Using SIP Extensions for Signalling in Stream Service System

Rami Lehtonen, Jarmo Harju, Petteri Heinonen, Jani Peltotalo and Sami Peltotalo
Tampere University of Technology, Telecommunications Laboratory
Email: rampe@cs.tut.fu, harju@cs.tut.fi, hl57056@cs.tut.fi, p154184@cs.tu.fi, p154183@cs.tut.fi

Key words: SIP, XML, bandwidth broker

Abstract: This paper describes some aspects of signalling in an access network independent service control system, which can be used for instance to order stream based personalized services for a number of terminals, wired or wireless, with different capabilities. Functions of the control system include the handling of capability negotiations and service initiation between the client and the media server system. Overview of the system is given and special attention is paid to the use of SIP and SIP extensions with XML content for managing the parameter negotiations. Resource reservation is also made possible to the service control system by making use of the bandwidth broker of the network service provider.

1. INTRODUCTION

The adaptation of IP-based networking paradigm in practically all types of communications networks opens huge possibilities for truly network independent provision and access of many kinds of services. While the Internet may remain as a global open source for information, teleoperators and content providers can use the same basic technology to create specialized services for registered subscribers. These service specific virtual private networks (SVPNs) are a combination of customer related databases, service databases, and network connections supporting sufficient security

and QoS features. However, before we can talk about access network independent service system, we need a control system that adapts the service to the different access networks and manages the user information within the SVPN.

The stream service architecture described in this paper is based much on the earlier work with the stream services done in [1] and [2]. The main idea is to concentrate on the signalling methods used by one key element of the SVPN concept, the destination entity. To explain the signalling needs of a destination entity, we will first relate it to the concept of a client, which has a twofold role in the system. Firstly, the client is an entity, which is connected to the SVPN system through some user interface, e.g. a web browser. This entity is able to browse and order some services through the SVPN system. We call this entity a subscriber. Secondly, the client can be considered as the entity, which is responsible for receiving the subscribed service. This entity is called the destination entity. These two different substances of a client may be completely apart, e.g. they may reside in different hosts in different networks. For example, the subscriber might be a mobile phone and the destination entity might be a IP-capable future TV. This paper focuses solely on the destination entity and its signalling needs, although an overview of the whole SVPN concept is also given.

From the destination entity's point of view, there are three objects with which signalling is needed: SVPN's core control system, bandwidth broker and media server. The SVPN's control system is the heart of the whole SVPN concept, and signalling is needed by the control system to inform the destination entity about an incoming service. The bandwidth broker is not necessarily part of the SVPN system, but rather a service provided by the network operator for managing QoS related properties, such as bandwidth on demand. The media server is the entity, which eventually delivers the service (e.g. a video stream). This document focuses on the two former cases, because the signalling between the destination entity and the media server is usually dependent on service type and media server characteristics, while the signalling between the SVPN's core and the destination entity can be more service independent. The same is also true for signalling between the destination entity and the bandwidth broker.

2. ARCHITECTURE OF THE SYSTEM

The architecture of the system consists of six parts: Browser, CORBA based ordering system, Service Manager, Bandwidth Broker, Media Server and Destination Entity. Figure 1 gives a view of the parts involved in the architecture.

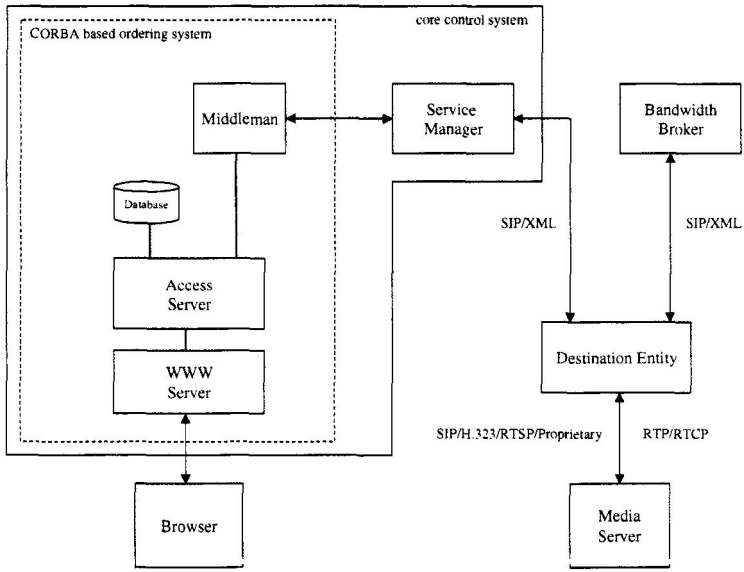

Figure 1. Architecture of the service control system

A subscriber uses a web browser to get into the service provider's web page. The CORBA based ordering system comprises a WWW Server, an Access Server (ACS) and a Middleman. The WWW-server handles communication with the subscriber and collects the needed service information. The Access Server is a set of functionalities including user and device management, service management, billing and access control. The Middleman's job is to transfer order and registration information between the ACS and the Service Manager. The Service Manager's task is to communicate with the ordering system (ACS) and the Destination Entity, and together with the ordering system it forms the SVPN's core control system. The Service Manager gets registering information from the Destination Entity and mediates it to the ordering system. The other part of the Service manager's responsibility is to negotiate about a set of service parameters with the Destination Entity when it gets the order request from the ordering system. After the parameter negotiation, the Service Manager informs the ordering system about the order status and it might also ask the user's confirmation on some parameters. The Bandwidth Broker then reserves the requested amount of bandwidth to the network between the Media Server (MS) and the Destination Entity. As can be seen from Figure 1, the signalling is handled by the Session Initiation Protocol (SIP) [3] and it uses extensible Markup Language (XML) [4] to describe the content. The Destination Entity, e.g. an IP-TV, gets the ordered stream from the Media Server by using appropriate protocols like RTSP/RTP.

3. THE ROLE OF SIP IN THE SYSTEM

The Session Initiation Protocol (SIP) is an application layer control protocol that is normally used for creating, modifying and terminating sessions between one or more participants. As such, it is a perfectly good candidate for a signalling protocol between the Media Server and the Destination Entity. Yet, SIP has been designed to be a very flexible protocol, with the possibility for other types of uses as well. In our system we apply SIP also in a very special way. Instead of managing sessions, SIP is used for forwarding and negotiating parameters between the Service Manager and the Destination Entity, and further between the Destination Entity and the Bandwidth Broker. All SIP messages are transported over TCP/IP. Actual media session is created between the Destination Entity and the Media Server afterwards, but this signalling, even though it might be handled with SIP, is not described further in this paper. Because of our special use of SIP, we have not implemented the whole protocol. We take some parts from RFC 2543 (SIP: Session Initiation Protocol) [3] and use also an extension named SIP Extension for Instant Messaging [5].

3.1 SIP Messages

A SIP message is either a request from SIP User Agent Client (SIP UAC) to SIP User Agent Server (SIP UAS), or a response from SIP UAS to SIP UAC. In our system the Service Manager and the Destination Entity acts both as a SIP UAC and a SIP UAS, while the Bandwidth Broker is only a SIP UAS. We utilize following messages:

- MESSAGE
- REGISTER
- OPTIONS
- 200 OK
- 603 Decline
- 606 Not Acceptable

The details of these messages are explained in Subsection 3.2. MESSAGE is adopted from the SIP Extensions and the others from the RFC 2543. The first three messages are requests and the others are responses. All these messages can transport content of any type. In our case the content type is XML. The message structure is based on the RFC 2543 and is the same for all messages. The structure and message example is shown below [5].

```
SIP message = Request-Line | Statues-Line
              message-header
                CRLF (an empty line)
                message-body  (optional)
```

```
MESSAGE sip:user1@user1pc.domain.com SIP/2.0
Via: SIP/2.0/UDP user2pc.domain.com
To: sip:user1@domain.com
From: sip:user2@domain.com;tag=ab8asdasd9
Contact: sip:user@user2pc.domain.com
Call-ID: asd88asd77a@1.2.3.4
CSeq: 1 MESSAGE
Content-Type: text/plain
Content-Length: 29

My name is User2, not Watson.
```

3.2 Meaning of SIP messages

In our stream service system we added new functionality to a SIP User Agent. We have designed a very simple protocol for the order negotiation using a SIP extension, a MESSAGE request.

The Destination Entity uses a REGISTER request to register itself to the ordering system via Service Manager (see Figure 2). The message body contains the Destination Entity's capability set, e.g. the screen size and the media codecs. The ordering system can accept the registration with a 200 OK response, or decline it with a 603 Decline response, e.g. if the Destination Entity has no access rights to that server.

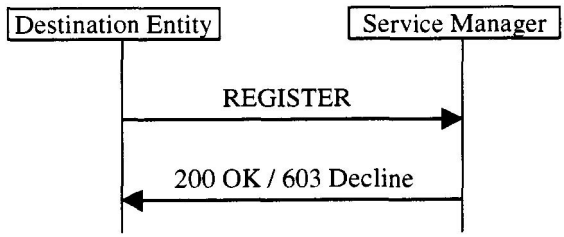

Figure 2. Registration

After the registration the ordering system knows that the Destination Entity is online and capable of receiving and negotiating of incoming orders.

Also when the Destination Entity is shut down it sends an empty REGISTER request, in which the expires header field contains a zero value.

The flowcharts in Figure 3 describe the order negotiation. There are two possible scenarios in this negotiation. The first one is that all goes well and the order succeeds. The second one happens when some parameters related to the order were unacceptable or it was not possible to carry out the service in the Destination Entity for some reason. This means that the order didn't succeed.

The Service Manager sends a MESSAGE request containing negotiable parameters, for example the bit rate and MIME type of the stream. The Destination Entity checks them and possibly reserves bandwidth from the Bandwidth Broker and then replies to the Service Manager. A 200 OK response contains the set of the parameter values that the Destination Entity has chosen. Next the Service Manager sends a new MESSAGE request containing more parameters in the message body, e.g. the media file name and Media Server's URL. The idea of sending two MESSAGE request comes from the fact that after the first MESSAGE, the Service Manager can be sure that the Destination Entity is able to handle the service, and then in the second MESSAGE it can include all the sensitive information that is needed to proceed further with the service delivery. The Destination Entity finally replies to this with a 200 OK response, whose message body is empty.

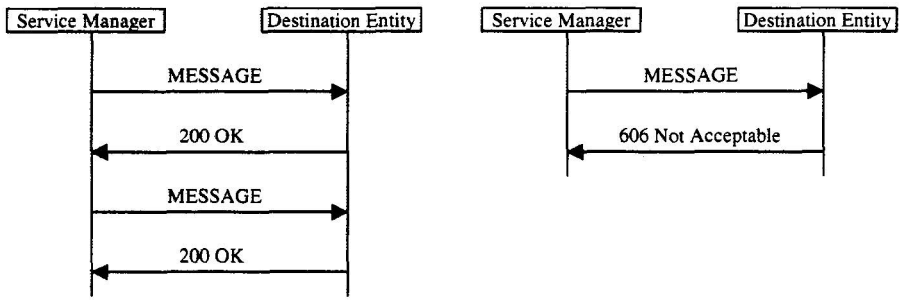

Figure 3. Parameter negotiation

In the second scenario (on the right-hand side) the Destination Entity did not accept the order proposed by the Service Manager. The Destination Entity replies in this case with a 606 Not Acceptable response. The reason for denying the service can be placed in the warning header field.

One optional request in our system is OPTIONS. The ordering system could query (via Service Manager) the Destination Entity's capabilities by

using this request. The Destination Entity replies with 200 OK response, which contains a special capability set in the message body (see Fig. 4).

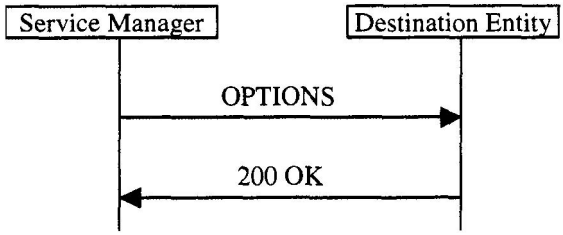

Figure 4. Query of capabilities

SIP signalling between the Destination Entity and the Bandwidth Broker is explained in Section 5.

4. XML CONTENT IN MESSAGE BODY

The common way is to carry Session Description Protocol (SDP) information in SIP messages when establishing multimedia sessions and negotiating parameters. But SDP is not the only way to transmit multimedia session information, one can also use e.g. XML for this purpose. In our case, one of the reasons in favour of XML is that SDP is too inflexible regarding the use of the fields specified in it. We also want to include many values for certain parameters inside one message and our parameters might have some relations to each other, e.g. "if screen size is X then the required bandwidth is Y". All our messages, which include the body part, have the content type set to text/xml. XML data has a certain format, which is defined in the Document Type Definition (DTD). By using a DTD one can make a parser, which understands the data that follows this DTD format. One very simple DTD that we could use in our system is shown below.

```
<!ELEMENT ORDER (NETWORK, MEDIA)>
<!ELEMENT NETWORK (PARAM|LIST-PARAM)*>
<!ELEMENT MEDIA (PARAM|LIST-PARAM)*>
<!ELEMENT PARAM (#PCDATA)>
<!ELEMENT LIST-PARAM (VALUE)*>
<!ELEMENT VALUE (#PCDATA)>
<!ATTLIST PARAM
  NAME CDATA #REQUIRED
<!ATTLIST LIST-PARAM
  NAME CDATA #REQUIRED
```

By using the previous DTD we can form the XML data which contains information about the order. The following example does not contain all the necessary parameters that we are using, just a few of them.

```
<!DOCTYPE ORDER SYSTEM "order.dtd">
 <ORDER>
   <NETWORK>
     <LIST-PARAM NAME="bandwidth">
       <VALUE>500</VALUE>
       <VALUE>100</VALUE>
     </LIST-PARAM>
     <PARAM NAME="MSIP">130.230.33.22</PARAM>
   </NETWORK>
   <MEDIA>
     <PARAM NAME="mediaFile">video.mov</PAM>
     <PARAM NAME="MSURL">
       rtsp://mediaserver.com/videos/</PARAM>
   </MEDIA>
 </ORDER>
```

5. USING A BANDWIDTH BROKER

5.1 Role of the Bandwidth Broker

Bandwidth Broker (BB) is a concept by which the customer of an Internet Service Provider (ISP) can affect the QoS properties of his connection. BB's internal structure and functionality is not in the scope of this paper. From our point of view it is used for reserving bandwidth for a stream, and possibly to balance the load between Media Servers. The bandwidth reservation is initiated by the Destination Entity based on the parameters it gets from the Service Manager, but the user (subscriber) can include a demand, desire or denial of the bandwidth reservation in his subscription. In the case of demand, the streaming is started only if the reservation of the bandwidth has succeeded. In the case of desire, reservation is tried, but even if it doesn't succeed, the streaming is started. The user can also forbid the Destination Entity to reserve resources from the BB. In this case the Destination Entity is not allowed to make any reservations.

5.2 Messages between Destination Entity and Bandwidth Broker

The communication between the Destination Entity and the BB is handled via SIP mechanisms, by carrying the reservation information in the XML-payload. All the defined SIP methods for this reservation use TCP transport. The following messages from the SIP and SIP extensions are used:

- MESSAGE (for reservation request)
- 200 OK (for reservation response, successful reservation)
- 603 Decline (for reservation response, reservation can not be fulfilled)

The Destination Entity starts by sending the reservation request, SIP MESSAGE, which includes the reservation parameters. The following list shows the parameters that must be sent by the Destination Entity to the BB in order to reserve bandwidth:

- Source IP address (=Media Server's IP address)
- Destination IP address (=Destination Entity's IP address)
- UserID (to identify user in BB)
- AccountID (for billing purposes)
- Bitrate to be reserved
- Reservation starting time
- Reservation ending time (optional)

IP addresses are clearly needed for BB so that it can reserve bandwidth by identifying the flow in the edge routers. UserID and AccountID are for authentication and billing purposes. It must be noted that the UserID might be different from the SVPN's UserID, because the service operator of the SVPN might be different than ISP, which controls the BB. The bit rate, reservation starting time and ending time should be included in the reservation request. Optionally the ending time might be left out from the reservation and the end of the reservation could be signalled to the BB when needed. All these reservation parameters are encoded in XML, which implements a special RAR-DTD (Resource Allocation Request-DTD), based somewhat on the issues defined in [6]. This DTD is defined as follows:

```
<!ELEMENT rar (peers, user, stream)>

  <! ELEMENT peers (host, host)>
```

```
<!ELEMENT user (userID, accountID)>
<!ELEMENT stream (bitrate, start-time, end_time?)>

<!ELEMENT host (host_ip, host_name?)>
  <! ATTLIST host host_type (server l client) #REQUIRED>

<!ELEMENT host_ip (#PCDATA)>
<!ELEMENT host-name (#PCDATA)>
<!ELEMENT userID (#PCDATA)>
<!ELEMENT accountID (#PCDATA)>
<!ELEMENT bitrate (#PCDATA)>
<!ELEMENT start-time(#PCDATA)>
<!ELEMENT end-time (#PCDATA)>
```

The XML fulfilling this DTD is packed in the SIP-message payload and then sent to the BB. The BB parses the XML and makes the decision whether it can reserve the requested bandwidth or not.

When the BB has reserved the bandwidth (or made the decision not to reserve it, e.g., because of insufficient network resources or insufficient privileges of the user etc.), it sends the reservation response to the Destination Entity. If the reservation was successful, the response is a SIP message 200 OK. In unsuccessful case a message 603 Decline is sent. In both cases, the SIP message carries a simple XML payload, which implements a special RAA-DTD (Resource Allocation Answer-DTD) [6] and contains the following parameters:

- Reserved bandwidth
- Available bandwidth

The SIP response indicates whether the reservation request was successful or not. With the reserved and available bandwidth parameters the BB can inform the Destination Entity for reserved and available bandwidth and their interpretation depends on the SIP response. In case of successful reservation (200 OK), the BB can use the reserved bandwidth parameter to inform the Destination Entity about the amount of reserved bandwidth and the available bandwidth can be used by BB to advertise that there is even more bandwidth available in case the Destination Entity would like to take advantage of it. In case of unsuccessful reservation (603 Decline), the reserved bandwidth is set to zero and with the available bandwidth the BB can report what is the maximum available bandwidth, which can be reserved at the time of the request. By setting the available bandwidth parameter to zero, the BB can also inform the Destination Entity that it can't handle any

reservations just now. RAA-DTD is defined as follows:

```
<!ELEMENT raa (avail-band, res_band)>

  <!ELEMENT avail_band (#PCDATA)>
  <!ELEMENT res_band (#PCDATA)>
```

5.3 Destination Entity's behaviour in different cases

The Destination Entity's behaviour in bandwidth reservation depends on the user given reservation parameter. In case where the user demands the reservation of bandwidth along with the actual service, the Destination Entity gets upper and lower limits for the possible bandwidths. These limits are set by the user and/or the Destination Entity, when it registers itself to the ordering system. The Destination Entity first tries to reserve the bandwidth defined by the upper limit. If this faults, a new reservation is done with value that the BB reported to be available. This happens only in case that reported available bandwidth is more than the lower limit. If it is not, the Destination Entity gives up the reservation and informs the user that the stream service can't be delivered because of the bandwidth reservation problems.

Also in the case where the user has desired the reservation of bandwidth, it must give limits for the bandwidth to be reserved. The Destination Entity functions similarly as in the previous case, except that streaming is started even if the bandwidth reservation fails. The last option for the user is to deny the bandwidth reservation totally.

6. CONCLUSIONS

This paper describes the use of SIP extensions with XML content for signalling purposes in a system, which provides streaming services for subscribers with different network access techniques. To facilitate the management of the system, a flexible parameter negotiation mechanism is essential. Negotiations and information exchange takes place between the order system and the destination entity, as well as between the destination entity and the Bandwidth Broker. We have shown that the basic SIP together with some of its extensions provides suitable protocol mechanisms to handle this information exchange. This is convenient as SIP can also be used as the signalling mechanism for the ordered media stream.

REFERENCES

[1] Rami Lehtonen and Jarmo Harju, “Access Network Independent Service Control System for Stream based Services,” EUNICE2000 Innovative Internet Applications proceedings, pp. 23-30, September 2000.

[2] Rami Lehtonen, Petteri Heinonen, Jani Peltotalo, Sami Peltotalo, Jarmo Harju and Veikko Hara, ”Implementation of a Flexible Control System for Launching Stream Services Independently of the Access Network,” GLOBECOM2000 Service Portability workshop, December 2000.

[3] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, “SIP: Session Initiation Protocol,” Request for Comments 2543, Internet Engineering Task Force, March 1999.

[4] T. Bray, J Paoli, and C.M. Sperberg-McQueen, “Extensible Markup Language (XML) 1.0,” W3C Recommendation, February 1998.

[5] J. Rosenberg, D. Willis, R. Sparks, B. Campbell, H. Schulzrinne, J. Lennox, B. Aboba, C. Huitema, D. Gurle, and D. Oran, “SIP Extension for Instant Messaging,” Internet Draft, Internet Engineering Task Force, June 2000.

[6] Ben Teitelbaum and Phil Chimento, “Qbone Bandwidth Broker Architecture,” Work in Progress, June 2000. http://qbone.internet2.edu/bb/bboutline2.html