Supporting Mobile Web Service Provisioning with Cloud Computing

Satish Narayana Srirama, Vladimir Šor and Eero Vainikko
Distributed Systems group
Institute of Computer Science, University of Tartu
J. Liivi 2, Tartu, Estonia
{srirama, eero, volli}@ut.ee

Matthias Jarke
Information Systems and Databases Group
RWTH Aachen University
Ahornstr. 55, 52056 Aachen, Germany
jarke@dbis.rwth-aachen.de

***Abstract*—Web services are going mobile. A Mobile Enterprise can be established in a cellular network by participating Mobile Hosts, which act as web service providers, and their clients. Mobile Hosts enable 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. However, establishing such a Mobile Enterprise poses several technical challenges, like the quality of service (QoS) and discovery aspects, for the network and as well as for mobile phone users. The paper summarizes the challenges and research in this domain, along with our developed mobile web service mediation framework (MWSMF). However, to scale Mobile Enterprise to the loads possible in cellular networks, we shifted some of its components to the new utility computing paradigm, cloud computing. The cloud based load balancing for the Mobile Enterprise can be provided at the middleware framework level or at the individual services level. This paper described both the approaches, with two Mobile Host application scenarios, in collaborative m-learning and multimedia services domains. The analysis concludes that MWSMF and its components are horizontally scalable, thus allowing to utilize elasticity of cloud platform to meet load requirements of Mobile Enterprise in an easy and quick manner.**

***Keywords*-Mobile web services, Mobile Host, Mobile Enterprise, cloud computing, QoS and enterprise service bus**

I. Introduction

* This journal paper is an extension to our prior work published at [1].

Mobile data services in tandem with web services [2] are seemingly the path breaking domain in current information systems research. In mobile web services domain, the resource constrained smart phones are used as both web service clients and providers (Mobile Host). Mobile terminals accessing the web services cater for anytime and anywhere access to services. Some interesting mobile web service applications are the provisioning of services like information search, language translation, company news etc. for employees who travel regularly. There are also many public web services like the weather forecast, stock quotes etc. accessible from smart phones. Mobile web service clients are also significant in the geospatial and location based services [3]. While mobile web service clients are common, the scope of mobile web service provisioning (MWSP) was studied at RWTH Aachen University since 2003 [4], where Mobile Hosts were developed, capable of providing basic web services from smart phones. Mobile web service clients and the Mobile Hosts in a cellular network, together form a Mobile Enterprise.

Mobile Hosts enable seamless integration of user-specific services to the enterprise, by following standard web service interfaces and standards also on the radio link. Moreover, services provided by the Mobile Host can be integrated with larger enterprise services bringing added value to these services. For example, services can be provided to the mobile user based on his up-to-date user context. Context details like device and network capabilities, location details etc. can be obtained from the mobile at runtime and can be used in providing most relevant services like maps specific to devices and location information. Besides, Mobile Hosts can collaborate among themselves in scenarios like Collaborative Journalism and Mobile Host Co-learn System and bring value to the enterprise. [5]

Once the Mobile Host was developed, an extensive performance analysis was conducted to prove its technical feasibility [4]. While service delivery and management from Mobile Host were thus shown technically feasible, the ability to provide proper quality of service (QoS), especially in terms of security and reasonable scalability, for the Mobile Host is observed to be very critical. Similarly, huge number of web services possible, with each Mobile Host providing some services in the wireless network, makes the discovery of these services quite complex. Proper QoS and discovery mechanisms are required for successful adoption of mobile web services into commercial environments. Moreover, the QoS and discovery analysis of mobile web services has raised the necessity for intermediary nodes helping in the integration of Mobile Hosts with the enterprise. Based on these requirements a Mobile Web Services Mediation Framework (MWSMF) [6] is designed as an intermediary between the web service clients and the Mobile Hosts within the Mobile Enterprise, using the Enterprise Service Bus (ESB) technology.

While we were successful in establishing MWSMF on standard servers, the scale of the Mobile Enterprise is leading us to the new utility computing paradigm, cloud computing. We also have observed that load balancing is the key in successful deployment of Mobile Enterprise in commercial environments. So, we established the mediation framework on a public cloud infrastructure so that the framework can adapt itself to the loads of the mobile operator proprietary networks, thus mainly helping in horizontal scaling and load balancing the MWSMF and its components and consequently the Mobile Enterprise. The remaining sections of the paper are ordered as follows.

Section II discusses the details of providing services from smart phones. Section III discusses the challenges associated with establishing a Mobile Enterprise. Section IV discusses the details of the MWSMF. Section V discusses cloud computing and load handling issues of the MWSMF along with the analysis and results. Section VI discusses the details and approach of dragging Mobile Enterprise to the cloud with detailed analysis. Section VII concludes the paper with future research options.

II. Mobile web service provisioning

The quest for enabling open XML web service interfaces and standardized protocols also on the radio link, with the latest developments in cellular domain, lead to a new domain of applications, mobile web services. The developments in cellular world are two folded; firstly there is a significant improvement in device capabilities like better memory and processing power and secondly with the latest developments in mobile communication technologies with 3G and 4G technologies, higher data transmission rates in the order of few mbs were achieved. In the mobile web services domain, the resource constrained mobile devices are used as both web service clients and providers, still preserving the basic web services architecture in the wireless environments. While mobile web service clients are quite common these days [3], the research with providing web services from smart phones is still sparse.

The main benefit with Mobile Host is the achieved integration and interoperability for the mobile devices. It allows applications written in different languages and deployed on different platforms to communicate with Mobile Hosts over the cellular network. Moreover, the paradigm shift of smart phones from the role of service consumer to the service provider is a step toward practical realization of various computing paradigms such as pervasive computing, ubiquitous computing, ambient computing and context-aware computing. For example, the applications hosted on a mobile device provide information about the associated user (e.g. location, agenda) as well as the surrounding environment (e.g. signal strength, bandwidth). Mobile devices also support multiple integrated devices (e.g. camera) and auxiliary devices (e.g. Global Positioning Systems (GPS) receivers, printers). For the hosted services, they provide a gateway to make available their functionality to the outside world (e.g. providing paramedics assistance). In the absence of such provisioning functionality the mobile user has to regularly

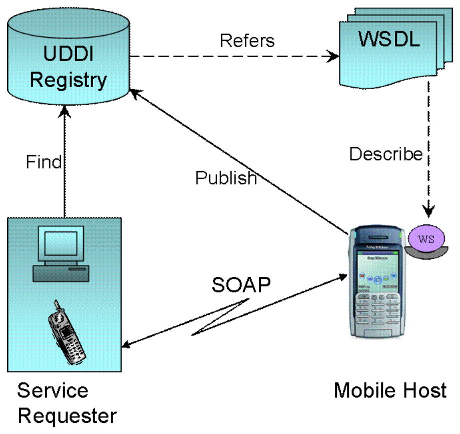

Figure 1. Basic mobile web services framework with the Mobile Host

update the contents to a standard server, with each update of the device's state.

With the intent, in our mobile web service provisioning project one such Mobile Host [4] was developed, proving the feasibility of concept. Figure 1 shows the basic mobile web services framework with web services being provided from the Mobile Host. Mobile Host is a light weight web service provider built for resource constrained devices like cellular phones. It has been developed as a web service handler built on top of a normal Web server. The SOAP based web service requests sent by HTTP tunneling are diverted and handled by the web service handler component. The Mobile Host was developed in PersonalJava on a SonyEricsson P800 smart phone. The footprint of the fully functional prototype is only 130 KB. Open source kSOAP2 [7] was used for creating and handling the SOAP messages. The key challenges addressed in Mobile Host's development are threefold: to keep the Mobile Host fully compatible with the usual web service interfaces such that clients will not notice the difference; to design the Mobile Host with a very small footprint that is acceptable in the smart phone world; and to limit the performance overhead of the web service functionality such that neither the services themselves nor the normal functioning of the smart phone for the user is seriously impeded.

The detailed performance evaluation of this Mobile Host clearly showed that service delivery as well as service administration can be done with reasonable ergonomic quality by normal mobile phone users. As the most important result, it turns out that the total web service processing time at the Mobile Host is only a small fraction of the total request-response time ($< 10\%$) and rest all being transmission delay. This makes the performance of the Mobile Host directly proportional to achievable higher data transmission rates. Further, the regression analysis of the Mobile Host showed that the Mobile Host can handle up to 8 concurrent requests for reasonable services of message sizes approximately 2

Kb. Mobile Host is also possible with other Java variants like Java 2 Micro Edition (J2ME) [8], for smart phones. We also have developed a J2ME based Mobile Host and its performance was observed to be not so significantly different from that of the PersonalJava version.

Mobile Host opens up a new set of applications and it finds its use in several domains like mobile community support, collaborative learning, social systems etc. Primarily, the smart phone can act as a multi-user device without additional manual effort on part of the mobile carrier. Several applications were developed and demonstrated with the Mobile Host, for example in a remote patient tele-monitoring scenario, the Mobile Host can collect remote patient's vital signs like blood pressure, heart rate, temperature etc. from different sensors and provide them to the doctors in real time. In the absence of such Mobile Host the details are to be regularly updated to a server, where from the doctor can access the details. The latter scenario causes problems with stale details and increased network loads. A second example is that in case of a distress call; the mobile terminal can provide a geographical description of its location (as pictures) along with location details. Another interesting application scenario involves the smooth co-ordination between journalists and their respective organizations while covering events like Olympics. [5]

III. Mobile enterprise

A Mobile Enterprise [5], [9] can be established in a cellular network by participating Mobile Hosts and their clients, where the hosts provide user-specific services to the clients as per the WS* standards. However, such a Mobile Enterprise established, poses many technical challenges, both to the service providers and to the mobile operator. Some of the critical challenges and associated research are addressed in this section.

A. *Challenges for establishing Mobile Enterprise*

Figure 2 shows the Mobile Enterprise and hints the critical challenges posed to the mobile phone users and the operators. As the Mobile Host provides services to the Internet, devices should be safe from malicious attacks. For this, the Mobile Host has to provide only secure and reliable communication in the vulnerable and volatile mobile ad-hoc topologies. In terms of scalability, the Mobile Host has to process reasonable number of clients, over long durations, without failure and without seriously impeding normal functioning of the smart phone for the user.

Similarly, huge number of available web services, with each Mobile Host providing some services in the wireless network, makes the discovery of the most relevant services quite complex. Proper discovery mechanisms are required for successful adoption of Mobile Enterprise. The discovery, moreover, poses some critical questions like: where to publish the services provided by the Mobile Hosts? Should

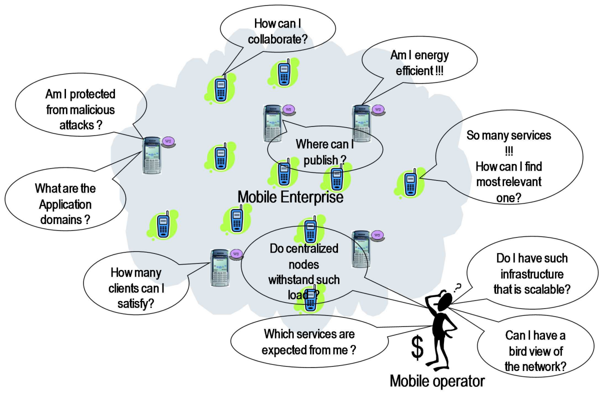
How can I collaborate?
Am I energy efficient !!!
Am I protected from malicious attacks ?
Where can I publish ?
So many services !!! How can I find most relevant one?
What are the Application domains ?
Mobile Enterprise
Do centralized nodes withstand such load ?
Do I have such infrastructure that is scalable?
How many clients can I satisfy?
Which services are expected from me ?
$
Can I have a bird view of the network?
Mobile operator

Figure 2. Mobile Enterprise and the critical challenges posed to the mobile phone users and the operator

they be published with the centralized Universal Description, Discovery, and Integration (UDDI) registries available in the Internet or the operator is going to offer some help? This also raises questions like whether centralized nodes can withstand such high loads or some alternatives are to be looked at?

From the mobile operator's perspective the Mobile Enterprise poses questions like: what are the services expected by the mobile users from the operator? Can the operator monitor the communication and have a bird view of the complete network, so that business scenarios can be drawn out of it? Do operators have such infrastructure that can scale and adapt to such huge oscillating requirements? What about the scalability of such infrastructure?

Our research in this domain focused at addressing most of these issues [5] and the remaining parts of this paper summarize the research and results.

B. *QoS aspects of the Mobile Host*

Providing proper QoS, especially, appropriate security and reasonable scalability, for mobile web service provisioning domain was observed to be very critical. The security analysis of the Mobile Host studied the adaptability of WS-Security specification to the MWSP domain and concludes that not all of the specification can be applied to the Mobile Host, mainly because of resource limitations. The results of our analysis suggest that the mobile web service messages of reasonable size, approximately 2-5kb, can be secured with web service security standard specifications. The security delays caused are approximately 3-5 seconds. We could also conclude from the analysis that the best way of securing messages in a Mobile Enterprise is to use AES (Advanced Encryption Standard) symmetric encryption with 256 bit key, and to exchange the keys with RSA 1024 bit asymmetric key exchange mechanism and signing the messages with RSAwithSHA1. But there are still high performance penalties when messages are both encrypted and signed. So we suggest encrypting only the parts of the message, which are critical in terms of security and signing the message. The signing on top of the encryption can completely be avoided in specific applications with lower security requirements. [10]

In terms of scalability, the layered model of web service communication, introduces a lot of message overhead to the exchanged verbose XML based SOAP messages. This consumes a lot of resources, since all this additional information has to be exchanged over the radio link. Thus for improving scalability the messages are to be compressed without effecting the interoperability of the mobile web services. Message compression also improves the energy efficiency of the devices as there will be less data to transmit.

In the scalability analysis of the Most Host [11], we have adapted BinXML [12] for compressing the mobile web service messages. BinXML is a light-weight XML compression mechanism, which replaces each XML tag and attribute with a unique byte value and replaces each end tag with 0xFF. By using a state machine and 6 special byte values including 0xFF, any XML data with circa 245 tags can be represented in this format. The approach is specifically designed to target SOAP messages across radio links. So the mobile web service messages are exchanged in the BinXML format, and this has reduced the message of some of the services by 30%, drastically reducing the transmission delays of mobile web service invocation. The BinXML compression ratio is very significant where the SOAP message has repeated tags and deep structure. The binary encoding is also significant for the security analysis as there was a linear increase in the size of the message with the security incorporation. The variation in the WS-Security encrypted message size for a typical 5 Kb message is approximately 50%. [5]

C. *Discovery aspects of the Mobile Enterprise*

In a commercial Mobile Enterprise with Mobile Hosts, and with each Mobile Host providing some services for the Internet, expected number of services to be published could be quite high. Generally web services are published by advertising WSDL (Web Services Description Language) descriptions in a UDDI 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.

Dynamic service discovery is one of the most extensively explored research topics in the recent times. Most of these service discovery protocols are based on the announce-listen model like in Jini. In this model periodic multicast mechanism is used for service announcement and discovery. But these mechanisms assume a service proxy object that acts as the registry and it is always available. For dynamic ad hoc networks, assuming the existence of devices that are stable and powerful enough to play the role of the central service registries is inappropriate. Hence services distributed in the ad-hoc networks must be discovered without a centralized registry and should be able to support spontaneous peer to peer (P2P) connectivity. [13] proposes a distributed peer to peer Web service registry solution based on lightweight Web service profiles. They have developed VISR (View based Integration of Web Service Registries) as a peer to peer architecture for distributed Web service registry. Similarly Konark service discovery protocol [14] was designed for discovery and delivery of device independent services in ad hoc networks.

Considering these developments and our need for distributed registry and dynamic discovery, we have studied alternative means of mobile web service discovery and realized a discovery mechanism in the P2P network. In this solution, the virtual P2P network also called the mobile P2P network is established in the mobile operator network with one of the nodes in operator proprietary network, acting as a JXTA super peer. JXTA (Juxtapose) is an open source P2P protocol specification. Once the virtual P2P network is established, the services deployed on Mobile Host in the JXME virtual P2P network are to be published as JXTA advertisements, so that they can be sensed as JXTA services among other peers. JXTA specifies Modules as a generic abstraction that allows peers to describe and instantiate any type of implementation of behavior representing any piece of ”code” in the JXTA world. So the mobile web services are published as JXTA modules in the virtual P2P network. Once published to the mobile P2P network, the services can later be discovered by using the keyword based search provided by JXTA. This approach also considered categorizing the services and the advanced features like context aware service discovery. We address the discovery solution as mobile P2P discovery mechanism. The evaluation of the discovery approach suggested that the smart phones are successful in identifying the services in the P2P network, with reasonable performance penalties for the Mobile Host. [15]

IV. Mobile web services mediation framework

Mobile Hosts with proper QoS and discovery mechanisms, enable seamless integration of user-specific services to the Mobile Enterprise. Moreover services provided by the Mobile Host can be integrated with larger enterprise services bringing added value to these services. However, enterprise networks deploy disparate applications, platforms, and business processes that need to communicate or exchange data with each other or in this specific scenario addressed by the paper, with the Mobile Hosts. The applications, platforms and processes of enterprise networks generally have non-compatible data formats and non-compatible communications protocols. Besides, within the domain of our research, the QoS and discovery study of the Mobile Host offered solutions in disparate technologies like JXTA. This leads to serious integration problems within the networks. The integration problem extends further if two or more of such enterprise networks have to communicate among themselves. We generally address this research scope and domain, as the Enterprise Service Integration.

The mobile web services mediation framework (MWSMF) [6] is established as an intermediary between the web service clients and the Mobile Hosts in mobile enterprise. ESB is used as the background technology in realizing the mediation framework. Similar mediation mechanisms for mobile web services are addressed in [16]. Especially, [16] describes the status of research with provisioning services from resource constrained devices. When considering mediation within semantic web services, Web Service Modeling Ontology (WSMO) has significant contributions [17]. However, we went with the ESB approach, due to the availability of several open source implementations.

Figure 3 shows the Mobile Enterprise and the basic components of the mediation framework. For realizing the mediation framework we relied on ServiceMix [18], an open source implementation of ESB, based on the JBI specification [19]. JBI architecture supports two types of components Service Engines and Binding Components. Service engines are components responsible for implementing business logic and they can be service providers/consumers. Service engine components support content-based routing, orchestration, rules, data transformations etc. Service engines communicate with the system by exchanging normalized messages across the normalized message router (NMR). The normalized messaging model is based on WSDL specification. The service engine components are shown as straight lined rectangles in the figure. Binding components are used to send and receive messages across specific protocols and transports. The binding components marshal and unmarshal messages to and from protocol-specific data formats to normalized messages. The binding components are shown as dashed rectangles in the Figure 3.

The HttpReceiver component shown in figure 3 receives the web service requests (SOAP over HTTP) over a specific port and forward them to the Broker component via NMR. The main integration logic of the mediation framework is maintained at the Broker component. For example, in case of the scalability maintenance, the messages received by Broker are verified for mobile web service messages. If the messages are normal Http requests, they are handled by the HttpInvoker binding component. If they comprise mobile web service messages, the Broker component further ensures the QoS of the mobile web service messages and transforms them as and when necessary, using the QoSVerifier service engine component, and routes the messages, based on their content, to the respective Mobile Hosts. The framework also

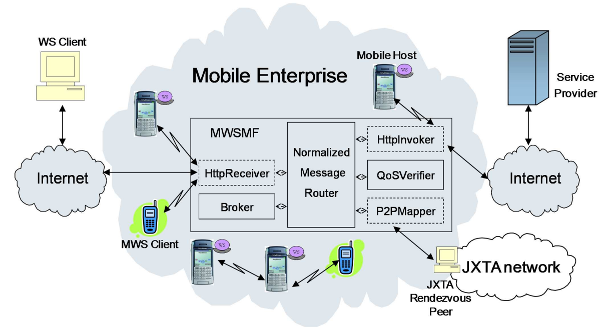

Figure 3. Mobile Enterprise setup with Mobile Hosts, MWS clients and MWSMF

ensures that once the mobile P2P network is established, the web service clients can discover the services using mobile P2P discovery mechanism and can access deployed services across MWSMF and JXTA network. [6]

Apart from security and improvements to the scalability, QoS provisioning features of the MWSMF also include message persistence, guaranteed delivery, failure handling and transaction support. External web service clients, that do not participate in the mobile P2P network, can also directly access the services deployed on the Mobile Hosts via MWSMF, as long as the web services are published with any public UDDI registry or the registry deployed at the mediation framework and the Mobile Hosts are provided with public IPs. This approach evades the JXME network completely. Thus the mediation framework acts as an external gateway from Internet to the Mobile Hosts and mobile P2P network. The framework also provides a bird view of the mobile enterprise to the cellular operator, so that business scenarios can be drawn out of it. Preliminary analysis of the mediation framework is available at [5].

V. MWSMF ON THE CLOUD

While the MWSMF was successful in achieving the integrational requirements of the Mobile Host and the Mobile Enterprise, a standalone framework again faces the troubles with heavy loads. The problems with scalability are quite relevant in such scenarios and the system should scale on demand. For example number of Mobile Hosts providing the services and the number of services provided by the Mobile Hosts can explode while some events are underway; like Olympics or national elections etc. Some of these application scenarios are addressed in [5]. This increases the number of MWS clients the framework has to support. Elasticity of the framework can be defined as its ability to adjust according to the varying number of requests, it has to support. As the study targets the scales of mobile operator proprietary networks, to achieve elasticity, horizontal scaling (scaling by adding more nodes to the cluster, rather than increasing performance of a single node) and load balancing for the

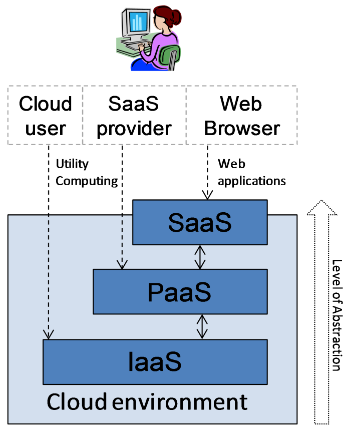

Figure 4. Level of abstraction and layers of cloud services

MWSMF, we tried to realize the mediation framework on a public cloud.

A. Cloud computing

Cloud computing is a style of computing in which, typically, resources scalable on demand are provided "as a service (aaS)" over the Internet to users who need not have knowledge of, expertise in, or control over the cloud infrastructure that supports them. Cloud computing mainly forwards the idea of utility computing along with virtualization. In the utility computing model, consumers pay based on their usage of computing resources, just like the traditional utility services e.g. water, electricity, gas etc. Just like any utility services model cloud computing benefits from economies of scale. On the other hand, virtualization technologies partition hardware and thus provide flexible and scalable computing platforms. Servers in the cloud can be physical machines or virtual machines. A cloud computing platform dynamically provisions, configures, reconfigures, and de-provisions servers as requested. [20], [21]

Cloud services are provided on demand and at different levels. Figure 4 shows the layers of cloud services, in terms of level of abstraction. The provisioning of services can be at the Infrastructural level (IaaS) or Platform level (PaaS) or at the Software level (SaaS). In the IaaS, commodity computers, distributed across Internet, are used to perform parallel processing, distributed storage, indexing and mining of data. IaaS provides complete control over the operating system and the clients benefit from the computing resources like processing power and storage, e.g. Amazon EC2 [22]. Virtualization is the key technology behind realization of these services. PaaS mainly provides hosting environments for other applications. Clients can deploy the domain specific applications on these platforms, e.g. Google App Engine [23]. These applications are in turn provided to the users as SaaS. SaaS are generally accessible from web browsers, e.g. Facebook. Web 2.0 is the main technology behind the realization of SaaS. However, the abstraction between the layers is not concrete and several of the examples can be argued for other layers.

While there are several public clouds on the market, Google Apps (Google Mail, Docs, Sites, Calendar, etc), Google App Engine (provides elastic platform for Java and Python applications with some limitations) and Amazon EC2 are probably most known and widely used. Elastic Java Virtual Machine on Google App Engine allows developers to concentrate on creating functionality rather than bother about maintenance and system setup. Such sandboxing, however, places some restrictions on the allowed functionality [23]. Amazon EC2 on the other hand allows full control over virtual machine, starting from the operating system. It is possible to select a suitable operating system, and platform (32 and 64 bit) from many available Amazon Machine Images (AMI) and several possible virtual machines, which differ in CPU power, memory and disk space. This functionality allows to freely select suitable technologies for any particular task. In case of EC2, price for the service depends on machine size, its uptime, and used bandwidth in and out of the cloud. Flexibility of EC2 environment and our existing Mobile Enterprise implementation were some of the reasons why EC2 was chosen for most of our experiments.

Moreover, there are free implementations of EC2 compatible cloud infrastructure e.g. Eucalyptus [24], that help in creating private clouds. Thus the cloud computing applications can initially be developed at the private clouds and later can be scaled to the public clouds. The setup is of great help for the research and academic communities, as the initial expenses of experiments can be reduced by great extent. Our research group is in the process of setting up a scientific computing cloud (SciCloud) on our clusters, using Eucalyptus technology. With this SciCloud [25], students and researchers can efficiently use the already existing resources of university computer networks, in solving computationally intensive scientific, mathematical, and academic problems. The project mainly targets the development of a framework, including models and methods for establishment, proper selection, state management (managing running state and data), auto scaling and interoperability of the private clouds. The preliminary results can be found at the project site [26] and will be addressed by our future publications.

B. Load balancing the MWSMF from the cloud

To achieve the scalability for the mediation framework, the MWSMF was installed on the Amazon EC2 cloud. Once the Amazon Machine Images (AMI) are configured, stateless nature of the MWSMF allows, fairly easy horizontal scaling by adding more MWSMF nodes and distributing the load among them with the load balancer. Figure 5 shows the deployment scenario with the load balancer (LB) and the MWSMF worker AMI nodes (W-1, W-2, W-3, and W-n).

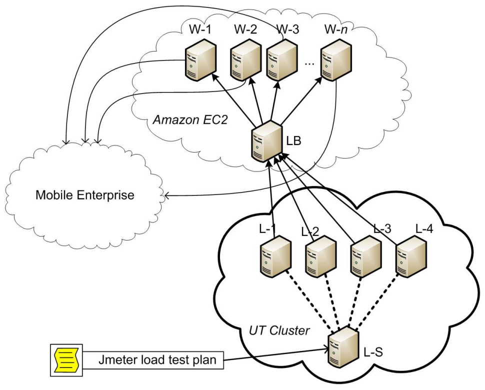

Figure 5. Load test setup for the MWSMF

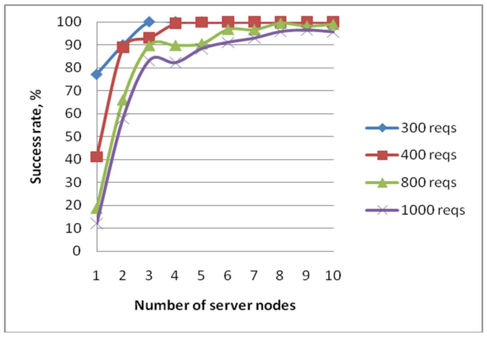

Figure 6. Success rate of concurrent requests over multiple server nodes

There are several load balancing techniques that can be used in this scenario. One approach is to use DNS based load balancing, where each call to the DNS server will result in different IP address. This means that each MWSMF node will be accessed by certain subset of clients directly, without an intermediary load balancing proxy as discussed below. This approach is not fault tolerant in case the framework node would crash but its IP would be cached on the client's DNS cache. However, this approach is inevitable, if loads on the single proxy based load balancer will grow to a level that a single load balancer itself will become a bottleneck. Another approach is to use load balancing proxy server in front of MWSMF nodes. Among other options, Apache HTTPD server with mod_proxy and mod_load_balancer is probably most commonly used configuration. It has one major drawback in elastic environment, as it doesn't allow dynamic reconfiguration of worker nodes. If we add or remove some MWSMF nodes we are required to restart load balancer as well, which is not convenient and potentially introduces some failed requests during restart.

Alternative http proxy load balancer HAProxy [27] allows such dynamic behavior. However we used Apache HTTP server with mod_proxy and mod_load_balancer [28] as a load balancer for the MWSMF because it is more widespread and we had experience in configuring such setup, which was important, as the aim of this research was to show the horizontal scalability rather than achieve maximum automation. However we have considered HAProxy [27] in the analysis of scaling the Mobile Enterprise in total. The scenario is explained in the next section with a usage scenario.

C. Scalability of the MWSMF

Load testing of MWSMF on the cloud was performed in a distributed manner using JMeter - open source load testing software. Figure 5 shows the deployment setup in detail. JMeter was deployed on one of the clusters in the University of Tartu (UT Cluster). Deployment consisted of 4 slave nodes (L-1, L-2, L-3, and L-4) and 1 master node (L-S). Load testing scenario (called a test plan in Jmeter) is loaded on the master node, which sends it to the slave nodes and initiates load testing. During the test run each slave node sends testing results back to the master node, where results are aggregated into a single report.

In our test scenario we performed 5 consequent requests by n concurrent threads, where n varied between 75 and 250 per slave node, which makes 300 to 1000 concurrent requests on a load balancer, thus simulating a large number of simultaneous clients for the MWSMF and the Mobile Host in Mobile Enterprise. Another important factor that impacts test results is a connection and response timeout on the client, in our test case - the slave node. Connection timeout is a time until connection to the server is established and response timeout is the time since call starts on the client side until response is received. If these timeouts are long enough, then observations showed, that even single MWSMF node can withstand large loads, due to the sufficient QoS of the ESB. However, in such scenario a call may last too long for a mobile client and the client may start retransmitting instead of waiting. In our tests we set connection-response timeout to 50-70 seconds. It must be also noted that, in the real life connection timeout on a client side is not a parameter that the service provider can affect nor predict. In case of interactive applications, where user interaction is involved, response should be preferably delivered in less than 10 seconds to keep user's attention [29].

On the cloud front a load balancer (LB) and up to 10 MWSMF worker nodes were set up. To show the elasticity of the cloud we increased the number of the server nodes from 1 to 10 after each test. All servers were running on Amazon EC2 infrastructure and all of them were using EC2 small instances. Small instance has 1.7 GB of memory, CPU power of 1 EC2 Compute unit, which is equivalent to CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor as of 07.12.2009 (CPU capacity of an EC2 compute unit do change in time). Both load balancer and worker nodes were running 32 bit Linux platforms. Apache HTTP server version 2.2.8 with mod_proxy and mod_load_balancer were used as a load balancer. Load balancer was setup to use request based scheduling, which means that all worker nodes received equal amount of requests. However, it is possible to configure load balancer based on traffic or busyness. Busyness means how many concurrent requests a worker node has at the moment the new request arrives. In real-life situation best load balancing algorithm for a particular scenario should be chosen based on the services provided by the mobile enterprise and the nature of the request/response traffic. For more details on load balancing algorithm refer [28].

In the load test of the MWSMF, we measured how success rate of the requests depends on a number of worker nodes depending on a number of concurrent requests. Success means that a request will get a response before connection or response timeout occurs and success rate shows how many requests from all performed requests succeeded. The results of the experiment are shown in figure 6. From the diagram it can be clearly seen that the percentage of succeeded requests grows logarithmically with the number of nodes and degrades exponentially as load grows. Performance of a single node drops rapidly already after 300 concurrent requests and even with 300 concurrent requests success rate is only 77%, however 3 nodes can handle this load with 100% success rate. It can be also seen, that with current setup adding more nodes does not show any visible effect after 6 nodes and performance is improved by an insignificant fraction in contrast to difference between 1, 2 and 3 nodes.

In summary we observed that, with current MWSMF implementation one single node can handle around 100-130 concurrent MWS requests with 100% success rate. Adding an additional node adds roughly 100 new concurrent requests to the total capability until the load grows up to 800 concurrent requests, when load balancer itself becomes a bottleneck and adding any additional nodes do not give desired effect. This analysis showed mediation framework to be horizontally scalable. However, certain loads demand more advanced load balancing techniques. The elastic cloud environment helps to achieve this required setup very quickly.

VI. Scaling mobile enterprise

While our earlier analysis proved that MWSMF is horizontally scalable, scaling the Mobile Enterprise in total is a different issue. Our earlier analysis only considered the load balancing ahead of the MWSMF itself. However, individual services also can become a bottleneck and thus the load on them has to be balanced. So the load balancing for the Mobile Enterprise as a whole has to be extended further. To sum it up – cloud based load balancing for the Mobile Enterprise can be at the mediation framework level or at the individual service level. We try to address both the approaches in terms of two application scenarios, MobileHost CoLearn System and lightweight application server (LAS) services for mobiles.

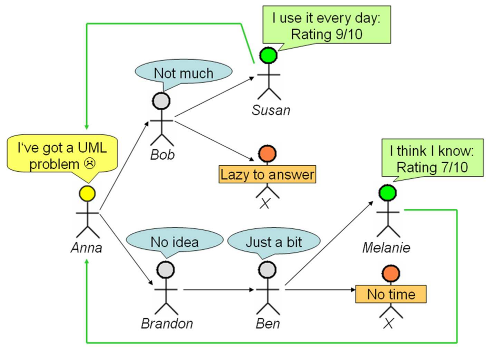

Figure 7. Expert finder scenario with the MobileHost CoLearn System

A. MobileHost CoLearn System

The MobileHost CoLearn System studied the scope of Mobile Host in m-learning (mobile learning) domain [30]. The system presents a novel approach to expert finding within a truly mobile collaborative learning environment. It targets the framework of the learner's social network, along with the social networks of her acquaintances, and the social networks of the acquaintances of her acquaintances, and so forth. Such an expert finder flow usually leads to the discovery of more than one potential expert, and the learner's subjective decision who of them is the most knowledgeable one can be based either on the rating for the expert's level of expertise in the field, or on the path that the expert finder request has traveled before reaching the respective expert. An example scenario, using the system, is illustrated in the figure 7. In the expert finder scenario of the system, every participant should act as both provider and client for the messages being exchanged, which the Mobile Host technology has made feasible. After having found an expert, the learner is provided with all the necessary information in order to contact that expert for further assistance regarding specific issues.

Alongside the valuable knowledge that flows within the system from the experts to non-experienced learners, the system supports the retrieval of a variety of literature resources, such as articles, proceedings, pictures, audio or video lecture recordings, location details, and other learning services. Most often the resources are tagged by the learners. As tagging is something subjective, a three-level scale of *relevance of a tag to a resource* has been introduced. The system also has support for image and audio resources within photocasting and podcasting channels. The channels automatically distribute resources to all subscribers, as soon as they become available. MobileHost CoLearn system is the first of its kind that adapts mobile web services for collaborative learning, bringing the benefits of the latest technological developments to the learner. [30]

B. Scalable MobileHost CoLearn System

In the MobileHost CoLearn System, the main load for the Mobile Enterprise was at handling large number of clients and at providing the QoS services from the MWSMF. For example, MWSMF has to convert the incoming XML based messages to BinXML format so that the messages can be exchanged across the radio link. The process is taken care by the QoSVerifier component of the MWSMF (figure 8). So to increase the elasticity for the Mobile Enterprise we can establish a load balancer in front of the MWSMF running on several nodes in the public cloud, handling the mobile clients by accessing services from several Mobile Hosts. This is what was showed by our earlier analysis.

The next subsections discuss scaling the Mobile Enterprise with respect to load balancing at the individual services level.

C. Mobile access to LAS services

LAS is a lightweight application server (LAS) designed as a community middleware that is capable of managing users and multiple hierarchically structured communities along with their particular access rights as well as a set of services accessible to users. LAS mainly offers MPEG-7 (Moving Picture Experts Group) multimedia services to the users. MPEG-7 is a well-established and widely used standard in multimedia data management. However, due to its inherent complexity it was not used in mobile data management that often. With new initiatives like the application profiles the use of MPEG-7 has become much easier, also for mobile data management. A community application can make use of the offered services by simply connecting to the server and then remotely invoking service methods. Server functionality of the LAS is easily extensible by implementing and plugging in new services and respective components. Many community information systems have been built on top of this framework including MIST; a MPEG-7 based non-linear digital storytelling system, ACIS; a multimedia information system, and CAAS; a mobile application for context-aware search and retrieval of multimedia and community members. [31]

Even though, LAS is a reliable application server, it is not a pure web service architecture; it was not designed under the SOA paradigm and important aspects like scalability and distributed services were not taken into account. QoS and performance problems have been observed recently by LAS users. For many years, LAS has been used on top of traditional networks infrastructures for providing the services required by social software such as Virtual Campfire [32]. Virtual Campfire, is an advanced framework to create, search, and share multimedia artifacts with context awareness across communities.

Recently, the multimedia services are also being offered to the Mobile Hosts and mobile phone users. The multimedia services can be accessed from mobile phones in three modes:

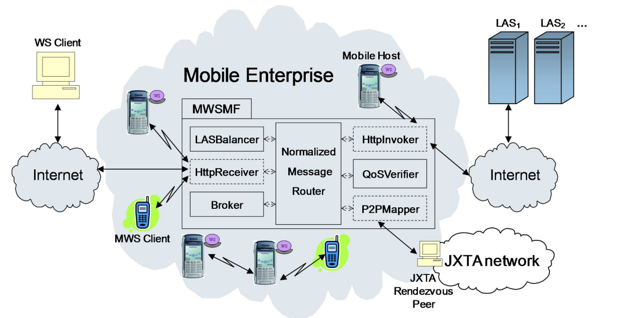

Figure 8. Mobile Enterprise setup with the LASBalancer and LAS server farm coming into the picture

1) Directly accessing the MPEG services through the Mobile Host.
2) Accessing the service through the mediation framework of the mobile enterprise.
3) Indirect way of using the Mobile Host to connect to the mediation framework.

Generally LAS services are extended and are provided as services from the Mobile Host. The extensions can be user specific. This entry of the Mobile Hosts into the LAS has further advanced the scalability problem. Load balancing and cluster support were observed to be the immediate requirements for improving the performance of the LAS.

D. Load balancing mobile access to LAS services

Contrary to the MobileHost CoLearn System, the QoS of the LAS can be improved either by changing the architecture of the LAS to have the cluster support or to employ a binding component on the MWSMF, taking care of the load balancing issues. In the first case, a hardware based load balancer can be deployed through specialized devices, like multilayer switches [33]. However, implementing, configuring and maintaining this solution is costly in terms of money and time.

Alternatively, we can deploy the LAS servers on the cloud and employ the same load balancer technique as in the case of the first scenario. As a third solution, we deployed the HAProxy node in the cloud and the requests are diverted to the respective LAS. If the load further increases, new LAS nodes can be deployed on the cloud. The main difference is that HAProxy allows dynamic behavior to the architecture and new LAS nodes can be added dynamically to the setup. This solution utilizes the elasticity, dynamic and on-demand provisioning features of the cloud, to the most. We are also studying the auto scaling of the cloud, as part of our SciCloud project. With this solution, the load balancing system can react to the sudden surges in usage patterns and can provision new nodes dynamically. The details will be addressed by our future publications.

For the third option, employing a binding component on the MWSMF, we adapted our knowledge from Mobile

Enterprise domain to the LAS. Moreover, since LAS services are also accessible to Mobile Host, we wanted to provide only a single entry to the LAS from the Mobile Enterprise. We developed components that provide web service interface to the LAS services. These integration components with the load balancer in front of them are designed to act as a cluster so that the requests are diverted to the less occupied server among a set of LASs. Connection to the LAS cluster is handled by the LASBalancer component at MWSMF. Modified Mobile Enterprise with the entry of the LASBalancer into the picture is shown in figure 8. You can see this component being present inside the MWSMF in the diagram. Inside LAS there are no necessary changes to do. Mobile users of LAS only need to connect to a single point, the MWSMF, in order to access any LAS server they are interested in. Without this solution, Mobile Hosts should have specific connection to the right LAS server based on the services offered by it. However, this architecture adds extra load to the mediation framework at LASBalancer level. QoS and fault tolerance features of ESB help to some extent, in handling this load. But, LAS requests don't need QoS transformation features of the MWSMF as the messages are sent via Internet. So the node that provided load balancing and web service interface for the LAS, is separated from the MWSMF, and we deployed it on the EC2 cloud. The HttpInvoker just diverts the LAS requests to this node. Now this node is horizontally scalable and we can apply business logic, fault tolerance and solution correctness to the cloud node without seriously affecting the performance of MWSMF. The results of the analysis are summarized in next subsection.

E. Testing the scalability of the Mobile Enterprise

In previous subsections we outlined requirements for scalability of the Mobile Enterprise and described the solution to integrate LAS and MWSMF in a scalable way. To verify our ideas, an experiment was conduced using Amazon EC2 services to scale the number of servers up and down. Deployment was made similarly to the MWSMF scalability experiment described in section V – we used Amazon EC2 public cloud infrastructure to deploy the Mobile Enterprise and the HPC (*High Performance Computing*) cluster of the University of Tartu to deploy JMeter in a distributed manner for load generation and measurement of the results. Deployment diagram is shown in Figure 9.

As we had limited access to the LAS installation, we substituted it with a mock application server. The server provides a web service that on request performs some image manipulation and sends the resulting image back to the client. As we are concerned only with the performance and scalability of the Mobile Enterprise, the functioning of the service and the application server do not affect the analysis and results. Servers hosting this image web service are shown as IS-1 . . . IS-n on the figure 9. These servers

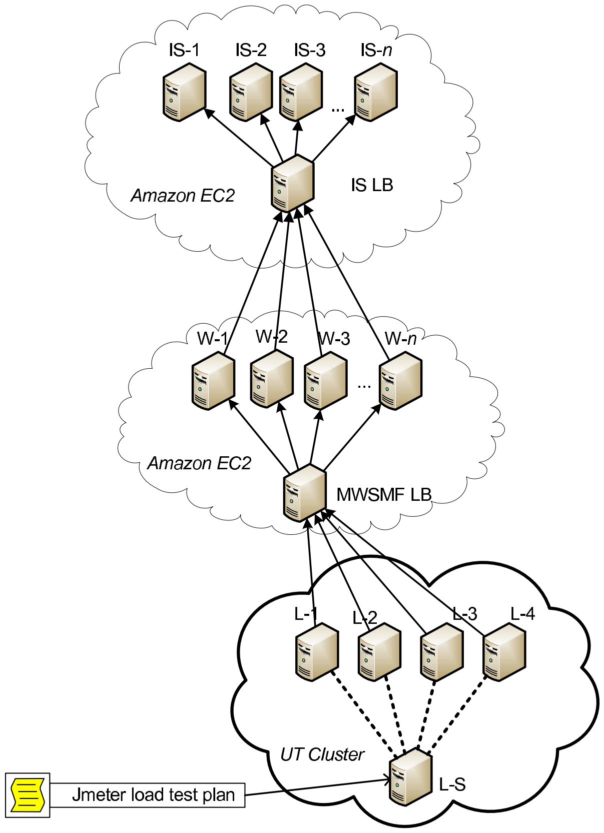

Figure 9. Deployment diagram of the scaling experiment.

constitute the cluster and the IS LB is the load balancer for it. HAProxy was used as a load balancer for this cluster.

MWSMF load balancing setup is the same used in the MWSMF scaling experiment – *MWSMF LB* is the load balancer and W-1 . . . W-n are mediation framework's nodes. However, this time we used HAProxy instead of the Apache with mod_proxy to have a consistent deployment with image cluster and to compare it with the setup from the previous experiment [1]. HAProxy showed itself more suitable for such dynamic setup because it allows easily specify the configuration file location as a command line parameter, which is a lot more convenient when lot of changes have to be made (each time a cluster was changed, configuration file had to be changed). Also, HAProxy comes with a dynamic dashboard containing extensive statistics which show a lot more information compared to the default statistics web page of mod_proxy.

This time we used 5 Apache JMeter servers instead of 4 to generate the load of 200, 400, 600, 800 and 1000 concurrent requests (which makes respectively 40, 80, 120, 160 and 200 concurrent requests per one JMeter server). In the experiment we varied the number of MWSMF and Image Server nodes in the respective clusters and tested the setup with aforementioned loads. As we concluded from

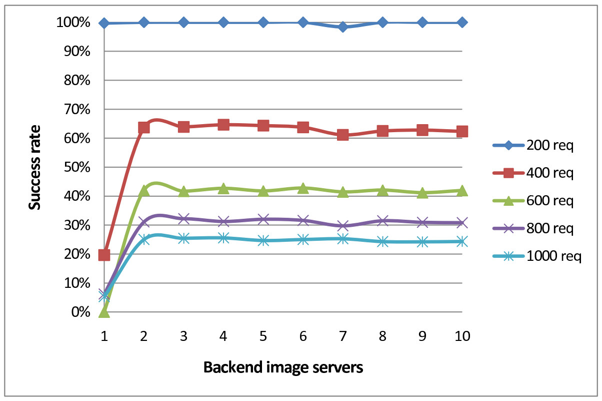

Figure 10. Scaling Image servers with 1 MWSMF node.

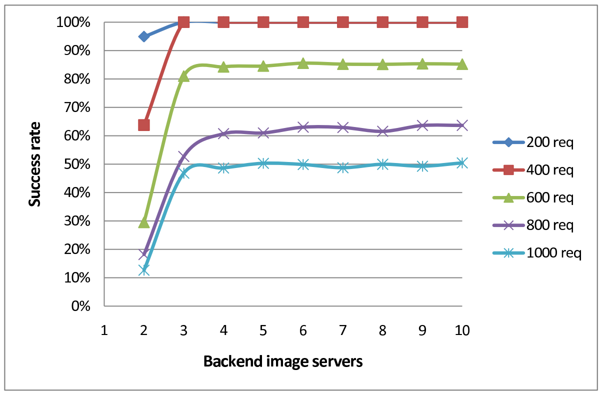

Figure 11. Scaling Image servers with 2 balanced MWSMF nodes.

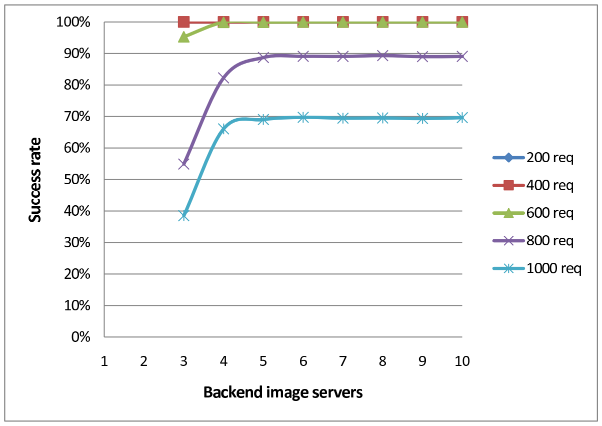

Figure 12. Scaling Image servers with 3 balanced MWSMF nodes.

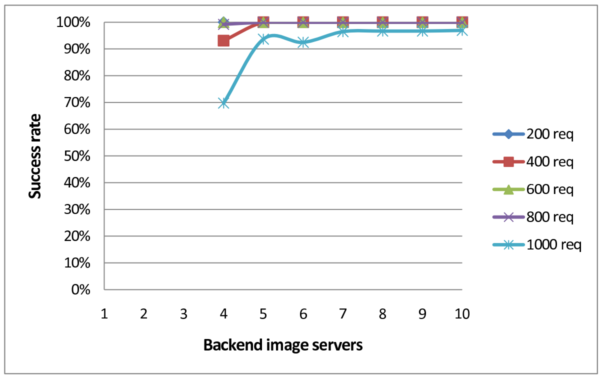

Figure 13. Scaling Image servers with 4 balanced MWSMF nodes.

the previous experiment, scaling of MWSMF makes some real impact only when scaled up to 4 nodes, after which difference in adding more servers is less visible. So this time we changed number of MWSMF nodes in the cluster from 1 to 4.

Number of servers providing Image Service was varied between 1 and 10. It must be noted, that the number of Image Service servers was always equal or bigger than the number of MWSMF nodes. The reasoning for this is an assumption, that when we model a contention of a particular service, then it shall be upscaled before mediation framework. This means that for this scenario the number of nodes for a particular service will always be bigger or equal to the number of mediation framework nodes.

Figures 10, 11, 12 and 13 summarize results of experiments. It can be seen, that increasing the number of servers for particular service results in the success rate growth and the tendency is closely similar to the characteristics observed during previous MWSMF scaling – large increase with first 3 nodes and after that difference is almost unnoticeable. Adding additional MWSMF nodes also adds to the success factor growth – it acts as a multiplier for the whole graph. This, however, also shows that mediation framework is a major factor for the scalability of the Mobile Enterprise as a whole.

VII. Conclusion and Future Research Directions

The developments in the web services domain, the improved device capabilities of the smart phones and the improved transmission capabilities of the cellular networks have lead to the mobile web services provisioning domain. With this paper, we summarized the challenges and research associated in this domain and establishing the Mobile Enterprise. The QoS aspects of the developed Mobile Host, like providing proper security and scalability, and the discovery of the provided services are addressed briefly. Further, the QoS and discovery analyses of the Mobile Host have raised the necessity for a middleware framework and the features and realization details of the MWSMF are discussed.

However to scale of Mobile Enterprise to the loads possible in mobile networks, we shifted some of its components to the cloud computing paradigm. The paper illustrated this categorical shift in terms of two application scenarios. It showed that MWSMF is horizontally scalable, thus allowing to utilize cloud's elasticity to meet load requirements in an easy and quick manner. It also illustrated different means to scale the LAS based MPEG-7 services. Thus cloud computing is shown to scale the Mobile Enterprise dynamically.

Our future research in this domain will focus at surge computing and auto scaling so that Mobile Enterprise can scale according to the oscillating loads automatically. In the experiments discussed in this paper, we configured load balancer manually and one of our future research directions is to achieve more automation in scaling process. The planned framework detects loads automatically, dynamically adds more working nodes and automatically configures load balancer to accommodate new worker nodes. After loads drop, dynamically scalable MWSMF should shutdown unnecessary worker nodes. Another future research direction is to use Eucalyptus framework for cloud infrastructure instead of Amazon EC2, to show that public cloud's elasticity is achievable also in private clouds. We also want to extend this experience to our scientific computing cloud (SciCloud) project.

Acknowledgment

The research is supported by the European Social Fund through Mobilitas program, the European Regional Development Fund through the Estonian Centre of Excellence in Computer Science and Eureka project "SITIO". The work was earlier supported by the Ultra High-Speed Mobile Information and Communication (UMIC) research cluster at RWTH Aachen University. The authors would also like to thank R. Levenshteyn and M. Gerdes of Ericsson Research for their help and support.

References

[1] S. N. Srirama, V. Šor, E. Vainikko, and M. Jarke, "Scalable mobile web services mediation framework," in *The Fifth International Conference on Internet and Web Applications and Services (ICIW 2010)*, 2010.

[2] K. Gottschalk, S. Graham, H. Kreger, and J. Snell, "Introduction to web services architecture," *IBM Systems Journal: New Developments in Web Services and E-commerce*, vol. 41(2), pp. 178–198, 2002. [Online]. Available: http://researchweb.watson.ibm.com/journal/sj/412/gottschalk.html

[3] B. Benatallah and Z. Maamar, "Introduction to the special issue on m-services," *IEEE transactions on systems, man, and cybernetics - part a: systems and humans*, vol. 33, no. 6, pp. 665–666, November 2003.

[4] S. N. Srirama, M. Jarke, and W. Prinz, "Mobile web service provisioning," in *AICT-ICIW '06: Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services*. IEEE Computer Society, 2006, p. 120.

[5] S. Srirama and M. Jarke, "Mobile hosts in enterprise service integration," *International Journal of Web Engineering and Technology (IJWET)*, vol. 5, no. 2, pp. 187–213, 2009.

[6] S. N. Srirama, M. Jarke, and W. Prinz, "Mobile web services mediation framework," in *Middleware for Service Oriented Computing (MW4SOC) Workshop @ 8th International Middleware Conference 2007*. ACM Press, 2007.

[7] kSOAP2, "kSOAP2 - An efficient, lean, Java SOAP library for constrained devices," SourceForge.net, 2007. [Online]. Available: http://sourceforge.net/projects/ksoap2

[8] Sun Microsystems, "Java^{TM} 2 Platform, Micro Edition (J2ME^{TM}) Web Services Specification - Datasheet," Sun Microsystems, Inc., Tech. Rep., 2007.

[9] S. N. Srirama and M. Jarke, "Mobile enterprise - a case study of enterprise service integration," in *3rd International Conference and Exhibition on Next Generation Mobile Applications, Services and Technologies (NGMAST 2009)*. IEEE Computer Society, September 2009, pp. 101–107.

[10] 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.

[11] S. N. Srirama, M. Jarke, and W. Prinz, "MWSMF: A mediation framework realizing scalable mobile web service provisioning," in *International Conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications (Mobilware 2008)*. ACM Press, 2008.

[12] M. Ericsson and R. Levenshteyn, "On optimization of XML-based messaging," in *Second Nordic Conference on Web Services (NCWS 2003)*, November 2003, pp. 167–179.

[13] 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.

[14] C. Lee, A. Helal, N. Desai, V. Verma, and B. Arslan, "Konark: A system and protocols for device independent, peer-to-peer discovery and delivery of mobile services," *IEEE transactions on systems, man, and cybernetics - part a: systems and humans*, vol. 33, no. 6, pp. 682–696, November 2003.

[15] S. N. Srirama, M. Jarke, W. Prinz, and H. Zhu, "Scalable mobile web service discovery in peer to peer networks," in *IEEE Third International Conference on Internet and Web Applications and Services (ICIW 2008)*. IEEE Computer Society, 2008, pp. 668–674.

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

[17] A. Mocan, E. Cimpian, M. Stollberg, F. Scharffe, and J. Scicluna, "Wsmo mediators," Online, December 2005, 10.12.2009. [Online]. Available: http://www.wsmo.org/TR/d29/

[18] Apache Software Foundation, "Apache ServiceMix," 2007, 10.12.2009. [Online]. Available: http://incubator.apache.org/servicemix/home.html

[19] R. Ten-Hove and P. Walker, "Java^{TM} Business Integration (JBI) 1.0 -JSR 208 Final Release," Sun Microsystems, Inc., Tech. Rep., August 2005.

[20] M. Armbrust et al., "Above the clouds, a berkeley view of cloud computing," University of California, Tech. Rep., Feb 2009.

[21] Dustin Amrhein et al., "Cloud computing use cases," A white paper produced by the Cloud Computing Use Case Discussion Group, Tech. Rep. Version 2.0, October 2009.

[22] Amazon Inc., "Amazon elastic compute cloud (amazon ec2)," Online, 10.12.2009. [Online]. Available: http://aws.amazon.com/ec2/

[23] Google Inc, "App engine java overview," 10.12.2009. [Online]. Available: http://code.google.com/appengine/docs/java/overview.html

[24] Eucalyptus Systems Inc., "Eucalyptus," Online, 11.12.2009. [Online]. Available: http://www.eucalyptus.com

[25] S. N. Srirama, O. Batrashev, and E. Vainikko, "Scicloud: Scientific computing on the cloud," in *10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2010)*. IEEE Computer Society, 2010, p. 579.

[26] S. N. Srirama, "Scientific computing on the cloud (scicloud)," Online, 10.12.2009. [Online]. Available: http://ds.cs.ut.ee/research/scicloud

[27] W. Tarreau, "Haproxy architecture guide, version 1.1.34," Online, January 2006. [Online]. Available: http://haproxy.1wt.eu/download/1.3/doc/architecture.txt

[28] Apache Software Foundation, "Apache module mod_proxy_balancr," Online, uRL last visited on 10th Dec 2009. [Online]. Available: http://httpd.apache.org/docs/2.2/mod/mod_proxy_balancer.html

[29] J. Nielsen, *Usability Engineering*. San Francisco: Morgan Kaufmann, 1994.

[30] M. A. Chatti, S. N. Srirama, I. Ivanova, and M. Jarke, "The mobilehost colearn system: Mobile social software for collaborative learning," *International Journal of Mobile Learning and Organisation (IJMLO), Special Issue on: "Developing Themes in Mobile Learning"*, vol. 4, no. 1, pp. 15–38, 2010.

[31] M. Spaniol, R. Klamma, H. Janen, and D. Renzel, "LAS: A lightweight application server for mpeg-7 services in community engines," in *Int. Conf. on Knowledge Management (I-KNOW)*, 2006, p. 592.

[32] Y. Cao, M. Spaniol, R. Klamma, and D. Renzel, "Virtual campfire - a mobile social software for cross-media communities," in *International Conference on New Media Technology and Semantic Systems (I-Media'07)*, September 2007.

[33] W. Tarreau, "Making applications scalable with load balancing, revision 1.0," Sep 2006. [Online]. Available: http://1wt.eu/articles/2006_lb/