Mobile Service Platform: A Middleware for Nomadic Mobile Service Provisioning

Aart van Halteren, Pravin Pawar, *University of Twente, The Netherlands.*

***Abstract*— Ongoing miniaturization and power efficiency of mobile devices have led to widespread availability of devices that have an increasing amount of processing power and storage, and that support multiple wireless network interfaces connecting to various auxiliary devices and to the Internet. It is now feasible for a mobile device to host services and participate in a service discovery network. Roaming of a mobile device from one wireless network to another entails nomadic characteristics to the hosted services. We denote this class of services as Nomadic Mobile Services.**

This paper discusses the requirements for Nomadic Mobile Service provisioning and proposes the Mobile Service Platform (MSP) as a supporting infrastructure and middleware which extends the Service Oriented Architecture paradigm to the mobile device. The MSP design is based on the Jini Surrogate Architecture Specification which enables devices that can not directly participate in a Jini Network to join a Jini network with the aid of a third party. MSP consists of an HTTPInterconnect protocol to meet the specifications of Jini Surrogate Architecture and provides a custom set of APIs to develop and deploy a Nomadic Mobile Service. This paper also presents case studies of MSP enabled services in diverse domains such as healthcare, robotics and positioning services. To conclude, we outline the need for a context-aware MSP.

***Index Terms*— Service Oriented Architecture, Nomadic Mobile Service, Mobile Service Platform, Mobile Middleware**

I. Introduction

A Service Oriented Architecture (SOA) is essentially a collection of services that communicate with each other to achieve a common goal. The SOA paradigm includes advertising, discovery and utilization of diverse services by means of service directories. The principal components of the SOA consist of a service, service description, service advertising and discovery and artifacts [14]. A service is a contractually defined behavior that can be implemented and provided by a component for use by another component. The service description consists of the technical parameters, constraints and policies that define the terms to invoke the service. The service advertises its service description for potential clients. A client interested to access the service obtains information about the existence of a service, its applicable parameters and terms through service discovery. An artifact specifies the associated data model for the service (such as XML schemas and web-service descriptions) to which a client should bind for using the service. A service provider may make an entry into the service directory to reference the artifact and explain how to bind to it. The clients may retrieve this information and use it to bind to the artifacts [14].

Nowadays, mobile devices have become an integral part of daily life. These devices are characterized by higher processing power, lower costs and the ability to connect to the Internet using a wireless network. Mobile devices often get equipped with multiple network interfaces including Infrared, Bluetooth, Wi-Fi, GSM, UMTS. This enables these devices to support multiple auxiliary devices (e.g. camera, robots). Furthermore, the applications running on a mobile device can provide context information (e.g., the computational and communication environment of the mobile device, positioning information) which is of interest in the area of context aware computing [4]. The SOA paradigm can be extended to the mobile device to model these auxiliary devices and applications as services. Such services provide the flexibility of allowing a mobile device to participate in the service discovery network and provide these services to the clients located anywhere in the Internet. Herewith, we name this class of services as *Nomadic Mobile Services*. A Nomadic Mobile Service is hosted by the mobile host such as a handheld device, mobile phone or any type of embedded device capable of connecting to the Internet using a wireless network. The mobile device roams from one mobile communication service to another which gives nomadic characteristics to the services it hosts.

This paper discusses the requirements for Nomadic Mobile Service provisioning and proposes the *Mobile Service Platform* (MSP) as a supporting infrastructure which extends the SOA paradigm to the mobile device. MSP is a middleware that facilitates the development and deployment of innovative services on the mobile device for clients located anywhere in the Internet. The MSP design is based on the Jini Surrogate Architecture Specification which enables a device which can not directly participate in a Jini Network to join a Jini network with the aid of a third party [21]. MSP consists of an *HTTPInterconnect* protocol to meet the specifications of the Jini Surrogate Architecture and provides a custom set of APIs for building and running services on a mobile device. Using

The authors are with Architecture and Services of Network Applications group, Department of EEMCS, University of Twente, The Netherlands (e-mail: {A.T.vanHalteren, P.Pawar}@utwente.nl, Phone: +31(0)534895466, Fax: +31(0)534895477).

This work is supported by Freeband Awareness project under grant BSIK5902390 (To be presented in WIMOB 2006, Montreal, Canada).

MSP, a service hosted on a mobile device can participate as a Jini service in the Jini network. This paper also presents case studies of MSP enabled services in diverse domains such as healthcare, robotics and location based services.

Section II of the paper lists the requirements for Nomadic Mobile Service provisioning. Section III introduces the Mobile Service Platform (MSP), how it addresses the requirements presented in Section II and a life cycle of an MSP enabled Nomadic Mobile Service. Section IV elaborates the implementation details of MSP. Section V presents case studies of Nomadic Mobile Services prototyped using MSP. Section VI of the paper discusses the related work. Section VII concludes the paper and outlines the need for a context-aware MSP.

II. Requirements for Nomadic Mobile Service Provisioning

A service in the fixed network relies on certain features provided by the underlying infrastructure for its advertising, discovery and utilization. These features include fixed location and IP address, sufficiently powerful servers, a certain level of guaranteed bandwidth, reliable connectivity and standardized protocols. Protocols such as RMI, CORBA IIOP or SOAP facilitate communication between the service and a client. Services are advertised and discovered using protocols such as JINI, UPnP, UDDI or SLP. However, the communication and computational environment of a mobile device does not always provide these features. A mobile device and its surrounding environment is characterized by reduced processing power, limited bandwidth, and limited storage capabilities as compared to their counterparts in the fixed network [7]. This section discusses the requirements for Nomadic Mobile Service provisioning in such a resource constrained environment.

A. *Seamless communication between service and client*

Protocols such as RMI, CORBA IIOP or SOAP necessary for communication between the service and client use an IP address or DNS name of the server hosting a service. A Nomadic Mobile Service is not fixed and changes its location resulting in variable IP address assignments to the mobile device. 2.5/3G wireless network operators typically assign an IP address to the mobile device dynamically from the private address space at the establishment of the wireless connection (e.g. GPRS) [17]. *Network Address Translation* (NAT) inhibits connection from the public Internet to host behind a NAT router. When a mobile device enters a network without a DHCP server, it may assign itself an IP address using *Automatic Private IP Addressing* (APIPA) [11]. However this address is not known globally. Consequently, connecting to the mobile device using its IP address becomes practically impossible. Assigning a permanent IP address to the mobile device and ensuring its reachability using Mobile IP is a promising solution. However, it is subject to the availability of IP addresses and price that the network operator charges. Mobile devices are able to connect to the Internet using multiple network interfaces including GSM, UMTS, and WLAN. While roaming from one network to another, every network assigns different IP address to the mobile device. It is required that the service running on the mobile device must be reachable and should be able to support a heterogeneous network environment as discussed above.

B. *Consideration of execution environment limitations*

Traditionally, a mobile device acts as a client for the service hosted in the fixed network. Hosting a service inverts the role of mobile device from ‘Client’ to ‘Server’. This inversion of the role implies new challenges that have to be considered for Nomadic Mobile Service provisioning [6]. For the 2.5G and 3G networks, the bandwidth available for the upload is an order of magnitude lower than that for download. This requires that the size of a reply to the request from a client should be optimized. Compression algorithms may be used, but limited processing power of the mobile device should be taken into account.

The intermittent network connectivity and limited battery capacity lead to uncertain lifetime of the service affecting its availability. Services hosted in the fixed network are generally capable of supporting bulky operations to fulfill user requirements. For example, with the Google Web APIs service, developers can query billions of web pages directly from their computer programs [8]. It is practically impossible to store a large amount of data locally and process it. Such kind of processing is required for services which collect certain data from the auxiliary devices, collect behavioral statistics of a mobile user over a prolonged time. For the services in the fixed network, security verifications to determine access rights to certain data is taken care of by the AAA infrastructure or by the application itself. However, for the mobile devices, security verification is computationally intensive and demands additional bandwidth [3]. Nomadic Mobile Service provisioning should take into account limitations of the execution environment.

C. *Scalability*

The fixed network can offer scalable services as communication and computational resources to a certain degree can be added effortlessly. This ensures that a potentially large number of services and clients can be supported. However, for the Nomadic Mobile Service, the number of clients simultaneously accessing the service depends on the available bandwidth and processing power of the mobile device. Offering a scalable service is of significant importance for a real-time service such as the one which transmits a patient’s vital signs data to a healthcare center [9]. Hosting Nomadic Mobile Services requires that the potential number of services hosted, service code size and memory utilization must stay within certain limits while serving the client requests.

III. MSP Enabled Nomadic Mobile Service

A Nomadic Mobile Service realized using Mobile Service Platform is composed of two components: 1) a service running on the mobile device (referred to as a *device service*); and 2) a representation of the device service in the fixed network which is referred to as a *surrogate*. Fig. 1 shows these components of a Nomadic Mobile Service.

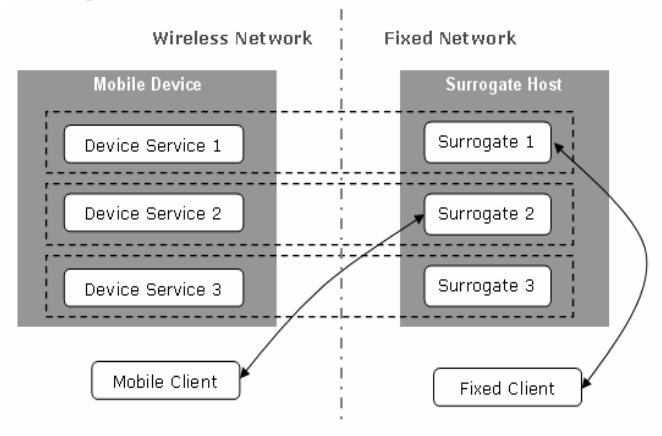

Fig. 1. Elements of a Nomadic Mobile Service

The device service communicates with its surrogate that is hosted by the *surrogate host*. The surrogate functions as a proxy for the device service. Both, the client in the wireless network as well as in the fixed network need to communicate with the surrogate to access the service. A mobile device can host multiple device services whereas each device service has its corresponding surrogate residing at the surrogate host.

Using a surrogate in the fixed network as a representation of device service in the wireless network has certain advantages. Due to the distributed implementation of a Nomadic Mobile Service, it is possible to offload a device service by caching/storing data at its surrogate. For the service demanding intensive computations, the processing can be divided between the device service and surrogate. The surrogate can be implemented as a Java RMI service, a CORBA service, or a Web service. By introducing the surrogate in the fixed network, it is feasible to serve a larger number of clients as compared to the wireless network [See Section II on requirements]. The clients are largely unaware of the fact that the environment in which the real service resides is resource constrained. A surrogate host can also perform essential security functions to provide secure services to the client.

However, splitting a Nomadic Mobile Service into a device service and surrogate also introduces a state synchronization problem. The surrogate must be aware of the change in the state of a device service. As well, the surrogate should be reachable by the clients as long as the device service is available to the surrogate. The Mobile Service Platform supports the communication between the device service and surrogate. The design of MSP and lifecycle of a Nomadic Mobile Service is presented in the Section II-A. Section II-B briefly explains the implementation of MSP.

A. *Lifecycle of Nomadic Mobile Service*

The MSP design is based on Jini technology. The Jini Surrogate Architecture Specification [21] defines the requirements on the communication between a device service and surrogate. The surrogate participates in a fixed network as a Jini service [20]. The device service runs on a mobile device and communicates with the surrogate host in a fixed network using an *Interconnect* protocol. An Interconnect is the logical and physical connection between the surrogate host and a device and is defined as a part of Jini Surrogate Architecture Specification [21]. Although [21] does not provide details for any specific Interconnect implementation, it requires an Interconnect protocol to fulfill at least three features: *device discovery*, *surrogate upload* and *keep-alive*. We have developed an HTTP implementation of the Interconnect protocol for MSP which fulfills these features as well as supports the exchange of messages between a device service and surrogate. HTTP has been specifically chosen for the implementation of the Interconnect protocol because more and more devices, such as mobile phones and PDAs have out-of-the-box HTTP support. Moreover, ISPs and 2.5/3G wireless network infrastructure also support connectivity to the Internet using HTTP. Fig. 2 shows the elements of MSP. The stages during the lifecycle of a Nomadic Mobile Service (as shown by 1 to 6 in Fig. 2) are:

1) *Stage 1: Registration of a device service at a surrogate host*

Fig. 3 shows the interactions during the registration of device service at a surrogate host. These interactions include the following:

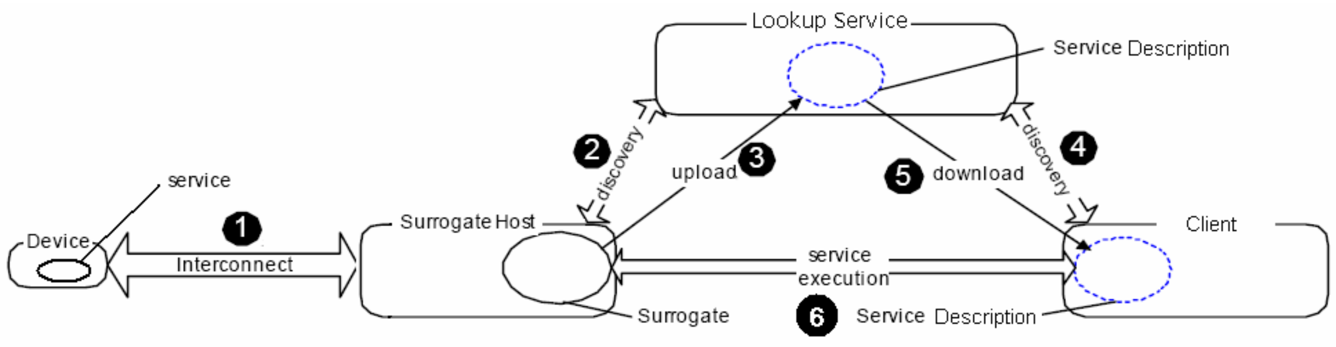

Fig. 2. The elements of Mobile Service Platform

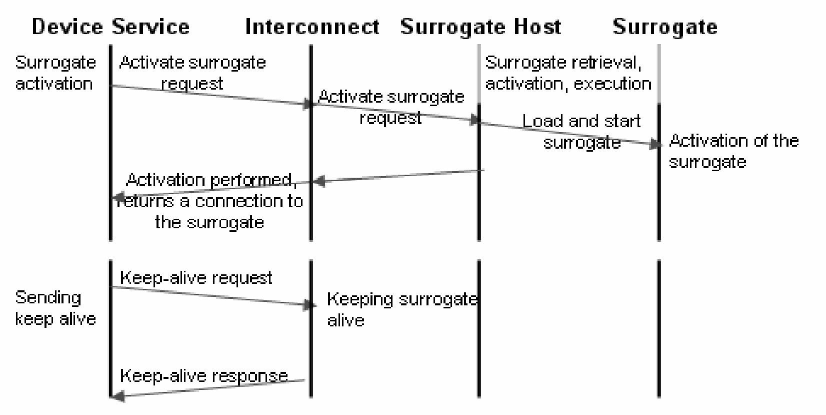
Device Service
Interconnect
Surrogate Host
Surrogate
Surrogate activation
Activate surrogate request
Surrogate retrieval, activation, execution
Activate surrogate request
Load and start surrogate
Activation of the surrogate
Activation performed, returns a connection to the surrogate
Keep-alive request
Sending keep alive
Keeping surrogate alive
Keep-alive response

Fig. 3: Surrogate loading, activation, execution and keep-alive

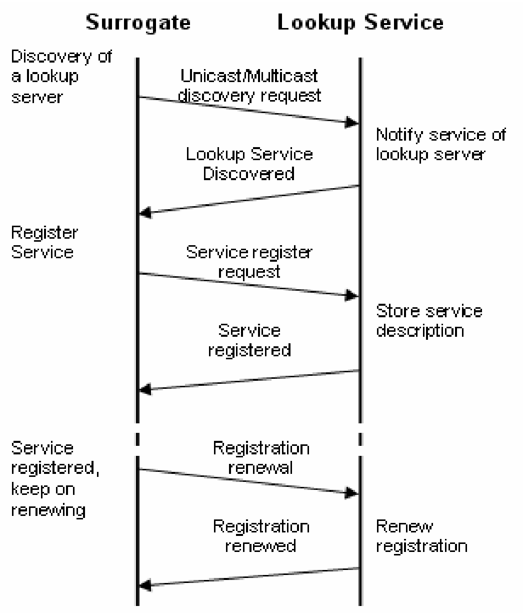
Surrogate
Lookup Service
Discovery of a lookup server
Unicast/Multicast discovery request
Notify service of lookup server
Lookup Service Discovered
Register Service
Service register request
Store service description
Service registered
Service registered, keep on renewing
Registration renewal
Registration renewed
Renew registration

Fig. 4: Lookup service discovery, service registration and renewal

Device discovery: The purpose of the device discovery mechanism (stage 1 in Fig. 2) is to make a surrogate host aware of the existence of a device hosting a Nomadic Mobile Service and vice versa. Once a device and the surrogate host have discovered each other, the device can register with the surrogate host. The device needs to authenticate itself with the surrogate host prior to the registration. This authentication is required to prevent the registration of malicious devices.

Surrogate upload: After the device service is started, the surrogate host must be provided with the surrogate. The device itself can upload the surrogate or it can send to the surrogate host the location from where the surrogate can be downloaded. Once the surrogate is downloaded, the surrogate host loads and starts the surrogate.

Keep-Alive: The keep-alive mechanism is used to inform the surrogate host that the device service is still active and connected. If the device service cannot confirm its online status after a certain time period (i.e., the surrogate host did not receive a keep-alive message in time), the surrogate host will deactivate the surrogate of corresponding device service.

Device service and surrogate interactions: Once a device service and its surrogate are active, they can interact with each other. MSP defines three types of interactions between the device service and surrogate, which are as follows:

One-Way messaging: The One-Way messaging allows for unconfirmed message delivery between the device service and its surrogate. This kind of message does not have a corresponding reply.

Request-Response messaging: The Request-Response messaging supports reliable message delivery. The request message must have a corresponding reply message. An example of this message is the *keep-alive* message, which is sent by the device service at fixed intervals and the surrogate host acknowledges this message by sending a response.

Streaming: Streaming supports exchange of continuous data (streams) between the device service and surrogate. For

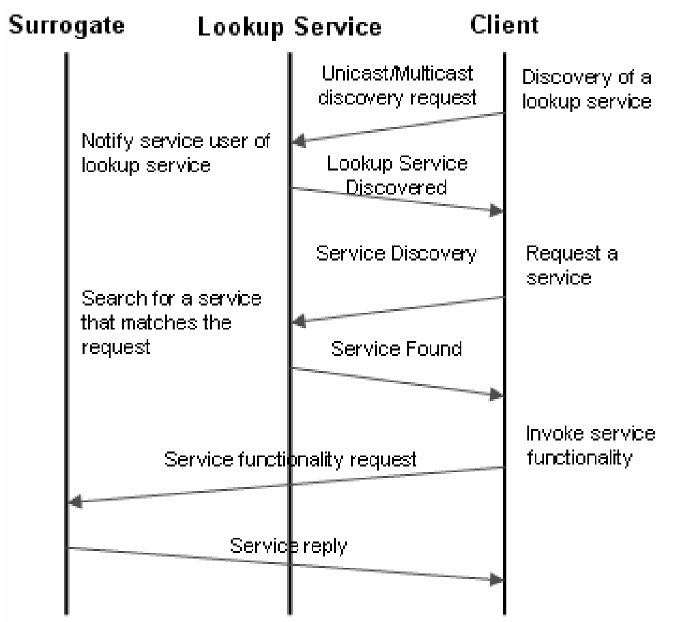
Surrogate
Lookup Service
Client
Unicast/Multicast discovery request
Discovery of a lookup service
Notify service user of lookup service
Lookup Service Discovered
Service Discovery
Request a service
Search for a service that matches the request
Service Found
Invoke service functionality
Service functionality request
Service reply

Fig. 5: Lookup Service discovery, service description download and service invocation

example, a mobile device continually receives the vital signs of the patient and streams these to the surrogate for (near) real-time delivery to the healthcare center [6].

2) *Stage 2 and 3: Lookup service discovery, service registration and renewal*

Once the surrogate is activated, it may contact the Jini lookup service [20] for service registration (stage 2). After the lookup service is discovered either through unicast or multicast discovery, the device service description is registered (stage 3) with the lookup service. The surrogate needs to periodically renew the service registration failing to which the lookup service will discard the registration. These interactions specific to Jini technology are shown in Fig. 4.

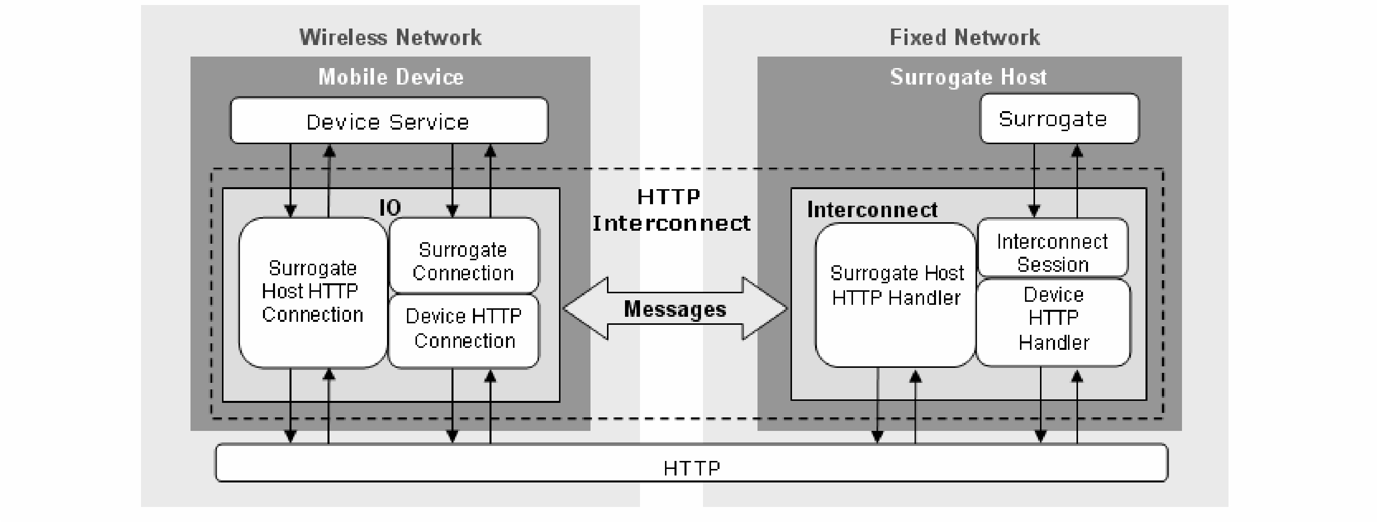

Fig. 6. HTTPInterconnect implementation

3) Stage 4, 5 and 6: Lookup service discovery, service description download, service invocation

After discovering a lookup service a client can request a suitable service by providing the desired service attributes (Stage 4). If the requested service is registered with the lookup service, the client receives a service description (Stage 5). The client uses the service according to its service description (Stage 6). Fig. 5 shows the communication between the client, lookup service and surrogate. These interactions are also specific to Jini technology [20].

B. MSP Implementation

We identify two main parts in the implementation of an MSP enabled Nomadic Mobile Service. The first part consists of the device service, surrogate and interactions between them. The second part consists of surrogate, lookup service, client and interactions between them. HTTPInterconnect, as described in Section III-A, is an HTTP implementation of the Interconnect protocol and it is responsible for communication between the device service and its surrogate. The device service is usually implemented using J2ME technology, which provides an optimized Java runtime environment for resource constrained devices. MSP provides a set of custom APIs necessary for managing service lifecycle on a mobile device, interactions between device service and surrogate and administering these interactions [13]. The second part of MSP implementation is based on Jini service provisioning [20].

1) HTTP Interconnect

The HTTPInterconnect implementation consists of three packages: *Messages*, *IO* and *Interconnect*. Fig. 6 shows the elements of HTTPInterconnect.

a) Messages

The messages package defines the structure of messages exchanged between the device service and the surrogate, via the interconnect protocol. This package contains functionality for encoding and decoding data that can be sent in these messages. The message starts with a *serviceId*, followed by an *operationId* and a *sequenceId*. The *serviceId* is the ID of the service that sent the message (or the ID of the service the message is destined for, when sending from the surrogate to the device service). The *operationId* is a unique ID that is given to the operation of the service or the surrogate. Each operation offered by the service to its surrogate and vice versa, need to have a unique ID, so each message can trigger a certain operation. The *sequenceId* identifies the order of the messages.

The body of a message contains data specific to the operation to be performed by the message. A nomadic positioning service may, for example, send temporary position updates to the surrogate. The body of these update messages can contain the new position of the device.

b) IO and Interconnect

The IO package contains the part of HTTPInterconnect implementation that resides on the mobile device. This package handles all the messages sent to and from the device service. The key parts of the IO package are *SurrogateHostHTTPConnection*, *SurrogateConnection* and *DeviceHTTPConnection*. The Interconnect package contains the part of the HTTPInterconnect implementation that exists in the surrogate host. This package handles the activation and deactivation of surrogates and all the messages sent to and from the surrogate. The essential parts of the Interconnect package are *SurrogateHostHTTPHandler*, *InterconnectSession* and *DeviceHTTPHandler*.

On start up, the surrogate host initiates the *SurrogateHostHTTPHandler* to handle the requests received by the surrogate host. When a device service is started, it sends a registration request to the *SurrogateHostHTTPConnection*. This class represents the connection to the surrogate host. If there are no device services registered before, on receiving the registration request, the surrogate host creates a *DeviceHTTPHandler* to handle requests from the mobile device hosting the service. The surrogate host later returns a *DeviceHTTPConnection* to the mobile device. Once the *DeviceHTTPConnection* is available, the sevice service sends a *Surrogate Activation* request to the *DeviceHTTPHandler*. The surrogate and the *DeviceHTTPHandler* share an *InterconnectSession* which is created by the surrogate host as a part of surrogate activation. Via the *InterconnectSession*, the surrogate can interact with the device service. If the activation request succeeds, the *DeviceHTTPConnection* returns a *SurrogateConnection* to the device service. This *SurrogateConnection* represents a connection to the surrogate and it allows the device service to interact with its surrogate.

The *DeviceHTTPConnection* receives the reply for these messages (in case of Request-Response messaging) and determines the corresponding device service for the message. A *DeviceHTTPConnection* can support up to 255 surrogate connections, thus allowing 255 device services to run on a mobile device.

When the device service is stopped, the surrogate host needs to be notified so it deactivates the surrogate. The device service requests *Register Surrogate* operation via the *SurrogateHostHTTPConnection*. This request is forwarded to the *DeviceHTTPConnection*. The *DeviceHTTPConnection* creates a deregister message and sends it to the surrogate host. The surrogate is deactivated and a confirmation is notified back. As a part of the surrogate deactivation, the *InterconnectSession* that existed between the *DeviceContextHandler* and the surrogate is terminated. The *DeviceHTTPConnection* removes the corresponding *SurrogateConnection*. If the terminated surrogate is the only surrogate for this device, the *DeviceHTTPConnection* is also closed.

To address the problem of low bandwidth and high latency in the 2.5G and 3G mobile networks, HTTPInterconnect features a number of optimizations. Along with the provision for One-Way messaging [See Section III-A], a number of messages for the same *DeviceContext* may be combined in one HTTP request. Additional improvement can be achieved with HTTP chunking [11], where the messages are conveyed as chunks of one long-term HTTP request. The other optimization is achieved by implementing a deflate compression algorithm [5] for the messages sent between mobile device and surrogate host. For secure communication, the messages can also be transmitted using HTTPS.

The message from the surrogate to a device service is piggybacked in an HTTP Response to the HTTP Request from the device. This solves the communication problem between the client and service as the device service keeps the connection alive for the response. Routers forward response messages to the correct address. These response messages are used to deliver data to the service that is behind NAT. To facilitate the response messages, the device periodically sends request messages to the surrogate host.

2) Publishing and utilizing service

On execution, the device service connects to the surrogate host via the HTTPInterconnect and requests the activation of its surrogate. The surrogate host downloads the byte code for the surrogate from a location (e.g. URL) that is provided by the device service during registration. MSP currently uses Madison as an implementation of the Surrogate host designed and implemented by Sun [22]. Madison offers an interface for device discovery, activation and keep-alive management which is implemented by our HTTPInterconnect protocol.

In MSP, the surrogate can join the Jini network [20] and has access to Jini services. The surrogate discovers the Jini lookup service and registers a service proxy with the lookup service. The service proxy implements all the interfaces of a service and contains the logic to communicate with the surrogate. It is also downloadable from the network. MSP does not specify any communication protocol between a Jini client and a surrogate. A service proxy and corresponding surrogate can make use of any communication protocol such as RMI, CORBA IIOP, or SOAP depending on the requirements.

The Jini client discovers a lookup service and requests a desired service either by the service interface or description (as discussed in the Section III-A.3). If this type of service is registered with the lookup service, a service proxy will be returned to the client. The client instantiates a service proxy to utilize the service.

IV. Mobile Service Platform Case Studies

This section presents the Nomadic Mobile Services prototyped using MSP in diverse domains. MSP has been made available publicly [13]. By going through the presented case studies, developers can get a general idea how to make use of MSP for innovative Nomadic Mobile Services.

A. Healthcare Domain: Tele-Monitoring of the Patient

The MSP has been originally designed, implemented, tested and verified for the MobiHealth tele-monitoring service as a part of MobiHealth project [9]. The real life trials of the mobile health applications developed in MobiHealth project have been successful [19][1] for the patients with ventricular arrhythmia, respiratory inefficiency, and high risk pregnancies. The purpose of the tele-monitoring service is to gather patient's data collected from medical sensors attached on the patient's body and to deliver this data in near real-time to the healthcare professionals. An important aspect of the tele-monitoring service is that a patient may reside at home or office performing daily tasks and is at the same time under a doctor's supervision (Fig. 7).

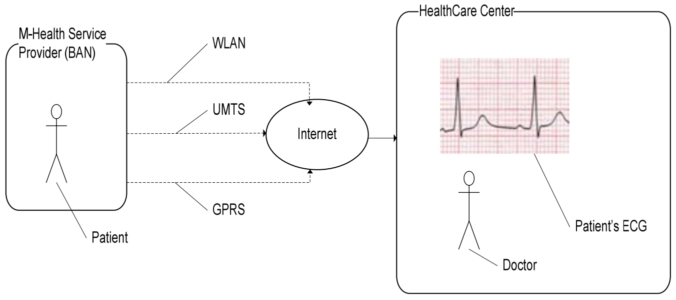

Fig. 7: Concepts of m-Health monitoring

To facilitate the acquisition process of medical data, each patient is equipped with a Body Area Network (BAN), where the BAN sensor system attached to patient's body collects vital sign data such as heart rate, blood pressure or ECG signals. MSP facilitates the vital signs delivery through public network infrastructures such as GPRS, UMTS, WLAN networks or any other type of network infrastructure that supports Internet access.

The tele-monitoring device service communicates with the BAN sensor system using Bluetooth connectivity. The tele-monitoring device service and in turn its surrogate can instruct the BAN sensor system to sample certain signals at a certain sampling frequency [10]. As per the instruction, the BAN sensor system streams the real-time sampled signals to the tele-monitoring device service using the optimized TMSI fiber protocol [2]. The tele-monitoring device service sends these signals to its surrogate. The tele-monitoring surrogate makes these signals available to the interested clients which are as follows:

- BAN Data Repository (BDR): Storage of BAN data.
- Alarm Service (AS): Detection of a critical patient condition based on the received signals and delivery of an SMS message to the mobile phone of a healthcare professional in case of emergency.
- BAN Streaming Service (BSS): Near real-time delivery of BAN data to a healthcare professional.
- Healthcare center: The doctors at the healthcare center can view the graphical representation of the patient's vital signs as well as instruct the BAN to sample interested data at certain available frequencies. It is possible to detect whether a BAN is active or not and whether a BAN is capable or not of delivering patient measurements.

The MobiHealth functionality relies on an XML schema for the configuration of a BAN and exposing sensor data to the end user applications [16].

B. Positioning Services Domain: Position of a Mobile Host

The nomadic positioning service provides the current position of a mobile device in a privacy sensitive way using MSP and positioning hardware. Such a service may be used as a standalone service or as a building block for context aware applications interested in the positioning information of a mobile host.

The positioning service uses the Place Lab library that determines a position of the mobile host by spotting beacons, such as WiFi access points, GSM cell phone towers, fixed Bluetooth beacons or other sensors. These beacons have a unique or semi-unique id (e.g. their MAC address). When a service detects one or more beacons, the position of the device can be determined based on the location of beacons [24]. The algorithm to calculate the position of a mobile device can be executed at the surrogate host or at the mobile device depending on the user preferences.

The positioning service is a device service. It is represented by positioning surrogate in the fixed network. The client subscribes to the location change event with the positioning surrogate. Whenever the positioning service detects a change in the location of mobile host, it sends the location change event to the positioning surrogate, which is later sent to the interested clients. Further details on the nomadic positioning service such as performance and accuracy measurements, and implementation details can be obtained from [24].

C. Robotics Domain: Controlling LEGO MindStorm robots

The Nomadic Robot Service [23] provides the capability to a client to control a Lego robot as desired. The robot service is a device service and the robot surrogate is the representation of that service in the fixed network. The robot service communicates with the LEGO Mindstorm robot using infrared connectivity. This service has different functionality requirements than those described in Section IV-A and IV-B as the robot needs to move in real-time as soon as the client issues a request.

V. Related Work

Hosting services on a mobile device is an emerging research area. This section describes the efforts in this direction.

A lightweight infrastructure referred to as Micro-Services capable of hosting web services from the mobile devices has been proposed in [15]. A Micro-Service is a subset of complex web services architecture and is specifically adapted for resource scarce devices. A Micro-Service is partitioned into three distinct components that include the Compact Listener, Core Server and Supporting Modules. The Compact Listener is the highest level component that is responsible for managing client requests received on a particular port. The Core Server receives encapsulated HTTP requests from the Compact Listener. It then performs the necessary validations and determines the appropriate Supporting Modules to forward these requests to. Lastly, the Supporting Module represents the implementation of a particular Internet protocol (i.e. HTML, SOAP and others [15]).

Micro-Services does not include the security verification to determine the access rights to service data [15]. It is limited to performing simple and short operations and does not adapt to the intermittent bandwidth characteristic of the wireless medium. It is not clear how addressing issues, such as NAT traversal, have been dealt with. The details of how a wireless service provider environment uses this framework are not provided.

The Mobile Web Server project under development by MAGION [12] aims to deliver a product prototype of the Mobile Web Services Framework, comprising a set of standard web services, which can be deployed on a mobile satellite user terminal. The standard services include network performance testing services, web cam service, GPS positioning service. Additional web services can be created within the framework, using specific APIs for tight integration with the satellite modem and other devices deployed.

MAGION employs the concept of a GateKeeper to perform essential security checks as well as caching data for the performance improvement. The GateKeeper is placed on the terrestrial Internet, and acts as the sole gateway to the Mobile Server. The GateKeeper may serve multiple Mobile Servers. The project targets mobile satellite terminals. It would be interesting to explore its suitability for other network service standards such as UMTS, GPRS, CDMA, and WLAN.

VI. Conclusions and Future Work

The SOA paradigm can be extended to mobile devices to model auxiliary devices and other applications running on a mobile device as a Nomadic Mobile Service. The Mobile Service Platform (MSP) addresses the challenges involved in Nomadic Mobile Service provisioning by means of the surrogate architecture. MSP is a middleware which facilitates the development and deployment of services on the mobile device for clients located anywhere in the Internet. MSP provides a custom set of APIs to build and run services on a mobile device. The MSP has been demonstrated successfully for healthcare services, nomadic positioning services and nomadic robot services.

Future work focuses on incorporating context awareness in the MSP. Context-aware computing is a paradigm closely related to mobile computing [4]. Context-aware software adapts according to various context elements such as the location of use, the collection of nearby people, hosts and accessible devices, as well as to changes to such things over time [18]. A system with these capabilities can examine the computing environment and react to changes in the environment.

It is desired that Nomadic Mobile Services and MSP should adapt themselves according to the change in context information. For example, for the tele-monitoring service described in Section IV-A when Mr. Janssen, an epileptic patient roams from a UMTS or WLAN communication service to GPRS, it is desired that the set of transmitted vital signs must be reduced to accommodate the transmission capacity of the GPRS communication service. Similarly, when the mobile device connects to an available WLAN; the full set of vital signs must be automatically transmitted to and displayed in the healthcare center.

The context aware MSP will be able to provide the best possible quality of service to the clients even under varying computing and communication contexts. We are approaching the challenges of the context aware nomadic service provisioning in our current research on the next version of the Mobile Service Platform.

Acknowledgement

The authors are thankful to Dr. Cristian Hesselman and Ricardo Neisse for reviewing the paper and providing valuable feedback. The authors also appreciate the contribution of Emiel Schoot Uiterkamp, Peter van Tol and Ivar Pruijn in the development of MSP and MSP enabled Nomadic Mobile Services.

References

[1] Alonso, A., "Detailed Description of Trial Scenarios", D1.3, MobiHealth project (http://www.mobihealth.org), October 2002.

[2] Broens, T., "Bandwidth Optimization of the TMSI fiber protocol", Internship report, University of Twente, Enschede, the Netherlands, 2003.

[3] Burnside M., Clarke D., Mills T. et. al, "Proxy-based security protocols in networked mobile devices", Proceedings of the 2002 ACM symposium on Applied computing, Pages 265-272, 2002, Madrid, Spain.

[4] Chen, G., Kotz, D., "A Survey of Context-Aware Mobile Computing Research", Technical Report TR 2000-381, Dept. of Computer Science, Dartmouth College, 2000.

[5] Deutsch P., "DEFLATE compressed data format specification", Request for Comments No 1951, Network Working Group, May 1996.

[6] Dokovsky, N., Halteren, A. V., Widya, I. "BANip: enabling remote healthcare monitoring with Body Area Networks", International Workshop on Scientific Engineering of Distributed Java Applications, 27-28 November, 2003; Luxemburg.

[7] Foreman, G., Zahorjan, J., "The Challenges of Mobile Computing", IEEE Computer, pages 38-47, April 1994.

[8] http://www.google.com/apis/index.html

[9] Konstantas, D., Bults, R., Herzog, R., "MobiHealth: Innovative 2.5/3G Mobile Services and Applications for Healthcare", 11th IST Mobile and Wireless Telecommunications Summit, June 2002; Thessaloniki, Greece.

[10] Konstantas, D., Bults, R., Wac, K., Halteren, A. V., "Final, Exploitation Ready MobiHealth BAN", D2.6, MobiHealth project (http://www.mobihealth.org), April 2004.

[11] Kozierok, C., "The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference", No Starch Press, 2005.

[12] Magion, "Mobile Web Services Framework", http://telecom.esa.int/telecom/www/object/index.cfm?fobjectid=12852, March 2005.

[13] "Mobile Service Platform – Developer's Resources", ASNA Group, University of Twente, http://janus.cs.utwente.nl:8000/twiki/bin/view/MSP/Developers, 2005; The Netherlands.

[14] Nickull D., "Service Oriented Architecture Whitepaper", Adobe Systems Inc., 2005.

[15] Pratistha, D., Nicoloudis, N., Cuce, S., "A Micro-Services Framework on Mobile Devices", International Conference on Web Services, 2003; Nevada, USA.

[16] Pruijn, I., "Web Services in the MobiHealth Service Platform", Bachelor Assignment Report, Architecture and Solutions of Network Applications Group, University of Twente, July 2004; The Netherlands.

[17] Rekhter, et al., "Address Allocation for Private Internets", RFC 1918, February 1996.

[18] Schilit, B., Adams, N., Want, R., "Context-aware Computing Applications", Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, pages 85-90, December 1994; Santa Cruz, California.

[19] Scully, T., "Overall Evaluation of the Mobihealth Trials and Services", D5.1, MobiHealth project (http://www.mobihealth.org), October 2003.

[20] Sun Microsystems, "The JINI Architecture Specification", http://www.sun.com/software/JINI/specs/ JINI1_2.pdf, December 2001.

[21] Sun Microsystems, "JINI Technology Surrogate Architecture Specification", http://surrogate.JINI.org/sa.pdf, October 2003.

[22] Sun Microsystems, "Design Document for Madison - A Contributed Surrogate Host Implementation", http://ipsurrogate.jini.org/specs.html, 2003.

[23] Tol P. V., "Service discovery of Lego Mindstorms based nomadic services", MasterThesis, Architecture and Solutions of Network Applications Group, University of Twente, December 2005; The Netherlands, http://asna.ewi.utwente.nl/

[24] Uiterkamp, E. S., "Nomadic Positioning Services for a Mobile Service Platform", Master Thesis, Architecture and Solutions of Network Applications Group, University of Twente, August 2005; The Netherlands, http://asna.ewi.utwente.nl/