Mobile Web Service Provisioning and Discovery in Android Days

Satish Narayana Srirama, Carlos Paniagua
Mobile Cloud Lab, Institute of Computer Science, University of Tartu
J. Liivi 2, Tartu, Estonia
{srirama, paniagua}@ut.ee

***Abstract*—Smart phones have pervaded almost all the environments where people perform their day-to-day activities. They are now equipped with embedded sensors, camera, touchscreen, more memory and processing capabilities as well as efficient power saving mechanisms. Together, these improvements enable the mobile devices to perform tasks that are normally run in personal computers like the provisioning of services to other devices. The idea of Mobile Host is not new and this paper tries to extend the research in the domain to the current generation platforms, technologies and standards such as Android OS, Open Services Gateway initiative framework, ZeroConf and Representational State Transfer. While the services provided from Mobile Host like the push notification mechanism and the feasible applications are interesting, for successful adoption of the technology, we need to provide better discovery mechanisms for the offered services. Mobile web service (MWS) discovery is highly dependent on the size of the network and the mobility of the nodes. A peer-to-peer based solution is proposed for the MWS discovery, to avoid the troubles with centralized registries. The approach is discussed thoroughly along with the extensions in semantic discovery support and overlay support discovery mechanism for small networks with low mobility using ZeroConf.**

***Keywords*-Mobile web services, Mobile Host, discovery, Android, performance analysis, ZeroConf.**

I. Introduction

Mobile devices such as smart phones, tablets, etc. have pervaded almost all the environments where people perform their day-to-day activities. Extensive R&D is being carried out in mobile technologies, which led to significant improvements in hardware, software and transmission. Mobile devices are now equipped with embedded sensors, camera, touchscreen, better memory and processing capabilities as well as efficient power saving mechanisms. Moreover, with the release of the Apple iOS and Android OS, developing mobile applications became simpler and this resulted in huge number of applications developed by the community. In addition, transmission rates of mobile networks also have increased, thanks to 3G and 4G technologies as well as ubiquity of WiFi networks, enabling mobiles to access the Internet universally.

Together, these improvements enabled the mobile devices to perform tasks that are normally run in personal computers like the provisioning of services to other devices. For example, context-awareness applications can greatly benefit from mobile web services like location with Global Positioning System (GPS), asynchronous notifications, file sharing, and weather forecast, among others. Similarly, with the advent of cloud computing, the mobile applications also started delegating their resource intensive tasks to cloud services, through their web service interfaces. This confluence between mobile cloud services and mobile web services fosters the creation of more complex and rich applications.

The idea of providing services from mobiles is not new and has been in the ground for some time [refer Table I]. Mobile Host [1], where the mobile device acts as a service provider, enables seamless integration of user specific services to the enterprise by following web service standards, also on the radio link and via resource constrained smart phones. While the feasibility of the Mobile Host has been proven in early developments, the recent improvements and updates in web services domain propose new architectures and protocols for enabling the communication between the clients and Mobile Hosts. For instance, the Representational State Transfer (REST) [2] architecture has emerged as an alternative to the Simple Object Access Protocol (SOAP), enabling the design of web services that focus on system's resources. In addition, the mobile horizon has also changed significantly, since the first implementations of Mobile Hosts have appeared in PersonalJava and J2ME. Symbian is toppled from its market capture of $\approx$90% and today Android takes the place with more than 70% market share, making it perfectly logical to call the contemporary period as Android days.

This paper proposes a new architecture for the Mobile Host, updating it to today's technologies and standards. The Mobile Host for Android is redesigned based on OSGi (Open Services Gateway initiative) framework and is capable of providing services in RESTful fashion using Hypertext Transfer Protocol (HTTP) and Extensible Messaging and Presence Protocol (XMPP). Applications from several domains, such as location based services, m-banking, collaboration and content sharing, context-awareness, social networks and smart environments, benefit from such Mobile Host. Furthermore, Mobile Host is well suited for the consumption and orchestration of mobile cloud services by enabling a scalable and efficient asynchronous communication mechanism for mobile devices and mobile cloud services [3].

Apart from this, the paper also proposes extensions to the mobile web service discovery [4]. Updated Mobile Host supports two discovery mechanisms: a directory-based with overlay support discovery mechanism for large networks with high mobility; and a directory-less with overlay support discovery mechanism for small networks with low mobility.

The paper is organized as follows. Section II provides the related work in the domain. Section III discusses the architecture of the Mobile Host. Section IV introduces two prominent services provided by Mobile Host along with details of developing a new service. Section V later provides a detailed performance analysis of the Mobile Host. Section VI discusses thoroughly the mobile web service discovery mechanism along with its performance latencies, while section VII concludes the paper with a summary.

II. Related work

Provisioning of services from smart phones was studied at RWTH Aachen University since 2003, where some of the first architectures of Mobile Host have appeared [5], [6]. The work addressed the challenges in design and implementation of Mobile Host and presented prototypes and summarized their evaluations. Several implementations of Mobile Hosts in platforms like PersonalJava, J2ME were developed, communicating messages across several protocols like HTTP, reliable UDP etc. The study also dealt with providing proper QoS for mobile web services, in terms of security and scalability, proper discovery approaches for mobile web services and integration frameworks for mobile web services [7]. In addition to this work done at RWTH Aachen University, mobile web service provisioning and discovery have also been addressed by several other researchers across the years. Table I focuses at the most prominent of them all, summarizing their key contributions and extensions to the state of the art.

III. Mobile Host Architecture

Due to the latest enhancements in mobile devices, it is feasible to provide services from smart phones, with reasonable performance latencies. However, several issues still arise when providing web services from smart phones. For instance, from the development perspective the services should be maintainable, easy to install, and most often focus on resources more than in services. In addition, from the provisioning perspective, the mobile nature of smart phones brings challenges such as addressability, reachability and reliability. The upgrades to Mobile Host [10] address these issues relying in several technologies and protocols such as OSGi, ZeroConf and REST.

OSGi framework is a service platform for Java that implements a complete and dynamic component model. Applications in the form of bundles for deployment can be remotely installed, started, stopped, updated, and uninstalled without requiring a reboot. The service registry allows bundles to detect the addition of new services, or the removal of services, and adapt accordingly. The OSGi specifications originally targeted embedded devices and home services gateways, but they are ideally suited for any project interested in the principles of modularity, component-orientation, and service-orientation. Mobile Host for android utilizes Apache Felix, which is a community effort to implement the OSGi R4 Service Platform and other interesting OSGi-related technologies.

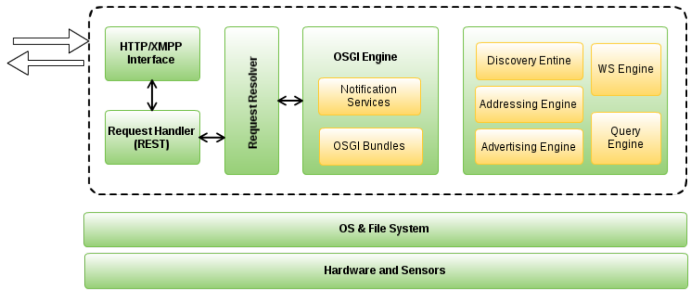

Fig. 1. Architecture of Mobile Host for Android

Similarly, REST and SOAP are two different mechanisms for interacting with web services. REST is the logical option for providing services from mobile devices due to its cacheable and stateless characteristics. Moreover, REST does not rely in heavy XML schemas which overload the network traffic and impact the resource consumption at the devices. Thus, Mobile Host has been upgraded from SOAP to REST fashion.

Mobile Host relies on ZeroConf networks for publishing, discovering and addressing the services, explained in detail in section VI. The architecture of the Mobile Host for Android is shown in the Figure 1.

Mobile Host listens for HTTP or XMPP connections. When a connection is received the HTTP/XMPP interface passes the connection to the Request Handler which parses the request and its parameters. As the Mobile Host for Android is designed to follow a RESTful philosophy, the key resources (entities, collections, services, etc.) are identified by their own URIs. The standard methods are mapped to resource-specific semantics and all the resources implement the same uniform interface. In this way, for example: the picture ”logo1.png” requested by the client can be accessed through a GET HTTP request to the URI ”/logos/logo1.png”. Once the REST handler parses the request to know the resource requested by the client, the request is passed to the Request Resolver which communicates with the OSGI engine for resolving the request.

The Request Resolver gets an instance of the service which is running in the OSGi Engine. As mentioned before, the OSGI Framework runs a pool of services, named OSGI Services, which can be managed remotely and easily deployed. Each OSGI Service deployed in the OSGI Engine implements a Java Interface enforcing the service to handle the HTTP methods such as GET, POST, DELETE, etc. Once the Request Resolver acquires the service instance, it invokes the method corresponding to the HTTP requests. For example, if the HTTP Request was sent as a GET then the Request Resolver invokes the *doGet* method of the service and the control is passed to the OSGI Service. The OSGI Service implements the service logic such as retrieving pictures, contacts, GPS location, etc. and prepares the response. The response can be in any format (XML, JavaScript Object Notation (JSON), plain text, etc.) or a mime type based on the logic of the service. Finally, the OSGI Service writes the response to the client socket according to the HTTP or XMPP protocol.

TABLE I
RELATED WORK OF MOBILE WEB SERVICE PROVISIONING AND DISCOVERY

Year	Authors	Description
2003	Yang et al. [8]	Proposes an infrastructure for organizing and efficiently accessing mobile web services (MWS) in broadcast environments that defines a multi-channel model to carry information about mobile web services. Discusses the preliminary MWS discovery still depending on centralized UDDI.
	Pratistha et al. [9]	Proposes an infrastructure that provides the capability of hosting web services on mobile devices.
2004	Srirama [5], Srirama et al. [1], [10]	Master thesis [5] which studied mobile web service provisioning in detail. Provides a prototype implementation in PersonalJava with fesibile applications, a detailed performace and scalability analysis of the Mobile Host, and addressing mechanisms for the Mobile Host in radio networks of that day like GPRS and HSCSD. Consolidated results are published in 2006 [1], [10].
2005	Gehlen and Pham [6]	Propose an application framework simulating a mobile P2P environment.
2006	van Halteran and Pawar [11]	Propose a proxy based solution for providing services from mobiles, in Jini technology
	Srirama [12], Srirama et al. [4]	Proposes MWS discovery mechanism exploiting P2P networks, especially JXTA. A detailed analysis of the discovery mechanism is addressed in 2008 [4].
	Srirama et al. [13], [14]	Proposes a mediation framework for integrating MWS scalability and discovery solutions. MWSMF was demonstrated with detailed performance analysis in 2007 [14].
	Dustdar and Treiber [15]	Propose a distributed P2P web service registry solution based on lightweight web service profiles.
2007	Srirama et al. [16]	Studied the feasibility of providing secured web services from smart phones.
	Asif et al. [17]	Proposes a lightweight Web service provider toolkit, which supports the security in embedded environment.
	Aijaz et al. [18]	Handles asynchronous requests of Mobile Hosts that support the execution of long-lived services.
	Doulkeridis et al. [19]	Work involved a semantic model that structures the service direction/discovery recommendation based on both client and server-side context.
2008	Srirama [7]	PhD thesis summarized the complete research in mobile web service provisioning domain. Addressed the QoS issues in terms of security and scalability, mobile web service discovery and mobile web service mediation framework. Also provided applications in domains like mlearning, healthcare etc.
2009	Kim and Lee [20]	Propose a lightweight framework that hosts web services on mobile devices and supports service migration.
	Steller et al. [21]	Proposes the mTableaux algorithm to optimize the reasoning process and facilitate web services discovery.
2010	AlShahwan and Moessner [22]	Compared SOAP and REST for providing services from smart phones.
2011	Flores et al. [23]	Proposes MCM which eases invocation of cloud services from mobiles, increasing the scope of offered services from Mobile Host. MCM in turn uses Mobile Host for asynchronous communication with devices.
	Asif and Majumdar [24]	Discuss partitioning frameworks for mobile web service provisioning.
	Chang et al. [25]	Introduces context awareness into mobile web service provisioning domain.
2012	Paniagua [3]	Master thesis which extended the architecture of Mobile Host to REST and OSGi frameworks. It also extended the MWS discovery with semantics support and overlay discovery mechanism for small networks with low mobility using ZeroConf. This work is the main basis for the current paper.
	Chang et al. [26]	Proposes a mobile device-hosted service-oriented workflow-based mediation framework for mobile social network in proximity by delegating some of the discovery activities to the cloud.

IV. WEB SERVICES PROVIDED BY MOBILE HOST

Mobile Host aids in the development of the next generation of context-awareness and ubiquitous applications. The implementation of Mobile Host for Android introduces, as samples, a Location (GPS) Data Provisioning Service and a Push Notification Service. Both services can be widely used for ubiquitous and context-awareness scenarios. In addition, the section also describes the procedure of developing custom services for the Mobile Host.

A. Location (GPS) Data Provisioning Service (Location Information Service)

This web service provides the location information of the mobile device by using the embedded GPS device. The GPS is a worldwide satellite-based radio navigation system developed by the Department of Defense which consists of 31 operational satellites. GPS provides two levels of service, Standard Positioning Service and the Precise Positioning Service. The Precise Positioning Service is a highly accurate US military positioning, velocity and timing service. The Standard Positioning Service is a positioning and timing service which is available to all GPS users on a continuous, worldwide basis with no direct charge. However, this standard positioning service lacks the accuracy when the receiver device is indoors. As an alternative, smart phones can also determine the device location by using Wi-Fi networks.

The Location Information Service developed for Mobile Host returns the latitude and longitude of the device, in JSON format. There are several scenarios that can benefit from such service. For instance, we can envision a pro-active service for recommendations. Typically, recommendation services such as foursquare [27] are reactive, it means the mobile user should ask for recommendations to the provider and send its location along with the recommendation request. With the GPS service in Mobile Host, the provider can shift to a proactive approach, asking the device for its current location and then push the recommendations to the device.

B. Push Notification Service

The current generation mobile applications provide more sophisticated functionality to the user but are generally resource intensive. Therefore, mobile technologies are looking into the emerging cloud computing domain to satisfy the increasing demand for processing power, storage space and energy. While offloading the resource intensive tasks to cloud is interesting, it is not a straight forward task. To help with the procedure, we have developed Mobile Cloud Middleware (MCM) [23], that eases the invocation of services provided by multiple cloud providers from the smart phones.

Generally, this cloud based processing is very time consuming. For example we have developed the CroudSTag [28] application, which takes as input, a set of pictures and videos collected at an event like a conference, that are stored on a cloud. The application later processes these pictures/videos to identify the people in them, recognizes them on social networking cites like facebook and later tries to form a social group with the identified people. The application uses several third party services and the face recognition on the cloud. CroudSTag takes approximately 35 minutes to process a three-minute video recorded with a mobile phone in high definition. Such waiting time is not tolerable from the user and mobile application usability perspective. Moreover, the operating system may kill the process when it is running out of memory (Android case). So such applications need an asynchronous notification mechanism for the mobile devices.

Today, several mobile application providers and platform vendors offer the push notification services to the respective clients, like the Google Cloud Messaging for Android (GCM) and Apple Push Notification Service (APNS). They are generally based on XMPP or Push Proxy Gateways (PPG) standard from Open Mobile Alliance (OMA). Most often, the quality of service (QoS) offered by some of these providers is low to the extent that they cannot be used in real-time applications, as they are public services and user is competing with several others [3]. However, with the Mobile Host, the notification mechanism can be exposed from the smart phone as a service, and the messages can be sent directly to the device, without having the necessity for a proxy or a third party service provider.

C. Developing Custom Services for Mobile Host

Apart from the above mentioned services, the architecture of Mobile Host, makes it easy to develop new services. The OSGI Services are deployed as bundles and each bundle needs to be registered in the OSGI engine to make the service available for invocation. During the registration process, the OSGI Service provides the OSGI Engine, information about itself such as the name of service. This name is later used by the Request Resolver for invocation purposes. From a developer perspective, an OSGI Service needs to implement two Java Interfaces, BundleActivator and SroidService. The BundleActivator contains the methods required to start, to stop and to register the service in the OSGI Engine. Similarly, SroidService contains the methods required for the service provisioning. The SroidService interface enforces the provisioning of the services in a REST fashion and guarantees that the OSGI Services are capable of handling the HTTP methods GET, PUT, DELETE and POST. In this interface, the method doCreate can be used for any variable or process initialization.

For handling requests and responses, Mobile Host provides two Java Classes, SroidRequest and SroidResponse respectively. SroidRequest encapsulates the logic for parsing the HTTP request and maps the parameters, if any, to a HashMap which is accessible from the OSGI Services. Similarly, SroidResponse encapsulates the logic for writing

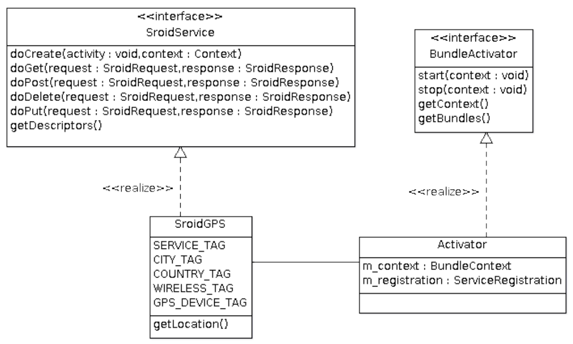
<<interface>>
SroidService
doCreate(activity : void,context : Context)
doGet(request : SroidRequest,response : SroidResponse)
doPost(request : SroidRequest,response : SroidResponse)
doDelete(request : SroidRequest,response : SroidResponse)
doPut(request : SroidRequest,response : SroidResponse)
getDescriptors()
<<interface>>
BundleActivator
start(context : void)
stop(context : void)
getContext()
getBundles()
<<realize>>
<<realize>>
SroidGPS
SERVICE_TAG
CITY_TAG
COUNTRY_TAG
WIRELESS_TAG
GPS_DEVICE_TAG
getLocation()
Activator
m_context : BundleContext
m_registration : ServiceRegistration

Fig. 2. Class diagram for a service provided by Mobile Host

information to the socket previously established by the client with Mobile Host. The Class diagram shown in Figure 2 describes the classes and relationships needed to create a mobile web service for Mobile Host in Android.

The OSGi services can be deployed as Java Archive (JAR) files containing the implementation of both interfaces, Activator and SroidService. When Mobile Host starts, it takes each JAR file from the folder SroidServices, in the Android file system, and deploys the services in the OSGI Engine. Once the services are registered in the OSGI Engine they are ready to be invoked.

V. Performance Analysis of the Mobile Host

The Mobile Host was also analyzed for its performance and here we discuss the analysis with the File Browsing Service. This web service provides access to files such as pictures and documents in the public folder of the mobile device. Following the REST philosophy, each file is a resource and can be accessed through its corresponding URL. The experiments consider a GET request for downloading a picture of 1.5MB size. Tsung, a load testing tool, was used for conducting the experiments. A Samsung Galaxy SII phone [29] was used for the analysis. The phone has a Dual-Core 1.2GHz Cortes-A9 CPU, 1GB of RAM and 16GB for storage. The services and applications were developed based on the Android platform, compatible with Android 2.2 API or higher. The experiments were conducted under Wifi and 3G networks. So, the tests were taken in a wifi network with upload rate of $\approx$ 4393 kbps and download rate of $\approx$ 5648 kbps, respectively and a 3G network with upload rate of $\approx$ 1487 kbps and download rate of $\approx$ 4597 kbps.

From Figure 3 we can observe that Mobile Host for Android can handle $\approx$ 20 concurrent users with a response time of $\approx$ 31 seconds (in the worst case) in Wi-Fi networks, and $\approx$ 6 concurrent connections with a response time of $\approx$ 30 seconds in 3G networks. In this case we are assuming that a waiting time of 30 seconds is reasonable from the user perspective. One should observe that the timestamps are for the slowest invocation in a set of n concurrent requests. Similarly, assuming that the user is willing to wait 60 seconds for the

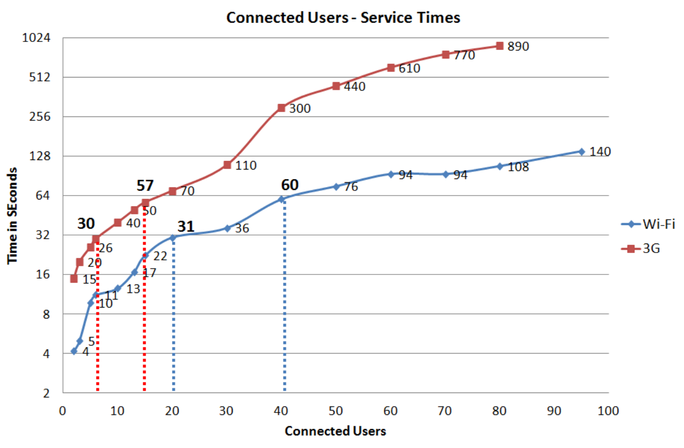

Fig. 3. Simultaneous connections handled by Mobile Host

results, Mobile Host is capable of handling ≈ 40 concurrent connections in Wi-Fi and ≈ 15 connections in 3G. Comparing these results with those of the PersonalJava based Mobile Host [10], actually show how far the capabilities of today's mobile devices and networks have advanced.

VI. Mobile Web Service Discovery Mechanism

From the performance analysis of the Mobile Host we can assume a commercial Mobile Web Enterprise with Mobile Hosts and mobile web service clients and with each Mobile Host providing some services for the Internet. However, in this case we need to provide better discovery mechanisms for the provided services. Generally web services are published by advertising WSDL (Web Services Description Language) descriptions in a UDDI (Universal Description, Discovery and Integration) registry. But with huge number of services possible with Mobile Hosts, a centralized solution is not the best idea, as they can have bottlenecks and can introduce single points of failure. Besides, mobile networks are quite dynamic due to the node movement. Devices can join or leave network at any time and can switch from one operator to another operator. This makes the binding information in the WSDL documents, inappropriate. Hence the services are to be republished every time the Mobile Host changes the network. This pushes the necessity for a dynamic service discovery mechanism for the mobile web services [4].

Mian et al. [30] characterized the service discovery protocols on the basis of the size of the network and mobility of the nodes. Mian defines three levels of mobility (low, medium and high) and three sizes of networks (small, medium and large). Small networks (up to 10 nodes) with low mobility (up to 5 km per hour, such as within a building) bring different requirements than a large network (greater than 100 nodes) with high mobility (greater than 50 km per hour, such as highway traffic speeds). For example, in large networks and low mobility, overlay networks provide an efficient routing mechanism for searching. Moreover, the low mobility does not bring any issues in terms of maintenance. However, if the mobility turns to be high, the maintenance of the overlay networks becomes an issue. Another example is the small networks with high mobility where the maintenance of directory service is unfeasible and the overlay networks are not necessary. Mobile Host supports two discovery mechanisms: a directory-based with overlay support discovery mechanism for large networks with high mobility (Peer to Peer (P2P) based); and a directory-less with overlay support discovery mechanism for small networks with low mobility.

A. *Mobile Host Directory-less with Overlay Support Discovery Mechanism for Mobile Ad Hoc Networks (MANET)*

MANETs are self-configuring infrastructure less networks of mobile devices which are connected by wireless links. Each device in a MANET can move independently in any direction, and therefore changes its links with other devices periodically. Moreover, each device forwards the traffic unrelated to its own use acting as a router in the network. These networks can be connected to a larger link such as the Internet. However, several challenges arise in this type of networks such as addressing, naming, and service discovery. Most often, MANETs are networks of small size with medium or low mobility.

Mobile Host uses ZeroConf for exposing itself and its services to external devices in MANETs. ZeroConf dynamically configures the host in the network assigning them an IP address and also a domain name. Furthermore, ZeroConf provides a mechanism for service discovery and domain resolution. Network users no longer have to assign IP addresses, assign host names, or even type in names to access services on the network. Users simply query what network services are available, and decide from the list. Applications can also automatically detect services they need or other applications they can interact with, allowing automatic connection, communication, and data exchange. Mobile Host uses JmDNS, a service discovery protocol which is an implementation of ZeroConf for Android devices. JmDNS assigns a local domain name to Mobile Host which can be used by other devices to access the services exposed by the host. JmDNS is also totally compatible with implementations of ZeroConf for other platforms such as Bonjour for Apple.

ZeroConf tackles the Addressing issue by self-assigned link-local addressing. This addressing approach uses a range of addresses reserved for the local network, typically a small LAN or a single LAN segment. The self-assigned addressing simply picks a random IP address in the link-local range and tests it. If the address is not already used, the device is assigned with that name, otherwise it picks another address and checks again whether the name is in use or not. Furthermore, the name-to-address translation capabilities of ZeroConf rely on Multicast DNS (mDNS). In Multicast DNS the DNS-format queries are sent over the local network using IP multicast, therefore no single DNS server with global knowledge is required to answer the queries. Each service or device can provide its own DNS capabilities in such a way that if a query is received asking for its own name the device provides a DNS response with its own address. This name-to-address translation mechanism requires the service names to be unique

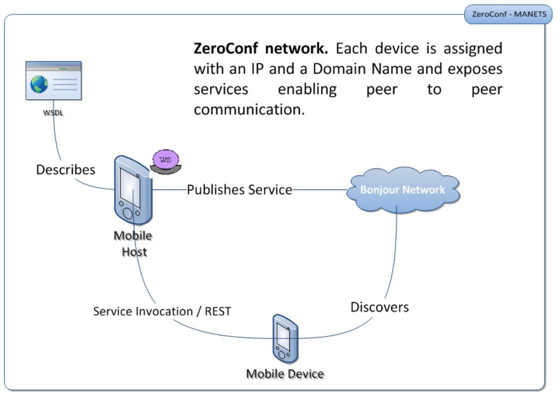
ZeroConf - MANETS
ZeroConf network. Each device is assigned with an IP and a Domain Name and exposes services enabling peer to peer communication.
WSDL
Describes
Publishes Service
Bonjour Network
Mobile Host
Service Invocation / REST
Discovers
Mobile Device

Fig. 4. Discovery Mechanism in Mobile Ad-hoc Networks

in the network. Implementations of ZeroConf such as JmDNS and Bonjour automatically rename the service in case of collisions. The service name convention follows a DNS-format which includes the service type, transport protocol, and the domain where the service is exposed. In case of MANETs the domain happens to be ”local.” and the type should follow the DNS SRV (RFC 2782) Service Types standards [31]. For example, the service name $\_http.\_tcp.local.$ corresponds to a Mobile Host service of type $http$, for web services, provided through tcp protocol in the $local$ network.

The final element of ZeroConf is the service discovery. This service discovery enables other devices to find all the available instances of a particular type of service and to maintain the service directory. Once a device has discovered a service, it resolves the service name to an IP address and port number which can be used later on for the service invocation. The service directory provides a layer for indirection between a service and its current IP address and port. Furthermore, this indirection enables the application to keep a persistent list of available services and resolve an actual network address just prior to using a service. The service directory can be relocated dynamically with low network traffic penalties. This dynamical updating capability of the service directory addresses the mobility of the devices which can enter and leave the MANETs at any time. The service discovery is accomplished by sending an mDNS query for a given type of domain. Later, all the matching services reply with their names. The service names received are listed in the local service directory.

ZeroConf is a service-oriented discovery mechanism. The devices query for service, not for host providing the services. The service directory stores service names instead of addresses. If, due to the mobility of smart phones, the IP address, port number, or host name changes, the devices can still invoke the mobile web services. Figure 4 illustrates the discovery mechanism in MANETs.

B. P2P based Mobile Web Service Discovery

Discovery of mobile web services is studied for a while and earlier implementations of Mobile Host relied on JXTA networks for advertising, indexing and addressing the services

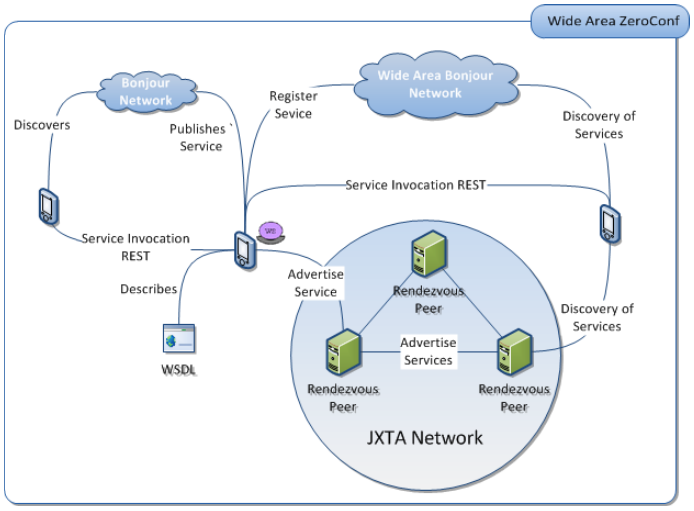

Fig. 5. Discovery Mechanism in Wide Area Networks

provided by the device [4]. This study updates the mechanism to the latest Android developments and extends it by adding ZeroConf support. In this P2P approach, a virtual P2P network is established, and then the services deployed on Mobile Host are published as JXTA advertisements containing the WSDL information of the service. Advertisements are language-neutral metadata structures represented as XML documents. Peers discover each other, the resources available in the network and the services provided by peers and peer groups, by searching for their corresponding advertisements. Peers, especially rendezvous peers, may cache any of the discovered advertisements locally. Every advertisement exists with a lifetime that specifies the availability of that resource. Lifetimes give the opportunity to control out-of-date resources without the need for any centralized control mechanism. To extend the life time of an advertisement, the advertisements are to be republished.

The services advertised in the mobile P2P network can be discovered by using the keyword-based search mechanism provided by JXTA, which also extends to the WSDL level. However, this search happens to be basic and thus returns a large number of services that match the keyword, which is not suitable for smart phones. To tackle this problem, the basic JXTA search is improved with Solr [32], an open source search platform based on Apache Lucene Project [33]. Solr prepares an index of the data and supports full-text search, hit highlighting, faceted search, dynamic clustering etc.

This mechanism shows reasonable performance and time responses. We observed that the highest time response (≈ 3.01 seconds) occurs under 3G networks. However, the time responses under Wi-Fi networks are below 0.4 seconds. The experiment considered a mobile client application in Android querying an index of 1000 services in Solr. However, this mechanism still lacks the accuracy in the results. For example, when querying for $notification$ services in $android$, the results included all the services that contain at least one of the keywords, thus returning a large list of services that include services for android, not necessarily notification services, and also notification services for other OS. Solr provides a high performance keyword-based service search but lacks semantics capabilities to enrich the meaning of the results.

To introduce semantics into the discovery, we developed Semantic Search Engine (SSE), on a machine also acting as rendezvous peer, which considers Jena [34] and OWL-S API [35] to support the Service Ontology and to resolve the queries for services from the clients. Jena provides a collection of tools and Java libraries for developing semantic web and linked-data apps, tools and servers. Similarly, OWL-S API provides a Java API for programmatic access to create, read, write, and execute OWL-S described atomic as well as composite services.

C. Complete Publishing and Discovery Process

When a Mobile Host joins the network, it advertises its services first into the local ZeroConf network. The services are later registered with the JXTA service directory in the form of advertisements and are included into the Mobile Web Service Ontology maintained by the SSE.

When a client needs to discover a service, it sends the query through the local network via mDNS to other peers and to the rendezvous peer. The request contains the required service, context information of the client such as geographical location, weather, bandwidth connection, if it is connected through Wi-Fi or 3G, among other, user preferences such as whether to automatically download the content or to be notified about the service availability, the maximum size of downloaded content, etc. It is logical that the services which are in the same radio-link are more relevant for mobiles and are discovered faster than those services in the global network. Consequently, the client receives first a list of services published via ZeroConf in the local radio link, using the text-based search provided by the framework.

Later the discovery is extended to the services in the wide area network. The search is handled by the rendezvous peer on behalf of the mobile. The rendezvous peer receives the search request and passes it to the SSE. The SSE keeps an ontology of a large number of services registered in the set of rendezvous peers available in the network. After the query is received, first the services are filtered using the keyword search feature provided by Solr which returns a reduced, but still large, list of services (e.g. 50 out of 1000 services). Later, the SSE performs a SPARQL query over the OWL-S description of the services returned by Solr, retrieving the services that semantically match the client's preferences and needs.

However, semantic applications are resource demanding and time consuming. So the result set is sent asynchronously to the device, along with the WSDL of each service, using Mobile Host based push notification. The results are then appended to the list already obtained by ZeroConf search. The WSDL of each service lets the mobile device to know about the service invocation details such as the end point, port, and parameters. The invocation process relies on the wide area bonjour and the client application does a simple HTTP request to the corresponding DNS/IP Address and port, as shown in figure 5.

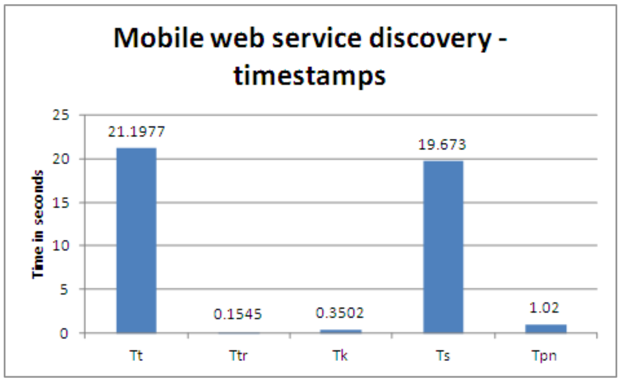

Fig. 6. Mobile web service discovery - timestamps (in seconds)

D. Performance of the Mobile Web Service (MWS) Discovery

In MWS discovery, the total service query time T_t is:

$$T_t \cong T_{tr} + T_k + \sum_{i=1}^{n}(T_{to_i}) + T_q + T_{pn} \tag{1}$$

Where, T_{tr} is the transmission time taken across the radio link for the service query delegation between the mobile phone and the SSE. The value includes the time taken to transmit the request and the time taken to send the acknowledgment back to the mobile. Apart from these values, several parameters also affect the transmission delays like the TCP packet loss, TCP acknowledgements, TCP congestion control etc. So a true estimate of the transmission delays is not always possible. Alternatively, one can take the values several times and can consider the mean values for the analysis. T_k is the time taken for the keyword text search by Solr. T_s is the sum of the time taken by the SSE to load each OWL-S description in the list of services returned by Solr ($\sum_{i=1}^{n}(T_{to_i})$) plus the time taken by the reasoner to query the ontology T_q. $\cong$ is considered in the equation as there may be other timestamps involved, like the client processing at the mobile phone. T_{pn}, represents the notification time, which is the time taken to send the response of the mobile service discovery to the device.

From Figure 6, it can be observed that the mechanism retrieves the list of services in $\approx$ 21 seconds (Note: mobile already got results from local Zeroconf network within a second). Most of this time is consumed by the reasoner when querying ontology (T_s) after the services are first filtered by Solr (T_k), and a little fraction of the total response time (T_t) is spent in the network transmission (T_{tr}) and the asynchronous notification (T_{pn}). The test used a pool of 1000 service advertisements in the JXTA network. Solr retrieved 50 services out of the 1000 services presented in the index and the SSE returned 5 services out of those 50. Thus the MWS discovery, improves the quality of the results and returns a smaller set fitting better to the mobile devices' screen.

The results also give a way to a very interesting discussion. Solr presents short time responses but low quality results while the semantic service discovery improves the quality of the service search but with some performance penalties. The decision can be left to the mobile user or the domain expert who may design an application on top of Mobile Host and such MWS discovery mechanism. Moreover, T_s can be reduced significantly by vertical scaling the capacity of SSE with much powerful servers. However, improvising the performance of semantic discovery is beyond the scope of this research.

VII. Conclusions

The improved capabilities of today's mobile devices make them feasible to provide services to other devices. Mobile Host is upgraded to Android, the architecture considered OSGi for service management, and ZeroConf for publishing the services. The clients can access the MWS via HTTP or XMPP protocol using REST philosophy in which services are abstracted as resources. The paper introduces prominent services provided by Mobile Host like the push notification, along with details of developing new services.

However, for successful adoption of the Mobile Host in the mobile web enterprise, we need to provide better discovery mechanisms for the offered services. MWS discovery is highly dependent on the size of the network and the mobility of the nodes. A P2P based solution is proposed for the MWS discovery, to avoid the troubles with centralized registries. The approach is discussed thoroughly along with the extensions in semantic discovery support and overlay support discovery mechanism for small networks with low mobility using ZeroConf. The detailed performance analysis of the Mobile Host and the MWS discovery show that the mechanisms can be adapted with reasonable performance latencies on mobile devices, paving the way for the mobile web enterprise.

Acknowledgment

This work is supported by European Regional Development Fund through EXCS, Estonian Science Foundation grant ETF9287 and Target Funding theme SF0180008s12.

References

[1] S. N. Srirama, M. Jarke, and W. Prinz, "Mobile web service provisioning," in *Int. Conf. on Internet and Web Applications and Services (ICIW'06)*. IEEE, 2006, pp. 120–125.

[2] R. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, 2000.

[3] C. Paniagua, "Discovery and push notification mechanisms for mobile cloud services," Master's thesis, University of Tartu, 2012.

[4] S. Srirama, M. Jarke, H. Zhu, and W. Prinz, "Scalable mobile web service discovery in peer to peer networks," in *Int. Conf. on Internet and Web Applications and Services*. IEEE, 2008, pp. 668–674.

[5] S. Srirama, "Concept, implementation and performance testing of a mobile web service provider for smart phones," Master's thesis, RWTH Aachen, Germany, 2004.

[6] G. Gehlen and L. Pham, "Mobile web services for peer-to-peer applications," in *Consumer Communications and Networking Conference, 2005. CCNC. 2005 Second IEEE*. IEEE, 2005, pp. 427–433.

[7] S. N. Srirama, "Mobile hosts in enterprise service integration," Ph.D. dissertation, RWTH Aachen University, 2008.

[8] X. Yang, A. Bouguettaya, B. Medjahed, H. Long, and W. He, "Organizing and accessing web services on air," *IEEE transactions on systems, man, and cybernetics - part a: systems and humans*, vol. 33, no. 6, pp. 742–757, November 2003.

[9] D. Pratistha, N. Nicoloudis, and S. Cuce, "A micro-services framework on mobile devices," in *International Conference on Web Services, Nevada, USA*, 2003.

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

[11] A. van Halteren and P. Pawar, "Mobile Service Platform: A Middleware for Nomadic Mobile Service Provisioning," in *Int.l Conf. On Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2006.

[12] S. Srirama, "Publishing and discovery of mobile web services in peer to peer networks," in *Proceedings of First International Workshop on Mobile Services and Personalized Environments (MSPE'06)*, vol. P-102. Lecture Notes in Informatics, GI, November 2006, pp. 15–28.

[13] S. N. Srirama, M. Jarke, and W. Prinz, "A mediation framework for mobile web service provisioning," in *2006 Middleware for Web Services (MWS 2006) Workshop @ 10th IEEE International Enterprise Distributed Object Computing Conference*. IEEE, 2006, p. 14.

[14] ——, "Mobile web services mediation framework," in *Middleware for Service Oriented Computing (MW4SOC) Workshop @ 8th Int. Middleware Conf. 2007*. ACM Press, 2007.

[15] S. Dustdar and M. Treiber, "Integration of transient web services into a virtual peer to peer web service registry," *Distributed and Parallel Databases*, vol. 20, pp. 91–115, 2006.

[16] S. Srirama, M. Jarke, and W. Prinz, "Security analysis of mobile web service provisioning," *International Journal of Internet Technology and Secured Transactions (IJITST)*, vol. 1(1/2), pp. 151–171, 2007.

[17] M. Asif, S. Majumdar, and R. Dragnea, "Hosting web services on resource constrained devices," *Web Services, IEEE International Conference on*, vol. 0, pp. 583–590, 2007.

[18] F. Aijaz, B. Hameed, and B. Walke, "Towards peer-to-peer long-lived mobile web services," in *Innovations in Information Technology, 2007. IIT'07. 4th International Conference on*. IEEE, 2007, pp. 571–575.

[19] C. Doulkeridis, V. Zafeiris, K. Norvåg *et al.*, "Context-based caching and routing for P2P web service discovery," *Distributed and Parallel Databases*, vol. 21, no. 1, pp. 59–84, 2007.

[20] Y. Kim and K. Lee, "A lightweight framework for mobile web services," *Computer Science-Research and Development*, vol. 24, no. 4, pp. 199–209, 2009.

[21] L. A. Steller, S. Krishnaswamy, and M. M. Gaber, "Cost efficient, adaptive reasoning strategies for pervasive service discovery," in *Int. conf. on Pervasive services (ICPS '09)*. ACM, 2009, pp. 11–20.

[22] F. AlShahwan and K. Moessner, "Providing soap web services and restful web services from mobile hosts," in *Int. Conf. on Internet and Web Applications and Services*. IEEE, 2010, pp. 174–179.

[23] H. Flores, S. Srirama, and C. Paniagua, "A Generic Middleware Framework for Handling Process Intensive Hybrid Cloud Services from Mobiles," in *The 9th International Conference on Advances in Mobile Computing & Multimedia (MoMM-2011)*. ACM, 2011, pp. 87–95.

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

[25] C. Chang, S. Ling, and S. Krishnaswamy, "Promws: Proactive mobile web service provision using context-awarenes," in *8th IEEE Int. Wrksp. on Managing Ubiquitous Communications and Services*, 2011.

[26] C. Chang, S. N. Srirama, and S. Ling, "An adaptive mediation framework for mobile p2p social content sharing," *Service-Oriented Computing*, pp. 374–388, 2012.

[27] foursquare Inc., https://foursquare.com/.

[28] S. N. Srirama, C. Paniagua, and H. Flores, "Social group formation with mobile cloud services," *Service Oriented Computing and Applications*, vol. 6, no. 4, pp. 351–362, 2012.

[29] Samsung Electronics, "Samsung Galaxy SII," http://www.samsung.com/global/ microsite/galaxys2/html/feature.html.

[30] A. Mian, R. Baldoni, and R. Beraldi, "A survey of service discovery protocols in multihop mobile ad hoc networks," *Pervasive Computing, IEEE*, vol. 8, no. 1, pp. 66–74, 2009.

[31] A. Gulbrandsen, "A dns rr for specifying the location of services (dns srv)," 2000.

[32] Apache, "Apache solr," http://lucene.apache.org/solr/.

[33] ——, "Apache lucene," http://lucene.apache.org/core/.

[34] B. McBride, "Jena: A semantic web toolkit," *Internet Computing, IEEE*, vol. 6, no. 6, pp. 55–59, 2002.

[35] D. Martin, M. Burstein, J. Hobbs *et al.*, "Owl-s: Semantic markup for web services," *W3C Member submission*, vol. 22, 2004.