gTFRC, a TCP friendly QoS-aware Rate Control for DiffServ Assured Service

Emmanuel Lochin, Laurent Dairaine, Guillaume Jourjon

gTFRC, a TCP friendly QoS-aware Rate Control for DiffServ Assured Service

Emmanuel Lochin[1], Laurent Dairaine[2], and Guillaume Jourjon[1]

[1] National ICT Australia Ltd, Australia,
[2] ENSICA - LAAS/CNRS, France,
{emmanuel.lochin, guillaume.jourjon}@nicta.com.au
laurent.dairaine@ensica.fr

Abstract. This study addresses the end-to-end congestion control support over the DiffServ Assured Forwarding (AF) class. The resulting Assured Service (AS) provides a minimum level of throughput guarantee. In this context, this article describes a new end-to-end mechanism for continuous transfer based on TCP-Friendly Rate Control (TFRC). The proposed approach modifies TFRC to take into account the QoS negotiated. This mechanism, named gTFRC, is able to reach the minimum throughput guarantee whatever the flow's RTT and target rate. Simulation measurements and implementation over a real QoS testbed demonstrate the efficiency of this mechanism either in over-provisioned or exactly-provisioned network. In addition, we show that the gTFRC mechanism can be used in the same DiffServ/AF class with TCP or TFRC flows.

Key words: Transport, Congestion Control, DiffServ, Assured Forwarding.

1 Introduction

The increasing capabilities of high performance end systems and communication networks have greatly accelerated the development of distributed computing. Distributed applications were originally characterized by very basic communication requirements mainly related to full packet reliability and order. Today, multimedia applications are more and more used and require strong delay and bandwidth guarantees. The Assured Forwarding (AF) class of the IETF/DiffServ [1] provides a guaranteed minimal throughput that these applications can take advantage of. The service offers is called Assured Service (AS) and built on top of the AF Per Hop Behavior (PHB). The minimum assured throughput (also called target rate) is given according to a negotiated profile with the user. This service is particularly designed for elastic flows. These flows are generated by applications able to adapt their network usage to the available network resources (also called adaptive applications). This means that the application is able to increase the traffic to use the available network resource and can decrease it when a congestion occurs.

Most of the today's Internet applications are supposed to be adaptive and use TCP [2] as a mean to transport their data. TCP offers a reliable and in sequence end-to-end stream oriented data transfer service. Moreover, TCP implements flow and congestion control mechanisms in order to avoid receivers' buffers overflowing and network congestion. In terms of congestion control, TCP performs at sender side on the application traffic as a bandwidth limiter according to the underlying network conditions. Despite of a good TCP behaviour in terms of network resource usage and bandwidth sharing, TCP is not appropriate for many applications that integrate time and bandwidth constraints and do not require full reliability.

A classical alternative to the use of TCP is the User Datagram Protocol. UDP is a minimalist transport protocol which does not provide any packet reliability, order and flow congestion control. As a result, UDP needs the application to implement user level control in order to compete fairly with other TCP flows. The Datagram Congestion Control Protocol (DCCP) is a recently standardized protocol offering a congestion controlled non reliable transport service [3]. DCCP is suitable to applications currently using UDP. DCCP aims to provide a transport service that combines both UDP efficiency and lightness with TCP congestion control and network friendliness. To realize that, one of the congestion control implemented into DCCP is the TCP-Friendly Rate Control. TFRC [4] is a congestion control mechanism for unicast flows operating in a best-effort Internet environment. Based on the TCP throughput equation [5], it is designed to be reasonably fair when competing for bandwidth with TCP flow. It generates a flow with a much lower throughput variation over time than TCP. As a result, it is particularly suitable for multimedia applications such as video streaming or telephony over the Internet.

In the particular case of the DiffServ/AF class, a minimal bandwidth is provided (called in-profile traffic part), with the possibility to reach higher bandwidth (called out-profile traffic part) depending on the network congestion level. As state previously, multimedia applications are natural candidates to use this service class. Nevertheless, as for classical TCP flows over DiffServ/AF class, TFRC does not use the full potential of the offered service and produces unexpected results in terms of user requirements. As TFRC mechanism models the TCP AIMD congestion control algorithm, its behaviour remains similar in average to TCP over this class. The flow RTT drives the obtained long term throughput, the guaranteed bandwidth being not efficiently used by the application in case of long RTT.

In this paper, we focus on how TFRC mechanism behaves in the context of a DiffServ/AF class. We show by simulation the good properties of classical TFRC in terms of bandwidth smoothing and sharing when it is mixed with others TFRC or TCP flows. Nevertheless, even if a throughput guarantee is provided to the application by the underlying network, as for TCP, the throughput obtained by TFRC mainly depends on RTT and loss probability. Then, the application does not always get the negotiated guaranteed throughput. To cope with this problem, we propose a simple TFRC adaptation, namely gTFRC, allowing the

application to reach its target rate whatever the RTT and the target rate value of the application's flow. Results from a large simulation campaign are presented to highlight the improvements and to exhibit the efficiency of gTFRC in various situations. The simulation study is complemented by an evaluation of TFRC and gTFRC behaviour over a real DiffServ/AF testbed.

This article is structured as follow. The section 2 provides related work about the DiffServ/AF and congestion control mechanisms. Section 3 gives the problem statement and presents the gTFRC mechanism. Section 4 enumerates the hypothesis and the simulation characteristics. Sections 5 and 6 evaluate gTFRC. Section 7 details a real implementation of gTFRC. Finally section 8 gives a perspective of this work and provides a conclusion.

2 Related work

In order to better understand the problem tackled in this paper, as TFRC models the TCP congestion control, we first recall the previous studies about TCP over DiffServ/AF. Then, we will present a related work part concerning TFRC.

2.1 TCP over DiffServ/AF class

Many studies related to the performance of TCP flow over assured service have already been achieved. In [6], five factors have been studied (RTT, number of flows, target rate, packet size, non responsive flows) and their impact has been evaluated in providing a predictable service for TCP flows. In an over-provisioned network, the target rate associated to the in-profile traffic is achieved regardless of these five factors. However, these factors have a deep impact on the distribution of the out-profile excess bandwidth. In their paper [7], Park and Choi demonstrate the unfair allocation of out-profile TCP traffic and conclude that the smaller target rate aggregate (resp. larger target rate) occupies more (resp. less) bandwidth than its fair-share regardless to the subscription level. As the TCP protocol uses the AIMD control congestion algorithm which fairly share the bandwidth available, the only meaning to obtain a service differentiation with TCP protocol is to use DiffServ traffic conditioners such token bucket color marker (TCM) [8] or time sliding window color marker (TSWCM) [9]. The behaviour of these traffic conditioners have a great impact on the service level, in terms of bandwidth, obtained by TCP flows. Several others conditioners have been proposed to improve throughput insurance [10], [11], [12], [13], [14], [15], [16]. In all these articles, it is clearly shown that the key of the TCP throughput guarantee problem is the values 3-uple $(loss\_probability, RTT, target\_rate)$ of a TCP flow.

2.2 TFRC over DiffServ AF class

TFRC is an equation-based rate control mechanisms aiming at reproducing the behaviour of TCP congestion control. The TCP equation presented in [17] and used in TFRC is as follow (1):

$$X = \frac{s}{(RTT \cdot \sqrt{\frac{p \cdot 2}{3}} + RTO \cdot \sqrt{\frac{p \cdot 27}{8}} \cdot p \cdot (1 + 32 \cdot p^2))} \quad (1)$$

The use of an equation instead of the AIMD algorithm in order to estimate the sending rate produces smoother throughput variations. Furthermore the TFRC congestion control is based on a datagram based communication instead of the stream based TCP connection.

To the best of our knowledge, there is a only few studies of TFRC behaviour in a DiffServ network. In particular, the authors in [18] investigate AF-TFRC performances and give a service provisioning mechanism allowing an Internet Services Provider (ISP) to build a feasible DiffServ system. In this study, the problem of high RTT difference between long and short transfer are not tackled. Moreover, for experiments purposes (based on loss rate estimation) all simulations are carried during 1000 seconds. This duration and the invariant network condition allows a TFRC flow to converge easily to the target rate. As a result, the flows achieve an average throughput near the target rate.

3 *g*TFRC: a QoS-aware rate control

In the context of the use of a DiffServ/AF class providing a known guaranteed rate, the flow throughput breaks up into two parts:

1. a fixed part that corresponds to a minimum assured throughput. In the event of congestion in the network, the packets of this part are marked like inadequate to loss (colored green or marked in-profile);
2. an elastic part which corresponds to an opportunist flow of packets (colored red or marked out-profile). No guarantee is brought to these packets. They are conveyed by the network on the principle of "best-effort" (BE) and are dropped first if a congestion occurs.

We assume that the network is well-provisioned and that the whole in-profile traffic does not exceed the resource allocated to the AF class. In case of excess bandwidth in the network, the application can send more than its target rate, so the network should mark its excess traffic out-profile. If the network becomes congested, this out-profile traffic is predisposed to losses.

As highlighted in section 2, the only way to make use of this service differentiation with TCP protocol is to set a DiffServ traffic conditioner. Even if the knowledge of the guaranteed bandwidth could be provided to the transport level, the AIMD principle integrated into TCP do not use the instantaneous throughput as an input value for its congestion control. Only acknowledgements and timeout analysis allow TCP to act on the rate control. On the contrary, the TFRC mechanism makes use of the instantaneous throughput in conjunction with the flow RTT and loss event . These parameters are used in order to compute the controlled rate. The resulted smooth rate control is particularly adapted to streaming media applications that do not require absolute reliability.

Nevertheless, the optimal rate estimated by TFRC still can be under the target rate needed by the application and provided by the underlying DiffServ network. TCP would react in a same manner by halving its congestion window. As for TCP in the AF class, the TFRC flow is not aware that the loss is corresponding to an out-profile packet and that it should not decrease its emitted throughput below the target rate. For TCP, the solution was to design a new conditioner able to better mark the TCP flows that a simple token bucket or propose to add a new QoS congestion window as in [19] or [20].

In contrast to TCP, the usage of the TCP equation makes the mechanism able to directly use the actual TCP throughput in conjunction with the flow RTT and loss event. In the present study, the gTFRC congestion control mechanism is made aware of the target rate negotiated by the application with the DiffServ network. Thanks to this knowledge, the application's flow is sent in conformance with the negotiated QoS while staying TCP-friendly in its out-profile traffic part. This is achieved by computing the sending rate as the maximum between the TFRC rate estimation and the target rate as given in (2).

$$G = max(g, X) \tag{2}$$

Where: G is the transmit rate in bytes/s; g is the target rate in bytes/s and X is the transmit rate in bytes/s computed by the TCP throughput algorithm. The rest of the gTFRC mechanism follow entirely the TFRC specification specified in [4].

gTFRC requires the knowledge of the underlying bandwidth guarantee the DiffServ/AF network service provides to the session. We assume this information is made available to the mechanism at socket creation time, directly by the application. So, the target rate parameter can be set e.g., by the `setsockopt()` function. Without loss of generality, this parameter is supposed to be known by application after it has been previously negotiated in an end-to-end basis. This can be accomplished through a proper signalization protocol that a DiffServ architecture should provide [21]. The main concern of this approach deals with security. Indeed, if we give the possibility to the application to instantiate through a `setsockopt()` function the target rate negotiated, we can imagine that a misbehaving person could abuse of this functionality by giving an higher value to g. In this case, the misbehaving person sends an UDP-like traffic and increases its out-profile traffic. The edge router still marks in-profile the packets in respect with the negotiated profile and out-profile the excess part. As as result, in case of congestion, the dropping precedence set in the core router drop this excess traffic. The misbehaving person does not take advantage of the situation as the number of losses of its own flow increases as well. The in-profile traffic remains protected in the network and the out-profile traffic receives a kind of flooding attack. As the out-profile traffic is a best-effort traffic, the use of gTFRC does not violate the equilibrium of the DiffServ network.

4 Simulation study

gTFRC is evaluated over a DiffServ network using simulation. The ns-2 simulator [22] and the Nortel DiffServ implementation [23] is used to achieve this study. We drive simulation on the testbed illustrated in the figure 1 with the two following scenarios:

1. the network is exactly-provisioned (when there is no excess bandwidth for the out-profile traffic);
2. the network is over-provisioned (there is excess bandwidth).

A network is under-provisioned when the amount of in-profile traffic is higher than the resource allocated to the AF class. This case could occur if the QoS Service Provider or the Bandwidth Broker [24] of a DiffServ network send or receive false information. In a DiffServ context, if the gTFRC source emits below its target rate and if the gTFRC flow gets losses, it means that the in-profile traffic is no guaranteed anymore by the network. In this case: *how should react gTFRC ?*

Two approaches are possible, the first one is to pursue to emit at the guarantee g. This behaviour is legitimate since the service provider must ensure to the client the service he paid for. The second one is to react to this congestion. We can imagine to add a new threshold (θ) to gTFRC . This threshold can be applied as following: if the emission rate X returned by the receiver is θ times below the target rate g, the sender must emit to X. In the case where $X < g/\theta$, it means that a bunch of losses has occurred in the in-profile part and that the congestion could be due to a wrong setting. We believe that this problem should not be solved inside gTFRC itself and should remain under the responsibility of the service provider. That is the reason why in this paper we do not consider the case of under-provisioned network.

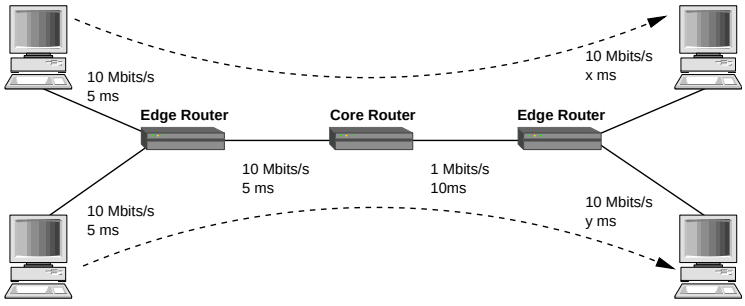


Fig. 1. The simulation topology.

An important known problem in a DiffServ network is the unfair bandwidth sharing of the out-profile part. Indeed, the out-profile part is a best-effort part

and is not necessarily taken into account by the network to achieve the negotiated target rate. In [7], the authors provide a way to solve this problem; they show that if a network is exactly-provisioned, there is no bias in favour of a flow or an aggregate that has a smaller target rate. Taking this as the starting point, they infer that the unfairness problem can be solved by making the network exactly-provisioned by simply adjusting the target rates of the token bucket marker in order to get a proportional differentiation. This assertion is strongly close to the RTT and the loss probability of the network. So, in a first part of our experiments, we measure the behaviour of TFRC and gTFRC in order to evaluate the gain brought by gTFRC in this network case. In a second part, we made measurements with an over-provisioned network. It is important to note that finding the best conditioning mechanism in order to improve throughput insurance remains out of the scope of this study. This problem should be tackled by the network provider. In our case, we propose an end-to-end approach to conform the resulted user-level traffic performance with the QoS negotiated in the network.

In all simulations:

- packet size is fixed to 1500 bytes;
- TCP version used is NewReno;
- we use a two color token bucket marker with a bucket size of 10^4 bytes in the edge router and the RIO[1] queue in the core;
- the queues size are 50 packets and RIO parameters are: $(min_{out}, max_{out}, p_{out}, min_{in}, max_{in}, p_{in}) = (10, 20, 0.1, 20, 40, 0.02)$
- the bottleneck between the core and the egress router is $1000Kbit/s$;
- measurements are carried during $300sec$.

In addition, we evaluate the throughput at the server side and report in the figures the instantaneous measured throughput.

5 Microscopic behaviour of TCP, TFRC and gTFRC into a DiffServ network

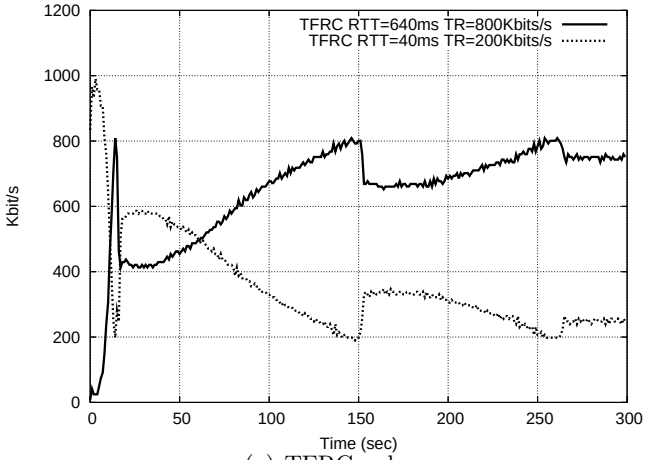
We perform experiments with many different RTTs and target rates configuration and give in this part representative measurements of the gTFRC efficiency. Other experiments are available in [25]. We present, in the next sections, experiments achieved in an exactly-provisioned network and in an over-provisioned network. As mentioned before, it means in the first case that all the traffic in-profile is allocated and in the second case, it means there is excess bandwidth in the network.

5.1 Experiments in an exactly-provisioned network

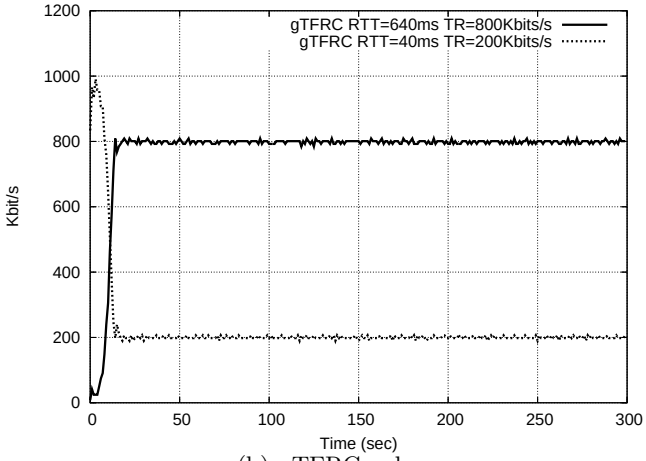
We measure the performance obtained by TFRC and gTFRC in the two scenarios. In figures 2, two flows are emitted on the testbed. The first one has a non

[1] RED in-profile out-profile

favourable conditions since it has the highest target rate to reach and a high RTT ($RTT = 640ms, TR = 800Kbit/s$). The second flow has the lowest target rate ($200Kbit/s$) and a low RTT ($40ms$). In a multi-domain DiffServ network, flows can cross one or several DiffServ domains and obtain low or high RTT. This fact motivates the large RTTs difference in our measurements.



(a) TFRC only



(b) gTFRC only

Fig. 2. Exactly-provisioned network

The results are presented respectively in figure 2 (a) for TFRC and in figure 2 (b) for gTFRC . We clearly see that gTFRC makes the traffic reach the target

rate much faster than TFRC. The reason is obvious since at the first rate decrease evaluation of the algorithm, gTFRC evaluates a rate equal to the target rate. In figure 2 (a), we can see that this case of decreasing occurs for TFRC at $t = 11sec$ and that gTFRC does not calculate a rate lower than the negotiated target rate in figure 2 (b). Figure 2 (b) shows that the flow with the lower target rate and the lower RTT is constrained to reach its own target rate of $200Kbit/s$.

Table 1 presents the packets statistics gathered during the simulation from figures 2. `ldrops` gives the number of packets dropped due to link overflow and `edrops` the number of packets dropped by RED queue for in or out-profile packets. As gTFRC increases the number of in-profile packets in the network, the number of `ldrops` out-profile packets increases too. Indeed, the dropping mechanism used on the core routers is the well-known RIO queue [26]. In order to decide whether to discard out-profile, respectively in-profile packets, RIO uses the average size of the total queue formed by in-profile and out-profile, respectively in-profile packets. Figure 3 illustrates this behaviour. For instance, the out-profile occupation rate is computed with $q\ min_{out}$ and $q\ max_{out}$ and the dropping probability represented by $p\ max_{out}$. As the in-profile traffic increases, the out-profile loss probability increases too.

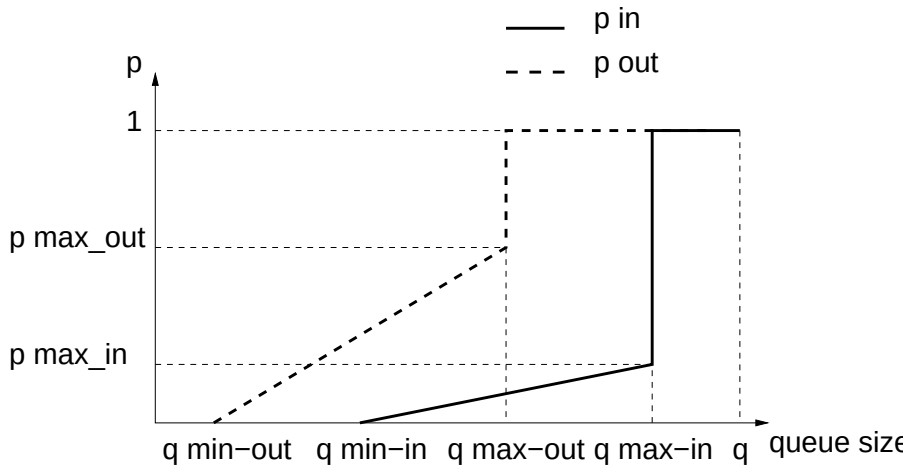


Fig. 3. Dropping probabilities of in-profile packets and out-profile packets in the RIO algorithm

However, gTFRC gives an average throughput for each flow near their target rate since the number of out-profile flow strongly decreases. As a result, it occurs less out-profile packets losses in the network and a better occupation of the in-profile traffic. These two measurements allow to conclude that gTFRC is DiffServ compliant and allow to get a throughput guarantee in the standard DiffServ AF class in case of exactly-provisioned network. Moreover, a simple token bucket

color marker is able to characterize gTFRC flows due to the non bursty nature of TFRC traffic[2].

Code Point	Total packets	ldrops	edrops
in-profile	83.5%	0.01%	0.02%
out-profile	16.5%	23.1%	49.0%

(a) Packets statistics with TFRC in figure 2 (a)

Code Point	Total packets	ldrops	edrops
in-profile	94.1%	0.01%	0.08%
out-profile	5.9%	53.8%	0.9%

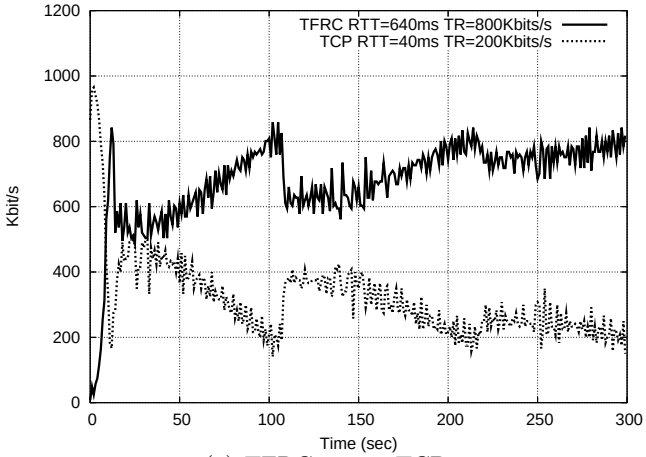
(b) Packets statistics with gTFRC in figure 2 (b)

Table 1. Corresponding packets statistics associated to figure 2

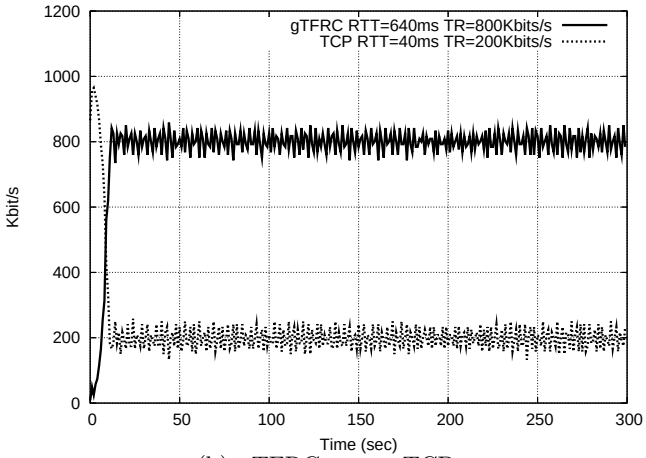
As the DiffServ/AF class has been designed to carry elastic flows and in particular TCP flows, we propose to compare gTFRC and TCP flows mixed together. TFRC has been built in order to be TCP-friendly. A flow is considered TCP-friendly or TCP-compatible when its long-term throughput does not exceed the throughput of a conformant TCP connection under the same conditions [28]. In case of a DiffServ network, this behaviour can change due to the network conditioning. Indeed, the aim of a DiffServ network is to perform a differentiation between flows and not to share in a fair manner the bandwidth. As gTFRC is a specific mechanism for DiffServ network, it is clearly not TCP-friendly in its in-profile. This TCP-friendly objective is not relevant in this profile as the user has paid for a fixed bandwidth guarantee. Nevertheless, the mechanism should remain TCP-friendly in its out-profile part, as several flow compete for this profile. We verify this point by keeping the same network scenario than in the previous part but we compare TFRC and gTFRC with a TCP flow. Results are presented in figure 4.

If we look at the previous experiments results reported figure 2, we can see that TCP and TFRC behaviours are similar and their long-terms throughput have the same order of magnitude. Indeed, both flows obtain respectively an average throughput equal to $354Kbit/s$ for TFRC in figure 2 (a) and $387Kbit/s$ for TCP in figure 4 (a). Table 2 presents the packets statistics obtained during the simulation in figure 4. These statistics are similar to these related on table 1. The oscillations of both figures 4 are due to the aggressive nature of TCP. In figure 4 (b), since TCP has the best condition to release it (lower target rate and lower RTT), the aggressive TCP behaviour tries to outperform its target rate and as a result, generates a high number of out-profile packets.

[2] On the contrary the token bucket is considered as a bad TCP traffic descriptor [6] [27]



(a) TFRC versus TCP



(b) gTFRC versus TCP

Fig. 4. Exactly-provisioned network

Finally, to conclude this comparison, we remind the already-known results obtained by two TCP flows in a DiffServ network in figure 5.

This figure illustrates that TCP does not reach its target rate even if the network is exactly-provisioned in case of high RTT difference. In figure 5 (a) the flow which requests a target rate of $800Kbit/s$ obtains a throughput around $200Kbit/s$. In [29] and [30], analytical studies show that it is not always possible to achieve service differentiation with a TCP token bucket conditioning in certain conditions. This example illustrates the case where the target rate parameter of the token bucket has no real impact on the negotiated target rate as highlighted

Code Point	Total packets	ldrops	edrops
in-profile	87.1%	0.03%	0.01%
out-profile	12.9%	3.0%	5.0%

(a) Packets statistics for scenario in figure 4 (a)

Code Point	Total packets	ldrops	edrops
in-profile	94.6%	0.03%	0.04%
out-profile	5.4%	56.3%	1.0%

(b) Packets statistics for scenario in figure 4 (b)

Table 2. Packets statistics for figure 4

in these studies. If we compare with figure 4, these results allow to verify that the use of gTFRC and TFRC flows in the same DiffServ class is not prejudicial for the TCP flows.

5.2 Experiments in an over-provisioned network

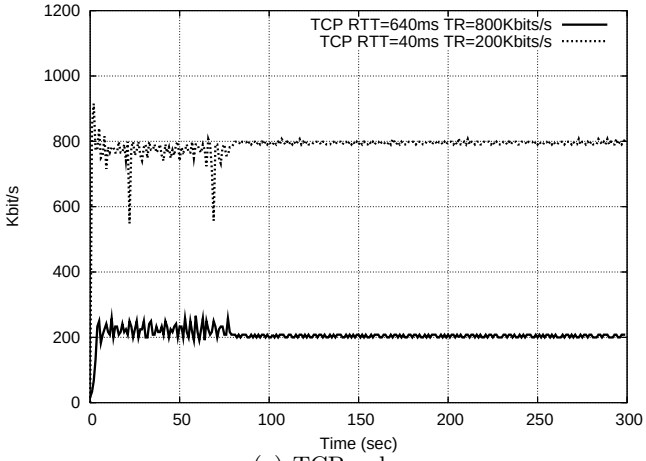
This section deals with scenario corresponding to an over-provisioned network. We present measurements where the network let 20% of unallocated bandwidth. We investigate the case of various RTTs and different targets rate.

These experiments achieve systematic comparisons between two target rates of $600Kbit/s$ and $200Kbit/s$ and two RTTs of $640ms$ and $40ms$. In figure 6, one flow is in the worst condition to reach its target rate. It has both the highest RTT and target rate. In figure 6 (a), the flow with the lowest RTT and target rate ($200Kbit/s$ and $40ms$) outperforms its target rate and the other flow ($600Kbit/s$ and $640ms$) never reach its target rate. So, as demonstrated in [29] and [30] for TCP flows, it seems that in certain conditions, the token bucket marker is not a good traffic descriptor for the TFRC flows too. On the other hand, figure 6 (b) shows that gTFRC enforces the requested target rate to be attained and that the gTFRC flow is able to reach it.

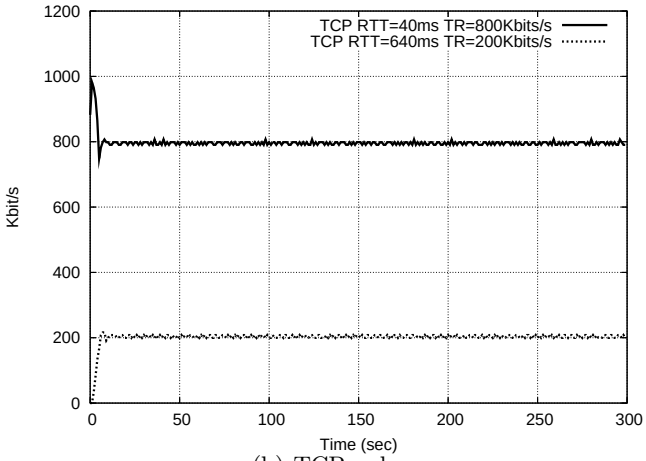
Table 3 gives packets statistics for this experiment and shows that gTFRC increases the number of in-profile packets in the network.

The next experiment aims at studying the mixing of a TCP flow with a TFRC or a gTFRC flow in an over-provisioned network. Figure 7 shows that the TCP flow is not disturbed by the gTFRC flow and remains the most aggressive in regard of its target rate. On both figures 7, TCP reaches a throughput about two times higher than its target rate.

Finally, in order to highlight the rapid convergence to the target rate of the gTFRC flows, another experiment is achieved in an exactly-provisioned network with ten flows. Each flow has an equal target rate of $100Kbit/s$ and a RTT ranging from $140ms$ to $1040ms$. Figure 8 presents the obtained results. Figure 8 (b) show that all flows reach the target rate before $t = 10$ seconds.



(a) TCP only

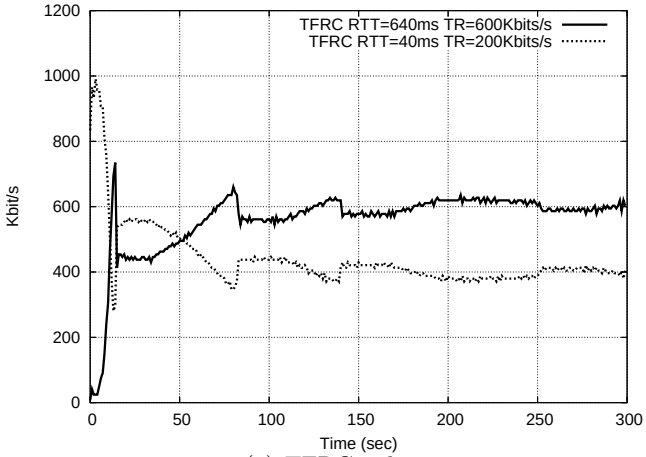


(b) TCP only

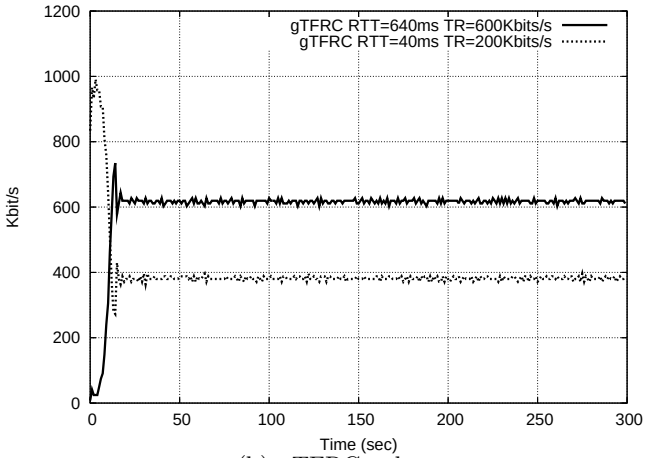
Fig. 5. Exactly-provisioned network with TCP

6 Macroscopic behaviour of TCP, TFRC and gTFRC into a DiffServ network

This section investigates the gain perceived by a user when using gTFRC flows in DiffServ/AF class. We look at the average throughput value obtained after a transfer of $100sec$ in function of the network load. First, in order to highlight the gTFRC robustness, we compare the throughput obtained by TCP, TFRC and gTFRC in a network with an increasing out-profile traffic. In this experiment, a single flow is emitted with a target rate corresponding to 60% of the bottleneck.



(a) TFRC only



(b) gTFRC only

Fig. 6. Over-provisioned network

In the same time, a UDP/CBR flow loads the network by emitting a traffic completely marked out-profile ranging from 0% to 80%. In figure 9, when the network load is equal to 100%, it means that the UDP/CBR source emits 40% of constant out-profile traffic. Each flow has an RTT equal to 200ms. Figure 9 gives the results obtained for either a TCP or TFRC or gTFRC flow against this UDP/CBR flow. This figure clearly shows the non-sensitivity of the gTFRC protocol to the increasing out-profile traffic load.

In order to supplement the experiment presented in figure 9, figure 10 presents a similar scenario where the UDP/CBR flow is replaced by a TCP aggregate

Code Point	Total packets	ldrops	edrops
in-profile	74.6%	0%	0%
out-profile	25.4%	0.5%	3.7%

(a) Packets statistics for scenario in figure 6 (a)

Code Point	Total packets	ldrops	edrops
in-profile	76.9%	0%	0%
out-profile	23.1%	0.6%	4.2%

(b) Packets statistics for scenario in figure 6 (b)

Table 3. Packets statistics for 6

ranging from 1 to 20 microflows with an $RTT = 80ms$. The increase of the aggregate size loads the network and decreases the throughput of the single flow which still have an $RTT = 200ms$. However, the gTFRC curve illustrates that this mechanism is non-sensitive to the increase of the aggregate size.

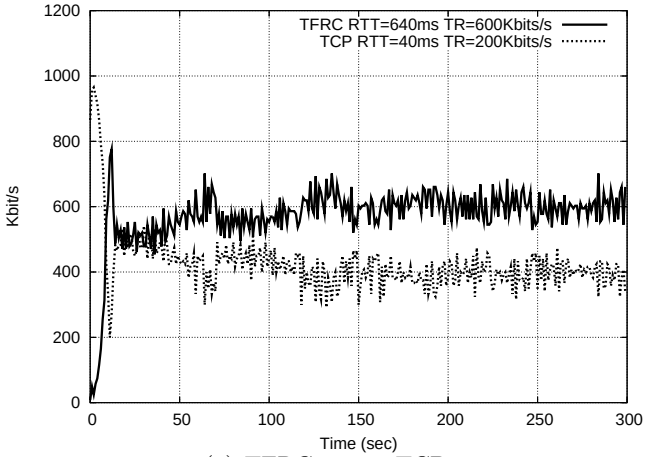
7 Implementation and integration of gTFRC in a multi-domain QoS Network

Many research works have been carried out on Quality of Service mechanisms for packet switching networks over the past ten years. The results of these efforts have still not been transformed into a large multi-domain network providing QoS guarantees [31].

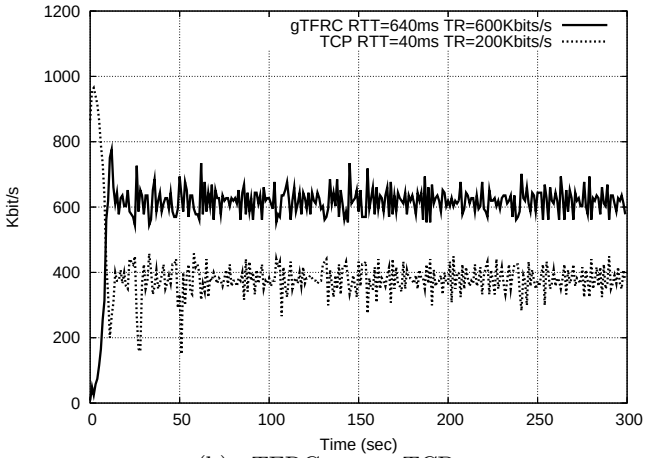
The EuQoS project[3] is an integrated project under the European Union's Framework Program 6 which aims at deploying a flexible and secure QoS assurance system over a pan-European testbed environment. The EuQoS System integrates many applications requiring QoS guarantees such as Voice over IP, Video on Demand or Medical applications over multi-domain heterogeneous environment such as WiFi, UMTS, xDSL or Ethernet technologies. Then, the EuQoS system integrates various architectural components such as signaling protocols, traffic engineering mechanisms, QoS routing, admission control to resource reservation scheme and of course multimedia application and transport protocols.

For this purpose, the EuQoS System integrates various architectural components such as signaling protocols, traffic engineering mechanisms, QoS routing, admission control to resource reservation scheme and tackles also the issue of QoS aware transport protocols. In this context, network configuration (i.e. resource allocation and reservation) is done according to the user's SLA[4] and applications requirements. This configuration is performed following a complex signaling process[5] which leads to the production of a QoS session descriptor.

[3] http://www.euqos.org/
[4] Service Level Agreement
[5] The details of this signaling process is out of scope of the present study.



(a) TFRC versus TCP

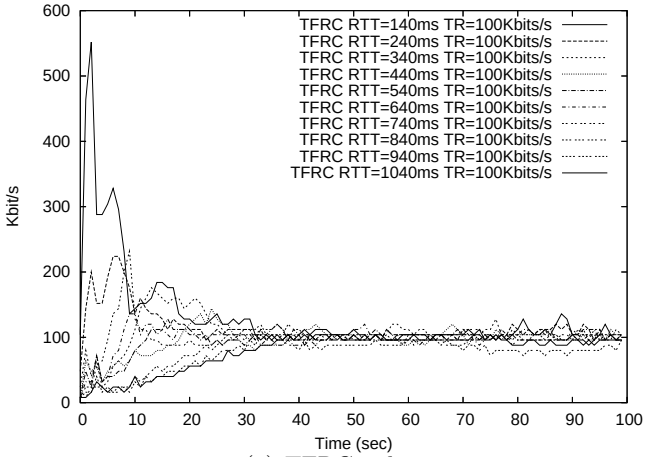


(b) gTFRC versus TCP

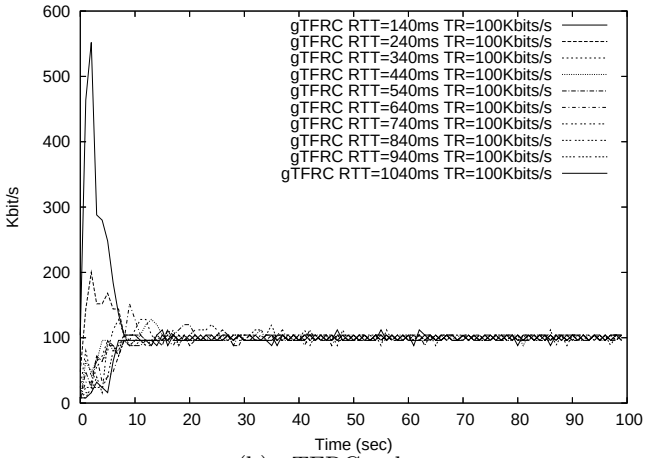
Fig. 7. Over-provisioned network with TCP

In the context of this project, both TFRC and gTFRC mechanism have been implemented as prototype in a Java transport framework. This TFRC mechanism has been enhanced, as described in the following, in order to take into account the QoS delivered by the underlying network.

Before deploying and measuring this protocol on the EuQoS network, a validation campaign has been achieved on both simulation for reference and with our implementation over a real network. We use the same topology given in figure 1. The real network testbed is composed by end-stations on GNU/Linux, and the edge and core routers run FreeBSD with ALTQ [32] to implement the



(a) TFRC only



(b) gTFRC only

Fig. 8. Ten TFRC or gTFRC flows with various RTTs in an exactly-provisioned network

token bucket marker and the RIO queue. Simulations parameters are similar to the ns-2 one given in 4 with a packet size fixed to 1000 bytes; the router queue size is 50 packets and measurements are carried during 180sec. For both experiments, we compute the average throughput at the server and at the receiver side. Figure 11 presents the results obtained with a similar scenario to this previously presented figure 2. As an example of result, the following figure illustrates the quasi-perfect correspondence between TFRC in simulated and executed mode. The complete results of this conformance testing are available in [33] and these

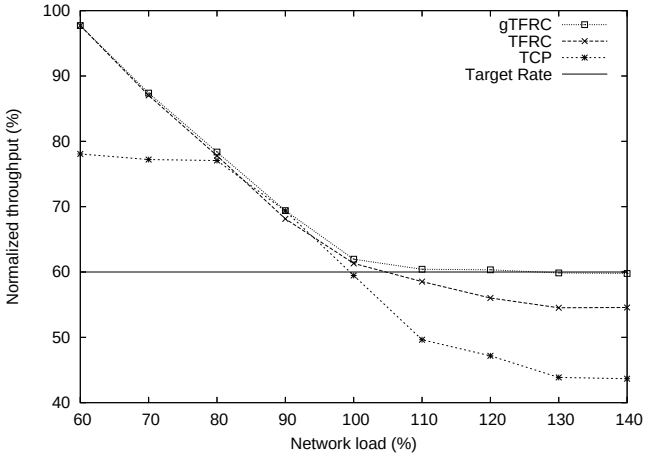


Fig. 9. UDP/CBR flow against TCP or TFRC or gTFRC with an RTT=200ms

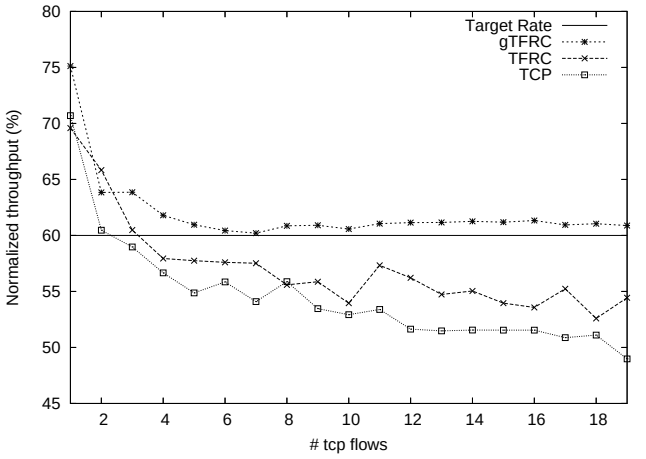
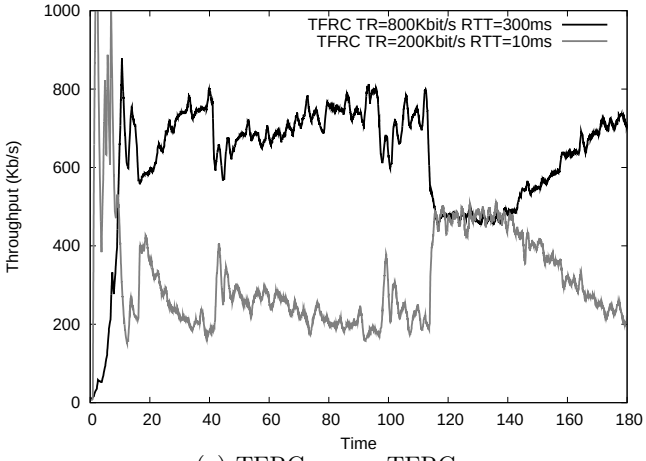


Fig. 10. TCP aggregate versus either TCP or TFRC or gTFRC flow RTT=80ms

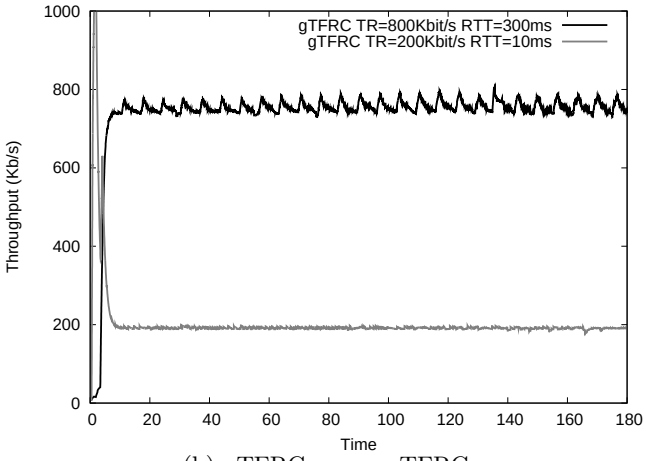
results show a very conformant behaviour of both TFRC and gTFRC compared to their reference.

8 Conclusions and further work

In this article we proposed a simple and efficient adaptation of TFRC congestion control for DiffServ network. gTFRC allows to reach a minimum guarantee throughput whatever the RTT or the target rate of a flow. It requires only the target rate negotiated by the application in order to become QoS aware. We



(a) TFRC versus TFRC



(b) gTFRC versus gTFRC

Fig. 11. Exactly-provisioned network

have demonstrated through many experiments its efficiency and that it can used in a standard AF/DiffServ class.

Future works are particularly devoted to the integration of gTFRC into the Enhanced Transport Protocol (ETP) [34], a configurable protocol offering a partially ordered, partially reliable, congestion controlled and timed controlled end-to-end communication service suited for message-oriented multimedia flows in the context of QoS network. ETP with gTFRC will be evaluated over the pan-European EuQoS network. Secondly, we are currently studying the possibility to add gTFRC as an extension into the CCID#3 of the DCCP protocol[35].

References

1. Heinanen, J., Baker, F., Weiss, W., Wroclawski, J.: Assured forwarding PHB group. Request For Comments 2597, IETF (1999)
2. Postel, J.: Transmission control protocol: Darpa internet program protocol specification. Request For Comments 793, IETF (1981)
3. Kohler, E., Handley, M., Floyd, S.: Datagram Congestion Control Protocol (DCCP). Request For Comments 4340, IETF (2006)
4. Handley, M., Floyd, S., Pahdye, J., Widmer, J.: TCP-Friendly Rate Control (TFRC): Protocol Specification. Request For Comments 3448, IETF (2003)
5. Padhye, J., Firoiu, V., Towsley, D., Kurose, J.: Modeling TCP throughput: A simple model and its empirical validation. In: Proc. of ACM SIGCOMM, Vancouver, CA (1998) 303–314
6. Seddigh, N., Nandy, B., Pieda, P.: Bandwidth Assurance Issues for TCP Flows in a Differentiated Services Network. In: Proc. of IEEE GLOBECOM, Rio De Janeiro, Brazil (1999) 6
7. Park, E.C., Choi, C.H.: Proportional Bandwidth Allocation in DiffServ Networks. In: Proc. of IEEE INFOCOM, Hong Kong (2004)
8. Heinanen, J., Guerin, R.: A Single Rate Three Color Marker. Request For Comments 2697, IETF (1999)
9. Fang, W., Seddigh, N., AL.: A Time Sliding Window Three Colour Marker. Request For Comments 2859, IETF (2000)
10. El-Gendy, M., Shin, K.: Assured Forwarding Fairness Using Equation-Based Packet Marking and Packet Separation. Computer Networks **41**(4) (2002) 435–450
11. Feroz, A., Rao, A., Kalyanaraman, S.: A TCP-Friendly Traffic Marker for IP Differentiated Services. In: Proc. of IEEE/IFIP International Workshop on Quality of Service - IWQoS. (2000)
12. Habib, A., Bhargava, B., Fahmy, S.: A Round Trip Time and Time-out Aware Traffic Conditioner for Differentiated Services Networks. In: Proc. of the IEEE International Conference on Communications - ICC, New-York, USA (2002)
13. Kumar, K., Ananda, A., Jacob, L.: A Memory based Approach for a TCP-Friendly Traffic Conditioner in DiffServ Networks. In: Proc. of the IEEE International Conference on Network Protocols - ICNP, Riverside, California, USA (2001)
14. Lochin, E., Anelli, P., Fdida, S.: AIMD Penalty Shaper to Enforce Assured Service for TCP Flows. In: Proc. of the International Conference on Networking - ICN, La Reunion, France (2005)
15. Lochin, E., Anelli, P., Fdida, S.: Penalty shaper to enforce assured service for TCP flows. In: IFIP Networking, Waterloo, Canada (2005)
16. Nandy, B., P.Pieda, Ethridge, J.: Intelligent Traffic Conditioners for Assured Forwarding based Differentiated Services Networks. In: IFIP High Performance Networking, Paris, France (2000)
17. Widmer, J.: Equation-Based Congestion Control. Diploma thesis, University of Mannheim, Germany (2000)
18. Kim, Y.G., Kuo, C.C.J.: TCP-Friendly Assured Forwarding (AF) Video Service in DiffServ Networks. In: IEEE International Symposium on Circuits and Systems (ISCAS), Bangkok, Thailand (2003)
19. Feng, W., Kandlur, D., Saha, D., Shin, K.S.: Adaptive Packet Marking for Providing Differentiated Services in the Internet. In: Proc. of the IEEE International Conference on Network Protocols - ICNP. (1998)

20. Singh, M., Pradhan, P., Francis, P.: MPAT: Aggregate TCP Congestion Management as a Building Block for Internet QoS. In: Proc. of the IEEE International Conference on Network Protocols - ICNP, Berlin, Germany (2004)
21. Hancock, R., Karagiannis, G., Loughney, J., den Bosch, S.V.: Next steps in signaling (nsis): Framework. Request For Comments 4080, IETF (2005)
22. http://www.isi.edu/nsnam/ns/.
23. Pieda, P., Ethridge, J., Baines, M., Shallwani, F.: A network simulator differentiated services implementation. Technical report, Open IP, Nortel Networks (2000)
24. Nichols, K., Jacobson, V., Zhang, L.: A two-bit differentiated services architecture for the internet. Request For Comments 2638, IETF (1999)
25. Lochin, E., Dairaine, L., Jourjon, G.: gTFRC: a QoS-aware congestion control algorithm. In: Proc. of the International Conference on Networking - ICN, Mauritius (2006)
26. Clark, D., Fang, W.: Explicit allocation of best effort packet delivery service. IEEE/ACM Transactions on Networking **6**(4) (1998) 362–373
27. Goyal, M., Durresi, A., Jain, R., Liu, C.: Effect of number of drop precedences in assured forwarding. In: Proc. of IEEE GLOBECOM. (1999) 188–193
28. Floyd, S., Fall, K.: Promoting the use of end-to-end congestion control in the Internet. IEEE/ACM Transactions on Networking **7**(4) (1999) 458–472
29. Sahu, S., Nain, P., Diot, C., Firoiu, V., Towsley, D.F.: On achievable service differentiation with token bucket marking for TCP. In: Measurement and Modeling of Computer Systems. (2000) 23–33
30. Malouch, N., Liu, Z.: Performance analysis of TCP with RIO routers. In: Proc. of IEEE GLOBECOM, Taipei, Taiwan (2002) 9
31. Cicconetti, C., Garcia-Osma, M., Masip, X., Sa Silva, J., Santoro, G., Stea, G., Taraskiuk, H.: Simulation model for end-to-end QoS across heterogeneous networks. In: 3rd International Workshop on Internet Performance, Simulation, Monitoring and Measurement (IPS-MoMe 2005), Warsaw (2005)
32. Cho, K.: Managing traffic with ALTQ. Proceedings of USENIX Annual Technical Conference: FREENIX Track (1999) 121–128
33. Jourjon, G., Lochin, E., Dairaine, L., Senac, P., Moors, T., Seneviratne, A.: Implementation and performance analysis of a QoS-aware TFRC mechanism. In: Proc. of IEEE ICON, Singapore (2006)
34. Exposito, E.: Specification and implementation of a QoS oriented Transport protocol for multimedia applications. Phd thesis, LAAS-CNRS/ENSICA (2003)
35. Floyd, S., Kohler, E., Padhye, J.: Profile for DCCP Congestion Control ID 3: TRFC Congestion Control. Request For Comments 4342, IETF (2006)