A new Service Description Language as basis for Service Composition in SIP-based Peer-to-Peer infrastructures

A. Lehmann, U. Trick
Research Group for Telecommunication Networks
University of Applied Sciences Frankfurt/Main, Germany
e-mail: {lehmann, trick}@e-technik.org

W. Fuhrmann
University of Applied Sciences Darmstadt, Germany
e-mail: w.fuhrmann@fbi.h-da.de

B. Ghita
Centre for Security, Communications, and Network research
University Plymouth, UK
e-mail: bogdan.ghita@plymouth.ac.uk

Abstract—The increasing demand for value added services is a driver for telecommunication companies to find simple and cost efficient solutions for automated service creation, as bespoke service development for small customer groups is not economically viable. The aim of this paper is to propose a peer-to-peer based environment that supports deployment and discovery of value-added services. The identified services can subsequently be aggregated to fulfil more complex individual demands. In this paper a new service description language is introduced to enable the basis for automatic service composition. The proposed approach may be implemented as a peer-to-peer based NGN using SIP signalling.

Keywords-component; Service Composition; NGN (Next Generation Networks); Peer-to-Peer; Session Initiation Protocol (SIP); Service Description Language

I. Introduction

Service provisioning is playing a key role in Next Generation Networks (NGN). The traditional service provisioning approach, relying on the mobile operator as the sole provider, does include a number of advantages, primarily service availability and control. However, the main disadvantage of the approach is the relatively limited level of choice for the consumer. In future, an important discriminating factor when choosing providers is likely to be the range of value-added services (VAS), dictated by the increasing customer demand for services tailored specifically for their needs. To meet these requirements, one of the critical steps is to implement support for openness towards new services. This will lead to a number of new challenges, such as opening the network for external services, beyond the ones from the mobile network provider, or possibilities to build up new services by giving customers facilities to extend them. The Open Mobile Alliance (OMA) specified the Next Generation Service Interface (NGSI) framework [1], in support of developers creating new services that integrate the telecommunication networks functionality. In addition, the Parlay [2] and Global Systems for Mobile Communications Association (GSMA) [3] consortia defined APIs for Telecom third party services. All these APIs are based on Web Services and they offer limited capabilities of the underlying network from a telecommunication provider via a gateway. These APIs were solely specified for centralised telecommunication infrastructures.

In NGN-based telecommunication, SIP (Session Initiation protocol), as an IP-based signalling protocol, plays a major role. SIP is based on a simple request-response interaction model that allows developers to interact with individual protocol messages and SIP can start/manage/tear-down sessions for any media type, be it voice, video or application sharing [4]. SIP is not clearly defined as a client-server architecture, but a hybrid peer-to-peer system implementation. In centralised architectures based on SIP, for example in NGN with IP Multimedia Subsystem (IMS) as core infrastructure, services are typically provided by SIP Application Servers (AS).

Given the likely future demand for diversified services and the limitations of the mobile operator to provide them, a viable alternative is to design a peer-to-peer infrastructure. A broad range of services can easily be offered by a peer-to-peer infrastructure, acting as a service environment, because of its benefits in a number of areas, from scalability to diversity. In such a peer-to-peer environment, the services may be provided by any peer, which would be acting as a distributed server. While appealing in terms of its potential offer, such an approach does raise a number of additional problems in relation to the discovery, establishment, combination and management of services. Unlike the single provider scenario, the discovery of new services in peer-to-peer environments is not automatic or controlled, but distributed and uncoordinated. In relation to this, service composition requires that basic services may be reused.

The distributed services facilitated by the peers will be named as atomic services and the composition of atomic services will be named molecular service. A service composition can be explained as follows: given a pool of atomic services A_i, $i=1,N$, a molecular service M_x can be created by combining a subset of the given atomic services A_j. Further, another molecular services M_y can be established using a different subset A_k of atomic services.

The concept of atomic services is not novel, given that the current NGN do support such functionality, the use of URI (Uniform Resource Identifier), with integrated searching and access for the users. In contrast, supporting search for atomic as well as for molecular services in the peer-to-peer (P2P) infrastructure requires a mechanism to aggregate services automatically, so subscribers have the possibility to search for new service, potentially having the option to describe molecular services [5] [6].

The aim of this paper is to address the issues related to the description and identification of atomic services in order to allow providers and customers to discover newly published atomic services provided by peers and allow providers to combine services to new sets of services, ultimately offering a wider choice of VAS. To address these issues, the peer-to-peer infrastructure has to support publishing and discovery of atomic services, as well as the ability to describe services. The proposed solution is a new Service Description Language that describes uniquely and exhaustively each atomic service, by defining its inputs, outputs, and functionality. A secondary aim of the paper is to optimise the provisioning and composition of distributed VAS for future SIP-based telecommunication networks that are based on a P2P infrastructure, based on the principles of Services Oriented Architecture [7].

The reminder of this paper is structured as follows: the next section presents the proposed architecture and its functionality. Section III describes the Service Composition with an exemplary scenario then section IV discusses the design of a Service Description Language for Service Composition. Section V shows an example of Service Composition based on the developed Service Description Language and section VI concludes the paper.

II. Architecture of SIP-based Peer-to-Peer network

Given its wide deployment, flexibility and ease of use, SIP represents the de facto standard for signalling and communication in modern telecommunication environments. A P2P infrastructure, allowing client-to-client communication, can be established using only the typical SIP network elements (User Agent, Proxy Server, Registrar Server, and Location Server); a proof-of-concept SIP-based infrastructure was already introduced in [8]. The communication mode in such an environment is based on the traditional SIP interaction, with extensions of messaging and entities to allow full support for participating peers. As shown in Figure 1, the SIP Registrar and SIP Proxy servers are needed to lookup the location of the called party (SIP User Agent B). Therefore the temporary SIP URI is resolved to route the initial SIP request and its response (`INVITE` and `200 OK`). It is obvious that the three-way-handshake (by sending `ACK`) is completed directly between the single peers, respectively clients. From this point all data (payload and signalling) will be exchanged in peer-to-peer manner. Such a hybrid P2P infrastructure can be used to provide quick and easy services, especially value-added services.

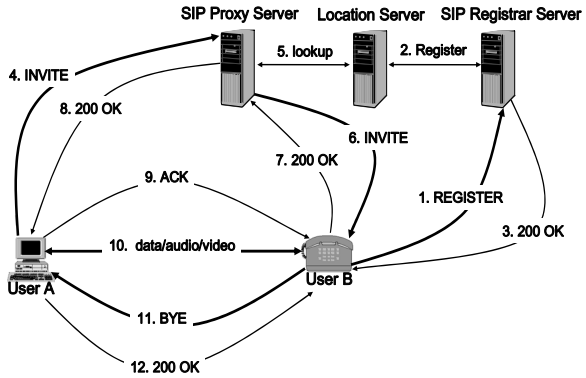

Figure 1. SIP interaction for a hybrid p2p infrastructure

Although SIP may provide the basic required functionality for session management, a complete P2P solution would require further support and entities in order to offer a service similar to the client-server architecture. First a P2P overlay composed of SIP Registrar/Proxy servers has to be created (see Figure 2). The domain for these elements can be resolved by DNS (Domain Name System) [9] [10], or dynamic DNS [11]. Now further SIP network elements can simply register with the overlay network.

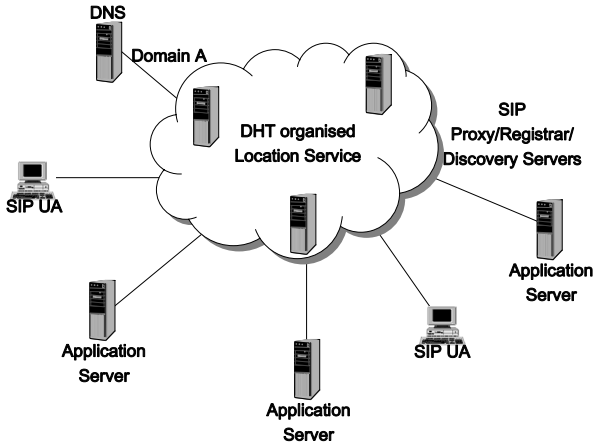

Figure 2. SIP-based P2P network

The location server functionality can be implemented using a Distributed Hash Table (DHT) implementation based on the SIPDHT2 project [12]. The DHT organised location server is holding the data of the registrations from the peers to provide the SIP routing. With the DHT in place,

application servers and media servers can easily be connected to the overlay as peers, ready to provide value-added services [13] [14]. To manage the addressability of application and media servers by their SIP URI, these servers also have to register with the overlay network. Furthermore the authentication of these servers is verified, so that only authorised servers can provide new services. The advantages of this approach are the usage of existing specifications such as SIP or DNS, as well as standard SIP User Agents. Furthermore, the fast lookups, which resolve the temporary SIP URI from the permanent SIP URI, are well provided by the hybrid approach. Further, the architecture allows to register and deregister new services, as demonstrated in the following sections, and also provides service discovery (supported through SIP messages e.g. `NOTIFY`, `SUBSCRIBE`). Due to the P2P concept, the reduction of expenses and scalability will be adopted.

Based on the presented environment, the functionality of the SIP application servers must be extended in order to provide these services to end users. Before providing services, an application server has to register with the network using the `REGISTER` request [8], then each new service has to register [15] [13] [14]. Assuming that an application server has already registered the following URI "as1@domain.co.uk", it is authorised to send the following SIP request (see Figure 3) to the SIP Registrar resolved by DNS to register service X.

```
REGISTER sip: domain.co.uk SIP/2.0
To: <sip:serviceX.as1@domain.co.uk>;tag=1234
Contact: <sip:serviceX.as1@88.88.88.88>
Expires: 3600
```

Figure 3. SIP service registration request

This registration process uses an URI that originates from the subdomains as they are used in DNS, where the service name is used as a subdomain of the application server "as1".

After registration, the new service is reachable by its URI and subscribers can make use of it. By using this model, registered application servers can provide any number of services. To support this functionality, the `Expires` SIP header field sets the period of time for the validity of the registration. As a result, the application server has to register the service with the underlying network in predefined intervals of time in order to make it accessible (here in one hour time slots). With the same SIP request, the service can also be deregistered from the network by sending the SIP header field `Expires` with the value 0. This mechanism fully supports openness for new services.

In addition to the described mechanisms, the overlay network has to be extended in order to construct an architecture that supports publishing and discovery of p2p services. Recently, the Internet Engineering Task Force (IETF) has standardised SIP and several extensions, particularly the SIP event framework [16], as the architecture for status delivery from and to any host in IP-networks and the SIP extension for event state publishing [17]. As part of the architecture, new services will be published using the SIP `PUBLISH` method. The body of this SIP method contains the service description (such as the service name, human readable service description, service id, and service URI) specified using XML [18]. The `PUBLISH` method is sent to the overlay network, where the SIP Proxy and Registrar servers are extended by a so-called SIP event server regarding to [19] to support the publishing and discovery of services. All information belonging to the events is hosted by the SIP event server, named SIP Discovery server for the remainder of this paper.

The message body of the SIP `PUBLISH` method contains the service description. Two further messages, namely `SUBSCRIBE` and `NOTIFY`, are used to realise the discovery functionality. The SIP messages `SUBSCRIBE` and `NOTIFY` will be used for service query and answering. Figure 4 depicts the service discovery process and the associated messages.

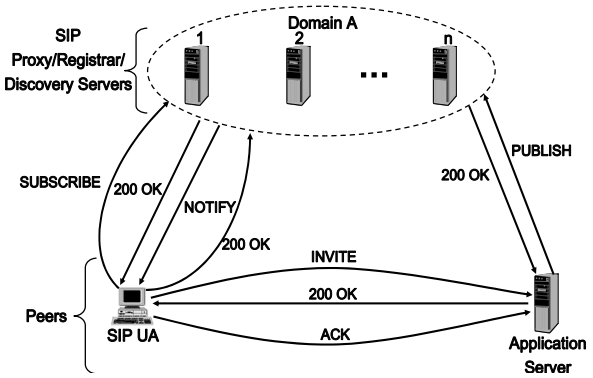

Figure 4. Find, bind and execute paradigm in SIP-based P2P networks

The SIP UA subscriber initiates the process by sending a `SUBSCRIBE` service query message to subscribe to an event and receives `NOTIFY` for the initial notification (which contains the service information). Depending on the lifetime specified within the subscription, subsequent `NOTIFY` messages might be created. In case of a lifetime of zero, the SIP Discovery server merely sends back all available and matching services at this point of time. If the lifetime is greater than zero, the exchange establishes an availability subscription to future services [18] [19].

The resulting architecture, presented in Figure 5, matches the *find, bind and execute* paradigm of the SOA. A so called service triangle could be identified and integrated into a general NGN strata view, as shown in Figure 5 [7]. The SIP Discovery servers are representing the service directory, the service provider is the AS, and the service customer is the SIP User Agent.

The depicted architecture includes all the necessary components and associated interaction to support the proposed service composition, therefore is the basis for Service Composition, presented in the next section. The presented architecture had already implemented for verification.

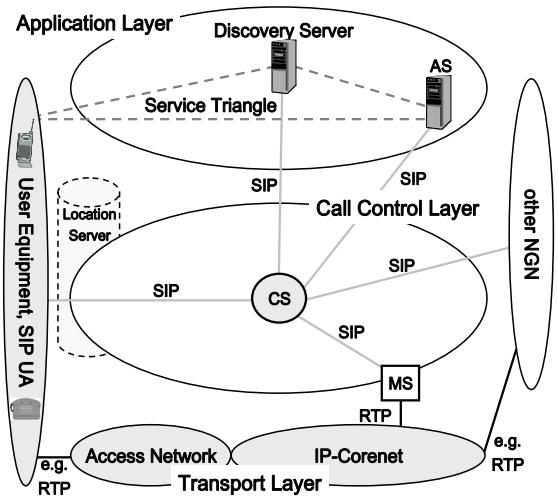

Figure 5. Service triangle in general NGN strata view

III. Service Chaining

The framework described in section II simplifies significantly the combination of distributed services. The following will present a simple method to realise invocation of combined services.

The SIP `Route` Header [8] will be used for invoking the services regarding to [20]. The SIP `Route` header holds a set of SIP URIs, each of them representing the addresses of the atomic services to be combined. Based on the formalised description a set of atomic services A_k,$k=1,N$ are invoked in the sequence they will appear separated by comma in the `Route` header ($R=A_1,A_2,A_3$) followed by the atomic service declared in the SIP request line L.

The concept introduced by the MultiService Forum (MSF) [21] is based on a centralised architecture referred to IMS. The Service Broker is defined as the central entity that manages service identification and routing. This approach was adjusted for realising service compositions in SIP-based peer-to-peer networks[7]. The following will illustrate the service chaining with the extension of manipulating media streams.

A typical scenario of service composition can be an Audio/Video conference with an embedded news ticker which presents text-based live information. In the proposed distributed service architecture, every atomic service is addressable by a permanent SIP URI, such as audio.conf@p2pnet.de, while composed services can be achieved by chaining the SIP URIs. The resulting service could be described by the following SIP message (note only the relevant header fields are shown) presented in Figure 6.

```
INVITE sip: videoconference.bt@99.99.99.99 SIP/2.0
…
Route: <sip: audio.conf@77.77.77.77;lr>,
       <sip: newsticker.bbc@88.88.88.88;lr>
…
```

Figure 6. SIP service chain

The SIP message would be routed using the `Route` headers, as shown in Figure 7.

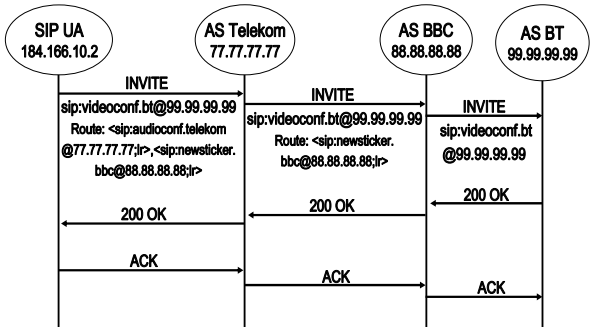

Figure 7. Exemplary service composition with SIP Route Header

As depicted in Figure 7, the SIP message `INVITE` is forwarded from one service to another, the services themselves may convert parts of the SIP message, and the news ticker service might modify the SIP body of the initial message. For the purpose of embedding the news ticker into the video stream, the SDP media description part, containing video information (see Figure 8), has to be modified by the news ticker service, so that all video data will be forwarded to the news ticker. As a result, the news ticker service can overlay the ticker information into the video data. This can be achieved by modifying the corresponding information from the SIP body as follows.

```
…
v=0
o=client 1234 0 IN IP4 184.166.10.2
s=ServiceCombination
t= 0 0
c=IN IP4 184.166.10.2
m=audio 1000 RTP/AVP 0 8 3
m=video 2000 RTP/AVP 34
…
```

Figure 8. SDP for an atomic video service

The submitted IP address given by the Connection Data parameter `c` and the submitted port given by the Media Description parameter `m` (which in Figure 8 has the value 2000, indicating that the video will run on port 2000) specify the socket the subscriber will expect the video data. The news ticker service adds a new `c` parameter to the SIP body in front of the `m` parameter (video) with its own IP address so that he can receive the video data. Figure 9 will present the modified SIP body.

```
…
v=0
o=client 1234 0 IN IP4 184.166.10.2
s=ServiceCombination
t= 0 0
c=IN IP4 184.166.10.2
m=audio 1000 RTP/AVP 0 8 3
c=IN IP4 88.88.88.88
m=video 2000 RTP/AVP 34
```

Figure 9. Modified SDP for a molecular service

By modifying the SIP message structure as described, the video payload can be routed as intended and the combining of services can be easily implemented. This approach takes advantage of the P2P architecture benefits, such as scalability, cost reduction, fault tolerance, and it provides full support for standardised SIP network elements, such as SIP User Agents and SIP Proxy/Registrar servers.

The basis for service composition is a new service description language that is explained in the next chapter. This service description language supports the requirement for appropriate designing molecular services and defines the further requirements, as for example temporal relations between atomic or molecular services.

IV. Service Description Language

A well defined service description language is the critical element of automated service composition for a number of reasons. Firstly, the SIP Discovery Server, which creates the service composition, needs to have knowledge about the services and their interfaces, so that it can combine them. Secondly, the service description needs to be precise enough for the service to be uniquely identifiable. Finally, the description language has to be machine readable and interpretable in order to streamline the service composition underlying process.

The proposed service description language consists of five constructs: *service Id*, *temporal composition*, *service goal*, *functional properties*, and *non-functional properties*. The *service Id* is an identifier for an atomic service represented by its permanent SIP URI. The *temporal composition* describes how different atomic or other molecular services are combined to produce the defined service; it is based on all temporal relations between the single services, as formalised in [22]. According to Allen [23], relations between two intervals can be divided into 13 different relations using the `before`, `during`, `meets`, `finishes`, `overlaps`, `starts`, and `equals` connections. Figure 10 shows seven of them. The other six relations are derived through inversion of these, apart from the `equals` one. The linkage between the service and the temporal relations is defined recursively (see Figure 11). An example presented in the next section will describe the *temporal composition* more deeply. The *service goal* is a set of keywords describing the functionality so that humans can imagine what the service will offer. For example a service goal with the key words "wake up" and "instant message" might describe a service that will send an instant message as a wake up message. *Non-functional properties* are, for example, cost or security. They are used to limit the space of compositions that fulfil the service request and to rank the generated set of compositions, such as ranking higher a service with lower costs and higher reliability.

The *functional properties* are the *conditions* (pre-, continuous and post-condition), *connectors* (in-, through- and output), *media objects* (source, sink and filter), *communication* (a/synchronous) and *interaction* (non-/real-time). The *conditions* are properties that must be considered before, during or after service processing; an example of a condition is that, prior to any interaction, the SIP session must be established.

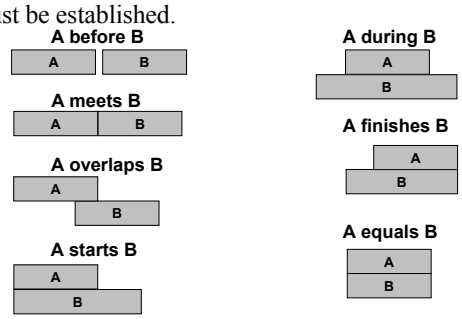

Figure 10. Possible temporal relations

The *connectors* are major elements for aggregating services. All in-, out- and throughputs offering attributes to identify if the *connectors* are for the media stream of the originating party (the party who initiated the service invocation) or if they belong to the terminating party. *Connectors* also may be optional, this means they can be utilised if they are needed, for example in conferences up to 10 participants. Furthermore, the *connectors* are linked to *media types*. These *media types* are derived from human senses (hearing, sight, smell, touch, taste) also another *media type* is defined which is not sensually imperceptible. Four further senses (pain, balance, proprioception and temperature) are known, which will be assigned to the sense touch, only for simplification. These *media types* will cover all possibilities for transferable media in telecommunication. Table I gives an overview of the media types and their characteristics regarding to [22].

TABLE I. MEDIA TYPES AND THEIR CHARACTERISTICS

Media Type	Characteristics
Hearing	sound, speech, music
Sight	video, animation, text, still image, light
Smell	camphorous, fishy, malty, minty, musky, spermous, sweaty, urinous
Touch	pressure, temperature, balance, proprioceptive
Taste	sweet, sour, bitter, spicy, umami
Imperceptible	file, sensor, actuator, trigger, information

The *connectors* are forming ports for linking them with the matching opposite ones. For example Out- and Throughputs can only be linked to In- or Throughputs which will fit, that is they have to be from the same *media type* and must own the same characteristic (such as Hearing and Sound). Ports of different *media types* and different characteristics are incompatible and therefore cannot be combined. The characteristics shown in Table I are

specifications of the *media types*, so they can be assigned more precisely. This is essential in communication networks, because different media type characteristics may have different bandwidths/sampling rates; for example, standard sampling rate for speech is 8 kHz in contrast to music in CD-quality, which is 44.1 kHz. In telephony special Codecs exists for these different demands. The grading of quality aspects within one characteristic e.g. speech is a *non-functional property*. In contrast, the imperceptible media type is characterised by different technical aspects or formats. This is required for aggregating the same characteristics of imperceptible *media types*.

The *media objects* constructs include media *sources*, *sinks* or *filters*. The *sources* output data streams and the *sinks* receive data streams of concrete media types. *Filters* combine the characteristics of *sources* and *sinks* and they are mostly used to manipulate data streams. For this purpose, the *filters* are divided into different functionalities namely *input/output-selector*, *synchroniser* (synchronise different *media types* such as speech and video), *inter-media* and *intra-media*. The *input-selector* is a device that selects one of several input *media types* and forwards the selected *media type* as output. On the other hand an *output-selector* takes a single input *media type* and forwards it to a chosen output from several. Both selector objects are controlled by an input *media type* (e.g. an imperceptible media input like sensor controls hearing *media types*). *Inter-media filters* are processing different *media types* however *intra-media filters* are only processing one *media type*. These two media *filters* can be divided into *converter* (for example Text-to-Speech), *splitter* (for example separate subtitle from video) and *merger* (for example bundle speech and video to a file as output) for *inter-media filters* and *intra-media filters* can be divided into *mixer* (for example video conferences or text chat), *multiplier* (duplicate the media type) and *modifier* (for example change the size of a video).

Also general information has to be defined for all *media objects,* this is the form of *communication* (for example one-to-one, many-to-one) and last but not least the way of *interaction* has to be defined. For example a real-time service will be a normal voice call in contrast to instant messaging which is a non-real-time service.

Figure 12 gives an overview of the basic structure for the service description language, which is based on XML. The *temporal composition* defines how services (atomic and molecular services) are arranged. This is done recursively (see Figure 11).

```
<Service>
 <ServiceID id="serviceComposition"/>
 ...
 <TemporalComposition>
  <ServiceID id="atomicService">
   <TimeRelation relation="before">
    <ServiceID id="molecularService">
     <TimeRelation relation="equals">
       ...
     </TimeRelation>
    </ServiceID>
   </TimeRelation>
  </ServiceID>
 </TemporalComposition>
</Service>
```

Figure 11. Temporal Composition

Figure 11 shows that also molecular services might be contained in the *temporal composition*. Every molecular service holds its own *temporal composition*. This means that *temporal compositions* can be nested.

Based on this service description language compositions can be arranged so that machines can interpret them.

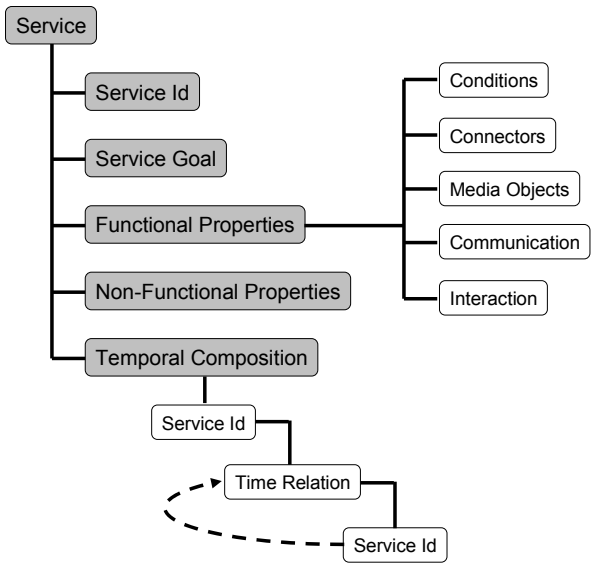

Figure 12. Structure of the Service Description Language

The following section will present an example to clarify the usage and construction of the service description language.

V. Example

The service scenario presented in section III is a good example to demonstrate how service composition is described using the service description language.

First it is necessary to take a look at the service descriptions of the three atomic services (audioconference, videoconference and news ticker). Only the relevant properties of these services will be discussed in the following subsections. The *service Ids* are audio.conf@p2pnet.de for the audio conference service, videoconference.bt@p2pnet.de for video conference service and newsticker.bbc@p2pnet.de for the last service. The properties of the audio conference service are shown in Figure 13.

```
<ServiceID id="audio.conf@p2pnet.de"/>
<FunctionalProperties>
 <Connectors>
  <Input id="1" optional="false" case="originating">
   <hearing audtion="speech"/>
```

```
 </Input>
      ...          further inputs
 <Output id="1" optional="false" case="originating">
  <hearing audition="speech"/>
 </Output>
      ...          further outputs
</Connectors>
<MediaObjects object="filter">
 <Intra-Media type="mixer">
  <InputIDs> 1, 2, ..., m </InputIDs>
  <OutputIDs> 1, 2, ..., n </OutputIDs>
 </Intra-Media>
 <CommunicationForm>
  <Many-Many inParties="m" outParties="n"/>
 <CommunicationForm>
</MediaObjects>
<Communication synchronisity="synchronous"/>
<Interaction reaction="real-time"/>
<FunctionalProperties>
```

Figure 13. Audio Conference Service Description

The only differences between the description of the video and the audio conference services are in the *connectors*. The *media type* of the audio conference service is hearing and the *media type* for the video conference service is sight with characteristic video.

The properties of the news ticker service are presented in Figure 14.

```
<ServiceID id="newsticker.bbc@p2pnet.de"/>
<FunctionalProperties>
 <Connectors>
  <Input id="1" optional="false" case="terminating">
   <sight vision="video"/>
  </Input>
  <Output id="1" optional="false" case="originating">
   < sight vision="video"/>
  </Output>
 </Connectors>
 <MediaObjects object="filter">
  <Intra-Media type="modifier">
   <InputIDs> 1 </InputIDs>
   <OutputIDs> 1 </OutputIDs>
  </Intra-Media>
  <CommunicationForm>
   <One-One/>
  <CommunicationForm>
 </MediaObjects>
<Communication synchronisity="synchronous"/>
 <Interaction reaction="real-time"/>
<FunctionalProperties>
```

Figure 14. News Ticker Service Description

In this scenario, service composition is defined by the *temporal composition* properties. The audio conference *A* service and the news ticker service *B* have to run in parallel. Translated in the *temporal composition* syntax defined earlier on, *A = B* (*A equals B*). During these two services the video conference service is running (see Figure 15), because the outgoing media streams are connected to the news ticker service so that the news tickers data can be embedded into the media streams of the video conference (see Figure 16).

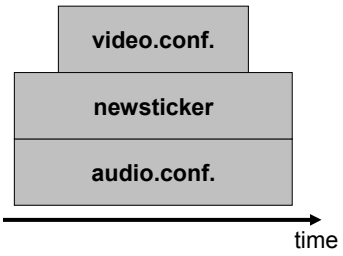

Figure 15. Temporal relation between services

As shown in Figure 16 the incoming video data will be handled first by the video conference service and the resulting data will be delivered to the news ticker. Therefore the news ticker must have the ability to receive data before the video conference service can transmit the data to it. So the news ticker has to be started before the video conference service, and the audio conference service should run in parallel.

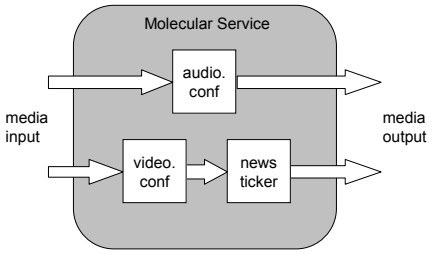

Figure 16. Media processing

The temporal composition within the service description language is shown in Figure 17. The resulting service invocation presented in Figure 7 can easily derived from the temporal composition.

```
<TemporalComposition>
 <ServiceID id="videoconference.bt@p2pnet.de">
  <TimeRelation relation="during">
   <ServiceID id="audio.conf@p2pnet.de">
    <TimeRelation relation="equals">
     <ServiceID id="newsticker.bbc@p2pnet.de"/>
    </TimeRelation>
   </Service>
  </TimeRelation>
 </Service>
</TemporalComposition>
```

Figure 17. Temporal Composition

The synchronisation between the audio and video streams is not handled by the service, because there is no synchronizing media object included. The synchronicity of media streams could be managed by the terminal equipments.

This short example details on how to use the proposed service description language to form molecular service compositions out of existing atomic services. The following

chapter is concluding this article and will discuss further steps to implement this approach into a prototypical environment for validation.

VI. Conclusion

Automated service composition for value-added services is the most important means of innovation in peer-to-peer based telecommunication networks. In particular, automated service composition enhances the economy and allows service creation for small customer groups.

The presented approach empowers customers to use specialised services in a cost efficient way. The presented architecture was already tested under laboratory conditions. First tests also have shown that automatic service composition is possible.

Next steps to go are development and implementation of an adequate algorithm to automatically build service compositions based on the presented service description language. Furthermore the service composition process has to be validated. This will be done by specifying a set of atomic services, which have to be combinable. Also the reusability and modularity of atomic services has to be analysed. This means further research on atomic services and their specialisations has to be done.

References

[1] Open Mobile Alliance: Application Programming Interfaces (APIs), Available at: www.openmobilealliance.org/comms/pages/oma_apis.htm [Accessed 16 June 2011].

[2] 3GPP: 3GPP Specification series TS 29.199-01 up to TS 29.199-22, Available at: www.3gpp.org/ftp/Specs/html-info/29-series.htm [Accessed 16 June 2011].

[3] GSMA: GSMA OneAPI, Available at: www.gsmworld.com/oneapi/ [Accessed 16 June 2011].

[4] Banerjee, N.; Chakraborty, D.; Mittal, S.: "Service Delivery Platforms: Developing and Deploying Converged Multimedia Services", Syed, A.A; Ilyas, M., Auerbach Publications p.1 – 28 ISBN 978-1-4398-0089-8

[5] Silva, E; Pires, L; van Sinderen, M.: "An Algorithm for Automatic Service Composition", 1st International Workshop on Architectures, Concepts and Technologies for Service Oriented Computing, ICSOFT 2007, Barcelona, Spain, July 2007.

[6] Zhao, Z.; Laga, N.; Crespi, N.: "The Incoming Trends of End-User driven Service Creation", First International ICST Conference, DigiBiz 2009, London, UK, June 2009.

[7] Lehmann, A; Trick, U.; Fuhrmann, W.: "SOA-basierte Peer-to-Peer-Mehrwertdienstebereitstellung", 14th VDE/ITG Mobilfunktagung, Osnabrück, May 2009.

[8] Rosenberg, J.; Schulzrinne, H.; Cammarillo, G.; Johnston, A.; Peterson, J.; Sparks, R.; Handley, M.; Schooler, E.: "SIP: Session Initiation Protocol", RFC 3261, IETF, June 2002.

[9] Mockapetris, P.: "Domain Names – Concepts and Facilities", RFC 1034, IETF, November 1987.

[10] Mockapetris, P.: "Domain Names – Implementation and Specification", RFC 1035, IETF, November 1987.

[11] Vixie, P.; Thomson, S.; Rekhter, Y.; Bound, J.: "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, IETF, April 1997.

[12] Fandrianto, A.: "SIPDHT: An Open Source Project for P2PSIP", Implementation report, Cisco Systems, April 2007.

[13] Lehmann, A.; Eichelmann, T.; Trick, U.: "Neue Möglichkeiten der Dienstebereitstellung durch Peer-to-Peer-Kommunikation", 13th VDE/ITG Mobilfunktagung, Osnabrück, May 2008.

[14] Lehmann, A.; Fuhrmann, W.; Trick, U.; Ghita, B.: "New possibilities for the provision of value-added services in SIP-based peer-to-peer networks", University of Wrexham, Proc. of SEIN, November 2008.

[15] Schmidt, H.; Guenkova-Luy, T.; Hauck, F.J.: "Service Location using the Session Initiation Protocol (SIP)", International conference on Networking and Services (ICNS), Silicon Valley, CA, July 2006.

[16] Roach, A.: "Session Initiation Protocol (SIP) Specific Event Notification", RFC 3265, IETF, June 2002.

[17] Niemi, A.: "Session Initiation Protocol (SIP) Extension for Event State Publication", RFC 3903, IETF, October 2004.

[18] Rahman, M.; Buford, J.: "Service Discovery using Session Initiation Protocol (SIP)", United States Patent Application Publication, June 2006.

[19] Trossen, D.; Pavel, D.: "Service discovery & availability subscriptions using the SIP event framework", IEEE International Conference on Communications, Seoul Korea, May 2005.

[20] MSF-IA-SIP.006-FINAL: "Implementation Agreement for SIP interface between Service Broker and Application Server", Implementation Agreement, MultiService Forum, April 2005.

[21] MSF (MultiService Forum): http://www.msforum.org [Accessed 5 July 2011].

[22] Lehmann, A.; Fuhrmann, W.; Trick, U.; Ghita, B.: "A new approach to classify and describe telecommunication services", Proc. of SEIN 2009, University of Applied Sciences Darmstadt, Germany, November 2009.

[23] Allen, J.: "Maintaining knowledge about temporal intervals", Communications of the ACM, ACM, 1983.