Providing Web Services over DVB-H: Mobile Virtual Web Services

Ana Fernández Vilas, Rebeca P. Díaz Redondo, José J. Pazos Arias, Alberto Gil Solla, Jorge García Duque

Abstract — In this paper we introduce our experiences in accessing Web Services (WSs) from DVB-H terminals. Because of the reduced capabilities of the mobile devices, clients residing in these terminals should be as simple as possible. So, we propose to move their complexity to the service provider by defining Virtual WSs. A Virtual WS groups one or more WSs sharing a conceptual functionality, although they could be implemented in different locations and with different characteristics. Because of the WSs virtualization, the provider makes the mobile client independent of these implementation details as well as of their availability. Virtualization also gives the possibility of defining QoS-aware services as well as caching-based systems to process the responses of clients more efficiently. Finally, we propose how to broadcast the simple clients and the information needed to access the Virtual WS by using the technology provided by the DVB-H network.

Index Terms — DVB-H, Web Services, Virtualization.

I. INTRODUCTION

Web Services (WSs) [1] combine the best aspects of component-based development and the Internet. Like components, Web Services represent black-box functionality, which can vary from simple requests to complicated business processes; besides, they can be reused without worrying about how the service is implemented. Unlike current component technologies, Web Services are not accessed via object-model-specific protocols, such as DCOM, RMI, or IIOP. Instead, Web Services are accessed via ubiquitous Web protocols (ex: HTTP) and data formats (ex: XML). Nowadays, the Web Services field is being enriched with several standards to support not only a B2C infrastructure, but also a B2B one. WSDL, UDDI and SOAP are considered the core of this Web Service standards set. WSDL (Web Services Description Language [2]) defines a XML format for describing network services. These services are published and registered in an UDDI (Universal Description, Discovery and Integration [3]) registry; a platform-independent, XML-based registry for Web Services descriptions that is roughly equivalent to an automated online ‘phone directory’ of WSs. Finally, the exchange of information between clients and WSs is supported by SOAP (Simple Object Access Protocol [4]), a XML-based lightweight protocol suitable for communication in a decentralized, distributed environment. To sum up, Web Services are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. So, once a Web service is deployed, other applications (and other Web Services) can discover and invoke it.

From the above standard technologies, the new open Web should enable the interaction between any device (mobile phones, PDAs, computers, etc.) and the appropriate WSs. However, the services offered by wireless carriers (news, weather, etc.) to their mobile subscribers are usually supported by proprietary solutions, which are often incompatible with the new Web and the WS concept. The perception is that, today, Mobile Services live apart from Web Services world, so a more open framework where any mobile device could access to any WS would be desirable. This framework would be even more interesting if any mobile client is able not only to access to a concrete service provider but also, depending on run-time factors, to any other available provider capable of solving its demands: i.e. flexibility in changing the service provider should be a must in a mobile environment. Finally, this scenario would be more applicable if all this technical details were hidden to mobile clients, which would become not conscious about how both the WSs access and the mobility problems are overcome.

However, two relevant problems make it difficult to achieve this desirable open framework where flexibility is the key factor to communicate WSs and mobile clients. On the one hand, and inherently to the WS technology, clients are bound to a concrete service provider at design time. On the other hand, today’s mobile devices do not support the excessive complexity Mobile Web Services would need to allow changing the provider at run-time.

With the aim of overcoming both problems, we propose to differentiate between a conceptual service and its implementations by means of WSs virtualization. So, we propose to introduce an intermediate virtualization layer between mobile clients and WS providers. Thus, a Virtual WS (VWS) is defined within this intermediate layer as a conceptual service grouping one or more WSs. Then, a light client for this heavy virtual WS is delivered to the mobile receivers while the VWS is published as a standard WS. Consequently, having a light client solves the complexity problems linked to the capabilities of the mobile devices, as long as having a heavy virtual WS drawing together a group of WSs hides their complexity, availability and location to mobile clients.

Regarding to delivering aspects, broadcasting is a highly cost-effective way to reach large audiences. These broadcasting solutions revolve around building dedicated

1 Partly supported by the R+D project TSI 2004-03677 (Spanish Ministry of Education and Science)

The authors are with the Department of Telematics Engineering, University of Vigo, 36310 Vigo, Spain (e-mail: {avilas, rebeca, jose, agil, jgd}@det.uvigo.es).
networks using specialized technologies. Among them, DVB-H (Digital Video Broadcasting-Handheld [5]) is considered a good candidate for being widely adopted, especially in Europe. In fact, DVB-H is forecast to be accessible by approximately 350 million mobile users by 2008. Thus, we propose to deliver the VWS clients using the DVB-H infrastructure, more specifically, including this new kind of applications into the DVB-H stream.

The paper is organized as follows. The next section describes a desired scenario, which will be used as example all over the paper with the aim of offering a clearer exposition. Sect. III is devoted to detail our proposal, justifying the use of two well-known technologies: Web Services (for communication between clients and providers) and DVB-H (for delivering purposes). Sect. IV overviews the basis of the virtualization of WSs, describing the architecture which support this new feature as well as a new language to specify virtualization. In Sect. V we detail the special technical characteristics which are involved in the DVB-H specification and how to publish the way to access to virtual WSs from mobile devices. The explanation given in the previous sections is put into practice in Sect. VI, where the scenario is completely solved. Finally, a brief discussion including related work, conclusions and future work is presented in Sect. VII.

II. SCENARIO: DESCRIPTION

With the aim of clarifying our proposal, we will use the following scenario:

“In the country where John lives, there are several traffic information centers: some of them are free of charge, while others are supported by their subscribers. Although the information provided by the former kind is really useful, the information provided by the latter is usually more detailed. Since John works as a taxi-driver he is keen on receiving traffic information without delays and as accurate as possible. With this aim, he has installed a new traffic service that has been broadcast to his DVB-H car navigator. This new service is able to connect to the most adequate traffic information center according to the car location and the centers availability to give John the traffic information he needs. It also filters no relevant information and shows the important one in an easily understandable way: ‘slow traffic flow and severe incidence’ or ‘free flow’, for instance.”

This scenario shows how both virtualization and broadcast techniques are especially useful in this environment. On the one hand, there is a VWS which agglutinates the WSs belonging to the different traffic information centers, hiding implementation details to its clients. On the other hand, the traffic information application that John has installed in his car navigator is a light client able to dialog with the remote VWS. This client is delivered over the DVB-H network to all the mobile devices, including John's car navigator, from which it access the VWS, the conceptual service.

As previously mentioned, the VWS shows the traffic information in an easily understandable way. In our scenario the VWS only gives two types of information: (i) the traffic flow in the set \{Slow, Sluggish, Free Flow\} and (ii) the existence and severity of any incident, from the set \{None, Minor, Moderate, Severe\}. However, as we really know, there are different WSs providers (official sites, different driver associations, etc.), whose existence is conveniently hidden to the car navigator (WS client). What is more, these WSs could return the above information according to different representations: returning the traffic level from 1 to 5; or the average time in driving 50 meters, for instance. Anyway, all this complexity is hidden to the WS client, from whose perspective there is only one access point: the VWS at the DVB-H provider side. Consequently, the availability problem, especially important in mobile environments, is also solved.

III. PROPOSAL

The scenario described in the previous section could be solved with a proprietary solution: an ad-hoc traffic information service and its correspondent client. This client could be delivered to the subscriber’s mobile device using any available network and, once installed, it would be able to connect to the traffic service to ask and obtain the most adequate information. However, we are committed to standard solutions that support open environments and horizontal markets. Therefore, on the one hand, we apply WS technologies, which are widely used in the Internet field, to define the communication between the service and clients. On the other hand, we used the infrastructure provided by the DVB-H network for delivering purposes as well as the 2G/3G network for individual communication between a concrete client and the required WS (see Fig. 1).

Regarding the WSs field there are two factors which are really inconvenient to mobile clients. Firstly, clients are bound to a concrete service provider at design time; however, flexibility in changing the service provider should be a must in a mobile environment. Secondly, clients’ complexity is...
excessive for this kind of light receivers, which are characterized for reduced computational capabilities. With the aim of overcoming both problems, we propose to apply virtualization of WSs as an intermediate layer between traditional WSs and mobile clients (Fig. 1). Virtual or conceptual services, which are published as standard WSs, will reside within this layer grouping one or more implementations of WSs. So, the complexity traditionally linked to clients is now supported by the VWSs. Additionally, these light clients always communicate with the same VWS, without being conscious about the different WSs the VWS hides.

For delivering purposes, we propose to use the DVB-H technology, since broadcasting is a highly cost-effective way to reach large audiences. Although there are several technologies sharing the same philosophy (broadcast for mobile devices), they are territorially distributed: DMB (Digital Multimedia Broadcasting) is expected to be adopted by South Korea; MediaFLO (Media Forward Link Only) is a proprietary technology which is being developed by the American private company Qualcomm; and ISDB-T (Integrated Services Digital Broadcasting-Terrestrial) is used in Japan. For European countries, DVB-H is the reference standard: on the one hand, it is an approved standard by ETSI (European Telecommunications Institute); on the other hand, it benefits from existing DVB-T (DVB Terrestrial) infrastructure components, reducing initial investments; finally, DVB-H is clearly supported by the majority of mobile device manufacturers like Motorola, Nokia, Samsung, LG, BenQ and Sagem, which already have DVB-H devices.

Technically speaking, DVB-H has improved the DVB-T standard to overcome the communication problems derived from the inherent characteristics of small and portable devices. Firstly, timeslicing has been introduced to reduce the average power consumption of the receiver (a light battery powered terminal) and to enable smooth and seamless frequency handover, which is essential as the target are nomadic users. Secondly, it is incorporated a module of forward error correction for multiprotocol encapsulated data (MPE-FEC, Multiprotocol Encapsulation Forward Error Correction), which enhances the reception in the special conditions of handheld terminals (indoor and outdoor reception, moving vehicles, man-made noise environments, etc.). However these changes are not enough on their own. In fact, IPDC (Internet Protocol Datacasting) is what converts DVB-H in an end-to-end broadcast system for delivering any type of digital content and services using IP-based mechanisms ---in the IPDC specification the download and streaming services are based on FLUTE (File deLivery over Unidirectional Transport) and RTP (Real Time Protocol) respectively. Finally, the DVB-H ESG (DVB-H Electronic Service Guide) describes the available IP Datacast services. It is broadcast by means of FLUTE sessions and enables accessing to the available contents.

To conclude, DVB-H provides the appropriate technology for broadcasting purposes, but the communication between VWSs and clients must be supported by a bi-directional cellular path, like 2G/3G.

IV. VIRTUAL WEB SERVICES

Virtualization was introduced by the authors in [6]-[7]-[8] as an elegant solution to solve several open problems in WS technologies, like asynchronous invocation, error control, high availability, quality of service, etc. These open problems stem mainly from two limitations on WS architecture: the direct invocation model and the non-distinction between the service provider and the web service provider. These limitations impede Web Services technologies to be the base technology for mobile web services.

Especially in mobile environments, the binding process in WS technologies, binding at design-time, is an important limitation. When a client is bound to a provider, the execution of the client remains bound to the web service of the service provider for the whole client application’s life. If the client decides to change its provider, it will find the necessity of binding a new web service to the client application. This would imply re-developing code, performing binding again and adapting the client application’s logic. It is easy to understand that this way of doing is not applicable to mobile scenarios like the one proposed in this paper where several traffic information centers are available depending on the mobile location; and the kind of information provided by the center also vary depending on the subscriber payment.

WS virtualization is based on grouping one or more web services inside a unique wrapper, which is then published as a standard web service, called Virtual Web Service (VWS). Clients use the new virtual web service as a standard one, i.e. there is no difference between real and virtual from the client’s point of view. With virtualization, some additional logic can be performed out of the client applications (error management, provider selection, etc.). In this way, the software complexity, especially the client software is radically...
reduced, which is essential for the resource-restricted nature of mobile devices.

Proposing a virtualization technique for web services requires a change in the standard web services architecture. The most important change comes from the fact that the virtual web service must reside in an intermediate element different from the client and the provider. Taking this factor into account, the architecture proposed for the use of VWS determines the existence of five elements (Fig. 2):

- **Client**: the entity that needs a service, it is equivalent to a client in a SOA structure.
- **Delegate client**: the agent on who the client delegates the responsibility for executing a service; generally speaking, it represents client applications.
- **Service provider**: the entity that offers a service: it must be seen as a provider in a SOA architecture.
- **Delegate provider**: the agent on who the provider delegates the responsibility for offering a service; in web services technology, a delegate provider is a web service.
- **Finally, engine or intermediary**: the entity in charge of putting delegate clients and delegate providers in contact; it performs communication processes between both delegates following different algorithms, and pursuing different objectives depending on the runtime environment.

**A. Implementation of VWSs**

Virtualization will guide us to the use of a new kind of services: virtual web services (VWS). A VWS is published in a directory by means of a WSDL document, in the standard way. However the implementation of this VWS is virtually defined by means of another document, the VWSDL document (VWS Definition Language). VWSDL is an XML-based language which has been defined as a stand-alone language, i.e. not as a WSDL extension, because it is intended for a distinct use. The clients do not use VWSDL documents, since these documents are just a definition of an implementation of a virtual service, the VWSDL definition is only useful to a VWS engine.

VWSDL documents, which describe virtual web services, must contain, at least, a list of the methods provided by the service (the virtual equivalence of a WSDL portType), as well as the input and output parameters (and their data types). The definition of the virtual methods also contains the web services (delegate providers) and implementation methods that can be invoked in order to accomplish the execution of a virtual method. In fact, all those web services and their methods represent the implementation of a virtual method, and they can be specified (inside the virtual method element) using as many invoke elements as needed (see the VWSDL fragment of the scenario in Sect. VI).

Each method element has its own name and type attributes. The type attribute is used to specify the type of implementation that is being defined for the method. For instance, the sequence value states that, in order to execute a virtual method, the engine must invoke a set of delegate providers (specified inside invoke elements) following a predefined sequence. Other values for the type attribute can be used to specify different engine behaviours (equivalent, alternate, parallel, iterative...).

Finally, on top of VWSDL, we have defined VWSEL (Virtual Web Services Extension Language). This extension language allows the clients to use SOAP-Header elements to add control information which modifies the behavior of the engine, for instance, specifying the QoS parameters to select the provider or the caching characteristics allowed in the invocation.

**B. Deployment and use of VWSs**

Fig. 3 and Fig. 4 show the development and production cycles for virtual WS, respectively.

On the one hand, the development cycle in Fig. 3 (virtual development in fact) consists of mainly three steps:

1. The developer searches for the web services which are going to be used as the real implementation of the VWS. This first step implies that the developer locate the providers/delegate providers by using a directory (usually UDDI).
2. The providers are aggregated in a new virtual WS. The aggregation can be as simple as a list of providers but it can also incorporate additional specifications of availability, SLA (Service Level Agreement), etc. This second step can be considered the implementation of the VWS.
3. After that, the designer has to build the interface of the new VWS, the public methods and their signatures. Remark that the methods and parameters published by a VWS can be totally different from the ones published by
the delegate providers. This step leads to the automatic generation of the WSDL description of the VWS which will be published in a UDDI directory.

On the other hand, the production cycle, in Fig. 4, is totally different to the one in standard web services; mainly because of the change in the invocation model: from direct invocation in standard WS to indirect invocation in VWS. This architectural change is made quite clear on the bottom part of the Fig. 4.

1. The production cycle starts when the client invokes a VWS, this invocation is for the client a standard invocation, since the client does not have to know that the invoked service is a VWS or not.
2. When the invocation is received by the engine, it extracts the possible information in the SOAP message and finds the VWSDL document which describes the invoked VWS.
3. From this information, the engine decides the delegate provider which will be used to resolve the client query: a SOAP message is built and sent to the delegate provider which is in charge of executing the service.
4. On success, the response from the delegate provider is received by the engine which, once again, builds a SOAP message to be sent to the client.

V. VIRTUAL WEB SERVICES OVER DVB-H

To make WS technologies fully accessible from DVB-H infrastructure, the DVB-H provider needs also to play the role of WS aggregator, making WS available to its DVB-H subscribers. As previously mentioned, the DVB-H standard specifies two ways for broadcasting content by using, respectively, the RTP and FLUTE protocols. RTP contents are mainly devoted to TV-like broadcasting, whereas other binary objects (textual, applications, multimedia, etc.) can be delivered in a data carousel over FLUTE. Both kinds of contents can be locally accessed by using the information in the ESG. Therefore, we propose to publish VWS as DVB-H services in the ESG and to broadcast their corresponding light clients as applications in the data carousel as Fig. 5 shows.

The DVB-H ESG Data Model is based on XML Schema and it is aimed at being consistent across all implementations of a system to ensure interoperability. An ESG instance of the ESG Data Model is a consistent set of ESG data describing the available services. These instances are fragmented into ESG XML fragments which are encapsulated, processed and transmitted by means of FLUTE sessions. As the upper part of Fig. 4 shows, the ESG Data Model describes the nature of the content items (Content) which makes up a Service, the broadcast time they have been scheduled to (ScheduleEvent) and the information for accessing them (Acquisition). Consequently, any broadcast information (streaming content over RTP, off-line contents over FLUTE) is correctly placed into the ESG Data Model linked to its corresponding DVB-H service. Finally, the DVB-H Transport Stream (TS) is generated by multiplexing the different TV Services and the IPDC data.
Continuing with communication aspects, once the broadcast service is located (its light client and the location information of the VWS), it is directly accessed by using the bidirectional 2G/3G network. Thus, the interaction between client and VWS is supported by the operators' cellular network through the return channel. This communication is based on the SOAP specification, similarly to the traditional interchange of information between client and WS.

As it was previously mentioned, the key point of this proposal is the virtualization of the broadcast services. This technique, which differentiates between conceptual services and its implementations, allows pushing the complexity of clients into the VWSs, so it is possible to have light clients more appropriate for the reduced capabilities of mobile devices. Additionally, the VWSs group one or more WSs, hiding details like availability or location to clients, which is especially interesting for devices on the move. With this aim, we introduce an intermediate layer between the WS providers and the WS clients: the Virtualization Layer where the VWSs reside (Fig. 6) as they were defined in Sect. IV.

VWSs are designed by VWS providers, who are responsible for describing new VWSs applying the VWSDL language and for implementing the software needed to interact to the traditional WSs by using SOAP messages over HTTP. The VWS provider is also in charge of implementing a simple WS client able to invoke the VWS.

Once a VWS has been defined, it will be published in the DVB-H broadcast network by using ESG entries. So, the standard discovery method in WS technologies (lookup in a UDDI registry) turns into using a navigator in the mobile device for exploring the ESG. Apart from including VWS entries in the ESG, the corresponding WS clients for the VWS, light clients, are added to the data carousel. The objects in that carousel are referenced by the VWS ESG entries (using the Acquisition element). When the user selects a VWS entry in the ESG, the light client is recovered from the carousel and executed. This light client obtains the location of its corresponding VWS from the information included in the ESG. Finally, it is ready to communicate to the VWS using the bi-directional network, 2G/3G. Remark that the only task of this light WS client is invoking the VWS since all the logic for provider selection has been moved to the intermediary element in the architecture: the VWS engine (Fig. 6).

VI. SCENARIO: IMPLEMENTATION DETAILS

The traffic information service we proposed in the scenario of Sect. II is a VWS grouping different traditional WSs provided by different traffic information centers, like transportation offices and drivers associations.
The VWS provider must define this new virtual web service by using VWSDL. Following the development cycle, he/she must know the individual WSs belonging to the information centers in order to define the public interface of the VWS and the appropriate mappings. Thus, the VWS provider is obliged to know all the details of each WS. Any VWSDL description must contain, at least, a list of the methods provided by the service. Additionally, for each method published inside a service, the input and output parameters and how their corresponding data types are mapped must also be specified.

For our scenario, the fragment shown in Fig. 7 focuses on the virtual method traffic. Each virtual method has name and type attributes, as well as it defines its virtual input and output parameters. As input parameter, the method traffic defines the location of the mobile device (the John’s DVB-H car navigator), being the output parameters both the traffic_flow and the existence and severity of any incident.

The type attribute of the virtual method is used to specify the concrete kind of implementation: for instance, equivalent means there is a set of “equivalent” services available to accomplish the execution of the virtual method; just as occurs in our virtual method traffic. Other values for the type attribute are used to specify different engine behaviors (sequence, alternate, parallel, iterative ...). This set is specified using as many invoke elements as needed. In our example, the method traffic specifies as equivalent the two real methods with identifiers TF1 and TF2, respectively.

Since SOAP parameters are nominal, an invoke description can use a map field which establishes the correspondence between virtual (input/output) and real (in/out) parameters. For instance, the input parameter location of the virtual method should be mapped into the input parameter loc of the real method traffic_report. So, the VWS engine maps the in/out parameters before and after each invocation; if mapping involves different data types, the conversion must be specified by using XPath expressions via the type attribute.

Besides, VWSDL descriptions can contain information to automate the WS provider selection according to QoS factors and/or to specify the most suitable caching, as follows:

A. Implementation of QoS

When developing software that uses web services, the uncertainty inherent to Internet (unpredictable workload, uncertain and highly oscillating number of users, etc.) must be taken into account to select the most adequate service provider. Thus, not only the availability is a key factor to select one service provider or another, but also the QoS they are able to offer (reliability, response, time, etc.). VWSDL considers this important issue by allowing the VWS provider to specify different selection criteria.

Therefore, it is possible to define different classes of service (CoS) by using the select element of VWSEL. In our example we have defined two different classes: COS#1 and COS#2. Linked to each CoS, the VWSEL allows defining expressions to specify in which proportion each quality factor is important. In our example, COS#1 gives more priority to ‘response time’ than to ‘cost’; whereas COS#2 specifies the opposite behavior. If the VWS client does not specify the preferred CoS, the default one is used (COS#1, in this case).

User-defined Quality-of-Service parameters make possible to model third-party rating systems. It is possible to include a trustworthy rating system which, for instance, qualifies the different traffic info providers according to their prediction quality.

Finally the QoS criterion for provider selection can also be determined directly by the client, that is, not in the VWSDL description which resides in the VWS provider. In this case, the SOAP message for the invocation of the VWS includes a SOAP header with the select element.

B. Introducing caching

Additionally, the VWS engine can act like a caching system. This kind of caching systems avoid repeated invocations to the same ‘read-only’ web service with the same request data, which often produce the same response from the provider. For virtual WSs, the benefits of using a cache-based system are even more important, because the VWS can take advantage not only of the response of one WS provider, but also of the responses of all the involved WS providers.

The characteristics of caching are specified in the VWSDL description by means of the cache directive; for instance, when caching is used (from/to) and the caching period (interval) in our scenario. We assume the caching period must be shorter in the rush hour (from 08:00 to 17:00), because the flow of vehicles is higher, which entails more variability in traffic conditions. However, the caching period could be larger from 17:00 to 08:00, because the less traffic flow, the less variability in road conditions.

Authorized licensed use limited to: National Cheng Kung University. Downloaded on April 10, 2009 at 01:14 from IEEE Xplore. Restrictions apply.
VII. RELATED WORK AND DISCUSSION

In computing, virtualization is a broad term that refers to the abstraction of computer resources, the base of this technique is hiding the physical characteristics of computing resources from the way in which other systems, applications, or end-users interact with those resources. This includes making a single physical resource (such as a server, an operating system, an application, or storage device) appear to function as multiple logical resources; or it can include making multiple physical resources (such as storage devices or servers) appear as a single logical resource. Since virtualization is being successfully applied on different computing environments (such as storage virtualization, network virtualization and hardware virtualization), our previous work in this field was focus on applying virtualization to the Web Services area (Sect. IV). In fact, by means of defining new virtual Web Services it was possible to achieve a high degree of decoupling and independence between clients and providers, much more than the degree supported by standard Web Services.

Being more precise, the virtualization schema we have proposed is innovative as a global solution for a range of problems that have only been addressed individually so far. In fact, aspects related to SLA management, quality of service, high availability or caching are the subject of study by public and private entities. However, the solutions proposed are specific for each one of these issues: architectures and languages to support SLA management [9]; sets of metrics for QoS [10]-[11]; software and hardware architecture to improve the availability of the implementation of web services [12]; and mechanisms to improve the performance of WSs by using caching systems [13]-[14]. Additionally, the use of intermediate elements (engines in our proposal) is a technique that is being implemented in some software platforms, but always with a specific use and using proprietary languages and/or systems. For example, WS-DBC [15] employs an intermediate element as a security system, while WS-Gateway [16] isolates the private networks of clients and/or providers, also supporting certain protocol changes (from SOAP to HTTP/POST, for example).

However, all these improvements do not focus on the WSs interface, which is what clients perceive as a web service. On the contrary, we have proposed to define and use a common interface, which at the same time provides a standard way to construct the interfaces that intermediate elements must offer. We have proved the success of this virtualization as well as clustering and caching techniques [6]-[8] by testing in our lab (http://idtv.det.uvigo.es).

Virtualization is especially suitable for mobile environments, where devices are characterized by their low capabilities and nomadic use. Therefore, we have applied our virtualization scheme in a mobile scenario over the DVB-H communication infrastructure. In fact, merging these two aspects (VWS and DVB-H) suppose a further step in our research work in both WS and digital TV area (DVB-H network supports mobile TV).

To sum up, we define an intermediate layer between the WS provider and the WS mobile client where virtualization is encapsulated, hiding complexity, availability and location problems to mobile clients. Besides, as complexity resides in the virtual Web Services, we have light clients, more adequate for mobile devices. Finally, by combining virtualization of services (through VWS) and broadcast infrastructures (FLUTE carousel over DVB-H); we enable mobile devices to access Web Services through the new open Web.

In the scenario proposed in this paper, it is assumed that the user navigates the ESG in order to find the service he needs or, on the contrary, a more sophisticated utility exists which allows to find a specific service from the user requirements. Regarding this topic two lines of future work are today under way in a mobile TV context.

Firstly, it could be interesting that a VWS can be linked to audiovisual contents which are conceptually related to it. The most standardized way to synchronize the WS light client to a TV program is via the ESG. However, the current specification of the ESG does not allow linking any element to a specific temporal instant but only to the whole service. It is expected that the new release of the ESG will improve the specification in this respect.

Secondly, in order to meet the habits of mobile clients, which access to DVB-H services at snatched moments, instead of a reactive navigation application, it could be interesting to offer a proactive agent which knows the user preferences and inform them about the availability of appealing WSs. In this line, we plan to apply our previous work in recommendation algorithms, to be precise, the result obtained in AVATAR [17], a TV contents recommender system.

ACKNOWLEDGMENT

The authors wish to thank Manuel Ramos, Marta Rey, Yolanda Blanco, Martín López and the EUREKA ITEA Project PASSEPARTOUT for their help in this research.

REFERENCES


Rebeca P. Díaz Redondo was born in Sarria, Spain in 1974. She is an associate professor in the Department of Telematics Engineering at the University of Vigo, and a member of the IDTV laboratory. She received the Ph.D. degree in Computer Science from the University of Vigo in 2002. Her major research interests are interactive applications for TV, like t-learning, as well as how they interact with the smart home environment.

José Juan Pazos Arias was born in Bayona, Spain in 1964. He received his degree in Telecommunications Engineering from the Polytechnic University of Madrid (Spain-UPM) in 1995, and his Ph.D. degree in Computer Science from the Department of Telematics Engineering at the same University in 1995. He is the director of the Networking and Software Engineering Group in the University of Vigo, which is currently involved with projects on middleware and applications for Interactive Digital TV.

Alberto Gil Solla is an associate professor in the Dept. of Telematics Engineering at the Univ. of Vigo, and a member of the Software Engineering Research Group. He received his Ph.D. in Computer Science from the University of Vigo in 2000. He is involved with different aspects of middleware design and interactive multimedia services.

Jorge García Duque was born in Leon, Spain in 1972. He is an associate professor in the Department of Telematics Engineering at the University of Vigo. He received the Ph.D. degree in Computer Science from the University of Vigo in 2000. His major research interests are related to the development of new software methodologies and services for Interactive Digital TV.

Ana Fernández Vilas was born in Vigo, Spain in 1971. She received her Ph.D. degree in Computer Science from the University of Vigo in 2002. In 1997, she joined the Department of Telematics Engineering (University of Vigo) where today she is an associate professor. She is engaged in web services technologies and ubiquitous computing environments, being a member of the Interactive Digital TV Lab.