Ann. Telecommun. (2009) 64:175–185
DOI 10.1007/s12243-008-0082-x

A next-generation service overlay architecture

Emmanuel Lavinal · Noëmie Simoni · Meng Song · Bertrand Mathieu

Received: 6 March 2008 / Accepted: 5 December 2008 / Published online: 16 January 2009

Abstract The rapid evolution of next-generation networks and, in particular, fixed mobile convergence infrastructures raises the issue of providing personalized services adapted to the user's context such as its device, access network, preferences, or quality of service (QoS) requirements. To design such value-added services, one solution consists in composing dynamically distributed service entities. In this paper, we propose a service overlay architecture in which a service level path is dynamically established to fulfill the user's requirements. In order to meet this goal, two main issues have to be considered: service components discovery and service path management (i.e., setup, reconfiguration, release). The former issue is addressed based on a peer-to-peer approach in which QoS features are integrated in service lookup. For the latter issue, we rely on the Session Initiation Protocol to automate the setup of the service composition as well as its adaptation in case of perturbations (e.g., user switching device or service component failure).

E. Lavinal · N. Simoni (✉)
TELECOM ParisTech,
46 rue Barrault,
75634 Paris cedex 13, France
e-mail: simoni@telecom-paristech.fr

E. Lavinal
e-mail: lavinal@telecom-paristech.fr

M. Song · B. Mathieu
France Telecom R&D,
2 av. Pierre Marzin,
22300 Lannion, France

M. Song
e-mail: meng.song@orange-ftgroup.com

B. Mathieu
e-mail: bertrand2.mathieu@orange-ftgroup.com

Keywords Next-generation services · QoS-based service discovery · SIP-based service paths

1 Introduction

The deployment of a next-generation network (NGN) capable of carrying any and all services independently of the underlying transport technologies raises inevitably the issue of the design and development of these services. While many work has been conducted at the transport layer, few has been carried out at the service layer. For instance, most of the research and development done currently in the IP Multimedia Subsystem (IMS) addresses the transport and the core IMS layers (home subscriber server and call session control functions), leaving behind the services. Therefore, taking into account this NGN context, it is essential to propose a service architecture which offers personalized services that can be accessed ubiquitously and continuously and that can adapt to the user's environment (location, device type, preferences, etc.). We designate such services as next-generation services (NGS).

The design of next-generation services should address mobility, heterogeneity, and, most importantly, user-centric issues. By mobility, we mean both spatial (e.g., a user changing access point) and temporal (e.g., recovering a session previously opened) mobility. Heterogeneity includes access networks, devices as well as service providers heterogeneity. In addition to these two dimensions, next-generation services should also consider a user-centric dimension (cf. Fig. 1). As a result, the service is not only provided according to system or network constraints but also and mainly according to constraints imposed by the user such as its profile, preferences, and context. Therefore, a service is designed according to user requirements as

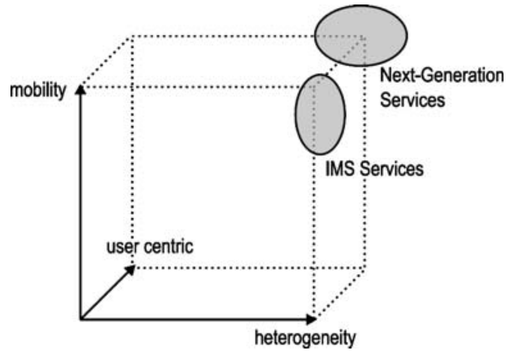

Fig. 1 The fundamental dimensions of next-generation services

much as systems requirements. A simple example of user-centric service is a messaging service that can be used with a vocal interface when it detects that the user is in its car. Another example is a streaming video service that delivers content based on the user's language, in a format adapted to its current device, and at a cost in line with the user's preferences. By combining these three dimensions in a service architecture, our goal is to provide context aware services in which service continuity is preserved that is a user's session persisting seamlessly despite perturbations (e.g., user mobility, device substitution).

In this paper, in order to build these NGS, we follow an approach based on a service-oriented design where the "service" concept is considered as the basic element to conceive and develop distributed applications. A global service can thus be built by dynamically composing modular independent services. The target applications that we consider are mainly multimedia stream-oriented communications in which a data flow is processed by intermediate service components (e.g., transcoding, encryption, caching, etc.). One of the main characteristics of these applications is that they rely on a form of communication in which quality of service (QoS) plays a crucial role. For example, when selecting a transcoding service between two audio codec, it is essential that it has enough processing capacities at a given time to be able to process all of its input audio streams without generating significant delay.

Creating dynamically a global service from other modular service components should allow providers to rapidly develop and deploy new value-added services. Moreover, it facilitates the dynamic and adaptive management of the global service such as personalization (e.g., stream a video with the user's language subtitles), fault tolerance (e.g., replace a particular service component in case of failure), or QoS adjustment (e.g., adapt codec according to user's device).

The remainder of the paper is organized as follows. Section 2 presents the general approach we follow to design our service overlay architecture. Section 3 presents a peer-to-peer (P2P) solution for service lookup based on QoS features and Section 4 describes the Session Initiation Protocol (SIP) signaling to automate the setup and reconfiguration of the service overlay path. Finally, Section 5 concludes and presents future works.

2 General approach

The service architecture we propose is based on the concept of service overlay network that we designate as Virtual Private Service Network (VPSN). A VPSN is a network of service components which provides a personalized value-added service according to the user's requirements. It follows a "Node-Link-Network" model where the node represents a service component, the link represents a media connection between two service components, and the network represents the VPSN itself. A service component is basically a service that can be provisioned and that can process multiple sessions simultaneously (this number depending on the service's capacity). Many instances of a service component achieving a particular functionality can exist, each coming from different providers and having equivalent of different QoS. Figure 2 illustrates a VPSN example of a vocal SMS service composed of three service components.

Two main issues have to be considered in order to achieve this VPSN concept. First, it is essential to be able to dynamically discover and locate service components. This is important since the VPSN is setup dynamically at runtime according to the user's location, preferences, or QoS requirements. The second fundamental issue is related to the control of the VPSN itself that is mainly the signaling plane to establish, reconfigure, and release the media communications.

To address the first issue, we introduce the concept of Virtual Service Community (VSC) in which we group service components that are functionally equivalent. One of the main objectives of these communities is to facilitate and optimize the discovery process of service components that match specific criteria (particularly in terms of QoS). It is

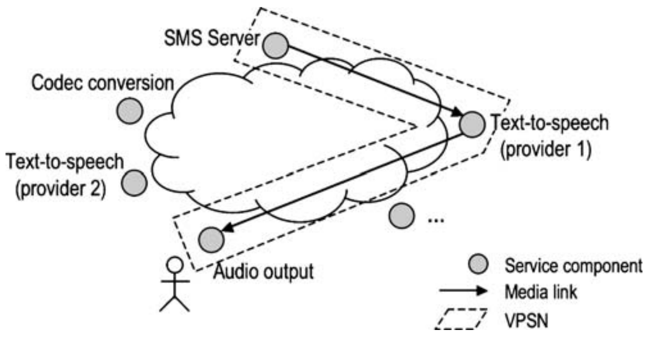

Fig. 2 VPSN example of a vocal SMS service

the responsibility of the VSC to manage the group members, their registration, discovery, and localization. As we will see in Section 3, we implement these communities as peer-to-peer groups where the different service components included in a VSC are managed in a decentralized way.

The second issue is related to the management of the VPSN itself. We dissociate here two levels of abstraction: the abstract service layer and the concrete service layer (cf. Fig. 3). The abstract service layer corresponds to the composition logic, in other words the functional workflow required to achieve the overall service. This level is independent of any service instance. At the concrete service layer, we have the different instances of the abstract service layer, i.e., the different VPSN, each dedicated to a specific user session. Therefore, in our context, a global service composed of different service components is accomplished by first, defining the abstract service layer (i.e., a workflow of VSC) and second, by instantiating this abstract service layer at runtime according to specific criteria (display resolution adapted to the user's device, user preference defining blacklisted providers, etc.).

To address this second issue, we adopted the SIP as the signaling protocol. We selected SIP because not only is it an Internet Engineering Task Force (IETF) standard to create, modify, and terminate multiparty sessions [1] but also SIP-based media services are increasingly implemented on the Internet. As for the description of the media session, we rely on SIP's natural companion, the Session Description Protocol (SDP) [2].

Several existing works deal also with application-level service composition in multimedia environments. For example, [3] proposes an architecture for the creation and recovery of service-level paths based on an overlay network of service clusters. In [4], the authors have done significant work on composing stream applications in peer-to-peer environments where centralized versus distributed algorithms are investigated taking into account QoS requirements. We can also quote [5] in which media delivery in mobile networks is achieved by specialized processing nodes that are aggregated along the communication path. However, these works do not address issues related to dynamic service discovery as well as automatic setup of the service-level path once it has been identified. Moreover, no standard signaling protocol is used to establish and control the end-to-end session.

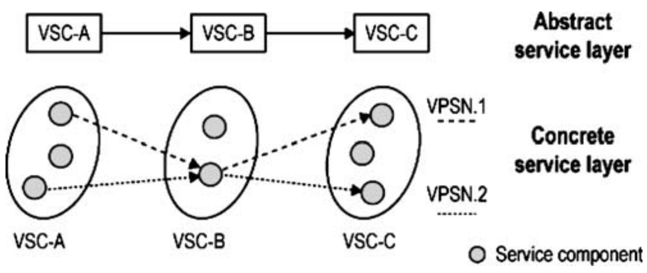

Fig. 3 VPSN/VSC approach to compose application services

3 QoS-based peer-to-peer service discovery

In this section, we present a service discovery and localization mechanism that is used within virtual service communities to find and locate specific service components. As we will see in Section 4, this discovery process will be used for both VPSN setup and reconfiguration. The hard part in this multiprovider service discovery process is finding service components in conformance with the desired QoS. To address this issue, we first start by explaining why we chose a peer-to-peer approach rather than a centralized directory and then we present the solution we propose to integrate QoS features in a structured peer-to-peer system based on a content addressable network (CAN).

3.1 Using a peer-to-peer approach

We have chosen a P2P approach to support VSCs, i.e., communities of peers providing functionally equivalent services with different QoS and possibly from different providers. We have decided to follow a P2P approach for mainly two reasons:

- Scalability: The P2P paradigm provides a scalable solution to publish and discover distributed resources. This is important in our context since we want to rely on a wide-area lookup service with multiple service components and multiple providers.
- Fault tolerance: Unlike a centralized directory, a P2P indexing system keeps running if one of the peers fails or leaves the system. This robustness is essential in a next-generation network where user mobility and service continuity have to be guaranteed.

However, current peer-to-peer lookup systems do not take into account the quality of the resource that is searched for. Several studies such as [6] or [7] exploit knowledge of underlying network characteristics to optimize the routing strategy and minimize network latency but none consider the quality of the peer service itself. Thus, it is not possible to lookup for a service filtering on its quality attributes (such as its delay or reliability).

3.2 Integrating QoS features into a CAN-based system

A CAN is a scalable distributed indexing system [8]. It is based on a distributed hash table which manages (key, value) pairs across the different nodes of the system. Each

CAN node stores a zone of the entire hash table as well as a small number of adjacent zones in the table. The basic operations performed on a CAN are the insertion, lookup, and deletion of (key, value) pairs. These operations are achieved by requests routed within the CAN. Requests for a particular key are routed by intermediate nodes towards the node whose zone contains that key. The value associated to a key can, for example, consist in the resource's description and IP address of the node hosting the resource (e.g., a file or in our context, a service).

In a CAN, the hash table is designed around a virtual d-dimensional Cartesian coordinate space. As a result, to store a pair (K_1, V_1), key K_1 is mapped onto a point P in the coordinate space using a uniform hash function. (K_1, V_1) is then stored at the node that owns the zone within which the point P lies.

The main reason why we chose a CAN for our service discovery process resides in its d-dimensional Cartesian coordinate space. No constraints are imposed by the CAN on the number of dimensions of the coordinate space, nor their unit of measurement. Therefore, we propose to use the different dimensions of the coordinate space as QoS metrics. Usually, a key is obtained from the name of the resource to find (e.g., a file name) and then mapped onto a point in the coordinate space. In our context, the resource to find is a service and we would like to discover it not only by its name but also and mainly according to its QoS characteristics. Using the dimensions of the coordinate space as QoS metrics will allow us to position the service component on the coordinate space according to its QoS characteristics and thus find it also according to these characteristics. This approach is illustrated in Fig. 4 where two dimensions are used to represent, respectively, reliability and availability metrics. Thus, a service component will be placed deterministically on the coordinate space according to its reliability and availability values. For example, a service component with 85% of reliability and availability will thus be registered within node B that holds the entire zone from 60% to 100% of availability and from 80% to 100% of reliability. Defining another QoS characteristic for the service consists simply in adding a third dimension to the coordinate space such as a delay metric. With three dimensions, a zone in the coordinate space is represented by a hypercube as illustrated in Fig. 5.

Defining a QoS-based coordinate space implies a fundamental property: Two points that are near to each other in the coordinate space represent two services that have close QoS values. In other words, services that map to points included into a same bounded zone of the coordinate space can be regarded as services that have close QoS. As we will see below, this is an important point for the service discovery process but it also implies another issue: The service indexes will not be distributed evenly among the CAN nodes. Indeed, in practice, the different service instances do not have qualities that are evenly distributed among the range of values, and therefore, some zones will aggregate most of the service indexes while others may have very few service indexes. In order to compensate this phenomenon, one mechanism can consist in tuning the scale of the different dimensions to optimize the placement of the services (for example, in Fig. 4, the scale of the two axes does not start at 0 but 20 since it does not make any sense to consider services which have a reliability and availability less than 20%). Another mechanism deals with zone splitting when a node joins the CAN. Instead of randomly choosing a point in the space, the node joining the CAN chooses several points and selects the one that belongs to the zone with the higher amount of registered service components (this has to be done by querying respectively the nodes owning the zones). In this way, each node joining the CAN will be efficiently assigned to half a zone containing a significant numbers of service keys.

Relying on this QoS coordinate space, the basic functionalities of the CAN (routing, construction of the coordinate overlay, and maintenance of the CAN overlay) are done as described in [8]. A CAN node learns and

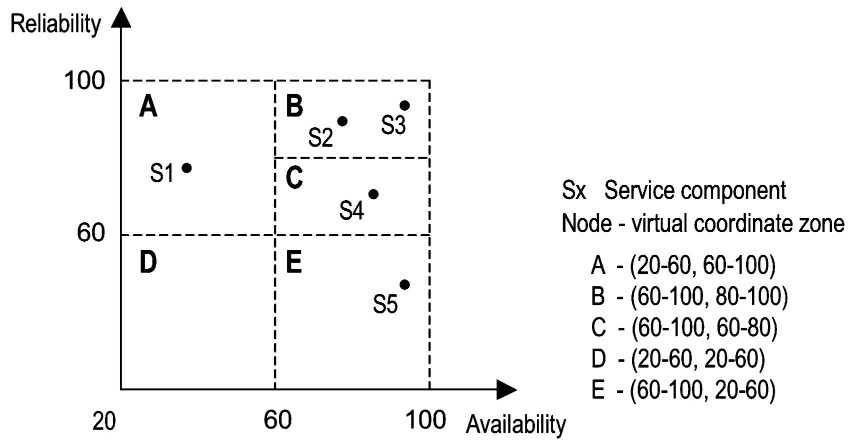

Fig. 4 Example of a CAN with a QoS-based two-dimensional coordinate space

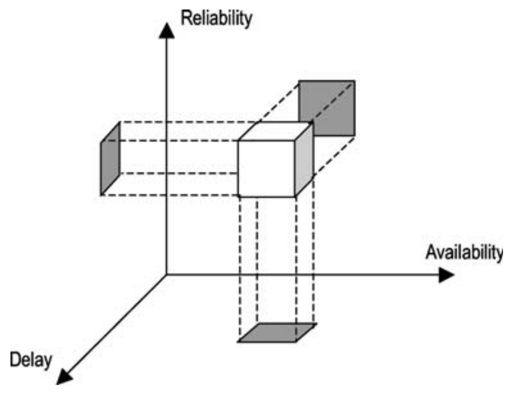

Fig. 5 Example of QoS-based three-dimensional coordinate space

maintains the IP address and virtual coordinate zone of each of its immediate neighbors in the coordinate space. Therefore, a node routes a message towards its destination by simply forwarding it to the neighbor whose coordinates are the closest to the destination coordinates. For example, in Fig. 4, if the node D requests the value of a key which maps to a point included in node's B zone, its request is first forwarded to node A and then to node B. Note that many different paths exist between two points in the coordinate space and thus if a node's neighbor were to crash, then the message can be routed along the next best available path. Again, in Fig. 4, D can also reach B by passing the message through nodes E and C.

A common problem when discovering services from their QoS characteristics concerns the desired accuracy level. This problem is amplified in our virtual coordinate space since two points match only if they have the exact same coordinates and thus identical QoS values. However, our CAN approach based on a QoS coordinate space offers a solution to this problem. Indeed, as we have previously mentioned, two services included into the same zone (or subzone) can be regarded as two services with close QoS values. Consequently, if a CAN node receives a query for a key that maps to its zone and if no entry exists for this key, then it can spontaneously enlarge the search space in order to find a service whose QoS is close to the one requested. For example, in Fig. 4, if node D seeks for a service with a QoS that maps to a point included in node's B zone and if this exact point does not have any entry, then B can widen the query and reply with services that have near by QoS such as S2 or S3. That mechanism is a direct consequence of the QoS-based coordinate space used to organize the service components indexes. It is interesting since, compared to randomly affected coordinates, it allows the discovery process to be much more flexible and efficient (e.g., being able to discover and locate a service component in a certain QoS range while another solution based on an "all-or-nothing" model would have returned an empty set).

4 SIP-based service overlay network

In addition to discovery and localization mechanisms, it is essential to rely on signaling protocols to establish, maintain, and release the end-to-end media stream between the different service components. We have identified three signaling models that can be used to control the service composition. Figure 6 illustrates these different approaches:

1. A dedicated controller, not part of the composition itself, manages the service path. This entity embeds the composition logic and controls the overall signaling.
2. A central point also controls the end-to-end service path but this entity is one of the endpoints of the path, such as the caller's device.
3. The service path is established and released in a distributed way; no central signaling point is required.

The first two models are based on a form of third party call control while the third model is totally distributed. The second and third models are also addressed in IETF's work in progress concerning the transcoding framework for the Session Initiation Protocol [9]. However, as we will see below, their approach is slightly different.

Taking into account these three models and relying on the SIP protocol, we will first present the dynamic setup of the service composition and second, its self-adaptation to environmental changes such as user mobility or service failure. The third subsection gives a recap and comparison of the three approaches.

4.1 Dynamic service overlay setup

We will detail the first and third signaling models of Fig. 6 since, from a signaling point of view, the second approach is very similar to the first one, the only difference being that

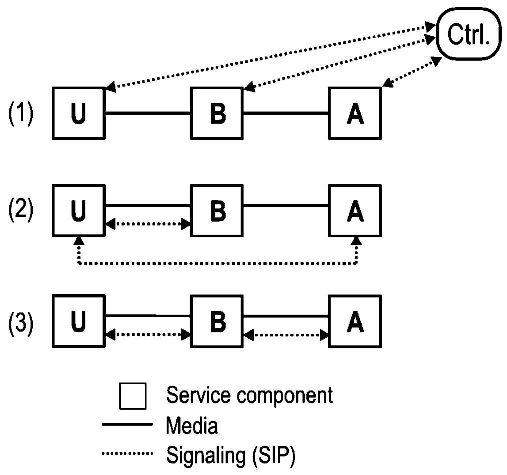

Fig. 6 Service-level path signaling models

the controller is implemented within an endpoint of the service path. To illustrate both of these approaches, we consider four service components: two end services (S.U and S.A) and two midservices (S.B and S.C). The service S.A can, for instance, produce a text flow which is transcoded into an audio flow by S.B, then processed by S.C to adapt the codec, and then delivered to S.U located on the user's device.

4.1.1 Centralized signaling

The SIP call flow achieving the composition of these four service components is illustrated in Fig. 7. The VPSN controller, which implements the setup algorithm, acts as a back-to-back user agent with S.U and as a third party call controller (3pcc) with S.A, S.B, and S.C.

When the user invokes the composite service, an INVITE (1) request containing its session description is sent to the VPSN controller. The VPSN controller then dynamically instantiates its abstract service layer. This process consists in finding at least one service instance in each community. As described in the previous section, this is done at runtime according to one or several criteria (e.g., required QoS) by sending a request within the QoS-aware CAN. Once the VPSN controller has obtained a concrete service-level path, it has to provision each service component; this is done by its 3PCC behavior.

The VPSN controller invites respectively the different service instances included in the service-level path. It first sends an INVITE (3) request containing the SDP data received in (1; i.e., the user's session description) to the second service component of the sequence. Notice that the SDP content included in the INVITE describes only one of the two media flows required to provision the midservice, the specification of the other media flow is provided by the ACK message sent afterwards.

When the VPSN controller receives a 200 OK response emanating from a midservice, the SDP content contains two media descriptions: one for each media that it has to process. Figure 8 presents an example of SDP data generated by the midservice S.B and sent to the VPSN controller (message 6 in Fig. 7). The VPSN controller extracts and splits these two media descriptions to generate, from the first one, an SDP answer intended for the previous service component (sent in an ACK message, e.g., 7 in Fig. 7) and from the second one, an SDP offer intended for the next service component in the service-level path (sent in an INVITE request, e.g., 8 in Fig. 7). The controller reiterates this process each time it receives a 200 OK response until it reaches the last service component of the sequence. When the last service component of the service-level path responds to the INVITE with a 200 OK, the controller simply sends an ACK to this last component as well as to the previous service component (messages 10 and 11 in Fig. 7).

At the end of this setup phase, each service component maintains state information only with the controller (from a signaling point of view) and with the components they exchange media with. For example, service S.B has a session description specifying its media connections with S.C and S.A, a signaling dialog with the controller but ignores the presence of S.U in the overall composition. Therefore, the VPSN controller is the only entity that has the complete view of the service-level path; it is responsible for the aggregation of the different SIP transactions to guarantee a global unique session.

Several differences exist between our proposition and the technique exposed in RFC 4117 to aggregate multiple transcoding services in series [10]. The first important difference is that the controller is not in the caller's agent but in a dedicated entity, thus belonging to Fig. 6 first model (not the second one). Therefore, the composition logic complexity is not hosted on the endpoints and the signaling on the user's device is very much reduced. The

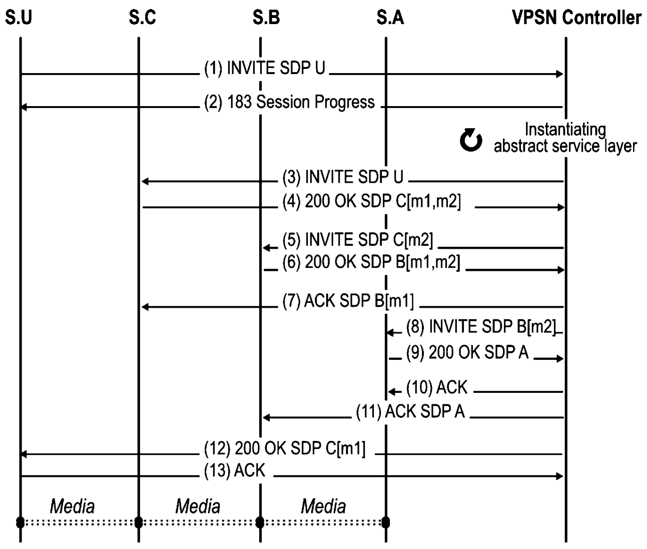

Fig. 7 SIP call flow to establish VPSN with dedicated controller

```
# Session description (text-to-speech service)
   v=0
   o=S.B 1376284526 1376289026 IN IP4 service-B.enst.fr
   s=vpsn01
   t=0 0
# Media description (intended for S.C)
   m=audio 3771 RTP/AVP 0
   c=IN IP4 service-B.enst.fr
   a=sendonly
# Media description (intended for S.A)
   m=text 5060 RTP/AVP 96
 c=IN IP4 service-B.enst.fr
 a=rtpmap:96 t140/1000
 a=recvonly
```

Fig. 8 Example of SDP content included in message 6 of Fig. 7

second difference lies within the control flow itself. In [10], each 200 OK is *acked* immediately (to prevent timeouts), thus requiring each time a reINVITE with the SDP data of the next service component of the sequence. In our proposition, we reduce the number of SIP messages by sending an ACK only when the controller has the complete session description of the service component (messages 7 and 11 in Fig. 7). No timeouts occur since this delay is very short, the midservices responding immediately to the invites (no manual answer).

4.1.2 Decentralized signaling

Based on the same four service components of the previous example, Fig. 9 illustrates the SIP call flow achieving the composition setup in a distributed fashion. Since there is no central entity that controls the different service components, the composition logic has to be conveyed in the SIP messages.

When the user wishes to access a global value-added service, it has first to instantiate its abstract service layer. This supposes that this information is known by the user's device or that it can delegate this task to another entity (e.g., a proxy that is included in the QoS-based CAN).

Once the caller has obtained a concrete service-level path, it can trigger the composition setup by sending an INVITE to the first service component with which it should establish a media connection. The body of this INVITE message includes not only the caller's session description but also an ordered list containing the other service components that should be provisioned. We express this content as in Fig. 10 where the first part of the message is the user's device SDP offer and the second part of the message is an Extensible Markup Language (XML) description representing the remainder of the service-level path.

The process of conveying both an SDP offer and a Uniform Resource Identifier (URI) list in an INVITE request is a mechanism also used in IETF drafts related to conference establishment where a conference server receives a list of participants in a SIP message [11]. This list of participants is specified in the XML resource list format [12]. The main difference is that in their case, the process is not reiterated, it stops at the conference server whereas in our case, the list is propagated and updated along the path of services. Another difference is that their URI list is unordered since it represents a set of independent participants.

On receiving an INVITE with a URI-list body, the midservice component generates another INVITE request towards the first SIP address contained in the URI list. The service component includes in the body of this message its own session description as well as the received URI list after having removed the first entry (cf. message 3 in Fig. 9). In addition, it replies with a session progress message to prevent timeouts. Since the service component generates a new INVITE (with in particular new From, To, and Call-ID header fields) before responding with a 200 OK, it is considered as a back-to-back user agent. This process is reiterated by every service component until the URI list is empty, indicating that the last service component of the sequence has been reached. Then, the 200 OK responses are propagated back to the first service component, containing each time the SDP answer describing the sender's media connection (cf. messages 7 through 11 in Fig. 9).

At the end of this setup phase, each service component maintains state information only with the components they exchange media with. For example, service S.B has a session description (both media and signaling) with S.C and S.A but ignores the presence of S.U in the overall composition. The caller has a view of the complete service

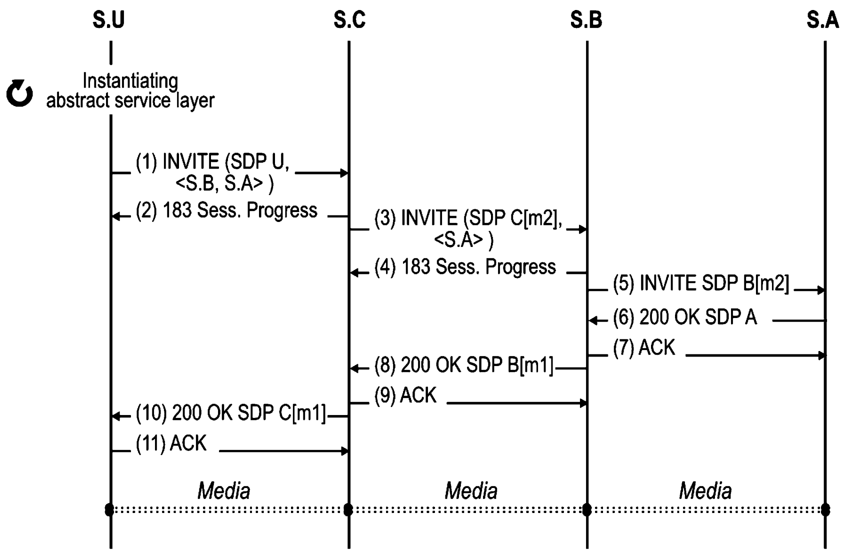

Fig. 9 SIP call flow to establish VPSN with a distributed approach

```
--boundary1
Content-Type: application/sdp
  v=0
  o=S.U 4134284526 4134289029 IN IP4 visitor22.enst.fr
  s=vpsn01
  t=0 0
  m=audio 8991 RTP/AVP 98
  a=rtpmap:98 G722/8000
  c=IN IP4 visitor22.enst.fr
  a=recvonly

--boundary1
Content-Type: application/service-path+xml
  <?xml version="1.0" encoding="UTF-8"?>
  <service-path xmlns="urn:enst:params:xml:ns:service-path">
     <ordered-sequence>
       <entry uri="sip:s-b@instant-texttospeech.com" />
       <entry uri="sip:s-a@mySmsProvider.com" />
     </ordered-sequence>
  </service-path>
--boundary1--
```

Fig. 10 Example of SIP body included in message 1 of Fig. 9

level path but, unlike the previous approach, it does not have any control over the service components it does not exchange media with. Therefore, it is the aggregation of the distributed service components' states that guarantees the global end-to-end session.

4.2 Service overlay self-reconfiguration

In this section, we address the dynamic reconfiguration of the service-level path when one of the service components cannot participate anymore in the VPSN (because, for example, it experiences QoS degradation or it is simply shutting down). Our objective is to allow the VPSN to adapt itself in order to maintain the end-to-end service.

In order to fulfill this goal, the first phase is addressed by the VSC concept whose role is to find one or more service components capable of replacing the first one. The second phase consists in moving the session to this new service component. We addressed the first phase in Section 3; we will thus only describe here the second phase.

Moving an active SIP session from one device to another raises issues currently addressed in an IETF draft on session mobility [13]. In that document, two transfer modes are defined: the mobile node control mode (MNC) and the session handoff (SH) mode. The first one is based on a third party call control while the second one uses the SIP REFER method which indicates that the recipient should contact another entity whose address is provided in the request. The main drawback of the MNC approach is that it requires the mobile node (i.e., the service component leaving the VPSN) to remain active, which is not possible in our context. As for the SH approach, it cannot be directly applied in our service path context. Indeed, in the centralized signaling approach, a service component cannot refer its neighbors (i.e., the service components with which it exchanges media with) since, from a signaling point of view, it is not aware of their existence and it maintains a SIP dialog only with the VPSN controller. As for the decentralized approach, a midservice has at least two media connections and thus if it leaves the service overlay network, it needs to refer multiple entities; this mechanism is not currently possible with the REFER method. Consequently, we will describe how we address these two issues.

4.2.1 Centralized signaling

In order to obtain a session mobility solution compliant with the first and second signaling models exposed in Fig. 6, we propose to combine the MNC and SH modes (as in [14]). A REFER request is sent by the service component leaving the overlay network to the new service component, but instead of "referring" it to its neighbors, it "refers" it to the VPSN controller which acts as the third party call controller of the session mobility process. The controller has to remain active once the session mobility has occurred but here, it is not a drawback since in any case it remains active during the entire end-to-end session. The controller is indeed responsible of establishing and releasing the service overlay network; it is thus logical that it should also update it.

We illustrate this reconfiguration process in Fig. 11 on three service components: two end services (S.U and S.A) and one midservice (S.B). Let us assume that the composition adaptation results in the replacement of the service component S.B1 by S.B2. In steps 1 to 3, S.B1

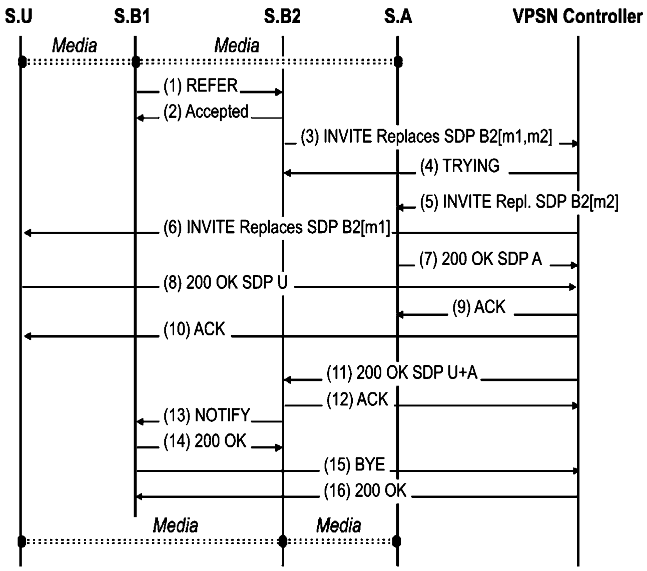

Fig. 11 SIP call flow for a dynamic VPSN reconfiguration (with controller)

refers S.B2 to invite the refer target (the VPSN controller) into a session. In this INVITE message, S.B2 sends its own SDP offer containing its two media descriptions (and possibly authentication credential from the original REFER request). In steps 5 to 10, the VPSN controller plays a third party call controller role by splitting these two media descriptions and (re)inviting, respectively, S.A and S.U with this new offer. Finally, in steps 13 to 16, S.B1 is notified that the REFER has succeeded and thus terminates its own SIP session, completing the session mobility process.

4.2.2 Decentralized signaling

In the decentralized signaling approach, a mobile service component cannot rely on a central controller to operate the session mobility process. In addition, a service component can have multiple media connections (such as a transcoding service) and thus if it leaves the service overlay network, all of its media connections have to be transferred. The problem is that presently, the REFER method does not allow multiple targets in the refer-to-header field. This is currently being addressed in IETF work in progress where an extension to the REFER method is being defined [15]. We rely on this document to achieve the session mobility process on service components having multiple media connections.

In Fig. 12, we illustrate the session mobility process (without any central controller) based on the previous example involving three service components. Let us assume again that the composition adaptation results in the replacement of the service component S.B1 by S.B2. In step 1, S.B1 requests a multiple refer to S.B2 containing the SIP addresses of the service components involved in both of its media connections (i.e., S.U and S.A in the example).

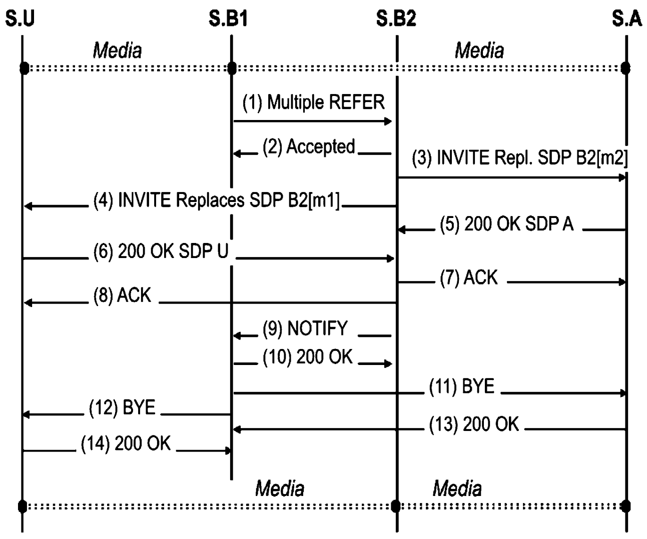

Fig. 12 SIP call for a dynamic VPSN reconfiguration (no controller)

Based on [15], an illustration of this multiple refer request is exposed in Fig. 13. The refer-to-header field contains a pointer (i.e., a content-ID uniform resource locator) that points to the body part carrying the refer targets URI list. This list is specified thanks to the XML format for representing resource lists [12] in which we have added a connection type attribute in order to be able to differentiate the media connections (in Fig. 13 example, we suppose that S.B1 and S.B2 are text-to-speech services).

Once S.B2 has accepted the multiple refer, it sends a (re)INVITE request to each SIP agent included in the refer target URI list (messages 3 and 4 in Fig. 12). These requests contain respectively an SDP offer of each of S.B2's media connection (and possibly authentication credential from the original REFER request). In steps 5 to 8, the normal INVITE procedure carries on and finally, in steps 9 to 14, S.B1 is notified that the REFER has succeeded and thus terminates its own SIP sessions for both of its media connections.

4.3 Comparative summary

In this section, we recap and compare the three signaling models presented in Fig. 6. For each of them, we review issues related to service discovery and signaling (e.g., the number of SIP messages required for the composition setup and reconfiguration). The results are exposed in Table 1.

The composition logic of the service overlay network, i.e., the abstract service layer, has to be known by the entity which is going to instantiate this logic (it can be known either by design or obtained at runtime from another entity). In the case of centralized signaling with a dedicated controller, the discovery and localization of the

```
REFER sip:s-B2@ft.example.com  SIP/2.0
From: <sip:s-B1@enst.example.com>;tag=33231
To: <sip:s-B2@ft.example.com>
Call-ID: fa84b4c76e71
CSeq: 2 REFER
Contact: <sip:s-B1@enst.example.com >
Refer-To: <cid:lo31jf07@local.example.com>
Refer-Sub: false
Require: multiple-refer, norefersub
Content-Type: application/resource-lists+xml
Content-Length: 362
Content-ID: <lo31jf07@local.example.com>

<?xml version="1.0" encoding="UTF-8"?>
<resource-lists xmlns="urn:ietf:params:xml:ns:resource-lists"
        xmlns:vpsn="urn:enst:params:xml:ns:connecType">
  <list>
    <entry uri="sip:s-U@example.com" vpsn:connecType="speech" />
    <entry uri="sip:s-A@example.org" vpsn:connecType="text" />
  </list>
</resource-lists>
```

Fig. 13 Example of SIP multiple REFER request

Table 1 Comparison of the three signaling models

	3pcc in controller	3pcc in caller	No 3pcc
Composition logic	Within controller	Within caller	Within caller
Service discovery for composition setup	Done by controller	Done by caller	Done by caller
Service discovery for session mobility	Done by mobile service component	Done by mobile service component	Done by mobile service component
SIP signaling	Centralized in controller	Centralized in caller	Distributed
SIP traffic to setup comp.	$3n$	$3(n-1)$	$3(n-1)$
SIP traffic on caller//callee to setup comp	3//3	$3(n-1)$//3	3//3
Session mobility method	Mix of SH and MNC	Mix of SH and MNC	Multiple Refer method
SIP traffic for session mobility	$9+3c$	$9+3c$ (except for session mobility on caller's device)	$4+5c$

service components are done by the controller itself whereas in the other approaches (3pcc in caller and decentralized signaling), these functions are accomplished by the caller. As for the reconfiguration of the service-level path, whichever signaling model, the service discovery is done by the component that is leaving the session (we designate it as the mobile service component). This is not mandatory but we think that since a service component cannot fulfill its contract anymore, it is its responsibility to find an alternate service component that can replace it.

Concerning the SIP traffic, we note n the total number of service components included in the service overlay network ($n \geq 2$) and c the number of media connections required by a particular service component ($c \geq 1$). For example, in Fig. 7, we have $n=4$ and $c=1$ (in the case of S.U and S.A) or $c=2$ (in the case of S.C and S.B). Notice that to simplify, in the total number of SIP messages, we have omitted any informational messages such as Session Progress and Trying responses.

As expected, to establish the VPSN, the SIP traffic increases linearly according to the number of service components whether there is a central controller or not. The difference between the signaling models is not very significant since there are only three more SIP messages in the controller approach (respectively $3n$ and $3(n-1)$ messages), which corresponds to the initial INVITE procedure from the caller to the controller.

As for the replacement of a service component at runtime, two different methods are used: a combination of the session handoff and mobile node control transfer modes for the 3pcc signaling approaches and a session handoff with multiple refer method for the distributed signaling approach. In both of these methods, the number of SIP messages required to achieve the session mobility process depends only on the number of media connections the mobile service component has. For example, if a service component has two media connections ($c=2$), the SIP traffic is almost equivalent in both approaches (15 messages with a 3pcc vs. 14 messages without). However, the distributed signaling approach requires that the mobile service component supports the multiple refer method. As for the 3pcc within the caller, we can highlight two drawbacks: First, all of the SIP traffic is going to transit by the user's device (this can be a problem on devices with limited energy and processing resources) and second, performing a session mobility on the user's device is going to be quite complicated and time consuming since all the other service components have to be notified so that they can update their SIP dialog's state.

As exposed in the introduction, no matter which signaling model is used, next-generation services should address mobility, heterogeneity, and user-centric issues. In the approach we have presented, mobility is achieved at the service layer by allowing the VPSN to reconfigure itself and thus move an active SIP session from one node to another. Heterogeneity is carried out by the session description protocol attributes (codec, bandwidth, etc.) which are negotiated during the connection establishment between the participants of the session. As for user-centric issues, they are mainly addressed in the service discovery process when instantiating the composition logic according to specific user criteria (the service path is then setup dynamically for each user session).

5 Conclusion and future work

An important issue related to the evolution of communication networks concerns the design of personalized robust services that can be accessed ubiquitously and that can adapt to the user's context such as its device or location. A possible approach to provide such services consists in dynamically establishing a service-level path composed of distributed modular services. To achieve this goal, two important issues have to be considered: a multiprovider service discovery and localization mechanism and a signaling plane to setup and maintain the end-to-end stream-oriented communication.

For the first issue, we proposed a QoS-aware peer-to-peer lookup system based on a content addressable network. The design of this CAN centers around a virtual multidimensional coordinate space in which service keys are mapped onto the space based on the service's QoS characteristics, thus natively considering QoS features in the discovery process. For the second issue, we relied on the SIP signaling protocol to enforce the composition of multiple services in series. More precisely, we proposed algorithms that automate SIP call flows to dynamically setup the overall service-level path as well as its reconfiguration thanks to session mobility mechanisms. Centralized versus decentralized signaling approaches were investigated.

Current works intend to simulate the QoS-aware CAN lookup system in order to evaluate the system's behavior (e.g., in terms of traffic overhead). We are also working on integrating the quality of the connectivity link into the discovery process. As for the SIP-based service composition, we have already implemented the centralized signaling approach and we are working on the decentralized one in order to obtain experimental results allowing us to compare the performance of both approaches. Moreover, we are studying the automation of more complex composition patterns such as parallel sessions.

References

1. Rosenberg J, Schulzrinne H, Camarillo G, Johnston A, Peterson J, Sparks R, Handley M, Schooler E (2002) SIP: Session Initiation Protocol. IETF RFC 3261, June
2. Handley M, Jacobson V, Perkins C (2006) SDP: Session Description Protocol. IETF RFC 4566, July
3. Raman B, Katz RH (2003) An architecture for highly available wide-area service composition. Comput Commun J 26(15):1727–1740, September
4. Gu X, Nahrstedt K (2006) On composing stream applications in peer-to-peer environments. IEEE Trans Parallel and Distrib Syst 17(8):824–837, August
5. Hartung F, Niebert N, Schieder A, Rembarz R, Schmid S, Eggert L (2006) Advances in network-supported media delivery in next-generation mobile systems. IEEE Commun Mag 44 (8):82–89, August
6. Zhao BY, Duan Y, Huang L, Joseph AD, Kubiatowicz JD (2002) Brocade: landmark routing on overlay networks. 1st International Workshop on Peer-to-Peer Systems (IPTPS), Cambridge, USA, March
7. Ratnasamy S, Handley M, Karp R, Shenker S (2002) Topologically-aware overlay construction and server selection. IEEE INFOCOM'02, New York, USA, June
8. Ratnasamy S, Francis P, Handley M, Karp R, Shenker S (2001) A scalable content-addressable network. ACM SIGCOMM'01, San Diego, United States, August
9. Camarillo G (2006) Framework for transcoding with the Session Initiation Protocol (SIP), draft-ietf-sipping-transc-framework-05 (work in progress). IETF Internet-Draft, November
10. Camarillo G, Burger E, Schulzrinne H, van Wijk A (2005) Transcoding services invocation in the Session Initiation Protocol (SIP) using third party call control (3pcc). IETF RFC 4117, June
11. Camarillo G (2006) The Session Initiation Protocol (SIP) Conference Bridge Transcoding Model, draft-ietf-sipping-transc-conf-03 (work in progress). IETF Internet-Draft, May
12. Rosenberg J (2007) Extensible Markup Language (XML) formats for representing resource lists. IETF RFC 4826, May
13. Shacham R, Schulzrinne H, Thakolsri S, Kellerer W (2007) Session Initiation Protocol (SIP) session mobility, draft-shacham-sipping-session-mobility-04 (work in progress). IETF Internet-Draft, July
14. Mani M, Crespi N (2006) Session mobility between heterogeneous accesses with the existence of IMS as the service control layer. IEEE International Conference on Communications Systems (ICCS), Singapore, October
15. Camarillo G, Niemi A, Isomaki M, Garcia-Martin M, Khartabil H (2007) Referring to multiple resources in the Session Initiation Protocol (SIP), draft-ietf-sip-multiple-refer-02 (work in progress). IETF Internet-Draft, November