Configuration of DiffServ routers for high-speed links

Albert Banchs*, Sandra Tartarelli*, Fabrizio Orlandi*
Shohei Sato, Kazutomo Kobayashi**, Huanxu Pan****
***NEC Europe Ltd.**
****NEC Corporation**

Abstract The Internet is now widely expected to become an important communication infrastructure of the society, and therefore it is no longer sufficient to simply be able to provide connections. A higher quality of service (QoS) in communications is increasingly being required. As a new framework for providing QoS services, DiffServ is undergoing a speedy standardization process at the IETF. DiffServ not only can offer tiered level of services, but can also provide guaranteed QoS to a certain extent. In order to provide this QoS, however, DiffServ must be properly configured; to determine this proper configuration, a deeper understanding of DiffServ and its interaction with the different traffic types (specially TCP) is required. However, while much work in the past has focused on understanding the behavior of DiffServ with low-speed links, much less work has been invested for high-speed links. In this paper, we take up the subject of configuring high-speed DiffServ routers. We reuse previous work of the authors on DiffServ configuration and run an exhaustive set of simulations with high-speed links. We observe substantial differences in the resulting behavior with respect to previous work for low-speed links.

1 Introduction

The current Internet is basically a "best effort" type of network. As the Internet becomes a communications infrastructure of the society, it is increasingly essential to minimize the impact on the society of any deterioration in the QoS, due to failure, traffic congestion and so on. As a promising mechanism for controlling QoS, DiffServ is undergoing a speedy standardization process at the IETF [1]-[3].

A considerable number of papers in the literature have been devoted to study the performance of DiffServ networks via simulation (see e.g. [4]-[7]). However, in most cases simulations are performed with low-speed links (i.e. with a link capacity ranging from hundreds of Kbps to few Mbps).

In [8] the authors proposed a practical way of operating a DiffServ network, along with a concrete method for configuring the corresponding control parameters, and evaluated the resultant traffic characteristics based on simulation results. From that work we learnt that TCP traffic aggregate behavior is substantially different for low-speed links (hundreds of Kbps to few Mbps) than for high-speed links (hundreds of Mbps).

In this paper we extend our work in [8] by thoroughly studying the configuration of DiffServ routers for high-speed links. The focus is on the Assured Forwarding Per-Hop Behavior (AF PHB), since this is the PHB most likely to be used together with TCP. In order to gain insight into the performance of TCP with high-speed links, we ran a large number of simulations. In this paper we summarize the results obtained, presenting a subset of the simulations we ran.

The rest of the paper is structured as follows. In Section 2, we summarize the configuration rules that we proposed in [8]. These rules constitute the basic configuration that we have used in all the simulations of this paper. Section 3 reports on the simulation results for UDP traffic. Then, simulation results with TCP traffic are given in Section 4. As expected, results with TCP traffic are much less predictable: we report a number of interesting lessons learnt. In Section 5 we study the behavior of TCP traffic when being mixed with UDP, and in Section 6 we address the effectiveness of WRED for TCP in high-speed links. The paper closes with a summary and conclusions in Section 7.

2 Configuration rules

The terms of the contract between an ISP (Internet Service Provider) and a customer are defined in an SLA (Service Level Agreement), that comprises a number of specifics, including traffic performance guarantees. In this paper we consider the following traffic guarantees:

- a minimum guaranteed throughput (the committed throughput CTH),
- no losses for conforming traffic,
- limited delay for conforming traffic.

We restrict to those network services that are most efficiently transported by an AF class. The latter is in fact the most challenging DiffServ traffic class in terms of configuration difficulty. In this section, we recall the rules we will use throughout the paper to configure AF queues. For a more complete discussion the reader should refer to [8].

The first general rule is to separate UDP and TCP flows in different SLAs, since unfairness among UDP and TCP flows sharing the same traffic class is unavoidable [6]. However, this separation might not be always feasible, therefore we consider also a third AF class, conveying both UDP and TCP flows.

A customer that is assigned a token rate CIR has all its traffic up to a rate equal to CIR marked as green by the token bucket, while traffic in excess is marked yellow. In case of congestion, yellow packets are more likely to be discarded as compared to green packets.

In [8] we proposed some configuration rules to determine values for the token bucket and the WRED parameters, that guarantee the performance requirements reported earlier in this section. The configuration guidelines we propose depend on the transport protocol, i.e. we give slightly different rules for "not responsive" UDP only traffic, for TCP only traffic and for the mixed UDP and TCP case.

For UDP traffic, the token rate is set equal to the committed throughput, i.e. CIR = CTH. Since UDP is not responsive to packet drops, early dropping packets should be avoided. Accordingly, we set the WRED thresholds as $Y_{min}=Y_{max}=Y$ and $G_{min}=G_{max}=G$. Moreover, the packet discarding is performed based on the instantaneous queue occupancy. Under these assumptions, the no-losses requirement is guaranteed by the following rule:

$$\sum_{i=1}^{N} CBS_i + Y = G \qquad (1)$$

where N is the total number of UDP-only SLAs. On the other hand the delay is bounded by:

$$\text{delay} \le d + \frac{Y}{C} \qquad (2)$$

where C is the server capacity and d is the token bucket depth in seconds, which is related to the token bucket parameters according to the equation $CBS_i = d * CIR_i$.

TCP adjusts its sending rate based on the congestion (drops) experienced in such a way that the resulting throughput can be as low as three quarters of the allocated bandwidth. Therefore we set the token rate CIR to CTH/0.75. The WRED mechanism has been proposed to improve the performance of TCP traffic, by early notifying to the source a situation of congestion. In this case the WRED uses an exponentially weighted moving average for the buffer occupancy and separate values for Y_{min} and Y_{max} (as well as for G_{min} and G_{max}). We set $Y_{max}=G_{min}$ and the probability correspondent to the maximum discarding thresholds equal to 0.1. Equations (1) and (2) have to be adjusted accordingly as follows:

$$\sum_{i=1}^{N} CBS_i + Y_{max} = G_{max} \qquad (3)$$

$$\text{delay} \le d + \frac{Y_{max}}{C} \qquad (4)$$

In the mixed TCP-UDP case, we can choose to configure the parameters either according to the UDP or to the TCP case, depending on the actual applications transported by the SLA. In this paper we considered the second option, since we are interested in guaranteeing a throughput.

In [8] we provide empirical guidelines to set the bucket depth d for both the UDP and TCP cases and for the maximum yellow threshold Y_{max}. In this paper we study the sensitiveness of the performance on these parameters.

3 UDP traffic

In order to evaluate the configuration rules given in the previous section we first ran some simulations with only UDP traffic. The fact that UDP is not responsive to packet drops makes the behavior easier to predict.

The scenario used for the UDP simulations is given in Figure 1. This scenario consists of a single bottleneck link of 480 Mbps. All simulations are performed with the ns-2 network simulator and are based on the scenario in Figure 1.

We ran a large number of different simulations, varying the load of the bottleneck and the discarding thresholds. In the following we report the results of one of the most aggressive scenarios, where the total committed rate is equal to the link capacity and the green and yellow thresholds are respectively 700 and 100 packets. We set d equal to 10 ms, in order to have a small value for delay bound. The SLAs and traffic description of this scenario are given by Table 1. We modeled UDP sources as ON/OFF flows, with ON periods following a Pareto distribution with a shape parameter of 1.3 and OFF periods exponentially distributed with average 50 ms. During the ON period, the source transmits at its peak rate. We distinguish 4 types of UDP sources with different average and peak rates:

- Type 1: average rate 3 Mbps, peak rate 10 Mbps.
- Type 2: average rate 3 Mbps, peak rate 5 Mbps.
- Type 3: average rate 2 Mbps, peak rate 5 Mbps.
- Type 4: average rate 2 Mbps, peak rate 3 Mbps.

The packet length is set to 1 KB for all packets.

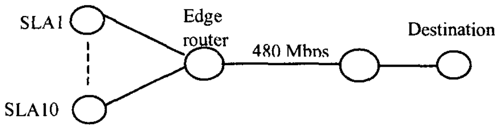

Figure 1 UDP–only simulation scenario

	CTH (Mbps)	CIR (Mbps)	CBS (Kbyte)	Flows Type 1	Flows Type 2	Flows Type 3	Flows Type 4
SLA1	100	100	125	10	10	10	10
SLA2	100	100	125	10	10	10	10
SLA3	50	50	62.5	5	5	5	5
SLA4	50	50	62.5	5	5	5	5
SLA5	50	50	62.5	5	5	5	5
SLA6	50	50	62.5	5	5	5	5
SLA7	50	50	62.5	5	5	5	5
SLA8	10	10	12.5	1	1	1	1
SLA9	10	10	12.5	1	1	1	1
SLA10	10	10	12.5	1	1	1	1

Table 1 UDP only simulations - SLAs definition

The results obtained for the above scenario are given in Figures 2 and 3. Figure 2 illustrates the normalized throughput for green and yellow packets (S(G) and S(Y) in the figure) and the normalized rate of yellow drops (S(Ydrops)), where the normalized throughput is defined as the throughput obtained divided by the SLA's committed throughput CTH. It can be seen that, since in average all SLAs are sending approximately at their CTH, there is no permanent congestion and all the SLAs get a throughput approximately equal to their sending rate (in all cases above the 90% of the agreed CTH). The important observation from that figure is that green packets are never dropped. However, when dealing with UDP traffic the most relevant performance measure is delay. Figure 3 illustrates the values obtained for average, standard deviation and maximum delay in ms. It can be seen that the design goals with respect to the delay are also achieved: average delays are very low, namely one order of magnitude shorter than the upper bound of Equation (2), expressed as MaxTh in the figure.

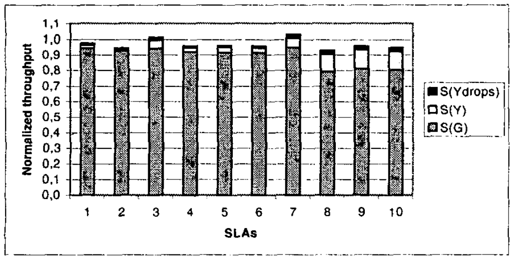

Figure 2 UDP simulations: Throughput.

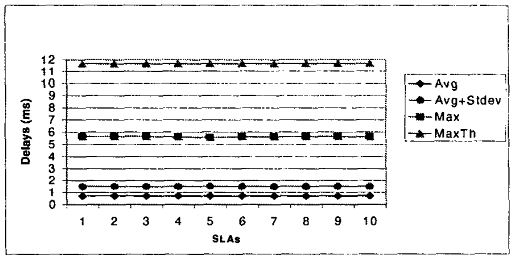

Figure 3 UDP simulations: Delay.

4 TCP traffic

To study the performance for SLAs delivering TCP only traffic, we considered again the topology depicted in Figure 1. Table 2 completes the description by defining the characteristics of the different SLAs. Note that RTT (round trip time) in Table 2 takes into account only the propagation delay, since the queuing delay is not known beforehand. We restricted our analysis to long lived TCP flows and we chose TCP Reno, with maximum window size equal to 64 KB (corresponding to 64 packets in our simulations).

	CTH (Mbps)	CIR (Mbps)	RTT (ms)	Number of flows
SLA1	100	133.33	100	40
SLA2	100	133.33	20	40
SLA3	50	66.67	20	40
SLA4	50	66.67	20	10
SLA5	10	13.33	100	40
SLA6	10	13.33	50	40
SLA7	10	13.33	20	40
SLA8	10	13.33	100	10
SLA9	10	13.33	50	10
SLA10	10	13.33	20	10

Table 2 TCP only simulations - SLAs definition

For the described scenario we ran nine different simulations, covering all combinations of $d \in (20,50,100)$ ms and $Y_{\max} \in (20,50,100)$ packets.

Figure 4 shows the corresponding results, by reporting the maximum and minimum normalized throughput achieved in the different simulations. (In Figure 4 Smax means the maximum throughput observed over the nine simulations and (10) indicates that the corresponding SLA transmits 10 flows. The other cases follow the same logic). The committed throughput is achieved in almost all cases, demonstrating that the configuration guidelines we suggest perform well for the given scenario.

Figure 4 highlights that a major role is played by the number of sources delivered by one customer (i.e. SLA). Indeed, SLAs with a higher number of sources

	CTH (Mbps)	CIR (Mbps)	CBS (MB)	Number TCP sources		Number of UDP sources			
				RTT 20 ms	RTT 50 ms	Type 1	Type 2	Type 3	Type 4
SLA1	100	100	0.625	0	40	10	10	10	10
SLA2	100	100	0.625	0	40	5	5	5	5
SLA3	50	50	0.3125	0	20	5	5	5	5
SLA4	50	50	0.3125	0	20	2	2	2	2
SLA5	50	50	0.3125	0	20	1	1	1	1
SLA6	30	30	0.1875	0	20	0	0	0	0
SLA7	30	30	0.1875	0	10	0	0	0	0
SLA8	30	30	0.1875	10	0	0	0	0	0
SLA9	20	20	0.125	0	0	4	4	4	4
SLA10	20	20	0.125	0	0	2	2	2	2

Table 3 TCP + UDP simulations - SLAs and traffic description

not only achieve the committed throughput, but are also favored in the sharing of the excess bandwidth. This result is confirmed also by Table 4, that points out the SLAs that experience the worst performance, when varying *d* and Y_{max}. The worst cases are in fact always SLAs delivering only 10 flows.

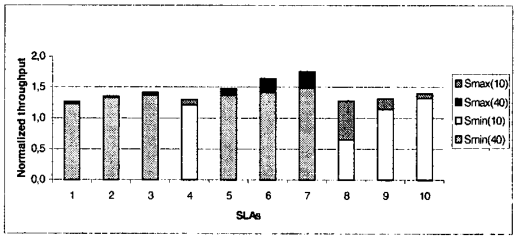

Figure 4 Throughput normalized to the CTH value.

The configuration guidelines we propose leave a certain freedom in the setting of the bucket temporal depth *d* and of the Y_{max} threshold. In Table 4 we show the impact of these two parameters on the observed performance. For each combination of the couple (d,Y_{max}) we selected the SLA that experiences the worst treatment and we report the normalized throughput (>1 means that all SLAs obtained the contracted CTH).

Observing both Table 4 and

Figure 4, we can assess that the role played by *d* and Y_{max} is minor as compared for instance to the number of flows transmitted by a customer. In general, the achievement of the minimum guaranteed throughput is not significantly influenced by these two parameters, while the latter seems to have a more relevant impact on the distribution of the excess bandwidth.

	Y_{max}=20	Y_{max}=50	Y_{max}=100
d=20	0.66 (SLA 8)	0.86 (SLA 8)	0.88 (SLA 8)
d=50	> 1	0.94 (SLA 8)	0.84 (SLA 8)
d=100	> 1	> 1	> 1

Table 4 Impact of Y_{max} and *d*

Table 4 supports the intuitive rule given in [8], where we suggest that *d* should not be very small, to avoid a too frequent yellow marking that would prevent the source from achieving the maximum window size. We recommend values for *d* in the range (50-100) ms.

The impact of the RTT was also object of our study. In all cases, we obtained that shorter RTTs lead to larger throughputs, which was the expected behavior (see e.g. [12]).

Another clear result that emerged from our analysis is that SLAs that require a smaller CTH are favored in both achieving the contracted rate and in the sharing of the excess bandwidth (see Figure 4).

5 TCP + UDP

In order to understand the behavior of TCP traffic when being mixed with UDP, we ran a simulation with the SLAs and traffic description of Table 3. UDP sources are as described in Section 3, Y_{max} is set to 100 packets, and *d* is equal to 50 ms.

The results obtained from simulating the above scenario are depicted in Figure 5. In this figure, we distinguish the traffic sent by each SLA normalized to the CTH depending on: 1) whether it has been dropped or not, and 2) whether it is TCP or UDP traffic.

The results obtained are reasonably good with respect to the throughput (non-dropped traffic) obtained by each SLA. We can observe that all SLAs achieve a throughput fairly close to their CTH, even though, as we already expected, SLAs sending no or little UDP traffic are in a less favorable position because of the low level of aggressiveness of their sources.

The drawback of mixing TCP and UDP, however, is not the distribution of throughput between the different SLAs but the distribution of the throughput used by an SLA among its flows. If we look at how throughput of SLAs 1 and 3 is distributed among TCP and UDP, we can observe a high level of intra-SLA unfairness, with TCP flows almost starving. We conclude that TCP and UDP should better be separated into different AF classes, as we already mentioned in Section 2.

Figure 6 shows the average, standard deviation and maximum delay experienced by green packets. We can observe that these values are below the maximum value allowed by Equation (2), which is represented in the figure as MaxTh.

Another interesting observation from the results of Figure 5 is the effect of increasing the RTT in this mixed scenario. We can observe that TCP traffic with short RTT (SLA8) suffers a worse performance than with a larger RTT (SLA7). This result might be surprising, since typically the TCP throughput is inversely proportional to the RTT. However, a short RTT has the drawback that the source is quickly notified the presence of congestion and there is the danger of synchronization among the different flows. This is in fact what happens in this case for SLA8, as confirmed by Figures 7 and 8. This result contrasts with the results obtained for TCP only traffic (Section 4.3). The reason is that UDP sources do not react upon yellow drops and keep sending traffic, which leads to a queue occupation inevitably close to the threshold Y_{max}. With this queue occupation, the probability that a TCP packet marked as yellow is dropped is much higher, which leads to the resulting synchronization. This observation also suggests the convenience of separating TCP and UDP in different AF classes in order to avoid TCP synchronization.

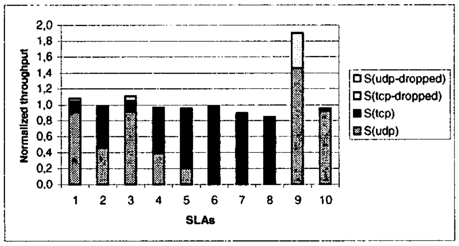

Figure 5 TCP+UDP: Throughput.

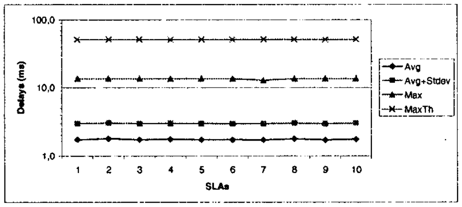

Figure 6 TCP+UDP: Delay for green packets.

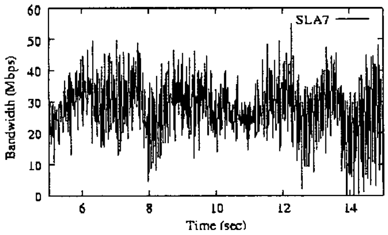

Figure 7 Instantaneous bandwidth for SLA 7.

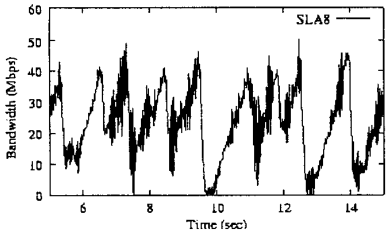

Figure 8 Instantaneous bandwidth for SLA 8.

6 WRED effectiveness

Active Queue Management (AQM) schemes are based on the idea of early dropping packets upon detecting upcoming congestion in order to let TCP flows react before congestion actually occurs. AQM was designed in order to improve the behavior of TCP flows such that:

1. TCP flows do not synchronize, which results in a gain in the total link utilization.
2. The occupation of the queue is kept small, which leads to smaller queuing delays.

Recently, a number of research papers [9],[10] have questioned the utility of AQM schemes.

In this section we attempt to study the usefulness of AQM in high-speed DiffServ routers by repeating the simulations performed in the previous section but using Tail Drop for each color instead of WRED (i.e. we set Y_{min}=Y_{max}, G_{min}=G_{max} and use the instantaneous value of the queue instead of the average).

Throughput results are illustrated in Figure 9. We can observe that these results are almost identical to the ones of Figure 5. We conclude that in the case of high-speed links, AQM does not improve the link utilization. The reason for this is that, in high-speed links with a large number of flows, synchronization of TCP flows is unlikely to occur, and the link utilization is already almost 100%. So, in such links, there is little or no room for improvement when using AQM. We believe this result to be important because of the contrast with the results obtained for lower speed links [11].

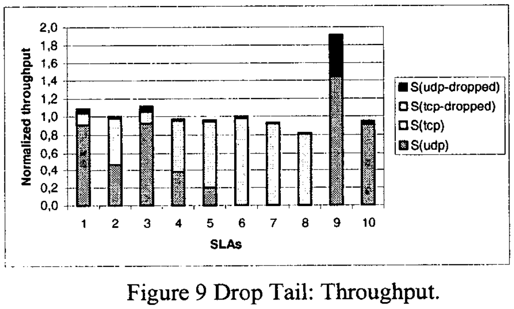

Figure 9 Drop Tail: Throughput.

Figure 10 illustrates the results obtained for delay with Drop Tail. We can observe that these delays are larger than with WRED. If TCP and UDP are to be used in the same AF class, this could be a reason to keep using WRED in high-speed links, since UDP performance benefits from low delay. However, as explained in the previous section, we recommend to use different AF classes for these two different traffic types.

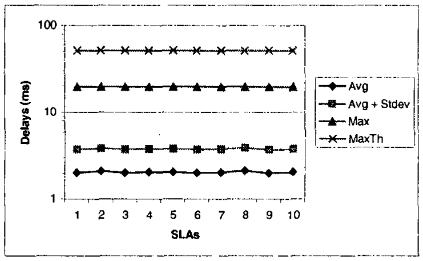

Figure 10 Drop Tail: Delay for green packets.

7 CONCLUSIONS

In this paper we have exhaustively evaluated via simulation the configuration rules that we proposed in [8] for high-speed links. The conclusions extracted from this extensive evaluation are summarized in the following points:

- The rules proposed in [8] by us are effective in high-speed links, since they guarantee
  - no losses for green traffic and a bounded queuing delay in each router for the UDP case.
  - the committed throughput in the TCP case.
- The objective of guaranteeing a throughput close to the CTH is achieved reasonably well. However, throughput results are a bit worse than for low-speed links (in [8], with low-speed links, all SLAs achieved a larger throughput than the CTH. This contrasts with the results obtained in this paper for high-speed links).
- When sending only TCP traffic, SLAs with larger number of flows and smaller CTH are favored in the sharing of the excess bandwidth.
- A too small value of the token bucket *d* leads to too frequent yellow drops and a consequent bad behavior with respect to the throughput. On the other hand a too large value of *d* leads to unfairness due to different RTTs. We recommend an intermediate value for *d*, in the range (50-100) ms.
- Shorter RTTs should lead to a larger throughput. However, this does not always hold true: in certain circumstances, synchronization effects may punish SLAs with shorter RTT. In order to avoid such synchronization, we recommend to use a value for Y_{max} equal to 100 packets.
- Mixing UDP and TCP traffic in the same AF class leads to a high level of intra-SLA unfairness. We recommend to separate these traffic types into different SLAs and AF classes.
- The use of early dropping in high-speed links provides no improvement in throughput for TCP. Unless UDP and TCP are mixed in the same AF class (which we explicitly do not recommend), we see no reason for using early dropping in high-speed links.
- If nevertheless TCP and UDP are conveyed in the same AF class, early dropping has the advantage of leading to lower queuing delays, which is a benefit for UDP.

We conclude that the behavior observed for high-speed links differs substantially than the behavior for low-speed links. The above observations give an insight into this behavior and help to better configure high-speed DiffServ routers. In the future we plan to extend the work presented here to network topologies with more than one bottleneck link.

REFERENCES

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services", RFC 2475, Dec 1998.

[2] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, Jun 1999.

[3] V. Jacobson, K. Nichols, K. Poduri, "An Expedited Forwarding PHB", RFC 2598, Jun 1999

[4] D.D.Clark and W. Fang, "Explicit allocation of best-effort packet delivery service", IEEE/ACM Transaction on networking, Vol.6, No.4, pp.362-373, Aug. 1998

[5] J. Ibanez, K. Nichols, "Preliminary Simulation Evaluation of an Assured Service", Internet draft, Feb. 1999.

[6] N. Seddigh, B. Nandy and P. Pieda, "Study of TCP and UDP Interactions for the AF PHB", Internet Draft, <draft-nsbnpp-di serv-tcpudpaf-00.txt>, June 1999.

[7] S. Sahu, D. Towsley, J. Kurose, "A Quantitative Study of Differentiated Services for the Internet", Proc. of GLOBECOM '99, pp.1808-1817, Dec. 1999

[8] S. Sato, K. Kobayashi, H. Pan, S. Tartarelli, A. Banchs, *Configuration Rule and Performance Evaluation of DiffServ Parameters*, in Proc. 17th International Teletraffic Congress (ITC17), Salvador da Bahia, Brazil, December 2001.

[9] M. May, J. Bolot, C. Diot, B. Lyles, "Reasons not to deploy RED", in *Proc. of IWQoS'99*, London, March 1999.

[10] M. Christiansen, K. Jeffay, D. Ott, F. Donelson Smith, "Tuning RED for Web Traffic", in *Proc. ACM SIGCOMM 2000*, Stockholm, August 2000.

[11] S. Floyd, V. Jacobson, "Random Early Detection gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, August 1993.

[12] S. Floyd, "Connections with multiple congested gateways in packet-switched network Part 1: one way traffic", *Computer Communications Review*, vol. 21, no. 5, October 1991.