Implementation and performance analysis of a QoS-aware TFRC mechanism

Guillaume Jourjon, Emmanuel Lochin, Laurent Dairaine, Patrick Sénac, Tim Moors, Aruna Seneviratne

Implementation and performance analysis of a QoS-aware TFRC mechanism

Guillaume Jourjon[1,2] Emmanuel Lochin[1] Laurent Dairaine[2] Patrick Sénac[2] Tim Moors[1]
Aruna Seneviratne[1]
[1] National ICT Australia Ltd, Australia, [2] ENSICA - LAAS/CNRS, France,
{guillaume.jourjon, emmanuel.lochin, tim.moors, aruna.seneviratne}@nicta.com.au
{laurent.dairaine, patrick.senac}@ensica.fr

***Abstract*— This paper deals with the improvement of transport protocol behaviour over the DiffServ Assured Forwarding (AF) class. The Assured Service (AS) provides a minimum level of throughput guarantee that classical congestion control mechanisms, like window-based in TCP or equation-based in TCP-Friendly Rate Control (TFRC), are not able to use efficiently. In response, this paper proposes a performance analysis of a QoS aware congestion control mechanism, named *g*TFRC, which improves the delivery of continuous streams. The *g*TFRC (guaranteed TFRC) mechanism has been integrated into an Enhanced Transport Protocol (ETP) that allows protocol mechanisms to be dynamically managed and controlled. After conformance tests between ns-2 simulation and our implementation of the basic TFRC mechanism, we show that ETP/gTFRC extension is able to reach a minimum throughput guarantee whatever the flow's RTT and target rate (TR) and the network provisioning conditions[1].**

***Index Terms*— Transport, Congestion Control, TFRC, DiffServ, Assured Service.**

I. 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 that could be satisfied by a basic fully reliable and ordered transport service. Today, many applications are demanding more complex requirements, specially in terms of delay and bandwidth, which cannot be delivered without a network support such the one proposed by the IETF DiffServ architecture. In the DiffServ framework, the Assured Forwarding (AF) class of service provides a high delivery probability as long as the aggregated traffic of each site does not exceed its subscribed information rate [1]. Therefore, the AF class of service is of special interest for multimedia continuous flows such video streams which need a minimum guaranteed bandwidth and can support some losses and take advantage of excess bandwidth (with layered coding for instance). More generally, the AF service fits well with traffic generated by adaptive applications that can increase their throughput as long as there are available resources and can decrease it down to a minimum rate when congestion occurs. This minimum assured throughput (also called target rate) delivered by the AF service is given according to a negotiated profile with the user. Nevertheless, such QoS support alone is not sufficient to cope with either the full spectrum of application requirements (e.g., reliability, timing) or the network control requirements. Indeed, the transport layer is devoted to applying an efficient adaptation between the network services and the application requirements. However, TCP is used by the vast majority of applications. TCP is fully oblivious of the new application layer QoS requirements, and applies error and order control mechanisms that adversely affect continuous stream. Moreover, TCP applies a congestion control mechanisms that focuses on the network status while fully ignoring application layer QoS requirements. The TFRC mechanism has been introduced in order to reduce the disastrous impact of the potentially large rate variations entailed by the AIMD based TCP congestion control mechanism, while preserving the fair share of the available bandwidth. Although doing one step further for improving the service delivered to multimedia applications, TFRC is still oblivious both of the target rate needed by the application and the one offered in response by a network service such as AF.

This article focuses on the integration of a first level of QoS awareness (i.e. the target rate) in the TFRC mechanism. Our implementation of the basic TFRC mechanism has been integrated, tested and validated in a compositional transport protocol, named Enhanced Transport Protocol (ETP) [2]. We show that this implementation is compliant with the TFRC RFC [3] thanks to a cross-comparison between real measurements and the ns-2 reference implementation. Nevertheless, we show that when using TFRC in an AF network service, it has difficulties in reaching a guaranteed throughput. In order to solve this problem, thanks to the compositional transport architecture, we propose to make the mechanism QoS-aware following the proposal presented in [4]. This previous study based on ns-2 had shown the benefit of using *g*TFRC in a DiffServ Assured Forwarding (AF) class. In this paper, we validate and quantify the impact of this contribution from real network measurements. This paper is structured as follow. The section II presents the context of this study and provides some background about the compositional framework. Section III tackles a validation of the ETP/TFRC implementation. Section IV evaluates QoS aware TFRC implementation and finally section V gives some perspectives of this work.

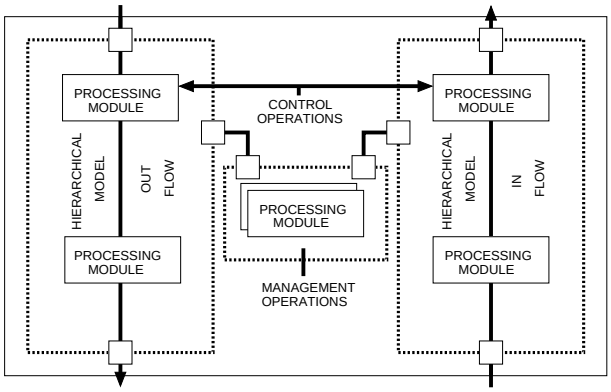
[1]This research work has been conducted in the framework of the EuQoS European project.

II. CONTEXT

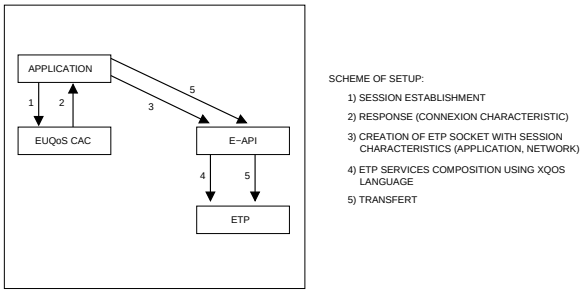
Many research works have been carried out on Quality of Service mechanisms for packet switching networks over the past ten years. The results of these efforts have still not lead to multi-domain network providing QoS guarantees [5]. Without loss of generality, this study takes place in the EuQoS project. The EuQoS project [6] is an integrated project under the European Union's Framework Program 6 which aims at deploying a flexible and secure QoS insurance system over a pan-European testbed environment. The EuQoS System aims at delivering QoS insurance to 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.

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 and applications' requirements. This configuration is performed following a complex signalling process[2] which leads to the production of a QoS session descriptor. This descriptor is implemented in an XML based language named xQoS [7]. XQoS session descriptors give all the details, related to the current session, that can be supported by the EuQoS service (with respect to the users' profile and the network status) including application level information (e.g., coding schemes, application data unit types, etc.) and the underlying network's QoS features. This document owns all the details regarding the current session including application level information (e.g., coding schemes, application data unit types, etc.) and the underlaying network's QoS characteristics. This session descriptor can be used to decide which transport level service, protocol and mechanisms offer the most efficient adaptation between the application needs and the offered network services. In this context, the Enhanced Transport Protocol (ETP) [2] has been introduced for offering a generic transport service and a dynamically configurable transport protocol. ETP is a connection oriented and messages oriented transport protocol. ETP offers among other things, a partially ordered, partially reliable, congestion controlled and timed controlled end-to-end communication service. ETP has been designed to be statically or dynamically configured according to the application layer QoS requirements. ETP services are implemented by the composition of configurable micro-mechanisms suited to control and manage the QoS needed by sessions' flows. The figure 1 (a) shows a high level view of the ETP architecture composed of control, management and provisioning processing modules that can be dynamically bound and configured. The processing modules represent micro-mechanisms, such as a rate control, reliability control, multiplexing, time control.

ETP framework has been initially modelled and evaluated in a best-effort network [2], [8]. However the context of the previously mentioned EuQoS system allows the transport protocol to be informed of the underlying network's QoS characteristics. In such context, the network service description can be provided to ETP through an Extended Application Programming Interface (E-API) for deciding which micro-mechanisms to compose in relation to the associated ETP session. This configuration scheme can be modelled as described in figure 1 (b). In this figure the EUQoS CAC[3] module represents the interface between the application and the negotiation of the EuQoS system. This module provides the XML configuration document as describe above.



(a) compositional architecture



(b) configuration in EuQoS

Fig. 1. Enhanced Transport Protocol

We have implemented the TFRC mechanism as a processing module in this compositional architecture. This TFRC mechanism has been enhanced, as described in the following, in order to take into account the QoS delivered by the underlying network. ETP uses an object oriented approach to instantiate dynamically micro mechanism. The Java language has been used for implementing ETP, because of its object oriented properties. The rest of this paper focuses on this mechanism and its behavior in the DiffServ/AF service. We show that the basic TFRC mechanism is not able to use efficiently the underlying level of service and then propose an extension which improves the QoS delivered to continuous flows.

[2]The details of this signaling process is out of scope of the present study.

[3]Call Admission Control

III. Non Functional Conformance of TFRC implementation

In this section we present a part of the validation measurements that have been achieved on both the ns-2.28 simulator (named in the following the reference TFRC implementation) and the ETP framework (named in the following ETP/TFRC) with an underlying network of which the behavior is emulated and controlled by the Dummynet tool [9]. We made several tests and give in this section an overview of this validation.

A. General hypothesis and model

In order to validate the TFRC implementation, we used the simple topology given in figure 2 for ns-2 simulator and the real testbed.

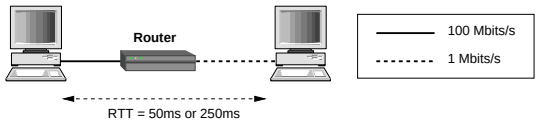
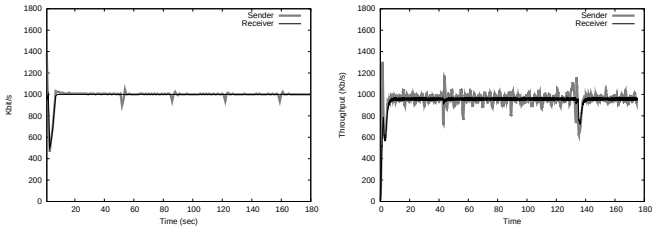


Fig. 2. The simulation topology for TFRC validation

The real testbed is composed by two end-stations on GNU/Linux, and one router with FreeBSD. We use a Dummynet pipe in order to emulate RTT and a packet loss rate (PLR). For both simulations: the packet size is fixed to 1000 bytes; the router queue size is 50 packets; measurements are carried during $180sec$. For each experiments, we compute the average throughput at the server and at the receiver side.

B. Network with constant bandwidth

In the scenario presented in figure 3, we show that in a network without any loss and with a constant bandwidth, ETP/TFRC implementation using the framework described in section II acts like the reference implementation. Figure 3 (a) shows the reference implementation results and figure 3 (b) the ETP/TFRC results. In this scenario, the bandwidth is fixed to $1000Kbits/s$ and the $RTT = 50ms$. No loss is introduced in the network.



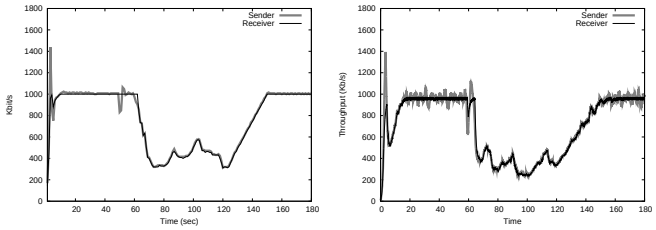
(a) TFRC ns-2 (b) TFRC real

Fig. 3. BW=1000Kbits/s RTT=50ms PLR=0

These figures show that at receiver side, the measured throughput is identical on both figures. The throughput oscillations on the sender side are more important on 3 (b) than on 3 (a). This slight difference can be explained by the different environments (i.e., simulation and real systems) and particularly in the real implementation host processing and the CPU load influences the packet treatment and as a result the delay in the network oscillates more. Nevertheless, the ETP/TFRC behavior remains strongly similar to ns-2 and the most important result is that on the receiver side, the same throughput is obtained.

C. Impact of losses and end-to-end delay

The aim of this experiment is to show that in case of high RTT ($250ms$) and with 1% of losses, the ETP/TFRC implementation reacts in a similar way than the reference implementation and that the convergence toward the available rate is identical after a loss period. In the figure 4, we show that ETP/TFRC implementation answers properly to the loss detection, during a large time period, of a specific packet loss rate. The readjustment to a normal sending rate is done in roughly the same amount of time (nearly $25sec$ in this particular RTT case).



(a) TFRC ns-2 (b) TFRC real

Fig. 4. BW=1000Kbits/s RTT=250ms PLR=1%

D. Impact of an UDP flow

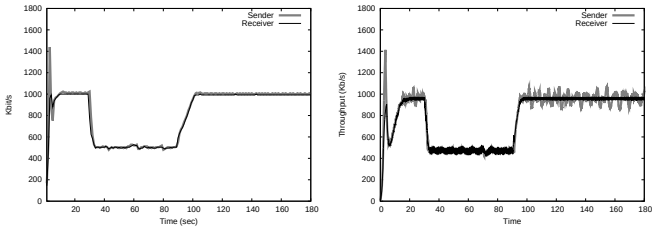
In this experiment, the bandwidth remains unchanged. There is no losses and the RTT equals $100ms$. An UDP flow with a rate equal to $500Kbits/s$ is emitted between $t = [30sec, 90sec]$. In figure 5, due to the packet multiplexing with non responsive UDP flow, both implementations brutally decrease during the UDP emission. Furthermore, ETP/TFRC implementation answers to the detection of losses due to the UDP flow in the same way than the reference implementation. When the UDP flow stops, the response of both implementations remains similar.

E. Conclusions

We made experiments with several others scenarios similar to those defined in [10] for the TFRC ns-2 validation. As this is not the purpose of our paper, all these experiments deliver very similar results and allow us to consider that our ETP/TFRC implementation complies to ns-2 implementation.

IV. Testbed measurements in a DiffServ network

This part deals with the use of ETP/TFRC implementation in the DiffServ/AF class. We present our adaptation of ETP/TFRC allowing the application to reach its target rate whatever the RTT value of the application's flow is.



(a) TFRC ns-2 (b) TFRC real

Fig. 5. PLR=0 BW=1000Kbits/s UDP fbw 500Kbits/s $t = 30sec, t = 90sec$

A. Problem statement

In the assured service class, the throughput of a flow breaks up into two parts: a fixed part which corresponds to a minimum assured throughput; packet belonging to this part are marked in-profile and an elastic part which corresponds to an opportunist flow of packets; packets of this part are marked out-profile.In the event of congestion in the network, the in-profile packets are considered inadequate for loss. At the contrary, out-profile packets are conveyed on the principle of "best-effort" (BE) and are dropped first if a congestion occurs. In this study, we assume that the network is well-provisioned and that the whole amount of in-profile packets belonging to all the traffics carried does not exceed the resource allocated to the AF class.

In case of excess bandwidth in the network, the application could send more than its target rate, so the network should mark its excess traffic out-of-profile. Then, if the network becomes congested, many out-of-profile losses occur and the optimal rate estimated by TFRC could fall down under the target rate requested by the application. TCP would react in the same manner by halving its congestion window. As for TCP in the AF class [11], the TFRC mechanism is not aware that the loss corresponds to out-profile packet and that it should not decrease its actual sending rate less than the target rate. For TCP, the solution was to introduce a conditioner able to better mark the TCP flows by taking into account the sporadic nature of the TCP flows [12], [13]. But the proposed conditioners are not all really efficient in certain network conditions such as long RTT and are sometimes complex to use.

In contrast to TCP, as TFRC explicitly computes the actual sending rate with the TCP throughput model given by (1).

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

Where the sending rate (X) depends on the packet lost rate (p), the mean packet size (s) and the Round Trip Time. RTO refers to the TCP retransmission timeout value. Thanks to this TCP equation, it is possible to directly act on this rate to avoid the under-usage of the network service. Therefore, the present proposal consists of making the sending rate estimator aware of the target rate. This scheme avoids the indirect processing of traffic conditioners while enhancing efficiently the performances in terms of application throughput and TCP-friendliness.

The target rate is supposed to be known by the transport layer by the way of the xQoS network service descriptor. During the session, the transmit rate is computed at sender side as the maximum between the TFRC rate estimation and the target rate, with the following equation (2):

$$G = max(g, X) \quad (2)$$

Where: G is the transmit rate in bytes/second, g is the target rate in bytes per second and X is the transmit rate in bytes/second computed by the TCP throughput equation specified in [3].The rest of the ETP/*g*TFRC [4] mechanism follow entirely the TFRC specification. Thanks to this adaptation, the application's flow is sent in conformance with the negotiated QoS while staying TCP-friendly in its out-profile part.

B. Model and general hypothesis

ETP/*g*TFRC performances have been evaluated over the DiffServ testbed presented in figure 6. The hosts are PCs on GNU/Linux and routers run FreeBSD with ALTQ [14] in order to implement the DiffServ network. The simulations have been carried out using the following configuration: the packet size is fixed to 1500 bytes; the two colors token bucket marker with a bucket size of 10^4 bytes is used on the edge router; routers are configured with a queues size of 50 packets and RIO parameters in the core router is corresponding to (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 $1000Kbits/s$; measurements are carried out during $180sec$.

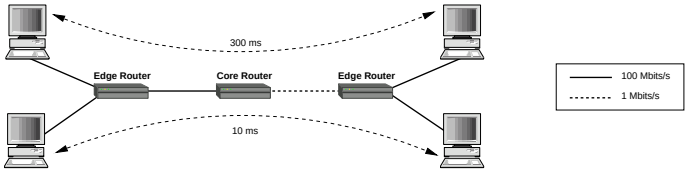


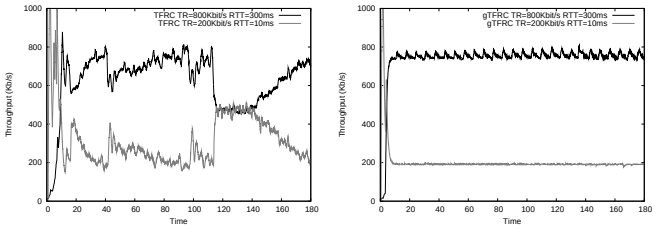
Fig. 6. The simulation topology for DiffServ experiments

We made experiments with many different RTTs and target rates configuration and give in this part a representative measurement of the efficiency of ETP/*g*TFRC . We measure the performance obtained by ETP/*g*TFRC in three scenarios.

C. Exactly-provisioned network

In figures 7, two flows are emitted on the testbed. The first one has non favorable conditions since it has the highest target rate to reach and a high RTT ($RTT = 300ms, TR = 800Kbits/s$). The second flow has the lowest target rate ($200Kbits/s$) and a low RTT ($10ms$). The results for ETP/TFRC are presented on figure 7 (a) and for ETP/*g*TFRC on figure 7 (b). We can see that ETP/*g*TFRC allows to reach the target rate more quickly than with TFRC and that ETP/*g*TFRC keeps this target rate. The reason is obvious since at the first rate decrease evaluation of the TFRC algorithm, ETP/*g*TFRC evaluates a rate equal to the target rate.

In figure 7 (a), we can see that the decreasing phase occurs for TFRC around $t = 10sec$ and that ETP/*g*TFRC does not



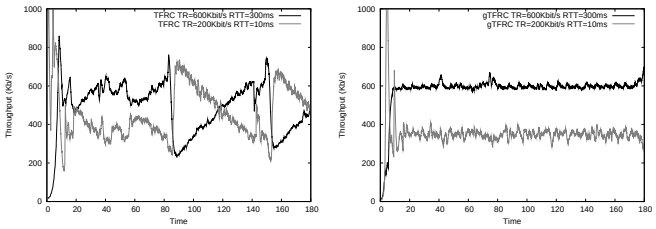
(a) TFRC versus TFRC (b) gTFRC versus gTFRC

Fig. 7. Exactly-provisioned network

at this time deliver a rate lower than the negotiated target rate (figure 7 (b)). Figure 7 (b) shows that the flow with the lower target rate and the lower RTT is constrained to reach its own target rate of $200Kbits/s$.

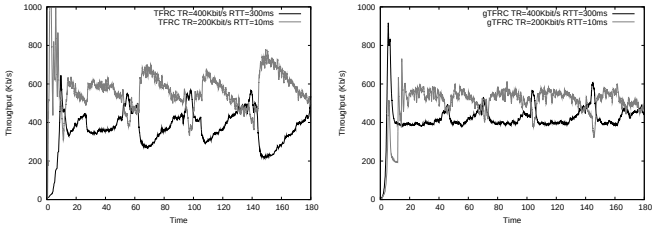
D. Over-provisioned network

These experiments deal with an over-provisioned network in two different situation, where respectively, the sum of the target rates is equal to $800Kbits/s$ (figure 8) and the sum of the target rates is $600Kbits/s$ (figure 9).



(a) TFRC versus TRFC (b) gTFRC versus gTFRC

Fig. 8. Over-provisioned network 20%



(a) TFRC versus TRFC (b) gTFRC versus gTFRC

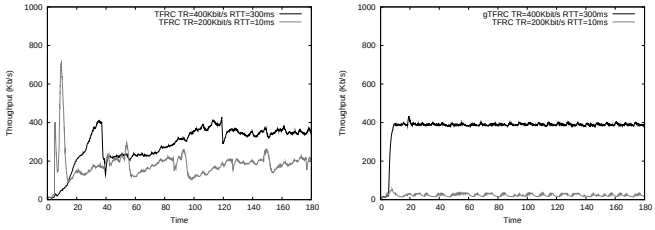
Fig. 9. Over-provisioned network 40%

The networks have respectively 20% and 40% of excess bandwidth. Moreover, the more there is excess bandwidth in the network, the more the flow with the highest target rate has difficulty to reach its target rate. This is due to the increase of the out-profile traffic which involves more losses in the network. These losses are more prejudicial for the flow with the highest target rate and the highest RTT than the lowest one. Indeed, the TFRC algorithm can estimate an ideal rate under the negotiated target rate and due to a long RTT, the flow can have difficulty to retrieve its initial throughput as during the period $[80sec, 140sec]$ on figure 8 (a).

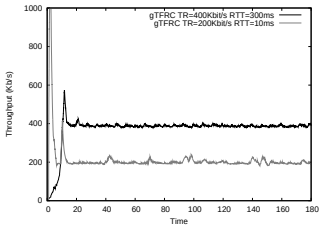
This is not the case with the use of ETP/*g*TFRC . Nevertheless, we can see in figures 8 (b) and 9 (b) that the flow with a lower RTT and lower target rate obtains a higher part of excess bandwidth. It is important to take into consideration that the proportional sharing of the excess bandwidth was not the aim of this study. This problem should remain under the responsibility of the edge router conditioning.

E. Interaction with a TCP aggregate in an over-provisioned network

The last experiment shows the interaction of TFRC or *g*TFRC and a TCP aggregate. In this experiment, two ETP flows with either TFRC or gTFRC mechanisms are sent versus an aggregate of ten TCP flows.The TCP aggregate crosses a token bucket marker with a target rate of $200Kbits/s$ and has an RTT equal to $1ms$. Both ETP flows have respectively a target rate of $400Kbits/s$ and $200Kbits/s$ for RTT equal to $300ms$ and $10ms$ respectively. Figures 10 give the results obtained for both ETP flows.



(a) 2 TFRC and 10 TCP (b) TFRC and gTFRC and 10 TCP



(c) 2 gTFRC and 10 TCP

Fig. 10. Over-provisioned network 20% with ten TCP fbws

Concerning the TCP aggregate, the throughput obtained is respectively for figures 10 (a, b, c) of $447Kbits/s$, $403Kbits/s$, $362Kbits/s$. So the TCP aggregate always reaches its target rate. In figure 10 (a), we see that with ETP/TFRC , both flows have difficulties to reach their respective target rate and that the flow with the higher target rate and RTT, does not reach a correct throughput value before $t = 120sec$. In figure 10 (b), ETP/*g*TFRC flow reaches easily its target rate. Nevertheless, due to the increase of the in-profile

traffic and the aggressive nature of the TCP aggregate, the other flow with ETP/TFRC strongly decrease its rate. Finally, on figure 10 (c), both flows use ETP/*g*TFRC and reach their target rate while the TCP aggregate still remains aggressive and reaches its target rate too.

V. Conclusions and future works

This paper has proposed a performance evaluation of an implementation of TFRC-based mechanisms into a Java composition framework named ETP.
ETP/TFRC and ETP/*g*TFRC are both evaluated on a real testbed. ETP/TFRC is compliant with [3] and a measurement campaign shows its conformance with the ns-2 reference implementation. As an extension, ETP/*g*TFRC allows to reach a minimum guarantee throughput in a DiffServ/AF class whatever the network conditions and negotiated guarantees are. These mechanisms are particularly promising in the context of Quality of Service networks. Moreover, ETP framework will alleviate their integration with other transport mechanisms such as partial reliability or timing control. The ETP protocol is expected to be deployed and evaluated over the pan-European EuQoS network.

References

[1] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB Group," Request For Comments 2597, IETF, June 1999.

[2] E. Exposito, *Specification and Implementation of a QoS Oriented Transport Protocol for Multimedia Applications*. PhD Thesis, LAAS-CNRS/ENSICA, Dec. 2003.

[3] M. Handley, S. Floyd, J. Pahdye, and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," Tech. Rep. 3448, IETF, Jan. 2003.

[4] E. Lochin, L. Dairaine, and G. Jourjon, "gTFRC: a QoS-aware congestion control algorithm," in *To appear: 5th International Conference on Networking (ICN'2006)*, (Mauritius), Apr. 2006.

[5] C. Cicconetti, M. Garcia-Osma, X. Masip, J. Sa Silva, G. Santoro, G. Stea, and H. Taraskiuk, "Simulation model for end-to-end QoS across heterogeneous networks," in *Proc. of International Workshop on Internet Performance, Simulation, Monitoring and Measurement (IPS-MoMe)*, 2005.

[6] EuQoS project, "End-to-end quality of service support over heterogeneous networks. http://www.euqos.org/," 2005.

[7] Ernesto Exposito, Mathieu Gineste, Romain Peyrichou, Patrick Sénac, and Michel Diaz, "XQOS: XML-based QoS Specification Language," in *Proc. of International Multimedia Modelling Conference (MMM)*, 2003.

[8] Ernesto Exposito, Patrick Sénac, and Michel Diaz, "UML-SDL modelling of the FPTP QoS oriented transport protocol," in *Proc. of International Multimedia Modelling Conference (MMM)*, 2004.

[9] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," *ACM Computer Communications Review*, vol. 27, Jan. 1997.

[10] J. Widmer, *Equation-Based Congestion Control*. Diploma thesis, University of Mannheim, Germany, Feb. 2000.

[11] N. Seddigh, B. Nandy, and P. Pieda, "Bandwidth Assurance Issues for TCP Flows in a Differentiated Services Network," in *Proc. of IEEE GLOBECOM*, (Rio De Janeiro, Brazil), Dec. 1999.

[12] M. El-Gendy and K. Shin, "Assured Forwarding Fairness Using Equation-Based Packet Marking and Packet Separation," *Computer Networks*, vol. 41, no. 4, pp. 435–450, 2002.

[13] E. Lochin, P. Anelli, and S. Fdida, "Penalty Shaper to Enforce Assured Service for TCP Flows," in *IFIP Networking*, (Waterloo, Canada), May 2005.

[14] K. Cho, "Managing Traffic with ALTQ," *Proceedings of USENIX Annual Technical Conference: FREENIX Track*, June 1999.