated according to (1); and $PL(n)$ and $RTT(n)$ are computed every ΔT sec by means of control packets sent by the receiver (from the server to the client). The first constraint takes into consideration a maximum level of PL to protect the rest of applications, and to comply with the specifications of the speech/audio/video scheme. The second condition, as mentioned above, compels the UDP application to use only a given percentage of $TotalB_{TCP}(0)$. The estimation of $B_{UDP}(n+1)$ according to the restrictions (4) and (5) is made by

$$B^{PL}_{UDP}(n+1) = B_{UDP}(n) + A_{PL} \cdot [MaxPL - PL(n)] \cdot \gamma_{PL}(n) \qquad (6)$$

$$B^{Btcp}_{UDP}(n+1) = B_{UDP}(n) + A_{Btcp} \cdot [(1-\beta)\cdot B_{TCP}(0) - B_{TCP}(n)] \cdot \gamma_{Btcp}(n) \qquad (7)$$

where A_{PL} and A_{Btcp} are the adaptation rates; $\gamma_{PL}(n)$ and $\gamma_{Btcp}(n)$ are estimating according to:

$$\gamma_{PL}(n) = \frac{\sum_{w=0}^{W} max\left(\frac{\Delta B_{UDP}(n-w)}{\Delta PL(n-w)}, 0\right)}{W+1} \qquad (8)$$

$$\gamma_{Btcp}(n) = \frac{\sum_{w=0}^{W} min\left(\frac{\Delta B_{UDP}(n-w)}{\Delta B_{TCP}(n-w)} \ , \ 0 \right)}{W+1} \qquad (9)$$

where : $\Delta B_{UDP}(n) = B_{UDP}(n) - B_{UDP}(n-1)$

$\Delta PL(n) = PL(n) - PL(n-1)$

$\Delta B_P \ (n) = B_P \ (n) - B_P \ (n-1)$

The average operation and the bound equal to 0 in (6) and (7) are introduced because $PL(n)$ and $B_{TCP}(n)$ do not depend only on $B_{UDP}(n)$, and are also functions of the concurrent traffic that is not known. It is expected that the cross-correlations $E[\Delta PL(n) \cdot \Delta B_{UDP}(n)]$ and $E[\Delta B_{TCP}(n) \cdot \Delta B_{UDP}(n)]$ would be positive and negative, respectively. Nevertheless, due to the current traffic behavior, $PL(n)$ and $B_{TCP}(n)$ could decrease and increase, respectively, when $B_{UDP}(n)$ increases. Finally, $B_{UDP}(n+1)$ is given by,

$$B_{UDP}(n+1) = min\left[B_{UDP}^{PL}(n+1), B_{UDP}^{Btcp}(n+1), B_{Target} \right] \qquad (10)$$

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 (7) cannot be employed to estimate $B_{UDP}(n+1)$. Finally, if $\Delta PL(n) = 0$, $\frac{\Delta B_{UDP}(n)}{\Delta PL(n)}$ is made equal to $Max\frac{\Delta B_{UDP}}{\Delta PL}$; and, if $\Delta B_P \ (n) = 0$, $\frac{\Delta B_{UDP}(n)}{\Delta B_P \ (n)}$ is made equal to $Min\frac{\Delta B_{UDP}}{\Delta B_P}$.

Notice that the estimation of $B_{UDP}(n+1)$ according to (6) and (7) computes $\Delta B_{UDP}(n+1)$ that satisfies the constraints (4) and (5). This is also interesting for real-time applications that attempt to sustain a constant bit rate. In contrast, TCP-friendly protocols (Padhye et al., 1999)(Rejaie et al., 1999)(Sisalem et al., 1997) (Sisalem & Wolisz, 2000) (Sisalem & Wolisz, 2000-2). (Zhang et al., 2000) usually introduce discontinuities in the packet-rate depending on the packet-loss and round-trip-time. The transmission of speech/audio/video in real time is certainly very sensitive to discontinuities in the allocated bandwidth that should be preserved from abrupt transitions.

A lower bound for $B_{UDP}(n+1)$, $MinB_{UDP}$, is also considered 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.

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 can easily be overcome by setting the transmission rate as the lowest $B_{UDP}(n+1)$ estimated in both directions.

IV. EXPERIMENTS

The protocol proposed here was tested in the same environment described in section II. A client process at UCh in Santiago, Chile, sent UDP packets to a server process at UNM in Albuquerque, USA. The packet-rate, $B_{UDP}(n+1)$, from client to server was adapted with the protocol in Fig. 3, which in turn makes use of the information provided by control packets sent back by the server to the client. Results are presented in Fig.4. Figure 4 shows $B_{UDP}(n)$, $PL(n)$ and $B_{TCP}(n)$ with only one UDP process between both hosts. The following configuration was employed in the experiments reported here: $W=1$; $\beta = 30\%$; $MaxPL = 15\%$; $Max\frac{\Delta B_{UDP}}{\Delta PL} = 7$; $Min\frac{\Delta B_{UDP}}{\Delta B_{TCP}} = -7$; $MinB_{UDP} = 5$ packets/sec; and finally, $\Delta T = 15$ sec.

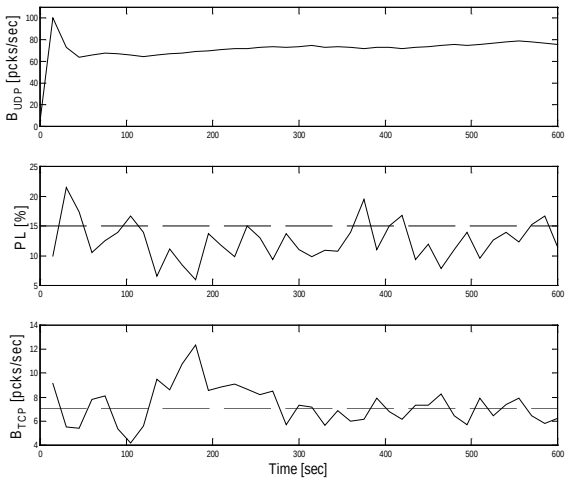

Figure 4: Experiment with one UDP process employing the real-time protocol proposed in this paper. From top to bottom: UDP application packet-rate (B_{UDP}) vs. time; packet-loss (PL) vs. time and the desired response $MaxPL = 15\%$ (dashed line) according to (4); and, the throughput TCP bandwidth (B_{TCP}), according to (1), vs. time and the desired response $(1-\beta) \cdot B_{TCP}(0)$ (dashed line) as in (5).

As can be seen in Fig. 4, $PL(n)$ and $B_{TCP}(n)$ converge to $MaxPL$ and $(1-\beta) \cdot B_{TCP}(0)$, respectively,

while $B_{UDP}(n)$ is adapted with (6) and (7). The variation of $B_{UDP}(n)$ must be due to fluctuations of the TCP applications, which in turn suggests that the proposed protocol is sensitive to concurrent traffic and network congestion. Notice that a TCP-friendly protocol would transmit at a packet-rate similar to $B_{TCP}(n) = (1-\beta) \cdot B_{TCP}(0)$. It is worth highlighting that $B_{UDP}(n)$ is much higher than $B_{TCP}(n)$, and the bandwidth allocated to TCP traffic was reduced in β=30% when compared with $B_{TCP}(0)$, which would correspond to the average bandwidth allocated to each TCP processes at $t = 0$, when the UDP process starts to transmit. This is certainly true in the two-router problem shown in Fig.2, 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. 2. 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 when the TCP and UDP packets share the same destination and path; and the lowest reduction in the allocated bandwidth to TCP applications would correspond to the situation when the TCP and UDP packets share only the end router.

V. CONCLUSIONS

A real-time protocol is proposed based on the assumption that a reduction in average bandwidth allocated to TCP applications and packet-loss are employed as desired responses to adapt the UDP packet-rate. The method proved to be effective in a real Internet connection. The scheme seems to offer a good compromise between conservative TCP-friendly protocols and the ordinary UDP open-loop scheme, and has not been found in the specialized literature. Moreover, 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.

VI. ACKNOWLEDGEMENT

The authors wish to thank Dr. Fergus McInnes, from the University of Edinburgh, UK, 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. Ramiro Jordan and John Salas, from the University of New Mexico, USA, for having provided the remote host employed in the experiments reported here. The authors would also like to thank Dr. Mark Winskel, also from the University of Edinburgh, for having proofread this paper. The research described here was supported by Conicyt-Chile.

VII. REFERENCES

Padhye, J., Firoiu, V., Towsley, D., Kurose, J. (1998). *Modeling TCP Throughput: A Simple Model and its Empirical Validation*. Proceedings SIGCOMM'98, ACM, 1998.

Padhye, J., Towsley, D., Kurose, J., Koodli, R. (1999). *A Model based TCP-friendly Rate Control Protocol*. Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), Basking Ridge, NJ, June 1999.

Rejaie, R., Handley,M., Estrin,D. (1999). *RAP: An End-to-end Rate-based congestion control mechanism for real time streams in the Internet.* In Proceedings of INFOCOMM'99, 1999.

Roberts, J.W. (2001). *Traffic theory and the Internet*. IEEE Communications Magazine, No.1, Vol. 39, January, 2001.

Roberts, J.W. (2002). *Personal communications*. 2002.

Sisalem, D., Schulzrinne, H., Emanuel, F. (1997). *The direct adjustment algorithm: A TCP-friendly adaptation scheme*. Technical report, GMD-FOKUS, Aug. 1997. Available from http://www.fokus.gmd.de/usr/sisalem

Sisalem, D., Wolisz, A. (2000). *LDA+ TCP-Friendly Adaptation: A Measurement and Comparison Study*, in the 10th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'2000), Chapel Hill, NC, USA, June 25-28, 2000.

Sisalem, D., Wolisz, A. (2000-2). *MLDA: A TCP-friendly congestion control framework for heterogeneous multicast environments*. Eighth International Workshop of Quality of Service (IWQoS 2000), Pittsburg, PA, June 2000.

Stallings, W. (1998). *High-Speed Networks.TCP/IP and ATM design principles*. Prentice-Hall, New Jersey, 1998.

Tanenbaum, A.S. (1996). *Computer Networks*. Prentice-Hall, New Jersey, 1996.

Zhang, Q., Zhu,W., Zhang, Y. (2000). *Network-adaptive Rate Control with Tcp-friendly Protocol for Multiple Video Objects*. IEEE International Conference on Multimedia and Expo (ICME) 2000, New York, USA, July, 2000.