Impact of cell discard strategies on TCP/IP in ATM UBR networks

Vincent Rosolen[*]*, Olivier Bonaventure*[+] *and Guy Leduc*[*]

[*]*Research Unit in Networking, Montefiore Institute of Electricity (University of Liège, Belgium).*
E-mail: {rosolen,leduc}@montefiore.ulg.ac.be.

[+]*Alcatel Alsthom Corporate Research Center (Antwerp, Belgium).*
E-mail: bonaveno@rc.bel.alcatel.be.

This work was supported by the Flemish Institute for the promotion of Scientific and Technological Research in the Industry (IWT) in the framework of the TCP project.

Abstract

Keywords
Simulation, performance, fairness, TCP/IP, UBR.

1 INTRODUCTION

2 CELL DISCARD STRATEGIES

Several techniques have been proposed to improve performance in ATM switches. Namely, dropping policies such as Tail Drop, Drop From Front [Lakshman 96] and variants with relation to the data unit format (*i.e.* cell or frame) were developed early to help switches deal with congestion. All these methods were partially successful, in the sense that they did achieve acceptable performance on congested links, but lacked fairness when throughputs were analyzed. Further investigation on this issue led researchers towards schemes aiming at both levels of performance.

2.1 Early Packet Discard (EPD)

EPD was first proposed in [Romanow 95], along with Partial Packet Discard (PPD), as a first notable improvement in cell discard policies. The idea behind EPD is that a discarded cell makes its corresponding packet incomplete, and therefore useless: cells from this packet continue to flow even though it will have to be retransmitted. This can lead to severe throughput degradation. To improve this situation, when the occupancy of an EPD switch reaches a fixed threshold τ, the switch drops entire packets. Thus, EPD completes end-to-end congestion control mechanisms in acting *inside* the congested network.

EPD has already been extensively discussed in the literature , as well as numerous variants and improvements. As a result, this method is now widely adopted in ATM switches.

2.2 Fair Buffer Allocation (FBA)

While EPD greatly improves throughput results, it does not attempt to improve fairness between the competing VCs. The reason is that the EPD scheme keeps track of *per-VC states* to implement the policy.

With *per-VC accounting* , one could also keep track of each VC by counting the number of cells from each VC in the buffer. This observation is the basis of Selective Packet Dropping (SPD) and allows better (fairer) allocation of the buffer resources to the active VCs.

In this scheme, an AAL5-frame from one connection is discarded if the buffer occupancy reaches a given threshold, and if this connection takes more than its fair share of the buffer resource. This simple algorithm is not yet quite satisfactory, since buffer occupancy can remain low. Indeed, a particular connection that has reached its share of the buffer will remain stuck to this allocation, even if the other active connections use little of their respective share. A better result can be achieved with the FBA algorithm [Heinanen 98], which proposes a smoother (but more complicated) scheme. Instead of rejecting cells as soon as the threshold is reached, the switch allows greedy connections to exceed their fair share, with respect to a rejection function such as illustrated in figure 1.

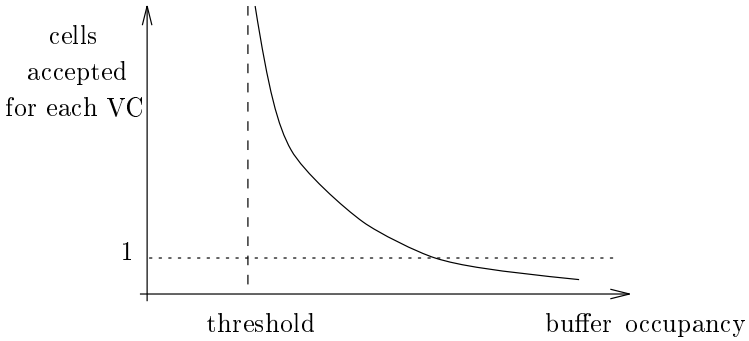

Figure 1 Rejection function in the FBA algorithm.

Each VC is thus allowed to use the buffer more efficiently, and cells from it are discarded if the rejection function goes below 1. The FBA algorithm also introduces a *scaling factor* to increase the flexibility of parameter tuning. The effect of this parameter (between 0 and 1) is to roughly translate the curve down as the scaling factor decreases to zero, enabling the algorithm to behave differently with relation to the total buffer size.

2.3 Random Early Detection (RED)

The RED scheme was first proposed in 1993 [Floyd 93] for IP gateways. Its objective was and still is to provide a fair bandwidth allocation, along with a simple implementation. The algorithm relies on an approximation of the *average* queue size in order to improve buffer utilization, and can be summarized as follows:

- The average queue size $\bar{Q}$ is estimated through an exponential weight w_q :

  $$\bar{Q}_{n+1} = (1 - w_q)\bar{Q}_n + w_q Q \tag{1}$$

  where Q is the *instantaneous* queue size. n refers to a time granularity which is mandatory for this sort of calculation. This formula can be seen as a low-pass filter through which the signal "instantaneous queue size" passes, yielding the output "average queue size". w_q is the time constant of the filter.
- If $\bar{Q}$ keeps under a fixed threshold min_{th}, then no discarding occurs.
- If $\bar{Q}$ goes over min_{th}, packet discarding must occur on each arriving packet with a probability p_a. This probability increases with $\bar{Q}$, and is a function of min_{th}, max_{th} and max_p, where max_p defines the "slope" of the first part of the dropping probability function. The specific function chosen for this paper is given in figure 2. A near-optimum function for probability p_a has still to be found [Floyd 97a].

More specifically, the RED algorithm gives an elegant answer to the *global synchronization* syndrome: the probabilistic approach allows switches to discard packets roughly proportionally to the connection's

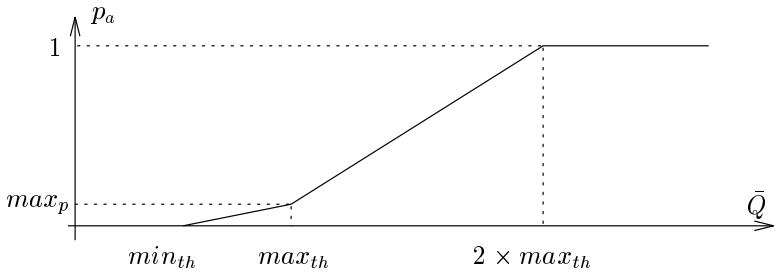

Figure 2 Dropping probability function in the RED algorithm.

share of the bandwidth through the gateway. This ensures that if multiple discards must be made, they will probably concern the greediest connections; thus, it is unlikely that all connections see one of their packets discarded, which avoids simultaneous beginnings of slow-start phases. This technique has proven very efficient and as a consequence, RED has been recommended for the default queue management mechanism in legacy routers [Braden 98].

(a) ATM-oriented implementation

While seeming an attractive strategy, RED needs core adaptation in order to fit with the ATM schemes. We describe in this section a specific adapted ATM-RED algorithm.

The algorithm is based on two boolean variables: *mark* (true if the decision to mark, or discard in our case, must be taken) and *EOM* (true if the arriving cell is an "end-of-message" cell). The operations can be represented by the state diagram on figure 3. On this figure, the two boolean variables have been labeled m (for *mark*) and e (for *EOM*) respectively. We take the convention that

- e is on if the arriving cell is an EOM cell, otherwise it is off ($\bar{e}$);
- m is on if the algorithm has calculated that discarding must occur, otherwise it is off ($\bar{m}$);
- the action taken by the switch is either accept an incoming cell (A) or discard it (D).

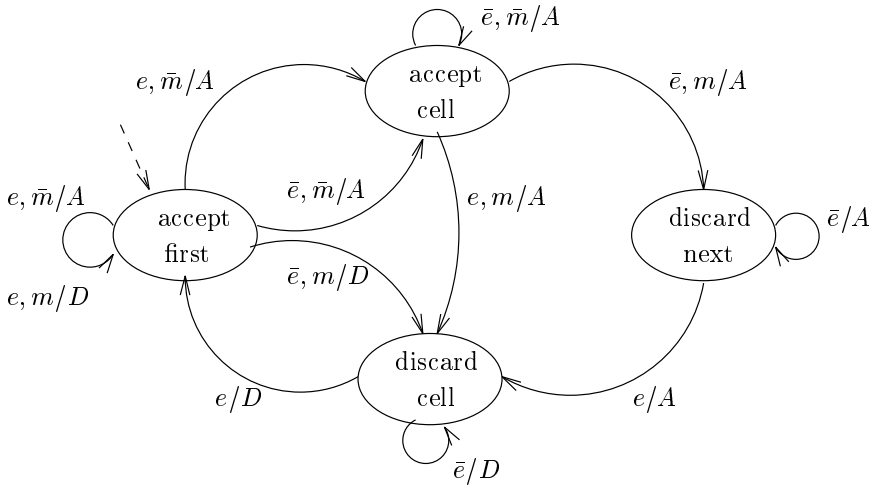

Figure 3 ATM-RED state diagram.

The probability that a packet must be marked (p_a in RED's original algorithm) becomes here a cell probability; it is thus evaluated on each arriving cell, but only if the state is either *accept-first* or *accept-cell*. RED's original parameter max_p has also to be adapted to a cell-oriented situation, and its value must thus be divided by the number of cell inside one MSS for our implementation. However, the "packet-

oriented" value will be referenced in the following sections to allow easier comparison with other RED related work.

The initial state in which the switch lies is *accept-first*, as represented on the figure with a dotted arrow. As long as a generated random number is above the aforementioned probability, the switch assumes that there is no congestion, and successively hops between states *accept-first* and *accept-cell* as long as new packets arrive for the particular VC. If the random number is under the probability, the bit *mark* is turned on, and the state changes according to bit *EOM*'s current value. Either the current cell (if *EOM* is on) or the next packet will then be discarded. The discarding of an entire packet can be visualized by the $\bar{e}/D$ transition in state *discard-cell*: the state cannot change until an EOM cell arrives. If, at that time, congestion is still present in the network, the RED algorithm will continue to mark cells, and it may be possible that an entire series of packets be discarded: the state machine will hop between states *accept-first* and *discard-cell*.

The main feature of this adaptation is that if a packet discard decision is made, it will concern the *next* packet (with relation to the cell that has been marked). Indeed, it is unlikely that whenever a cell gets marked, this cell is the first one of its corresponding packet. Thus, the discarding of an *entire* packet can only be achieved on the next packet, because the decision is always taken on an arriving cell basis.

Several drawbacks exist in this implementation. For example, bimodal-distributed connections can encounter undesired discards: if a cell belonging to a data packet is marked, the next (discarded) packet could be an ACK, which would result in suboptimal network behaviour. These issues could be dealt with in considering more than two bits for each VC state. Nevertheless, such a solution would bring additional implementation costs, and it is thus a matter of compromise which is beyond the scope of this paper.

It must be noted that a prior proposal [Elloumi 97] has been made to achieve either "cell-RED" or "packet-RED", but the former is unefficient, while the latter requires per-VC accouting and thus may meet implementation issues. Our algorithm does not require per-VC accounting, but *per-VC state*, which represents minimal housekeeping (only two bits are required, which keeps the implementation complexity similar to EPD).

3 SIMULATIONS

All our simulations were run on the *STCP simulator* (developed by Sam Manthorpe at the EPFL [Manthorpe 1996]), which includes the complete BSD 4.4 TCP/IP implementation. The source code has been thoroughly revisited in order to emulate the discard methods discussed here. For the sake of easier interpretations, the following assumptions are made in all types of environments:

- All sources are assumed to be identical with respect to their *equipment*. In other words, features such as interface cards, link delays and bandwidth, are unique for one type of environment.
- The sources are based on an on-off model, with a null off period. This type of source is more realistic than the infinite source model, with respect to common applications using TCP/IP. For instance, elements like TCP's slow-start algorithm [Jacobson 88] have a non-negligible impact on simulation results. All our simulations were run for an amount of time designed to have the sources successfully transmit a dozen files to reach steady state in the network.
- The queues that model the switches' buffers have a unique size, which is fixed at 8192 cells. The reason for this choice is that easier comparisons can be made between the three discard methods, regardless of buffer resources. The choice of 8192 reflects a good average of what is implemented in most of today's ATM switches.
- TCP's timeout values have been chosen in order to fit with modern TCP implementations, *i.e.* 200 ms for slow time-out granularity and 50 ms for fast time-out granularity.

3.1 Environments

ATM networks are intended to be widely developed, and are supposed to support a large number of applications in all types of environments. As a consequence, the more flexible a particular switch, the better behaviour it will exhibit. This holds particularly for the TCP/IP protocol suite, whose share of

global networking keeps growing [vBNS 97]. This flexibility is thus an important feature of the methods evaluated in this paper, and this is the reason why we consider two different environments: LAN and access network.

(a) Local environment

We model a LAN with a single bottleneck link between two switches, as shown in figure 4.

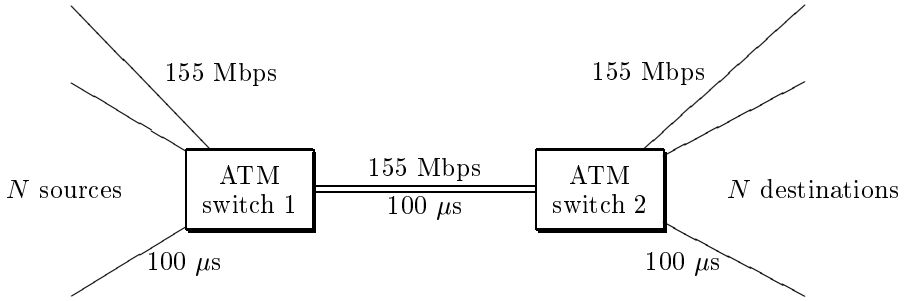

Figure 4 LAN simulation model.

This simple topology can be used as a basis for evaluation, without considering delay-related issues. Besides, this model (also called *N-source TCP configuration*, after its particularity to easily vary its number of sources) is often used in numerous previous analyses. The characteristics of this environment are:

- the bottleneck is shared by 10 sources, sending from left to right in figure 4;
- all links have a 155 Mbps capacity and a 100 μs delay;
- the transmitted files have a size of 5 Mbytes;
- the TCP sources use a 64 kbytes window, which is the standard value and is sufficient in this environment not to bring disturbing effects.

(b) Access network

The access network topology is basically identical to the LAN, as shown in figure 5.

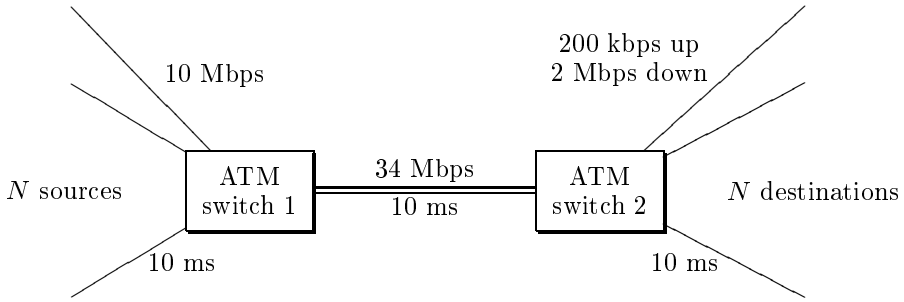

Figure 5 Access network simulation model.

The WAN backbone is again modelled by a single bottleneck between two border switches. The main differences concern delays and bandwidths. We have chosen to simulate an asymmetrical environment, close to the one encountered in xDSL access schemes. ATM is indeed currently deployed as a backbone technology for such wide area networks. We consider here characteristics that are typical of ADSL environments:

- the sources are assumed to be servers on legacy Ethernets, and are limited to a bandwidth of 10 Mbps each;
- the bottleneck link has a bandwidth of 34 Mbps;
- the destinations are assumed to be ADSL clients with access links of 200 kbps up and 2 Mbps down;
- all delays are 10 ms long;
- the transmitted files have a size of 512 kbytes;
- the TCP window is 128 ko to avoid issues related to the bandwidth-delay product in the network.

3.2 Parameters

In the following sections, we will refer to simulations that were run for a relatively broad spectrum of parameters specific to each method. For the sake of clarity as well as readability, only the best overall results will be presented, with the corresponding set of parameters. Additional observations will be explicitely made if a remarkable behaviour appears throughout several simulations.

Other parameters that can be taken into account relate to the source itself – namely, TCP options. Apart from the maximum window (which is fixed for a given environment, see section 3.1), we assume that delayed acknowledgements are active, as well as Fast Retransmit and Recovery [Jacobson 92].

Finally, we evaluate the impact of the MSS size in our simulations. We consider the two values 1460 and 9140 bytes, since 512 bytes becomes less frequent in most actual networks. We will see that the results sometimes vary dramatically when this parameter changes.

3.3 Evaluation criteria

The overall performance of a given discard method has several aspects. The most obvious one is to improve resource utilization in avoiding the retransmission of useless cells. Nevertheless, one must also take into account undesired effects which could result from the chosen scheme. Indeed, a certain discard method maintaining high throughput under poor utilization conditions would, for example, be useless. This type of situation can occur in a network whose switches do not implement any discard strategy: in this case, we would expect useless cells to pollute bandwidth, which would result in possible excellent throughput but very low utilization. The choice to investigate multiple evaluation criteria also completes the choice to simulate different environments: the best method is the one that behaves correctly, in the sense of the largest number of criteria, and under the broadest range of situations. These are the reason why we address here three performance issues: efficiency, throughput and fairness.

(a) Throughput and efficiency

The most important end-to-end parameter remains the throughput achieved by a connection. In all our simulations, we analyze the *goodput* of each TCP source; that is, the ratio of the bytes succesfully acknowledged and the simulation duration. This choice allows us to give interpretations at a TCP level, which guarantees a minimal relevance with pure applicative throughputs. Moreover, *ideal values* can be defined in taking into account protocol overheads, and can give a precise idea of a given strategy's overall behaviour. Considering the goodput reflects thus also the first criterion (utilization or efficiency): this presents the asset to show two features inside one single result. Note that considering TCP data for throughput *and* efficiency calculations reflects not only "how the pipe is filled with TCP data" but also "how *well* the pipe is filled with TCP data".

(b) Fairness

The macroscopic behaviour of the simulated environment can be visualized by means of a fairness index, which we define as follows:

$$\Phi = 1 - \frac{1}{N} \sum_{i=1}^{N} |\phi_i| \qquad (2)$$

where N is the number of sources and ϕ_i is the *normalized throughput difference* of source i in terms of the average throughput $\bar{t}$, *i.e.*

$$\phi_i = \frac{t_i - \bar{t}}{\bar{t}} \tag{3}$$

Thus, when all sources get the same fraction of the bandwidth, $\phi_i = 0$ for all i, and $\Phi = 1$.

4 RESULTS

In this section, we give general simulation results in relation with the criteria defined above. We present values in a *normalized* fashion; that is, the actual results have been divided by the corresponding ideal results, who are described at the beginning of each section.

In the tables, each cell with the form x/y refers to a result obtained with an MSS of either 1460 (for x)or 9140 bytes (for y), respectively. The entry named "UBR" refers to vanilla UBR, *i.e.* UBR with tail-drop policy.

4.1 LAN environment

Table 1 gives an overview of the simulations results for the LAN environment. As mentioned in section 3.3, we can estimate the ideal throughputs for this configuration. As a SONET equivalent overhead is applied to our ATM links, the ideal throughputs for an MSS of either 1460 or 9140 bytes are 12.9 and 13.5 Mbps respectively.

In these first results, not only can we note that the overall performance seems insensible with regard to the discard method, but also that this performance is indeed very close to the optimum. However, FBA's relative superiority in fairness can already be noticed here.

The main difference between the respective results does not appear in the table; it concerns the *buffer occupancy*. Indeed, we can verify that the average buffer size stays minimal with RED (about 2800/4000 cells), while it remains very high with other methods (typically 5500/6200 cells). This of course is due to RED's monitoring function, and this type of buffer behaviour may presume a network in good health, as recommended in [Braden 98].

Table 1 Results for the LAN environment

	Parameters	*Goodput*	*Fairness*
UBR		0.977 / 0.990	0.895 / 0.916
EPD	$\tau = 7373$ (90 %)	0.975 / 0.991	0.916 / 0.930
FBA	$K = 8000$, $R = 6000$, $Z = 0.9$	0.971 / 0.998	0.968 / 0.974
RED	$w_q = 0.002$, $max_p = 0.02$ $min_{th} = 2000$, $max_{th} = 6000$	0.980 / 0.974	0.964 / 0.941

A rather surprising fact is that the best RED results are obtained with sets of parameters that do not really fit with RED's philosophy: a max_{th} of 6000 (*i.e.* close to the buffer size) yields an "abnormal" dropping probability function (refer to figure 2). We think that this contradictory observation is the most obvious proof that *a discard method is of little importance in this type of environment*. Note that this holds for a given configuration; we will see that in even more restricting conditions (for example if we consider fifty sources instead of ten), the benefit brought by a discard strategy will be more obvious.

4.2 Access network

The results in table 2 are presented in the same fashion as for the LAN environment. Note that with this configuration, the ideal throughputs for MSS sizes of 1460 bytes and 9140 bytes become 584 kbps and 609 kbps respectively (fifty TCP sources sharing an E-3 type 34 Mbps bottleneck.

Table 2 Results for the access network

	Parameters	*Goodput*	*Fairness*
UBR		0.814 / 0.604	0.898 / 0.085
EPD	$\tau = 90\%$	0.932 / 0.858	0.926 / 0.748
FBA	$K = 8000$, $R = 4000$, $Z = 0.75$	0.930 / 0.778	0.973 / 0.847
RED	$w_q = 0.001$, $max_p = 0.1$ $min_{th} = 1000$, $max_{th} = 8000$	0.961 / 0.896	0.947 / 0.906

A typical behaviour that can be observed is that an MSS of 9140 bytes always yields inferior throughput. The reason of this issue comes from two facts :

1. delays become sufficiently significant to be a potential cause of trouble in congestion control ;
2. the Fast Retransmit and Recovery mechanism is assumed in all simulations.

These two characteristics mean the following : whenever congestion is experienced, the workstations try to recover quickly by using Fast Retransmit and Recovery. Therefore, we must expect this scheme to work less efficiently with larger packets, since more data is needed to trigger the algorithm. Moreover, the round-trip time is comparable to the timer granularity used in TCP, while it is much lesser in the LAN environment, which is why this feature is not observed in that environment.

For an MSS of 1460 bytes, FBA exhibits a clear superiority with respect to fairness, but at the price of lesser throughput. Finally, it must be noted that EPD's behaviour remains very satisfactory regardless of the chosen criterion. On the other hand, the result regarding vanilla UBR with a 9140-byte MSS illustrates what can occur if no discard strategy is adopted. In this simulation, the very poor fairness is due to the fact that out of the 50 sources, 23 were unable to send an entire file successfully. The reason is that the main bottleneck (*i.e.* the left switch on figure 5) experienced a major congestion collapse, as shown in figure 6. As a consequence, it can be calculated that the 27 sources that could send files successfully had an aggregate mean throughput of 680 kbps, while the other 23 were left with only 2 kbps each.

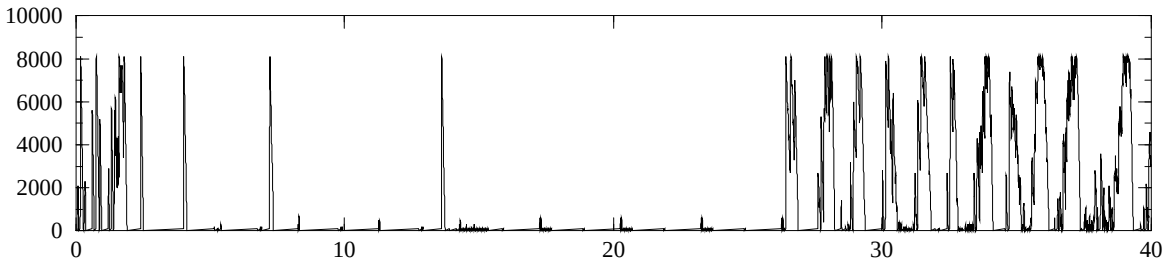

Figure 6 Congestion collapse with an MSS of 9140 bytes on vanilla UBR.

5 CONCLUSIONS

No "universal" strategy, *i.e.* no discard method and set of parameters that fits all situations : parameter tuning can prove tricky and environment dependent

ACKNOWLEDGMENTS

Jordi Nelissen
Jean-Charles Henrion
Kristoffer Rose and XY-pic

REFERENCES

Braden B., Clark D., Crowcroft J., Davie B., Deering S., Estrin D., Floyd S., Jacobson V., Minshall G., Partridge C., Peterson L., Ramakrishnan K., Shenker S., Wroclawski J., Zhang L., *Recommendations on Queue Management and Congestion Avoidance in the Internet*, Internet Draft, Feb. 1998.

Elloumi O., Afifi H., *RED Algorithm in ATM Networks*, Proc. IEEE ATM'97 workshop, May 1997.

Fang C., Lin A., *TCP Performance in ATM Networks : ABR Parameter Tuning and ABR/UBR Comparisons*, Proc. of SICON'97, Apr. 1997.

Floyd S., Jacobson V., *Random Early Detection Gateways for Congestion Avoidance*, IEEE/ACM Transactions on Networking, 1, (4), p. 397-413, Aug. 1993.

Floyd S., *RED : Optimum functions for computing the drop probability*, email available at `http://www-nrg.ee.lbl.gov/floyd/REDfunc.txt`, Oct. 1997.

Floyd S., *RED : Discussions of setting parameters*, email available at `http://www-nrg.ee.lbl.gov/floyd/REDparameters.txt`, Nov. 1997.

Goyal R., Jain R., Kalyanaraman S., Fahmy S., *UBR+ : Improving Performance of TCP over ATM-UBR service*, Proc. ICC'97, June 97.

Heinanen J., Kilkki K., *A fair buffer allocation scheme*, Computer Communications, V.21, p. 220-226, Jan. 1998.

Jacobson V., *Congestion Avoidance and Control*, Proc. of SIGCOMM'88, Aug. 1988.

Jacobson V., Braden R., Borman D., *TCP Extensions for High Performance*, Internet RFC 1323, May 1992.

Kalampoukas L., Varma A., *Performance of TCP over Multi-Hop ATM Networks : A Comparative Study of ATM-Layer Congestion Control Schemes*, Proc. of ICC'95, June 1995.

Kawahara K., Kitajima K., Takine T., Oie Y., *Packet Loss Performance of Selective Cell Discard Schemes in ATM switches*, IEEE JSAC, June 1997.

Lakshman T., Neidhardt A., Ott T., *The Drop From Front Strategy in TCP over ATM and its interworking with other control features*, Proc. of Infocom'96, March 1996.

Manthorpe S., *STCP 3.2.6*, see website `http://lrcwww.epfl.ch/~manthorp/stcp/`.

Mathis M., Madhavi J., Floyd S., Romanow A., *TCP Selective Acknowledgments Options*, Internet RFC 2018, Oct. 1996.

Romanow A., Floyd S., *Dynamics of TCP Traffic over ATM networks*, IEEE Journal on Selected Areas in Communications, 13, (4), p. 633-641, May 1995.

vBNS : for interesting statistics, see for example `http://www.vbns.net/`.