Extending SIP for QoS support

D. Papalilo[1], S. Salsano[2], L. Veltri[3]

[1] Università di Roma "La Sapienza" (Italy)
[2] DIE – Università di Roma "Tor Vergata" (Italy)
[3] CoRiTeL – Via Anagnina 203, 00040 Roma (Italy)
e-mail: {papalilo,salsano,veltri}@coritel.it

Abstract. – **SIP is currently having a lot of attention as a protocol for session signaling over the Internet. It can cover voice, video and multimedia sessions. Most of these applications are sensitive to the QoS provided by the underlying IP network. Therefore a lot of interest is currently devoted to the interaction of SIP with the QoS mechanism in IP networks. This work will describe an enhancement to SIP protocol for the interworking with a QoS enabled IP network. The proposed mechanism is simple and it fully preserves backward compatibility and interoperability with current SIP applications. Moreover the paper describes the realization of the proposed solution in a testbed implementation.**

1. INTRODUCTION

Basically, SIP [1] is an end-to-end session setup protocol. In order to provide satisfying quality to audio and video communication services, the reservation of resources may be needed. In the current view, the SIP user agent should rely on existing QoS protocols (e.g. RSVP) for the support of resource reservation [2].

This fact has two main drawbacks: i) the user application must be aware of the QoS mechanism used in the access network and the relative QoS signaling protocol (e.g. RSVP, COPS, or other), ii) user application must implement such QoS protocol, with the increase of the client complexity. Moreover, if RSVP is used as signaling protocol, both user terminals should implement the RSVP protocol.

Currently two main approaches have been proposed in the IETF for the support of QoS in an IP network: the Integrated Services (Intserv) model (strictly based on the use of RSVP), and the Differentiated Services (Diffserv) model. A very good introduction to IP QoS topics can be found in [3], [4]. An IP telephony (SIP) architecture with end-to-end QoS support which can rely on the Intserv model is described in [2].

Although the Intserv model seems to be suitable for services that requires strict QoS guarantees, as for the IP telephony, it is more complex and suffers of scalability problems.

For this reason the Diffserv model is now obtaining a lot of interest within the IETF and, for the same reason, it has been chosen as QoS model in this work.

Figure 1 shows the reference scenario considered in this draft.

The SIP terminals are connected through access networks to a core network with QoS support. The QoS provided in the core network is accessed via some QoS Access Points at the border of such network. Without no loss of generality, we suppose that the QoS Access Points coincide with the network Edge Routers (ERs) (as in Figure 1). The QoS in the access networks depends on the QoS model used by the ISP for the access, but it is outside the scope of the mechanisms described in this document.

In this document we propose a very simple solution for QoS call setup that is based on the enhancement of the SIP protocol to convey end-to-end QoS related information. We will refer to such QoS aware SIP implementation as Q-SIP.

The proposed QoS architecture (see Figure 3) eliminates the need of QoS supports on the user terminals since all the QoS related functions can be moved to SIP servers that will control both call setup and resource reservation, thus relieving the terminals from unneeded complexity.

Basically, when a call setup is initiated, the caller SIP client can start a SIP call setup session through an outbound SIP proxy server. If needed, the server (a Q-SIP server) starts a QoS session interacting with a remote Q-SIP server and with the QoS provider (a QoS Access Point). When the QoS provider responds, the call setup can continue and finally the data session starts.

The requirements at the basis of the Q-SIP proposal are:

i) it should be possible to use existing SIP clients; no enhancements/modifications are needed in the SIP client applications,

ii) it should be possible to have a seamless interaction with other parties which do not intend or are not able to use QoS,

iii) the protocol enhancements should preserve backward compatibility with standardized SIP protocol,

iv) the resulting architecture should be as simple and scalable as possible.

The QoS setup procedure is dealt entirely by QoS aware agents, generally on SIP servers, and all protocol extensions needed for the QoS setup are hidden from not-QoS-aware SIP agents. Hence the solution preserves backward compatibility with current SIP applications and it de-couples as much as possible the SIP signaling from the handling of QoS.

Note that, it is reasonable that in a Diffserv QoS scenario there will be servers dedicated to policy control, accounting and billing aspects. A solution based on a SIP server is really suited to this QoS scenario.

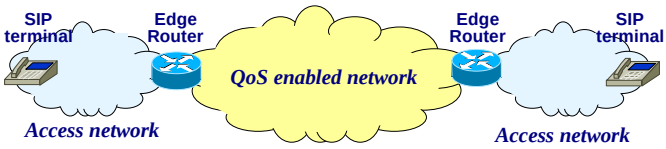

Figure 1 – Reference QoS scenario

2. SIP AND QOS: PREVIOUS SCENARIOS

Since the Integrated Services model can provide the stricter guarantees for bandwidth reservations for single flows, it seems to be a good candidate as reference model for QoS support for IP Telephony services. The interaction of the QoS signaling (i.e. the RSVP - Resource Reservation Protocol) and the SIP protocol has been already considered (see [2]).

The basic concept is to let the terminals start a RSVP based bandwidth reservation during the SIP call setup. The SIP user agents, that should "natively" include the RSVP support, are connected to an Integrated Services based IP network; all routers are Intserv aware and support RSVP, per flow bandwidth reservation and per flow packet scheduling. According to this scenario, when the calling User Agent wants to establish a QoS call, it sends the SIP INVITE message to the callee, specifying that a bandwidth reservation is requested.

Upon the receiving of the INVITE message, a 183 "session progress" response is sent from the callee and then the resource reservation procedure can start. Depending if the bandwidth reservation is requested for one or two-ways traffic flows, the caller or/and the callee starts a RSVP session by sending PATH messages to the peer party, followed by the classical RSVP signaling flow.

Upon the reception of the RESV messages each user agent realizes that the reservation has been successfully setup and the SIP call setup can continue with the 180 "ringing" message, the 200 OK and the caller ACK.

Figure 2 describes the SIP/RSVP signaling flow for the call setup between two SIP/RSVP aware user agents.

The main disadvantage of this scenario is that it suffers of the well known scalability problem of the Intserv approach ([5]).

This problem can be overcome by using a Differentiated Services approach. Different proposals are available in literature for QoS support through Diffserv networks [6]. The most common approach is to let RSVP signaling within access networks and to use Diffserv mechanisms within the core of the network. The bandwidth reservation is still requested by the terminals by means of RSVP signaling, but the resources in the core network are handled with Diffserv mechanisms.

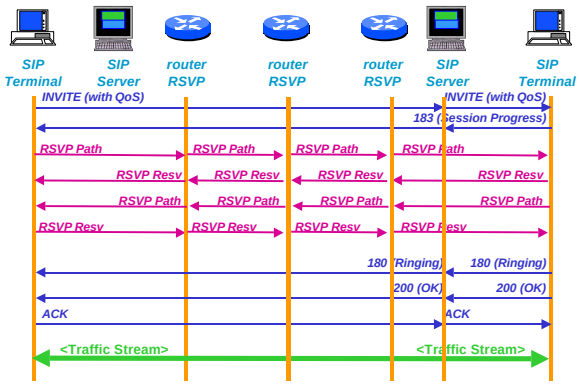

Figure 2 –The SIP/RSVP call setup signaling flow

This combination of the two approaches tries to benefit from both Intserv and Diffserv features. Scalability is achieved by the Diffserv aggregation in the core, while is kept the advantages of end-to-end signaling by the use of RSVP. Admission control within the Diffserv network is enforced at the edge of the network by the Edge Routers (ERs), (i.e. the routers placed at the boundary between the IntServ access networks and the Diffserv core network, while different architectures can be used to perform the resource management. A possible solution is to have a logically centralized entity (the Bandwidth Broker - BB) in charge of managing resources of the whole Diffserv network. The BB may act as an Admission Control Server. At the reservation setup time, the ingress Edge Router queries the BB and admits or rejects the new flow depending on its response.

Although this second architecture represents a significant step towards scalability, there are still some problems in common with the previous pure SIP/RSVP architecture. First, it requires ad-hoc user agents that are aware of both SIP and RSVP signaling; therefore no generic SIP client can be used. Second, the support of both SIP and RSVP protocols may be critical for very light terminals such as mobile phones small IP devices or other handheld IP based terminals, due to their limited memory and processor capacities. Third, a lot of signaling messages have to be sent and processed by terminals, servers and routers.

Moreover it is important to consider that the increased complexity of the SIP clients due to the support RSVP signaling in conjunction with the SIP signaling, is not justified by benefit derived by the use of RSVP. The objective of bandwidth reservation is usually the core network that, in this case, implements the Diffserv model, while the RSVP is used just as a generic protocol to ask QoS to the core network.

The above considerations are the basis of the proposed QoS architecture. The main idea is to eliminate the RSVP signaling from the terminals, and to use the SIP as unique call setup protocol for QoS (and not QoS) calls. An additional advantage is that all the QoS related functions can be moved to SIP proxy servers that will control both call setup and resource reservation, thus relieving the terminals from unneeded complexity.

3. QoS SIP: Overview

The basic idea is that SIP clients use a default SIP proxy server in their domains for both outgoing and incoming calls. The client sends SIP messages to its proxy server and receives the messages from its server. The SIP servers are therefore involved in the message exchange between the clients and can add (and read) QoS related information in the SIP messages. This QoS information exchange is made transparent to the clients. The SIP server will extract from SIP signaling QoS parameters among them and will interact with the network QoS mechanisms. The enhanced SIP server will be called Q-SIP server (QoS enabled SIP server) in the following.

The originating Q-SIP server adds QoS information in the SIP messages. This is meant as an offer to terminating SIP server, or as a hint that the originating side is capable of QoS and is willing to exploit it. If the terminating SIP server is able to handle QoS in a compatible way and it is willing to exploit it, it will answer positively with proper information in the response SIP messages. A legacy SIP server on the terminating side will not understand the QoS information in the SIP message and will silently ignore it. Obviously, the SIP session will be setup with no QoS.

The reference architecture for the proposed SIP QoS scenario is depicted in Figure 3. The involved actors are the two SIP clients, the two SIP servers and a QoS enabled network. The QoS provided by the QoS enabled network is accessed by QoS Access Points located at the border of the network in the ERs.

The setup of QoS session in such scenario is logically composed of two aspects: the end-to-end signaling mechanism to exchange QoS information and the QoS negotiation between the SIP agents and the QoS network.

In order to design a clean and flexible solution it is important to de-couple these two aspects as much as possible. Therefore the SIP

protocol mechanism to exchange QoS information should be generic and independent from the actual QoS mechanisms.

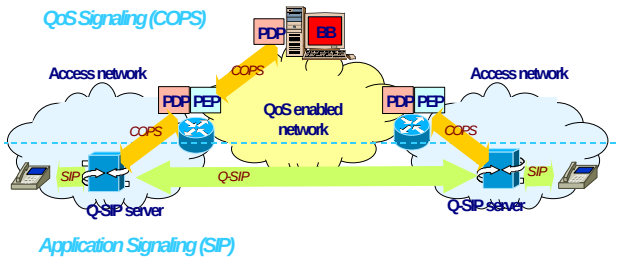

Figure 3 - Q-SIP architecture based on the use of Q-SIP servers

Although the proposed QoS architecture will be kept very general with respect to the used QoS mechanism, for completeness we will consider a particular scenario in which the QoS aspects in the Diffserv core network are dealt via the COPS protocol [7], with specific extension as proposed in [9].

An important assumption in our scenario is that unidirectional QoS reservations for IP flows are provided by the QoS enabled network. Therefore in order to setup a bidirectional QoS communication, two different reservations have to be requested to the QoS network (RSVP QoS model works in this way). Extensions to consider QoS network that can provide bi-directional reservation are currently under study.

Note that we mainly refer to a scenario where the SIP clients are un-aware of QoS aspects and the local SIP servers do all the QoS job. Actually, the proposed SIP QoS mechanism can be applied to a scenario where the SIP user applications are enhanced in order to handle the QoS aspects by themselves. The resulting scenario is depicted in Figure 4.

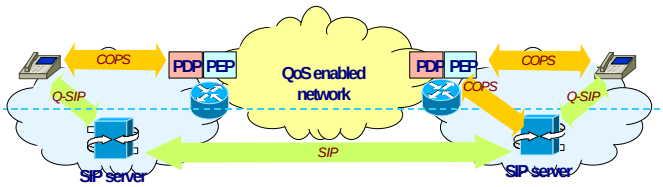

Figure 4 – Q-SIP architecture with Q-SIP agents on user terminals

Compared to Figure 3, note that SIP clients become Q-SIP clients and Q-SIP servers become SIP servers. There can even be asymmetric scenarios where one side is using a server and the other side uses a SIP application based solution.

4. Q-SIP SIGNALING MECHANISM

In this section we provide the detailed description of the signaling mechanisms of the proposed SIP based reservation architecture (Q-SIP) [10].

We consider a QoS scenario in which a Diffserv backbone network serves different access networks (Figure 1). The QoS requests are handled at the border of the core network by the QoS Access Points, that is, for simplicity, the Edge Routers. The ERs should implement all the mechanisms needed to perform admission control decisions (possibly with the aid of the BB) and policing function. The QoS scenario can be based on COPS as the protocol for QoS reservations.

The IP phones/terminals are located on the access networks; standard SIP clients can be used and explicit SIP proxying configuration is set. When a call setup is initiated, the caller SIP client starts a SIP call setup session through the SIP proxy server. If a Q-SIP server is encountered, this will start a QoS session interacting with a remote Q-SIP server and with the QoS provider for the backbone network (i.e. the access ER). Figure 3 shows the architecture.

4.1. Q-SIP message flow

With reference to Figure 3 and Figure 5, the call setup starts with a standard SIP INVITE message sent by the caller to the local Q-SIP server. The message carries the callee URI in the SIP header and the session specification within the body SDP (media, codecs, source ports, ecc). The Q-SIP server is seen by the caller as a standard SIP proxy server. The Q-SIP server, based on the caller id and on session information, decides whether a QoS session has to be started or not. If a QoS session has to be setup, it inserts the required descriptors within the INVITE message and forwards it towards the invited callee; the INVITE messages can be relayed by both standard SIP proxy servers and Q-SIP servers.

When the callee responds with a 200 OK message, it is passed back to the last Q-SIP server that is the Q-SIP server that controls the callee access network.

At this point the Q-SIP server on the callee side has all the information to request a specific QoS reservation to the ER on the callee access network for the callee-to-caller traffic flow. When the callee Q-SIP receive the response for the QoS reservation request, if it is positive, it stores such QoS information and send it within the 200 OK message toward the caller. The QoS information data should be stored by the Q-SIP server in order to maintain trace of the current QoS session (see also later); we call such data "QoS state". If the response for the reservation is negative, the Q-SIP server does not set a new QoS state, but it still inserts in the 200 OK response the fields needed for the caller-to-callee reservation in order to give the possibility to the caller Q-SIP server to make the reservation.

When the caller Q-SIP server receives the 200 OK message with the complete QoS session indicators, it completes the QoS session setup by performing the QoS request to the ER on the caller access network for the caller-to-callee traffic flow. If the response for this flow is negative, the caller-to-callee flow will not have QoS support. The handling of these reservations refusals is different depending on QoS service model (i.e. QoS-Assured or QoS-Enabled services see [2]). Assuming a QoS-Enabled service, the Q-SIP server will simply continue with the signaling.

When a call is terminated all resources that have been reserved must be released. This action is triggered by the BYE messages; when a BYE matching an installed QoS state is received, the Q-SIP server sends a release request to the QoS provider and removes the QoS state.

It is important to note that the proposed architecture keeps the compatibility with standard SIP clients and standard SIP servers. As we will see in the rest of this section, all the information needed by the Q-SIP servers to perform the QoS session setup is inserted within the SIP messages in such a way that non Q-SIP aware agents can transparently manage the messages.

4.2. Q-SIP protocol

When the first Q-SIP server (i.e. the caller Q-SIP server) is encountered, it inserts a new field in the SIP header that is:

```
CallerER: <caller ER address>
```

By means of the *CallerER* field the other Q-SIP servers know the IP address of the caller ER; this information is used by the callee Q-SIP server to specify the remote endpoint of the reservation in the reservation request to the QoS provider.

Moreover, the caller Q-SIP server add its *VIA* field (as every SIP proxy), in which it includes some specific information (considered as SIP "comments") that will be not visible to any other SIP or Q-SIP server, since they are within its own *VIA* field. This information will be used by the same Q-SIP server while processing the 200 OK responses. The *VIA* field is structured as follows:

```
VIA: SIP/2.0/udp <SIP server address>[:<port>]
(FirstQSIP/CallerER:<caller ER address>[/<next comment>])
```

Note that according to the standard SIP protocol processing rules each SIP proxy that manages the INVITE message adds a new *VIA* field; while all the other field, as the *CallerER* field should be forwarded. Each Q-SIP server that manages an INVITE message containing the *CallerER* field, will also copy the caller ER address within its *VIA* field, as follows:

```
VIA: SIP/2.0/udp <SIP server address>[:<port>]
(CallerER:<caller ER address>[/<next comment>])
```

When the INVITE message reaches the callee host, the user client processes the call, and, at last, generates the 200 OK response (if the call is accepted).

If the client is not aware of Q-SIP it simply discards each Q-SIP field (i.e. the *CallerER*) when forming the new response message. According to the SIP protocol, the fields that it has to copy from the INVITE message are the *Via, To, From, CSeq and Call-ID* fields.

When the 200 OK reaches the callee Q-SIP server, the corresponding *VIA* field is read, the QoS session information are extracted (including the caller ER address) and a QoS request for the IP flow in the callee-to-caller direction can be started. (As we are considering only unidirectional reservations, two reservations in the two directions are needed)When this QoS reservation request/response phase is concluded and the resource is reserved, the QoS state is stored and the 200 OK messages is relayed toward the caller.

Within this new response message, the corresponding *VIA* field is dropped (as required by SIP) and a new field specifying the callee ER address is inserted, that is:

```
CalleeER: <callee ER address>
```

Even if the QoS reservation for the callee-to-caller flow was not successful, this field is still inserted to make possible to reserve the QoS for the caller-to-callee flow in a "QoS-Enabled" scenario. For a "QoS-Assured" one a different behavior should be performed but this is outside the scope of this document.

If there are additional SIP servers handling this response in the path between the callee Q-SIP and the caller Q-SIP servers, they will only drop their own *VIA* field according to standard SIP rules. The Q-SIP servers recognize that they are not the callee Q-SIP server because the CalleeER field is already present in the message. When the first Q-SIP server is encountered (i.e. the caller Q-SIP server), it recognizes the field *FirstQSIP* within its *VIA* field and extracts the QoS session information (including the callee ER address). Then, it starts the QoS request for the IP flow in the caller-to-callee direction and stores the "QoS state" for this flow (if the reservation has success).

It is very important to note that the use of the previously defined *VIA* fields lets each Q-SIP server extract all information needed for the QoS reservation directly from the SIP message that it is processing. This mechanism allows the Q-SIP not to keep per session information until a QoS call is completely installed and can be used in light Q-SIP implementations.

This "QoS state" is instead needed when the call setup is completed for a correct tear-down procedure, for accounting and for resource control.

In the Q-SIP, a key rule is played by the capacity of the Q-SIP servers (both the caller and the callee servers) to gather the necessary information from SIP messages in order to select the appropriate QoS reservation. Particularly the Q-SIP servers have to specify the bandwidth parameters and the flow characterization parameters (i.e. for traffic policing) in the QoS reservation request messages. The Q-SIP servers have to select the appropriate level of bandwidth, the ingress and egress ERs, and the session identification parameters (i.e. the socket identifiers). Let us now consider how the Q-SIP servers can obtain this information.

As for the bandwidth that has to be requested to the QoS provider, this is selected on the base of the type of codecs specified by the end clients for the RTP streaming traffic, and found within the SIP INVITE and 200 OK messages. In Appendix B, it is reported an example of mapping table that can be used to derive the required bandwidth for well known audio codecs. It reports both the payload bit rates and the required bandwidths (taking into account the IPv4 and IPv6 headers).

As for the session identification, in general different filters can be used. For example, RSVP defines for basic flow filter the destination IP address, the transport protocol identifier and (optionally) a transport address, i.e., in case of UDP/TCP, the destination port.

In our architecture we use a three-fields filter composed by the source address, the destination address and the destination port. This information can be extracted from the INVITE/200 OK messages directly by the caller/callee Q-SIP servers.

Note that the caller address and port information needed to setup the QoS for both directions are found within INVITE messages. The reservation is made by the caller and callee Q-SIP servers when they receive the 200 OK message. The mechanism, similarly to that used to take trace of ER information, uses a new field within the *VIA* field added during the processing of the INVITE message.

For the caller Q-SIP server the *VIA* field becomes:

```
VIA: SIP/2.0/udp <SIP server address>[:<port>]
(FirstQSIP/CallerER:<caller ER address>/Caller-<media>-
endpoint:<caller address>:<caller port>[/<next comment>])
```

Where the parameter <*media*> indicates which media uses the specified caller address; the address is used as source address for the caller-to-callee flow.

For the other Q-SIP server (hence also for the callee Q-SIP server) the *VIA* field becomes:

```
VIA: SIP/2.0/udp <SIP server address>[:<port>]
(CallerER:<caller ER address>/Caller-<media>-
endpoint:<caller address>:<caller port>[/<next comment>])
```

Where the parameter <*media*> indicates which media uses the specified caller address and port, as destination for the callee-to-caller flow.

The remaining information is extracted directly from the 200 OK message (the callee address from the callee Q-SIP server and the callee address and port from the caller Q-SIP server).

The Q-SIP call setup flow is shown in Figure 5.

The tear down procedure is triggered at the caller/callee Q-SIP servers by the receiving of the BYE and 200 OK messages. When a Q-SIP server receives the BYE request associated to a session with QoS, it requests the releasing of the bandwidth for that session to the QoS provider. If required, the resource details could be retrieved from the stored QoS state. In appendix A there is a sample of the QoS state that can be associated with each QoS call.

When a BYE request matches one of the stored call-leg, the Q-SIP server releases the resources by interacting with the QoS provider and frees the QoS state. If a BYE message gets lost due to a terminal failure, the session tear-down should be initiated (automatically) by the other SIP terminal as a result of a session time-out.

In order to ensure that the SIP signaling will cross the Q-SIP servers, the Record-Route and Route headers are used as defined by SIP [1]. The Q-SIP server inserts the Record-Route header for the sessions with QoS requests, making sure that further signaling will cross the Q-SIP server itself.

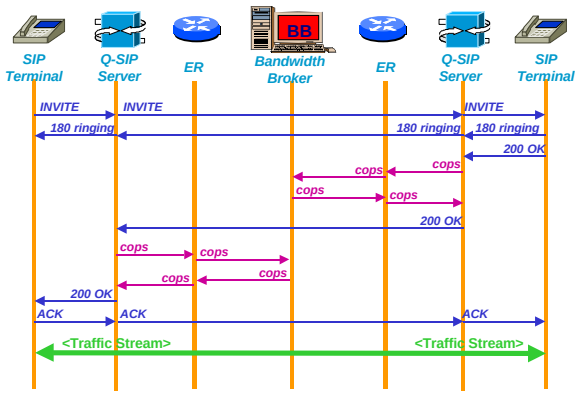

Figure 5 – Q-SIP call signaling flow

A set of example Q-SIP messages is reported in Appendix C.

5. SIP TERMINALS

Although it has been supposed that the SIP user clients are not aware of the Q-SIP reservation mechanism, Q-SIP aware clients can be also considered (Figure 4).

Q-SIP aware clients should simply include Q-SIP as described in the previous sections. In that case, the clients could directly request QoS reservation to the QoS providers and the Q-SIP signaling would transparently bypass any SIP or Q-SIP proxy server. Moreover the architecture is fully compatible also for calls starting from Q-SIP aware clients and directed to standard SIP clients with Q-SIP proxy servers, and vice-versa.

6. Q-SIP SERVERS

A basic design choice in the design of a SIP proxy server is whether to make it stateful or stateless. Being stateful means that it keeps a record of active SIP session and the processing of SIP messages can depend on the session status. Being stateless means that each message is processed by itself with no relations with previous messages of the same session. A stateful server of course is more powerful as it can better handle additional aspects (like for example policy and accounting), but the SIP protocol has been designed so that stateless server can work as well.

Looking at our approach, we note that the Q-SIP server handles the QoS for a SIP session, by making a reservation in the QoS enabled network. The Q-SIP server has to care about this reservation, for example the resources must be properly released when the session is closed. For this reason we believe that the Q-SIP server must be stateful once the session has been established.

Nevertheless, we have designed our Q-SIP extensions preserving the SIP design goals: there is no need to store state information during the session establishment and all the needed information is carried by the SIP messages itself.

7. IMPLEMENTATION OF THE Q-SIP PROXY SERVER

The proposed architecture has been implemented in a test-bed compposed of a set of Linux PCs. The Diffserv components of the test-bed have already been discussed in [11]. The overall picture of the test-bed is described in Figure 6.

Here we focus on the design of the Q-SIP proxy server and on its interaction with the COPS protocol for admission control.

A SIP proxy server has basically two (related) tasks: 1) to receive message destined to a client which is registered and to forward them to the client itself; 2) to receive messages coming from a user in its domain and to propagate them outbound. A basic design choice in the design of a SIP proxy server is whether to make it stateful or stateless. Being stateful means that it keeps a record of active SIP session and the processing of SIP message can depend on the session status. Being stateless means that each message is processed by itself with no relations with previous session messages (only the registration status of clients is taken into account). A stateful server of course is more powerful as it can better handle additional aspects (like for example usage accounting). In our implementation we chose to design a quasi-stateless server that takes trace only of already established QoS sessions; no information of ongoing call setup sessions or of non-QoS sessions is kept. Note that such server imposes more stringent requirements on the correctness of SIP protocol mechanism, since it is not possible to rely on information previously exchanged during the SIP session. In our test-bed we demonstrated that the proposed SIP enhancement works with such a quasi-stateless server, hence verifying the robustness of the protocol. A real life implementation could be based on a stateful server.

The Q-SIP proxy server has been realized on a PC running the Linux Redhat 6.2 operating system. The Q-SIP server is developed in Java (running on Sun JDK 1.2.2 virtual machine) and a COPS DRA client and server are developed in C. The internal architecture of the test bed elements is shown in Figure 7.

The SIP server and the COPS DRA client are two different Unix processes communicating through a socket interface. The Edge Routers, that act as QoS Access Points, include a COPS DRA server that communicate through a socket interface with a process implementing the Local Decision Server and the COPS DRA client. This process communicates with the Diffserv traffic control mechanism provided by the Linux kernel. The PDP/BB is composed by a COPS DRA server and a Decision Server, that interact through a socket based interface.

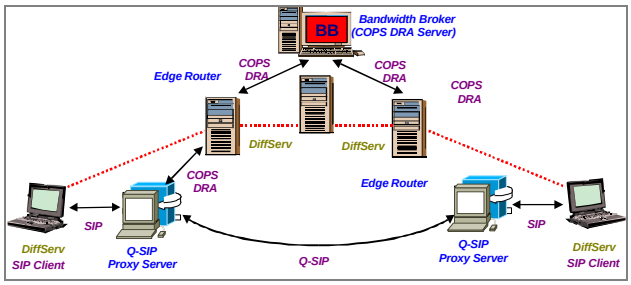

Figure 6 – Overall test bed scenario

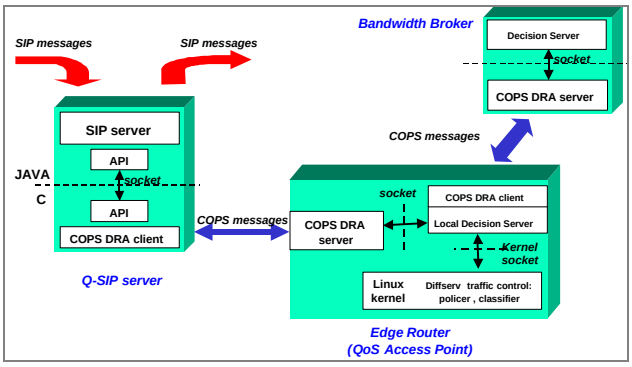

Figure 7 – Q-SIP server, ER, and BB internal architectures

8. CONCLUSIONS

In this paper we propose some simple extensions of the SIP protocol in order to interact with QoS mechanisms for QoS support in Diffserv networks. The enhancement to the SIP protocol is basically independent from the specific QoS scenario. A possible deployment scenario based on Q-SIP proxy servers is proposed, having the advantage that "legacy" SIP user application can be fully reused. The solution is also fully backward compatible with current SIP based equipment that does not support QoS, allowing a smooth migration. Finally the testbed implementation of the proposed solution, including the internal architecture of the Q-SIP proxy server has been described.

9. REFERENCES

[1] M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg, " SIP: Session Initiation Protocol", IETF Internet Drafts < draft-ietf-sip-rfc2543bis-02.txt>, November 2000.

[2] W. Marshall et al. "Integration of Resource Management and SIP", IETF Internet Draft <draft-ietf-sip-manyfolks-resource-02.txt>, August 2001, Work in Progress.

[3] X. Xiao, L.M. Ni "Internet QoS: A Big Picture", IEEE Networks, March 1999

[4] W. Zhao, D. Olshefski and H. Schulzrinne, "Internet Quality of Service: an Overview" Columbia University, New York, New York, Technical Report CUCS-003-00, Feb. 2000.

[5] A. Detti, M. Listanti, S. Salsano, L. Veltri, "Supporting RSVP in a Differentiated Service Domain: an Architectural Framework and a Scalability Analysis", IEEE International Communication Conference (ICC'99), June 1999, Vancouver, pp. 204-210.

[6] Y. Bernet, R. Yavatkar, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wrocklaski, E. Felstaine, "A Framework for Integrated Services Operation Over Diffserv Networks", IETF RFC 2998, November 2000.

[7] D. Durham, Ed., J. Boyle, R. Cohen, S. Herzog, R. Rajan, A. Sastry, The COPS (Common Open Policy Service) Protocol, IETF RFC 2748, January 2000.

[8] R. Mameli, S. Salsano "Use of COPS for Intserv operations over Diffserv: Architectural issues, Protocol design and Test-bed implementation", ICC2001, Helsinki

[9] S. Salsano "COPS usage for Diffserv Resource Allocation (COPS-DRA)", <draft-salsano-cops-dra-00.txt>, September 2001, Work in Progress, http://www.coritel.it/projects/cops-bb [Note to the reviewer: the draft will be submitted in September, but it is already available at the given URL]

[10] L. Veltri, S. Salsano, " SIP Extensions for QoS support in Diffserv Networks", <draft-veltri-sip-qsip-00.txt>, October 2001, Work in Progress, http://www.coritel.it/projects/qsip

[11] W.Almesberger, S.Giordano, R. Mameli, S. Salsano, F.Salvatore "A prototype implementation for Intserv operation over Diffserv Networks", IEEE Globecom 2000, S. Francisco, December 2000.