Internet Draft
October 2002
Expiration: April 2003

Luca Veltri
CoRiTeL
Stefano Salsano
Univ. of Rome "Tor Vergata"
Donald Papalilo
CoRiTeL

File: <draft-veltri-sip-qsip-01.txt>

SIP Extensions for QoS support

Status of this Memo

Abstract

This work describes an enhancement to SIP protocol for the interworking with QoS enabled IP networks. The proposed mechanism is simple and it fully preserves backward compatibility and interoperability with current SIP applications. The draft describes also, as an example, the application of this mechanism to a particular QoS enabled IP network, which implements Diffserv as transport mechanisms and COPS as protocol for QoS requests and for admission control.

Table of Contents

Glossary

SIP	Session Initiation Protocol
RSVP	Resource Reservation Protocol
Intserv	Integrated Services
Diffserv	Differentiated Services
BB	Bandwidth Broker
ER	Edge Router
COPS	Common Open Policy Service
PDP	Policy Decision Point
PEP	Policy Enforcement Point
Q-SIP	QoS enabled SIP
NSIS	Next Steps in Signaling
UA	User Agent

1. Introduction

Basically, SIP is an end-to-end session setup protocol. In order to provide an adequate quality of services for audio and video communications, a form of resource reservation may be needed. In the current view [1][3], the SIP user agents should rely on existing QoS protocols (e.g. RSVP) for the support of such resource reservation. This fact has two main drawbacks: i) the user applications 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 applications must implement such QoS protocol, with the increase of the 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.
An IP telephony (SIP based) architecture with end-to-end QoS support which can rely on the Intserv model is described in [1]. 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 has been chosen as QoS model in this work.
The NSIS IETF WG [2] is currently elaborating the signaling aspects that could support IP QoS. The reference model it is still under a discussion phase, and it is not completely defined. However, the architecture here presented seems to be quite aligned with the drafts under discussion within NSIS.

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.

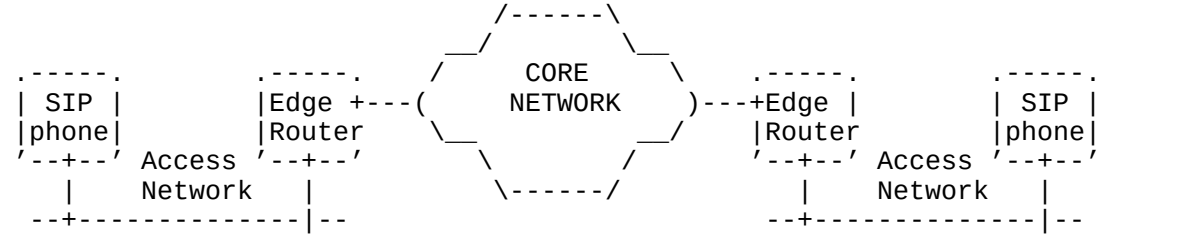


Figure 1 - Reference QoS scenario

In this draft 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 2) eliminates the need of QoS supports on the user terminals since all the QoS related functions can be moved to (local) 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 UA 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 UAs; no enhancements/modifications are needed in the SIP UA 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,
v) the architecture should be extendible to new models of QoS support for IP networks.

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. In the light of the current discussion in NSIS, the proposed Q-SIP server would act as a QoS Initiator interacting with a QoS Controller. The end-to-end SIP signaling can interact with the reservation of resource using "out of band" NSIS signaling.

2. QoS SIP: Overview

The basic idea is that SIP UAs use a default SIP proxy server in their domains for both outgoing and incoming calls. The UA 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 UAs and can add (and read) QoS related information in the SIP messages. This QoS information exchange is made transparent for the UAs. 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).
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 2 and Figure 3. The involved actors are the two SIP UAs, the two SIP servers and a QoS enabled network. The QoS provided by the QoS enabled network is accessed by QoS Access Point(s) located at the border of the network in the ERs. Depending on the mechanism implemented inside the core network in order to handle the reservation, there can be two logical types of QoS Access Points distinguished by the type of reservation offered: unidirectional and bidirectional from an ingress to an egress point (ER).

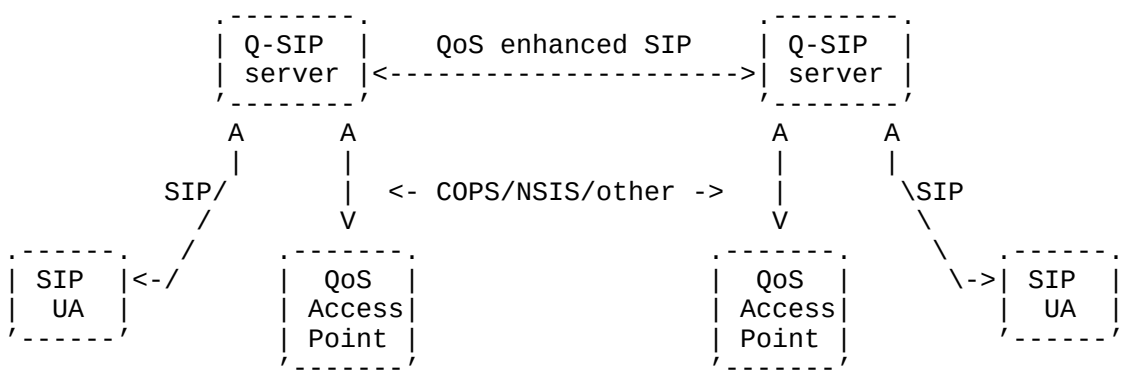


Figure 2 - Q-SIP architecture based on the use of Q-SIP servers on QoS IP networks that offer unidirectional flow reservation.

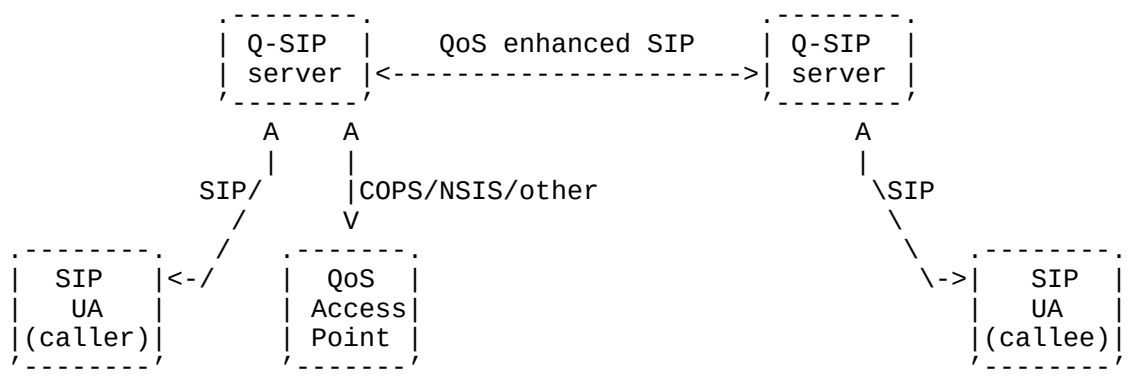


Figure 3 - Q-SIP architecture based on the use of Q-SIP servers on QoS IP networks that offer bidirectional flow reservation.

The setup of QoS sessions 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.
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 [4], with specific extension as proposed in [5].
In our scenarios, the QoS enabled network can provide unidirectional or bidirectional QoS reservations. In the first case, two different reservations have to be requested to the QoS network (also RSVP QoS model works in this way). When considering a QoS IP network that can provide bidirectional reservations, the difference is that we have a single QoS Access Point and a single reservation request made by the Q-SIP.
Note that we mainly refer to a scenario where the SIP UAs 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 also 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.

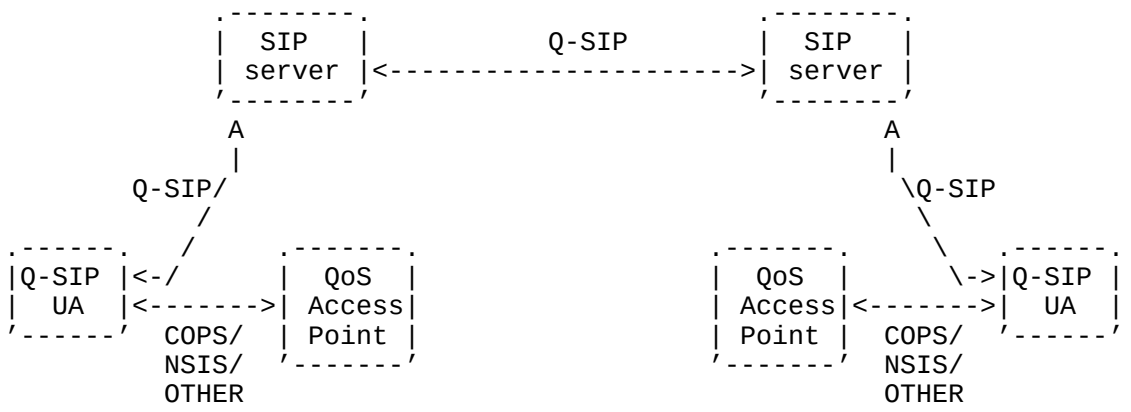


Figure 4 _ Q-SIP architecture with Q-SIP agents on user terminals.

Compared to Figure 2, note that SIP UAs become Q-SIP UAs 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 (see Figure 5).

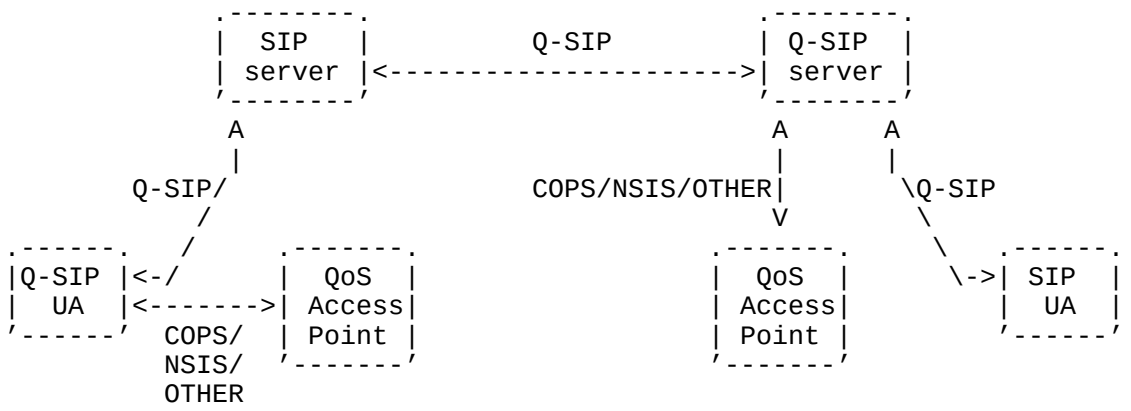


Figure 5 _ Asymmetric Q-SIP architecture.

2.1 QoS reservation modes

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.

2.2 QoS models

When the requested resource needed for a QoS call is not available, two options are possible: i) the call is setup without QoS, ii) the call is rejected. These two options are at the basis of the following two QoS models:
i) "QoS-Assured", that is the session should not be established if resources are not available; in this case the QoS should be setup before alerting the user avoiding that a user may respond to a call when resources are not available.
ii) "QoS-Enabled", that is the session is established regardless of the availability of QoS resources; eventually the user may be signaled about the presence of QoS.
In [1] the "QoS-Assured" model is considered. A possible interaction between SIP and resource management and a precondition framework is described.
On the contrary, the present 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. The extension of our proposal to a "QoS-Assured" model (conforming to [1]) is reported in Appendix A.

3. Q-SIP signaling mechanism

This section provides the detailed description of the signaling mechanisms of the proposed SIP based reservation architecture (Q-SIP). 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 Point(s). In the following we assume that the Edge Router(s) act as QoS Access Point and implement all the mechanisms needed to perform admission control decisions (possibly with the aid of a Bandwidth Broker (BB)) and policing function. As an example, the QoS scenario will be based on COPS as the protocol for QoS reservations.
The IP phones/terminals are located on the access networks; standard SIP UAs can be used and explicit SIP proxying configuration is set. When a call setup is initiated, the caller SIP UA 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). 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. The reference architecture is shown in Figure 2.
The basic goal of the Q-SIP signaling mechanism is to let the two parts (i.e. the Q-SIP servers) that are willing to setup a QoS session to interact each other and to exchange the needed information (e.g. IP addresses of ingress and egress QoS elements). A new SIP header (QoS-Info) will be defined for this purpose. We defined two variants of the procedure depending on the state information that is kept in the Q-SIP server during the session setup. One of the design goal of the SIP protocol is that a SIP proxy server should operate in a stateless way whenever possible, i.e. it should not be required for it to record any

session state. If we want to keep this principle for our Q-SIP server operation, some information will be recorded in the SIP request messages in a special way so that the servers will find this information in the SIP response messages. This will be the "stateless" variant of the Q-SIP protocol. Anyway, considering that the Q-SIP server is probably interested to store a QoS state for the call session after its establishment, it can be reasonable to store some state also during the session setup. Relying on this state, the information that will be transported in SIP messages will be simpler ("stateful" variant of the Q-SIP protocol).

3.1 Q-SIP message flow

In this section the description of the Q-SIP procedure is given, for the two types of reservations offered by the QoS enabled network: unidirectional or bidirectional modes. Note that the Q-SIP server must be aware (e.g. by configuration) of the type of reservation offered by the QoS enabled network.

3.1.1 Q-SIP message flow - unidirectional QoS reservation

With reference to Figure 6, the call setup starts with a standard SIP INVITE message sent by the caller to the local Q-SIP server (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, etc). The Q-SIP server is seen by the caller as a standard SIP proxy server. The Q-SIP server, based on the caller identity and on session information, decides whether a QoS session has to be started or not. Note that the service admission decision can be handled locally relying in a user profile or demanded to another external admission control entity, but this is outside the scope of this work.
If a QoS session has to be setup, the Q-SIP server extracts the required information from the message, inserts the additional Q-SIP header and the Record-Route header information (to assure that all the messages for this session will pass through itself) within the INVITE message.
If the stateful variant is used, some information is stored by the Q-SIP server in order to maintain trace of the current QoS session. We will refer to such information as "provisional QoS state".
If the stateless variant is used, the required information is stored as additional fields in the Record Route header.
Then the Q-SIP forwards the INVITE message towards the invited callee; the INVITE messages can be relayed by both standard SIP proxy servers and Q-SIP servers. When the Q-SIP server on the callee side (callee-side Q-SIP server) receives an INVITE message that contains the SIP QoS extensions, it understands that a session with QoS has to be setup. Therefore it extracts the needed information from the message, removes the Q-SIP extension and inserts Record-Route header. In case of the stateful variant, it initializes the "provisional QoS state", like the caller-side Q-SIP. In case of the stateless variant, it adds additional information in the Record-Route header.
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 access

network of the callee. 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-side Q-SIP receive a positive response for the QoS reservation request, it stores such QoS information completing the QoS state and sends the extension information for the callee side within the 200 OK message toward the caller. The QoS information data is stored by the Q-SIP server. In case of the stateful variant, this QoS information complete the QoS state previously stored. If the response for the reservation is negative, the Q-SIP server update the QoS state and it still inserts in the 200 OK response the extension header field needed for the caller-to-callee reservation in order to give the possibility to the caller-side Q-SIP server to make the reservation. The update of the QoS state reports the failure of the reservation so retransmissions of the 200 OK does not trigger QoS reservation request and only the extension header field is inserted to handle correctly the message retransmission. Actually, the handling of these reservations refusals is different depending on QoS service model (i.e. QoS-Assured or QoS-Enabled). Assuming a QoS-Enabled service, the Q-SIP server will simply continue with the signaling.
When the caller-side 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 and the QoS state previously installed is treated as in the callee-side Q-SIP server in order to handle correctly retransmissions. If the response is positive, the QoS state is completed.
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. Another way to assure the release of the resources, based on the use of time-outs and the INFO method, is described in section 5.

It is important to note that the proposed architecture keeps the compatibility with standard SIP UAs 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.

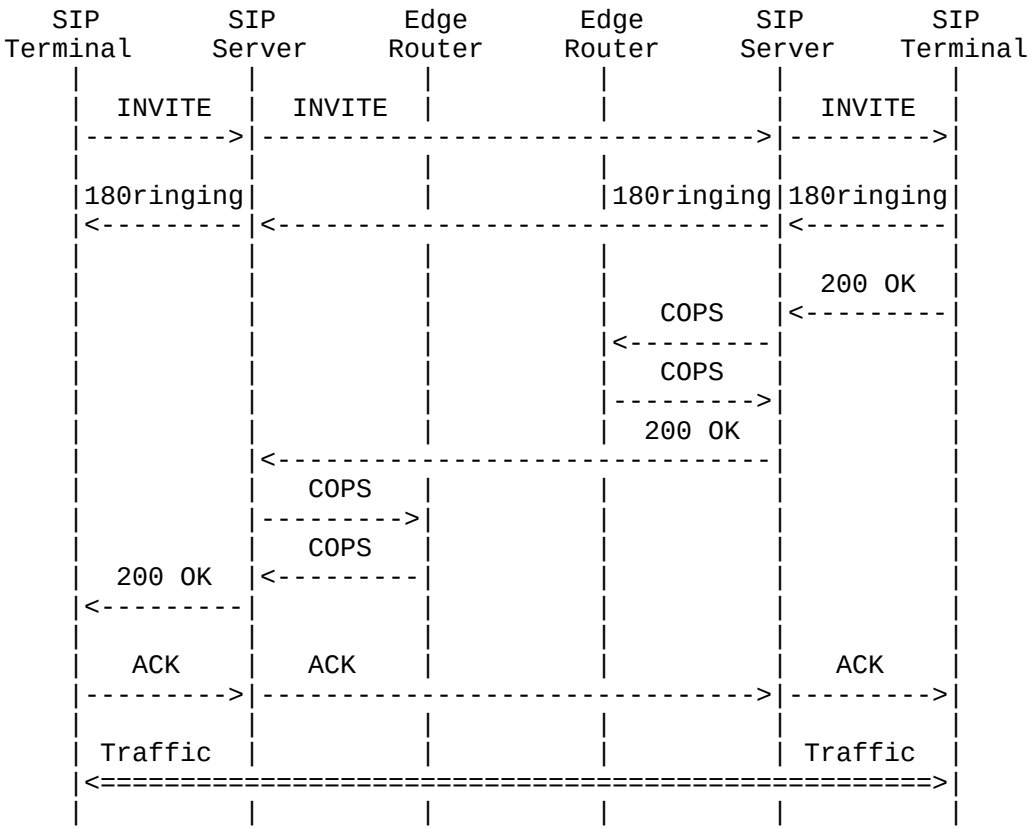


Figure 6 _ Q-SIP call signaling flow - unidirectional QoS mode

3.1.2 Q-SIP message flow - bidirectional QoS reservation

The fundamental difference from the QoS unidirectional reservation mode is that now there is only one interaction with the QoS provider, as depicted in Figure 3. In this case when the caller-side Q-SIP receives a 200 OK response message for a QoS call, it starts a "bidirectional" QoS reservation with the local QoS provider. The callee-side SIP server still participates to Q-SIP signaling but does not talk with a QoS provider. Analogously to the unidirectional case, the caller-side Q-SIP server reads the needed information from the first INVITE and inserts the Q-SIP extension header. The caller-side Q-SIP also keeps the "provisional QoS state", adds the Record-Route header (to remain along the path of the successive requests) and forwards the message. When the first INVITE reaches the callee-side Q-SIP, it installs the "provisional QoS state". When the callee-side Q-SIP receives a 200 OK matching a previously installed "provisional QoS state" it adds the QoS extensions (ER IP address etc) and forwards the 200 OK message. Note that the "provisional QoS state" on the callee-side Q-SIP is removed only when ACK message from the caller is received, in order to handle possible 200 OK retransmissions. When caller-side Q-SIP receives the 200 OK it acts

in the same way as for the unidirectional case, but asking the QoS provider for bi-directional reservation.

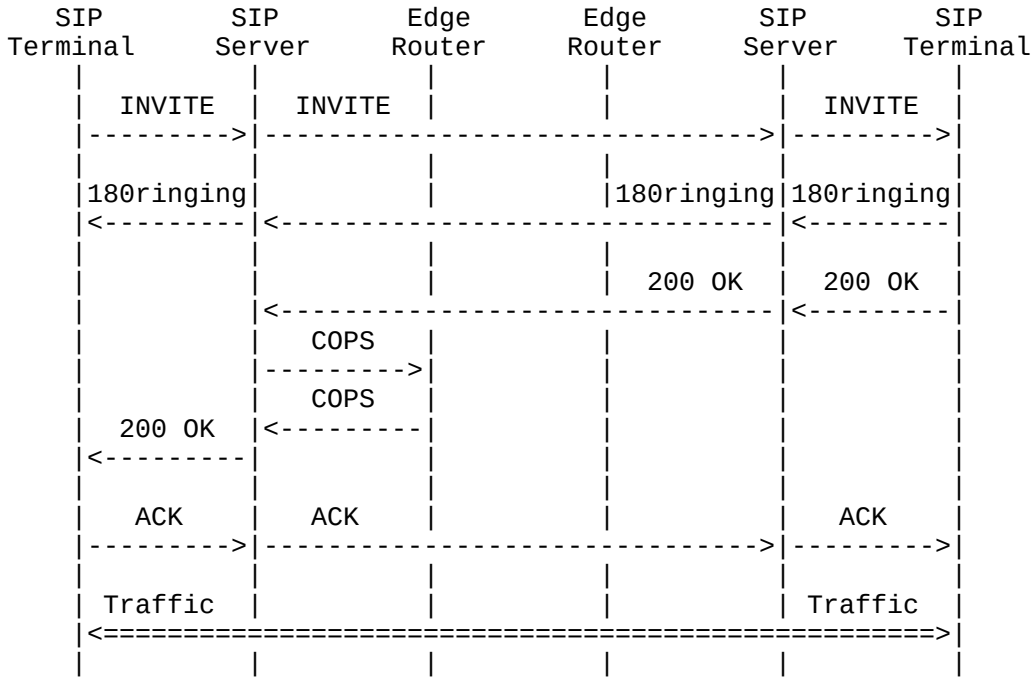


Figure 7 _ Q-SIP call signaling flow - bidirectional QoS mode

3.2 Q-SIP protocol

Regarding the management of QoS SIP sessions within Q-SIP servers, as introduced in the previous sections, two different approaches are considered:
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.
Note that, considering that it is reasonable that a Q-SIP server will be stateful after the session is setup (to keep track of QoS reservations), we think that the stateful version can be preferred.
The next two sections describe separately the two variants.
The two variants differ on the manner in which the initial transaction QoS information is kept by Q-SIP servers; in case of "stateful" Q-SIP variant, such initial information is maintained within the server by a "provisional QoS state", while in the "stateless" Q-SIP variant, this information is inserted as parameter in the Record-Route header within the request messages and read from response messages. The latter option is used according to the "RFC 3261" [3], stating that the Record Route parameters can be used as a solution for keeping state in the messages

rather than within the proxies. For this reason all parameters included must be echoed by the user agents (server side) within response messages.

3.2.1 Stateful variant of Q-SIP protocol

In this variant a "provisional QoS state" is kept in the Q-SIP proxy servers during session setup.
When the first Q-SIP server (i.e. the caller-side Q-SIP server) receives a new INVITE message, it inserts the following new field :

QoS-Info: <qos-param> *(;<qos-param>)
Wherein <qos-param> can be some of:
<qos-mode> | <er-ingress> | <er-egress> | <qos-domain> | <caller-media-addr> | caller-media-port> | <other>

example:

```
QoS-Info: qos-domain=coritel.it;er-ingress=192.168.77.5;
          qos-mode=unidirectional
```

By means of the "er-ingress" field the caller-side Q-SIP informs the callee-side Q-SIP server about the IP address of the caller-side ER; instead the "er-egress" field is inserted by the callee-side Q-SIP server for similar reason. This information is used by the Q-SIP servers to specify to the corresponding Q-SIP server the remote endpoint of the reservation. Note that we have assumed the "caller to callee" as an "ingress to egress" reference direction. The "qos-domain" field is used to identify the QoS enabled domain that the reservation has to be accomplished. These wouldn’t be strictly required in a intra-domain scenario (one QoS enabled domain); however it could be useful for possible interdomain extensions.
The Q-SIP server that initiates the QoS session sets also the "qos-mode" field according to the type of QoS provider it supports (unidirectional or bidirectional) and according to user profiles (in a scenario where unidirectional and bidirectional QoS providers are both possible).
A Q-SIP server that receives a message and recognizes that it is for a QoS session, 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 UA processes the call and if the call is accepted, generates a 200 OK response.

When the 200 OK reaches the callee-side Q-SIP server, the server associates the response to the previously stored provisional QoS state. The registered "qos-mode" specifies the kind of reservation to be applied. 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-domain=coritel.it;er-egress=192.168.90.3;
          qos-mode=unidirectional
```

In 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.

Even if the QoS reservation for the callee-to-caller flow was not successful, the extension is still inserted to make possible to reserve the QoS for the caller-to-callee flow in a "QoS-Enabled" scenario; however, in this case, the "provisional QoS state" is removed at the receipt of the ACK for the same session. For a "QoS-Assured" model see Appendix A.
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 UAs (within the SDP body) and/or according to the particular user profile. For most audio codecs it can be relatively easy to prepare a mapping table of codecs and required bandwidths, for RTP streams. For video codecs this is not so simple therefore one could have to rely on user profiles. In Appendix C, it is

reported an example of mapping table 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 filtering 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-side/callee-side Q-SIP servers.
Note that the caller address and port information needed to setup the QoS for both directions are found within INVITE messages. Instead, the reservation is made by the caller-side and callee-side Q-SIP servers when they receive the 200 OK message. The callee media address and port is extracted directly from the 200 OK message (the callee address from the callee-side Q-SIP server and the callee address and port from the caller-side Q-SIP server).

The Q-SIP call setup flow is shown in Figure 6 and Figure 7.

The tear down procedure is triggered by the receiving of the BYE and 200 OK messages at the caller-side /callee-side Q-SIP servers. 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 a stored "QoS state". In Appendix B there is an example of "provisional QoS state" that can be associated to a new QoS setup transaction and the "QoS state" that can be associated to the active 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. Another possibility to force a resource release procedure is based on the use of time-outs and INFO method ([7]) by the Q-SIP servers, as described in section 5. Note that this mechanism can be used only if the UA supports the INFO method.
In order to ensure that the SIP signaling will cross the Q-SIP servers, the Record-Route and Route headers are used. For this reason, the Q-SIP server inserts the Record-Route header within requests for all QoS SIP sessions.

Appendix D reports an example of Q-SIP messages using the stateful Q-SIP variant.

3.2.2 Stateless variant of the Q-SIP protocol

Let us consider the stateless Q-SIP specification, i.e. the Q-SIP variant that let the server stateless during the call setups. For this reason, a mechanism is needed in order to allow a Q-SIP server that receives a 200 OK message to retrieve all the information needed to

setup a reservation. Instead of maintaining a provisional QoS state within the Q-SIP server, the QoS information is included in the SIP request messages and retrieved when intercepting the responses. The insertion mechanism has been defined in such a way that does not require the collaboration of SIP UAs (in order to allow the use of QoS-unaware SIP UAs). For such objective, the Q-SIP makes use of the Route/Record-Route SIP mechanism. According to the SIP specification, the Record-Route header is returned opaquely by the called UA within the response messages. Such functionality allows the Q-SIP server to "store" the QoS information as Record-Route header parameter and to obtain it back in the response messages.
When the caller-side Q-SIP receives an INVITE request, as specified for a stateful Q-SIP server, it appends the previously defined QoS-Info header and the Record-Route header.
The Record-Route header is now extended with the following Q-SIP parameters:

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

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

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

example:

```
Record-Route: <sip:qsip1.coritel.it;lr;
               qos-mode=unidirectional;
               qos-domain=coritel.it;
               er-ingress=192.168.77.5;
               caller-media-addr=192.168.10.44;
               caller-media-port=3220>
```

Note that, although it could appear redundant, both the qos-info Record-Route parameter and the QoS-Info header is inserted by the Q-SIP.
In the same way, the callee-side Q-SIP server appends its Record-Route header, that becomes:

```
Record-Route: <sip:qsip1.coritel.it;lr;qos-mode=unidirectional;
              qos-domain=coritel.it;er-ingress=192.168.77.5;
              caller-media-addr=192.168.10.44;caller-media-port=3220>,
              <sip:qsip2.coritel.it;lr;er-egress=192.168.90.3;
              caller-media-addr=192.168.10.44;caller-media-port=3220>
```

When the INVITE message reaches the callee host, the UA processes the call, and, at last, generates the 200 OK response (if the call is accepted).
If the UA is not aware of Q-SIP it simply discards the Q-SIP header (the QoS-Info header if it is not removed by the callee-side Q-SIP server) 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, Call-ID and Record-Route header.
When the 200 OK reaches the callee-side Q-SIP server, the Record-Route field is read and the QoS session information are extracted. In case of unidirectional reservation mode a QoS request for the callee-to-caller direction is started. When this QoS reservation request/response phase

is concluded and the resource is reserved, a QoS state may be stored and the 200 OK messages is relayed toward the caller.
As for the stateful Q-SIP variant, a new QoS-Info header in added to the response.
Even if the QoS reservation for the callee-to-caller flow was not successful, or a bidirectional reservation is handled, this field is still inserted to inform the callee-side Q-SIP about the callee QoS end-point. The complete procedure for a "QoS-Assured" model is described in Appendix A.
It is very important to remember that the use of the previously defined Record-Route parameters 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 robust tear-down procedure, for accounting and for resource control.

Regarding the caller media end-point (caller address and port), although it is extracted from INVITE messages, it is used for making the reservation when receiving the 200 OK message. Since no state is maintained within the servers, both caller-side and callee-side Q-SIP servers also store caller media end-point information within the Record-Route qos-param (see previous examples).

Appendix E reports an example of Q-SIP messages using the stateless Q-SIP variant.

4. Q-SIP syntax and rules

4.1 Syntax

```
  QoS-Info Header  = "QoS-Info" HCOLON qos-param *(SEMI qos-param)
  qos-param        = qos-mode / er-ingress / er-egress /
                     qos-domain / other
  qos-domain       = "qos-domain=" domain-name
  er-ingress       = "er-ingress=" ingress-ER-address
  er-egress        = "er-egress=" egress-ER-address
  qos-mode         = "qos-mode=" qos-reservation
  qos-reservation  = "unidirectional" / "bidirectional"
  domain-name      = alphanum / alphanum *( alphanum / "-") alphanum
  caller-media-addr= "caller-media-addr=" caller-addr
  caller-media-port= "caller-media-port=" media-port
  other            = token [EQUAL alphanum]

Record-Route Header= "Record-Route" HCOLON "<"server-uri;
                     qos-info *(;next_param)">"
  qos-info         = qos-param *(;qos-param)
```

4.2 Rules

Every Q-SIP server MUST be able to act as both caller-side and callee-side Q-SIP servers.

QoS-Info Header : it is inserted by the caller-side Q-SIP server when processing INVITE messages, and by the callee-side Q-SIP server when processing 200 OK response messages (referring to an INVITE method). The QoS-Info Header may carry both mandatory and optional parameters. Table I reports for each QoS-Info parameter, whether it is mandatory (m) or optional (o) for caller-side and callee-side Q-SIP servers. These rules apply for both stateful and stateless Q-SIP variants. If no qos-mode is specified, the unidirectional reservation mode is supposed.

Parameter	caller-side Q-SIP	callee-side Q-SIP
qos-domain	o	o
er-ingress	m	-
er-egress	-	m
qos-mode	o	o
caller-media-addr	o	-
caller-media-port	o	-

Table I - Mandatory and optional QoS-Info Header parameters

Record-Route Header : it is inserted by both caller-side and callee-side Q-SIP servers by both stateful or stateless Q-SIP variants. The Record-Route guaranties that the Q-SIP remains along the SIP signaling path. The "qos-info" Record-Route parameter is inserted only for the stateless Q-SIP variant. The implementation of the stateless Q-SIP extension variant is not mandatory for a Q-SIP server; however if it is implemented, all stateless Q-SIP rules MUST be satisfied.
Both caller-side and callee-side Q-SIP servers MUST insert the "qos-info" Record-Route parameter.
Table II reports for each QoS-Info parameter, whether it is mandatory (m) or optional (o) for caller-side and callee-side Q-SIP servers. These rules apply for both stateful and stateless Q-SIP variants. If no qos-mode is specified, the unidirectional reservation mode is supposed.

Parameter	caller-side Q-SIP	callee-side Q-SIP
qos-domain	o	o
er-ingress	m	-
er-egress	-	m
qos-mode	o	o
caller-media-addr	m	m
caller-media-port	m	m

Table II - Mandatory and optional qos-info parameters for Record-Route Header

5. Use of INFO method for robust tear-down procedure

The tear-down of resources must be robust with respect to events like terminal failures or network failures that may prevent the Q-SIP server to receive the BYE message closing the session. A way to assure the correct release of the previously reserved resources is the use of time-outs and INFO method ([7]).
A Q-SIP proxy can use timeouts associated with the call session. The timeout expiration triggers the generation of an INFO request matching the characteristics of the dialog ID associated to the call state and directed to the controlled SIP terminal (User Agent). This request and its associated responses can be used as a "ping" for the call session activity.
The INFO request, generated by the client side of the proxy, is sent only to the local UA for a call session that the outbound Q-SIP proxy has reserved resources to, so only local additional signaling messages are generated.
The server side of the UA can respond with several messages that are interpreted and used by the Q-SIP proxy:

UA response to INFO	Q-SIP Proxy action
200OK : Call/transaction exists	renew timeout associated
481 : Call/transaction doesn't exist	release reserved resources
405 : Method not allowed/supported	don't use this mechanism for this call session
501 : Not implemented	don't use this mechanism for this UA

The chose of the time-out value is left to vendor implementation.

6. SIP Terminals

Although it has been supposed that the SIP user UAs are not aware of the Q-SIP reservation mechanism, Q-SIP aware UAs can be also considered (Figure 4).
Q-SIP aware UAs should simply include Q-SIP as described in the previous sections. In that case, the UAs 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 UAs and directed to standard SIP UAs with Q-SIP proxy servers, and vice-versa (Figure 5).

7. 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 the proposed 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: is possible either to store state information in Q-SIP server during the session establishment or to carry all the needed information in the SIP messages.

8. Security Considerations

A proxy that performs resource reservations triggered by the reception of unauthenticated requests can be an easy target of a DoS (Denial of Service) attack. Requests for a possible QoS session SHOULD be authenticated.
In order to assure the correct handling of the QoS service offered to the UA by the outbound Q-SIP server, proxy authentication SHOULD be used. In this way, the Q-SIP before initiates a QoS session and reserving resources, can use the authorization/authentication mechanism to assure the right access control and availability of the service in accord to the user profile.
The user profile can contain user password and the type of service that the user is enabled to, so it can be used as authentication and resource reservation support.

9. Change log and prototype implementation

This version v1 is the second version of the Q-SIP draft. The changes with respect to previous version v0 are:

- QoS state information in SIP messages is now carried in Record Route headers instead of Via headers (according to the change in SIP specification of [3])
- The stateful variant of the Q-SIP protocol has been specified.

- The use of bidirectional reservation (according to 2.1.)is supported
- The use of INFO messages to support robust tear-down of resources has bee specified
- A discussion on QoS assured model (Appendix A) has been added

A prototype implementation of a Q-SIP server is available [6].

The messages reported in Appendix D are extracted from the current implementation in a simple successful SIP call that involves two Q-SIP servers.

Appendix A: QoS-Enabled vs. QoS-Assured

In the previous part of the document the QoS enabled resource reservation is considered. In a QoS assured scenario [1] the QoS can be a precondition to the establishment of the session indicated by SIP. An UA can use the Q-SIP proxy reservation mechanism in order to reserve resources before beginning the session. In this scenario a UA can be preconfigured to use the mechanism here described. Various situations depending on the type of reservation handled by the proxy are discussed. The UAs involved in this scenario supports the qos preconditions as specified in [1] and the reliable provisional responses [8]. A precondition is an information written in the SDP describing the SIP session. By means of this information the terminals can communicate each other that they want a QoS reservation and then that the reservation has been established.
The main idea is the following. A Q-SIP server receives a message for a local UA containing preconditions (i.e. stating that QoS reservation is needed). The Q-SIP server will take care of the resource reservation and change the preconditions in the message according to the reservation done. In other words the Q-SIP will ensure that preconditions are met with no need for the UAs to setup the QoS reservations..
This can be considered an alternative scenario to those presented in [1] that consider only UAs supporting RSVP.

In the following sections A.1 and A.2 the technical details of the possible interaction of the QoS assured scenario described in [1] and the Q-SIP architecture are provided.

Note that in the described scenarios the Q-SIP server needs to modify the SDP inside the SIP message. Another more elegant solution could be to insert a new SIP header to report the result of the resource reservation to the UA. The UA will then change the SDP as described in [1]. The drawback in this case is that the UAs need to supports the new defined header.

A.1 Q-SIP using unidirectional QoS Network reservation

A.1.1 Bidirectional e2e reservation sender initiated

Figure 8 reports the signaling flow with the most important interactions.

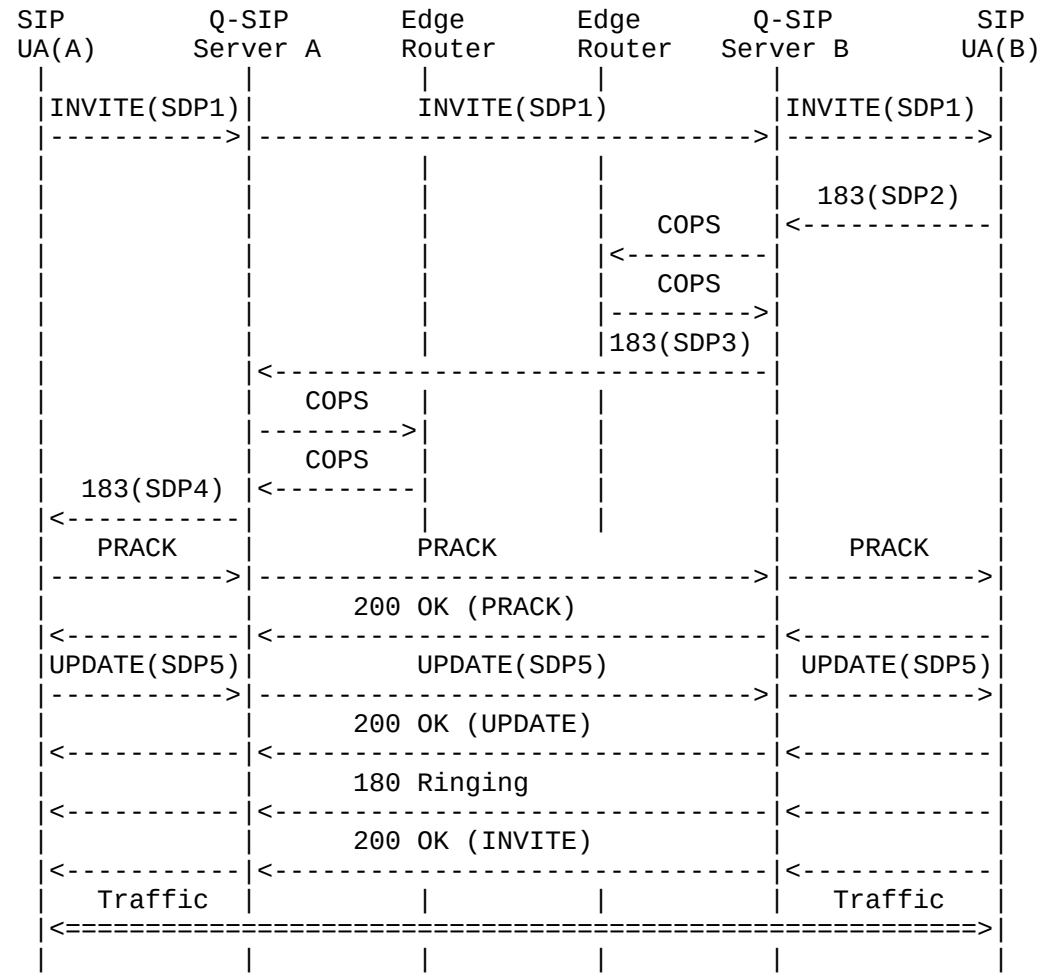


Figure 8 _ Bidirectional e2e successful reservation using Q-SIP in the QoS assured sender initiated case

When Q-SIP A receives an INVITE containing an offer from a UA that is preconfigured (user profile defined) to use it for the resource reservation in a QoS assured mode, it reads SDP1. If it contains the SDP attribute "a=des:" with the "qos" precondition_type, "mandatory" strength tag, "e2e" status type and "send" or "sendrecv" direction-tag the Q-SIP starts a QoS session as described previously. Almost the same for Q-SIP B, that relies in the user profile of the called UA (UA (B)) to start the QoS session. The difference is even in the direction-tag that must be "recv" or "sendrecv" to initiate the QoS session.
UA (B) relies in Q-SIP proxy QoS handling (preconfigured for proxy resource reservation) so it responds with a reliable 183(SDP2) if it wants to set-up the call with QoS. If for any reason it does not want,

it responds with a 580 Failure that is used also by the Q-SIP proxies to terminate the pending QoS session.
When the Q-SIP B receives the 183(SDP2), it has all the information to perform the resource reservation and depending on the result it changes the SDP: if reservation (callee to caller) fails it sends SDP2 (SDP not changed), if is OK sends SDP3(SDP changed!!!).
If the Q-SIP A receives a 183(SDP2), it understands that the reservation in the callee to caller direction is failed , terminates the initiated QoS session and proxies the message to the UA(A). If the message received is a 183(SDP3), it performs the resource reservation. If the reservation is successful the Q-SIP changes SDP3 in SDP4, else it terminates the QoS session and does not change the SDP3.
If the UA(A) receives a 183 with SDP4 it sends immediately the new offer in SDP5 using the UPDATE message. In any other case it assumes that the QoS session has failed so it sends SDP6 in the UPDATE message.
Q-SIP A and Q-SIP B do nothing in case of UPDATE with SDP5, in case of SDP6 the Q-SIP B releases the resources previously reserved.
Hereafter the relevant parts of the SDPs are listed:

```
        SDP1:   a=curr: qos e2e none
                a=des: qos mandatory e2e sendrecv

      SDP2: a=curr: qos e2e none
            a=des: qos mandatory e2e sendrecv
            a=conf: qos e2e recv

      SDP3: a=curr: qos e2e send
            a=des: qos mandatory e2e sendrecv
            a=conf: qos e2e recv

      SDP4: a=curr: qos e2e sendrecv
            a=des: qos mandatory e2e sendrecv
            a=conf: qos e2e recv

      SDP5: a=curr: qos e2e sendrecv
            a=des: qos mandatory e2e sendrecv

      SDP6: a=curr: qos e2e ****
            a=des: qos failure e2e sendrecv
```

A.1.2 Unidirectional e2e reservation sender initiated

In these cases only one flow is required to have the QoS support as reported on the SDP1 (the offer).

Caller to callee QoS e2e required:

```
      SDP1: a=curr: qos e2e none
            a=des: qos mandatory e2e send
```

Q-SIP A handles the reservation and if it successful it changes the SDP of the answer: SDP2 in SDP3. Q-SIP B only use Q-SIP extensions to transmit to Q-SIP A the callee-side ER.

```
SDP2: a=curr: qos e2e none
      a=des: qos mandatory e2e recv

SDP3: a=curr: qos e2e recv
      a=des: qos mandatory e2e recv
```

Callee to caller QoS e2e required:

```
SDP1: a=curr: qos e2e none
      a=des: qos mandatory e2e recv
```

Q-SIP A only uses Q-SIP extensions to transmit to Q-SIP B the caller ER. Q-SIP B handles the reservation and if it is successful it changes SDP2 in SDP3 as reported below.

```
SDP2: a=curr: qos e2e none
      a=des: qos mandatory e2e send

SDP3: a=curr: qos e2e send
      a=des: qos mandatory e2e send
```

In all cases if a failure situation occurs the UA(A) sends the UPDATE with the new offer containing SDP4.

```
SDP3: a=curr: qos e2e ****
      a=des: qos failure e2e ****
```

Note: **** mean send or recv.

A.1.3 Bidirectional e2e reservation receiver initiated

In this case the first INVITE does not contain an SDP so the Q-SIP entities cannot distinguish at this point if the session is to be set or not with QoS. Even in this case the outbound proxy for the caller and the callee side may remain on the signaling path using the Record-Route support. As reported in Figure 9 it is the UA(B) that initiates the offer-answer exchange sending the reliable 183(SDP1).
When Q-SIP B receives the 183(SDP1) and an associated QoS session does not exist, it initiates the QoS session and uses the Q-SIP extensions to transmit the callee-side ER.
When Q-SIP A receives the 183(SDP1) reporting also the extensions, it initiates the QoS session.
UA(A) knows that it is configured with the Q-SIP for supporting the QoS so it can send immediately the PRACK(SDP2)[7][8].
In the Q-SIP A the receipt of the PRACK(SDP2) for a QoS session of a UA that is configured to have the QoS assured support triggers the reservation (now we have all the needed information). If the reservation is successful this is reported inside SDP3 and Q-SIP A uses the Q-SIP

extensions to transmit to the callee-side Q-SIP proxy the caller-side ER, if not the SDP is removed (compliant with [1]) and the QoS session is terminated.
If the Q-SIP B receives PRACK without SDP2 and the extensions, it removes the QoS session and simply proxies the message. If the message received is PRACK(SDP3) with the extensions, it tries to reserve the resources requested. If the reservation is successful this is reported inside SDP4, if not the SDP is removed, the QoS session is terminated and the message is forwarder to UA(B).
Hereafter the relevant parts of the SDPs involved are reported:

```
      SDP1: a=curr: qos e2e none
            a=des: qos mandatory e2e sendrecv
            a=conf: qos e2e recv

      SDP2: a=curr: qos e2e none
            a=des: qos mandatory e2e sendrecv

      SDP3: a=curr: qos e2e send
            a=des: qos mandatory e2e sendrecv
      SDP4: a=curr: qos e2e sendrecv
            a=des: qos mandatory e2e sendrecv
```

Note that the if UA(B) receives SDP4, it knows that the preconditions are meet so it can immediately send 200 OK (of PRACK) and the 180 Ringing without the need of an UPDATE. The UA(A) receives the 180 Ringing that assures that the preconditions are met.

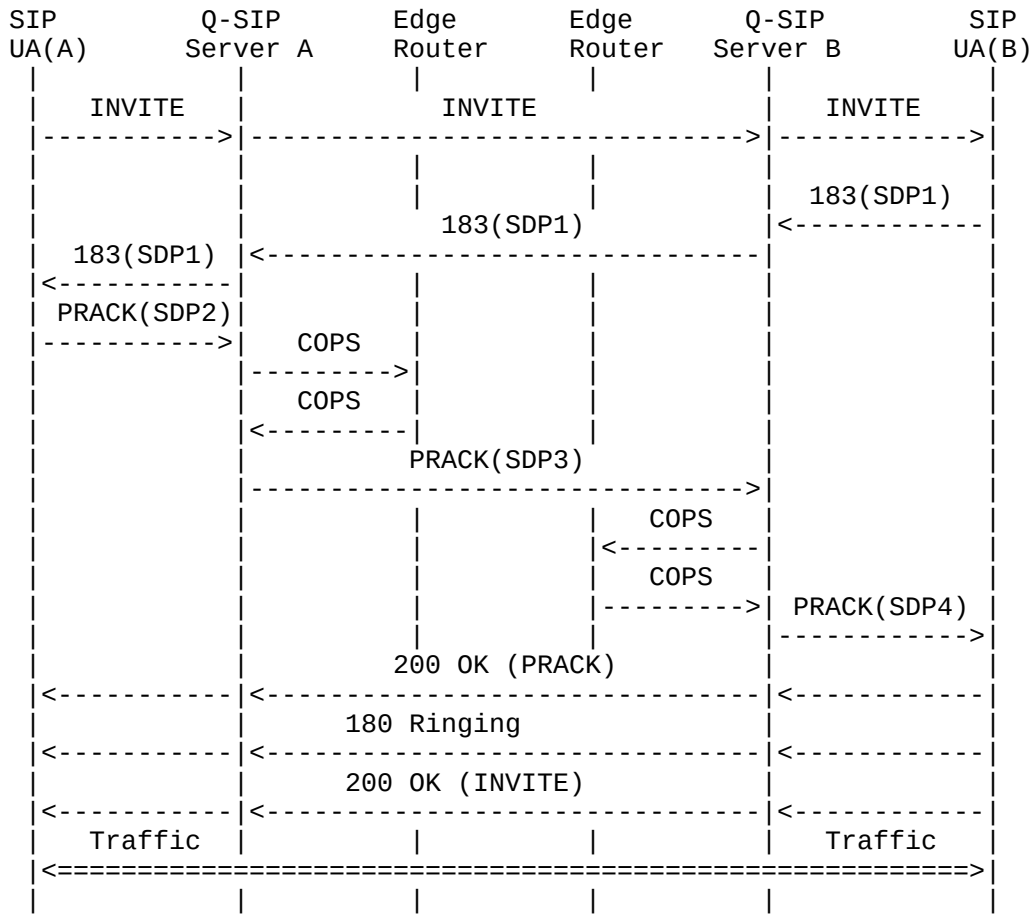


Figure 9 _ Bidirectional e2e successful reservation using Q-SIP in the QoS assured receiver initiated case

If the PRACK received by UA(B) does not contain SDP, we have the precondition failure case that is handled according to [7].

A.1.4 Unidirectional e2e reservation receiver initiated

In these cases only one flow is required to have the QoS support as reported on the SDP1 (the offer).

Callee to caller QoS e2e required:

```
SDP1: a=curr: qos e2e none
      a=des: qos mandatory e2e send
```

Q-SIP A only uses Q-SIP extensions to transmit to Q-SIP B the callee-side ER. Q-SIP B handles the reservation and if it successful change SDP of the answer: SDP2 in SDP3. If not it removes the SDP from the PRACK.

```
SDP2: a=curr: qos e2e none
      a=des: qos mandatory e2e recv

SDP3: a=curr: qos e2e recv
      a=des: qos mandatory e2e recv
```

Caller to callee QoS e2e required:

```
SDP1: a=curr: qos e2e none
      a=des: qos mandatory e2e recv
      a=conf: qos e2e recv
```

Q-SIP B only uses Q-SIP extensions to transmit to Q-SIP B the caller ER using the 183(SDP1). Q-SIP A handles the reservation and if it is successful, change SDP2 in SDP3 as reported below. If not removes the SDP from the PRACK.

```
SDP2: a=curr: qos e2e none
      a=des: qos mandatory e2e send

SDP3: a=curr: qos e2e send
      a=des: qos mandatory e2e send
```

A.2 Q-SIP using bidirectional QoS Network reservation

A.2.1 Bidirectional e2e reservation sender initiated

In the Figure 10 is reported the signaling flow with the most important entity interactions. The main differences are that only one of the Q-SIP (the caller one) is involved in the network reservation and the other one needs only as support to have the needed information. Here below are listed the entity interactions:

Q-SIP A receives INVITE(SDP1) from an UA that is enabled to receive the QoS assured support: Initiate an QoS session and proxy the message containing the Q-SIP extensions for this case.
Q-SIP B receives INVITE(SDP1) with the Q-SIP extensions: It installs the QoS session and proxy the message.
UA(B) receives INVITE(SDP1) and it is preconfigured to have the QoS proxy support (if need), so it sends the 183(SDP2).
Q-SIP B receives 183(SDP2) for an existing QoS session: It inserts the Q-SIP extensions and proxy the message.
Q-SIP A receives 183(SDP2) for an existing QoS session: It handle reservation; if it is successful change SDP in SDP3, if not don’t change-it.
When UA(A) receives 183(SDP3) it sends PRACK and UPDATE(SDP4). In the other cases (preconditions failure), it sends PRACK and UPDATE(SDP5).

The SDPs involved in the signaling flow:

```
      SDP1: a=curr: qos e2e none
            a=des: qos mandatory e2e sendrecv

      SDP2: a=curr: qos e2e none
            a=des: qos mandatory e2e sendrecv
            a=conf: qos e2e recv

      SDP3: a=curr: qos e2e sendrecv
            a=des: qos mandatory e2e sendrecv
            a=conf: qos e2e recv

      SDP4: a=curr: qos e2e sendrecv
            a=des: qos mandatory e2e sendrecv

      SDP5: a=curr: qos e2e none
            a=des: qos failure e2e sendrecv
```

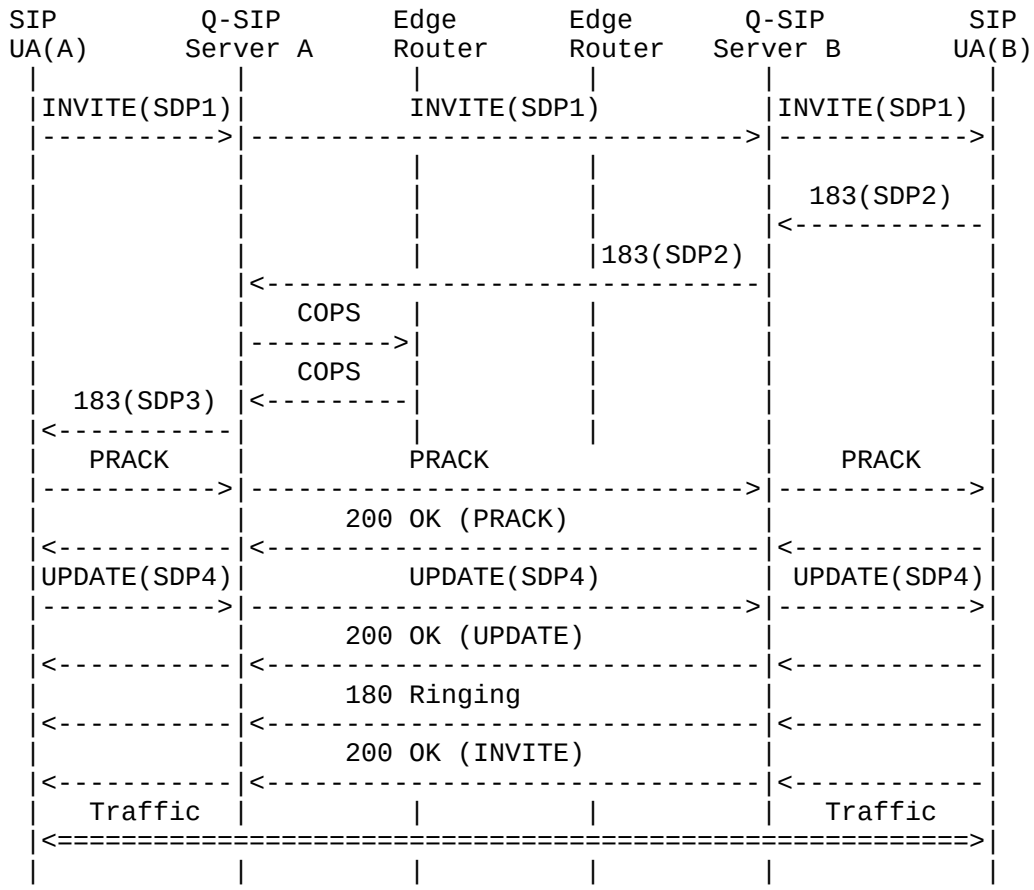


Figure 10 _ Bidirectional e2e successful reservation using Q-SIP in the QoS assured sender initiated case

Note that the QoS session on the Q-SIP B is removed when the PRACK for the 183 is received.

A.2.2 Bidirectional e2e reservation receiver initiated

The difference from the 3.1.3 is that only one reservation is done by the caller-side Q-SIP and the Q-SIP B only supports this reservation by giving the callee-side ER.
Here after the SDP involved in the signaling messages (shown in the Figure 11) are reported:

SDP1: a=curr: qos e2e none
a=des: qos mandatory e2e sendrecv
a=conf: qos e2e recv

SDP2: a=curr: qos e2e none
a=des: qos mandatory e2e sendrecv

SDP3: a=curr: qos e2e sendrecv
a=des: qos mandatory e2e sendrecv

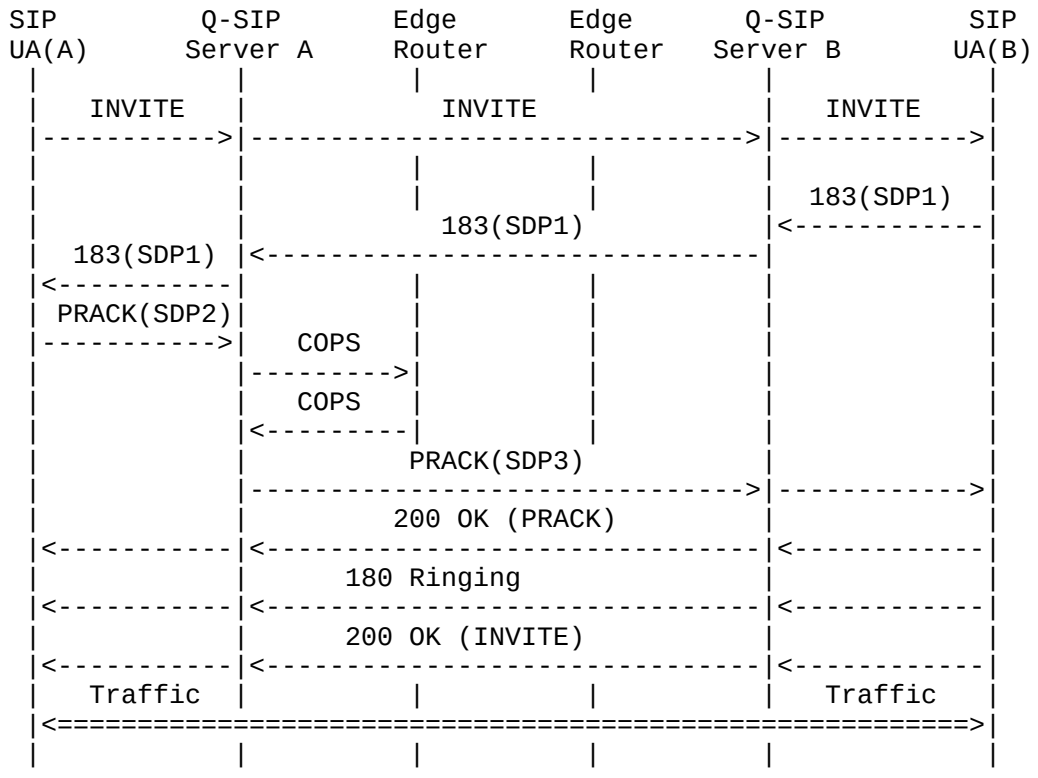


Figure 11 _ Bidirectional e2e successful reservation using Q-SIP in the QoS assured receiver initiated case

Appendix B - Description of the QoS State

A possible implementation of the QoS State :

```
<QoState> ::= <Call-Identification>
              <Type of state >
              <Scope of the reservation>
             <Type of the reservation>
             <Session identification filter>
```

The Call-Identification has the following format:

<Call-Identification> ::= <Call-ID>

The type of state has the following format:

< Type of state > ::= <Provisional/Call>

The scope of the reservation has the following format :

```
<Scope of the reservation> ::= <QoSdomainID>
                               <Ingress ER>
                               <Egress ER>
                               <Bandwidth>
```

The type of the reservation has the following format :

<Type of the reservation> ::= <Unidirectional/Bidirectional>

The Session identification filter has the following format:

```
<Session identification filter> ::= <Source address>
                                    [<Source port>]
                                    <Destination address>
                                    <Destination port>
```

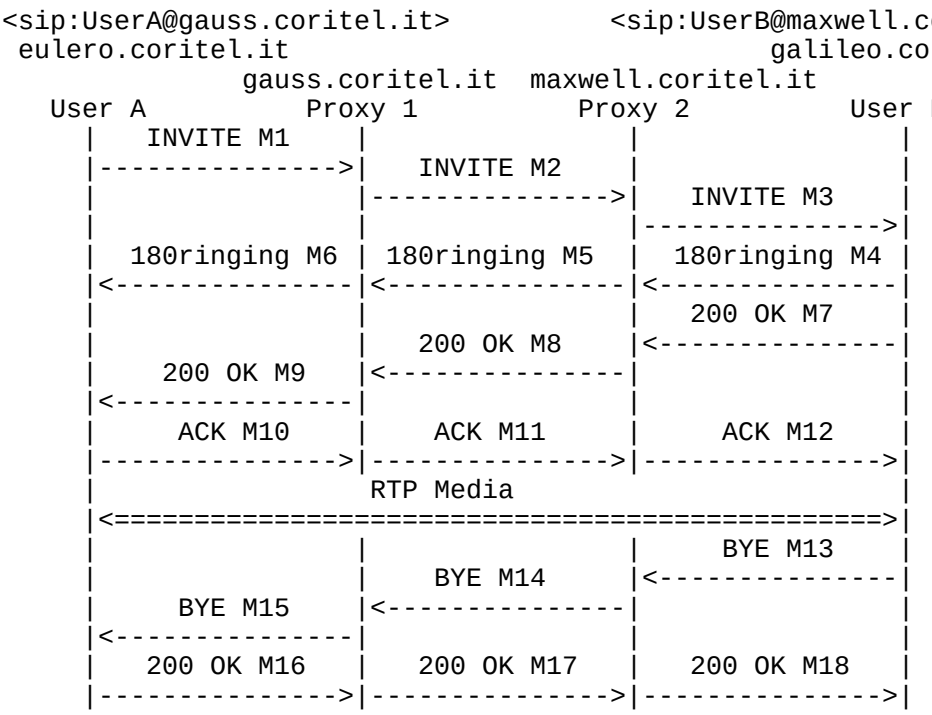
Note that the source/destination port is to intend as the ingress ports for the media flow (the port where the User Agent wait for the media data). According to the assumptions made before, that the QoS state in our scenario refers to a unidirectional or bidirectional flow inside the core network.

Appendix C - Payload type vs. bandwidth

Code	Payload Type	Payload Bit-Rate (kbit/s)	Bandwidth (IPv4/IPv6) (kbit/s)
PCMU	0	64	81.6 / 88
1016	1	16	33.6 / 40
G.721	2	32	49.6 / 56
GSM	3	13	30.6 / 37
G.723	4	6.3	23.9 / 30.3
DV14	5	32	49.6 / 56
DV14(2)	6	64	81.6 / 88
LPC	7	2.4	20 / 26.6
PCMA	8	64	81.6 / 88
G.722	9	64	81.6 / 88
MPA	14	32	49.6 / 56
G.728	15	16	33.6 / 40
G.729	18	8	25.6 / 32

Appendix D - Examples of Q-SIP messages

Examples of Q-SIP messages in the successful reservation scenario using the "provisional QoS state" approach are depicted in the picture hereafter. The messages are numbered from M1 to M18. Only the messages sent by the proxy servers are reported in detail.



Message M2 (INVITE from Proxy 1 to Proxy 2):

```
INVITE sip:UserB@maxwell.coritel.it SIP/2.0
Via: SIP/2.0/UDP gauss.coritel.it:5060;branch=z9hG4bKzksdfse3re
Via: SIP/2.0/UDP eulero.coritel.it:5060;branch=z9hG4bKzkdui3jfid
From: UserA<sip:UserA@gauss.coritel.it>;tag=938108741
To: UserB<sip:UserB@maxwell.coritel.it>
Server: Coritel SIP Server 1.0
Record-Route: <sip:gauss.coritel.it;lr>
QoS-Info: qos-domain=coritel.it;er-ingress=192.168.90.3;qos-mode=
unidirectional
Call-ID: 1234567001@eulero.coritel.it
Max-Forwards: 69
CSeq: 1 INVITE
Contact: <sip:UserA@151.100.37.131>
Content-Type: application/sdp
Content-Length: 148

v=0
o=UserA 2890844526 2890844526 IN IP4 eulero.coritel.it
s=Session SDP
c=IN IP4 151.100.37.131
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Message M3 (INVITE from Proxy 2 to User B):

```
INVITE sip:UserB@galileo.coritel.it:5060 SIP/2.0
Via: SIP/2.0/UDP maxwell.coritel.it:5060;branch=z9hG4bKkvjg1kk5gf
Via: SIP/2.0/UDP gauss.coritel.it:5060;branch=z9hG4bKzksdfse3re
Via: SIP/2.0/UDP eulero.coritel.it:5060;branch=z9hG4bKzkdui3jfid
From: UserA<sip:UserA@gauss.coritel.it>;tag=938108741
To: UserB<sip:UserB@maxwell.coritel.it>
Server: Coritel SIP Server 1.0
Server: Coritel SIP Server 1.0
Record-Route: <sip:gauss.coritel.it;lr>,<sip:maxwell.coritel.it;lr>
Call-ID: 1234567001@eulero.coritel.it
Max-Forwards: 68
CSeq: 1 INVITE
Contact: <sip:UserA@151.100.37.131>
Content-Type: application/sdp
Content-Length: 148

v=0
o=UserA 2890844526 2890844526 IN IP4 eulero.coritel.it
s=Session SDP
c=IN IP4 151.100.37.131
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Message M8 (200 OK from Proxy 2 to Proxy1):

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP gauss.coritel.it:5060;branch=z9hG4bKzksdfse3re
Via: SIP/2.0/UDP eulero.coritel.it:5060;branch=z9hG4bKzkdui3jfid
From: UserA<sip:UserA@gauss.coritel.it>;tag=938108741
To: UserB<sip:UserB@maxwell.coritel.it>;tag=181046899
Record-Route: <sip:gauss.coritel.it;lr>,<sip:maxwell.coritel.it;lr>
QoS-Info: qos-domain=coritel.it;er-egress=192.168.90.9;qos-mode=
unidirectional
Call-ID: 1234567001@eulero.coritel.it
CSeq: 1 INVITE
Contact: <sip:UserB@151.100.37.143>
Content-Type: application/sdp
Content-Length: 148

v=0
o=UserB 2890844527 2890844527 IN IP4 galileo.coritel.it
s=Session SDP
c=IN IP4 151.100.37.143
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Message M9 (200 OK from Proxy 1 to User A):

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP eulero.coritel.it:5060;branch=z9hG4bKzkdui3jfid
From: UserA<sip:UserA@gauss.coritel.it>;tag=938108741
To: UserB<sip:UserB@maxwell.coritel.it>;tag=181046899
Record-Route: <sip:gauss.coritel.it;lr>,<sip:maxwell.coritel.it;lr>
Call-ID: 1234567001@eulero.coritel.it
CSeq: 1 INVITE
Contact: <sip:UserB@151.100.37.143>
Content-Type: application/sdp
Content-Length: 148

v=0
o=UserB 2890844527 2890844527 IN IP4 galileo.coritel.it
s=Session SDP
c=IN IP4 151.100.37.143
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Message M14 (BYE from Proxy 2 to Proxy 1):

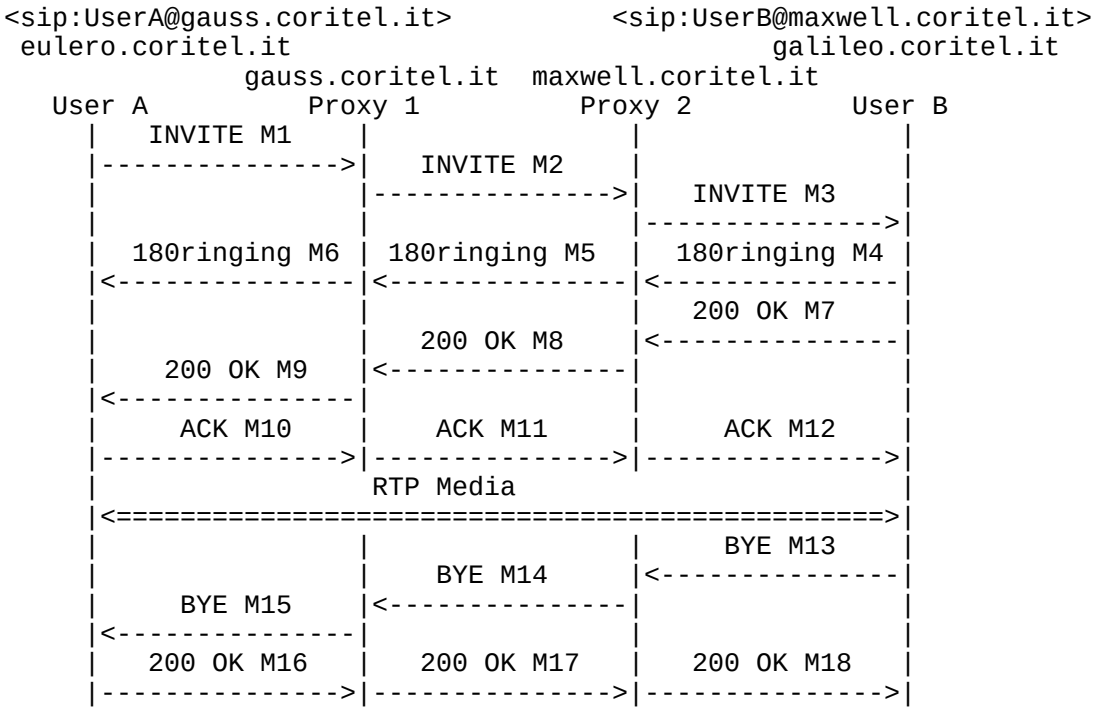
```
BYE sip:UserA@gauss.coritel.it SIP/2.0
Via: SIP/2.0/UDP maxwell.coritel.it:5060;branch=z9hG4bKljfds7df8s
Via: SIP/2.0/UDP galileo.coritel.it:5060;branch=z9hG4bKpinfjd6h3h
Route: <sip:gauss.coritel.it;lr>
From: UserB<sip:UserB@maxwell.coritel.it>;tag=181046899
To: UserA<sip:UserA@gauss.coritel.it>;tag=938108741
Server: Coritel SIP Server 1.0
Call-ID: 1234567001@eulero.coritel.it
Max-Forwards: 69
CSeq: 1 BYE
Content-Length: 0
```

Message M15 (BYE from Proxy 1 to user A)

```
BYE sip:UserA@eulero.coritel.it:5060 SIP/2.0
Via: SIP/2.0/UDP gauss.coritel.it:5060;branch=z9hG4bKlj2kl4jdik
Via: SIP/2.0/UDP maxwell.coritel.it:5060;branch=z9hG4bKljfds7df8s
Via: SIP/2.0/UDP galileo.coritel.it:5060;branch=z9hG4bKpinfjd6h3h From:
UserB<sip:UserB@maxwell.coritel.it>;tag=181046899
To: UserA<sip:UserA@gauss.coritel.it>;tag=938108741
Server: Coritel SIP Server 1.0
Server: Coritel SIP Server 1.0
Call-ID: 1234567001@eulero.coritel.it
Max-Forwards: 68
CSeq: 1 BYE
Content-Length: 0
```

Appendix E

Examples of Q-SIP messages in the successful reservation scenario keeping "provisional QoS state" in the messages are depicted in the picture hereafter. The messages are numbered from M1 to M18. Only the messages sent by the proxy servers are reported in detail.



Message M2 (INVITE from Proxy 1 to Proxy 2):

```
INVITE sip:UserB@maxwell.coritel.it SIP/2.0
Via: SIP/2.0/UDP gauss.coritel.it:5060;branch=z9hG4bKicd7op8ocx
Via: SIP/2.0/UDP eulero.coritel.it:5060;branch=z9hG4bKjasldjl2oi
From: UserA<sip:UserA@gauss.coritel.it>;tag=734578133
To: UserB<sip:UserB@maxwell.coritel.it>
Server: Coritel SIP Server 1.0
Record-Route: <sip: gauss.coritel.it;lr;qos-mode=unidirectional;qos-
domain=coritel.it;er-ingress=192.168.90.3;caller-media-
addr=151.100.37.131;caller-media-port=49172>
QoS-Info: qos-domain=coritel.it;er-ingress=192.168.90.3;qos-mode=
unidirectional
Call-ID: 1234567801@eulero.coritel.it
Max-Forwards: 69
CSeq: 1 INVITE
Contact: <sip:UserA@151.100.37.131>
Content-Type: application/sdp
Content-Length: 148

v=0
o=UserA 2890844526 2890844526 IN IP4 eulero.coritel.it
s=Session SDP
c=IN IP4 151.100.37.131
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Message M3 (INVITE from Proxy 2 to User B):

```
INVITE sip:UserB@galileo.coritel.it:5060 SIP/2.0
Via: SIP/2.0/UDP maxwell.coritel.it:5060;branch=z9hG4bKsdfpogir4r
Via: SIP/2.0/UDP gauss.coritel.it:5060;branch=z9hG4bKicd7op8ocx
Via: SIP/2.0/UDP eulero.coritel.it:5060;branch=z9hG4bKjasldjl2oi
From: UserA<sip:UserA@gauss.coritel.it>;tag=734578133
To: UserB<sip:UserB@maxwell.coritel.it>
Server: Coritel SIP Server 1.0
Server: Coritel SIP Server 1.0
Record-Route: <sip:gauss.coritel.it;lr;qos-mode=unidirectional;qos-
domain=coritel.it;er-ingress=192.168.90.3;caller-media-
addr=151.100.37.131;caller-media-port=49172>,
<sip:maxwell.coritel.it;lr;er-ingress=192.168.90.3;caller-media-
addr=151.100.37.131;caller-media-port=49172>
Call-ID: 1234567801@eulero.coritel.it
Max-Forwards: 68
CSeq: 1 INVITE
Contact: <sip:UserA@151.100.37.131>
Content-Type: application/sdp
Content-Length: 148

v=0
o=UserA 2890844526 2890844526 IN IP4 eulero.coritel.it
s=Session SDP
c=IN IP4 151.100.37.131
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Message M8 (200 OK from Proxy 2 to Proxy 1):

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP gauss.coritel.it:5060;branch=z9hG4bKicd7op8ocx
Via: SIP/2.0/UDP eulero.coritel.it:5060;branch=z9hG4bKjasldjl2oi
From: UserA<sip:UserA@gauss.coritel.it>;tag=734578133
To: UserB<sip:UserB@maxwell.coritel.it>;tag=234857984
Record-Route: <sip:gauss.coritel.it;lr;qos-mode=unidirectional;qos-
domain=coritel.it;er-ingress=192.168.90.3;caller-media-
addr=151.100.37.131;caller-media-port=49172>,
<sip:maxwell.coritel.it;lr;er-ingress=192.168.90.3;caller-media-
addr=151.100.37.131;caller-media-port=49172>
QoS-Info: qos-domain=coritel.it;er-egress=192.168.90.9;qos-mode=
unidirectional
Call-ID: 1234567801@eulero.coritel.it
CSeq: 1 INVITE
Contact: <sip:UserB@151.100.37.143>
Content-Type: application/sdp
Content-Length: 148

v=0
o=UserB 2890844527 2890844527 IN IP4 galileo.coritel.it
s=Session SDP
c=IN IP4 151.100.37.143
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Message M9 (200 OK from Proxy 1 to User A):

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP eulero.coritel.it:5060;branch=z9hG4bKjasldjl2oi
From: UserA<sip:UserA@gauss.coritel.it>;tag=734578133
To: UserB<sip:UserB@maxwell.coritel.it>;tag=234857984
Record-Route: <sip:gauss.coritel.it;lr;qos-mode=unidirectional;qos-
domain=coritel.it;er-ingress=192.168.90.3;caller-media-
addr=151.100.37.131;caller-media-port=49172>,
<sip:maxwell.coritel.it;lr;er-ingress=192.168.90.3;caller-media-
addr=151.100.37.131;caller-media-port=49172>
Call-ID: 1234567801@eulero.coritel.it
CSeq: 1 INVITE
Contact: <sip:UserB@151.100.37.143>
Content-Type: application/sdp
Content-Length: 148

v=0
o=UserB 2890844527 2890844527 IN IP4 galileo.coritel.it
s=Session SDP
c=IN IP4 151.100.37.143
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Message M14 (BYE from Proxy 2 to Proxy 1):

```
BYE sip:UserA@gauss.coritel.it SIP/2.0
Via: SIP/2.0/UDP maxwell.coritel.it:5060;branch=asdfhkjksdf3kjj2f
Via: SIP/2.0/UDP galileo.coritel.it:5060;branch=sdlfhk4f3gmpsdfo3
Route: <sip:gauss.coritel.it;lr;qos-mode=unidirectional;qos-
domain=coritel.it;er-ingress=192.168.90.3;caller-media-
addr=151.100.37.131;caller-media-port=49172>
From: UserB<sip:UserB@maxwell.coritel.it>;tag=234857984
To: UserA<sip:UserA@gauss.coritel.it>;tag=734578133
Server: Coritel SIP Server 1.0
Call-ID: 1234567801@eulero.coritel.it
Max-Forwards: 69
CSeq: 1 BYE
Content-Length: 0
```

Message M15 (BYE from Proxy 1 to user A)

```
BYE sip:UserA@eulero.coritel.it:5060 SIP/2.0
Via: SIP/2.0/UDP gauss.coritel.it:5060;branch=h2kerpuighber5d4l
Via: SIP/2.0/UDP maxwell.coritel.it:5060;branch=asdfhkjksdf3kjj2f
Via: SIP/2.0/UDP galileo.coritel.it:5060;branch=sdlfhk4f3gmpsdfo3
From: UserB<sip:UserB@maxwell.coritel.it>;tag=234857984
To: UserA<sip:UserA@gauss.coritel.it>;tag=734578133
Server: Coritel SIP Server 1.0
Server: Coritel SIP Server 1.0
Call-ID: 1234567801@eulero.coritel.it
Max-Forwards: 68
CSeq: 1 BYE
Content-Length: 0
```

References

[1] G. Camarillo et al. "Integration of Resource Management and SIP", IETF Internet Draft <draft-ietf-sip-manyfolks-resource-07.txt>, April 2002, Work in Progress.
[2] IETF WG NSIS - Next Step In Signaling http://www.ietf.org/html.charters/nsis-charter.html
[3] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, " SIP: Session Initiation Protocol", IETF RFC 3261, June 2002.
[4] 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.
[5] S. Salsano " COPS Usage for Diffserv Resource Allocation (COPS-DRA) ", draft-salsano-cops-dra-00.txt, October 2001, work in progress
[6] CoRiTeL The Q-SIP project http://www.coritel.it/projects/qsip
[7] S. Donovan, "The SIP INFO method", RFC 2976, October 2000.
[8] J. Rosenberg, H. Schulzrinne, "Reliability of Provisional Responses in the Session Initiation Protocol (SIP)", RFC 3262, June 2002.

Author Information and Acknoledgements

Special thanks to Jocelyn Fiorina for his comments and suggestions and for the work on the prototype implementation for the previous version of the draft.

Luca Veltri
CoRiTeL - Consorzio di Ricerca sulle Telecomunicazioni
Via Anagnina, 203
00040 Roma - ITALY
email: veltri@coritel.it

Stefano Salsano
DIE - University of Rome "Tor Vergata"
Viale Politecnico, 1
00133 Roma - ITALY
email: stefano.salsano@uniroma2.it

Donald Papalilo
CoRiTeL - Consorzio di Ricerca sulle Telecomunicazioni
Via Anagnina, 203
00040 Roma - ITALY
email: papalilo@coritel.it