Mobile Web Services: State of the Art and Challenges

Khalid Elgazzar, Patrick Martin, Hossam Hassanein School of Computing, Queen's University, Canada
Kingston, Ontario, K7L 3N6
email{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 latest achievements in wireless communication developments are both key enablers for shifting the role of mobile devices from service consumers to service providers. This paradigm shift is a major step towards the realization of pervasive and ubiquitous computing. Mobile Web service provisioning is the art of hosting and offering Web services from mobile devices, which actively contributes towards the direction of Mobile Internet. In this paper, we provide the state of the art of mobile service provisioning as it currently stands. We focus our discussions on its applicability, reliability, and challenges of mobile environments and resource constraints. We study the different provisioning architectures, enabler technologies, publishing and discovery mechanisms, and maintenance of up-to-date service registries. We point out the major open research issues in each provisioning aspect. Performance issues due to the resource constraints of mobile devices are also discussed.**

***Index Terms*—Mobile Web services, service provisioning, mobile devices, ubiquitous computing, mobile computing.**

I. Introduction

Mobile devices (such as smartphones and PDAs) are traditionally recognized as resource-limited devices. Designers of mobile applications take these resource constraints into account in order to improve the performance of applications. While this is true, the manufacturers of mobile devices have recently achieved breakthroughs in extending mobile devices' capabilities in terms of memory, computational power, storage capacity, and display screen. 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.

In parallel, the revolution in wireless communications achieved astonishing developments in increasing transmission rates and improving the spectral efficiency. The 4G network introduces a flexible and programmable platform to provide users access to future services and applications from a single terminal. Cellular networks are able to accommodate more users and offer a wide range of customized services with various quality of service (QoS) levels. New services are increasingly offered to mobile users, capitalizing on the ever-expanding mobile customer base. According to the latest Mobile FactBook released by PortioResearch [1], the global mobile customer base exceeded 6.5 billion subscribers in the beginning of 2013, which represents 87% of the world's current population. Additionally, 1.5 billion of those subscribers have broadband access to Internet services. Mobile users are always demanding better user experience and service personalization that can fit their dynamic context change and accommodate their preferences. The demand for such smart services that can fully utilize the user's mobility and remove barriers between network technologies is on the rise.

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 on the rise. Consequently, researchers and industry are inspired to pave the road for mobile Web service provisioning [2], [3], [4], [5], [6], [7], [8].

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. In this paper, we describe the state-of-the-art of mobile Web service provisioning, address its applicability and reliability, point out the research efforts, and explore the challenges and open research problems. Throughout the remainder of this paper, we refer to mobile Web service provisioning as mobile services.

The remainder 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 discusses 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 [9]. In SOA, a service is defined by a Web interface that supports interoperable operations between dif-

ferent software applications using a standard messaging protocol [10]. 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 [11]. A service, in contrast, is the actual value provided by the service invocation [12]. 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 Service Design

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 [13] 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) [14], 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). SOAP-based Web services expose only a single endpoint, by which users communicate with offered functionality.

The strength of SOAP 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 primitive data types. Each system or platform maps these types to their internal data types. SOAP has a rigid type checking mechanism, by which SOAP performs most of the standard data verifications. SOAP messages are not tied to any particular current or future transport protocol.

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. 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*: High 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, include redundant information which in turn increases the network traffic and consumes more resources.
- *Data Model*: SOAP-based approach hides the data model behind the Web service interface. This feature dictates that the service consumer and provider have to share a common model to communicate. However, SOAP advocates argue that keeping the data model away from the clients is safer and less risky.
- *Scalability*: The SOAP messages are interpreted only outside the Web by different applications. Since consumers and providers must establish a common ground to be able to communicate, scalability is an issue as it fails to achieve the proper integration with the Web as a shared information model.

2) REST-based: 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. RESTful Web services [15] conform with the concepts of REST in order to avoid the performance degradation resulting from the use of SOAP and XML. Services designed with the RESTful approach expose their functionality as Web resources, each resource

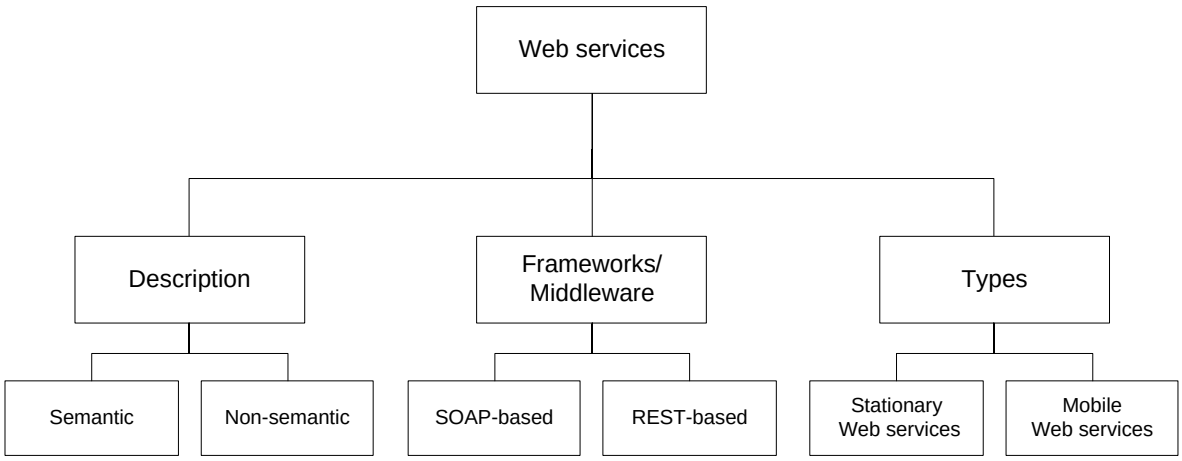

Fig. 1. A general overview of Web service descriptions, design styles, and service types.

is addressed with a unique URI. A user can access the desired resource directly by the associated URI or traverse offered functionality through a hierarchical structure. RESTful approach offers great flexibility and scalability. Although it is tightly coupled with the HTTP protocol, the approach is more suitable to mobile domains. Performance evaluation studies show that RESTful Web services outperforms their SOAP-based counterparts within resource-constrained environments In this regard, extensive performance analysis has been conducted to investigate the performance aspects of the two design frameworks [16], [17], [18], [19], [20], [21], [22].

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

- *Scalability*: The RESTful approach inherits the underlying scalability of HTTP.
- *Addressability*: Resources (services) are exposed and accessed through a valid URI rather than requiring a centralized repository 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*: Resources can link to each other using hyperlinks and 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 [11].

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 [11].

- *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

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	No standard
Data Model	Hidden	Exposed
Data Representation	XML	XML, JSON, etc.
Scalability	None	Connected hyperlinks
Security	WS-security-based	HTTP-based

style.

B. Web Service 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: Non-semantic Web services are described by the Web Service Description Language (WSDL). WSDL enable service providers to describe their services and explain to potential customers how to consume offered functionalities [14]. WSDL 2.0 is the latest WSDL standard specification [14]. It describes the service in two levels; "emphabstract" 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 involves various 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 [23]. 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 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* [24]. An *ontology* is a formal explicit specification of a shared conceptualization [25]. 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 [12]. 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 turn makes the interactions between services more logical and facilitate composition and integration of Web services. In contrast to the non-semantic approach, the 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 [23]. 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 [26], [27].

Web services may use various semantic description languages such as Web ontology languages (OWL-S) [28],Web Service Modeling Ontology (WSMO) [12], WSMO-Lite [23], Web Services Semantics (WSDL-S) [29], and Semantic Web Services Ontology (SWSO) [30], [12]. 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.

A similar approach to ontologies that can be categorized under semantic descriptions is folksonomies [31]. Folksonomies adopts Web 2.0 social participation to tag Web resources based on user conceptions using user-generated metadata. Folksonomies are recently employed in the Web service domain to tag and discover required services [32]. Although this approach is scalable and offers great flexibility, it may suffer from inaccuracy due to lack of common terminologies or ambiguous tagging by inexperienced users. Filtering techniques and vocabulary controlling mechanisms, therefore, are required to improve the accuracy and reliability of the folksonomy approach. In contrast to ontologies, folksonomies share the same concept but with different implementation. However, non-taxonomic relations between tags can be generated from folksonomies in order to construct ontology-like structures [33]. Folksonomy structures are less resource-intensive and more lightweight that can better accommodate the resource constraints of mobile environments, while offering higher flexibility and maintaining reasonable reliability.

With the adoption of mobile services, the expected number of offerings is quite high. Therefore, searching for the right service(s) for a particular objective is a key challenge. The syntactic level of description, offered by a 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 on the syntax. However, semantic discovery is a resource-intensive process that conflicts with the resource constraints of mobile environments. This problem can be approached in two ways, either augmenting the capability of mobile devices thorough cloud computing [34], [35] or optimizing semantic descriptions and reasoning to accommodate the various constraints of mobile environments [36]. Cloud computing seems to be more reasonable approach, especially with the latest widespread deployments and adoption of cloud services. Bringing the power of the Cloud to the realm of mobile services opens up the opportunities to employ advanced techniques that were deemed beyond the capability of resource-constrained environments, such as semantic approaches.

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*. This paper focuses on mobile Web services and applicability of provisioning from resource-constrained mobile devices. The next section explains and elaborates more on both types of Web services.

1) Stationary Web Services: Stationary Web services are deployed on fixed servers, such as Amazon and Google Web services. They are usually tied to the availability of local resources such as databases hosted on the local data center [37]. These Web services are accessible by both fixed and mobile consumers through their advertised unique addresses. Stationary Web services are reliable and can provide guaranteed QoS due to abundant resource availability in fixed environments. Computational and network resources can scale up and down to accommodate variations in demands, while maintaining a high level of resource utilization. Stationary Web services, therefore, can serve a huge number of users. Services can be replicated on multiple servers to support distributed provisioning and/or avoid a single point of failure and offer better service reliability and guaranteed quality. Service replication decisions could be static based on prespecified conditions, at design time, or dynamic at runtime according to context changes.

Web service communication schemes are independent of the type of consumers and providers, which means that the

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

communication strategies do not care whether one or both of the communication parties are mobile nodes [38]. Traditional Web services are designed to behave synchronously, i.e. users are blocked while executing. In mobile domains, the synchronous execution of long-lived processes is not the right choice. Regular functionality of mobile terminals, such as voice calls or running applications, must be maintained during the execution of Web services whether the terminal is a consumer or provider.

2) Mobile Web Services: Mobile Web services are deployed on mobile devices and provisioned over wireless networks [2], where devices may play the role of consumer, broker, or provider. Mobile services is a fairly new and many challenges with respect to the limitations of mobile devices and the characteristics of broadband wireless access remain open. Mobile Web services was first introduced as a computing paradigm in the early twenty-first century. Since then, much of the of research efforts have focused on enabling reliable mobile service provisioning despite the limitations of mobile environments. These limitations are summarized as the following along with respective challenges:

Limited Resources. Although the capabilities of mobile devices have improved in terms of processing power, memory space, and embedded sensors, they continue to lag behind other forms of computing devices. The major constraint for mobile devices is their limited display screens which can relatively display smaller amounts of data at a time, resulting in compromising the usability of applications. So, mobile devices (smartphones in particular) are still recognized as resource-constrained computing devices [39], [40].

Intermittent Connectivity. Mobile devices frequently change network operators and may handover between different technologies within the same network. Mobile devices are expected to experience frequent link failures and consequently any services they may be offering would become temporarily unreachable. This presents big challenges for providing reliable Web services in highly dynamic mobile wireless environments. Therefore, mobile Web services are not suitable for services that are expected to be continuously available [41].

Addressability. Mobile devices may frequently change their point of attachment to the network as they relocate. Changing the network provider or network technology typically results in changing the mobile provider's IP address (unless a static IP is assigned), which in return makes services' binding information invalid if not properly updated [42]. Therefore, mobile Web services might become stale or inaccessible.

Scalability. Given the limited resources of a mobile device, mobile Web services do not scale well when a large number of customers are expected to concurrently access the Web service [17].

Battery Power. The possibility of battery power outage remains a major challenge for the potential growth of mobile computing. Recent developments in mobile computing and the growing popularity of mobile applications outpace what current battery technologies can provide [43].

Resource Heterogeneity. The operating systems and software platforms on mobile devices span multiple vendors and feature a wide range of characteristics and supporting functionalities. Providing an interoperable mobile service that is platform-independent and still can perform uniformly across heterogeneous platforms is another dimension of stringent constraints imposed by mobile environments.

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

- **Java ME**: Java platform, Micro Edition (J2ME) [44] 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 tablets 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 [45] is a lightweight XML parser designed specifically for constrained environments.
- **kSOAP2**: kSOAP2 [46] 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.

In the next section, we present the current and potential mobile applications and domains that can benefit from the mobile services.

III. Potential Applications of Mobile Services

The approach of mobile Web services introduces a new range of mobile applications that promise advanced mobile computing paradigms, differential user experience, and seamless data access across different platforms. One of the chief benefits that mobile services offer is the enabling of personalized service provisioning, where services are specifically customized and tailored at the best interest of the service requester and most appropriate for the current situation. This is due to the fact that mobile devices are associated with users who have personal preferences, beside the ability to access various contextual information. In general, mobile Web services open up opportunities for users who wish to share personal information and functionalities, yet still maintain full control over their personal data. The utility of mobile Web services may extend to domains that are deemed beyond the capabilities of mobile devices. Such domains include:

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 personal services' ability to guarantee privacy under the provider's control.

Healthcare monitoring. Mobile devices may provide low-cost mobile and efficient remote health monitoring by utilizing the mobile services approach. Mobile devices can collect a patient's vital signs in a real-time fashion using the sensors embedded in the mobile device, where applicable, or via communication with a body sensor network without interfering with the activities of the patient [47]. These data, then, may be offered or made available upon request to caregivers of interest who wish to follow changes in the patient's health conditions. Consequently, caregivers may provide the appropriate medical assistance or promptly respond to a critical health concern. Mobile services could also provide instant access to continuously changing context attributes, such as a patient's location.

Personal publishing. An author writing articles or blog posts may request feedback from close friends or professionals using mobile services. While giving a lecture or presentation, a speaker could also receive questions and comments from the audience via Web services hosted on his/her mobile devices.

Mobile learning. In mobile learning (m-learning) scenarios [48], learners can share resources such as videos, audio, documents and comments. Participants in m-learning may also assume one or more roles including learners, mentors and peer-tutors. With mobile services, participants can manage their own learning profile, including their progress and expertise, and share their experience with friends or learning partners while maintaining their privacy.

Personal information. Personal profile, photo/video sharing, and schedules are amongst the applications that can potentially benefit from the concept of mobile services.

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 [49], [50]. Personal Web 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. Architectures of Mobile Services

Over the past few years, several researchers have proposed different architectures and frameworks for providing Web services from 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 such as reachability, reliability, and scalability.

Most of the current mobile service architectures are still in their infancy stages [16]. Pawer et al. [51] discuss briefly different mobile service architectures and classify them into three categories, proxy-based, Peer-to-Peer, and asymmetric. In this section, we discuss in details current architectures, illustrating the merits and shortcomings of each one, and comparing them side-by-side.

A. Proxy-based Architecture

The proxy-based mobile service 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. 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 offload the resource-constraint devices and perform the resource-intensive processes. Proxy-based architectures offer caching to support disconnected mobile services and accommodate high access demands, while maintaining reasonable latency.

From implementation perspectives, proxy-based architectures relies on Jini technology [52]. 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.

Figure 2 shows an abstract overview of the proxy-based architecture. It 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

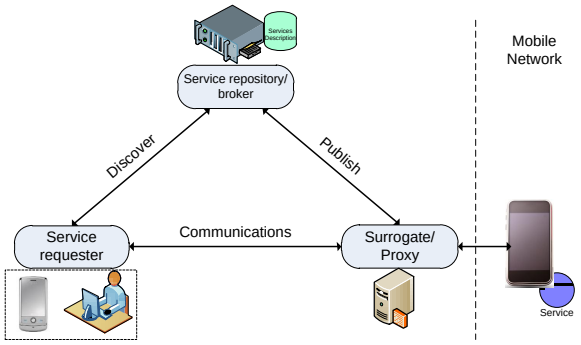

Fig. 2. An abstract overview of proxy-based mobile Web service architecture.

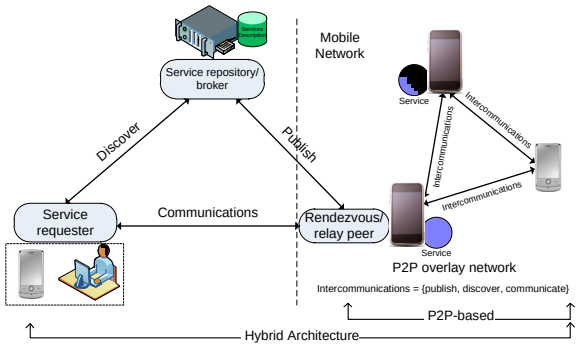

Fig. 3. P2P-based architecture of mobile Web service architecture.

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.

Proxy-based architectures resolve many challenges facing mobile Web services. For example, proxies are more capable than mobile devices to serve a large number of clients simultaneously with acceptable performance and response time. Proxies also deal with different protocol translations, since most of Web service protocols are not originally designed for wireless communications (i.e. are not optimized to tolerate high-latency, unreliable communications, and intermittent connections). A proxy-based architecture can guarantee a reasonable QoS in contrast to a fully wireless domain wherein providing a particular level of QoS is quite difficult. Proxies can also hide the heterogeneity of various mobile devices and support mobile terminals with disconnected states [53].

B. P2P-based Architecture

Peer-to-Peer (P2P) technology is a distributed, low-cost, and collaborative computing paradigm. The vision of the P2P mobile Web service architecture is to overcome the limitations of centralized approaches and to take advantage of the flexibility that P2P architectures offer. Figure 3 illustrates the basic architecture of providing Web services from mobile devices over P2P networks. It also shows how P2P-based, proxy-based, and asymmetric architectures may coexist to form a hybrid architecture. P2P mobile service architecture 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 must be republished 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.

The most popular implementation of P2P mobile Web services uses JXTA technology [54], [55], where publication and discovery of services are handled using JXTA protocols. Services are advertised as JXTA modules, where a module is composed of a module class, a module specification, and a module implementation. This corresponds to WSDL and UDDI in the traditional Web services approach. Clients must join the P2P network and get a *PeerID* before they are able to discover services. The mapping between the current IP address and the PeerID of a client is managed by the underlying JXTA protocols. Clients can query the network for required services while they are connected. Once the required service is located, the client communicates with the service provider through JXTA pipes in order to send and receive messages.

Personal Service [56], [57] is a recent implementation of mobile Web service provisioning based on the P2P architecture. Personal services are intended to offer a range of personal data services either with limited authorized access (e.g., healthcare services [58] and multimedia sharing) or public access (e.g, crowdsourciong and participatory sensing). The authors propose a set of new techniques for service publication, discovery, and directory. All these techniques promote collaborative computing through distributing required processing amongst all participating nodes.

C. Asymmetric Architecture

The asymmetric approach follows the traditional Web service architecture as shown in Figure 4 except that the service provider is a mobile device. The term asymmetric was coined by Porta [53] for those approaches that address the resource limitations of mobile devices. Figure 4 shows the architecture of asymmetric mobile service architecture.

Several adaptations have been proposed for traditional Web service protocols and mechanisms to accommodate the limitations of mobile devices. For example, asymmetric architecture supports only simple XML data types such as *String*, *Integer*, and *Char*), to avoid complex type extraction [51]. Another example is the utilization of the *Asynchronous Service Access Protocol (ASAP)* [59] in the communication of mobile services. ASAP is specifically designed by OASIS to target the

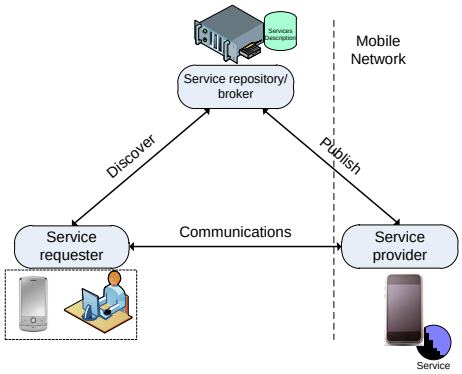

Fig. 4. Overview of asymmetric mobile service architecture.

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 device during the execution. The response is send back whenever it is available or a separate request can be made later by the client to communicate the results of the Web service operations. ASAP implements asynchronous interaction techniques such as *"Callback"*, where the server sends the response to the client whenever it is ready or *"Polling"*, where the client re-establishes a connection later to check whether the results are ready. ASAP allows the client to query the Web service for its current status during the execution time. It also enables clients to send updates to change their previous information. This is to accommodate changes in the client requirements during the course of prolonged execution times. Asynchronous Web services though should have the ability to update their behaviour accordingly at runtime.

Recent studies have shown the feasibly of asynchronous mobile service provisioning. Elgazzar et al. [41] propose a generic framework for efficient Web services provisioning in mobile heterogeneous environments. The framework aims at exploiting the inherent ubiquity of mobile devices and their association to a specific context in order to provide reliable and personalized service in an ad-hoc fashion. Aijaz et al. [38] 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. Kim and Lee [5] propose a high level description of framework that hosts Web services on mobile devices and supports service migration. Their framework handles interrupted connections through service migration to a new suitable host to maintain service reliability. The migration can be triggered by providers or upon detection of service disruption for example, due to overload or battery outage.

Table III provides a comparison between different approaches. While the existing proxy-based and P2P architectures rely on other technologies for setup, such as Jini and JXTA, the asymmetric approach can work with no infrastructure support, for example over direct WiFi links. However, several performance concerns associated with the asymmetric architecture remain challenging, most importantly the resource limitations of mobile providers. Recently, mobile cloud computing has become a driving force for mobile services, where the cloud offers computational resources on demand to augment the capability of resource-constrained mobile providers [35]. Service architectures that adopt mobile cloud computing fall under the asymmetric category, since mobile providers remain the destination for service requests.

D. Open Research Issues

The distinct features of mobile services and the constraints of mobile devices introduce a number of critical design challenges for efficient architectures. None of the proposed architectures achieves some sort of balance between performance and reliability. The simplicity and scalability of the proxy-based architecture compromise its portability and consistency. The robustness of the P2P architecture sacrifices its reliability. The performance of the asymmetric approach falls short due to the resource limitations on mobile devices as well as the possibility of intermittent connectivity.

Although significant steps have been taken towards the realization of reliable mobile Web services, many challenges remain open. Next, we highlight some of the open research issues that require further investigation.

- **Architecture**: Current mobile Web service architectures are basically adapted from the traditional Web service approach. Rethinking the architecture would lead to better solutions for mobile service provisioning. Robust architectures that limit the overhead on resource-limited providers and that meet the requirements of mobile environments are needed. Cloud computing is a viable option that needs further investigations on how it can support reliable mobile service provisioning.
- **Frameworks**: SOAP/WSDL are the defacto standards used by Web services. This framework poses challenges on resource-constrained environments due to verbose XML and its significant resource demands of parsing. Optimizations made to accommodate the constraints of mobile environments often compromise the performance. Efficient frameworks must be generic to fit the various requirements of heterogenous mobile platforms and device form factors. Therefore, a platform-independent framework for mobile services is definitely required.
- **Performance**: The widespread adoption of the mobile service approach is only possible if architectures and frameworks are capable of achieving reasonable performance without seriously affecting the regular functionality of mobile devices, such as voice services. The performance of mobile service needs further investigation.
- **Context-awareness**: Context information makes service personalization possible. Mobile services must take advantage of the association of mobile devices to particular users to provide differential user experience through personalization. Additionally, context information such as mobile capabilities, available add-on devices, location and user profile offers great opportunity to augment mobile service provisioning in terms of target publishing,

TABLE III
COMPARISON SUMMARY BETWEEN DIFFERENT MOBILE SERVICE 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 a 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

discovery, and usage. Efficient models are desired in order to incorporate context information to mobile services.

- **Data Formats**: Though XML is a portable format, its message parsing is resource-intensive due to verboseness and redundancy. Less complex and concise data formats would bootstrap the performance of mobile services.
- **Supporting Toolkits**: Mobile devices have limited input capabilities and small screens. Deploying and managing mobile Web services in such environments is quite challenging. However, developers can take advantage of unique features of mobile systems, such as multimodality and embedded capabilities (cameras, GPS, microphones, etc.), to develop robust and powerful interfaces that facilitate the deployment and user interactions with services. Such interfaces would leverage the use of mobile services.
- **Asynchronous Execution**: Users cannot tolerate lengthy locking of their terminals while executing mobile services. Limited bandwidth and intermittent connectivity pose additional challenges to lengthy Web service processes. In addition, clients of mobile Web services may prefer to disconnect while the service continues execution. Therefore, robust asynchronous techniques are of high interest to the mobile Web service approach.
- **User Feedback**: Web 2.0 has enabled users to share their user experience with others. Mobile users tend to trust the user-perceived quality of service more than claimed by service providers. Efficient utilization of user feedback leverages the adoption of mobile services. Therefore, new approaches are required to efficiently handle users feedback and maintain reliable service ratings.

V. WEB SERVICES PUBLISHING AND DISCOVERY

Publishing a Web service is the process of notifying users with the existence of such a service and providing all the required information to access it. Providers have two options to publish their services: either to register services with a service broker in a public service repository, using the UDDI standard, or to advertise them in a local service directory that is publicly accessible. 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 system failure is 100% if the discovery process fails to find the correct service.

Most of the existing discovery techniques belong to one of three main categories [60]: UDDI Business Registry (UBR), specialized search engines, and generic search engines. A comprehensive comparison between these approaches including advantages and shortcomings of each, is given in [39].

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 [61]. 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.

The discovery process is quite different according to the Web service description method. Semantic Web services are discovered by high level match-making approaches [6], whereas non-semantic Web services discovery use information retrieval techniques [59] based on keyword matching.

A. Non-semantic Discovery

In the WSDL-based service discovery approach, a Web service discovery engine partially matches the search terms (keywords) entered by the user with the Web service name, location, business, or tModel [62] 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. [63] 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 difficult, especially in pervasive environments [64]. In such environments, service discovery should be robust and lightweight enough to cope with the network's dynamics and resource-constrained 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

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 [65]). Then, a matchmaking process is performed to match the request requirements with the offered capabilities. The matchmaking process is comprised of three steps: 1) parsing both the user request for requirement and service profile for capabilities, 2) using a semantic reasoner to load the ontologies used by the user request and service advertisement, 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 [1] and Fact++ [2]. 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.

Amigo-s [64] 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.

[1]Racer: http://www.sts.tu-harburg.de/r.f.moeller/racer/
[2]Fact++: http://owl.man.ac.uk/factplusplus/

C. Service Discovery 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. [66]. 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. [67].

Several studies have focused on overcoming specific limitations of mobile service discovery, such as semantic reasoning. Steller et al. [68], [69], [36] propose the mTableaux algorithm to optimize the reasoning process and facilitate Web services selection for limited-resource mobile providers. Similarly, Gu et al. [70] discuss the design principles and implementations of supporting ontology and reasoning for mobile context-aware applications. Bhuvaneswari et al. [71] 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. The composer generates composition plans and stores it in a plan repository in a cloud. Yang et al. [72] propose an architecture for mobile Web service discovery, aiming at avoiding intermittent connections and overcoming delay and bandwidth limitations. Their architecture enables mobile users to download and execute services locally in order to avoid unnecessary back and forth communication. However, this approach overlooks many issues that typically exist in mobile service such as, access remote local, depletion of mobile resources, and consistency of services.

Much attention has been given to context-aware service discovery in heterogeneous mobile environments. Such context includes user preferences, device profiles, environment parameters and service ratings. Elgazzar et al. [73] propose a personalized Web service discovery approach that uses various context information to discover services that best fit the user interests. DaaS [74] is a context-aware cloud-based platform that offers discovery as a service. It exploits the capability of cloud computing to leverage elastic resource provisioning to support advanced resource-intensive discovery techniques. DaaS offers robust service discovery through the integration of the user context and preferences with the discovery process. García et al. [75] 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. [76] 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 used to improve Web service discovery [77].

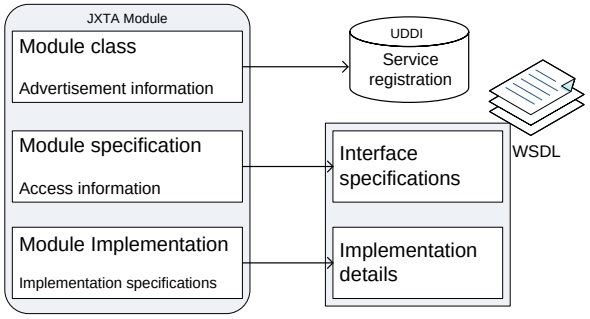

Fig. 5. Mapping between JXTA advertisements and traditional Web service publishing architecture.

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 falsely attract customers [78]. Maamar et al. [79] discuss the development, discovery, and composition of capacity-driven Web services, which change their behavior according to environment changes. Similar research on services with different qualities to cope with environment context is presented in [80].

D. Discovery in P2P Networks

Peers in P2P networks are always on the move, changing their point of connection to the network. Consequently, the binding information of an advertised mobile service needs to be updated accordingly. Improper handling of service binding information results in failed invocations. Maintaining such binding information consistent is costly and challenging in mobile environments.

In contrast to the the traditional Web service model, advertisement and discovery of mobile services in P2P follows the announce-listen model. Most of the existing research efforts concerning publishing and discovery in P2P networks relies on JXTA technology. Such implementation publishes Web services as a JXTA modules, where each module include module class, module specification, and module implementation [54]. 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 [4]. 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 advertisement valid, the service should be periodically republished or re-announced.

Peers discover the required services by sending a search request over the network [81]. The JXTA API supports only keyword-based search in advertised modules. The user's query is matched against 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 [4] propose an advanced search mechanism through categorization of advertisements, based on functionalities, and filtering of retrieved 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 [82]. The scalability of P2P-based mobile Web service discovery is also studied by Zhu [83].

Sioutas et al. [84] take advantage of P2P overlay networks and propose 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 supports keyword-matching on service name, category, and tModel. Vu et al. [78] 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.

Recently, Elgazzar et al. [56] propose a new approach to advertise and discover services in P2P mobile environments, capitalizing on the unique features of mobile devices. The approach uses contact lists, ranging from phonebook and emailing lists to social circles, to advertise and discover personal services. Their approach also proposes an access control scheme that places service providers in full control over service functionality and access management. The authors adopt a distributed service directory approach, in which each node maintains it's own copy of service registry to distribute the processing load across participating nodes and alleviate the burden on limited mobile resources.

E. Open Research Issues

In addition to the publishing/discovery approaches discussed above, several open issues still exist requiring improvements or possibly new mechanisms all together.

- **Publishing Techniques**: In mobile domains, centralized approaches are prone to failure due to limited resources and unreliable connections. In contrast, distributed publishing approaches entail much overhead in maintaining the consistency of service registries. This raises a fundamental question, what is the best way to publish services in mobile environments? Is location-based publishing beneficial in reducing network traffic and latency or functional-based publishing is more appealing for mobile users? Is it possible and beneficial for mobile network operators to favor local services, i.e. offered from inside their own networks, at the expense of similar services

offered outside? These questions remain open and warrant further investigation.

- **Discovery Mechanisms**: With the adoption of mobile services, a significant number service offerings are expected. This will make the discovery of the most relevant Web services to a certain user objective more challenging. Although semantic approaches yield better results, their resource demands are beyond what mobile devices can afford. Efficient discovery mechanisms, that are capable of utilizing the various context information that mobile devices can offer, are at the core of leveraging service personalization.
- **User Interface**: Services with user-friendly interfaces are of greater interest to mobile users. Developers have more options to develop appealing multimodal interfaces for mobile users, leveraging embedded capabilities of mobile devices. Robust toolkits that build upon such features are highly desired for the adoption of mobile services. The decision of whether to generate adaptive user interfaces during discovery or at the development time is a significant research question in of itself, taking into consideration the constraints of both mobile devices and networks.

VI. Performance of Mobile Services

Overcoming the resource limitations of mobile devices is a challenge that continues to be at the core of the research interests for the realization of reliable pervasive and mobile services. Notwithstanding the research efforts that have focused on enabling Web service provisioning from resource-constrained providers, many limitations with respect to mobile environments remain. Several studies address the resource limitations on mobile devices while providing mobile services from different perspectives. Some studies propose offloading resource-intensive tasks to either the cloud [34], [43], [85] or nearby capable computing machines [86]. The offloading approach offers mobile devices the flexibility to customize the service interactions and optimize the resource consumption [87]. Even though the offloading approach is a good candidate to augment the capabilities of mobile devices, it needs a good partitioning strategy to balance the tradeoff between the amount of data transfer and the reduction in the amount of mobile resource consumption. Using the cloud to augment mobile capabilities is still an immature research venue and requires several optimizations to bootstrap mobile services.

From another perspective, researchers propose techniques to fit service-related aspects within constraints and peculiarities of mobile environments, such as semantic reasoning strategies [70], [36], [71]; location-based mobile services [67]; context-aware discovery [88], [89]; incorporating user preferences with the discovery process [75]; device-aware discovery [76]; capacity-driven services [79], [80]; service composition in a mobile environment [71]; adaptive interfaces and web content presentations for mobile devices [90]. It turns out that these approaches still incur lots of overhead on resource-constrained mobile providers. This is due to the fact that these approaches and proposals try to cure the symptoms and not the cause of the problem. In fact, the core problem is that the architectures and technique borrowed from other domains (viz. standard Web service approach) are inefficient in providing reliable and scalable Web services in mobile domains due to the distinct characteristics of mobile environments and the constraints of mobile devices.

Four different approaches proposed in the literature to tackle the performance issue of mobile services are as follows:

- **XML Compression**: Compression of XML messages is one option to boost the mobile Web services performance [91], [92]. However, the performance benefits that compression may bring are compromised by the decompression overhead. Therefore, service developers need to tradoff between bandwidth and computing resources. Most of the proposed XML compression schemes 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 services. Several studies investigated the performance of SOAP-based and REST-based services within resource-constrained environments [18], [22], [16] and results show that RESTful Web services outperform SOAP services.
- **Partitioning**: Partitioning the execution of Web service components is a third direction that has been proposed to hide the limitations of mobile devices. Typically the Web service execution environment encompasses many components to facilitate the hosting and execution of services, such as a request listener, SOAP/XML engine, and encryption and decryption modules. Most of these components are computationally intensive. Deploying all the required components on mobile devices is difficult due to their resource constraints. Asif et al. [93], [94] propose a partitioning technique to execute some of the Web service components on an intermediary 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 scalability. However, Web services must be developed with this concept in mind, as XML elements would have to assignment attributes for each SOAP engine. It is worth noting that this technique is a variation of the proxy-based architecture.
- **Service Replication**: Availability and reliability are major challenges for mobile services due to the intermittent connectivity and limited host capability. Service replication is another proposed option to improve the performance of mobile services. Sheng et al. [37] presents an approach for mobile service replication on idle potential providers. The primary service provider maintains a ready-to-deploy (a bundle that contains all the necessary files) version of the Web service for various mobile and desktop platforms. A Web service manager constantly maintains a pool of potential service hosts and triggers replication once a prespecified performance constraint is violated, such as response time or concurrent invocation

requests exceed a certain threshold.

VII. Summary

Recent years have witnessed a paradigm shift in the role of mobile devices as service providers. The mobile Web service paradigm has emerged due to the successful coupling between the advancements in mobile device manufacturing and the developments of wireless technologies. 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. Mobile services opens up new range of mobile applications that promise advanced mobile computing paradigms, differential user experience, and seamless data access across different platforms. This paper provides the state-of-the-art of mobile services, pointing out enabling technologies and potential applications and bringing forward various challenges and open research issues.

Mobile Web services may be designed following one of two approaches, SOAP-based or REST-based. SOAP-based is an object oriented approach, where operation is the core component, while REST-based is a resource oriented approach, where a resource is the main constituent. Although SOAP services are built on rigid specifications, RESTful approach has been proven the better choice for resource-constrained environments. Several architectures are proposed in the literature for mobile services including, 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 service provisioning. Existing publishing and discovery mechanisms were originally developed for fixed hosting and wired networks. However, mobile environments are dynamic and users constantly change their point of connection to the network. Efficient publishing and discovery mechanisms that are capable of capturing the characteristics and accommodate the constraints of mobile environments are definitely required and crucial to the widespread adoption of mobile services.

In conclusion, although significant steps have been taken towards the realization of reliable mobile services, many research challenges remain open and need further investigation.

References

[1] Portio Research Mobile Factbook 2013, http://www.portioresearch.com/media/3986/Portio\%20Research\%20Mobile\%20Factbook\%202013.pdf. [Accessed: July, 2013].

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

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

[4] S. N. Srirama, M. Jarke, and W. Prinz, "Mobile web service provisioning," in *Tthe Advanced International Conference on Telecommunications andInternational Conference on Internet and Web Applications and Services*, pp. 120–125, 2006.

[5] 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.

[6] S. Srirama, M. Jarke, and W. Prinz, "Mobile host: A feasibility analysis of mobile web," in *The 4th International Workshop on Ubiquitous Mobile Information and Collaboration Systems*, pp. 942–953, 2006.

[7] 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.

[8] 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.

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

[10] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard, "Web services architecture," February 2004. http://www.w3.org/TR/ws-arch. [Accessed: March 2014].

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

[12] 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.

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

[14] 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. [Accessed: March 2014].

[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*, pp. 1–4, 2009.

[16] 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.

[17] 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*, pp. 184 –191, September 2011.

[18] 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.

[19] K. Wagh and R. Thool, "A comparative study of soap vs rest web services provisioning techniques for mobile host," *Journal of Information Engineering and Applications*, vol. 2, pp. 12–17, 2012.

[20] F. Belqasmi, R. Glitho, and C. Fu, "Restful web services for service provisioning in next-generation networks: a survey," *IEEE Communications Magazine*, vol. 49, pp. 66 –73, December 2011.

[21] C. Pautasso, O. Zimmermann, and F. Leymann, "Restful web services vs. "big"' web services: making the right architectural decision," in *The 17th international conference on World Wide Web*, pp. 805–814, 2008.

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

[23] 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. [Accessed: March 2014].

[24] 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 *The 5 th International Semantic Web Conference*, pp. 201–214, 2006.

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

[26] 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.

[27] 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.

[28] 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. [Accessed: March 2014].

[29] 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. [Accessed: March 2014].

[30] 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.w3.org/Submission/SWSF-SWSO/. [Accessed: March 2014].

[31] A. Mathes, "Folksonomies - cooperative classification and communication through shared metadata," 2004. http://www.adammathes.com/academic/computer-mediated-communication/folksonomies.html.

[32] P. Reddy and A. Damodaram, "Web services discovery based on semantic similarity clustering," in *CSI Sixth International Conference on Software Engineering*, 2012.

[33] C. Trabelsi, A. B. Jrad, and S. B. Yahia, "Bridging folksonomies and domain ontologies: Getting out non-taxonomic relations," *The IEEE 12th International Conference on Data Mining Workshops*, pp. 369–379, 2010.

[34] M. Hassan, W. Zhao, and J. Yang, "Provisioning web services from resource constrained mobile devices," in *The IEEE 3rd International Conference on Cloud Computing*, pp. 490–497, 2010.

[35] K. Elgazzar, P. Martin, and H. Hassanein S., "Empowering mobile service provisioning through cloud assistance," in *The 6th IEEE/ACM International Conference on Utility and Cloud Computing*, December 9-12 2013.

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

[37] 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.

[38] 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.

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

[40] Q. Z. Sheng, B. Benatallah, and Z. Maamar, "User-centric services provisioning in wireless environments," *Communications of the ACM*, vol. 51, pp. 130–135, Nov. 2008.

[41] 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*, Springer's LNICST, October 2011.

[42] S. Srirama, V. Łor, E. Vainikko, and M. Jarke, "Supporting mobile web service provisioning with cloud computing," *International Journal On Advances in Internet Technology*, vol. 3, no. 3& 4, pp. 261–273, 2011.

[43] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: Making smartphones last longer with code offload," in *The 8th international conference on Mobile systems, applications, and services*, pp. 49–62, 2010.

[44] The Java ME Platform, http://www.oracle.com/technetwork/java/javame/index.html. [Accessed: March 2014].

[45] N. Balani, "Using kxml to access xml files on j2me devices," 2003. https://www6.software.ibm.com/developerworks/education/wi-kxml/wi-kxml-a4.pdf.

[46] kSOAP2, http://ksoap2.sourceforge.net/. [Accessed: March 2014].

[47] K. Elgazzar, M. Aboelfotoh, P. Martin, and H. S. Hassanein, "Ubiquitous health monitoring using mobile web services," in *The 3rd International Conference on Ambient Systems, Networks and Technologies*, August 2012.

[48] C.-S. Wang and Y.-H. Wang, "Design of an soa-based ubiquitous learning environment," in *The IEEE International Conference on Granular Computing*, pp. 697–702, November 2011.

[49] 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*, (New York, NY, USA), pp. 509–512, ACM, 2010.

[50] D. Brooker, T. Carey, and I. Warren, "Middleware for social networking on mobile devices," in *The Australian Software Engineering Conference*, (Auckland, New zealand), pp. 202–211, 2010.

[51] 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.

[52] Jini Architecture Specification. http://river.apache.org/doc/specs/html/jini-spec.html. [Accessed: March 2014].

[53] 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.

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

[55] JXTA(TM) Community Project, https://jxta.dev.java.net.

[56] K. Elgazzar, H. S. Hassanein, and P. Martin, "Towards personal mobile web services," in *The 2013 IEEE Wireless Communication and Networking Conference (WCNC)*, pp. 4606 – 4611, 7-10 April 2013.

[57] K. Elgazzar, H. S. Hassanein, and P. Martin, "Personal web services: Architecture and design," Technical Report 2012-597, Queen's University, Kingston, ON K7L2N8, October 210. 17 pages.

[58] K. Elgazzar, M. AboElFotoh, P. Martin, and H. S. Hassanein, "Ubiquitous health monitoring using mobile web services," in *The 3rd International Conference on Ambient Systems, Networks and Technologies (ANT)*, pp. 332–339, August 2012.

[59] J. Fuller, M. Krishnan, K. Swenson, and J. Ricker, "Oasis asynchronous service access protocol (asap)," May 2005. http://www.oasis-open.org/committees/documents.php?wg_abbrev=asap. [Accessed: March 2014].

[60] S. Hagemann, C. Letz, and G. Vossen, "Web service discovery - reality check 2.0," in *The 3rd International Conference on Next Generation Web Services Practices*, (Washington, DC, USA), pp. 113–118, IEEE Computer Society, 2007.

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

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

[63] 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.

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

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

[66] N. S. K. Bashah, I. Jørstad, and D. v. Thanh, "Service discovery in future open mobile environments," in *The Fourth International Conference on Digital Society*, (Washington, DC, USA), pp. 47–53, IEEE Computer Society, 2010.

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

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

[69] L. A. Steller, S. Krishnaswamy, and M. M. Gaber, "Cost efficient, adaptive reasoning strategies for pervasive service discovery," in *The International Conference on Pervasive Services*, pp. 11–20, 2009.

[70] T. Gu, Z. Kwok, K. K. Koh, and H. K. Pung, "A mobile framework supporting ontology processing and reasoning," in *The 2nd Workshop on Requirements and Solutions for Pervasive Software Infrastructures*, (Austria), September 2007.

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

[72] 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.

[73] K. Elgazzar, P. Martin, and H. Hassanein S., "Persoalized mobile web service discovery," in *The IEEE 9th World Congress on Services*, July 2013.

[74] K. Elgazzar, H. S. Hassanein, and P. Martin, "Daas: Cloud-based mobile web service discovery," *Pervasive and Mobile Computing*, 2013.

[75] 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.

[76] 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.

[77] 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*, (Berlin, Heidelberg), pp. 33–48, Springer-Verlag, 2009.

[78] L.-H. Vu, M. Hauswirth, and K. Aberer, "Towards P2P-based Semantic Web Service Discovery with QoS Support," in *BPM Workshops, LNCS 3812*, pp. 18–31, Springer Berlin / Heidelberg, 2006.
[79] 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*, pp. 403–410, 2009.
[80] A. Tao and J. Yang, "Context aware differentiated services development with configurable business processes," in *The 11th IEEE International Enterprise Distributed Object Computing Conference*, (Washington, DC, USA), pp. 241–252, IEEE Computer Society, November 2007.
[81] 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.
[82] S. N. Srirama, M. Jarke, and W. Prinz, "Scalable mobile web service discovery in peer to peer networks," in *The 3rd International Conference on Intenet and Web Applications and Services*, pp. 668–674, 2008.
[83] H. Zhu, "Scalability of p2p based mobile web services discovery," Master's thesis, RWTH Aachen University, Germany, 2008.
[84] 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.
[85] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso, "Calling the cloud: Enabling mobile phones as interfaces to cloud applications," vol. 5896 of *Lecture Notes in Computer Science*, pp. 83–102, Springer Berlin / Heidelberg, 2009.
[86] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *Pervasive Computing, IEEE*, vol. 8, no. 4, pp. 14 –23, 2009.
[87] P. Bahl, R. Y. Han, L. E. Li, and M. Satyanarayanan, "Advancing the state of mobile cloud computing," in *Proceedings of the 3rd ACM workshop on Mobile Cloud Computing and Services*, MCS '12, (New York, NY, USA), pp. 21–28, ACM, 2012.
[88] E. Al-Masri and Q. H. Mahmoud, "A context-aware mobile service discovery and selection mechanism using artificial neural networks," in *The 8th international conference on Electronic commerce*, pp. 594–598, 2006.
[89] N. Blefari-Melazzi, E. Casalicchio, and S. Salsano, "Context-aware service discovery in mobile heterogeneous environments," in *The 16th IST Mobile and Wireless Communications Summit*, pp. 1–5, July 2007.
[90] B. Adipat and D. Zhang, "Adaptive and personalized interfaces for mobile web," in *THe 15th Annual Workshop on Information Technolgies & Systems*, pp. 21–26, 2005.
[91] 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.
[92] Y. Natchetoi, H. Wu, and G. Babin, *Euro-Par Parallel Processing*, ch. A Context-Dependent XML Compression Approach to Enable Business Applications on Mobile Devices, pp. 911–920. Springer Berlin Heidelberg, August 2007.
[93] 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.
[94] M. Asif and S. Majumdar, "A graph-based algorithm for partitioning of mobile web services," in *The IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems*, 2009.

Khalid Elgazzar is a postdoctoral research fellow in the School of Computing at Queen's University. Dr. Elgazzar received his PhD from Queen's University in 2013. He has 8+ years of industrial experience in software design and development, and 9+ years of interdisciplinary academic teaching and research. His research interests span the areas of mobile and ubiquitous computing, context-aware systems, mobile services and applications, and elastic networking paradigms to cope with dynamic applications. Dr. Elgazzar has received several recognitions and best paper awards at top international conferences. He leads a team on novel Ubiquitous Cloud paradigms.

Patrick Martin is a Professor of the School of Computing at Queens University. He holds a BSc from the University of Toronto, MSc from Queens University and a PhD from the University of Toronto. He joined Queens University in 1984. He is also a Visiting Scientist with IBM's Centre for Advanced Studies. His research interests include database system performance, Web services, autonomic computing systems and cloud computing.

Hossam Hassanein is a leading authority in the areas of broadband, wireless and mobile networks architecture, protocols, control and performance evaluation. His record spans more than 400 publications in journals, conferences and book chapters, in addition to numerous keynotes and plenary talks in flagship venues. Dr. Hassanein has received several recognition and best papers awards at top international conferences. He is also the founder and director of the Telecommunications Research (TR) Lab at Queen's University School of Computing, with extensive international academic and industrial collaborations. Dr. Hassanein has. He is a senior member of the IEEE, and is currently chair of the IEEE Communication Society Technical Committee on Ad hoc and Sensor Networks (TC AHSN). Dr. Hassanein is an IEEE Communications Society Distinguished Speaker (Distinguished Lecturer 2008-2010).