Robust Connections for TCP Transfers Over ATM Through an Active Protocol in a Multiagent Architecture[1]

José Luis González-Sánchez (*), Jordi Domingo-Pascual (**) and Alfonso Gazo Cervero (*)

(*) University of Extremadura. Escuela Politécnica de Cáceres Avda. Universidad S/N. (10.071) Cáceres (Spain)
Phone: +34-927.257.259 Fax:+34-927.257.202 E-Mail: jlgs@unex.es agazo@unex.es
(**) Polytechnic University of Catalunya. Campus Nord, Modul D6, Jordi Girona 1-3 (08034) Barcelona (Spain)
Phone: +34-93.401.6981 Fax: +34-93.401.7055 E-Mail: jordi.domingo@ac.upc.es

***Abstract*- TAP (Trusted and Active PDU transfers) is a new distributed architecture and a protocol for ATM networks that provides assured transfers to a set of privileged VPI/VCI. The distributed architecture manages the privileged connections and offers an improvement in performance when network connections cause some cell loss. Our AcTMs (Active ATM switch) model supports the trusted protocol. This research also offers an attractive solution to the chaotic nature of TCP Congestion Control. Several simulations demonstrate the effectiveness of the mechanism that recovers the congested PDU locally at the congested switches with better end-to-end goodput in the network. Also, the senders are alleviated of NACKs and end-to-end retransmissions.**

I. INTRODUCTION

The ATM technology is characterized by its good performance with the different traffic classes and by its negotiation capacity of the QoS (Quality of Service) parameters. Congestion causes the most common type of errors, and it is here that our work is intended to offer guaranteed transfers through our TAP (Trusted and Active Protocol) architecture. TAP adopts ARQ (Automatic Repeat Request) with NACK (Negative Acknowledgement) using RM (Resource Management) cells to alleviate the negative effect of implosion. The intermediate active nodes are responsible for local retransmissions to avoid end-to-end retransmissions. We have implemented a modification of EPD (Early Packet Discard) as a means of congestion control that we have called EPDR (Early Packet Discard and Relay) in order to alleviate the effect of congestion and PDU fragmentation. Currently, congestion control is delegated to protocols that solve it with end-to-end retransmissions such as TCP. This is an easy technique to implement at high speeds by simplifying the switches and routers, but the whole network is overloaded with the retransmissions and this does not offer protection against egoist sources. Also, we make fair assignations of bandwidth based on a delegated scheme by extending WFQ (Weighted Fair Queuing) [1] so as to reduce their complexity of implementation and achieving constant cost *O(1)*.

At present, the ATM networks are used as a technology to support all kinds of traffic, with predominance of TCP/IP protocols. Therefore we present the advantages that our mechanism of congestion retrievals can offer, not only for the native ATM traffic, but also for the traffic generated by TCP/IP sources. The TCP protocol has become in the last years the standard of data communications. As we know, TCP is a reliable protocol of the transport layer of the TCP/IP architecture that uses error control and flux control based on window mechanisms with end-to-end control [2]. A considerable amount of research has intended to integrate two technologies as different as ATM and TCP/IP; however, their integration offers [3] poor results in the behavior of TCP throughput over ATM. While ATM is a connection-oriented technology, of switched cells of 53 octets and uniform size, TCP and IP are based on routing mechanisms of segments and datagrams of variable size.

These characteristics cause a very negative effect on the throughput when the TCP segments cross ATM switches with buffer size less than the window size of TCP. This causes loss of cells and retransmissions due to timeout. Moreover, the loss of only one cell causes the loss of a TCP segment at the receiver of the communication that will request the retransmission to the source, which must resend the whole segment and not only the lost cell.

We present simulations to demonstrate the chaotic behavior of TCP when suffering congestion. We use the NS (Network Simulator) [4] to study the window mechanisms of TCP, to analyze the effects of the threshold, of the congestion window, the probability of loss, and the effect of all this on the throughput. We will see how the goodput degenerates when the congestions appear in simulated scenarios. If we introduce TAP in the network, we can improve the goodput of TCP transmissions, thus managing to reduce (or eliminate) end-to-end retransmissions and the actions of selfcontrol of the congestion window are also reduced in TCP sources.

Firstly, we shall comment on the general characteristics of TCP, and simulate several scenarios with the NS (Network Simulator). Section 3 presents the TCP characteristics over ATM and the next two sections propose the use of TAP in an IPoverATM scenario. Our conclusions are presented at the end of this paper.

0-7803-7661-7/03/$17.00©2003 IEEE

II. TCP CONGESTION CONTROL CAN BE IMPROVED

The TCP protocol is a set of algorithms that sends packets to the network without any previous reservation, but can react if any event appears. Within this set of algorithms, the Congestion Control algorithm and the Loss Segment Retrieval algorithm are the most important.

The mechanism of congestion control in TCP has two different phases: Slow Start and Congestion Avoidance. At the beginning of a connection, or in a retransmission due to the loss of segments, the size of the congestion window (CWND) is one packet, and then, is incremented to twice the size in each ACK received in RTT time. During this phase, CWND is incremented linearly, and not exponentially as in the Slow Start phase. The following aspects are basic to the functioning of TCP:

- CWND: represents the congestion window, and is a variable that limits the amount of data that TCP can send. Its size varies depending on network conditions. If the network does not discard packets due to congestion, the size of CWND increases, also allowing for the increase in the rate of transfers in traffic sources.
- INITIAL_WINDOW is the initial value of the congestion window (CWND).
- SMSS represents the maximum amount of data that a source of traffic can send.
- RWND is the maximum quantity of data that a receipt of TCP traffic can receive.
- RTT is the Round Trip Time. This determines the transmission rate of TCP, because the source sends in this time the quantity of data fixed by the CWND window.
- CURRENT_WINDOW represents the amount of information that the source sends each RTT. This window takes the lesser value of CWND and RWND.
- SSTHRESH determines which algorithm of the congestion control is used. When CWND < SSTHRESH the algorithm Slow Start is used; and when CWND ≥ SSTHRESH, the congestion control applied is determined by the Congestion Avoidance algorithm.

A source TCP fixes the amount of data to be sent by using the CWND window, and transmits a window of segments for each RTT. The TCP adjusts the size of this window depending on the conditions of the network. Thus, the size of CWND increases to twice the segments for each ACK received in Slow Start algorithm, and increases by 1/CWND for each ACK received in the Congestion Avoidance algorithm. This is the scenario in a connection without loss of segments that is illustrated in *Fig. 1*, where we can see the congestion window without loss.

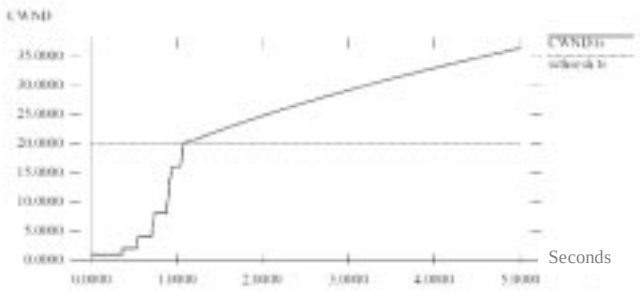

Fig.1. Evolution of CWND without losses

CWND increases exponentially while the size is less than SSTHRESH (using the Slow Start algorithm that progressively increases the number of segments (1, 2, 4...) when the ACKs are received). When the size of CWND is equal to the SSTHRESH, the congestion control of Congestion Avoidance works. Thus, the CWND window increases linearly by 1/CWND for each ACK.

The Slow Start algorithm is used by TCP to check the capacity of the network (whose capacity is unknown) and the amount of segments that it can support without congestion. When congestion is imminent, TCP passes the control to Congestion Avoidance which changes to a lineal increase of CWND until the congestion is detected.

A. Reaction in case of congestion

When the capacity of transmission of the network is less than the amount of information to be transmitted, the network will discard segments so that the sources reduce the volume of generated information. TCP detects a discarded segment caused by congestion when the number of repeated ACKs received is 3, or when the timeout of retransmissions expires. TCP reacts by resetting SSTHRESH to the half of CWND, and reduces CWND to the INITIAL_WINDOW, thus decreasing the quantity of sent segments.

Fig. 2 represents this scenario showing the evolution of CWND with losses in the network due to congestion. *Fig. 1* and *Fig. 2* are obtained with the simulation of the same topology over NS. This topology has 7 nodes. Each link has a bandwidth of 1 Mbps, a delay of 10 ms and uses DropTail as the queue type. The difference between the two scenarios is that *Fig. 2* introduces loss probability of 0.02 with error models at links 2-3 and 3-4 of the topology.

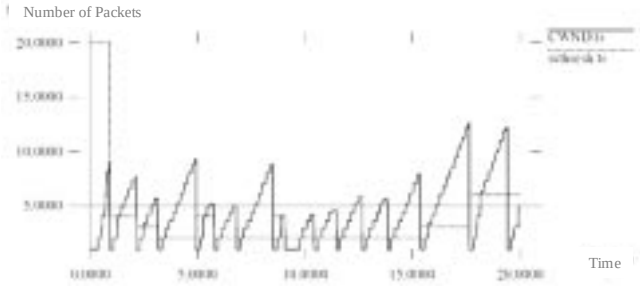

Fig. 2. CWND and SSTHRESH in a congested network

In *Fig. 2* we have used a higher time of simulation (20 seconds) to show the effect of losses over the congestion window that is reduced to 1 at several points to solve congestion problems.

We should point out that the TAP protocol solves these problems of loss that affect the decrease in size of the congestion window and also the subsequent retransmission of end-to-end losses. Thus, the source will not be obliged to reduce and to adjust its rate of transmission all the time as *Fig. 2* shows, and also, when congestions appear, these are solved locally in the affected nodes.

B. Throughput and losses

If it is supposed that the maximum size of CWND window is *W* segments and, according to the definition of the Congestion Avoidance algorithm, in [5] it can be deduced that the total quantity of delivered data in the network in each cycle can be calculated by the expression,

$$(\tfrac{W}{2})^2 + \tfrac{1}{2}(\tfrac{W}{2})^2 = \tfrac{3}{8}W^2 \quad (1)$$

Therefore, we can suppose a network without losses and a constant RTT, because it has enough bandwidth and a low total charge that does not overflow the queues. According to [5], we can estimate the random packets lost with a *P* constant probability assuming that the link delivers *1/P* consecutive packets approximately followed by a discard, without considering the transmitted data during retrievals.

We have then two approximations of the delivery of packets, expression (1) and *1/P*. So, by adding both approximations, we obtain,

$$W = \sqrt{\tfrac{8}{3p}} \quad (2)$$

On the other hand, we can apply these known data to the next expression (3) that calculates the transmitted bandwidth (where *MSS* is the maximum segment size of TCP),

$$AB = \frac{datos/ciclo}{tiempo/ciclo} = \frac{MSS * \frac{3}{8}W^2}{RTT * \frac{W}{2}} = \frac{MSS/P}{RTT * \sqrt{\frac{2}{3P}}} \quad (3)$$

We can reorganize equation (3) by grouping the constant term $K = \sqrt{\frac{3}{2}}$; thus obtaining,

$$AB = \frac{MSS}{RTT}\frac{K}{\sqrt{P}} \quad (4)$$

Equation (4) expresses the throughput in the network and calculates the performance of TCP after all previous simplifications. Paper [5] presents other references with several approximations to the value of the *K* constant regardless of its value that is always less than 1. We can finally choose the next equation (5) as a good expression of the TCP throughput,

$$AB < \left(\frac{MSS}{RTT}\right)\frac{1}{\sqrt{P}} \quad (5)$$

We have also analyzed the TCP improvement in several situations and we planned the effect and behavior of throughput, studied in relation to the probability of loss of packets. Therefore, we have reorganized the expression (5) of the Congestion Avoidance algorithm which expresses the "*steady state*" behavior of TCP over low conditions of charge and a moderate loss of packets, in the next general equation,

$$TH = \frac{MSS}{RTT\sqrt{P}} \quad (6)$$

If in (6) *TH* represents the throughput, and we consider the values of *MSS* and *RTT* to be constant, we can obtain equation (7) where we show that the throughput is inversely proportional to the loss probability,

$$TH = \frac{K}{\sqrt{P}} \quad (7)$$

Thus, equation (7) represents the behavior experienced by the throughput when there are lost packets at congested routers. In equation (7) we can obtain a new equation that estimates the size of the *W* window used by TCP with an average loss rate *P*. So, by adjusting the value of the *K* constant in (4), we obtain the next equation which is the result of the adaptation of (7) in [5] and the proposals presented in [6],

$$W = \frac{0{,}866}{\sqrt{P}} \quad (8)$$

We can reorganize (8) and according to reference [7] we will obtain the average loss rate *P* in the next equation,

$$P = \frac{0{,}75}{W^2} \quad (9)$$

Equation (8) can be understood as if the network discards a percentage of segments independently of actions that have been performed by the source. So, this describes how the source can react.

In order to study this behavior, we have simulated with NS a new scenario with links of 2 Mbps of throughput, with delays of 10 ms. and a DropTail queue. We have also associated at each link an initial loss probability of 0.001 that is incremented by 5% every 5 seconds at all links. The objective of this scenario is to study the behavior of the throughput in relation to the *P* error probability. As a

result, we have obtained the graph in *Fig. 3*, where we can see again the macroscopic behavior that *Fig. 1* indicates intuitively. We can see in *Fig. 3* how the loss probability *P* determinates the throughput of the TCP source, as the (8) previous expression intuitively indicates. When the loss probability increases, the throughput decreases logarithmically to achieve a lineal evolution. The negative logarithmic slope represents the fall of the throughput in the network when this is experiencing loss of packets. The problem is that TCP duplicates the intervals of retransmission times between successive loss of packets. So, if we do not consider the RTT value in equation (6) to be constant, the effect shown in *Fig. 3* will be even more negative for the throughput of the network. TAP proposes to decrease the loss, to improve the throughput, because as we can see in *Fig. 3*, small improvements in loss probability can mean higher gains in throughput.

III. TCP OVER ATM

The research in the throughput evaluation of TCP over ATM is divided into three main groups [8]: 1) those research papers studying the dynamism of TCP; 2) those analyzing the throughput of ATM; and 3) those observing the interaction between TCP windows and the mechanisms of congestion control of the ATM layer. Although the throughput evaluation of TCP over ATM has been the objective of several research papers, the proposals only solve particular problems such as the fragmentation of TCP, the buffers required, the interaction between congestion schemes of TCP and ATM, and the degradation of TCP. There are a lack of proposals to solve all or even some of these problems. Our research has looked into these aspects and offers a MAS (MultiAgent System), optimizing the goodput with an improvement of entry queues. Moreover, an accurate policy of buffer management is used through the delegation of activities in agents that constitute the MAS.

Most data applications cannot predict their own need of bandwidth, and this is the reason why we need some services that allow active users in the network to share dynamically all the available bandwidth.

We know that in ATM technology the ABR and UBR classes of service are the standard proposal to support the data traffic.

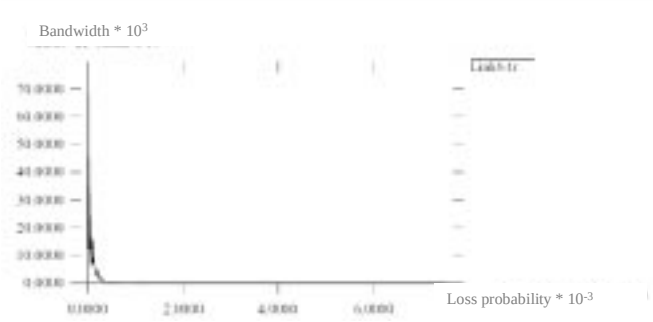

Fig. 3. Simulation of equation (4) with NS

Reference [3] presents the study of congestion in TCP networks over ATM and shows how the TCP throughput also falls when the discard of cells at ATM switches begins. The low throughput obtained is due to the waste of the bandwidth at congested links that transfer packets of corrupted cells; that is, packets with some dropped cells. Other research papers have demonstrated that TCP over UBR, with EPD suffers a considerable degradation of fairness, and also need a big buffer size, even if there are few connections. However, if ABR is used with schemes of explicit rate feedback, TCP offers better behavior of fairness and exploitation of links even with a buffer size less than at UBR.

The literature [9,10] also describes other ways to avoid the degradation of throughput at TCP sources over UBR. In order to do this, the discard of ATM cells is disconnected when there is congestion. So, the timeouts of TCP are avoided although they are the main cause of fall at TCP throughput, and also the periods of congestion are reduced, avoiding the big delay experienced with the fast retransmission algorithm of TCP before the source receives the duplicate ACKs.

There are two basic references in schemes of congestion control of TCP and ABR ATM: 1) The mechanisms of feedback of ABR with RM cells to control the transmission cell rate from the source (control rate), while the feedback mechanisms of TCP controls the size of a window as we have already seen (control credits). 2) The feedback mechanisms of ABR can be performed by intermediate switches, or by the receipt of traffic, while at TCP the feedback mechanisms are only realized by the receipt node with end-to-end ACK which provides reliability to the TCP.

At TCP sources, the CWND window controls the maximum traffic as we can see in the previous section. However, in the ABR class of ATM service, the traffic is controlled by several parameters such as MCR (Minimum Cell Rate), PCR (Peak Cell Rate) and ACR (Allowed Cell Rate). For TCP over ATM aspects, the mechanisms of traffic management used at TCP end nodes, at ATM end nodes and at network switches are also key elements which, in conjunction, provide a better goodput and reduce the delay caused by retransmissions. The delay in processing TCP packets is a random time period that represents the average delay of processing packets with a delay variation. This is applied to data packets, and to ACKs packets between sources and receivers.

With all these differences, and as ATM is a protocol placed under the TCP transport layer, solutions are required to solve the throughput problems due to the integration of these different technologies. These solutions propose changes at ATM switches inside the network; or the implementation of new extensions for TCP; or perhaps, specialized protocols for nodes placed at the limits of the ATM network and the TCP network. Our TAP protocol solves these problems by working inside the network, with

hardware (active ATM switches - AcTMs) and also software (multi-agent system MAS with TAP protocol) mechanisms. All this configures the whole TAP architecture.

IV. ADVANTAGES OF TAP

We propose EAAL-5, as an extension of AAL-5, specifically designed for data communications over ATM. At TCP over ATM, the datagrams are transferred to the data field (payload) of EAAL-5, as we can see in *Fig. 4* which shows the stack protocols of sources and receipts of TCP over ATM.

As we know, congestion control is a key aspect of ATM technology, and this is the reason why we have paid special attention to aspects such as: scalability, fairness, robustness and easy implementation.

The TAP architecture is active, because it provides active nodes at strategic points that implement an active protocol to allow the user's code to be loaded dynamically into network nodes at run-time. TAP also provides support for code propagation in the network thanks to the RM cells. TAP is also a distributed architecture in the sense that the protocol uses several active coordinated and self-collaborative agents.

In our previous work [11,12] we have presented the architecture based on TAP-MAS, constituted by software agents and equipped with a DMTE dynamic memory to solve the local retransmissions. The architecture of active AcTMs switches is presented in *Fig. 5*. We have also implemented the PQWFQ (PDU Queues PDU based on Weighted Fair Queuing) algorithm to apply fairness at sources. Also, the EPDR algorithm manages the buffer congestion and avoids PDU fragmentation.

The general motivation of this work is to find solutions to alleviate this negative problem of end-to-end retransmissions. Thus, TAP proposes to solve this problem locally when and where this appears to avoid the total cost of RTT time and to pay only the local *rtt* at the congested local link. The protocol does this when the links are at idle time. In other words, we use the idle time at links to solve the point-to-point retransmissions instead of the end-to-end ones.

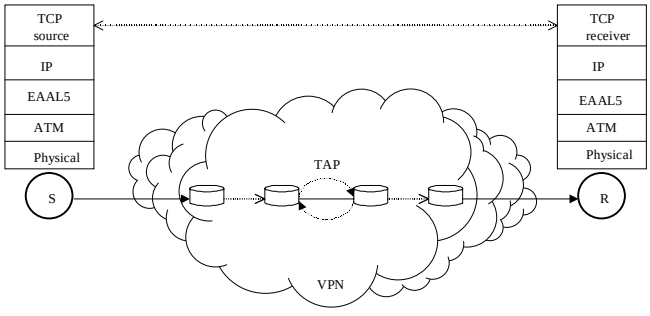

Fig. 4. TCP over ATM with TAP

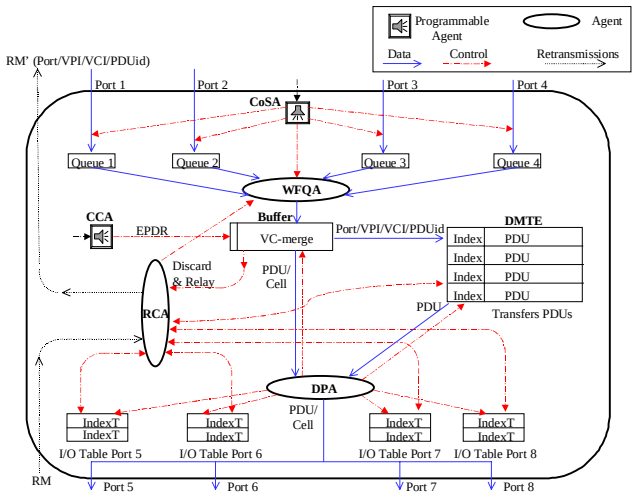

Fig. 5. TAP architecture with the TAP-MAS

Fig. 6 shows a more intuitive scenario, constituted by a network with 6 links, all with the same delay of d=10 ms. In an ideal network, the total delay D is 60 ms, so the RTT is 120 ms end-to-end. Our objective is to locate the TAP protocol between two links to reduce the negative effect of congestion at switch 3. If we suppose that switches 2 and 3 support the TAP architecture, when a packet congests switch 3, its retransmission is not 150 ms. but 20 ms, that is, the delay due to the local *rtt* with local congestion. If we want to consider the effect of *N* TCP sources sharing a link, this link can be a bottleneck determined by the size of the router queue that we could suppose to have the B value. The total size of the TCP queues will be *B*, and so *W=B/N*. If these values are substituted in equation (9), we can obtain a new approximate prediction of the packet loss probability:

$$P = 0{,}75\frac{N^2}{B^2} \tag{10}$$

The previous equation shows how high rates in the network will generate a high loss of packets. Also, the number of the sources sharing the link and the buffers of the routers affect the results. Moreover, the possibility of traffic management depends on the packet capacity of the buffer that works as congestion point. For *N* sources, the store capacity in the network will be the sum of the TCP sources sharing *RTT* multiplied by the bandwidth [7,6]. So, if TCP sources have the same *RTT*, the store capacity at the link is RTT times the bandwidth. In equation (10) the B value is the store capacity at the link, plus the buffers of the router queues. Small limits in the queue length mean in equation (10) that the B variable is converted into a constant, and causes the loss rate to grow when the number of TCP connections competing for the link are increased. As the loss rate grows, each TCP window will be reduced and each TCP source decreases its rate. So, the V rate of each source can be obtained from equations (4) and (8),

$$V = \frac{0{,}866}{RTT\sqrt{P}} \tag{11}$$

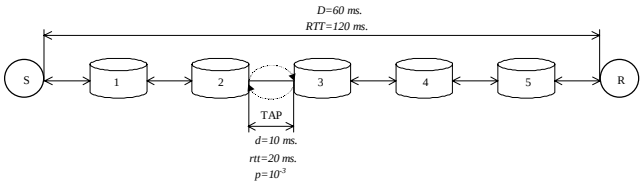

Fig. 6. Local retransmission at a congested switch

We should point out that the incorporation of the DMTE memory (see *Fig. 5*) in the TAP architecture contributes to an increase in the buffer size. Observing (10), DMTE contributes to decreasing the P loss probability which is inversely proportional to the buffer size of the queue routers. However, our proposal goes far beyond a simple aggregation of the memory. Therefore the most important gain we obtain is the possibility of local retrievals at the congested router (switch) with a lower value of RTT. So, TAP decreases the *P* loss probability and also the RTT. We can see how these two variables affect equation (11) where low values of RTT and *P* enable higher rates at sources of traffic.

Fig. 6 reminds us of equations (10) and (11) that express the relations between RTT, B (buffer size), P (loss probability) and V (source rate). In the ideal scenario without loss in the network, the DMTE memory of TAP offers a higher buffer size, which prevents the losses. On the other hand, if there is loss, TAP retrieves it at the point where this appears, reducing the end-to-end RTT to the point-to-point *rtt*.

V. EVALUATION OF TAP PERFORMANCE

Fig. 7 shows the effect of varying the CAR (Cell Arrival Rate) between 86 and 2,667 cells per second (33,000 Kbps to 1 Mbps respectively). In this simulation we fixed the congestion probability at 10^{-3}. We used an input buffer of 3,000 octets and the DMTE stored 2 PDUs of 1,500 bytes for each connection. If the value for CAR is 64 Kbps (167 cells/s.); Ton=0.96 s.; and Toff=1.69 s, over the 50 total PDUs discarded by congestion, 50 PDUs are retrieved via TAP. Also, when the CAR=56 Kbps and 33 Kbps, the TAP retrieves all the congested PDUs. Thus, the performance is optimized (50 retrieved PDUs out of 50 congested PDUs) since all the lost PDUs are retrieved and there are no DMTE failures (all the requested PDUs are in the DMTE).

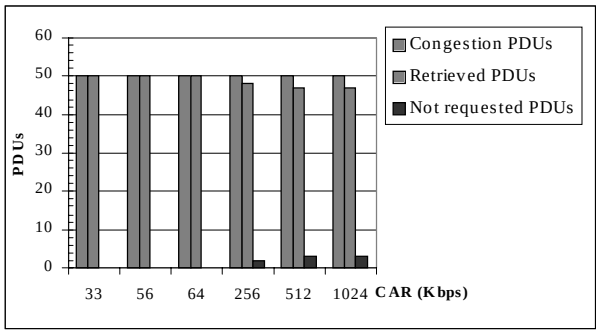

Fig. 7. Number of retrieved PDUs for different CARs.

As we can see, when the arrival rate is low, the number of retrieved PDUs increases. When the CAR increases, 256 Kbps, TAP retrieves 48 out of 50 PDUs, but the 2 lost PDUs are not requested because the protocol detects insufficient idle time (Toff) to do the retransmission. We can see how the number of NACKs (Not requested PDUs) not sent is greater when the PCR value increases. Thus the network is not over-charged with useless retransmissions when there is not a sufficient aggregate Toff.

We note that at a high CAR (1 Mbps) the number of retrieved PDUs is 47 and also the 3 not retrieved PDUs are not requested. As we can see, the goodput is optimized when the number of trusted sources do not exceed the service capacity.

To manage the buffer and the input queues at each AcTMs switch we have implemented the PQWFQ (PDU Queues based on Weighted Fair Queuing) algorithm as part of the WFQ agent at TAP-MAS. This algorithm achieves a fair treatment of the PDUs that arrive at AcTMs switches. We must treat the PDUs of connections with GoS (Guarantee of Service) as privileged traffic. PQWFQ uses a weighted mechanism to implement the priority required at privileged connections, and also supports a technique to manage the request of retransmissions when an input queue is congested.

Fig. 8 shows the statistics of the retransmissions sent over a scenario in which there were by three AcTMs switches equipped with our PQWFQ algorithm. This figure presents the retransmissions sent by each of the active switches. As we can see, switch A begins to serve the retransmissions requested from switch B when its buffer is congested (due to the requested retransmissions from switch C). When switch A starts to deal with the retransmissions, the fluctuation of the throughput at this switch also begins. In *Fig. 8* we can see how switch C does not serve any requested retransmission because this is the last switch. This simulation only studies the PQWFQ behavior isolated from other TAP mechanisms.

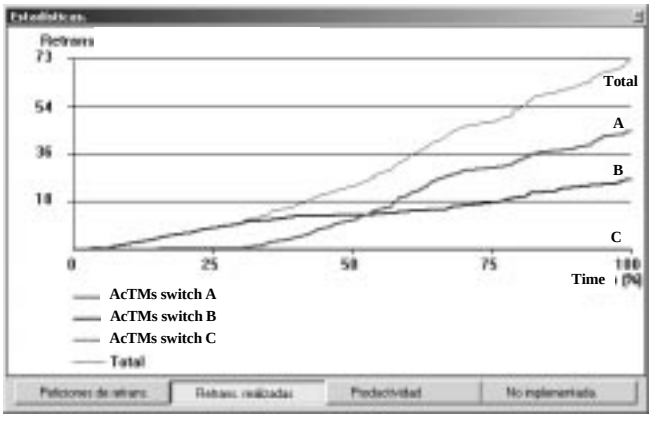

Fig. 8. Statistics of served retransmissions

These and another simulations demonstrate that the TAP architecture, which is active and distributed, takes advantage of the AcTMs switches. We have verified that it is possible to retrieve an important number of PDUs only with DMTE and a reasonable additional complexity of the AcTMs switches supported by software agents. Our simulations also demonstrate that the intuitive idea of taking advantage of silent states in ON/OFF sources is valid. Thus, we can achieve better goodput and QoS in ATM networks supporting the TCP traffic.

VI. CONCLUSIONS

In protocols of transport layer such as TCP over ATM, a packet is discarded by the network when one or several cells are lost, and the destination node requests the whole retransmission of the corrupted or lost packet. We have demonstrated through simulations the degradation experienced by the throughput of TCP. We have also studied how this falls logarithmically when the probability of loss of the ATM cells increases. The TAP protocol makes the retransmissions locally, and this decreases the loss probability and affects the throughput of TCP that alleviates the delays due to the end-to-end RTT. The TAP distributed architecture is constituted by a multiagent system with a protocol working over active AcTMs nodes with software agents.

REFERENCES

[1] Yoshihiro Ohba, "QLWFQ: A Queue Length Based Weighted Fair Queueing Algorithm in ATM Networks", *INFOCOM'97, Proceedings IEEE*, pp. 566-575 vol.2 (1997).

[2] W. Richard Stevens, "TCP/IP Illustrated, Volume 1," *Addison-Wesley Professional Computing Series*, (1994).

[3] Romanow, A. and Floyd, S., "Dynamics of TCP traffic over ATM networks," *IEEE Journal on Selected Areas in Communications*, pp. 633-641, (1995).

[4] UCB/LBL/VINT Network Simulator - ns, *http://www-mash.cs.berkeley.edu/ns/*

[5] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott, "The Macroscopic behavior of the TCP Congestion Avoidance Algorithm," *Computer Communications Review of ACM SIGCOMM*, vol 27, n. 3, (1997)

[6] Sally Floyd, "Connections with Multiple Congested Gateways in Packet-Switched Networks Part 1: One-way Traffic," *Computer Communications Review*, vol 21, n. 5 (1995).

[7] Robert Morris, "Scalable TCP Congestion Control," *Proceedings IEEE INFOCOM'2000*, pp. 1176-1183, (2000).

[8] K. Djemame, and M. Kara, "Proposals for a Coherent Approach to Cooperation between TCP and ATM Congestion Control Algorithms," *Proceedings UKPEW'99*, pp. 273-284, *http://www.cs.bris.ac.uk/Events/UKPEW1999/proceedings/* (1999).

[9] Hongqing Li, Kai-Yeung Siu, Hong-Yi Tzeng, Ikeda, C., and Suzuki, H., "A simulation study of TCP performance in ATM networks with ABR and UBR services," *Proceedings IEEE INFOCOM'96*, pp. 1269-1276, (1996).

[10] Shunsaku Nagata, Naotaka Morita, Hiromi Noguchi, and Kou Miyake, "An analysis of the impact of suspending cell discarding in TCP-over-ATM," *Proccedings IEEE INFOCOM'2000*, pp. 1147-1156, (2000).

[11] José Luis González-Sánchez and Jordi Domingo-Pascual, "TAP: Architecture for Trusted Transfers in ATM Networks with Active Switches," *ATM'2000 IEEE Conference on High Performance Switching and Routing. Joint IEEE ATM Workshop 2000 and 3rd International Conference on ATM (ICATM'2000 Heidelberg)*, pp. 105-112 (2000).

[12] José Luis González-Sánchez and Jordi Domingo-Pascual, "Trusted and Active Protocol over a Distributed Architecture in ATM Networks with Agents," *IEEE International Conference on Computer Communication and Networks (IEEE IC3N'2000, Las Vegas)*, pp. 484-490 (2000).

[1] This work is sponsored in part by the CICYT under Grant No. TEL99-1117-C03-03