Enabling Mobile Web Service Provisioning
Technical Report 2012-598

Khalid Elgazzar, Patrick Martin, Hossam S. Hassanein
School of Computing, Queen's University, Canada
Kingston, Ontario, K7L 3N6
{elgazzar, martin, hossam}@cs.queensu.ca

***Abstract*—For many years mobile devices were commonly recognized as Web consumers. However, the advancements in mobile device manufacturing, coupled with the astonishing achievements in wireless communication developments are both key enablers for shifting the role of mobile devices from Web service consumers to Web service providers. This paradigm shift is a major step towards the realization of pervasive and ubiquitous computing paradigms. Mobile web service provisioning is the art of hosting and offering Web services from mobile devices, which actively contributes to the direction of Mobile Internet. In this paper, we address the applicability, reliability, and challenges of mobile Web service provisioning. We study the different provisioning architectures along with the enabler technologies for publishing, discovering, and maintaining up-to-date mobile Web services. We point out the major open research issues in each provisioning aspect. Different performance aspects due to the resource constraints of mobile devices are discussed. *Index Terms*—obile Web services, Service provisioning, Mobile devices, Ubiquitous computing, Mobile computingobile Web services, Service provisioning, Mobile devices, Ubiquitous computing, Mobile computingM**

I. Introduction

Hand-held devices and smart phones are traditionally recognized as resource-limited devices. Designers of mobile applications usually take these resource constraints into account to achieve improved performance. While this is true to some extent, the manufacturers of mobile devices have recently achieved breakthroughs in extending mobile devices' capabilities in terms of memories, computational power, storage capacities, and display screens. In addition, many devices such as built-in cameras, infrared ports, bluetooth technology, and a large variety of sensors are embedded within the devices to expand their capabilities and functionalities. It is quite common now to see a single mobile device that can offer navigation information, measure ambient temperatures, control home devices such as TVs and air conditioners, be used as a wireless presentation remote control, and even perform fingerprint secured transactions.

On the other hand, the revolution in wireless communications achieved astonishing developments in increasing transmission rates and improving the spectrum efficiency. Cellular networks are able to accommodate more users and offer a wide range of customized services with different degrees of quality of service (QoS). New services are being offered to mobile users constantly. The evolving 3G and the next generation of wireless mobile network (4G) introduce a flexible and programmable platform to provide users access to future services and applications from a single terminal. Therefore, user expectations are constantly increasing with no limits to their hopes of removing barriers between different network technologies in order to foster flexible and agile mobile applications that are able to fully benefit from the fact that terminals are carried around.

With the advancements in mobile devices' capabilities on one hand and the revolutionary achievements in wireless communications on the other hand, the global interest of mobile applications are significantly increased worldwide. Consequently, researchers and industry are inspired to pave the road for mobile Web services provisioning [1]–[7].

The role of mobile devices as a Web service consumer is fundamental. Shifting the role of mobile devices from Web service clients to providers is feasible only if they can offer standard Web services with acceptable performance and with no impact on the regular use of mobile devices. The mobile devices we refer to in this paper are defined as any handheld device that is ready to connect wirelessly to the Internet and can move freely in the space such as smartphones, PDAs, iPads, etc.

In this paper, we describe the state-of-the-art of mobile Web services provisioning, address its applicability and reliability, point out the research efforts, and explore the challenges and open research problems in the mobile Web services provisioning paradigm.

The rest of this paper is organized as follows. Section II gives a brief background on the Web services approach. Section III discusses the current and potential applications that benefit from mobile services provisioning. Section IV presents different architectures for providing Web services from mobile devices. Section V explores various publishing and discovery techniques of mobile Web services. Section VI discuss the performance of mobile services provisioning. Section VII concludes the paper and outlines future research directions.

II. Web Services

Service-oriented Architecture (SOA) has become a driving force for Web applications development. SOA uses services as the basic constructs to support rapid, low-cost, and easy composition of distributed applications even in heterogeneous environments [8]. In SOA, a service is defined by a Web interface that supports interoperable operations between different software applications using a standard messaging pro-

tocol [9]. In the early nineties, SOA offered the promise of robustness and agility to business enterprises to perform their business efficiently by supporting software reuse, application-to-application interoperability, design flexibility, and a loosely coupled architecture. Web services are the most popular implementation of SOA.

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 [10]. A service, in contrast, is the actual value provided by the service invocation [11]. Web services have a wide scope of applications ranging from simple stock quotes to very complex applications such as Internet banking, weather forecasts, map services. Figure 1 depicts a breakdown of the Web services approach in terms of design style, interface and functionalities description, and type categorization.

A. Web Services Design

Web services design techniques dramatically changed over the course of the past two decades. Web services enable *software as a service* to deliver software services over the network using technologies such as XML. Web services that comply with SOA architecture and use the SOAP protocol to communicate between the client interface and provider are called SOAP-based Web services.

In 2000, Fielding [12] proposed a new architecture style for network-based applications called "*REpresentational State Transfer (REST)*". REST aimed at the generalization of interfaces, scalability of interactions, and independent deployments of software components. Web services built on top of REST principles are called RESTful Web services. The next two subsections shed light on these two architectural approaches with a comprehensive comparison between them.

1) SOAP-based: SOAP-based Web services are designed to allow RPC-like interactions with remote systems. In this design style the service provider and potential consumers need to establish a common understanding of the service syntax and the operations it offers. Each SOAP-based Web service has its own unique interface and is described by means of the Web Services Description Language (WSDL) [13], and that description is published in a public Universal Description Discovery and Integration (UUDI) registry. The UDDI manages and maintains these Web services' entries and keeps a reference for the Web service description file (WSDL document). XML is used to construct the basic blocks of Web service communication by means of some form of XML messaging protocol, such as SOAP (Simple Object Access Protocol) or XML-RPC (XML-Remote Procedure Call).

The basic idea of the SOAP protocol is to enable RPC-like calling for remote software entities using XML, the most widely accepted data format representation [14]. In the interface description of SOAP-based Web services, there is only one endpoint to communicate with, defined by a single URI which points to the service location (address). This address is posted on the UDDI and used by the Web service to receive SOAP messages (requests). The Web service itself is responsible for handling all the communications directed to its internal operations.

Generally, the strength of SOAP as a messaging protocol comes from its ability to work in heterogeneous environments and independently of the underlying platform. For example, SOAP handles the heterogeneity in data types across different platforms using XML Schemas to define some simple XML data types. By knowing these basic XML data types, each system or platform is able to interoperate and map them accordingly to their internal data types. Nevertheless, SOAP has a rigid type checking mechanism, by which SOAP performs most of the standard data verifications. Additionally, SOAP is just a messaging protocol; hence, a message can travel to its destination through the network on top of any transport protocol and is not tied to any particular current or future transport protocol. This adds to SOAP's heterogeneous nature. Consequently, SOAP-based Web services have several years of successful deployment within enterprisers. The SOAP-based approach is heavily promoted by major software vendors who offer fully automated solutions for migrating existing APIs with SOAP code generation.

While SOAP-based Web services have been widely adopted by the industry and supported by almost all development tools, the SOAP-based approach has the following limitations [15]:

- *Complexity*: Deploying a SOAP-based service requires much experience due the complexity of the Web service protocol stack. Only capable developers understand how to deploy a service. Additionally, serializing and deserializing requests written in native languages into SOAP messages is a time-consuming and resource-intensive process, which contradicts the limitations of mobile devices.
- *Accessibility and interface*: The service is exposed to the public using a single endpoint API. Therefore, all the service functionalities and access information are encapsulated within the service description file, hence, all operations use the POST method.
- *Interoperability*: Each Web service has its own service interface. The description information is unique for each service and is exposed by a single WSDL file. Once the client discovers the service, the enclosed binding information in the WSDL file is used to communicate with the service and to construct the requests. Whenever these bindings change, the corresponding communications and requests have to change accordingly.
- *Performance*: Using XML is costly in terms of memory and processing power. Therefore, performance overhead exists in SOAP-based Web service due to the usage of XML and lengthy SOAP messages. Moreover, WSDL file and SOAP messages usually, by design, include redundant information which in turn increases the network traffic and consumes mroe resources.
- *Data Model*: SOAP-based approach hides the data model behind the Web service interface. The interface is not designed to reveal any information about the underlying data model. This feature dictates that the service con-

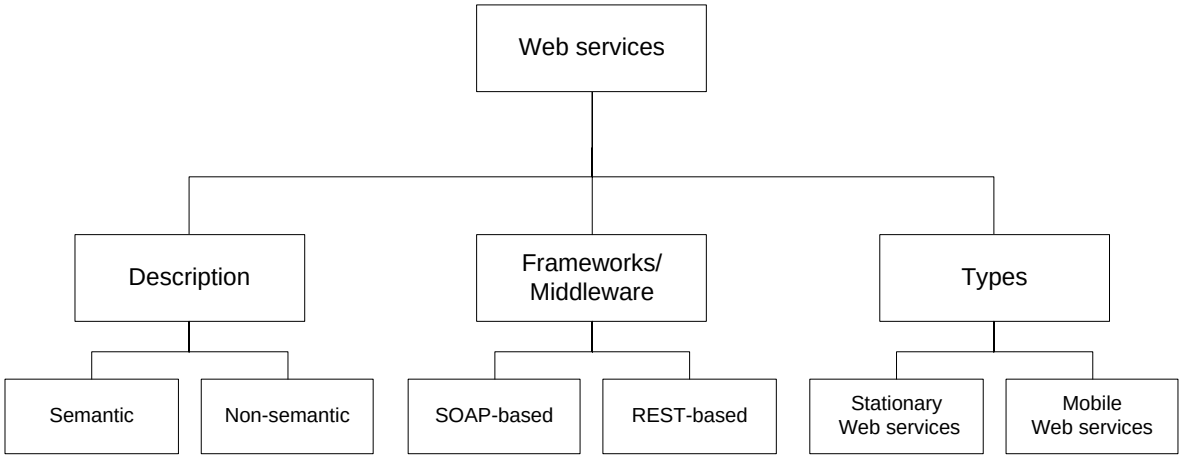

Fig. 1. A general overview of Web service descriptions, design styles, and service types.

sumer and provider have to share a common model to communicate. SOAP advocates argue that keeping the data model away from the clients is safer and less risky.

- *Scalability*: SOAP-based Web services use the Web strictly as a means of transportation for their messages. The messages are interpreted only outside the Web by different applications. Accordingly, the consumer and the provider have to establish a common ground to their communications, hence, scalability is an issue because it fails to achieve the proper integration with the Web as a shared information model.

2) REST-based: In contrast to the SOAP-based approach, REpresentational State Transfer (REST) [12], also called Resource Oriented Architecture (ROA), is a style of software architecture that relies on the fact that any resource (such as Web services) can be identified by their URLs. In his dissertation, Fielding defines the REST approach as follows [12]: *"Representational State Transfer is intended to evoke an image of how a well-designed Web application behaves: a network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use"*.

RESTful Web services [15] are Web services that conform with the concepts of REST and avoid the performance degradation resulting from the use of SOAP and XML. They are tightly coupled with the HTTP protocol however and compromise their flexibility and portability. RESTful Web services gained much attention from the Web community due to their simplicity and scalability. Major Web services providers such as Google, Amazon, Yahoo, and eBay adopted the RESTful Web services approach in their offered Web services. Since 2000 when Fielding proposed the REST architecture principles for the first time, the global interest in RESTful Web services have been constantly increasing.

The RESTful approach features the following advantages over the SOAP-based approach that make it more desirable and widely adopted [15].

- *Scalability*: The RESTful approach inherits its ability to scale from the underlying scalability of HTTP.
- *Addressability*: Resources (services) are easily exposed and accessed through a valid URI rather than requiring a centralized repository such as UDDI to manage publishing and discovery. Each resource has its own unique URI, which can be fetched while the user navigates though the link connections between resources.
- *Links and Connections*: Borrowing the notion of hyperlinks, resources can link to each other using hyperlinks. More importantly, state transfer can be managed through the referral to links.
- *Stateless*: Requests in the RESTful approach are self-contained. This independence allows the ability to delete the related information to a request once it is done. REST principles dictate that HTTP messages should be "self-descriptive", which implies that any intermediary node can fully interpret messages, understand it, and take actions upon its content on behalf of the user.
- *Unified Interface*: All resources are dynamically handled by a limited set of standard HTTP methods, namely, GET, PUT, DELETE, and POST. Any HTTP client can communicate directly with any HTTP server without any further special configuration. In contrast, SOAP needs both client and server (or consumer and provider) to agree and be aware of method names, data types, and addressing model. The main reason for this is because SOAP is a protocol framework, whereas HTTP is an application protocol [10].

Despite the aforementioned features and advantages for the RESTful approach, there are some valuable lessons that SOAP can teach RESTful to add beyond what HTTP can contribute [10].

- *Security*: Data that needs to be secure can not be sent inline with the URI. HTTP GET encapsulates data parameters in the request, which risks the data and becomes itself a security threat. SOAP is the better solution when it comes to wrapping a large amount of data. Though HTTP has security features, adopting the WS-Security model (supported by SOAP) would strengthen the RESTful approach.
- *Routing*: Routing HTTP messages between different players is controlled by the underlying network. This means in cases where control over routing is required to determine a path between the client and the provider, HTTP is not the best solution. For example, SOAP messages have the ability to allow the headers to be directed to a particular intermediary (i.e. a proxy or cache)
- *Asynchronous Execution*: It does not make sense to have the client wait for a lengthy execution to complete. There should be a way, such as a call back, for either the client or the server to re-establish the communication channel whenever the result is available. SOAP has the ability to perform either synchronous or asynchronous execution.
- *Service Level Agreement*: Usually, SOAP-based services have a contract between the service provider and consumer. This contract includes terms and conditions, guaranteed QoS, reliability and availability, payments, etc. It also specifies how to handle conflict between providers and the consumers whenever terms or conditions are violated.

Table I gives a summary of our comparison between SOAP-based Web services and REST-based Web services design style.

B. Web Services Description

The interactions of a Web service usually involve three parties: a service provider, a service consumer, and occasionally a service broker. Once a Web service is developed, the provider has to define the specification of how to perform service requests and describe the Web service functionalities and how potential consumers can access and invoke the required functionalities. Generally speaking, Web services can be described using either *semantic* or *non-semantic*approaches.

1) Non-semantic Description: In the non-semantic approach, Web services are described by the Web Service Description Language (WSDL). WSDL is an XML-based language that provides a model for describing non-semantic Web services [13]. WSDL documents enable service providers to describe their services and explain to potential customers how to consume the services' functionalities [13]. WSDL 2.0 is the latest WSDL standard specification [13]. 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.

A Web service description usually involves different aspects such as information model, functional capabilities, nonfunctional parameters, and technical specifications. The information model defines the data model comprising input/output messages and other data relevant to the service operation. Functional capabilities determine the operations offered by the service and how potential customers can interact with the service. Nonfunctional parameters specify both the environmental and running parameters such as QoS, reliability, availability, etc. Technical specifications are mainly concerned with implementation details such as message structures, transport protocols, service location, and access information. The non-semantic description approach describes Web services at a *syntactic* level [16]. According to the previously mentioned description aspects, non-semantic approach describes the information model using XML schema, while a WSDL interface describes the functional capabilities. The nonfunctional parameters are determined by a means of WS-specifications in terms of policies and agreements. The technical details are defined through service bindings and endpoints information.

A WSDL 1.1/WSDL 2.0 document describes a Web service using six major components:

- `<types>/<types>` element is an XML data type definition that describes the data containers used in message exchanges. The element name did not change in WSDL 2.0.
- `<message>/NA` element is an abstract representation of the transmitted information. Typically, a message contains one or more logical parts (parameters). These parts are associated with a type definition. In the skeleton of WSDL 2.0 the `message` element is removed as a global element and the description of messages is encapsulated in the `interface` element.
- `<port>/<endpoint>` The port/endpoint defines the access point of the Web service.
- `<portType>/<Interface>` is an important component in WSDL documents, in which a set of abstract operations (functions) that can be performed by the Web service are defined. Each operation is associated with an input and/or output message.
- `<binding>/<binding>` component specifies the communication protocol and data format for each operation and message defined in a particular portType/interface element.
- `<service>/<service>` element is a composite operation that aggregates multiple related ports or functions.

WSDL 2.0 specifications enable the integration of the REST approach and Web services through the introduction of HTTP binding specifications. For each operation provided by the service description, some HTTP parameters (if applicable)can be defined such as URI, HTTP method, input/output data serialization, etc. The main objective of providing such a specification extension in WSDL 2.0 is to enable services with both SOAP and HTTP bindings.

2) Semantic Description: Describing Web services semantically relies on *ontologies* [17]. An *ontology* is a formal

TABLE I
A SUMMARY COMPARISON OF WEB SERVICES DESIGN APPROACHES

Feature	**SOAP-based**	**REST-based**
Architecture Style	Service-centric	Resource-centric
Coupling	Tightly coupled	Loosely coupled
Transport Protocol	Any	HTTP only
Access Scheme	Single end-point	URI for each resource
QoS	WS specifications	Transport-dependable (HTTP)
Invocation	RPC-like	HTTP methods
Interface	Interface for each Web service	Web browser
Description	WSDL	WSDL 2.0
Data Model	Hidden	Exposed
Data Representation	XML	XML
Scalability	None	Connected hyperlinks
Security	WS-security-based	HTTP-based

explicit specification of a shared conceptualization [18]. From this conceptual definition the essential components are extracted which constitute the individual ontologies; they define an agreed common terminology by providing concepts, and relationships between the concepts [11]. Ontologies are structured in a class hierarchy; each class represents a property or a function.

Semantic descriptions of web services aim to provide unambiguous definitions to the description terms and to address the lack of understanding of the semantic meaning of messages and data, which in turns makes the interactions between services more logical and facilitate composition and integration of Web services. In contrast to the non-semantic approach, semantic description can incorporate non-functional specifications such as those requirements that can be observed at the runtime such as availability, reliability, and security, or those that can be realized at the design time such as extensibility and scalability. Semantic Web services are anticipated to contribute in the transformation of the Web from information-based to knowledge-based services.

In the semantic approach, services are described by profiles, models, and groundings. The service profile contains the information related to the service functionalities, which is needed by the service requester to match the service with the required task. The service model describes the service implementations, required inputs, and expected outputs. It also can be used by the requester to refine the search results. The service information model is defined through domain ontologies. Functional details are represented by capabilities and functional categories whereas non-functional parameters are described using ontologies that describe different non-functional properties. Technical issues such as bindings and protocols are defined the same way as in WSDL documents [16]. Service grounding defines the service accessibility. More precisely, the service is advertised, registered, and discovered through the service profile. Once the service is located, the requester uses the service model and grounding together in order to access it [19, 20].

Web services may use various semantic description languages such as Web ontology languages (OWL-S) [21],Web Service Modeling Ontology (WSMO) [11], WSMO-Lite [16], Web Services Semantics (WSDL-S) [22], and Semantic Web Services Ontology (SWSO) [11, 23]. Even with the standardization efforts, each description language has its own notation and no universally accepted formal notations yet exist for semantic descriptions. Services that are described in a particular semantic description language would only be discovered by requests constructed by the same semantic formalism.

With the enormous offerings of Web services, searching for the right service/s for a particular objective is a key challenge. The syntactic level of description, offered by non-sematic description, does not enable the automation of service discovery and integration. Semantic annotations augment the capabilities of service description and make it machine consumable based on meaning and understanding not just the syntax.

Table II summarizes the differences between non-semantic and semantic description approaches, emphasizing the advantages and disadvantages of each approach.

C. Types of Web Services

We classify Web services based on the type of host. Web services that are provided by fixed servers and consumed by stationary clients are called *stationary Web services*, whereas services that are hosted and provided or consumed by mobile devices are called *mobile Web services*. Our focus in this paper is on mobile Web services and how applicable it is to provide them by resource-constrained mobile devices. However, the next section explains and elaborates more on both of these two types of Web services.

1) Stationary Web Services: Stationary Web services typically are location-dependant as they basically designed for static networks and deployed on fixed servers. Usually stationary Web services are tied to the availability of local resources such as databases hosted on the same server [24]. These Web services can be accessed and consumed by fixed or mobile clients through the announced address for the Web service. Since they are that it is hosted on fixed servers, stationary Web services normally are capable of providing reliable performance and guarantee reasonable QoS. Computational resources can be added or shrunk according to the service demands. Therefore, stationary Web services can serve, theoretically, a huge number of users. Stationary Web services can be replicated on multiple servers to support distributed

TABLE II
A COMPREHENSIVE COMPARISON BETWEEN SEMANTIC AND NON-SEMANTIC WEB SERVICE DESCRIPTION APPROACHES

Feature	**Non-semantic**	**Semantic**
Information model description	XML-Schemas	Domain ontologies
Functional descriptions	WSDL interface	Capabilities and functional categories
Nonfunctional descriptions	NA	Ontologies (policies and properties)
Behavioral descriptions	NA	pre & post-conditions
Technical descriptions	WSDL bindings and communication protocols	Same as in WSDL
Search	Keyword based	Semantic reasoning

provisioning and/or avoid a single point of failure and enhance the service reliability and availability. Service replication could be a static or dynamic decision or, in other words, at design time or at runtime.

The general Web service communication schemes are independent of the nature of service consumers and service providers, which means that the communication strategies do not care whether one or both of the communication parties (consumer and providers)are fixed or mobile nodes [25]. Web services are typically, designed and developed to behave synchronously, i.e users are blocked while executing even if it is a lengthy execution until a response comes back or the connection becomes timed out. In mobile domains, the synchronous execution of long-lived processes is not the right choice. Regular usage of mobile terminals has to be maintained during the execution of Web services whether the mobile terminal is a service consumer or provider.

2) Mobile Web Services: Mobile Web services are Web services that can be delivered from mobile devices [1]. This paradigm emerged from the successful coupling of the flexibility offered by Web services and the convenience of mobile devices. Mobile Web services are supported by advancements in wireless communication technologies and benefit from the swiftly expanding mobile customer base. In mobile services, mobile devices can play the role of service client, service broker, and/or service provider.

As in traditional Web services, the interactions in mobile Web services encompass three main parties, service provider, service registry, and service client. Occasionally a service proxy may exist to facilitate the communication process between the provider and the client. The interactions in mobile services differ from those with stationary services due to the following characteristics.

- **Limited Resources**: Mobile devices are resource-constrained. Although, there has been a revolutionary advancement in the capabilities of mobile devices in terms of processing power, memory space, battery lifetime, and embedded sensors, they are not capable of running desktop applications as efficiently as desktop machines. Additionally, the major constraint for mobile devices is their limited display screens which can relatively display little data at a time, resulting in compromising the usability of applications. If mobile devices act as service providers, these resource concerns become dominant. For example, mobile providers cannot support too many service consumers concurrently due to the limited computation power and cannot claim or guarantee a particular QoS due to limited bandwidth and the intermittent nature of wireless connections.
- **Dynamic Environments**: Mobile devices are always on the move and sometimes change the network operator or even handover between different network technologies within the same network. This results in challenges for providing Web services in a highly dynamic mobile wireless environments. Services could be unreachable because of frequent link failures. Mobile providers may change location, IP address, point of attachments to the network, which make services' binding information invalid if not properly updated. Moreover, discovering the right valid Web service in such highly dynamic environments is a crucial issue.
- **Resource Heterogeneity**: The operating systems and software platforms on mobile devices span multiple vendors and feature a wide range of characteristics and supporting functionalities. Mobile providers are can make use of the local context, functionalities supported by different platforms, and runtime environment capabilities provided by different operating systems to enhance their services provisioning.
- **Context-Aware Service**: Traditional Web services are designed for fixed networks. Most of the concepts and developed mechanisms throughout the Web service interactions are not personalized or make use of the user's context. The context can include the user location, device storage, display screen, network bandwidth, transmission rate, etc. Now, because of the emergence of mobile Web services, these vital parameters can be exploited and counted on for efficient delivery of adaptable and personalized Web services.
- **Uninterrupted Voice Service**: During the provisioning of Web services from mobile devices, the regular voice functionalities should not be interrupted or affected by services provisioning. Therefore, a trade off should be made clear such that neither the service performance nor the regular voice usage of mobile devices is seriously affected.

Due to the aforementioned mobile wireless characteristics, many conventional Web service protocols and mechanisms have been adapted as well as new supporting platforms have been developed suitable for the deployments on mobile wire-

less domains. We briefly highlight a non-exclusive list of these platforms as follows.

- **Personal Java**: Also known as "pJava" [26], it is a lightweight Java programming environment with a smaller memory footprint aimed to support developing applications for resource-limited mobile devices. The strategy of pJava is to use a reduced set of class libraries with some substantial optimizations to provide acceptable performance of Web applications on mobile devices.
- **Java ME**: Java platform, Micro Edition (J2ME) [27] is the successor of PJava and is the most ubiquitous Java application platform for mobile devices. A broad range of embedded devices use J2ME as the Java runtime platform. J2ME comes pre-installed on most of the current smartphones. Currently, J2ME comes in one of two platforms. The first one is *Connected Device Configuration (CDC)* to support high end mobile devices such as iPADs,PDAs, and some powerful smartphones with the base set of APIs and virtual machine. The other one is *Connected Limited Device Configuration (CLDC)* to provide the same kind of support but for mobile devices that experience intermittent wireless connections and have relatively limited resources such as smartphones. On top of CLDC and CDC, *Mobile Information Device Profile (MIDP)* provides the Java runtime environment for Java mobile applications on most of today's mobile devices.
- **kXML2**: kXML2 [28] is a lightweight XML parser designed specifically for constrained environments.
- **kSOAP2**: kSOAP2 [29] is a lightweight SOAP implementation adapted for resource-constrained devices to overcome the significant overhead of the original SOAP implementation. kSOAP2 is an open source SOAP web service client library that processes SOAP messages based on kXML2 parser. The basic functionality of kSOAP2 is to convert the data types in SOAP messages into Java data objects. gSOAP is the kSOAP equivalent for C and C++ with some additional capabilities for creating Web services stubs from WSDL.

For the remainder of this paper we use the terms "*mobile Web services provisioning*" and "*mobile hosting*" interchangeably. In the next section, we present the current and potential mobile applications and domains that can benefit from the mobile hosting computing paradigm.

III. Mobile Hosting Applications

The Mobile Web services paradigm finds its way into real life applications through the huge demands and interest from the mobile users' community. A wide range of applications in domains such as social systems, collaborative learning, real-time healthcare systems, emergency services, and communications are anticipated for this paradigm due to the robustness and flexibility it offers. Next are some application examples and corresponding domains, wherein mobile hosting could be of interest.

- **Multimedia Sharing**: Mobile devices nowadays are aided with a wide range of embedded devices and sensors. Two of the important embedded devices that a smart phone or mobile device might have are cameras and a Global Positioning System (GPS) device. Each one of these devices (or combined together) may have its own interesting applications. For example, the terminal location information via GPS data can be used to provide assistance for people making a distress call in case of emergencies. By knowing the exact location, supported by pictures of the person in need this would greatly help rescuers to come well prepared and provide the necessary assistance accordingly. On the other hand, travelers might be interested to share photos or videos of places they have visited. These photos and/or videos could be offered to interested people free or with a fee. With the help of mobile Web services provisioning, owners of these multimedia data services can host them on their mobile devices without the need to rent a host or use a third party Web service and pay for the hosting and consuming. In such cases, mobile Web services provisioning enables payment shifting to the actual consumer. Interestingly, a traveler can make use of the GPS capability and make the offered media even more interesting by mapping photos and videos to the actual locations to which they belong.
- **Location-based applications**: Most users carry their mobile phones with them for the majority of the time. While users move, they may offer access (with various privileges) to real-time context, such as location, air pollution levels, luminosity, and noise levels. Location-based applications are expected to benefit the most from mobile Web services provisioning.
- **M-learning**: In learning media sharing scenarios, learners can share videos, audio, documents, comments, make illustrative figures, etc. In this regard, mobile Web services provisioning has been well demonstrated in *Collaborative M-Learning* [1]. In this scenario, learners can manage their own learning profile including their advancements and expertise and make it accessible for others. They can search for experts in particular areas and seek their advice, learners may be able to cache this information to forward it later to other interested learners. Learners can retrieve literature resources from other learners and manage their own database of articles. They can write or receive comments on certain topics
- **Collaborative Journalism**: Another interesting commercial application envisioned by Srirama in his PhD thesis [1] is collaborative journalism. Editors can coordinate between their journalists and have immediate access to their contents. Journalists themselves can collaborate and synchronize with each other to better cover distributed events. Organizations can make use of this paradigm to track their employees and have them connected at all times to the organization's facilities, databases, and information.
- **Personal information**: Personal profile and contact infor-

mation, personal photo/video sharing, and schedules are amongst the applications that can potentially benefit from the paradigm of mobile Web services. Skulason [2] in his master's thesis demonstrated the effectiveness of mobile Web services provisioning through a demo application called MobCal [1]. The application extends the default calendar of mobile devices. Users' entries are retrieved from their mobile devices into a public website. With a username and password, users can access and manage their calendar entries. Furthermore, users can synchronize their mobile calendar with their Google calendar. Entries that are marked as *public* will be made available on the website, whereas *private* entries will be kept private. Visitors to the website can request a meeting with a particular authorized user, and if request approved, the user's calendar will be automatically updated.

- **Healthcare Systems**: In healthcare systems mobile hosting can be of great help as mobile devices, using any short range wireless communication technology (such as Bluetooth or ZigBee), may collect patients' data and vital parameters from sensors attached to the body and send them to healthcare centers for the purpose of monitoring. Once any of these vital parameters seriously changes, an emergency alert can be issued and medical assistance may be provided promptly and efficiently.
- **Personal social networking**: A clique of friends may dynamically form a private social network while keeping all their personal information, social status, posts, and updates on their mobile devices [30, 31]. Mobile services would enable users to maintain their social profile and allow friends to access their social information without sharing data with a third party.

IV. Mobile Hosting Architectures

Over the past couple of years, several researchers have proposed different architectures and frameworks for providing Web services by mobile devices. These studies center around the possibility of hosting Web services on resource-limited devices. Each one of these approaches addresses and deals with certain challenges facing mobile services provisioning such as reachability, reliability, and scalability.

Most of the current mobile Web services provisioning architectures are immature and still in the early stages [32]. Pawer et al. [33] discuss briefly different mobile services provisioning approaches and classify them into three categories, namely, proxy-based, Peer-to-Peer, and asymmetric infrastructure. In this section, we discuss in details the current available architectures, illustrating the merits and shortcomings of each one, and comparing them side-by-side.

A. Proxy-based Architecture

The proxy-based mobile services provisioning architecture is the easiest approach for avoiding many challenges facing the implementation of providing Web services from resource-constrained devices such as traditional protocol compatibility and scalability. While at the same time it takes advantage of being hosted on mobile devices to get an instant access to the device resources (such as embedded sensors) or realtime context information (such as location or any other environment parameters). The proxy is usually a high end machine attached to the fixed networks. Therefore, it theoretically has unlimited bandwidth to minimize the bandwidth usage in mobile networks and enough processing power to off-load the resource-constraint devices and perform the heavy load processes. The proxy-based approach relies heavily on Jini technology [34]. Jini is an infrastructure based on Java to enable building federated network services. The infrastructure is comprised of a join/discovery protocol and lookup service. The lookup service is the major component of the system which serves as a repository of services, whereas the join/discovery protocol publishes and discovers network services.

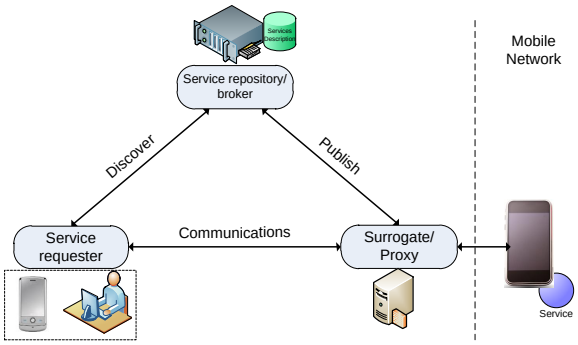

Fig. 2. An overview of proxy-based mobile Web services Provisioning.

Figure 2 shows an overview of the proxy-based approach. The architecture consists of a mobile device hosting Web services and is connected wirelessly to a high-end machine acting as a proxy. The proxy represents the endpoint of Web services to the clients. Web services are published by the proxy to the look-up directory/registry which represents a service broker for both providers and clients. Potential clients discover the requested services through the lookup directory, get the binding information, and contact the proxy directly to use the corresponding service. The intercommunication between the mobile service provider and the proxy server ensures that the provided service is up-to-date.

Relying on a proxy in mobile Web service architectures resolves many challenges. For example, the proxy is more capable than resource-limited mobile devices to serve a large number of clients simultaneously with acceptable performance and response time. Additionally, one of the most common tasks for proxies is to deal with different protocol translations, since most of Web service protocols are not originally designed for wireless communications and are not optimized for low-rate transmissions, high error rates, higher latency, and intermittent connections. Moreover, a proxy-based architecture can guarantee a reasonable QoS in contrast to a fully wireless

[1] http://mobilews.mobcal.com

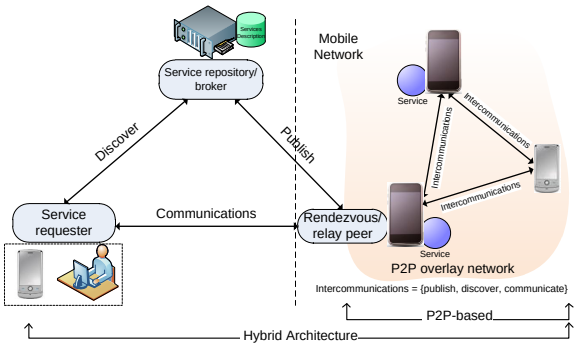

Fig. 3. P2P-based architecture of mobile Web services provisioning.

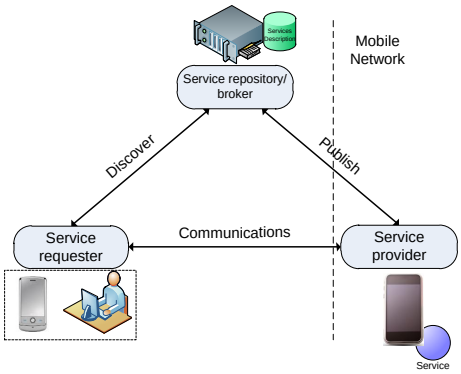

Fig. 4. Overview of asymmetric mobile services provisioning.

domain wherein providing a particular QoS is quite difficult. Proxies can also hide the heterogeneity of various mobile devices and support mobile terminals with disconnected states [35].

B. P2P-based Architecture

P2P technology is a distributed, low-cost, and collaborative computing paradigm. Over the past two decades, Peer-to-Peer (P2P) technology has gained much popularity and its applications acquired the interest of the research community. P2P applications are classified into three main categories, namely, content management applications, parallel executions over anonymous peers, and collaborative P2P applications. The most famous implementations for these three types are Napster, SETI@Home, and Skype respectively. It is quite interesting to combine the two technologies, P2P and mobile Web services and take advantage of features from both paradigms.

The vision of the P2P mobile Web services provisioning architecture is to overcome the limitations of the centralized approach of UDDI based registries and to take advantage of the flexibility of the P2P network approach. Figure 3 illustrates the basic architecture of providing Web services from mobile devices over P2P networks. Figure 3 shows also how P2P-based, proxy-based, and asymmetric architectures may coexist to form a hybrid one. In a P2P-based approach, mobile devices act as a Web service provider, whereas any Web client can consume these services whether it is a mobile or a stationary client. P2P provisioning relies on the P2P network advertising mechanism to publish and discover Web services. The P2P network advertising mechanism handles node mobility and dynamically manages the location and binding information of the Web service in WSDL documents through the lifetime concept. P2P advertisements are associated with a lifetime parameter, whenever the lifetime expires the advertisement has to be re-published to stay valid, otherwise, it is removed automatically or marked invalid. This mechanism of managing the publication of services eliminates the task of keeping centralized services registries consistent and up-to-date.

Publication and discovery of Web services in the P2P architecture are handled using JXTA [36, 37]. JXTA advertises services as modules, where a module is composed of a module class, a module specification, and a module implementation [36]. This corresponds to WSDL and UDDI in the traditional Web services approach. To discover Web services, Web clients have to join the P2P network and get a *PeerID*. The mapping between the client IP and the PeerID is managed by the underlying JXTA protocols. After the clients get successfully connected to the JXTA network, they can query/search the network for the required services. Once the required service is located, the client communicates with the service provider through JXTA pipes in order to send and receive messages.

C. Asymmetric Architecture

The architecture of the asymmetric mobile services provisioning approach follows the traditional Web services architecture as shown in Figure 4 except that the service provider is a resource-limited mobile device. The term asymmetric was coined by Porta [35] for those approaches that address the resource limitations of mobile devices. Figure 4 shows the architecture of asymmetric mobile services provisioning. To make the performance of this architecture acceptable by the Web services consumers and the services community, many adaptations have been proposed for traditional Web service protocols and mechanisms to cope with the limitations of mobile devices. For example, asymmetric architecture supports only simple XML data types (such as *String*, *Integer*, *Char*) to avoid the complexity of complex type extraction [33]. Another example is the utilization of the *Asynchronous Service Access Protocol (ASAP)* [38] in the communication of mobile services. ASAP is specifically designed by OASIS to target the service interactions in long-lived mobile services. ASAP enables services to run asynchronously and independently from their caller service. In such scenarios, the client invokes the service and waits for the response without blocking the client during the execution. The response is send back whenever it is available or a later separate request can be made by the client to communicate the results of the Web service operations. ASAP implements asynchronous interaction techniques such as ”Callback” and ”Polling”, which mean that the server sends the response to the client whenever it is ready or the client has

to check back for the results respectively. However, ASAP enables the client to query the Web service for its current status during the execution time. Furthermore, due to the long-term execution of the asynchronous services, the requirement of the client may change during the course of the service execution. ASAP enables the client to send an update or modification message to change the previous information. The asynchronous Web service should have the ability to update itself accordingly at runtime.

As a proof-of-concept, Aijaz et al. [25] purpose asynchronous mobile Web services middleware that supports the asynchronous execution of long-lived services. The framework supports both "Callback" and "Polling" service interaction techniques. Due to the dynamic nature of mobile environments, Kim and Lee [4] propose a lightweight framework that hosts Web services on mobile devices and supports service migration. Their framework is composed of six modules, namely, request and response manager, publication and discovery manager, directory manager, migration manager, and context information manager. The framework aims to achieve a seamless Web services provisioning by mobile devices in wireless ad-hoc networks. The migration and context managers ensure that the service is uninterrupted and reliable through the migration to a new suitable host whenever needed. Service migration can be triggered upon a request from the provider or due to a change in the provider's context that makes the service unavailable, such as CPU load or battery outage. The framework is described generically without many details about individual components or how they are actually deployed.

Table III provides a comparison between different mobile services provisioning approaches. While the proxy-based and P2P approaches rely on other technologies for setup such as Jini and JXTA, the asymmetric approach needs no extra infrastructure for running.

D. Open Research Issues

The distinct features of mobile Web services provisioning and the constraints of mobile devices introduce a number of critical design issues for architectures for this paradigm. Despite the various proposed architectures, no perfect solution has been identified that exploits the pervasiveness of mobile devices to provide ubiquitous services. The simplicity and scalability of the proxy-based architecture compromise its portability and consistency, yet impose limitations on the ubiquity of provided services.

The pervasive nature and agility of the P2P approach sacrifice its performance. The performance of the asymmetric approach suffers due to the resource limitations on mobile devices as well as the possibility of intermittent connectivity. The realization of mobile Web services provisioning is still far from being achieved and open research problems requiring further investigation include the following:

- **Architecture**: Current mobile Web services provisioning architectures are basically adapted from traditional Web services approaches. Adaptations often lead to inefficient solutions for different environments. Rethinking the architecture could lead to better solutions for mobile Web services provisioning; taking advantage of pervasive mobile devices to fulfill ubiquity requirements. More agile and robust Web service architectures that limit the overhead on resource-limited providers and that meet the requirement of pervasive environments are needed.
- **Frameworks**: SOAP/WSDL are the defacto standards used by Web services. However, this framework is not suitable for resource-limited environments. Optimizations made for pervasive and mobile-related constraints in this environment are at the expense of other issues that may substantially impact the performance of services. Services must be available in an "anywhere, anytime" fashion, irrespective of specific platform or device form factor. In this regard, a generic and adaptable framework must be developed that can be implemented efficiently on every platform.
- **Performance & Monitoring**: A successful architecture/framework for mobile service provisioning should maintain the regular usage of mobile devices for voice service as well as delivering ubiquitous Web service with performance comparable with their counterparts on fixed networks. Performance is a crucial requirement of mobile services delivery, especially if the user is running some important mobile applications that need the devices resources (such as memory and processing power) to function properly. To achieve reasonable performance without seriously affecting the normal functionality of mobile devices, mobile providers must process a reasonable number of service requests. Thus, a lightweight monitoring mechanism is required.
- **Prototypes**: Proof-of-concept applications and use case scenarios, especially in pervasive environments, are at the core of pushing forward the adoption of mobile services provisioning approach.
- **Context-awareness**: One of the very distinct features of Web services provisioning from mobile devices is the use of available context to personalize services. Context information such as mobile capabilities, available add-on devices, location and user profiles can augment mobile services provisioning in terms of publishing, discovery, and usage to achieve highly ubiquitous services.
- **Data Formats**: Though XML is very portable, XML messages are very verbose. XML parsing is a heavy weight process that is resource (CPU and memory) intensive. Less complex data formats should be considered for use in the mobile domain.
- **Toolkits**: Mobile devices have limited resources such as small screens and limited input capabilities. Web service toolkits are required to facilitate the development and deployment of mobile web services, and leverage the usage of these services through automatic user interface creation, scalable discovery, etc.
- **Asynchronous Execution Support**: Pervasive mobile domains are characterized by limited bandwidth and intermittent wireless connectivity. Offline mode may be

TABLE III
COMPARISON SUMMARY BETWEEN DIFFERENT MOBILE WEB SERVICES PROVISIONING ARCHITECTURES

Feature	**Proxy-based**	**P2P-based**	**Asymmetric**
Architecture style	Decentralized	Distributed	Centralized
Core technology	Jini architecture	JXTA protocols	Traditional Web services architecture
Communications	Through the proxy	Peer talk to Peer	Clients talk directly to providers
Addressing	Announce the proxy address	Unique PeerID	IP-based
Service publishing	Jini Join request	JXTA advertisements	UDDI
Service discovery	lookup service discovery (UDDI-like)	JXTA resource discovery	Query the UDDI
Service invocation	Access the proxy + RMI	Communicate the provider peer + HTTP	Access the provider + HTTP
Scalability	Can serve large number of concurrent customers	Scale as peers join	Limited number of concurrent customers
Consistency	Synchronization between the proxy and the mobile provider	Advertisements associated with lifetime	Consistent
QoS	Guaranteed	Unguaranteed	Unguaranteed

required because of these limitations or to unblock user terminals while executing lengthy Web service processes. Clients of mobile Web services may lose the connection while they are moving or may voluntarily choose to disconnect while the service continues execution. Asynchronous Service Access Protocol [38] is one step to achieve this objective, however, more research is required in this direction.

- **User Feedback**: Mobile users are typically concerned about quality of service and completeness of coverage for their voice service. When it comes to data services they have different requirements such as simplified data entry, efficient utilization of network resources, asynchronous operations and service personalization. Therefore, improving user experience is critical to the ultimate success of data service providers.

V. WEB SERVICES PUBLISHING AND DISCOVERY

Publishing a Web service is the process of notifying Web user of that Web service and providing all the information required to access it. Publishing is a simple process where providers typically have two options; either registering their services with a service broker in a public service repository, using the UDDI standard, or publishing the services in a local service directory and advertise these services. Service brokers usually provide a Web interface for their service registry that accepts information about providers, their service technical interface (tModel), and description files. In contrast, Web service discovery is the process of finding a Web service that fulfils a certain task. Service discovery is a crucial component of any service centric system. The possibility of such system failure is a 100% if the discovery process fails to find the correct service.

Most of the existing discovery techniques belong to one of three main categories [39]: UDDI Business Registry (UBR), specialized search engines and generic search engines. Each one of these approaches has advantages and disadvantages as follows:

UDDI Business Registry: UBR is the discovery approach used by the standard Web service architecture. It relies on centralized repositories which 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, tModel, implementation, etc. This approach has not been widely adopted by the Web services community, which explains why major UBRs shut down their services in 2006 [40]. However, there are still a few public registries offering their services with different capabilities such as RemoteMethods, StrikIron, and X-Methods. Problems with UBRs include their centralized architecture, limited scalability, single point of failure, consistency maintenance, keyword search or category browsing only, and outdated service records. However, UBRs enable service subscription for interested users to keep them updated. UBRs provide extra features such as service trail, transaction facilitation, WSDL parser, different pricing schemes, performance monitoring, programmatic interface, and ratings.

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 service 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 service 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 service capabilities. Woogle [41] and WSCE [42] are examples of these search engines. Web service search engines are able to find services that are more relevant to users' requests; as the search does not solely rely on keywords, but also on functionalities and other running parameters such as QoS. Additionally, the retrieved services should all 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. User ratings and feedback can also be taken into account in ranking the retrieved list of services. However, so far this approach only supports searching for non-semantic Web services.

Generic Web search engines: Web content search engines

are another alternative to finding 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, such as Google, to locate Web services by customizing the search query to look for specific files types (ex. wsdl and owl files). The major drawback of 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 using generic search engines include, robustness, scalability; and no extra infrastructure is required.

Service discovery should demand minimal user involvement, especially in mobile domains where users have limited input capabilities. A comparison between current discovery mechanisms focusing on the autonomic capability of service discovery is presented in [43]. The comparison is carried out based on eight criteria that evaluate how autonomous these approaches are. These criteria are service description, matchmaking/reasoning, scalability, robustness, service composition, Quality and cost of service, up-to-dateness , and service replacement.

As mentioned earlier, Web services can be described semantically or non-semantically (WSDL-based). The discovery process is quite different according to the Web service description method. Semantic Web services are discovered by high level match-making approaches [5], whereas non-semantic Web services discovery use information retrieval techniques [38].

A. Non-semantic Discovery

WSDL documents are used to describe the functionalities and binding information of the Web services they describe. WSDL documents are usually accessible by standard protocols such as HTTP and SOAP. In the WSDL-based service discovery approach, a web service discovery engine partially matches the search terms entered by the user with the Web service name, location, business, or tModel [44] defined in the service description file. The use of these types of keywords is, by design, limited in WDSL specifications. A relevant service may not be retrieved if the search terms do not include part of the Web service name. A user may even miss services that use synonyms or variations of these keywords. For example, a service that contains *"car"* in its name may not be retrieved by a query looking for *"vehicle"* service. A solution to this problem is proposed by Elgazzar et al. [45] via clustering WSDL documents based on functional similarity.

The WSDL-based service discovery approach works on the syntactic level and lacks the understanding of the semantics of Web service functionalities. Thereby, building a common ground between the provider and the consumer is hardly achievable, especially in pervasive environments [46]. In these environments, service discovery should be robust and lightweight enough to cope with the network's dynamics and resource-limited mobile devices. Furthermore, important parameters such as QoS, user context, and other non-functional parameters cannot be exploited in discovering the most appropriate Web service to the requested task using WSDL-based approach. The semantic approach introduces solutions to these discovery issues using semantic reasoning.

B. Semantic Discovery

Semantic Web service discovery relies on domain *ontologies* to discover relevant Web services for user requests. The foundation of a semantic discovery method is the semantic description formalism used to describe Web services. The most prominent semantic formalisms adopted, to date, are OWL, WSMO, and WSDL-S [47]. Amigo-s [46] is a semantic description language developed to advertise and discover Web services in pervasive and resource-constrained computing environments using a dedicated Service Discovery Protocol (SDP). A number of optimizations were considered such as, offline classification for offered ontologies, hierarchical categorization for requested capabilities, and distributed service directories. However in the case of resource-constrained mobile providers/brokers, the overhead of service classification could be infeasible; especially if the set of available services is constantly changing as providers/brokers move around. A common problem for all current semantic approaches is that they must apply the same formalism to describe the service capabilities and the service request (a solution is introduced to tackle this problem in [48]). Then, a matchmaking process is performed to match the request requirements with the offered capabilities.

The matchmaking process comprises three steps, namely: (1) parsing both the user request and service advertisement profile for requirement and capabilities, respectively; (2) using a semantic reasoner to load the ontologies used by the user request and service advertisement, and (3) finding the semantic relation between inputs, outputs, and properties of the requested and provided functionalities and capabilities. Step 2 and 3 are carried out by a semantic reasoner such as Racer [2] and Fact++ [3]. A service is discovered if a "match" relation holds between advertised capabilities (C_A) and requested capabilities (C_R). In such a relation, a semantic match means that C_A subsumes C_R. More precisely, all required inputs, expected outputs, and required properties of C_R are matched with the expected inputs, offered outputs, and provided properties by C_A, respectively. The result would be a group of Web services which are ranked according to a best-fit criteria.

The performance parameters that distinguish one semantic reasoner from another are: (1) the response time, which is the time taken to match a request with the capabilities provided by Web services (2) the computational resource requirements used in matching. Thus, in order to feasibly employ the semantic discovery approach in mobile services, the matching process should be optimized for these performance metrics.

[2]Racer: http://www.sts.tu-harburg.de/r.f.moeller/racer/

[3]Fact++: http://owl.man.ac.uk/factplusplus/

C. Service Discovery in Mobile Environments

In mobile environments, limited resource availability on mobile devices and unreliable communication in wireless networks present unique challenges for service discovery. A vision for discovery schemes in open mobile environments is presented by Bashah et al. [49]. Mobile providers and users are constantly changing their locations and might offer, or be interested in, location-based services. Mobile users' attitude, preferences and demands for location-aware mobile services are discussed from the user perspective by Kaasinen et al. [50].

Yang et al. [51] propose an architecture for mobile Web service discovery. Their approach aims at avoiding intermittent connections and overcoming delay and bandwidth limitations that characterize wireless communications. Their architecture is based on fixed providers, mobile users, and a single broker per particular area. The main idea of their architecture is to have mobile users downloading and executing the requested service locally; to avoid back and forth communication. However, this approach overlooks many issues that typically exist in mobile Web service such as, the services' need to access local resources such as databases, exploiting user's valuable resource (ex. battery, memory, and processing power), and management overhead if there are multiple brokers and frequent service updates.

Although semantic Web services are promising, their discovery architectures present a significant challenge in mobile domains. A typical semantic reasoner, the core of the semantic matchmaking process, is a resource-intensive process and is only suitable for deployment on high-end servers. Recently, some researchers have focused on optimizing specific parts of current semantic discovery approaches to adapt to resource-constrained mobile domains. Steller et al. [52]–[54] propose the mTableaux algorithm to optimize the reasoning process and facilitate Web services selection for limited-resource mobile providers. Similarly, Gu et al. [55] discuss, in their framework, the design principles and implementations of supporting ontology and reasoning for mobile context-aware applications on handheld devices.

Composing multiple services is an alternative to resolving a user request if no atomic services were found to satisfy the user's objective. Service composition is more feasible with semantic descriptions rather than non-semantic ones. Bhuvaneswari et al. [56] propose a framework for semantic Web service composition in mobile environments. It converts WSDL files into an OWL-S specification and generates a service profile for the request. Then, it performs sematic reasoning between the advertised service profile and the requested one. Accordingly, the composer generates composition plans and stores them in a plan repository in a cloud.

Much focus has been given for context-aware service discovery in heterogeneous mobile environments (context includes: user preferences, device profiles, environment context and service ratings). García et al. [57] 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 ones. Al-Masri et al. [58] have developed a device-aware service discovery mechanism that is capable of selecting Web services that adhere to mobile device constraints. The mechanism takes advantage of HTTP sessions to collect device information and store it at the server side. This information is later used to ensure that the discovered services will function properly within the user's device. User feedback and rating is also another important aspect that could be exploited to improve Web service discovery [59]. However, mechanisms that collect the feedback should prevent false ratings as well as providers who dishonestly claim a certain QoS for their advertised services to deceivably attract interested customers [60]. Maamar et al. [61] discuss the development, discovery, and composition of capacity-driven Web services, which are services that have the capability of changing their behavior according to environment changes. Similar research on services with different qualities to cope with environment context is presented in [62].

D. Discovery in P2P Networks

As mentioned earlier, P2P technology is being used in ubiquitous computing environments, and particularly in mobile Web services provisioning. Peers in P2P networks are always on the move and their point of attachment to the network keeps changing. Consequently, the binding information of an advertised mobile Web service needs to change; otherwise the point of service attachment would be void, resulting in a failed invocation. Maintaining this binding information is costly and difficult. Therefore, a more dynamic, robust, and distributed publishing and discovery mechanism is required in the mobile wireless domain. Using a UDDI-based approach is not the best way to deal with service advertisement and discovery in P2P networks.

In contrast to the traditional Web services model, advertising and discovery of mobile Web services in P2P follows the announce-listen model. Publishing and discovery in P2P networks relies heavily on JXTA. A Web service is often published as a JXTA module, which includes module class, module specification, and module implementation [36]. The module class represents the information needed to declare the services. The module specification contains the information required for the potential consumer to access the service. Its implementation indicates the methods (possibly across different platforms) of the advertised specifications. Modules are searchable and can be queried for a certain Web service requirement or functionality. Its class maps to the UDDI entry in the traditional Web service architecture as shown in Figure 5, while the module specification and module implementation together map to the WSDL document information [3]. Advertisements in JXTA are represented as XML documents and broadcast/multicast over the P2P network. A determined lifetime is associated with each advertisement; once it expires the corresponding service or advertisement becomes invalid or automatically deleted. This feature reduces the need of maintaining up-to-date centralized registries. To keep a service

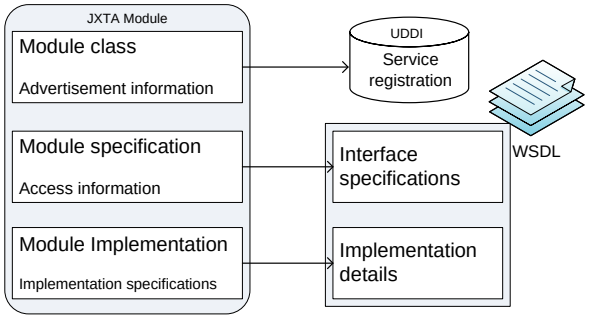

Fig. 5. Mapping between JXTA advertisements and traditional Web service publishing architecture.

advertisement valid, the service should be periodically re-published or re-announced.

Peers discover the required services by sending a search request over the network [63]. Similar to WSDL-based service discovery, the JXTA API only supports keyword-based search in advertised modules. Usually, the user's query matches the information in the module class. Therefore, information such as the user's context is not used to find the relevant service using the basic JXTA search. Sirarma [3] propose an advanced search mechanism by proper categorization to the advertisements based on functionalities, then filtering the retrieved services to find out the most relevant services. The filtering algorithm relies on the word importance calculation across all the retrieved advertisements considering the word frequency and its distribution. However, such search mechanisms need a high-end JXME peer due to the resource limitations of the regular mobile devices [64]. The scalability of P2P-based mobile Web service discovery is also studied by Zhu [65].

Sioutas et al. [66] take advantage of P2P overlay networks and proposed a fault tolerant search infrastructure based on indexing techniques to leverage Web service discovery in P2P networks. Sets of descriptive keywords are extracted from WSDL description files, indexed, then stored at peers. Request-query matching then supports keyword-matching on service name, category and tModel. Also, Vu et al. [60] propose a decentralized service discovery framework based on indexed P2P service registries. A semantic service description, including functional and non-functional properties, is stored on a peer registry on P2P overlay network. The requirements of a potential requester are expressed in the same ontology concept used to describe *the characteristic vector* of the service.

E. Open Research Issues

In addition to the publishing/discovery approaches discussed above, several open issues still exist and dictate improvements or possibly require new mechanisms all together, most importantly, to achieve ubiquitous access to services.

- **Publishing Techniques**: Mobile providers need to advertise their Web service to potential customers. In pervasive mobile domains, centralized solutions are not the optimal one. In contrast, distributed publishing approaches entail much overhead in maintaining the consistency of service registries. This raises critical research issues, including whether location-based publishing is beneficial in reducing network traffic and latency to provide better service or functional-based publishing is more appealing for mobile users. From the network operator perspective, would it be possible to promote Web services offered from inside their own networks at the expense of other services, with the same functionality, offered outside their networks?
- **Context-Awareness Discovery**: With the adoption of mobile Web services provisioning, a significant number of services are to be offered. This will make the discovery of the most relevant Web services to a certain user objective more complex, and probably inefficient, using the current mechanisms. Proper discovery techniques that exploit context information, such as location, device profiles, and user profiles, are crucial to the success of the mobile service provisioning approach.
- **Persistent Discovery**: In pervasive mobile domains, mobile devices are frequently changing their point of attachment to the network or making handoff between networks or 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 ubiquitous service access either by providing an alternative access to the same service or by quickly finding another equivalent service [67].
- **Lightweight Semantic Discovery**: Semantic Web service discovery architectures present a significant challenge in resource-constrained domains. The discovery of semantic Web services requires a heavyweight matchmaking process at the server side which could be a provider with limited resources. 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 are required.
- **User Interface**: Toolkits are required for the automatic creation of multimodal user interfaces; whether at runtime or deployment. The decision of whether a user interface should be created at runtime (service discovery time) or deployment-time is a significant research question itself when it comes to mobile and pervasive environments; taking into account mobile device constraints and the characteristics of wireless networks.

VI. Mobile Hosting Performance

Web services, by design, usually incorporate high overhead due to the usage of XML in message interactions. XML messages are verbose and costly to be processed in terms of memory, cpu, and network bandwidth. The performance of Web services hosted on mobile devices is crucial to the realization of this computing paradigm. Successful deployment of Web services on mobile devices requires that clients should not notice that these services are provided from a resource-

limited host. On the other hand, the regular functionalities of mobile devices have to be maintained without a serious impact while provisioning Web services. Therefore, the performance of mobile services provisioning needs to be extensively explored and analyzed. Researchers use various performance factors to evaluate the performance of Web services hosted on mobile devices, including: response time, battery consumption, how the system scales to accommodate an increasing number of service requests, the number of service requests processed per unit of time, and service request rejection rate [68, 69]. From the customer's perspective, response time is an important factor that indicates the perceived Web service performance. So far, little research has been done in this direction. Four different approaches proposed in the literature to tackle the performance issue of mobile services provisioning are as follows:

- **XML Compression**: Compression of XML messages is one option to boost the mobile Web services performance; however, the benefits of compression may be compromised by the decompression overhead, even with the increasing computing power of mobile devices. In such cases, a tradoff between bandwidth and computing power has to be made clear. Most of the proposed XML compression schemes [70, 71] allow users to choose whether they prefer to receive XML messages compressed or not.
- **REST Design**: The RESTful approach is another option to enhance the performance of mobile Web services [72]–[74]. AlShahwan and Moessner [32] developed two frameworks, SOAP-based and REST-based, to provide Web services from mobile devices. The study compares the performance of the two approaches in wireless resource-constrained domains. The authors claimed that REST-based Web services have proved to be more lighter, scalable, and have less footprint than SOAP-based Web services.
- **Partitioning**: Partitioning the execution of Web service components is a third direction that has been proposed to hide the limitations of mobile devices [75]. Typically the Web service execution environment encompasses many components to facilitate the hosting and execution of services, such as request listener, SOAP/XML engine, and encryption and decryption modules. Most of these components are computationally intensive and require more resources. Deploying all the required components on mobile devices is very difficult due to their resource constraints. Asif et al. [76, 77] propose a partitioning technique to execute some of the Web service components on an intermediate node, called a surrogate node. Their basic idea is to build a distributed SOAP engine, a static partition resides on the surrogate node and a mobile partition resides on the mobile device, to improve the response time and extend the scalability. The static SOAP engine processes part of the incoming SOAP message and delivers the rest to the mobile partition. Web services have to be built with this concept in mind, as XML elements would have assignment attributes for each SOAP engine. It is worth noting that providing mobile Web services with this partitioning technique follows the proxy-based architecture.
- **Service Replication**: Availability and reliability are also major challenges for mobile web services due to the intermittent connections of wireless communications and potential server down time due to the increase in the number of invocation requests [78]. Guaranteeing the availability of Web services in mobile wireless domains is difficult and as Sheng [24] point out, there is a significant difference between things that work and things that work well. Sheng et al. in their work [24] propose an on-demand replication approach on idle potential providers (ex. idle or less-loaded mobile devices in P2P architecture) for robust Web services provisioning. According to their approach, the service provider has to maintain a ready-to-deploy (a bundle that contains all the necessary files) version of the Web service for different platforms. To accommodate more execution needs, a Web service manager can seek, on-demand, for a potential service host (from a pool of Web service hosts) to deploy the service on, if the invocation requests exceed a certain limit or to guarantee a particular performance or availability.

VII. Summary

It has been recognized that mobile devices are the most convenient interfaces for pervasive and ubiquitous computing. The advancements in the manufacturing of mobile devices coupled with the latest achievements in wireless technology contribute to extend the role of mobile devices to not only consume Web services and Internet applications, but also to host and provide them. The chief advantage of providing Web services from mobile devices is that both provider and consumer can utilize the context information to personalize mobile services provisioning. In this paper, we discuss the idea of mobile Web services provisioning, point out its enabling technologies, and its envisioned applications.

Mobile Web services design may use the SOAP protocol as a messaging framework (SOAP-based) or comply with REST principles (REST-based). RESTful Web services are more suitable for resource-constrained mobile providers as they are scalable, lightweight and flexible, and easy to deploy. Several architectures are proposed in the literature for providing Web services by mobile devices to accommodate the constraints of mobile devices and the characteristics of wireless networks. Current provisioning architectures include: proxy-based, P2P-based, and asymmetric architecture. Despite the fact that a proxy-based architecture can hide the limitations of mobile devices, such a solution compromises the portability of services provisioning. The P2P-based service provisioning architecture, however, is promising. In a P2P-based architecture mobile devices can be a service provider, a service consumer, or both simultaneously.

Existing publishing and discovery mechanisms were originally developed for fixed hosting and wired networks. Mobile

networks are dynamic and their users constantly change their point of attachment to the network, handoff between different networks, and frequently and unpredictably change their location and context information. It is important that publishing and discovery mechanisms in this highly dynamic environment take into account all these characteristics. Though the JXME network offers many advantages in the discovery of services and access mechanisms in P2P networks, novel publishing and discovery mechanisms that are capable of exploiting the context information are crucial to the success and widespread adoption of the mobile services provisioning paradigm.

For efficient mobile services provisioning, different challenges and open research directions are identified, including: robust architecture that limits the overhead on resource-constrained providers, flexible provisioning frameworks that are able to make proper optimizations for mobile-related constraints, context-awareness mechanisms to personalize services provisioning, and data formats which are less complex yet flexible and agile.

It is important to keep in mind that the normal functionality of mobile devices should not be compromised while provisioning Web services. In conclusion, although hosting Web services on mobile devices is possible and proof-of-concept applications have been demonstrated, this concept remains far from realization and much more research work is yet to be done.

REFERENCES

[1] S. N. Srirama, *Mobile Hosts in Enterprise Service Integration*. PhD thesis, RWTH Aachen, Germany, September 2008.

[2] M. A. Skulason, "Mobile devices as web service providers," Master's thesis, Technical University of Denmark, Denmark, September 2008.

[3] S. N. Srirama, M. Jarke, and W. Prinz, "Mobile web service provisioning," in *Proceedings of the Advanced International Conference on Telecommunications andInternational Conference on Internet and Web Applications and Services, AICT/ICIW'06*, pp. 120–125, 2006.

[4] Y.-S. Kim and K.-H. Lee, "A lightweight framework for mobile web services," *Computer Science - Research and Development*, vol. 24, pp. 199–209, 2009.

[5] S. Srirama, M. Jarke, and W. Prinz, "Mobile host: A feasibility analysis of mobile web," in *Service Provisioning, Proc. UMICS 2006, @ CAiSE06*, pp. 942–953, 2006.

[6] A. Meads, A. Roughton, I. Warren, and T. Weerasinghe, "Mobile service provisioning middleware for multihomed devices," in *Proceedings of the 2009 IEEE 5th International Conference on Wireless and Mobile Computing, Networking and Communications(WIMOB 2009)*, pp. 67–72, 2009.

[7] A. van Halteren and P. Pawar, "Mobile service platform: A middleware for nomadic mobile service provisioning," in *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, 2006 (WiMob'2006).*, pp. 292–299, IEEE Computer Society Press, June 2006.

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

[9] 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.

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

[11] 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.

[12] R. T. Fielding, *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, USA, 2000.

[13] 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.

[14] E. Wilde, "What are you talking about?," in *IEEE International Conference on Services Computing (SCC 2007)*, pp. 256 –261, July 2007.

[15] J. Meng, S. Mei, and Z. Yan, "Restful web services: A solution for distributed data integration," in *International Conference on Computational Intelligence and Software Engineering, 2009. CiSE 2009.*, pp. 1–4, 2009.

[16] 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/.

[17] 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.

[18] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199–220, 1993.

[19] M. Klusch, B. Fries, and K. Sycara, "Automated semantic web service discovery with owls-mx," in *Proceedings of 5th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2006.

[20] J. D. Garofalakis, Y. Panagis, E. Sakkopoulos, and A. K. Tsakalidis, "Contemporary web service discovery mechanisms," *Contemporary Web Service Discovery Mechanisms*, vol. 5, pp. 265–290, September 2006.

[21] 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.

[22] 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/.

[23] 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/.

[24] Q. Z. Sheng, Z. Maamar, J. Yu, and A. H. H. Ngu, *Information Systems: Modeling, Development, and Integration*, ch. Robust Web Services Provisioning through On-Demand Replication, pp. 4–16. Springer Berlin Heidelberg, April 2009.

[25] F. Aijaz, B. Hameed, and B. Walke, "Towards peer-to-peer long lived mobile web services," in *Proceedings of the 4th International Conference on Innovations in Information Technology*, (Dubai, UAE), pp. 571–575, IEEE, November 2007.

[26] S. Microsystems, "Personaljava," 2007. http://javasun.com/products/personaljava/.

[27] S. Microsystems, "The java me platform - the most ubiquitous application platform for mobile devices," 2007. http://java.sun.com/javame/index.jsp.

[28] N. Balani, "Using kxml to access xml files on j2me devices," 2003. http://www.ibm.com/developerworks/edu/wi-dw-wi-kxml-i.html.

[29] kSOAP2. http://ksoap2.sourceforge.net/.

[30] K. Church, J. M. Pujol, B. Smyth, and N. Contractor, "Mobilehci'10 workshop summary: social mobile web," in *The 12th international conference on Human computer interaction with mobile devices and services*, MobileHCI '10, (New York, NY, USA), pp. 509–512, ACM, 2010.

[31] D. Brooker, T. Carey, and I. Warren, "Middleware for social networking on mobile devices," in *The Australian Software Engineering Conference (ASWEC'10)*, (Auckland, New zealand), pp. 202–211, 2010.

[32] F. AlShahwan and K. Moessner, "Providing soap web services and restful web services from mobile hosts," *International Conference on Internet and Web Applications and Services*, vol. 0, pp. 174–179, 2010.

[33] P. Pawar, S. Srirama, B.-J. van Beijnum, and A. van Halteren, "A comparative study of nomadic mobile service provisioning approaches," in *The 2007 International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST 2007)*, pp. 277–283, 2007.

[34] S. Microsystems, "Jini architecture specification," 2001. http://www.jini.org/wiki/Jini_Architecture_Specification.

[35] T. F. L. Porta, K. K. Sabnani, and R. D. Gitlin, "Challenges for nomadic computing: Mobility management and wireless communications," *Mobile Networks and Applications*, vol. 1, no. 1, pp. 3–16, 1996.

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

[37] S. Microsystems, "Jxta(tm) community project." https://jxta.dev.java.net/.

[38] J. Fuller, M. Krishnan, K. Swenson, and J. Ricker, "Oasis asynchronous service access protocol (asap," May 18 2005. http://www.oasis-open.org/committees/documents.php?wg_abbrev=asap.

[39] 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, (Washington, DC, USA), pp. 113–118, IEEE Computer Society, 2007.

[40] C. Legner in *Service-Oriented Computing - ICSOC 2007 Workshops* (E. Nitto and M. Ripeanu, eds.), ch. Is There a Market for Web Services?, pp. 29–42, Berlin, Heidelberg: Springer-Verlag, 2009.

[41] 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.

[42] 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.

[43] M. Rambold, H. Kasinger, F. Lautenbacher, and B. Bauer, "Towards autonomic service discovery a survey and comparison," in *Proceedings of the 2009 IEEE International Conference on Services Computing*, SCC '09, (Washington, DC, USA), pp. 192–201, IEEE Computer Society, 2009.

[44] R. Nayak, "Data mining in web services discovery and monitoring," in *International Journal of Web Services Research*, vol. 5, pp. 63–81, 2008.

[45] K. Elgazzar, A. E. Hassan, and P. Martin, "Clustering wsdl documents to bootstrap the discovery of web services," in *The 8th IEEE International Conference on Web Services (ICWS'10), Miami, Florida, USA*, July 2010.

[46] S. Ben Mokhtar, A. Kaul, N. Georgantas, and V. Issarny, "Efficient semantic service discovery in pervasive computing environments," in *Middleware '06: Proceedings of the ACM/IFIP/USENIX 2006 International Conference on Middleware*, (New York, NY, USA), pp. 240–259, Springer-Verlag New York, Inc., 2006.

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

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

[49] 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*, (Washington, DC, USA), pp. 47–53, IEEE Computer Society, 2010.

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

[51] X. Yang, A. Bouguettaya, and B. Medjahed, "Organizing and accessing web services on air," *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 33, pp. 742–756, 2003.

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

[53] L. A. Steller, S. Krishnaswamy, and M. 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.

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

[55] 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.

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

[57] 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 of *LNCS*, pp. 1–14, 2010.

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

[59] A. Averbakh, D. Krause, and D. Skoutas, "Exploiting user feedback to improve semantic web service discovery," in *Proceedings of the 8th International Semantic Web Conference*, ISWC '09, (Berlin, Heidelberg), pp. 33–48, Springer-Verlag, 2009.

[60] L.-H. Vu, M. Hauswirth, and K. Aberer, "Towards P2P-based Semantic Web Service Discovery with QoS Support," in *BPM 2005 Workshops, LNCS 3812*, pp. 18–31, Springer Berlin / Heidelberg, 2006.

[61] 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)*, vol. 0, pp. 403–410, 2009.

[62] 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*, (Washington, DC, USA), pp. 241–252, IEEE Computer Society, November 2007.

[63] S. Dustdar and M. Treiber, "Integration of transient web services into a virtual peer to peer web service registry," *Distrib. Parallel Databases*, vol. 20, no. 2, pp. 91–115, 2006.

[64] S. N. Srirama, M. Jarke, and W. Prinz, "Scalable mobile web service discovery in peer to peer networks," in *3rd International Conference on Intenet and Web Applications and Services*, pp. 668–674, 2008.

[65] H. Zhu, "Scalability of p2p based mobile web services discovery," Master's thesis, RWTH Aachen University, Germany, 2008.

[66] 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.

[67] K. Elgazzar, H. Hassanein, and P. Martin, "Effective web service discovery in mobile environments," in *P2MNETS, The 36th IEEE Conference onLocal Computer Networks (LCN)*, pp. 697–705, October 2011.

[68] R. Mizouni, M. Serhani, R. Dssouli, A. Benharref, and I. Taleb, "Performance evaluation of mobile web services," in *The 9th IEEE European Conference on Web Services (ECOWS*, pp. 184 –191, September 2011.

[69] F. AlShahwan and K. Moessner, "Providing soap web services and restful web services from mobile hosts," in *The Fifth International Conference on Internet and Web Applications and Services (ICIW'10)*, pp. 174 –179, May 2010.

[70] M. Tian, T. Voigt, T. Naumowicz, H. Ritter, and J. Schiller, "Performance considerations for mobile web services," *Elsevier Computer Communications Journal*, vol. 27, pp. 1097–1105, 2003.

[71] Y. Natchetoi, H. Wu, and G. Babin, *Euro-Par 2007 Parallel Processing*, ch. A Context-Dependent XML Compression Approach to Enable Business Applications on Mobile Devices, pp. 911–920. Springer Berlin Heidelberg, August 2007.

[72] H. Hamad, M. Saad, , and R. Abed, "Performance evaluation of restful web services for mobile devices," *International Arab Journal of e-Technology*, vol. 1, pp. 72–78, January 2010.

[73] F. Aijaz, S. Z. Ali, M. A. Chaudhary, and B. Walke, "Enabling high performance mobile web services provisioning," in *Proceedings of the 2009 IEEE 70th Vehicular Technology Conference Fall*, p. 6, September 2009.

[74] F. Alshahwan, K. Moessner, and F. Carrez, "Distribute provision strategies of restful-based mobile web services," in *GLOBECOM - IEEE Global Telecommunications Conference*, pp. 1–6, 2011.

[75] M. Asif and S. Majumdar, "Partitioning frameworks for mobile web services provisioning," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 26, no. 6, pp. 519 – 544, 2011.

[76] M. Asif, S. Majumdar, and R. Dragnea, "Partitioning the ws execution environment for hosting mobile web services," *Services Computing, IEEE International Conference on*, vol. 2, pp. 315–322, 2008.

[77] M. Asif and S. Majumdar, "A graph-based algorithm for partitioning of mobile web services," in *2009 IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2009.

[78] K. Elgazzar, P. Martin, and H. Hassanein, "A framework for efficient web services provisioning in mobile environments," in *The 3rd International Conference on Mobile Computing, Applications, and Services(MobiCASE'11)*, Springer's LNICST, October 2011.