Admission Control for TCP Connections in QoS IP Network

Wojciech Burakowski and Halina Tarasiuk

Warsaw University of Technology, Institute of Telecommunications,
ul. Nowowiejska 15/19, 00-665 Warsaw, Poland
{wojtek, halina}@tele.pw.edu.pl

Abstract. The paper describes a new admission control (AC) algorithm for greedy TCP connections. The algorithm has passed positive tests in pre-production QoS IP network [2], developed inside European IST project AQUILA[1] network. The algorithm operates per single TCP flow and it assumes the setting of advertised receiver TCP window size for maintaining ideal TCP behaviour (no packet losses). The QoS objective is to guarantee the requested bit rate, say R_{req}, a user demands from the network. Furthermore, on the basis of the R_{req} and information about round trip time (RTT), the user request is mapped into the form of single token bucket parameters, i.e. token accumulating rate (R) and bucket size (BS), constituting input parameters for AC decision. For admitted TCP connection the R_{req} is guaranteed, even if running connections differ in R_{req} and RTT_{min}. Included simulation results confirm the efficiency of the algorithm.

1 Introduction

A packet flow associated with single TCP connection is commonly named as elastic one. This is due to the TCP behaviour, which allows for adjusting the TCP sending data rate depending on traffic conditions in network. Therefore, on the contrary to the flows produced by streaming applications usually requiring hard end-to-end QoS guarantees e.g. [8], the elastic flows were traditionally handled in best effort way. However, most of the currently available applications in Internet use, as a transport protocol, TCP or TCP-like. The examples are: FTP, Telnet, e-commerce, WWW, RealPlayer etc. Now, the challenge is to provide some QoS guarantees also for TCP flows with special focus on minimum throughput [2,3]. Satisfying this objective is of special interest in future QoS IP network. The key issue for providing QoS is the admission control (AC).

The most promising solution for QoS IP network is the DiffServ architecture [4, 5]. One of the proposals for such a network, developed inside the AQUILA IST European project [2], is an enhancement of generic Diffserv architecture by adding new functionalities for admission control and resource management as well as by defining new set of network services.

[1] European IST project „ AQUILA – Adaptive Resource Control for QoS in IP-based Layered Architecture" (2000-2003)

The AQUILA architecture is depicted on Fig. 1. It assumes that the AC agent (ACA) is located at each Edge Routers (ER) for assuring that traffic submitted to the core does not exceed given AC limit, being a part of total link capacity between the ER and CR (Core Router). The value of AC limit is allocated by the Resource Control Agent (RCA), situated on the top of the network [2]. The call handling scenario is the following. A user, for requesting a connection, uses the end-user application toolkit (EAToolkit) for sending his request to the ACA with traffic contract parameters. The ACA admits/rejects this call, depending on current load on ER-CR link and assigned AC limit.

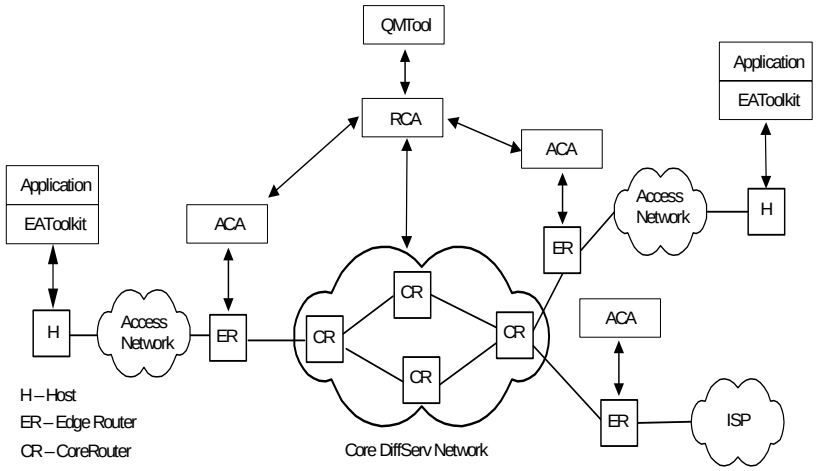

Fig. 1. General AQUILA network architecture

For now, four different types of packet flows have been recognized as typical in Internet and requiring QoS guarantees. There are the following: (1) streaming constant bit rate (e.g. VoIP), (2) streaming variable bit rate (e.g. video applications), (3) elastic, produced by greedy TCP or TCP-like sources (e.g. FTP), and (4) elastic, non-greedy TCP sources (e.g. home banking). In this spirit, four QoS network services (NS) have been defined and implemented in AQUILA. Each NS is optimised for specific type of packet flows and has its own traffic handling mechanisms, including admission control.

In this paper we focus on the AC algorithm for the NS aimed at handling greedy TCP flows with throughput guarantees as the QoS objective, packet flows of type (3). The excellent candidate for using this NS is a FTP user. The proposed AC algorithm operates per TCP flow and assumes the setting of advertised receiver TCP window size, allowing us to maintain the ideal TCP behaviour (no packet losses). It assumes that a user, before establishing TCP connection, submits its request to the network. The traffic contract specifies the target requested bit rate, say R_{req}. Furthermore, on the basis of the R_{req} and information about round trip time (RTT) of the TCP connection, the user declarations are mapped into the form of single token bucket parameters, say token accumulating rate (R) and bucket size (BS), constituting input parameters for AC decision.

The investigated by many authors AC scheme for TCP connection, e.g. in [3, 13], assumes to reject new TCP connections simply by dropping SYN and ACK SYN segments in the case of the network congestion. For instance, the congestion can be identified when number of waiting packets in the queue exceeds predefined threshold. However, this approach could guarantee a fair share of link capacity between running TCP connections but without possibility for providing them rate differentiation. Another interesting results, which can be adopted, but not in straightforward way, for the purpose of AC, were reported in [9] and corresponding to the possibility for getting QoS differentiation of TCP flows by using token bucket marking mechanism. The list of proposed AC algorithms for elastic traffic and based on some declarations is presented in [14], but none of them is explicitly targeted for guaranteeing the requested rate for TCP.

The rest of the paper is organized as follows. Sections 2 discusses the TCP flow control mechanisms and consequences of using token bucket mechanism for the purpose of TCP traffic description and policing, the factors taken into account in the proposed AC, described in Section 3. Section 4 shows numerical results illustrating effectiveness of the considered AC. Finally, the Section 5 concludes the paper.

2 Factors Influencing AC

Before describing the proposed AC algorithm for TCP greedy flows, we shortly recall the TCP mechanisms for traffic control and consequences of using token bucket mechanism for policing of TCP traffic. We argue that these two factors are crucial from the AC viewpoint.

There have been several implementations of TCP protocol. The most important modifications were the introduction of congestion and avoidance techniques. This version is usually referred to as Tahoe TCP. Tahoe TCP regulates the number of packets and it sends by inflating and deflating a window according to the network requirements. In order to do this, the TCP sender uses the cumulative acknowledgments sent by the receiver. If no packets are lost, TCP keeps inflating the window in the three main phases: slow start, congestion avoidance and maximum window [1,10,11,12], Reno, New-Reno and Sack were designed to improve the performance of Tahoe TCP when packets are lost. However, when no packet loses occur they behave like Tahoe TCP. Thus, in the ideal case of no packet loses all these different implementation should have the same performance.

From the proposed AC algorithm viewpoint, the critical issue is to control the changes (if any) of transmission window size. In the TCP case, it is rather difficult to predict TCP behaviour when the packet losses are not under control. Therefore, for the purpose of AC it is reasonable to consider ideal case, as discussed in [6]. The modelling issues of TCP behaviour in best effort Internet network, which are taking into account impact of packet losses on received throughput were discussed by many authors, e.g. [7,9]. Additionally, in [9] the authors pointed out that using token bucket marking mechanism is not sufficient for getting ideal TCP service differentiation. Anyway, these studies assumed the overload network conditions, which are not adequate when AC algorithm is employed.

2.1 Studied System for AC

The system for the AC is depicted on Fig.2. It consists of C capacity link with associated buffer size B, and corresponds to the bottleneck link between ED and CR routers in the AQUILA network (see Fig.1). A number of running TCP connections share the system. In general case, these TCP connections may differ in the requested bit rate R_{req} and RTT.

A user, before establishing TCP connection, submits its request to the network. The traffic contract specifies the target requested bit rate R_{req}. Furthermore, on the basis of the R_{req} and information about round trip time (RTT) of the TCP connection, the user request is mapped into the form of single token bucket parameters, say rate (R) and bucket size (BS), constituting input parameters for AC decision. This traffic is policed at the network entry point.

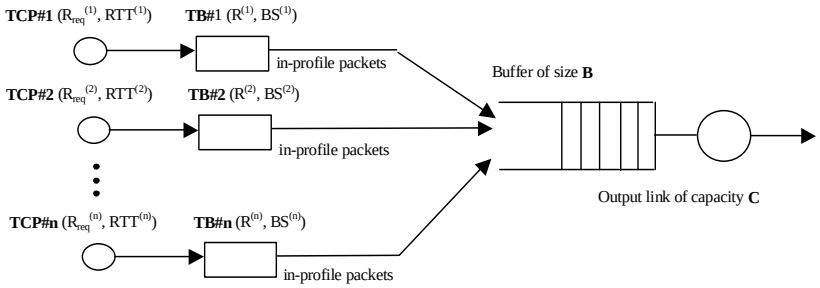

Fig. 2. System with n running TCP connections: TCP#n –n-th TCP source; TB#n – Token Bucket mechanism dedicated for flow#n

2.2 Consequences of Applying Token Bucket Mechanism

Ideal AC should guarantee that TCP source would get throughput/goodput close to R_{req} value. Let us remind that token bucket mechanism operates on deterministic parameters while TCP sends packets in a non-regular way accordingly to current sending/transmission window and RTT. As a consequence, one can observe that the traffic produced by a single TCP source has a tendency to follow rather statistical behaviour than deterministic one. Therefore, we argue that this what could be policed by the token bucket should be closely related to the value of TCP sending window size, like maximum guaranteed/requested value of TCP sending window size W_{req}. Let us remind, that token bucket polices the worst case traffic, which could occurs sporadically that one can observe in the TCP traffic. The value of W_{req} should be specified for traffic contract (in direct or non-direct way). Starting from this point, we deduce for token bucket parameters the following relations:

$$L_{in} * BS / (L_{in} - R) = W_{req} \quad (1)$$

, where L_{in} denotes the input link capacity (in bps).

The expression (1) shows the way for dimensioning BS on the basis of assumed W_{req}. The W_{req} denotes the maximum packet burst (ON period), counted in bits, which could be emitted by the TCP source. Therefore, the time duration of ON period, T_{ON}, in this case is:

$$T_{ON} = W_{req} / L_{in} \tag{2}$$

$$BS / R \geq T_{ON} + T_{OFF} \tag{3}$$

, where T_{OFF} is the time duration between packet bursts (OFF period). From (3) we can obtain additional constrains referring to the minimum value of RTT for TCP connection. The condition for RTT_{min} results from (2) and (3), and is the following:

$$RTT_{\min} = T_{ON} + T_{OFF} \leq BS / R \tag{4}$$

The condition (4) for RTT_{min} results from the fact that the token bucket size should be fulfilled to the maximum value (=BS) before starting new ON period. Notice that less rigorous traffic pattern is generated when current RTT is greater than RTT_{min}.

Summarizing, from the point of view of the token bucket mechanism the worst case traffic, which a TCP-controlled stream could produce is obviously of ON/OFF type, as depicted on Fig.3.

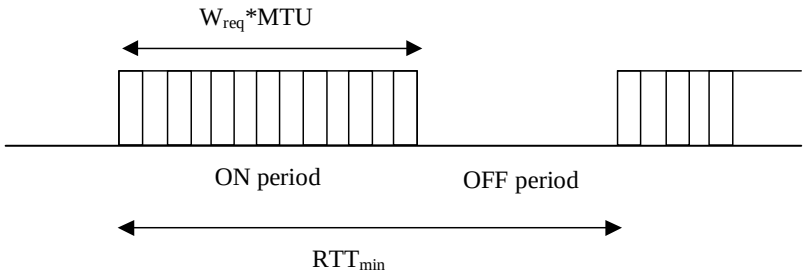

Fig. 3. The worst case traffic produced by a TCP-controlled stream from the point of view of the token bucket mechanism

In fact, the token bucket mechanism has ability for distinguishing between in- and out-of-profile packets. The in-profile packets are in accordance with the traffic declarations while the out-of-profile packets excess the policed profile. Therefore, there is reasonable to consider out-of-profile traffic as an additional, non-controlled traffic, which is not taken into account by AC. So, the only traffic for which we can give some guarantees corresponds to the in-profile traffic. As a consequence, in further part of this contribution we focus only on AC for in-profile traffic.

The requirement for the AC is to give guarantees that the declared (requested) volume of throughput by a user R_{req} will be provided. Taking into account that we use token bucket mechanism for traffic contract policing, with above mentioned limitations, we argue that additional information about RTT_{min} is needed. The knowledge of the RTT_{min} allows us for proper tuning of token bucket parameters (see eq. 4). Anyway, the guaranteed R_{req} will never exceed the assumed token accumulating rate R and this results in straightforward way from the token bucket concept. The relations between R_{req} and R will be discussed in the further part of the

paper. The BS value strictly depends on RTT_{min}; if RTT_{min} is greater then BS should also be greater (assuming given R).

2.3 Forming TCP Traffic

The desirable shape of sending window size evolution for any admitted TCP connection is shown on Fig. 4. In this scenario we assume that no packet losses are observed as long as the transmitted (sending) TCP window does not exceed the W_{req}. However, this requires setting W_{req} for advertised window size in the receiver.

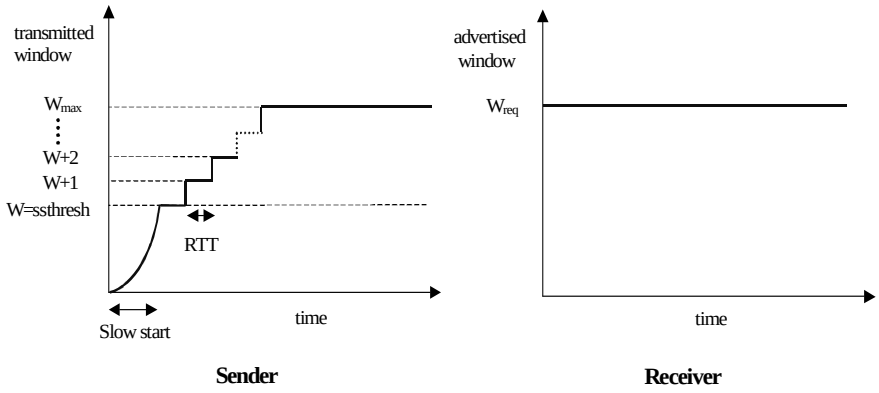

Fig. 4. Exemplary ideal TCP behaviour

TCP connection rate is changed and one can distinguish between two areas: (1) corresponding to the beginning phase of the connection, from the starting time to the moment when transmitted window size reaches W_{req}, and (2) corresponding to the second phase (if occurs), when the transmitted window size is fixed and is equal to W_{req}. During the phase (1) the TCP starts with slow start mechanisms, changing its transmitted window size in accordance to negative exponential function, next passing to the congestion avoidance, during which the transmitted window is increased by 1 each RTT. This means that the TCP connection rate is changed during the phase 1. As a consequence, it is rather difficult to consider for this period the requested rate R_{req}. On the contrary, the phase (2) is characterized by stable behaviour of transmitted TCP window size, which is equal to W_{req}. For this period, the TCP connection rate is almost fixed, with small changes caused by RTT variability. Summarizing, in the context of AC we will consider the R_{req} demand as reached in phase (2).

3 AC Algorithm

The starting point for AC algorithm constitutes the TCP connection demands, which are in this case expressed in the form of:

- the targeted requested rate R_{req}. This rate should be reached during phase (2);
- and, the information about minimum $RTT_{min.}$.

Now, on the basis of (R_{req}, RTT_{min}), the next step is to set the parameters of associated token bucket (R, BS) and advertised window size W_{req}. For phase (2) of ideal TCP behaviour the following relation takes place:

$$R_{req} = W_{req} / RTT_{avg} \tag{5}$$

,where RTT_{avg} denotes average round trip time.

The appropriate setting of token bucket parameter R should take into account the worst case traffic, which occurs when RTT_{min}. Therefore, R should satisfy:

$$R = W_{req} / RTT_{\min} > R_{req} . \tag{6}$$

Then,

$$W_{req} = R * RTT_{\min} . \tag{7}$$

By substituting (7) to (5), we receive the relation between R and R_{req}, which is:

$$R = R_{req} \frac{RTT_{avg}}{RTT_{\min}} . \tag{8}$$

However, for calculating R from (8), we need RTT_{avg}. Notice that RTT_{avg} depends on traffic load in the network. In the contents of AQUILA architecture, for evaluating RTT_{avg} we may do simplified assumption that variable part of RTT, caused by buffering packets in the routers, is mainly the packet waiting times in the edge routers. The core network is over-dimensioned and, as a consequence, the packet waiting times in core routers are negligible. We can write

$$RTT_{avg} = RTT_{\min} + D \tag{9}$$

,where D is the variable part of RTT. Now, we can concentrate on evaluation of D. The proposed approximation for D, which gives accurate results, was derived by assuming M/D/1 queuing system with $\rho=R_{req}/R$ and constant packet service time equal MTU/R, where MTU is the maximum transfer unit (in bits). The final formula for R is:

$$R = R_{req} + \sqrt{A/(R_{req} + \sqrt{A/(R_{req} + \sqrt{A/R_{req}})})} \tag{10}$$

, where $$A = R_{req}^2 MTU /(2RTT_{\min}) \tag{11}$$

The BS is calculated in straightforward way from expression (1) by:

$$BS = (W_{req} * (L_{in} - R)) / L_{in} \tag{12}$$

Now, we can define admission control rules, when the link capacity C and buffer size B (counted in number of packets). Assuming N running connections, new flow, characterized by (R_{req}, RTT_{min}) with mapping to (R_{new},BS_{new}), is accepted only if:

$$\sum_{i=1}^{N} R_i + R_{new} \le C \tag{13}$$

$$\sum_{i=1}^{N} BS_i + BS_{new} \le B \tag{14}$$

4 Simulation Results

This section contains simulation results showing the effectiveness of the investigated AC algorithm. In particular we present the results obtained assuming TCP connections differ in R_{req} and RTT. For this purpose the bottleneck network topology is considered as depicted in Fig.5. In this configuration two edge routers are connected 2 Mbps link. Furthermore, 5 PCs and 2 servers are attached by LAN Ethernet to Router 1 and Router2, respectively. The TCP connections may be established between a pair PC-server. In addition, we can tune value of RTT_{min} by introducing additional (constant) delay in the inter-router link or in the links between Router 2 and servers.

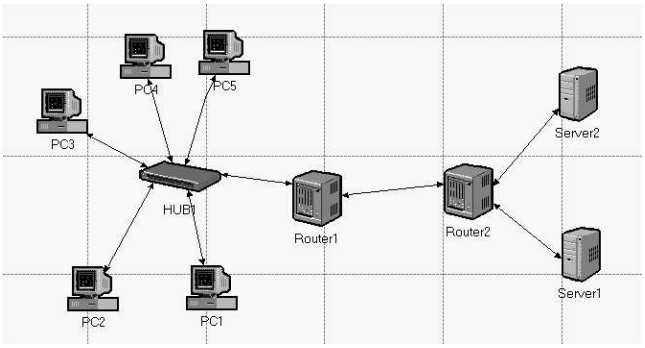

Fig. 5. Tested network topology

All tested TCP connections are of greedy type, sending packets of constant size, MTU =1500 bytes. The simulation results are obtained using OPNET software, assuming TCP Reno version.

The TCP throughput characteristics referring to the cases with different number of running TCP connections, requested bit rates and RTT_{min} are presented in Tables 1 and 2.

Table 1. shows the results corresponding to the case, when conditions for all running TCP connections are the same. The system is uploaded to the AC limit (link of 2 Mbps and minimum sufficient buffer size) and serves different number of TCP flows depending on their requested rate. In particular, we consider the test cases of 5, 4, 2 and 1 TCP connections. For instance, in the scenario with 5 connections, each flow has R value fixed as 400 kbps, while in the scenario with only 1 connection the R value is 2000 kbps. In addition, all cases were simulated for RTT_{min} equals 0.1 and 0.2 sec.

One can observe that the obtained minimum TCP throughput is a bit greater than the requested for all tested cases. Additionally, the simulations confirm the correctness of applied approximate formulas for RTT_{avg} assessment.

Table 1. Throughput characteristics (with 95% confidence interval): homogenous tcp connections. Thr- TCP throughput

Number of flows	**5**	**4**	**2**	**1**
RTT= 0.1s				
R (kbps)	400	500	1000	2000
W_{req}(bytes)	5000	6250	12500	25000
BS (bits)	38400	47500	90000	160000
RTT_{avg}(s) (anal)	0.1388	0.1347	0.1245	0.1109
RTT_{avg} (s) (sim)	0.116	0.114	0.113	0.114
R_{req} (kbps)	288.022	371.079	803.054	1 704.610
Thr (kbps)	332.7-354.9	387.7–411.0	818.4-846.7	1 721.8-1 746.3
RTT= 0.2s				
R (kbps)	400	500	1000	2000
W_{req} (bytes)	10000	12500	25000	50000
BS (bits)	76800	95000	180000	320000
RTT_{avg} [a](s)	0.2548	0.2490	0.2346	0.2245
RTT_{avg} [s](s)	0.2164	0.2165	0.2163	0.2144
R_{req} (kbps)	313.897	401.527	852.305	1 781.740
Thr (kbps)	335.6-357.9	428.5-452.5	892.5–926.1	1 835.3–1 860.9

Table 2 presents the simulation results for the case of heterogeneous TCP connections differing in both requested rate as well as RTT_{min}. The system is again uploaded to the AC limit equal 2 Mbps. The minimum received throughput is also close to the requested rate. In all tested cases the ideal behaviour of TCP was observed, as it was expected.

Table 2. Throughput characteristics (with 95% confidence interval): heterogeneous TCP connections. Thr – TCP throughput

TCP connections	R (kbps)	W_{req} [bytes]	BS [bits]	R_{req} (kbps)	RTT_{min} (s)	RTT_{avg} (s)(anal)	RTT_{avg} (s)(sim)	Thr (kbps)
PC1-Serv1	400	5000	38400	288.022	0.1	0.1388	0.1178	310 – 331
PC2-Serv2	400			288.022				
PC3-Serv3	600	15000	112800	490.268	0.2	0.2447	0.2194	515 – 540
PC4-Serv4	600			490.268				

5 Summary

In this paper we have proposed new admission control algorithm for handling greedy TCP connections with QoS guarantees, which was tested in AQUILA QoS IP

network. The QoS objective is to guarantee requested bit rates. For this purpose, the ideal TCP behaviour is maintained during the connection thanks to the appropriate setting of advertised window size in the receiver. The submitted parameters by a TCP source are: (1) requested rate, R_{req}, and (2) minimum round trip time RTT_{min}. These parameters are mapped into the form of the parameters of single token bucket, which constitutes the base for admission control. The included simulation results show that the effectiveness of the proposed AC is satisfied. The requested bit rate by each TCP connection is always guaranteed, even if a mix of TCP connections differing in rate requests and round trip times share the same network resources.

6 References

1. Allman, M., Paxson, V., Stevens, W.: TCP Congestion Control. Internet RFC 2581, April 1999
2. Bak, A., Burakowski, W., Ricciato, F., Salsano, S., Tarasiuk, H.: Traffic handling in AQUILA QoS IP Networks. Quality of Future Internet Services, Lecture Notes in Computer Science 2156, Springer 2001
3. Benameur, M., Ben Fredj, S., Delcoigne, F., Oueslati-Boulahia, S., and Roberts, J.W.: Integrated Admission Control for Streaming and Elastic Traffic. Quality of Future Internet Services, Lecture Notes in Computer Science 2156, Springer 2001
4. Blake, S. et al.: An Architecture for Differentiated Services. Internet RFC 2475, December 1998
5. Bernet, Y. et al.: An Informal Management Model for Diffserv Routers. Internet Draft, draft-ietf-diffserv-model-06.txt, February 2001
6. ElAarag, H., Bassiouni, M.: Performance evaluation of TCP connections in ideal and non-ideal network environments. Computer Communications 24(2001), pp. 1769-1776
7. Padhye, J., Firoiu, V., Towsley, D., Kurose, J.: Modeling TCP Throughput: A Simple Model and its Empirical Validation. Proc. ACM SIGCOM'98, August 1998
8. Roberts, J., Mocci, U., Virtamo J. (eds.): Final report COST 242, Broadband network teletraffic: Performance evaluation and design of broadband multiservice networks. Lectures Notes in Computer Science 1155, Springer 1996
9. Sahu, S., Nain, P., Towsley, D., Diot, C., Firoiu, V.: On Achievable Service Differentiation with Token Bucket Marking for TCP. Proc. ACM SIGMETRICS'00, Santa Clara, CA, June 2000
10. Stevens, W.R.: TCP/IP Illustrated, Volume1: The Protocols. Addison-Wesley Publishing Company, 1996
11. Stevens, W.R.: TCP/IP Illustrated, Volume2: The Implementation. Addison-Wesley Publishing Company, 1996
12. Stevens, W.R.: TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. Internet RFC 2001, January 1997
13. Mortier, R., Pratt, I., Clark, C., and Crosby, S.: Implicit Admission Control. IEEE Journal on Selected Areas in Communications, Vol. 18, No. 12, December 2000
14. Brichet, F., Mandjes, M., Sanchez-Canabate, M.F.: Admission control in multiservice networks. Proceeding of the Mid-Term Seminar, 1999, Villamora, Portugal