Effective Web Service Discovery in Mobile Environments

Khalid Elgazzar, Hossam Hassanein, Patrick Martin
School of Computing, Queen's University, Canada
{elgazzar, hossam, martin}@cs.queensu.ca

***Abstract*—Recent advancements in the design of mobile devices and wireless technologies have produced a successful coupling of mobile devices and Web services, where mobile devices can be a service provider or a consumer. However, finding relevant Web services that match requests remain a major hindrance to its booming. The challenges facing Web service discovery are further magnified by the stringent constraints of mobile devices, and the inherit complexity of wireless heterogeneous networks. While significant research has focused on service discovery protocols in isolation, they mostly lack a holistic capacity to address the different limitations collectively. We introduce a novel discovery framework that addresses all aspects of mobile Web service discovery, yet does not jeopardize the efficiency requirement for this discovery; especially as an application run in resource-constrained environments.**

***Index Terms*—mobile Web services, Web service discovery, service discovery framework, mobile devices**

I. Introduction

Service Oriented Architecture (SOA) [1] is a vision of developing software applications that can be composed of networked functionalities that are capable of interacting with each other automatically. SOA offers the promise of robustness and agility by supporting software reuse, loose coupling, flexible design, and interoperability between heterogeneous applications. With the emergence of the Mobile Internet (MI) and the constantly expanding mobile user base, the notion of networked functionalities helps end-users with limited resources to perform complicated tasks on high-end machines over the network. Web services implement the concepts of SOA to enable "*software as a service*", which delivers software services over the network using technologies such as XML.

Web services, however, have neither achieved the anticipated wide spread use, nor dominated Web applications development. Web services have, so far, failed to match the burst expansion of the Web. A main reason is the inefficiency of Web service discovery. That is, the ability for the user to find Web services that meet his/her needs and are appropriate for the current context [2]. The standard Web service development cycle includes description, discovery, and invocation (remote execution) [3]. Web service discovery in particular, is a key enabler to the adoption of Web services technology as a computing paradigm.

With the emergence of new generations of smartphones and high-end mobile devices, both equipped with extraordinary features, users became capable of running sophisticated Web applications. Furthermore, the unprecedented advancements of wireless technologies influence mobile users to become highly interested in the Mobile Internet. With the limitations existing in wireless networks and the limited resources of mobile devices, Web service discovery becomes even more difficult, and hence, challenging.

Broadly speaking, current service discovery techniques are fundamentally designed for static and wired environments. Most of the existing discovery approaches lack the comprehensive understanding of both mobile devices and wireless networks limitations, which makes these approaches incapable of efficient and reliable discovery in a mobile scenarios.

In this paper, we introduce a novel discovery framework that addresses all aspects of mobile Web service discovery in resource-constrained environments. For the various aspects of the framework, we identify approaches that could potentially be applied from current research.

The contributions of this paper can be summarized as follows:

- We provide an overview of existing Web service discovery approaches and point out their limitations in mobile environments. As well, we identify the essential requirements for a successful Web service discovery in resource-constrained environments.
- We propose a generic Web service discovery framework that can be used in a general Web service environment, but more specifically for mobile domains and resource-constrained environments. The framework can be implemented in client/server or P2P networks with sufficient robustness to implement components where they would perform most efficiently.
- We propose a novel request submission protocol that identifies the available service providers as well as the language used to describe the services they offer before submitting the service request.

The remainder of this paper is organized as follows. Section II outlines related research. Section III gives a brief background on Web service descriptions. Section IV discusses the current discovery approaches, points out the limitations, and identifies the essential requirements for efficient discovery in mobile environments. Section V describes the proposed framework and relevant research efforts that are applicable for each component. In Section VI we present a use-case scenario to illustrate how the proposed framework can be applied. Finally, Section VII concludes the paper and outlines future research avenues.

978-1-61284-928-7/11/$26.00 ©2011 IEEE

II. Related Work

Over the past few years, researchers have focused on optimizing specific parts of current Web service discovery approaches to enable them to fit within mobile domains. These aspects include: semantic reasoning strategies for pervasive discovery [4], [5], [6], [7], [8]; location-based mobile services [9]; context-aware discovery [10], [2]; incorporating user preferences with the discovery process [11]; device-aware discovery [12]; capacity-driven service discovery [13], [14]; service composition in a mobile environment [15]; dynamic discovery for P2P networks [16], [17]; future vision for discovery schemes in open mobile environments [18], and adaptive interfaces and web content presentations for mobile devices [19]. Although there have been many research efforts related to particular aspects of discovery, within heterogeneous mobile environments, there is no single publication that provides a complete picture of the discovery process.

III. Web Service Description Standards

A Web service is a computational software entity which is able to achieve a user's objective by a remote invocation. Web services allow applications written in different programming languages to interact seamlessly through standard protocols [20]. Web service standards such as description, discovery, publication, and invocation offer mechanisms for both service providers and consumers to carry out their tasks in a universal and standard way. Once the service is developed, the provider describes the service functionalities, operations it performs, functional and non-functional parameters , and how potential customers can communicate with the service. Description languages include semantic and non-semantic approaches.

WSDL 2.0 [21] is the latest standard specification for non-semantic services. It describes the service in two levels; "*abstract*" and "*concrete*". The *abstract* level describes the operations that can be performed by the service and the message structures used to communicate to these operations, as well as an interface which combines messages and operations. The concrete level specifies the service bindings associated with the network endpoints.

Semantic descriptions of Web services rely on domain *ontologies* [22] which aim to provide unambiguous definitions of the description terms and to address the lack of semantic understanding of messages and data, which consequently makes the interactions between services more logical, and facilitates service composition and integration. Web services may use various semantic description languages such as Web ontology languages (OWL-S) [23], Web Service Modeling Ontology (WSMO) [24], WSMO-Lite [25] for resource-constrained environments, Web Services Semantics (WSDL-S) [26], and Semantic Web Services Ontology (SWSO) [27]. Unfortunately, each description language has its own notations and no standard, universally accepted formal notations exist for sematic descriptions. Services that are described in a particular formalism are only discovered by requests constructed by the same formalism. Using the same formalism to describe both the service and the request may result in a mismatch between the request description and user intuition [28]. With the emergence of mobile Web services provisioning, which are services provided by mobile devices over wireless networks, most of these description languages must be revisited to accommodate device and network constraints.

IV. Service Discovery

Service discovery is the act of finding a relevant service for a particular request. Most of the existing techniques belong to one of three main discovery approaches [29]: UDDI Business Registry (UBR), specialized search engines, and generic search engines. Each one of these approaches has its strengths and weaknesses.

-*UDDI Business Registry*: UDDI is the discovery approach used by the standard Web service architecture. It relies on centralized repositories that providers use to publish their services and customers use to discover services that satisfy their requirements. Usually, UBR provides information about the service description, publisher, endpoint, technical interface (tModel), implementation, etc. This approach has not been widely adopted by the Web services community which explains why major UBR (such as IBM and Microsoft) shut down their services in 2006 [30]. However, there are still few public registries offering their services with different capabilities such as RemoteMethods, StrikIron, and X-Methods.

Problems with UBRs include the centralized architecture, limited scalability, single point of failure, consistency maintenance, searches that rely on keywords or category browsing only, and outdated service records. Customers also need to be aware of the UBR addresses to locate and query them. However, UBRs enable service subscription for interested users to keep them updated. UBRs add extra features such as service trail, transaction facilitation, WSDL parser, different pricing schemes, performance monitoring, programmatic interface, ratings, categorization, documentation, etc. UBRs also allow searching for providers and tModels [30], [29].

-*Specialized search engines*: This approach aims to distinguish a Web service search from a Web content search. The basic idea is to make use of Web services functionalities, operations, and other information provided in the description files in order to perform a meaningful search for services that best match a particular request. These search engines collect Web services description files from public UBRs and Web contents, extract the semantic meaning of these Web services from their description files, and perform semantic matching between requests and Web services capabilities. Woogle [31] and WSCE [32] are examples of these search engines.

Web services search engines are able to find services that are more relevant to users' requests as the search does not only rely on keywords but also on functionalities and other running parameters such as QoS. Additionally, the retrieved services should be valid and running as these engines are able to catch any updates or status changes while crawling the descriptions of Web services from the source.However, so far this approach supports only searching for non-semantic Web services.

-Generic Web search engines: Web content search engines are another alternative to find Web services using keyword search. Major providers of Web services, such as Google, Amazon and Yahoo, have decided to publish their Web services through their own websites instead of using UBRs. This trend is forcing users to discover Web services through Web content search engines. Users can use search engines. to locate Web services by customizing the search query to look for specific files types (ex. wsdl and owl files).

The major drawback of generic search engines is that they cannot understand Web service functionalities outlined in the description files and only rely on keywords to find services. The advantages of them include robustness, scalability, and no extra infrastructure is required.

A. Limitations of Current Discovery Mechanisms: A Mobile Perspective

Notwithstanding the research efforts that have focused on Web service discovery, many limitations with respect to mobile environments remain, including the following:

- Current approaches do not take into consideration the support that peers can provide in mobile domains.
- Current approaches lack open architectures which can guarantee robustness and scalability.
- User experience and satisfaction, user preferences, and device features and capabilities are important factors that must be incorporated into Web service discovery in mobile environments. A few proposals provide limited consideration to these factors. They tend to focus on either user preferences or mobile capabilities and ignore user channel or the network status.
- Current approaches do not take into account resource-constrained providers that may exist in mobile domains. These providers have limited resources to, for example, perform semantic matchmaking quickly and efficiently.
- In wireless networks, providers and customers communicate over wireless channels where signal quality is variable. Existing discovery mechanisms are unable to identify Web services that are able to promptly respond and adapt appropriately to such context change.
- In a mobile environment, selecting services that are located in physical proximity, perhaps belonging to the same home network, for instance, can reduce network traffic and, as a result, reduce costs. Identifying services that provide this selection priority is currently not available in existing approaches.
- Keyword-based service discovery is ineffective in retrieving the most relevant services to a specific request, while semantic-based discovery is resource intensive and not affordable for resource-constrained environments. Lightweight semantic reasoners are therefore required.

B. Service Discovery Requirements in Mobile Environments

Service discovery is a crucial component in Web services, especially in heterogeneous mobile environments. The process must be efficient and rapid to cope with the extremely dynamic nature of mobile domains. To ensure these features, service discovery in mobile domains must satisfy the following requirements:

1) In mobile domains, mobile devices are frequently changing their point of attachment to the network or making a vertical handoff between different access technologies. The services they provide then become inaccessible as their binding information become invalid. Service discovery should be an active process even while the service is executing to support seamless provisioning either by providing an alternative access to the same service or by quickly finding another equivalent service.
2) With mobile Web services provisioning services may become invalid or stale due to the provider's mobility. Discovery mechanisms in mobile environments must ensure that the discovered services are active and running.
3) With the existing diversity of mobile device form factors and platforms, discovery mechanisms should ensure the compatibility of the discovered services with the device capabilities.
4) Since mobile users are always on the move, service providers and/or service consumers could be location-dependent, hence request or offer services in particular areas. So, location-based service discovery is highly required.
5) Typically, mobile devices are associated with context information and users who usually have preferences. One of the chief benefits from mobile services is the enabling of personalized services provisioning that takes into account the user preferences and context information. Service discovery in such cases must incorporate the user profile and context information in ranking and selecting Web services.
6) Semantic Web service discovery architectures present a significant challenge in mobile domains. The discovery of semantic Web services requires a heavyweight matchmaking process at the server side which could be a limited-resources provider. Semantic reasoning, the core of the semantic matchmaking process, is a resource-intensive process and only suitable for deployment on high-end servers. Therefore, highly optimized semantic reasoners for mobile environments is required.
7) Generally speaking, finding a single service that fulfills a particular request is not the only way to respond to a request. Service discovery should be smart enough and be able to break down the request into small sub-tasks, if possible, and find a service for each sub-task separately before integrating them together to satisfy the original request.
8) Discovery approaches must be able to distinguish between finding services that interact with applications (technical services) and services that will interact with users (user-facing). The later, which is more desirable by mobile users, needs an easy-to-use interface that enables users to perform efficiently the different operations that

the service offers. Therefore, Web services that provide a more appealing (user-friendly) user interface would be ranked higher.

9) Due to the limitations of mobile devices and wireless networks, mobile clients may be able able to execute/support limited operations offered by Web services. Discovery protocols should be able to identify what services may not be fully executed and rank discovered services accordingly.
10) Mobile domains are highly dynamic and experience many environment changes. Therefore, developing services with multiple capacities (different behavior to the same functionality) is becoming more desirable and an essential requirement. Discovery protocols should be able to assign higher priority to Web services that are capable of adapting their behavior to environment changes (ex. Bandwidth and transmission rate).

V. Service Discovery Framework for Mobile Environments

Based on our analysis of the limitations of existing discovery mechanisms presented above, we propose a framework for mobile Web service discovery in resource-constrained and mobile environments. Figure 1 illustrates the essential components, represented in a three-layer structure, that are required for effective Web service discovery in mobile environments. *Layer 1* represents the components that reside on the customer side, and *layer 3* contains all the components that run on the provider side. *Layer 2* depicts the components that can be implemented either on the provider side or on the customer side based on resource availability and battery power. The decision whether to implement some components on the provider's side or the customer's side aims to enhance the performance. The framework is also independent of the Web service architecture. It supports service discovery in Peer-to-Peer (P2P) and client/server architectures. In P2P Web services provisioning, components at this layer may be distributed between providers and customers according to their resources, while in client/server architecture where servers are usually high-end machines, mobile clients can host the minimum number of components to save their precious resources and battery power.

A. Layer 1: Customer Layer

Service Request: The end-user constructs a request for a Web service that fulfills a specific objective. The request can be as simple as plain text that describes the user objective. Mobile users usually have limited input capabilities, which are unsuitable for a formatted service request or a formal service description language (i.e. in semantic services). Therefore, simple plain text fits well for the service request within mobile device constraints. The *Request Analyzer* then can perform some extra processing on this request to extract some meaningful information and construct a format service request. Another alternative is to provide a user-friendly multimodal interface that accepts user requirements with minimal input parameters.

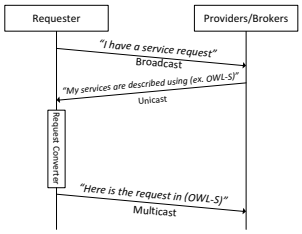

Fig. 2. A high level illustration of the proposed request submission protocol.

Request Submission Protocol: The Request submission protocol starts at the customer's side. The protocol broadcasts a brief message saying that "*I have a service request*" and can be encoded in a short format. Service providers/brokers who receive this message respond with a unicast message to the requester saying that "*I am a service provider and my services are described in a service description language (X), i.e. WSDL, WSDL-S, OWL, etc.*". The responses are unicasted to the *Service Request Converter* to format the request in the appropriate formalisms. Figure 2 depicts a high level abstraction of the proposed request submission protocol. The *Request Submission Protocol* takes advantage of P2P networks to disseminate the service request.

In P2P networks, peers typically cooperate to disseminate the service request. Peers receive the request, process it, and originate the request to their neighbors after reducing the Time To Live (TTL), that is associated with the request, by 1. Peers stop forwarding the request if (TTL = 0). The *Request Submission Protocol* aims to reduce the network traffic produced by request flooding and avoid message processing by non-corresponding nodes to save their valuable resources (i.e. battery power and CPU).

B. Layer 2: Provider/broker or customer side

This layer contains the components that can be implemented at either the provider/broker side or at the customer side according to resources availability.

Service Request Converter: Upon receiving the service request and the responses from different available providers, the request converter re-constructs the service request in the provider's specific language. If the *Request Converter* is implemented at the customer side, it then re-constructs (depending on previous responses) the request in multiple description languages and multicasts each formalism to the corresponding providers. However, if the *Request Converter* is implemented at the provider/broker side, the conversion is done locally. In such case, the benefits that the *Request Submission Protocol* brings would be limited.

Request Analyzer: The service discovery process in this framework is comprised of two levels. *Level 1:* The *Request Submission Protocol* searches first for atomic services that totally satisfy the user's request. *Level 2:* If no relevant services are found, the service *Request Converter* is notified to

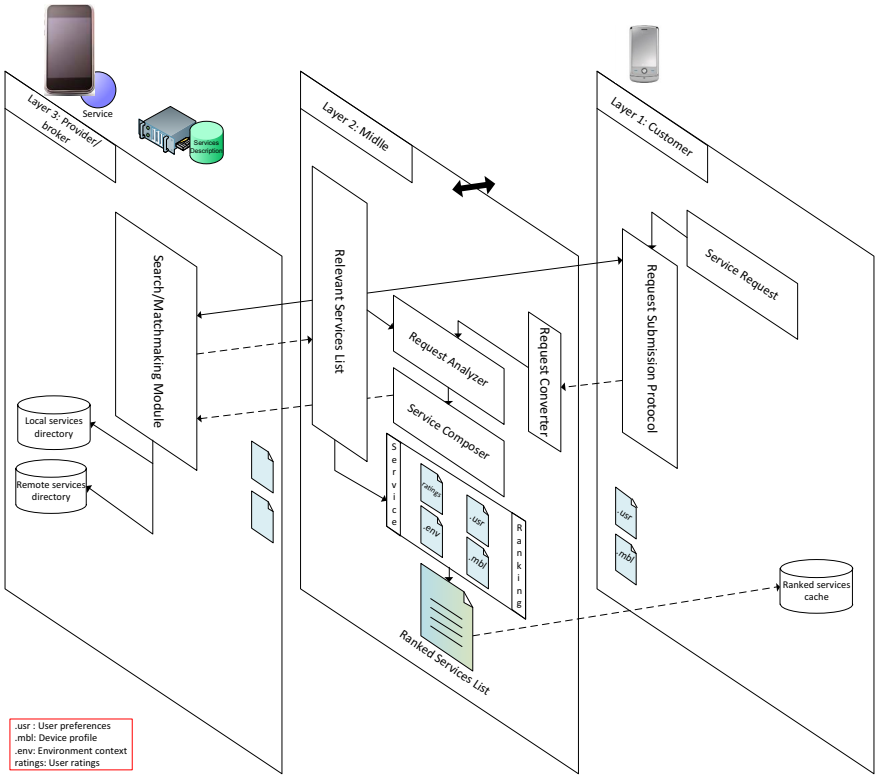

Fig. 1. An overview of the discovery framework in a three-layer structure.

forward a copy of the service request to the *Request Analyzer* which, in turn, will break down the request into sub-tasks, if possible, and try to satisfy each sub-task separately. Breaking down the request (in collaboration with the *Service Composer*) may follow one of two approaches: 1) Start with the request inputs to get the request outputs with exact or partial matches using, for example, the matchmaking algorithm mentioned in [15], 2) Find services that have exact matches with sub-tasks, i.e. the matchmaking algorithm yields an *EXACT* or *SUBSUMES* match between the sub-task inputs/outputs and the relevant service inputs/outputs. *EXACT* match means that all the sub-task inputs and outputs match exactly with the service input and output parameters respectively. *SUBSUMES* match means that all the sub-task inputs and outputs are subsets of the service input and output parameters respectively.

Service Composer: The service composer in our framework is responsible for the orchestration process and generating composition plans for atomic Web services that satisfy the individual sub-tasks, together to fulfill the original service request. Resolving users' requests via composite services (when it is applicable) meets requirement #7. Bhuvaneswari et al. [15] propose a framework for semantic Web service composition in mobile environments. Their framework converts WSDL files into OWL-S specification and generates a service profile for the request. Then, it performs a sematic reasoning between the advertised service profile and the request service profile. The composer then generates composition plans and stores them in a plan repository in a cloud.

Relevant Services List: This is the list of services that match the user request. The list contains atomic Web services that satisfy the user objective, and/or composite services along with composition plans and the plan evaluation to be used by the execution environment.

Service Ranking: The service ranking component receives the list of relevant Web services and works to rank them based on the user preferences, device profile, running environment conditions, and user ratings. Al-Masri [12] have developed a device-aware mobile service discovery solution called MobiEureka. The system is capable of ranking discovered services according to their best fit with the requester's mobile device. García et al. [11] proposed a user preferences model to be integrated with service description languages so that discovery mechanisms can rank relevant services accordingly. However, in our framework, we suggest ranking the relevant services based on the user preferences, device profile, environment context, and user ratings collectively.

Context Information: The framework uses the aforementioned four types of context information to rank the list of relevant Web services (if found). Incorporating these context information in mobile service discovery fulfils requirement #3,4,5,9 and 10.

`User preferences`: User preferences are not part of the semantic Web service descriptions, which implies that retrieved services may satisfy a user's request, but may not

be quite relevant to the user preferences (requirement #5). García et al. [11] propose a detailed user preferences model that can be applied as an extension to the existing semantic description languages. The model distinguishes between mandatory requirements and preferred requirements. However, we recommend expressing user preferences following the same model, but in separate files to ensure modularity, portability, and preserve the backward compatibility.

`Device profile`: The objective of incorporating the device profile information with the Web service discovery process is to ensure the compatibility of the discovered/selected service with the device constraints. In this regard, Al-Masri et al. [12], have developed a device-aware service discovery mechanism that is capable of selecting Web services that function properly within mobile device constraints. The mechanism takes advantage of HTTP sessions to collect device information and store it at the server side. This information is used later to ensure that the discovered services will function properly within the device constraints and to rank them accordingly (requirement #3). The mechanism supports caching the information of disconnected devices to be used if they successfully re-establish the connection within a specific period of time. The solution is limited to WSDL files and an extension to WSDL-M is proposed in order to include "required" features/ parameters on the client's device. Again, we recommend that service providers describe the "required" and recommended "to-have" features in mobile devices in a separate file for the same reasons mentioned above.

`Environment Context`: In wireless networks, the environment context may change frequently during the execution of the Web service as mobile consumers and/or providers may be on the move. Services behavior may be considerably affected by these changes. This impact is not of concern for traditional Web services provisioning. The objective of integrating the environment context into service discovery is to ensure that services run properly given these environment conditions/parameters. We prefer that providers identify the minimum and preferred required parameters for reliable execution in a standalone file that can be associated with the other service description files. By consulting this file while ranking the discovered Web services, the discovery process may realize that the requester could be able to execute limited functionalities of particular services and then rank services accordingly (requirement #9). Another opportunity that can be envisioned here is the support of multiple capacity services, where services can adapt to the current environment status by changing their behavior. Therefore, a discovery protocol would assign higher priority to services with multiple capacities (requirement #10). Consequently, the service execution platform/architecture should support the appropriate mechanisms for assessing the environment status at runtime so that services with multiple capacities would be able the respond promptly to the environment changes. Environment context also includes location information, which can be used to enhance the service provisioning according to the user's location (requirement #4). For example, providing a list of restaurants that are located nearby the requester.

Maamar et al. [13] discuss the development of capacity-driven Web services starting from the description, discovery, composition, to the invocation of proper desired functions. Similar research on services with different qualities to cope with environment context is presented in [14]. The authors named their approach "Service Differentiation" which aims to develop/provide a single service with multiple variations instead of several independent services.

`User ratings`: In Web 2.0 and open environments users are encouraged to provide feedback and rate different services they have used based on their experience. These ratings are important as they reflect the user satisfaction and QoS perceived by the user. The service rating could also be used as an effective tool to show the real user satisfaction for the user interface of user-facing services, which fulfils requirement #8 of our defined effective service discovery reuirements. Our framework takes advantage user ratings to rank Web services, given that the proper handling mechanisms for ratings, complaints, and comments are provided.

Ranked Services List: The ranked services list is the services list from which the user/application would choose the proper Web service to invoke.

Relevant Services Cache: Mobile environments are characterized by intermittent connections, unreliable channels, and high transmission error rates. Services may become easily unavailable or function improperly due to the lack of resources (ex. bandwidth). In such cases, service discovery mechanisms need to support alternatives either by providing different access to the same service or finding functionally similar services (which satisfies requirement #1). In this perspective, our framework proposes caching the retrieved relevant services list for a user specific request to choose the next best candidate service in case of the principle service fail to respond or not performing well. This caching is cleared once the desired service is successfully executed.

C. Layer 3: Provider/broker Layer

This layer contains the framework components that should be implemented at the provider/broker side.

Search/Matchmaking Module: The *Search/Matchmaking* module is where the functionalities described in the request and the capabilities offered by Web services are matched. Service providers decide on the appropriate matching approach that is used to match users' requests based on how they describe their services. For non-semantic Web services, i.e. described by WSDL files, the matching between the request and Web services is keyword-based and uses information retrieval techniques [33]. In this case, the Web services can be characterized by sets of keywords extracted from the description files. These keywords can be used to index Web services and later matched with keywords extracted from the user request [17]. Clustering techniques can be used to categorize WSDL files based on functional similarities to bootstrap the discovery process of non-semantic Web services [34]. Semantically described Web services are discovered

using high level match-making approaches. The most popular sematic discovery methods are OWL-S based and WSMO based approaches which, fundamentally use the information provided in the service profile and domain ontogies to match the user's requested functionalities [35].

In mobile domains, where mobile devices perhaps could be service providers, the matchmaking process is performed on mobile devices. Therefore, conducting the reasoning process on a resource-constrained device might possibly fail or produce out-of-memory/stack overflow errors due to insufficient resources. Highly optimized semantic reasoning in such cases is required. Steller el al. [6], [7], [8] propose the mTableaux algorithm to optimize the reasoning process and facilitate Web services selection for resource-constrained devices. Gu et al. [4] discuss in their framework, the design principles and implementations of supporting ontology and reasoning for mobile context-aware applications on handheld devices. Using such lightweight semantic reasoners for resource-constrained providers meets requirement #6 of our effective discovery requirement mentioned before.

Local Services Directory: Service providers maintain a local directory that contains all Web services offered by them.

Remote Services Directory: Remote services are services hosted and provided by other peers/providers, but could be proxied by other peers. Each Service provider may cache access to these services for future references. A coordination protocol is required to manage link updates, advertisement notifications, invalid service/link removal, and duplicate reference avoidance. JXTA protocols [16] are commonly used in this regard in P2P networks. JXTA periodically advertises services as JXTA modules. Advertisements are associated with a lifetime that determines how long peers would cache these services and mark them as valid. This feature satisfies requirement #2, where services are removed/marked invalid when their lifetime expires. However, maintaining such a directory for remote services has advantages and disadvantages. The advantages include, back up for temporary disconnected peers that may provide relevant services, enable services provisioning via a proxy, and reduce network traffic by communicating less providers. The disadvantages include more resource consumption on particular peers and duplication management overhead (in case of a reference to the same service reported from different sources). A good tradeoff approach must be made to ensure efficient and reliable discovery while performing better optimization for heterogeneous mobile environments.

VI. Use-case Scenario

Suppose that Adam is interested in attending an international multicultural event held in his hometown. Adam planned to take the bus to reach the event location before the opening ceremony, which will start with a speech by the mayor. Adam is excited and keen to catch the mayor's speech. While waiting at the bus stop the info panel showed that there is a delay in the arrival of the bus Adam planned to take, which means Adam would miss part of the opening ceremony.

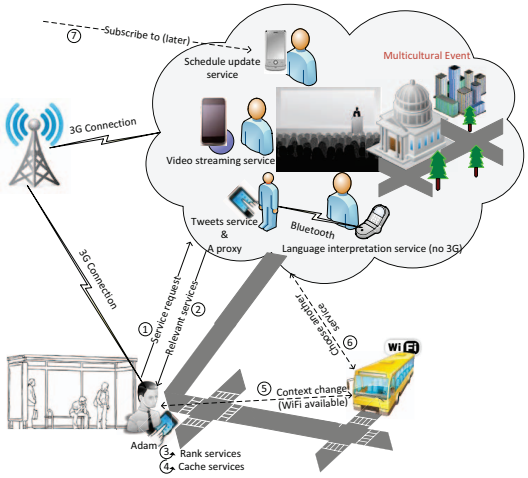

Fig. 3. A graphical illustration of the use-case scenario.

At the event location, some attendees (peers) are offering coverage to the event in various forms and with different capabilities (multiple-capacities). Some high-end smart phones are offering Web services that provide multiple-capacities, i.e. video streaming, audio, and photos to cope with the user available bandwidth. Other mobile providers are offering real-time interpretations to different international languages including, Spanish, German, French, and Arabic using their short-range wireless connectivity for the event attendees (peers). However, a few capable providers are offering a proxy service for other peers that are not connected to the Internet, so that their services could be reached by Internet users. Besides, different sections are offering a mobile Web service, hosted on their representatives' mobile device, to send schedule updates to subscribers. Other Web services are also available for audio, translations, photos, text tweets, and event schedule live updates as shown in Figure 3. It is worth mentioning that some providers are using OWL-S to describe their Web services while some others are using WSDL files. These Web services are published either locally on the provider's mobile device (local services directory) or using the available brokerage service offered by capable peers (remote services directory).

In this scenario, we will illustrate how Adam can use the framework to be connected to the event activities until he arrives on site. We assume that Adam has a powerful 3G-enabled smartphone with a multimodal wireless connectivity the can switch to Wi-Fi whenever it is available and as per the user settings. According to these assumptions, we recommend implementing the *Service Request*, *Request Submission Protocol*, *Request Converter*, and *Service Ranking* at the customer side, whereas the rest of the framework components are implemented at the provider/broker side. Figure 4 depicts our choice of where to deploy different component for this use-case and, in general, how different framework components interact together to discover the required Web services efficiently.

Adam, using his smartphone, uses the *Service Request* to construct a Web service discovery request. The service request

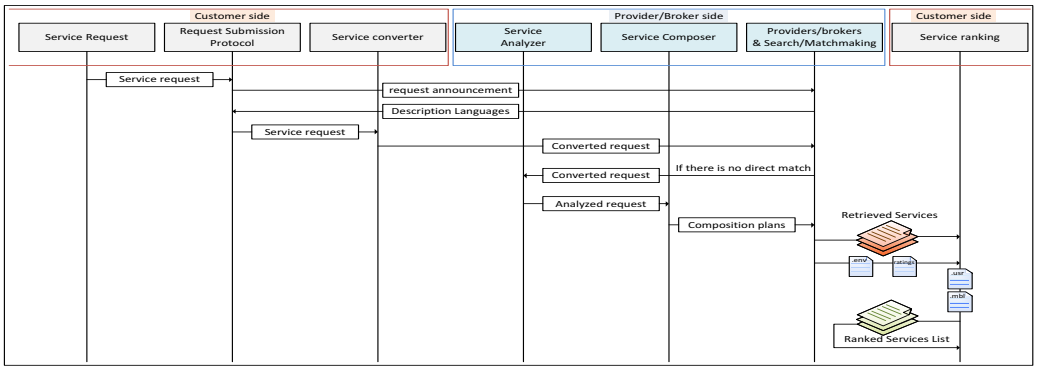

Fig. 4. The interactions between various framework components for the use-case scenario.

content is *"multicultural event coverage at my hometown"*, with *"Spanish interpretations and/or text translation"* as the user preferences. The *Request Submission Protocol* receives the request and broadcasts a request existence message "I have a Web service request". Upon receiving the request by the available service providers, each provider/broker replies back with a message to identify their description languages. The *Request Converter* then receives a copy from the service request and converts it to the appropriate formalisms. The *Request Submission Protocol* sends multicast messages with the converted request to the corresponding providers. The aim of the multicast is to avoid message processing by non-corresponding nodes and reduce the message flooding. Each provider matches the request with the available local and remote Web services. If no atomic Web services are found satisfying the customer's request, then the provider uses the *Request Analyzer* and the *Request Composer* to fulfill the request with composite services. The discovery framework then receives relevant services from different providers who found matches, along with the required/recommended environment conditions for each relevant service. Adam's discovery application ranking module integrates the received information and retrieved services list with the local context information (i.e. environment actual status and Adam's preferences) to rank the retrieved Web services according to the user preferences, device profile, environment conditions, and ratings.

To that end, suppose that the 3G signal strength and Adam's current location at the bus stop is relatively weak, which implies a low transmission rate and limited bandwidth. According to that status, our discovery mechanism framework ranks the service with multiple capacities first (i.e. the service provides photos for limited bandwidth, audio for moderate, and video streaming for high bandwidth availability). The ranking module in such a case would also rank audio services next, photos, Web services, text tweets, and then schedule updates. Adam's application provides the ranked list to Adam to choose from and invoke the selected Web service or the application takes the appropriate decision automatically by running the service that is ranked first and caches the rest in the relevant services cache. Upon the arrival of the required bus, Adam gets on the bus and immediately his smart phone changes automatically the Web connectivity to the bus Wi-Fi hot spot available onboard to match Adam's settings. The execution environment detects this change in the environment status and adapts the service behavior to provide real-time video streaming. The application may also re-evaluate the situation based on the new context change and choose from the relevant services cache the composite service plan that provides video streaming composed with a real time Spanish interpretation and onscreen translation provided by multiple providers.

Later on, if Adam wants to leave the event for some time and come back to continue enjoying the entertainment but wants to be updated about any changes on the event schedule for individual performances; the discovery framework may look at the relevant services cache and locate the services that provide schedule updates for different exhibitions. Adam then subscribes to the ones that he is interested in to get schedule updates.

VII. Conclusion and Future Work

Web service discovery is a key enabler to the adoption of Web services technology in mobile heterogeneous environments. In mobile environments, the limited resource availability and the unreliable communications in wireless networks presents unique challenges for service discovery. In this paper, we identified the limitations of current discovery approaches and the requirements of sound and reliable discovery mechanisms that can be used to efficiently discover Web services within resource-constrained environments. We proposed a generic framework for Web service discovery in mobile heterogeneous environments that can be implemented in client/server or P2P networks. The framework is presented in a three-layer structure and takes into account network characteristics, user preferences, device profile, and available resources at each participant. A comprehensive description is provided for each component along with the possible approaches that can be implemented as well as some of the research efforts in that respect. Some of the framework components can be implemented at either the customer or the

provider/broker side according to the available resources to achieve acceptable performance.

We claim that handling context information helps in providing personalized services in the best interest of the requester and most appropriate for the current situation. We believe that the proposed framework paves the road for a better understanding for development of robust discovery mechanisms that cope with the highly dynamic nature of a mobile environment. Mobile users would be more interested if services are tailored to their preferences and devices, and ones that are capable of changing their behavior to react promptly to the environment changes.

We plan to use the framework to develop a robust Web service discovery mechanism in mobile heterogeneous networks. We will take advantage of our research in mobile Web services provisioning to implement the framework component on a mobile provider and a mobile client. A smart decision-making module to choose where to implement different component at the middle layer is under investigation. We are also looking at evaluation schemes to evaluate the effectiveness and the performance of our framework.

REFERENCES

[1] M. N. Huhns and M. P. Singh, "Service-oriented computing: Key concepts and principles," *IEEE Internet Computing*, vol. 9, pp. 75–81, 2005.

[2] N. Blefari-Melazzi, E. Casalicchio, and S. Salsano, "Context-aware service discovery in mobile heterogeneous environments," in *Mobile and Wireless Communications Summit, 2007. 16th IST*, pp. 1–5, July 2007.

[3] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard, "Web services architecture," February 11 2004. http://www.w3.org/TR/ws-arch.

[4] T. Gu, Z. Kwok, K. K. Koh, and H. K. Pung, "A mobile framework supporting ontology processing and reasoning," in *Proceedings of the 2nd Workshop on Requirements and Solutions for Pervasive Software Infrastructures (RSPSI '07)in conjunction with the 9th International Conference on Ubiquitous Computing (Ubicomp '07)*, (Austria), September 2007.

[5] L. Steller and S. Krishnaswamy, "Optimised semantic reasoning for pervasive service discovery," in *Service-Oriented Computing ICSOC 2008*, vol. 5364 of *Lecture Notes in Computer Science*, pp. 620–625, Springer Berlin / Heidelberg, 2008.

[6] L. Steller and S. Krishnaswamy, "Efficient mobile reasoning for pervasive discovery," in *Proceedings of the 2009 ACM symposium on Applied Computing*, SAC '09, pp. 1247–1251, 2009.

[7] L. Steller, S. Krishnaswamy, and M. Gaber, "Cost efficient, adaptive reasoning strategies for pervasive service discovery," in *Proceedings of the 2009 international conference on Pervasive services*, ICPS '09, pp. 11–20, 2009.

[8] L. A. Steller, *Light-Weight and Adaptive Reasoning for Mobile Web Services*. PhD thesis, Monash University, Australia, May 2010.

[9] E. Kaasinen, "User needs for location-aware mobile services," *Personal and Ubiquitous Computing*, vol. 7, pp. 70–79, 2003.

[10] E. Al-Masri and Q. H. Mahmoud, "A context-aware mobile service discovery and selection mechanism using artificial neural networks," in *Proceedings of the 8th international conference on Electronic commerce: The new e-commerce: innovations for conquering current barriers, obstacles and limitations to conducting successful business on the internet*, ICEC '06, pp. 594–598, 2006.

[11] J. M. García, D. Ruiz, and A. Ruiz-Cortés, "A model of user preferences for semantic services discovery and ranking," in *ESWC 2010, Part II*, vol. 6089, pp. 1–14, 2010.

[12] E. Al-Masri and Q. H. Mahmoud, "Mobieureka: an approach for enhancing the discovery of mobile web services," *Personal Ubiquitous Computing*, vol. 14, pp. 609–620, October 2010.

[13] Z. Maamar, S. Tata, D. Belaid, and K. Boukadi, "Towards an approach to defining capacity-driven web service," *International Conference on Advanced Information Networking and Applications (AINA'09)*, pp. 403–410, 2009.

[14] A. Tao and J. Yang, "Context aware differentiated services development with configurable business processes," in *Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference*, pp. 241–252, November 2007.

[15] A. Bhuvaneswari and G. Karpagam, "Reengineering semantic web service composition in a mobile environment," *International Test Conference*, pp. 227–230, 2010.

[16] L. Gong, "Jxta: a network programming environment," *IEEE Internet Computing*, vol. 8, pp. 88–95, 2001.

[17] S. Sioutas, E. Sakkopoulos, C. Makris, B. Vassiliadis, A. Tsakalidis, and P. Triantafillou, "Dynamic web service discovery architecture based on a novel peer based overlay network," *Journal of Systems and Software*, vol. 82, pp. 809–824, May 2009.

[18] N. S. K. Bashah, I. Jørstad, and D. v. Thanh, "Service discovery in future open mobile environments," in *Proceedings of the 2010 Fourth International Conference on Digital Society*, ICDS '10, pp. 47–53, 2010.

[19] B. Adipat and D. Zhang, "Adaptive and personalized interfaces for mobile web," in *15th Annual Workshop on Information Technolgies & Systems (WITS'05)*, pp. 21–26, 2005.

[20] P. Prescod, "Roots of the rest/soap debate," August 2002.

[21] R. Chinnici, J.-J. Moreau, A. Ryman, and S. Weerawarana, "Web services description language (wsdl) version 2.0 part 1: Core language," June 26 2007. http://www.w3.org/TR/wsdl20.

[22] L. Cabral, J. Domingue, S. Galizia, A. Gugliotta, V. Tanasescu, C. Pedrinaci, and B. Norton, "Irs-iii: A broker for semantic web services based applications," in *In proceedings of the 5 th International Semantic Web Conference (ISWC 2006)*, pp. 201–214, 2006.

[23] M. Burstein, J. Hobbs, O. Lassila, D. Mcdermott, S. Mcilraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, K. Sycara, and D. M. (ed.), "Owl-s: Semantic markup for web services." W3C Member Submission, 2004. http://www.w3.org/Submission/OWL-S.

[24] D. Roman, U. Keller, H. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fense, "Web service modeling ontology," *Applied Ontology*, vol. 1, pp. 77–106, November 2005.

[25] D. Fensel, F. Fischer, J. Kopeck, R. Krummenacher, D. Lambert, and T. Vitvar, "Wsmo-lite: Lightweight semantic descriptions for services on the web." W3C Member Submission, 2010. http://www.w3.org/Submission/WSMO-Lite/.

[26] R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M.-T. Schmidt, A. Sheth, and K. Verma, "Web service semantics - wsdl-s." W3C Member Submission, November 2005. http://www.w3.org/Submission/WSDL-S/.

[27] S. Battle, A. Bernstein, H. Boley, B. Grosof, M. Gruninger, R. Hull, M. Kifer, D. Martin, S. McIlraith, D. McGuinness, J. Su, and S. Tabet, "Semantic web services ontology (swso)." W3C Member Submission, 2005. http://www.daml.org/services/swsf/1.0/swso/.

[28] M. Junghans, S. Agarwal, and R. Studer, "Towards practical semantic web service discovery.," in *ESWC (2)'10*, pp. 15–29, 2010.

[29] S. Hagemann, C. Letz, and G. Vossen, "Web service discovery - reality check 2.0," in *Proceedings of the Third International Conference on Next Generation Web Services Practices*, NWESP '07, pp. 113–118, 2007.

[30] C. Legner, "Is there a market for web services?," in *Service-Oriented Computing - ICSOC 2007 Workshops*, pp. 29–42, 2009.

[31] X. D. Alon, X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity search for web services," in *In Proceedings of VLDB'04*, pp. 372–383, 2004.

[32] E. Al-Masri and Q. H. Mahmoud, "Wsce: A crawler engine for large-scale discovery of web services," *Web Services, IEEE International Conference on*, pp. 1104–1111, 2007.

[33] J. D. Garofalakis, Y. Panagis, E. Sakkopoulos, and A. K. Tsakalidis, "Contemporary web service discovery mechanisms," *Journal of Web Engineering*, vol. 5, pp. 265–290, September 2006.

[34] K. Elgazzar, A. E. Hassan, and P. Martin, "Clustering wsdl documents to bootstrap the discovery of web services," in *Proceedings of the 2010 IEEE International Conference on Web Services*, ICWS '10, pp. 147–154, 2010.

[35] L. D. Ngan, M. Kirchberg, and R. Kanagasabai, "Review of semantic web service discovery methods," *IEEE Congress on Services*, pp. 176–177, 2010.