Negotiation and Authentication of QoS Flows

Carlos Rabadão[1, 2], crab@estg.iplei.pt
Edmundo Monteiro[2] edmundo@dei.uc.pt

[1] Superior School of Technology and Management
Polytechnic Institute of Leiria
Morro do Lena – Alto do Vieiro
2411-901 Leiria
Portugal
http://www.estg.ipleiria.pt

[2] Laboratory of Communications and Telematics
CISUC / DEI
University of Coimbra
Polo II, Pinhal de Marrocos, 3030-290 Coimbra
Portugal
http://lct.dei.uc.pt

Abstract

The main objective of the IETF Differentiated Services (DiffServ) model is to allow for the support of different levels of service to different flows of information aggregated in Classes of Service (CoS), on a TCP/IP infrastructure. This differentiated treatment will motivate some users to get better Quality de Service (QoS) for their flows however without assuming the associated costs. This leads to the theft of resources that, in extreme situations, will have as consequence the denial of quality of service (DQoS) contracted by users for its flows.
In the DiffServ model the authentication of flows is carried out on a per packet basis, at the entrance of each domain. The flow classification is supported by some of the IP packet header fields. This approach shows some security limitations that are inherent to the DiffServ model. To overcome these limitations, this work proposes a system for QoS negotiation and authentication, aimed to authenticate clients and to authorize flows, in a dynamic way, at the entrance of the DiffServ domains.
In this paper two scenarios for authentication of clients and authorization of flows with QoS are presented and discussed. The security aspects taken into account include the authentication of clients and the authorization of flows accessing the communication resources. The issues related to the confidentiality and the integrity of the information is relegated for other modules of the communication systems. To support the authentication and authorization of flows two protocols are proposed to operate in the intra and inter-domain environments.

Keywords: QoS negotiation, denial of QoS (DQoS), DiffServ authentication and authorization.

1. Introduction

In communication systems, the expression “Quality of Service” (QoS) is used to characterize the capacity of the system to support data flows with service guaranteed parameters (e.g. bandwidth, delay, jitter, losses) in a more or less strict way. The QoS mechanisms impose priorities and restrictions in the access of flows to available communication system resources. In the case of the DiffServ model [1] this traffic prioritization is supported by the identification of Classes of Traffic (groups of multiple flows) done on the specific field of the header of IP packets [2]. As discussed in [3, 4] this approach has some security limitations, namely authentication and authorization.
The IETF DiffServ working group considered some methods to reduce the inherent security limitations of the DiffServ model [4]. These include auditing and IPSec [5, 6]. However the vulnerabilities to security attacks, such as man-in-the-middle and Denial of QoS (DQoS), remain open issues [7].
To overcome some of the security limitations of the DiffServ model, an architecture for the support of negotiation QoS services with authentication is presented and discussed in Section 2. In this proposal QoS clients will need to be initially authenticated in an Authentication Server. This server will issue credentials to enable the client to contact the QoS Server. Subsequently, the client will request the QoS server the allocation of network resources with the characteristics needed for its flow. After that and according to the Service Level Agreement (SLA) already established between the provider and the client, the QoS Server will decide to authorize (or not) the new flow acting over the configuration of the Edge Router of the DiffServ domain. After the expiration of the authorization, the Edge Router will again be configured by the QoS Server to block the access to the DiffServ domain. To integrate the proposed architecture a QoS negotiation and authentication protocol is discussed in Section 3. The protocol intends to offer a safe method for the negotiation and authentication of QoS flows in DiffServ environments. The operation of the protocol will be discussed for the client and the server side. Later on the validation of the proposals will be addressed. In Section 4 the experimental test-bed is presented. Evaluation is discussed in Section 5. Finally, Section 6 will be devoted to conclusions and directions for future work.

2. Authentication and Authorization Architecture for DiffServ

The DiffServ model includes a set of relatively simple mechanisms. At the network entrance the traffic is

classified, conditioned and mapped into one of the available traffic classes [1], being each class identified by the Differentiated Service Code Point (DSCP) field of the IP packet [2]. This way, the DiffServ model allows the support of different levels of service to different flows of information aggregated in classes of services (CoS).
In the DiffServ model the authentication and authorization of flows is carried out on a per packet basis, at the entrance of the domain. As said before, this approach presents some security limitations that can be exploited by users less scrupulous that try to get better QoS for their flows without however assuming the associated costs, leading to the theft of resources that, in extreme situations, could result in Denial of QoS (DQoS) on the active flows. To reduce these security limitations an authentication and authorization architecture is proposed in this section and analyzed in two different scenarios allowing for dynamic authentication of clients and authorization of flows in intra and inter-domain environments.

2.1 Intra-domain scenario

In the intra-domain scenario, the QoS client uses the authentication and QoS negotiation and management available at the service provider. Thus, whenever the client wants to generate a flow, he will have to authenticate itself before at the Authentication Server of provider domain – usually called Internet Service Provider (ISP) – and get the credentials to contact the QoS Server – also called Bandwidth Broker (BB) – of the domain. After this, the client will request to the BB the allocation of network resources to support its flow. The BB will then authorize or reject the new flow, taking into account the existing Service Level Agreement (SLA). This will be done acting on the configuration of ingress Edge Router of the DiffServ domain. After the expiration of the authorization, the ingress Edge Router will be configured again to block the flow. Figure 1 shows the described scenario. Figure 2 shows the same scenario decomposed in 5 layers: SLA Management, Security, Accounting, QoS and Equipment.

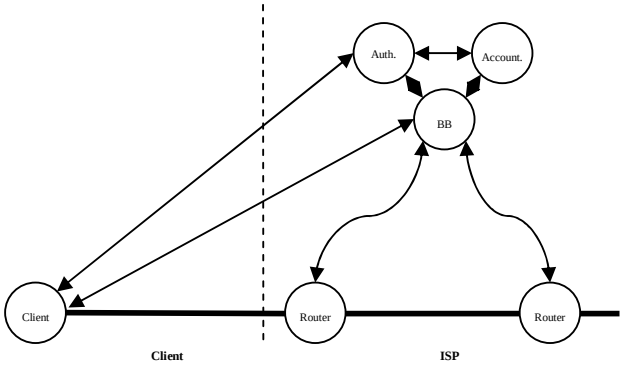

Figure 1 – Authentication and authorization in intra-domain

The functionality of the modules of the above architecture is described bellow for the client and server management sub-systems.

Client System

Following are the functionalities of the modules of the Client System of the proposed architecture:

- **SLA module:** responsible for negotiation of new SLAs and changes in the existing ones, at the SLA Management System of the ISP;
- **Accounting module:** responsible for the access to the information concerning accounting of the resources used by the client;
- **Authentication module:** responsible to authenticate the clients in the Authentication Server of the ISP and to receive credentials to get access to the QoS Server of the ISP;
- **QoS module:** responsible for the analysis of the QoS requirements of the client applications, for the forwarding of these requirements to the QoS Server and for the labelling of IP headers with the DSCP indicated by the QoS Server. The labelling is not mandatory because it can also be done in the ingress router of the DiffServ domain.

Management System

Following are the functionalities of the modules of the Management System of the proposed architecture:

- **SLA module:** responsible for the access policies of the DiffServ domain, for the negotiation of new SLAs and for the renegotiation of existing SLAs. The module is also responsible for updating the database of access policies, the accounting module with information of the new client and the authentication module with a new account to access to the domain;

- **Accounting module:** responsible for information regarding the state of clients' accounts. This information is collected from the Authentication module (duration the exchange of the credentials) and from the QoS module (QoS characteristics and resources used by the client);
- **Authentication module:** responsible for the authentication of the clients and for the issuing of credentials allowing the client to contact the QoS module. The duration of the credentials can be used by the accounting module, as previously described;
- **QoS module:** this module is responsible for the processing of resource requests from the clients, for the verifications of QoS requirements against client SLAs and configuration of ingress routers to allow the communication. The module will also will be able to accounting information concerning the used resources.

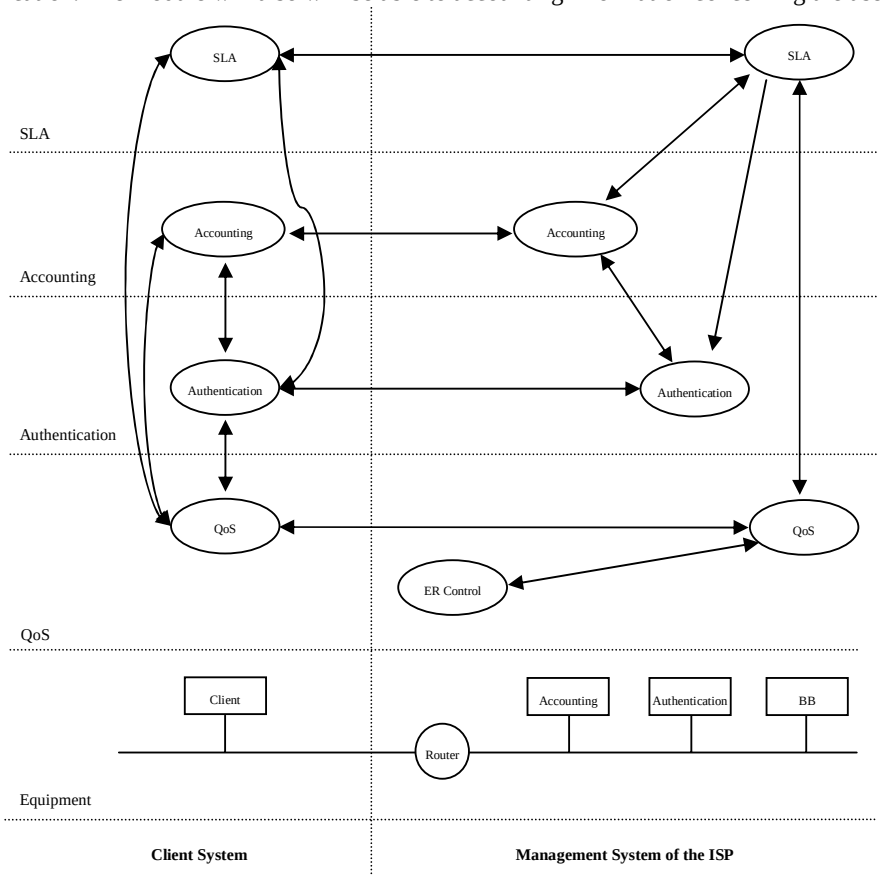

Figure 2 – Layered view of the intra-domain scenario

2.2 Inter-domain scenario

In the inter-domain scenario the client is inserted in a local infrastructure of communications that supports authentication and management of SLAs systems. The interaction between the client and the local servers are similar to the one described for intra-domain scenario. The main difference from this scenario pertains to the interactions between two distinct DiffServ domains, that is, the interactions with inter-domain scope. However, the proposed architecture is limited to the interaction between two consecutive DiffServ domains, not being adequate for a scenario of global communication, involving multiples ISPs and DiffServ domains. Figure 3 shows the described scenario.

The operation of the system is similar to the intra-domain scenario, until the point where the local BB receives the request for the allocation of resources from the client. After this point, the authentication module of the client domain requests credentials to the Authentication System of the ISP domain, to enable the local BB to be authenticated at the BB of the ISP domain. After this, and based in the already established SLA, the BB of the ISP domain will be able to authorize the new flow, informing the BB of the client domain and proceeding with the reconfiguration of the domain ingress router. After the reception of the confirmation of resource allocation, the local BB will inform the client and configure its ingress router, allowing the establishment of the flow.

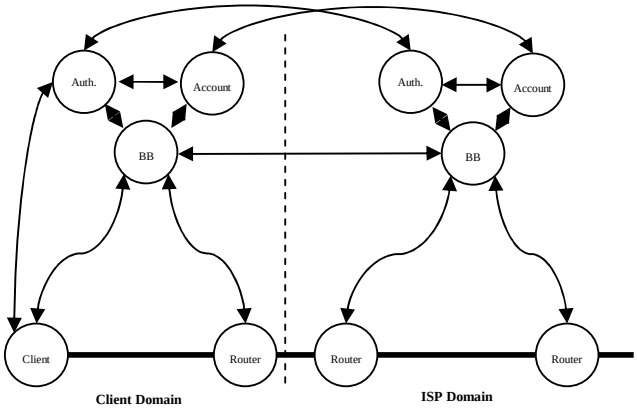

Figure 3 - Authentication and authorization in inter-domain

In Figure 4 a more detailed structure of the described scenario is illustrated, by layers. The operation of the various modules of the Client Management System combines the functionalities of Client Management System of the previous scenario. The operation of the Management System of the ISP domain is also similar to the one described in the previous scenario.

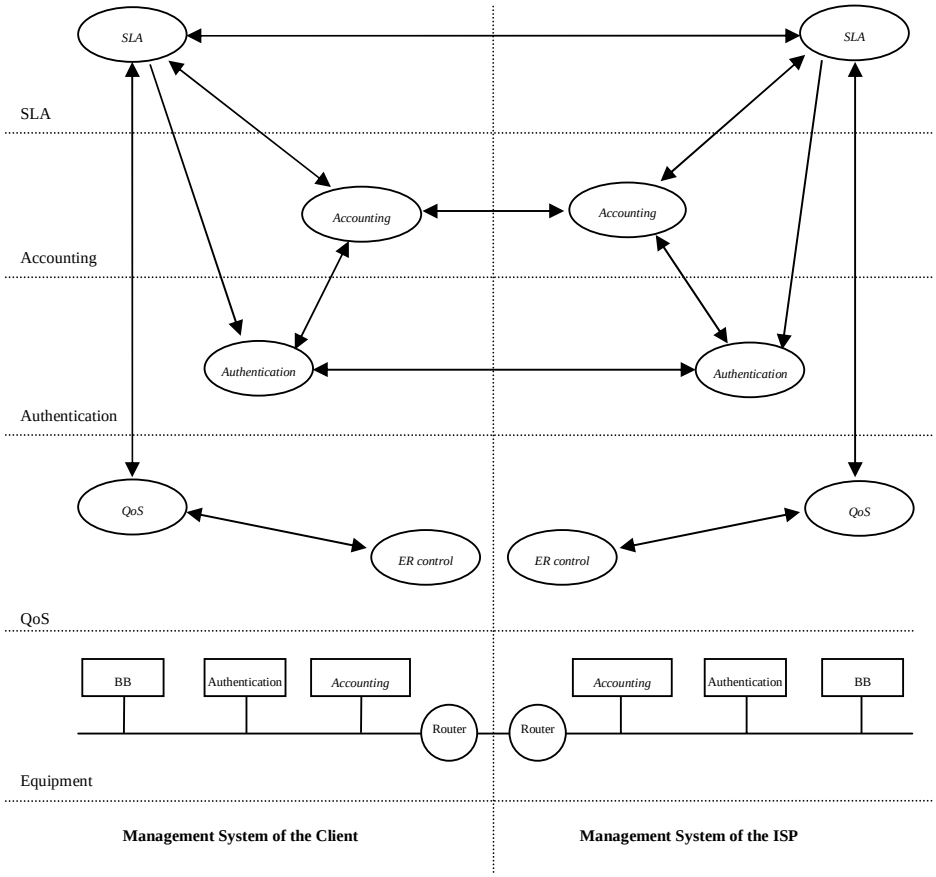

Figure 4 - Layered view of the inter-domain scenario

3. QoS Negotiation Protocol

In this section a QoS Negotiation Protocol for the scenario described in the previous section is proposed and described. The description will focus on the client and server side behaviour of the protocol.

3.1 Client Side

From the client side perspective, the QoS Negotiation Protocol enables the negotiation of QoS for communication applications running at the client. The description of the protocol uses State Diagram (Figure 5) and the correspondent Transition Table (Figure 6).

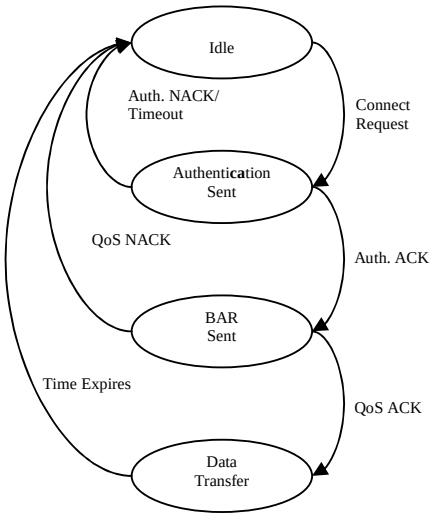

Figure 5 - Negotiation Protocol in the Client: State Diagram

Event / Present State	Connect Request	Auth. ACK	Auth. NACK	Timeout	QoS ACK	QoS NACK	Time Expires	
Idle (0)	Auth.							**Action**
	1							**New state**
Authentication Sent (1)		BAR	NA	NA				**Action**
		2	0	0				**New state**
BAR Sent (2)					StartData	NA		**Action**
					3	0		**New state**
Data Transfer (3)							StopData	**Action**
							0	**New state**

Auth - Sends authentication data; **BAR** – Bandwidth Allocation Request with QoS; **StartData** - Starts the Data transmission; **StopData** – Stop the data transmission; NA – No Action.

Figure 6 - Negotiation Protocol in the Client: Transition Table

As illustrated, the protocol has the following states:

- **Idle**: this state is reached when the protocol is initialized or when one of the following events occurs: *Time Expire*, *QoS NACK*, *Auth. NACK* and *Timeout.* The protocol remains in this state until *Connect Request* is received;
- **Authentication Sent:** this state is reached after reception of the event *Connect Request*. The actions taken in this state are: the request of credentials at the Authentication Server and the subsequent authentication in the QoS Server;
- **BAR Sent:** the protocol reaches this state after a successful authentication process, signalled by the *Auth ACK* event. After the reception of this event a *Bandwidth Allocation Request* (BAR) will be sent to the QoS Server;
- **Data Transfer:** if the request **BAR Sent** is satisfied by the QoS Server (*QoS ACK*), the protocol will go into this state, where the action to take will be the sending of data with the requested QoS parameters, specified in the BAR message. At the end of the time established by the flow, the protocol will leave this state and will go to the initial state **Idle**.

3.2 Server Side

The QoS Negotiation Protocol supports QoS negotiation between the client and the QoS Server. A State Diagram will be used to describe the server side behaviour (Figure 7) complemented with a Transition Table (Figure 8), to describe the operation of the protocol. As illustrated, the Negotiation Protocol in the Server has the following states:

- **Idle**: this state is reached when the protocol is initialized or when one of the following events occurs: Client

ACK, Client NACK, Auth NACK and Auth ACK. It remains in this state until one of the following events occurs: Auth.Request, Bandwidth Allocation Request (BAR) or BAR Timeout.

- **Authentication Verify**: after the reception of the *Auth.Request* message from the Client the protocol goes into this state. The actions taken here will be the output of credentials to the client by the Authentication Server and the validation of these credentials by the QoS Server. After this the client will be able to communicate with the QoS Server.
- **QoS Verify**: after the reception of event BAR, the protocol reaches this state. The action taken in this state is the validation of the BAR, by the QoS Server.
- **Router Configuration**: the protocol reaches this state when the reply of the QoS Server is positive (*BAR ACK*). Here it will go to add filtering rules to the Ingress Router, to allow for the new flow.
- **Wait Client**: the protocol will transit for this state if one of three situations occurs: BAR denied (BAR NACK). A filter is added to the ingress router (Router ACK) or removed from the ingress router (Router NACK). In this state the client will be informed about the occurred action.

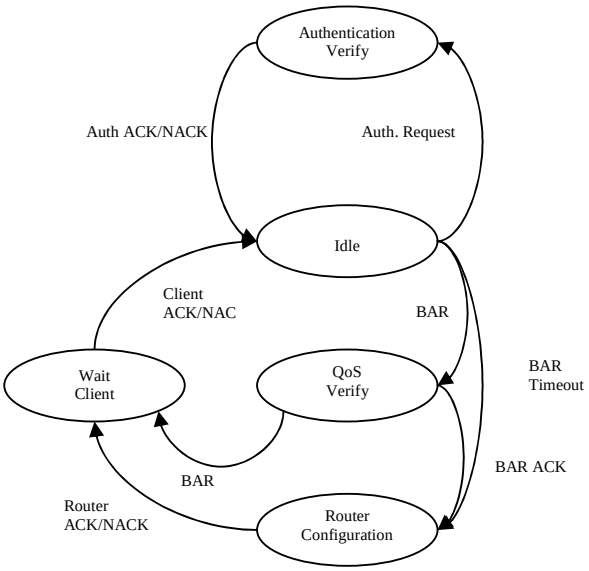

Figure 7 - Negotiation Protocol in the Server: State Diagram

Event / Present State	Auth. Request	Auth. ACK	Auth. NACK	BAR	BAR ACK	BAR NACK	Router ACK	Router NACK	Client ACK	Time Expires	
Idle (0)	AuthVer			SLAVer						RmFilt	**Action**
	1			2						3	**New state**
Authentication Verify (1)		SACK	SNACK								**Action**
		0	0								**New state**
QoS Verify (2)					AddFilt	SNACK					**Action**
					3	4					**New state**
Router Config. (3)							SACK	SNACK			**Action**
							4	4			**New state**
Wait Client (4)									NA		**Action**
									0		**New state**

AuthVer - Verify the authenticity of the client; **SACK** - Send ACK to the client; **SNACK** - Send NACK to the client; **SLAVer** - Verify the availability of resources in the SLA of client; **AddFilt** - Add filter to the domain ingress router; **RmFilt** - Remove filter from the domain ingress router; **NA** - No Action.

Figure 8 - Negotiation Protocol of the Server: Transition Table

4. Prototype Implementation

In this section the prototype implementation will be presented. The objective is the validation of the architecture and protocol presented. The first of the described scenarios was chosen to be implemented because of its simplicity, making however, use of all the functionalities needed for the evaluation of proposal. Figure 9 shows the structure of the prototype system developed to support the evaluation of the proposals. The modules that compose the test-bed and the equipments are the described for the client, the server and the router.

The QoS Client and Authentication Client are implemented on a Linux RedHat 6.2 operating system, with support of Kerberos V5 (Authentication Client) and patched (2.2.17-05) to allow the interception of request made to the kernel. The implementation of the QoS Client is based on the DiffServ Daemon – (DSD) developed ate the University of Kansas [8]. The application *ttcp* modified to support of DiffServ (*ttcpds*) was also installed. This application is intended to generate test flows with QoS.

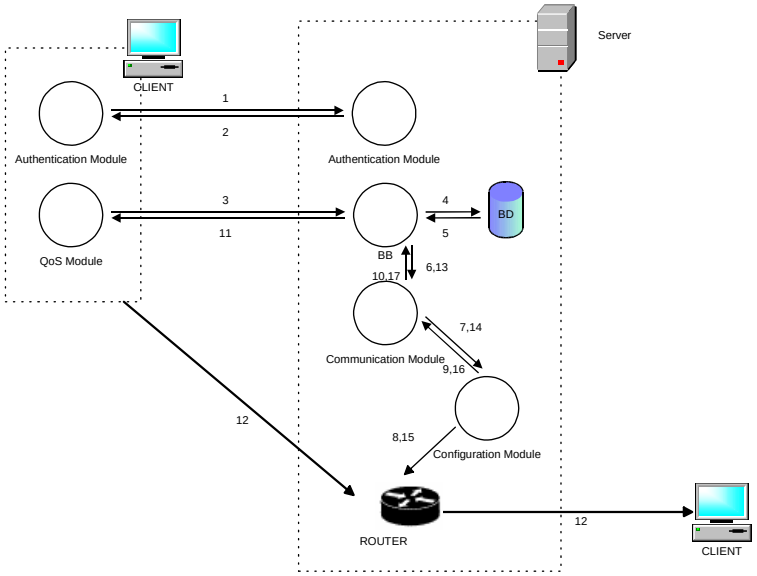

Figure 9 - Functional modules of the prototype

The QoS Server (QoS Management System of the ISP) is implemented in a single PC equipped with Linux RH 7.2 with support of DiffServ, Apache HTTPd with PHP, Database MySQL and Kerberos V5. The QoS Server is implemented in part with the software package from the University of the Kansas [8] together with Linux Traffic Control package [9] to allow the configuration of domain ingress routers.
The Router is responsible for packet classification, flow analysis, and for packet marking (or remarking) and treatment in accordance with the QoS agreement established during the authorization of the flow, as described in the previous subsection.

5. Validation

This section addresses the validation of the implemented prototype. The behaviour of the protocol is analysed to access its functionality. The description of protocol messages and events is made with reference to Figure 9.
To evaluate the behaviour of the prototype SLAs have been created to support QoS flows. The description of these SLAs is kept in the MySQL database (DB). When the Client wants to establish a QoS flow (using the application *ttcpds*) an authentication is needed in the Authentication Server to get the credentials (1) to communicate with the BB. After getting the credentials (2), the client will use the *ttcpds* application to specify the QoS parameters for the activation of the QoS flow previously authorized, with the following sintax:

ttcpds - t - q - TEF - S09 - P10 - R10 - b1 10.10.10.2

The flow is intercepted by the DSD *daemon* that opens a TCP socket and requests authorization in the BB for the establishment of the new flow, issuing the BAR massage (3). The BB will identify the client who originated the request consulting the SLA in the BD (4) (5), and verifies if the client still has available capacity to establish the flow. If the answer is positive, the BB will use the router communication module (6) that will send information to the router configuration module (7) that will configure the filtering rules of the router (8) to enable the establishment of the flow with the requested resources. The initial configuration state of the ingress router only allows the establishment of *Best Effort* flows. The described configuration rules are supported with the Linux Traffic Control that is also responsible for the policing and the marking / remarking of the flow packets. The configuration module of the router informs the communication module (9) that already carried out the configuration of the filtering rules and informs the BB (10).
After this, the BB authorizes the client (DSD) to establish the flow, provides the DSCP to mark the flow (11) and finally the flow is established (12). If the client doesn't have enough available resources, the flow is blocked by the DSD. If the requested parameters are exceeded, the behaviour of the ingress router will depend on the PHB of the flow. For the AF PHB exceeding packages will be remarked to *Best Effort*. For the EF PHB exceeding

traffic will be discarded. At the end of the time interval for which the reserve was established, the BB starts again the communication module of the router (13) that sends information to the router configuration module (14) to reconfigure the filtering rules of the ingress router (15) and subsequently the resources of the flow are released. The router configuration module informs the router communication module (16) that it already carried out the reconfiguration of the filtering rules and this in turn informs the BB (17).
Figure 10 shows a *screen shot* of the operation of the BB (a) and the router configuration system (b). These figures result from the answers of the BB and of the configuration module of the router, to a BAR of the client, whose reply was affirmative. After the duration of resource reservation, the connections are released by the reconfiguration of ingress router.

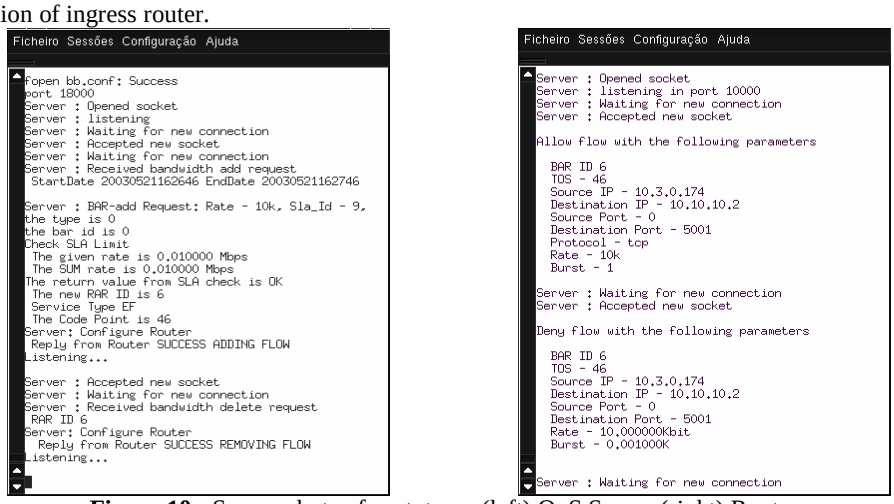

Figure 10 - Screen shots of prototype: (left) QoS Server (right) Router

5. Conclusions and Future Work

In this paper, an architecture for authentication and authorization of QoS flows establishment was proposed and discussed. The security aspects related to client authentication and flow authorization for access to the communication resources are taken into account. To integrate the architecture, a protocol for QoS negotiation was proposed and described.
Preliminary results obtained from a prototype implementation of the architecture and negotiation protocol show the potential of the solution for QoS negotiation with authentication of clients and authorization of QoS flow establishment.
Future work will focalize in more exhausting experimentation and evaluation of scalability and performance behaviour of the proposed solution. Relation with the accounting system will also be addressed.

Acknowledgements

This work was partially financed by the PRODEP program supported by the Portuguese Government and the European Union FSE Programme.

References

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

[2] K. Nichols, S. Blake, F. Baker, D. Black, *Definition of the Differentiated Services Fields (DS Fields) in the IPv4 and IPv6 Headers*, RFC 2474, IETF, December 1998.

[3] C. Rabadão, E. Monteiro, *Segurança e QoS no Modelo DiffServ (Security and QoS in the DiffServ Model)*, 5th Conference on Computer Networks (CRC2002), Faro, Portugal, University of Algarve, 26-27 September 2002.

[4] Zhi Fu, S. Felix Wu, T. S. Wu, He Huang, *Security Issues for Differentiated Service Framework*, Internet Draft, October 1999.

[5] S. Kent, R. Atkinson, *IP Encapsulating Security Payload (ESP)*, RFC 2406, IETF, November 1998.

[6] R. Atkinson, IP Authentication Header, RFC 1826, IETF, August 1995.

[7] A. Striegel, *Security Issues in a Differentiated Services Internet*, Proc. of Trusted Internet Workshop - HiPC, Bangalore, India, December. 2002.

[8] *Bandwidth Broker of the University of Kansas*, http://www.ittc.ku.edu/~kdrao/BB/

[9] B. Hubert et al, *Linux Advanced Routing & Traffic Control Howto*, Revision 1.3.2, LDP, March 2003.