m-Roam: A Service Invocation and Roaming Framework for Pervasive Computing

Alvin Chin
Dept. of Computer Science
University of Toronto
achin@cs.toronto.edu

Kostas Kontogiannis
Dept. of Electrical and Computer Engineering
University of Waterloo
kostas@swen.uwaterloo.ca

Abstract

This paper proposes an architectural framework for integrating services within an enterprise and accessing them from mobile devices in a pervasive-computing environment. Present network environments are prone to failures from disconnections and device crashes. Being able to maintain service transaction while moving to different locations and wireless networks (known as service roaming), is currently a major objective in pervasive and mobile computing research. This paper presents a framework (called m-Roam) that allows for mobile clients to perform service invocation and roaming. The framework addresses this problem by introducing a proxy-based architecture so that invocation context and transaction state can be maintained or updated accordingly, while the user roams in different locations.

1. Introduction

Today, there are an increasing number of users that carry mobile devices such as PDAs, laptops and cell phones. As a result, existing services and applications within an enterprise need to be extended towards these mobile users, so that they can access such plethora of corporate services and data. To date, there is no uniform protocol that enables these types of devices to access data and services on an enterprise network. Special mobile application servers such as Avantgo, which allow mobile devices to hook into an enterprise environment to access corporate resources like e-mail and RDBMSes [15], use proprietary technology and require significant resources to maintain and run. Users need to install special software on each device and configure the mobile application server to recognize and communicate with that device. Instead, users should be able to easily access a service within an enterprise or have services delivered to them when entering a particular area like a meeting room. The ability to obtain services and information from an environment anywhere at anytime is part of *pervasive computing* [17]. Users should be able to access their own personalized services on any type of device and with any type of wireless technology.

The challenge this paper aims to address is to determine how to deliver these services in a transparent, seamless, and customizable fashion, to the mobile client. Mobile clients have much lower resource constraints in terms of memory, processor speed, graphics resolution and bandwidth. To extend these enterprise services into a dynamic mobile environment, a mobile services framework needs to be developed. This enables any new mobile device to be easily integrated into the enterprise environment by following a uniform protocol. In mobile computing, wireless networks are unreliable and operate slower than wired LANs. Additionally, devices and users are not stationary, but are highly mobile. Furthermore, devices and networks are prone to failure and disconnection when migrating to different networks like Bluetooth, GPRS, 802.11, and CDMA. The complexities and problems of accessing enterprise services in a wireless environment, should be insulated from the user. A user should be able to access services in the location that the user resides in, and continue to use the same services without interruption when moving to another location or different wireless network.

In this paper, we discuss a software architectural framework called *m-Roam* designed for dynamically executing services, based on location (service invocation) and how a user can roam across different locations and networks (service roaming). Section 2 presents state of the art background and related work. Section 3 highlights the user scenario for the basis of our work. Section 4 outlines the requirements that we identified for developing our software architecture. Section 5 explains the current problems and issues in service invocation and roaming. We describe our design architecture in Section 6. Section 7 details the design of m-Roam according to

Proceedings of the 18th International Conference on Advanced Information Networking and Application (AINA'04)
0-7695-2051-0/04 $ 20.00 © 2004 IEEE

our requirements and addressing the problems and issues identified from Section 4. Section 8 describes future work and we conclude the paper in Section 9. Finally, references used in the paper are found in Section 10.

2. State of the Art

Previous work has addressed roaming at the physical and network layers. Roaming at the physical layer involves measuring the strength of the radio signal from the base station to the device, and determining if handoff occurs. Network-layer roaming involves performing seamless translation of network addresses between a home network and a foreign network, without being exposed to the application. Most applications use IP as the network communication protocol, thus roaming requires having to switch between IP addresses in different networks. Protocols such as MobileIP [13], Reverse Address Translation (RAT), multicast-based handover, HAWAII and CellularIP [9] do this. However, IP layer approaches are not sufficient to solve the problem of service roaming because they do not handle application state which service roaming requires. Additional infrastructure is required to create this transparency of change in IP address. For example, MobileIP requires a home agent and a foreign agent.

Since service roaming occurs at the application layer, it is advantageous to combine this with network-layer roaming to create a hybrid layer that uses the best of both. Wi-Fi Bridge [3] is one example of this which creates middleware at the application layer that enhances the protocol stacks for the mobile host and the gateway using CellularIP between 802.11b and GPRS networks and IP tunneling, similar to MobileIP. Takasugi et al. [16] create a seamless service platform on this hybrid layer by adding an overlay network at the application layer which interfaces to the transport and network layers using seamless proxies (S-proxies) for each node on the network. The same problems as in network-layer roaming still exist, they require the addition of network infrastructure like home and foreign agents. Complexity is introduced with this overlay network which has to bridge between the application layer and the network layer. Furthermore, these approaches do not take into account location changes when migrations occur, only network address changes.

2.1. Application-layer roaming

There are several approaches to perform roaming at the application layer. A directory can be used to store user information using RADIUS and LDAP [6] so login authentication can determine if the user moved. To hide migrations and disconnections from services, proxies can be used. Yao and Fuchs [18] use a recovery proxy that intercepts and caches client requests and service responses so that upon client reconnection, previous client requests or service responses are retrieved from the proxy. For roaming, they describe their client migration protocol involving transferring messages from the old proxy to the new proxy [19]. An extension of Yao and Fuchs' work to geographically dispersed location and client roaming, is the distributed proxy server system [10]. Here, a distributed proxy server is associated with a geographic location and serves mobile clients within that location. Data objects that are accessed and retrieved from the back-end servers are restored to clients after connection has been re-established in the same or different location. This presents the loss of data and interruption in service. Our work is similar to Yao and Fuchs for their proxy operation and our service roaming architecture is similar to the distributed proxy server system.

However, proxies form a part of the solution to service roaming, because they deal only with transferring data and ignore the user's activity of a service. Clients need to maintain state which is compromised during migrations and disconnections. Sessions provide this higher level of abstraction and the *Migrate* architecture [14] is one such work that does this, where network disconnections are handled as session continuations and connection migrations are handled as session migrations. The SOMA (Secure and Open Mobile Agent) middleware platform [2] extends the notion of a proxy into a shadow proxy by adding session information, which keeps track of the device's actions and maintains the device's state.

To discover services in a network on a mobile device, a framework for mobile web services is proposed in [4] using a mobile agent platform. Mobile agents act on behalf of clients and service agents process the user's request, find the service using UDDI and invoke on the web service using SOAP. However, there is no discussion on migration and service roaming. Our work also uses UDDI and web services for service invocation, but we add a service roaming architecture and protocols.

In addition to handoff of data performed by proxies, state and context information also needs to be transferred in this handoff. *iMASH* [1] achieves this through application session handoff which is based on existing work on proxies, content adaptation and client-awareness. However iMASH requires applications to be modified to support semantic session savepointing so that after migration, clients can obtain the appropriate delivered state. Our Client Migration Protocol is an application session handoff similar to Yao and Fuchs [19], but differs from iMASH in that applications do not

Proceedings of the 18th International Conference on Advanced Information Networking and Application (AINA'04)
0-7695-2051-0/04 $ 20.00 © 2004 IEEE

need to be modified because session savepointing is performed by our system, not the client. In addition, iMASH distinguishes between three types of application session handoff depending on client and middleware server. m-Roam does not distinguish between these, therefore reducing the complexity of the logic for application session handoff.

2.2. Approach and contribution

As seen above, much work in application-layer roaming for mobile devices in wireless networks adapt existing work for transactional systems in wired networks. This approach is suited towards mobile and pervasive systems, because it abstracts away the details of networking, it is agnostic of the type of wireless network and mobile device, and these methods have been proven in the transactional and database community. As such, there is no need to reinvent the wheel in solving the problems of disconnections and crashes, therefore we leverage on this work. They can be applied to the mobile realm with the addition of migration for roaming. Since transactional approaches work at the application layer, they are flexible and thus suited towards solving our problem. Our contribution to this area is the integration of proxies, sessions, application session handoffs and web services into a framework for solving service invocation and roaming, which is new and novel.

3. User Scenario

A typical scenario is that of a user who wishes to perform a particular task by searching for services in the location that the user is in. The user enters a query on a mobile device (like a cell phone, PDA or laptop) equipped with a wireless interface (such as Bluetooth, GPRS or 802.11) and then a list of services is delivered that satisfy the query. From here, the user can select a service and then execute operations on that service. After, the user can issue other queries to perform other tasks. During this time, the mobile device may get disconnected, the mobile device may crash, the user may use another device, or the user may move to a different location and/or a different wireless network. Upon recovery from any of these interruptions, the user can continue the service from where the user left off without having to start over. This scenario is realized with our system architecture, specifically our software architectural framework called *m-Roam*, which is the middleware that performs the service invocation and roaming.

4. Requirements

We identified six requirements that drove the rationale in developing our software architectural framework. First, users should be able to execute services on any client device that runs any operating system on any network. This requirement is derived from the vision of pervasive computing. Second, finding services on the network requires the necessity of a service discovery protocol (SDP) that is flexible and is widely used. Third, service interoperability is an important requirement to achieve so that services can interact with each other and with clients in a uniform manner. Fourth, related to this, is the easy integration of services into our framework. This is especially true for enterprises that wish to enable their services for use in a mobile and roaming environment, without having the hassle to drastically change their implementation and interface. Fifth, the software architectural framework must be scalable to support many clients and services without any significant degradation in performance. Finally, a sixth requirement is location awareness in which services are tailored to the location that the user is in. This is important because this provides focused service discovery that elicits a response that is customized to the user's environment.

5. Service Invocation and Roaming

To accomplish the user scenario described in Section 3, the problem that we are addressing is how to perform service invocation and roaming that requires minimal infrastructure changes. Service invocation refers to executing operations on a service in order to perform a particular task. We define a service as an entity that provides a group of related functions with a specific purpose and is exposed as a well-defined interface. An example of a service is e-mail. Service roaming involves having the ability to continue the service while migrating from one location or network to another.

The current problems and issues in this area are explained below, and addressed in our architecture.

1) Services are exposed using proprietary and/or disparate service discovery protocols

Service discovery protocols such as Jini, SLP, Salutation and UPnP are mostly proprietary and do not interoperate well with each other. A possible solution to this problem is to create bridges which perform the transformation from one SDP to another. However, this requires an additional component (bridge) which complicates the process. There is a risk that the target SDP may not be exactly the same in functionality as the

Proceedings of the 18th International Conference on Advanced Information Networking and Application (AINA'04)
0-7695-2051-0/04 $ 20.00 © 2004 IEEE

original. As well, a front-end service interface needs to be created to expose the services to clients, and most service interfaces are language dependent (for example, Jini requires the use of Java).

2) *Service paradigm does not address mobility issues such as disconnections and crashes*

When the client disconnects or crashes, the connection that was established between the client and the service is interrupted. As such, after the client reconnects to the service, a new connection is established and the client has to restart the service from the beginning. The goal is to insulate these failures from the client to maintain transparency and provide seamless interaction of services.

3) *Performing service roaming and handling user migrations*

There are many approaches in order to accomplish roaming. The most popular and used by cellular WAN technologies like GSM is physical layer roaming, which operates at the Physical layer of the network OSI model and involves base station handover. The second is network layer roaming which uses a signaling protocol that operates at the Network layer of the network OSI model like MobileIP. The third approach is application layer roaming which uses middleware and application level signaling to detect that the client is roaming, and operates at the Application layer of the network OSI model.

4) *Location awareness in service invocation and roaming*

Several questions come to mind like how to specify location in the service discovery protocol and in the client query, as well as the type of infrastructure that is required to support location-based services.

6. Design Architecture Overview

Understanding the requirements from Section 4 and the issues from the previous section, we provide a high-level overview of our design architecture to realize the user scenario from Section 3.

6.1. System architecture

The high-level system architecture is illustrated in Figure 1. The wireless client connects and authenticates to the enterprise network via a wireless access point. Once the client is successfully connected, the user logs in to m-Roam using an HTTP web browser, by sending a login request. After the user is authenticated by m-Roam, the user can start to use the services offered in this location.

The user enters a client query to perform a particular task by executing a client query request. For example in the context of a conference scenario, a client query could be to "find a service that can provide today's conference agenda in the conference lounge area". The client query is in the form of "find a service or list of services that provide features A, B, C.. in location L". We create this query as an HTTP request and append a location attribute to the request URL, because it is simple to process. This satisfies issue 4 from Section 5 addressing how to specify location in the client query. From these requests, m-Roam discovers for services that match the query request. Services in the enterprise are exposed to m-Roam as web services. Web services are services that are encapsulated into interfaces that can be accessed using HTTP for transport and XML for data communications. We choose web services for service invocation because it is standardized by the W3C, and they separate the content from the display using standardized protocols (HTTP and XML). As such, the display at the client is just rendered in HTML such that any client with an HTTP browser can be used. This satisfies requirement 1 (services can be executed on any client) and requirement 2 (flexible, widely used SDP). Since we adopt the notion that all services are encapsulated as web services and the service discovery protocol is UDDI [7], then service interoperability disappears thus satisfying requirement 3. Therefore, we solve the problem of using proprietary and/or disparate service discovery protocols (issue 1 from Section 5).

Services are described in WSDL [7] and are advertised to m-Roam using UDDI during service registration. Once the desired services are found, the user can select the service and then execute operations on that service using SOAP [7], by having m-Roam make service operation calls directly on its corresponding web service. The service executes the requested operation and returns the result back to its web service. A service operation response is generated by the web service and is forwarded to m-Roam, which then processes it and relays it back to the wireless client as the response to the original client query.

Proceedings of the 18th International Conference on Advanced Information Networking and Application (AINA'04)
0-7695-2051-0/04 $ 20.00 © 2004 IEEE

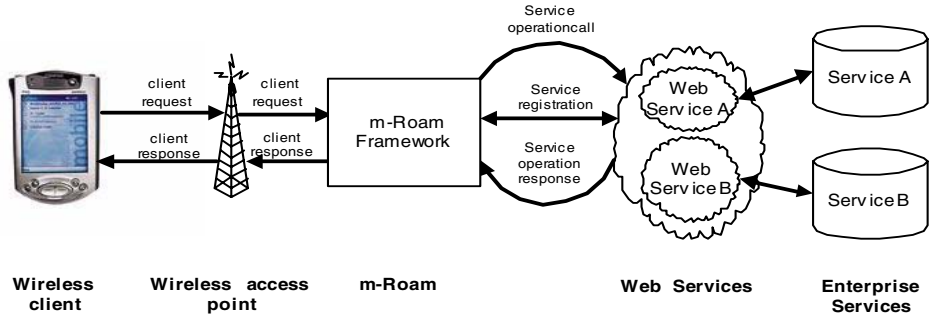

Figure 1. High-level system architecture

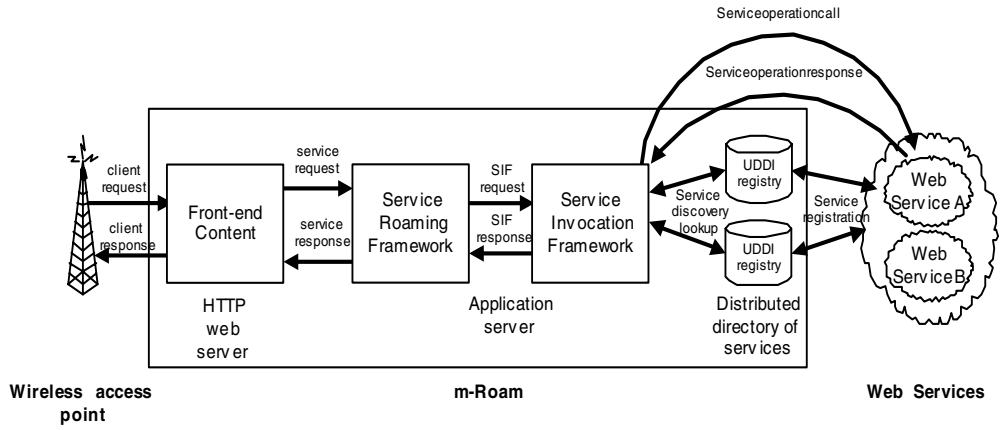

Figure 2. Architecture of m-Roam

If at any point in time, the client experiences a network disconnection, crash or a move to a different location and possibly a different type of network, then m-Roam is able to recover from the service that the user had previously accessed, and continue using it. After the user has finished with m-Roam, a logout request is sent in the client request. We explain the underlying details of the m-Roam architecture in the next subsection.

6.2. m-Roam architecture

m-Roam is responsible for processing client requests originating from client queries, performing service invocation and roaming logic, hiding disconnections and crashes, and delivering service responses to wireless clients. The detailed architecture of the m-Roam framework represented in Figure 1, is shown in Figure 2 and consists of four main parts: *Front-end Content*, *Service Roaming Framework*, *Service Invocation Framework* and *Distributed Directory of Services.*

Front-end content. The Front-end Content acts as the front-end interface to the Service Invocation and Roaming Framework, and is responsible for processing HTTP client requests which can be either a service query request (to find a particular service or select a service) or a service operation request (to invoke an operation on a service). From here, the HTTP service requests are forwarded to the Service Roaming Framework. Once the service response is received, the Front-end Content formats the response using the content repository and delivers it back to the wireless client's web browser.

Service roaming framework. The Service Roaming Framework consists of the logic needed to perform roaming, brokers client requests (service requests) and service responses, and insulates client disconnections, crashes and migrations from the Enterprise Services.

Service invocation framework. The Service Invocation Framework processes the service requests relayed from the Service Roaming Framework (designated as SIF requests), and discovers available services that satisfy that request based on authorization and location (returned as SIF responses).

Distributed directory of services. The Distributed Directory of Services provides a repository for the description, operation and lookup of back-end enterprise services and uses UDDI registries. Services are described using WSDL, published in the UDDI

Proceedings of the 18th International Conference on Advanced Information Networking and Application (AINA'04)
0-7695-2051-0/04 $ 20.00 © 2004 IEEE

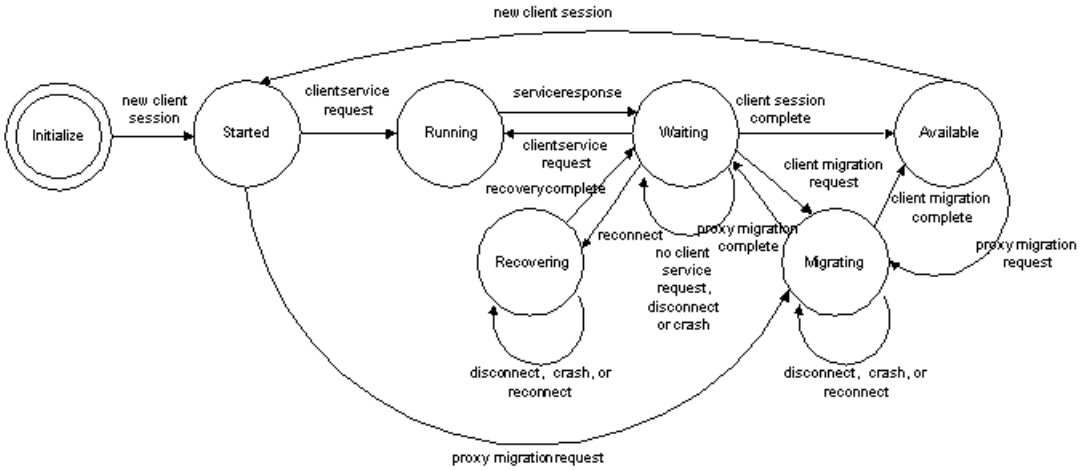

Figure 3. Proxy instance finite state machine

registry, discovered using UDDI, and executed using SOAP by the Service Invocation Framework.

7. Detailed Design of m-Roam

In m-Roam, we focus on solving the research issues involved in service invocation and roaming. We use the requirements from Section 4 to justify our design. Two components are used in m-Roam that collectively perform seamless service invocation and roaming: *session* and *proxy*. Together, they are fundamental in addressing disconnections and crashes, migrations and user queries.

When interacting with m-Roam, the system needs to associate certain activities for a specific client, so that when disconnections, crashes or migrations occur, the last activity performed can be continued without interruption. As a result, each user is assigned a session which is defined as user activity from the time that the user first logs on to the system until the time that the user successfully logs off.

Central to service roaming is the introduction of a middleware recovery and roaming proxy, similar in concept to that of an HTTP proxy for HTTP requests on the web. A proxy is assigned to each client in a particular location, and tracks service requests and responses throughout the lifetime of the client session inthat location by caching client requests and service responses. This lifetime is modeled as a proxy instance finite state machine, and is illustrated in Figure 3.

- Service invocation

Clients make queries for a particular service or for a list of services in two ways: 1) search for all services in this location or, 2) find service(s) that match specified user criteria in the form of "find a service or list of services that provide features A, B, C,... in location L". This service search query is processed as an ECA (Event-Condition-Action) rule [5] by the Service Invocation Framework which performs service discovery in the UDDI registries for the appropriate web service. The ECA Rule Engine returns the service(s) that match the service search query [5]. Once the web service is found from the service search query, clients can invoke operations on the web service (service operation call) using SOAP operation requests. SOAP (Simple Object Access Protocol) is a protocol for running operations on a remote web service using XML over HTTP. A SOAP response is returned from the web service that is embedded in an HTTP service response [7].

- Disconnections and crashes

In order to insulate client disconnections and crashes from the service, the proxy assigned to a client is used to cache client requests and service responses. When the client reconnects, the existing client session is recovered. The process that illustrates this can be shown from Figure 3. While the client is accessing services, the proxy is in the *Running* state and once the service response is received, it goes into the *Waiting* state, awaiting further service requests. When the client disconnects or crashes, the proxy is unaware of this and therefore still remains in the Waiting state. Just after the client gets reconnected, the proxy moves into the *Recovering* state where it has to retrieve the last saved client request and service response. Once recovery is complete, the proxy goes into the Waiting state.

Proceedings of the 18th International Conference on Advanced Information Networking and Application (AINA'04)
0-7695-2051-0/04 $ 20.00 © 2004 IEEE

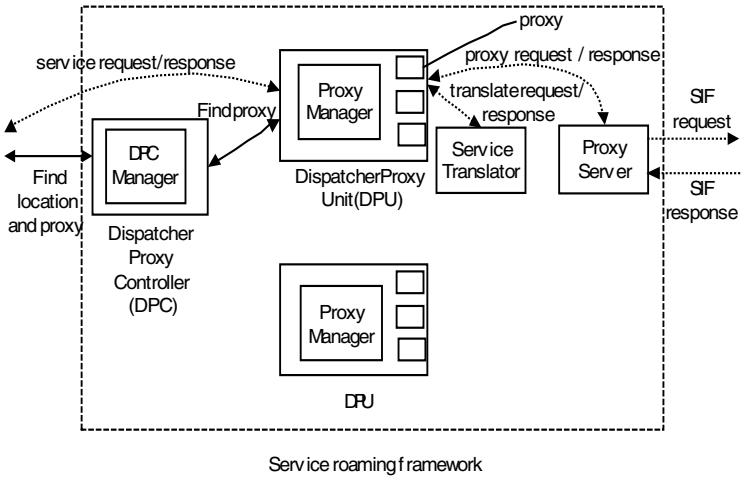

Figure 4. Service Roaming Framework

- Service roaming

Each location in the system has a Service Roaming Framework instance which can reside on a single server or can be distributed within the network. This architecture is illustrated in Figure 4.

When a client enters a new location, the system needs to locate a proxy to be allocated to the client in this location. This is performed by the *Dispatcher Proxy Controller* (DPC). If the system detects that the client has an existing session, then the client is in roaming mode and migration of service state and data from the previous location needs to be transferred. Migration is coordinated between the old and new DPCs using the Client Migration Protocol, and functions like a handover in physical layer roaming. The Client Migration Protocol operates as follows. From Figure 3, the proxy in the new location receives a migration request and enters the *Migrating* state. In this state, the proxy in the new location must notify the proxy from the previous location to transfer all proxy messages from the proxy queues over. For the old proxy in the previous location, its state moves from Waiting to Migrating, whereas for the new proxy its state moves from either Started or Available to Migrating. During the migration process, the client could disconnect, crash or recover from those failures. In either case, the proxy remains in the Migrating state since it has to finish the migration process. Once the migration is complete, the old proxy is made available for other clients to use so it enters the Available state. The new proxy can now service requests for the client in the Waiting state. After, m-Roam retrieves the last saved service response from this proxy or updates it, depending on whether the last saved service request is a location-dependent query. The retrieval process follows the recovery from disconnection or crash mentioned earlier.

To address scalability (requirement 5), multiple proxies are controlled by a *Dispatcher Proxy Unit* (DPU). The DPU locates an available proxy that can be used by this client for the duration that the client is in this location. Multiple DPUs are controlled by the DPC. The *Proxy Manager* maintains the proxies, keeps track of client state for all proxies, sends service requests to the Proxy Server, sends service responses from the Proxy Server to the client, and recovers from disconnections and crashes.

8. Future Work

Currently, the system only allows for keyword-based search queries for services. One area for future work is to extend our system for more intelligent search queries that involve a natural language query, and use DAML-S and ontologies [8,11] to add these semantics. In this case, UDDI would have to be extended to allow for DAML-S ontologies to be added. This would involve having to map DAML-S attributes to UDDI records, and creating a DAML-S matchmaking engine that would locate the appropriate service directly using a DAML-S query or by translating the DAML-S query into a UDDI query on the UDDI registry [12]. This will allow users to perform more accurate and useful semantic queries, in addition to keyword matching with UDDI and for the personalization of services according to location.

Another area for future work is to support user profiles which would have user preferences like a user's quality of service for a particular service, and the types of services that a user would want to access, based on

Proceedings of the 18th International Conference on Advanced Information Networking and Application (AINA'04)
0-7695-2051-0/04 $ 20.00 © 2004 IEEE

the context of where the user is located and what the user is doing. These user profiles can be integrated into an enterprise-wide network through the existing use of LDAP (Lightweight Directory Access Protocol). Security in our architecture and framework is very simple through the use of a user name and password, and access to services through a simple service authorization table. RADIUS can be employed to support user authorization and authentication for enterprise security.

9. Conclusion

In this paper, we have proposed an architectural framework called m-Roam that allows a user to continue using a service without interruption, despite network and client disconnections, client crashes, and user migration within and between networks. We have achieved this by creating a distributed proxy-based architecture that accommodates for location. The architecture was implemented in a prototype system and we have solved various issues in service invocation and roaming with our architecture. This architecture is by no means complete, and due to its openness, additional plug-in components can be inserted in order to extend and improve its functionality and performance. With the proliferation of many mobile devices, wireless networks and the rapid adoption of them, systems like m-Roam will assist users to roam for services, thus making it one step closer to achieving Weiser's vision.

10. References

[1] R. Bagrodia et al. "iMASH: Interactive Mobile Application Session Handoff". In *Proceedings of MobiSys 2003: The First International Conference on Mobile Systems, Applications, and Services*, The USENIX Association, 2003, pp. 259-272.

[2] P. Bellavista, A. Corradi, and C. Stefanelli, "The Ubiquitous Provisioning of Internet Services to Portable Devices". *IEEE Pervasive Computing*, July-September 2002, pp.81–97.

[3] A. Calvagna et al. "WiFi Bridge: Wireless Mobility Framework Supporting Session Continuity". In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)*, IEEE, 2003, pp. 1-8.

[4] S. Cheng, J. Liu, J. Kao, and C. Chen, "A New Framework for Mobile Web Services". In *Proc. 2002 Symposium on Applications and the Internet (SAINT) Workshops*, January 28-February 1, 2002, pp 218–222.

[5] K. H. Cheung, A Customizable Web Services Integration Environment. Master's thesis, University of Waterloo, 2002.

[6] C. Corbi and G. Sisto. "A Directory Enabled Solution for Internet Roaming". In *Proc. IEEE International Symposium on Computers and Communications*, IEEE, 1999, pp. 39-45.

[7] F. Curbera et. al. "Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI". *IEEE Internet Computing*, March-April 2002, pp. 86–92.

[8] DAML Services. DAML Coalition, http://www.daml.org/services/.

[9] A. Festag, H. Karl, and G. Schafer. "Current Developments and Trends in Handover Design for ALL-IP Wireless Networks". Technical Report TKN-00-007, Telecommunication Networks Group, Technical University Berlin, 2000.

[10] K. Kim et al., "A Distributed Proxy Server System for Wireless Mobile Web Service". In *Proc. 15th International Conference on Information Networking (ICOIN'01)*, 2001, pp. 749–754.

[11] S. A. McIlraith, T.C. Son, and H. Zeng, "Semantic Web Services". *IEEE Intelligent Systems*, March-April 2001, pp. 46-53.

[12] M. Paolucci et al., "Importing the Semantic Web in UDDI". In *Proc. Workshop on Web Services, e-Business, and the Semantic Web (WES): Foundations, Models, Architecture, Engineering and Applications, The Fourteenth International Conference on Advanced Information Systems Engineering (CAiSE'02)*, Toronto, Canada, May 27-28, 2002, pp. 225–236.

[13] Charles E. Perkins. Mobile Networking Through Mobile IP. IEEE Internet Computing Online, http://www.computer.org/internet/v2n1/perkins.htm.

[14] M. Snoeren. "A Session-Based Architecture for Internet Mobility". PhD thesis, Massachusetts Institute of Technology, 2003.

[15] V. Stanford, "Pervasive Computing Goes To Work: Interfacing to the Enterprise", *IEEE Pervasive Computing*, July-Sept 2002, pp. 6-12.

[16] K. Takasugi et al. "Seamless Service Platform for Following a User's Movement in a Dynamic Network Environment". In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03)*, IEEE, 2003, pp. 71-78.

[17] M. Weiser, "The Computer for the 21st Century". *IEEE Pervasive Computing*, January-March 2002, pp. 18–25.

[18] B. Yao and W. K. Fuchs, "Proxy-based Recovery for Applications on Wireless Hand-held Devices". In *Proc. 19th IEEE Symposium on Reliable Distributed Systems (SRDS'00)*, October 16-18, 2000, pp. 2–10.

[19] B. Yao and W.K. Fuchs. "Recovery Proxy for Wireless Application". In *Proc. 12th International Symposium on Software Reliability Engineering (ISSRE 2001)*, IEEE, 2001, pp. 112-119.