QoS Support for SIP Based Applications in a Diffserv Networks

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

[1] CNIT - Italian National Consortium for Telecommunications
[2] DIE – Università di Roma "Tor Vergata" (Italy)
[3] University of Parma (Italy)
e-mail: stefano.salsano@uniroma2.it, luca.veltri@unipr.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 application of this mechanism to a particular QoS enabled IP network, which implements Diffserv as transport mechanisms and COPS for resource admission control. A testbed implementation of the proposed solutions is finally described.**

1. INTRODUCTION

The SIP protocol ([1]) is used to initiate voice, video and multimedia sessions, for both interactive applications (e.g. an IP phone call or a videoconference) and not interactive ones (e.g. a Video Streaming). SIP is currently having a lot of attention within IETF and it is seems to be the more promising candidate as call setup signaling for the present day and future IP based telephony services. There are several available user terminal applications (SIP User Agents) supporting SIP on various platforms.

The IP telephony service seems to be the killer application for SIP protocol. In the near future, SIP based IP telephony will be applied in large-scale (intra) networks and in the long term it could even be a real competitor to the Plain Old Telephone Service. For the realization of these scenarios, there is the obvious need to provide a good speech quality. This quality in turn depends on the Quality of Service delivered by the IP network. Reservation and/or admission control mechanisms could be needed to get QoS from the IP network. Unfortunately, at present day, there is not a clear picture about the "elected" mechanism for QoS provisioning in IP network, as much research and standardization effort is ongoing in this area. The interaction of these QoS mechanisms with the call setup procedures (i.e. SIP) is therefore a very hot topic.

Looking at the standardization effort in the area of IP QoS the two main approaches that have been proposed in the IETF are the Integrated Services (Intserv) model and the Differentiated Services (Diffserv) model. Additional proposals consider a combination of the two approaches. In Addition, the MPLS technology is going to play an important role in this field, for example as transport backbone for Diffserv. A very good introduction to IP QoS topics can be found in [2], [3].

Currently, different scenarios have been proposed in order to bind the SIP signaling to the various IP QoS mechanisms. After a brief overview of these proposed approaches, in this work we propose a very simple solution that is based on an enhancement of the SIP protocol to convey QoS related information. 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.

The current proposals for the integration of the SIP based call setup (especially for IP Telephony services) and the bandwidth reservation procedures for IP QoS are dealt with in section 2. Section 3 describes the rationale and overall requirements related to our proposed solution, which is specified in section 4. Moreover, section 5 describes the interaction with a specific QoS mechanism based on COPS for admission control in a Diffserv network. As we have implemented and tested the proposed solution, section 6 describes the design of a SIP proxy server compliant to our enhanced SIP signaling and shows the testbed scenario.

2. SIP AND QOS: CURRENT SCENARIO

An Internet Draft of the SIP WG [4] deals with the interaction between SIP and resource management for QoS. It covers the problem of how to setup QoS before alerting the user, in order to avoid that the user answer the call but resources are not available. This is called "QoS assured" model: the session should not be established if resources are not available.

According to [4] the SDP (Session Description Protocol) must be enhanced to convey QoS related info and a new SIP message (*UPDATE*) is needed. This process requires the cooperation of the SIP User Agents (i.e. the terminals) that should be enhanced with respect to standard SIP.

The setup of the QoS reservation is described in [4] as a process originated by the user terminal. The model is not restricted to RSVP, but it was originally designed with RSVP in mind and the examples are based on this reservation protocol. In fact, despite the current effort in NSIS WG group, RSVP is the only standard protocol to setup reservations starting from the terminals.

Figure 1 describes an example of signaling flow for the call setup between two SIP/RSVP aware user agents. The basic concept is that the terminals start a RSVP based QoS reservation during the SIP call setup. The SIP user agents, that should "natively" include the RSVP support, must be connected to routers that support RSVP. 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 in the SDP that a QoS reservation is requested. Upon the receiving of the *INVITE* message, the callee sends a 183 "session progress" response and then the resource reservation procedure can start. Depending if the QoS 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. The SIP call setup can continue with sending of the newly proposed *UPDATE* method, the user is alerted and the *180 Ringing* message, the *200 OK* and the caller *ACK* messages are exchanged.

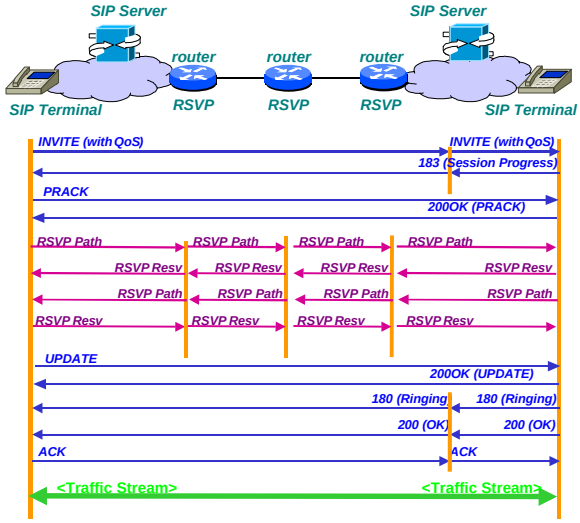

Figure 1 –The SIP/RSVP call setup signaling flow

A potential problem with this approach is that the resource reservation is handled by the terminal, which should be aware of the QoS model of the access network[1]. The complexity of the terminal increases and this could 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. The solution is not so flexible, because all QoS models in the access network should be equal, or the terminal has to support all the possible QoS models. In order to overcome this problem we propose that QoS is handled by the server and that the terminal does not need to start reservations. The main idea is to eliminate the need of QoS signaling from the terminals, and to use the SIP as unique call setup protocol for QoS (and not QoS) calls. The QoS related functions are moved to SIP proxy servers that will control both call setup and resource reservation, thus relieving the terminals from unneeded complexity. We consider a Diffserv based QoS scenario.

Actually, our proposal starts from the analysis of a "QoS enabled" model, where the reservation of resources is not a mandatory precondition and can be executed in parallel with normal session setup. Then, if a QoS assured model is needed, we analyzed the interaction with the signaling related to preconditions described in [4].

3. PROPOSED SIP BASED SCENARIO

In the definition of the proposed solution for Diffserv QoS support in SIP, we started from the following basic requirements:

- it should be possible to use existing SIP clients; no enhancements or modifications are needed in the SIP client applications;
- it should be possible to have a seamless interaction with other parties which do not intend or are not able to use QoS;
- the protocol enhancements should preserve backward compatibility with standardized SIP protocol.

[1] In the above case the QoS model is Intserv with its RSVP protocol, another simpler model is over-provisioning, another one is static Diffserv configuration… other models could require different signaling protocols

The solution foresees the enhancement of the SIP proxy server to handle QoS aspects. In the following, the enhanced SIP server will be called Q-SIP server (QoS enabled SIP server). All the QoS aspects can be covered by the Q-SIP servers in the originating and terminating sides. This is also justified by the fact that in a Diffserv QoS scenario there will be servers dedicated to policy control, accounting and billing aspects. Hence, a solution based on a SIP server is really suited to this QoS scenario.

The basic idea is that SIP clients use a default SIP proxy server in for both outgoing and incoming calls. Note that in case of user/terminal mobility this proxy server is not the "home" server of the user, but a server in the "visited" domain. Therefore this proxy server will be physically close to the user and will probably interact with the network elements that control the communications to and from the domain (e.g. firewall, NAT). The client sends SIP messages to this proxy server and receives the messages from it. 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 transparently to the clients and to non QoS-aware SIP servers, in order to preserve full backward compatibility.

Using the Q-SIP protocol extensions, the involved Q-SIP servers will exchange information needed to setup the QoS reservation (for example they will agree on the endpoint of the reservation). The Q-SIP servers will then interact with the network QoS mechanisms. When a new call setup is started, the originating Q-SIP server adds QoS information in the SIP messages. This is meant as an offer to terminating SIP server, i.e. 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 scenario for a successful SIP QoS scenario is depicted in Figure 2. The involved actors are the two SIP clients, the two Q-SIP servers and a QoS enabled network with its Admission Control entities located in the Edge Routers. In case of a mobility scenario, at least the "home" SIP server of the called user will be involved in the SIP call, but this is neglected for simplicity.

The setup of a QoS session in such a scenario is logically composed of two aspects. First aspect is the signaling and negotiation of QoS between the calling and called side, as represented by the horizontal arrow in Figure 2. The second aspect is the QoS negotiation between the QoS users (the SIP servers) and the QoS provider (the QoS enabled network).

In order to design a clean and flexible solution it is important to de-couple these two aspects as much as possible. The SIP protocol mechanism to exchange QoS information should be generic and independent from the actual QoS mechanisms. Then this mechanism can be particularized for a specific QoS mechanism by defining specific information elements. Specific QoS mechanisms that can be used are for example the ones based on RSVP, or on aggregated RSVP, on Diffserv with static reservation and Policy control, on Diffserv with Bandwidth Brokers and so on.

In this work we define the generic mechanism in the SIP protocol and we present as an example of the interaction with a specific QoS mechanism based on COPS [5] for admission control in a Diffserv network. Further details on the COPS based mechanism, first

proposed in [6], are given in [7] and [8]. The basic idea is that Admission Control entities running on the network borders (e.g. in the Edge Routers) dialogues with external QoS clients (the Q-SIP servers) and with Bandwidth Broker in the QoS enabled network. The Admission Control entities use a variant of the COPS protocol, called COPS-DRA (Diffserv Resource Allocation) to dialogue with the QoS clients and with the BB.

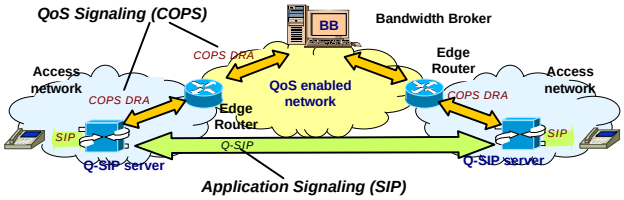

Figure 2 - Reference scenario for the proposed SIP QoS architecture

The scenario described in this paper and the implemented prototypes are based on the assumption that SIP client applications are not modified and the SIP servers do all the QoS job. This ensures backward compatibility with existing SIP clients. However, the proposed SIP QoS mechanism can be also realized by enhancing the SIP user applications in order to handle the QoS aspects directly in the end systems (the user terminals). 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

This section describes the signaling mechanisms used by the proposed SIP based reservation architecture (Q-SIP); further details are given in [9].

In the considered QoS scenario, a Diffserv backbone network serves different access networks; the QoS requests are handled at the border of the core network by the ERs that implement all mechanisms needed to perform admission control decisions (possibly with the aid of other entities) and policing function.

The IP phones/terminals are located on the access networks; standard SIP clients can be used, set with an explicit SIP proxying configuration. When a call setup is initiated, the caller SIP client starts a SIP call session through the SIP proxy server. If a Q-SIP server is encountered, this can start a QoS session interacting with a remote Q-SIP server and with the QoS providers for the backbone network (i.e. the access ERs). Figure 2 shows the reference architecture.

According to the direction of the call, the two Q-SIP servers are named caller-side Q-SIP server and callee-side Q-SIP server.

As far as the reservation procedure is concerned, two different models are possible: i) unidirectional reservations and ii) bidirectional reservations. In the unidirectional reservation mode, the caller-side Q-SIP server makes reservation for the caller-to-callee traffic flow, while the callee-side Q-SIP server reserves resources for the callee-to-caller flow; two reservations are hence needed for bidirectional flows. Instead, in the bidirectional reservation mode, it is the caller-side Q-SIP server that performs resource reservation for both directions. The choice between the two models can be done on the basis of a pre-configured mode or through the exchange of specific parameters (*qos-mode* parameters) between the Q-SIP servers during the call setup phase.

With reference to Figure 2 (see also Figure 3), the call setup starts with a standard SIP *INVITE* message sent by the caller to the local Q-SIP server (i.e. caller-side 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 is required/opportune, the server inserts the necessary descriptors (see later) within the *INVITE* message and forwards it towards the callee. The *INVITE* messages can be relayed by both standard SIP proxy servers and Q-SIP servers until they reach the callee-side Q-SIP server and then the invited callee. When the callee responds with a *200 OK* message, it is passed back to the callee-side Q-SIP server. At this point, if bidirectional reservation model is implemented, the callee-side Q-SIP server can request a QoS reservation to the ER on the callee access network (i.e. the QoS provider for the callee). If a unidirectional reservation model is considered for this call, the QoS reservation is demanded to the caller-side Q-SIP server. In both cases, the *200 OK* response, opportunely extended by the callee-side Q-SIP server, is forwarded back to the caller, via standard SIP servers and via the caller-side Q-SIP server.

When the caller-side Q-SIP server receives the *200 OK* message, it performs QoS reservation with the ER on the caller access network (i.e. the QoS provider for the caller). Depending on the reservation model, the caller-side Q-SIP server request QoS for the caller-to-callee traffic flow (unidirectional model) or for both directions (bidirectional model).

It is important to note that the proposed architecture keeps the compatibility with standard SIP clients and standard SIP servers. 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.

Regarding the management of QoS SIP sessions within Q-SIP servers, two different approaches have been considered and supported:

i) the Q-SIP servers maintain a provisional QoS state during the session setup (stateful Q-SIP),

ii) the Q-SIP servers are stateless respect to the QoS sessions during the session setups (stateless Q-SIP).

The latter approach will lead to a lighter server implementation, but more information has to be carried in the SIP messages. Considering that it is reasonable that a Q-SIP server will be stateful after that the session is setup (to keep track of QoS reservations), we think that the stateful version can be preferred.

4.1. Stateful Q-SIP: protocol extensions

When the first Q-SIP server (i.e. the caller-side Q-SIP server) receives a new *INVITE* message (referring to a new SIP session), it inserts the following new header:

```
QoS-Info: <qos-param> *(;<qos-param>)
```

Wherein <qos-param> can be some of:

```
<qos-mode> | <er-ingress> | <er-egress> | <qos-domain> | <other>
```

An example of *QoS-Info* Header inserted by the caller-side Q-SIP server could be:

```
QoS-Info: qos-mode=unidirectional;
          er-ingress=193.205.242.7;
          qos-domain=wonderland.net
```

By means of the *er-ingress* parameter the caller-side Q-SIP servers informs the callee-side Q-SIP server about the IP address of

the caller ER; this information is used by the callee-side Q-SIP server to specify the remote endpoint of the reservation in the reservation request to the QoS provider. The *INVITE* message is then forwarded by the caller-side Q-SIP server toward the invited callee.

A Q-SIP server that receives that message recognizes that the message is for a QoS session and, according to a stateful Q-SIP implementation, it may also decide to maintain a per-session provisional QoS state. The last Q-SIP server that stores QoS for that request message will play as callee-side Q-SIP server.

When the *INVITE* message reaches the invited callee, the user client processes the call and if the call is accepted, generates a *200 OK* response. All the Q-SIP headers are simply discarded by the client.

When the *200 OK* reaches the callee-side Q-SIP server, the server associates the response to the previously stored provisional QoS state. In case of unidirectional reservations, it starts the QoS reservation with the QoS provider (i.e. the egress ER). In order to make the QoS request, it needs to retrieve some information (i.e. ingress ER address, media port) from the stored provisional QoS info. When this QoS reservation request/response phase is concluded, the *200 OK* messages is opportunely extended with a new *QoS-Info* header as follows:

```
QoS-Info: qos-mode=unidirectional;
          er-egress=193.205.80.33;
          qos-domain=wonderland.net
```

If case of bidirectional reservations, the callee-side Q-SIP server will not start any QoS reservation and will forward the *200 OK* message including the *QoS-Info* header as shown above, where obviously *qos-mode=bidirectional*.

If there are additional SIP servers handling this response in the path between the callee-side Q-SIP and caller-side Q-SIP servers, they will process it according to standard SIP rules. If they had previously stored some QoS information for that session, they simply remove it. When the message reaches the caller-side Q-SIP server, it associates the message to the stored provisional QoS state and retrieves has all the information to start a QoS reservation (uni- or bi-directional) with the local QoS provider (the ingress ER). Finally, the SIP response is forwarded to the caller.

In the Q-SIP mechanism, 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/QoS parameters and the flow characterization parameters (i.e. for traffic policing) for the QoS reservation requests. The Q-SIP servers have to select the appropriate level of bandwidth or service classes, the ingress and egress ERs, and the session identification parameters (i.e. the port number to identify the media flows). This information can be obtained by the Q-SIP directly from the incoming SIP messages.

As for the bandwidth or service class that has to be specified to the QoS provider, this is selected on the basis of the type of media and codecs specified by the end clients (within the SDP body) and/or according to the particular user profile. For example for most audio codecs it can be relatively easy to prepare a mapping table (see [9]) of codecs and required bandwidths, for both RTP streams. For video codecs this is not so simple therefore one could have to rely on user profiles.

As for the session identification, in general different filters can be used. For example, in RSVP the flow filter can include source and destination IP address, the transport protocol identifier and source and destination transport address (i.e. UDP/TCP port number). In our architecture we use a three-fields filter composed by the source IP address, the destination IP address and the destination port. This information can be extracted from the *INVITE/200 OK* messages directly by the caller/callee Q-SIP servers.

The tear down procedure is triggered by the reception of the *BYE* and *200 OK* messages at the caller/callee-side Q-SIP servers. When a Q-SIP server receives the *BYE* request and *200 OK* response associated to a session with QoS, it requests the release of the reservation for that session to the QoS provider.

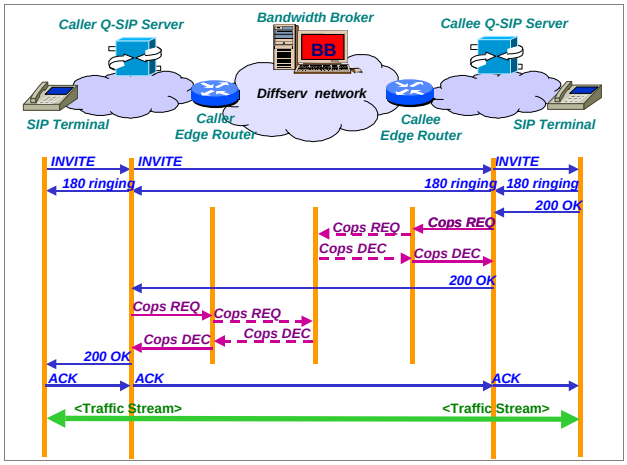

Figure 3 – Q-SIP call signaling flow - QoS enabled model

4.2. Stateless Q-SIP: protocol extension

Let us consider a version of the Q-SIP server that can be stateless during the call setups. We need a mechanism so that a Q-SIP server that receives a *200 OK* message can have all the information needed to setup a reservation, like ingress and egress routers of reservation, IP address and port of media flows. If the Q-SIP server does not want to keep any state, some information must be included in the SIP messages. Note that, in order to allow for QoS-unaware SIP clients, it is not possible to rely on SIP clients to copy the information contained in new headers on the response messages. The information should be inserted and maintained transparently for the clients. For such objective, the Q-SIP uses the Route/Record-Route SIP mechanism.

When the caller-side Q-SIP receives an *INVITE* request, it appends its Record-Route header with the following Q-SIP extension:

```
Record-Route: <server-uri>;
              <qos-info>*(;<param>)
```

Wherein <qos-info> can be of the form of:

```
<qos-param> *(;< qos-param >)
```

Wherein <qos-param> can be some of:

```
<qos-mode> | <er-ingress> | <er-egress> | <qos-domain>
| <media-spec>
```

With <media-spec> :

```
<media-type>:<ip-addr>:<port>
```

According to the SIP specification, the *Record-Route* information is returned opaquely by the called client within the response messages. Such functionality allows the Q-SIP server to "store" the QoS information in the *Record-Route* header and to obtain it back

in the response messages. Note that, although some information is redundant, we assume that also the full *QoS-Info* header is inserted by the Q-SIP server besides the *Record-Route* QoS parameters.

4.3. QoS-Assured and QoS Enabled models

As pointed out in section 2, the proposed architecture supports two different QoS models: QoS assured, and QoS enabled.

According to the "QoS assured" model, a call can be established only if the requested/required QoS can be set; in other words the QoS setup becomes a "precondition" for calls ([4]). On the other hand, in a "QoS enabled" model, the availability of QoS resources does not affect the success of a call; it only affects the effective level of QoS associated to the call. The Q-SIP can be implement according to both models.

Particularly, the Q-SIP protocol implements naturally the "QoS enabled" model, since the clients are completely not affected by the QoS related mechanisms. When the caller-side Q-SIP server (and in a bidirectional model the callee-side Q-SIP server) receives the *200 OK* SIP message, it starts a QoS reservation procedure; in both cases in which the QoS reservation succeeds or not, the *200 OK* response is still forwarded to the caller. The Q-SIP server may optionally signal the result of such reservation by including the QoS result parameter within the qos-infos (*Record-Route* and/or *QoS-Info* headers):

```
<qos-result>=<bandwidth>|<qos-class>|<yes>|<no>
```

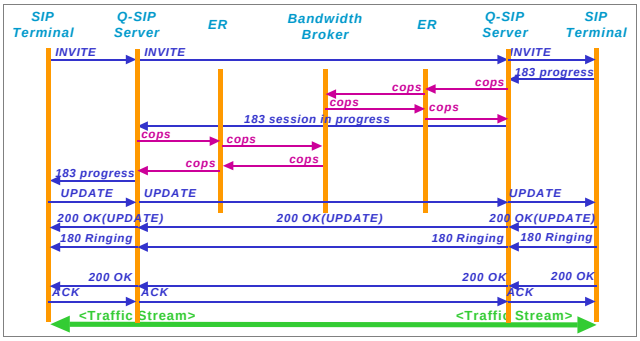

Figure 4 – Q-SIP call signaling flow - QoS assured model

In a "QoS assured" Q-SIP implementation, a QoS precondition verification phase should be executed before the call proceeds with a *180 Ringing* and *200 OK* messages ([4]). The most important difference between the Q-SIP procedure and that described in [4] is that all QoS related information are now exchanged only between Q-SIP servers. When the callee receive the *INVITE* message carrying precondition specification, it confirms it by sending a *183 Session in progress*. When the callee-side Q-SIP server receives the *183 response* message it starts the QoS reservation procedure (in case of bidirectional reservation model) and, if it succeeds, it forwards the *183* message toward the caller-side Q-SIP server. The caller-side Q-SIP server starts the reservation procedure as well and, if the reservation is successful, *it forwards the 183* response. Only if both the reservations succeed, the caller receives the *183* response and sends immediately an *UPDATE* request that let the call proceed normally as described in [4] and shown in Figure 4.

If one of the reservations fails, an error message is generated in place of the *183* by the Q-SIP server and the call is aborted by the caller with a *CANCEL* message. In case it is the callee that starts the precondition request (in the 183 response), the caller sends a PRACK request ([4]) and it is the *200 OK(PRACK)* sent by the callee that trigger the QoS reservation by the caller/callee Q-SIP servers.

Note that also in this "QoS assured" case, although the caller and callee are required to participate in the precondition verification mechanism, they are not involved in the QoS reservation procedure.

5. COPS-DRA SIGNALLING

The COPS (Common Open Policy Service) protocol is a simple query and response protocol that allows policy servers (PDPs) to communicate policy decisions to network devices (PEP). "Request" messages (REQ) are sent by the PEP to the PDP and "Decision" (DEC) messages are sent by the PDP to the PEP. In order to be flexible, the COPS protocol has been designed to support multiple types of policy clients. We have defined the COPS-DRA client type to support dynamic resource allocation in a Diffserv network. With reference to Figure 2, the COPS-DRA protocol is used on two different interfaces. First, it is used as a generic signaling mechanism between the user of a QoS enabled network and the QoS provider. In our case the Q-SIP proxy server plays the role of QoS user and will implement a COPS-DRA client, while the Edge Router plays the role of QoS provider and will implement a COPS-DRA server. On this interface, COPS-DRA provides the means to transport: the scope and amount of reservation, the type of requested service and the flow identification. The second interface where the COPS-DRA is applied is between the Edge Router and the Bandwidth Broker, in order to perform the resource allocation procedures. A flexible and scalable model for resource allocation is implemented. A set of resources can be allocated in advance by the Bandwidth Broker to the Edge Router in order to accommodate future request (according to the so-called COPS *provisioning* model). The amount of this "aggregated" allocation can also be modified with time. Moreover, specific requests can be sent by the Edge Router to allocate resources for a given flow (according to the so-called COPS *outsourcing* model). The set of Edge Routers and the Bandwidth Broker realize a sort of distributed bandwidth broker in a Diffserv network.

The COPS-DRA architecture is better described in [8], the protocol details can be found in [7]. Figure 3 provides an example of the message exchange between Q-SIP server, ER and BB.

6. Q-SIP / COPS-DRA TESTBED

The proposed architecture has been implemented in a test-bed composed of a set of Linux PCs. The Diffserv components of the test-bed have already been discussed in [12]. The overall picture of the test-bed is described in Figure 5. The described testbed implementation is based on the previous version of the Q-SIP protocol [10]. This previous version is a subset of the new one, as it only considers the QoS assured model and the stateless mechanism. Currently, the development of the stateful mechanism is also completed, while the development of the QoS assured model is undergoing. *[Note to the reviewer: beginning of October is our plan… so the final version this will be updated!]*

The Q-SIP proxy servers have been implemented on a Linux PCs based on Redhat 7.1distribution. The Q-SIP server is developed in Java (running on Sun JDK 1.2.2 virtual machine) and the COPS DRA client and server are developed in C. The internal architecture of the test bed elements is shown in Figure 6. The source code of the Q-SIP server is available under the GNU license at the URL in

[9]. Note that also the COPS-DRA source code is available under the GNU license. The publicly available "Ubiquity SIP User Agent" version 2.0.10 ([13]) has been used as SIP terminals, running on Win98 PCs.

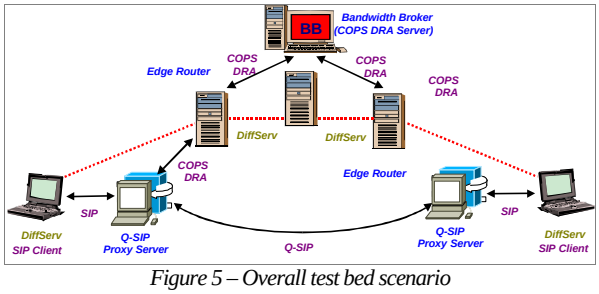

Figure 5 – Overall test bed scenario

The Q-SIP server has a modular architecture, in order to be able to handle different QoS mechanisms. As shown in Figure 6, there is a JAVA module called Generic Q-SIP Protocol Handler, which is independent of the underlying QoS mechanism. This module dialogues through a JAVA interface with a QoS-specific Interface module (realized in JAVA as well), which is specific of the underlying QoS model. In the picture, the COPS-DRA specific module is shown, which interacts through a socket interface to the COPS-DRA client process, realized in C. 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.

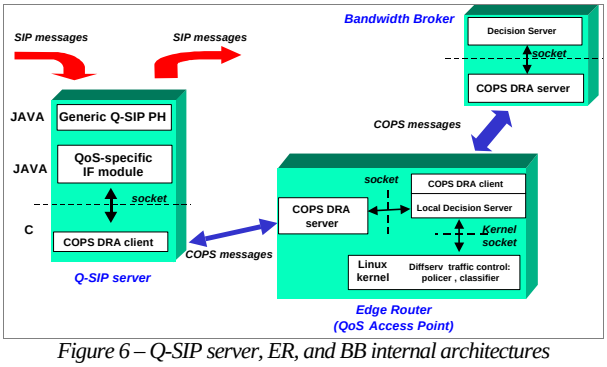

Figure 6 – Q-SIP server, ER, and BB internal architectures

6.1. Testbed in the AQUILA Project

In order to highlight the fact that the Q-SIP mechanism is independent on the underlying QoS mechanism, we briefly report that the Q-SIP proxy server has also been tested in the context of the trial of AQUILA project [11] founded by European Union. Here a different QoS signaling mechanism is used as the AQUILA project has defined a software component called EAT (End-user Application Toolkit) as a QoS portal for applications. Therefore a QoS-specific interface module in the Q-SIP server has been realized to interact with the AQUILA EAT for the QoS reservation using the CORBA Distributed Computing Environment. Then the EAT interacts with the AQUILA Edge Router to request the QoS to the network.

7. CONCLUSIONS

In this paper we have proposed an architecture for the interaction of SIP protocol with QoS mechanisms in a QoS enabled network. The enhancements to SIP protocol that have been given are basically independent of the specific QoS enabled network. Obviously the mechanism should be specialized for each specific case. In particular the interaction with a Diffserv network using COPS for admission control has been described in this paper. A possible deployment scenario based on a Q-SIP proxy server is proposed, having the advantage that "legacy" SIP user application can be fully reused. We note also that the solution is fully backward compatible with current SIP based equipment that does not support QoS, allowing a smooth migration. The test bed implementation of the proposed solution, including the internal architecture of the Q-SIP proxy server has been described.

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

[3] 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.

[4] G. Camarillo, W. Marshall, J. Rosenberg, "Integration of Resource Management and SIP", IETF Internet Draft <draft-ietf-sip-manyfolks-resource-07.txt>, February 2001, Work in Progress.

[5] 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.

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

[7] 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

[8] S. Salsano, L.Veltri, "QoS Control by means of COPS to support SIP based applications", IEEE Network, March/April 2002

[9] L. Veltri, S. Salsano, D. Papalilo, "QoS Support for SIP based Applications in Diffserv Networks", <draft-veltri-sip-qsip-01.txt>, September 2002, Work in Progress, http://www.coritel.it/projects/qsip

[10] L. Veltri, S. Salsano, D. Papalilo, "QoS Support for SIP based Applications in Diffserv Networks", <draft-veltri-sip-qsip-00.txt>, September 2001, Work in Progress, http://www.coritel.it/projects/qsip

[11] B. Koch, S. Salsano, "IP QoS at work: definition and implementation of the AQUILA architecture", accepted to IWDC 2001, Taormina, Italy, September 17-20, 2001 http://www.ist-aquila.org

[12] 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.

[13] "SIP User Agent", Ubiquity Software Corporation, http://www.ubiquity.net.