MDE-URDS – A Mobile Device Enabled Service Discovery System

Ketaki A. Pradhan*, Lahiru Sandakith Gallege*, Alfredo Moreno+, Rajeev R. Raje*

* Department of Computer and Information Science, *Indiana University Purdue University Indianapolis, USA, {ketpradh,lspileth,rraje}@cs.iupui.edu*

+ *Universidad de Granada, Departamento de Lenguajes y Sistemas Informáticos, Spain, amoreno@ugr.es*

***Abstract*—Service Discovery is an important step during the creation of distributed systems made out of independently developed and deployed and quality-aware components. Despite of many prevalent approaches, the task of service discovery that encompasses components running on resource constrained devices does not yet have a satisfactory solution. Here, we present a service discovery framework which is specially designed to include resource constrained devices and uses the Mobile IP protocol. A preliminary prototype of this framework, along with associated experiments, is described in this paper.**

***Keywords* — Resource Constrained Devices, Service Discovery, UniFrame.**

I. INTRODUCTION

Service Discovery has been a well studied field in recent times. Its wide spread applicability, before the composition of independently developed and deployed components, makes it a critical activity before creating a distributed system. This is evident from many alternatives such as Jini[1], UPnP[2], and SLP[3]. A comprehensive report[4], created by the researchers from the National Institute of Standards and Technology (NIST), compares these "first-generation" discovery systems, suggests alternative architectures for service discovery including two-party and three-party architectures, and indicates challenges for developing the "next generation" of service discovery systems. One of the main challenges that this NIST report mentions is the incorporation of mobile and resource constrained devices into the service discovery systems. This research aims to address this specific challenge. As an initial step, in this paper, we describe the incorporation of the principles of the Mobile IP[5] into the UniFrame Resource Discovery System (URDS)[6], to create a mobile device enabled discovery system. The URDS, which is developed by our research group in past, is a hierarchical and proactive Service Discovery System for locating heterogeneous components. This enhanced version, called MDE-URDS (Mobile Device Enabled URDS), is created by incorporating certain conceptual changes in the URDS architecture to handle the limitations of these mobile devices. Since mobility is the key characteristic of mobile devices, this paper focuses only on it.

We describe the related work in the Section II, the design of MDE-URDS in the Section III, the associated experimentation and empirical validation in the Section IV and the conclusions, along with future work, in the Section V.

II. RELATED WORK

There have been several service discovery architectures proposed by academia and industry. These architectures can be divided into two groups: a) Lookup Services, and b) Discovery Services.

Lookup services, such as Jini[1], UPnP[2], CORBA Trader Service[7], and UDDI[8], to name a few, have a central registry that maintains a

list of services in the system and a lookup service searches that registry for the required services when a client requests for them. A lot of these alternatives operate in a homogeneous environment. On the other hand, the category of Discovery Services includes specific architectures for resource discovery. SLP[3] is one such example that provides the service discovery with the help of the User Agents, Service Agents, and Directory Agents forming three-party architecture. Also, not many of these prevalent "first generation" approaches from both these categories take into account the need for incorporating mobile devices in the service discovery process, a shortcoming highlighted in the NIST report[4].

FRODO[9] is a recent approach that includes mobile devices in the service discovery process by categorizing them based on their capabilities of energy (power) and mobility. The device with better processing power is generally selected as the central registry where the services register and the ones with lesser power act as clients and services. However, their framework is designed for a particular home environment, without considering the other domains where the mobile entities could migrate to. There has also been research in the field of location-based service protocols and researchers have proposed architectures[10] based on the principles of Mobile IP. However, this approach does not include the concepts of Home Agents and Foreign Agents to manage the mobility. M-URDS[11] is another extension of the URDS architecture that uses mobile agents to discover new components. It, however, does not consider the incorporation of mobile devices in the URDS framework.

Our work differs from these approaches, as it addresses the key issues of heterogeneity (due to being an extension of the URDS) and mobility associated with the resource constrained devices (by incorporating the principles of Mobile IP) in the service discovery process.

III. OUR SOLUTION – MDE-URDS

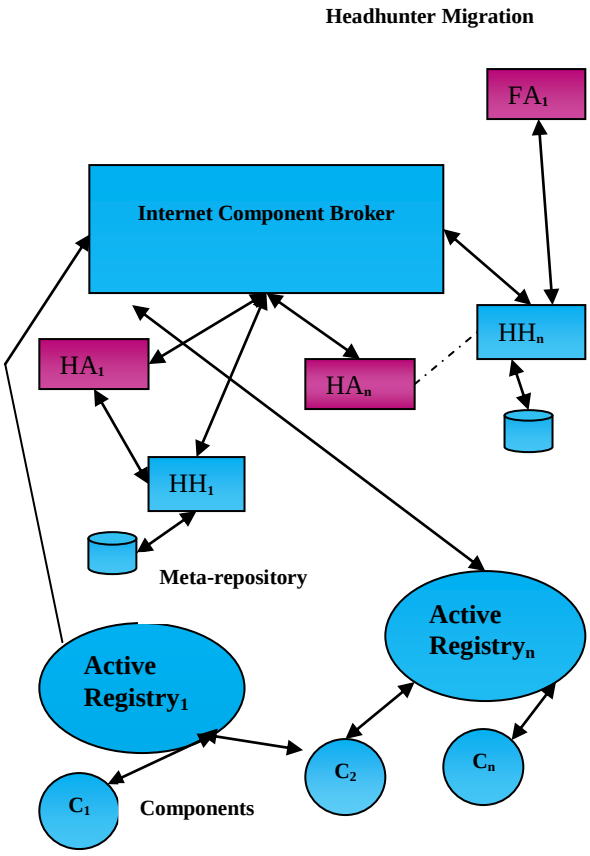

Fig 1: MDE-URDS Architecture
Diagram Key: HH = Headhunter, HA = Home Agent, FA = Foreign Agent,
C_1, C_2, …, C_n = Software Components.

The architecture of MDE-URDS is shown in Figure 1. In this figure, the components colored blue are the existing components of the URDS and the ones in the pink color are the ones added for including the Mobile IP principles. As indicated earlier, MDE-URDS is an enhanced version of URDS[6] and hence, below we provide a brief overview of URDS. More details of URDS are available in a previous paper[6]. The main components of the URDS are the Internet Component Broker (ICB), Headhunters, and Active Registries. The ICB, which is similar to the Object Broker in CORBA[7], consists of the following entities:

Domain Security Manager (DSM) which is responsible for maintaining the information about all the entities in the system and provides that information to only authorized entities, such as Headhunters.

Query Manager (QM) which is responsible for making queries on behalf of the client of the URDS and getting the results from the Headhunters.

Adapter Manager (AM) is used for handling heterogeneity by providing the necessary adapter components.

Link Manager (LM) links several of such ICBs together to create a federation of discovery services.

Headhunter is one of the important entities of the system which actively participates in the discovery process of software components and performs the matching of the available components with the input queries. Headhunters also have associated meta-repositories.

Active Registries are responsible for looking out for new components entering in the discovery system and registering them. It is from the active registries that the Headhunters update their meta-repositories.

We have extended the original URDS architecture by making some modifications to the architecture. First, we divided the resources, similar to the FRODO**Error! Bookmark not defined.** approach, into two categories: resourceful and resource-constrained devices. Then, we identified a mapping scheme for placing the entities of the URDS into these two types of resources. This mapping was performed based on heuristics about the functionality of each URDS entity and associated empirical evaluations. Table 1 indicates the result of the mapping process.

URDS Component	Resourceful device	Resource-constrained device
Domain Security Manager	Yes	No
Query Manager	Yes	Yes
Link Manager	Yes	No
Adapter Manager	Yes	No
Headhunter	Yes	Yes
Active Registries	Yes	Yes
Clients	Yes	Yes

Table 1: Mapping of URDS Components on the different categories of devices.

After the mapping scheme was finalized, the next issue was to handle the mobility of devices and associated entities deployed on them in the design of the MDE-URDS. As mobility is the key factor here to be considered, there must be a mechanism for the mobile entities to move across the network, of whose MDE-URDS is going to be a part of, and still be able to achieve the discovery functions. This is achieved by the use of the principles of Mobile IP in the form of Home Agents and Foreign Agents (as indicated in Figure 1).

Mobile IP is a protocol that provides transparent access to mobile entities moving across different domains. For every mobile entity, there is an associated Home Agent that is responsible to route the messages to the entity, irrespective of its location. This task is simple when the mobile node is in the same domain where its Home Agent is, which is called its “home world” and the messages can be directly sent to it. However, when the mobile node moves away from its home world, it registers with a new world with the help of another agent called as the Foreign Agent, whose task is to inform the corresponding Home Agent about the presence of the mobile entity and route the messages to that entity. So, any message sent to the mobile entity is now routed from its Home Agent to the Foreign Agent and then to the mobile entity. The mobile node is free to move across any new foreign world having a Foreign Agent which would inform its Home Agent about its current location and the appropriate routing of messages to the node would take place.

In the context of MDE-URDS, the query propagation to the headhunters on the mobile devices takes place as follows: if the mobile node is in its home world, then the query is sent directly to it, whereas when the mobile node is in any foreign world, then the query is routed from its Home Agent to its Foreign Agent and then to the mobile headhunter. With the help of Mobile IP, even the headhunters from different domains can be queried and a variety of components can be obtained for the client to select from. Therefore, we can have three scenarios for propagating queries to the mobile headhunters: a) when the headhunter is in its home world, b) when the headhunter is connected to an external foreign world, and c) when the headhunter is not connected to any world or is in transit.

From Table 1, we conclude that other components such as the registries and clients can be deployed on resource-constrained devices, however, for our initial setup, we have considered only the headhunters and two active registries deployed on these devices.

IV. EXPERIMENTATION

A prototype of MDE-URDS was created that contained Pharos Traveler GPS PDAs as the resource-constrained devices running Windows Mobile 5 operating system and several desktop machines that use Windows XP operating system. The desktop machines are connected by a LAN and wireless connectivity is provided by the different wireless networks on campus and also a private access point for a constrained experimental setup. The entire MDE-URDS implementation is created using Java RMI, and we have used J9 JVM[12] for running the java classes on the Pharos PDAs. We carried out an empirical scalability study of this prototype by considering different scenarios such as the failure of the mobile devices, the routing of the queries to the mobile device to its appropriate location, and the buffering of the queries when the mobile device is unavailable. We also divided the experimental setup into two "worlds" – a home world and a foreign world.

For each of these scenarios, we ran several experiments, querying different headhunters for components, moving seamlessly across different worlds (home and foreign) and sending queries simultaneously to them. We had deployed four headhunters on the mobile devices, each having their own Home Agent and moving across three different worlds. The two foreign worlds have a Foreign Agent with which the headhunters register, when connected to that foreign world. We have also considered four stationary headhunters for comparing the responses from headhunters on both types of devices. There are four active registries in the system that search for new components and every headhunter has its own meta-repository that updates from the active registries. For our setup, we have limited the number of components to a total of 15, with the meta-repositories having some overlapping components. We calculated the average query response time for each of the queries for three above mentioned scenarios. In each scenario, the query structure is composed of the *Headhunter name, its IP address, and the type of the component* to be retrieved from the headhunter repository. Thus, the matching semantics used in the following experiments is restricted only to type matching of components. These queries are sent out randomly to the headhunters with proper sequence numbers to provide duplicate filtering and all the queries are independent of each other, i.e., no query has any relation to other queries. Also, in these scenarios, the Headhunters are deployed on the Pharos PDAs and the clients are deployed on resourceful and/or PDAs. In figures 2-6, every point on the graph is the time value obtained from a particular headhunter.

First, we established a base case situation with headhunters deployed only on resourceful devices.

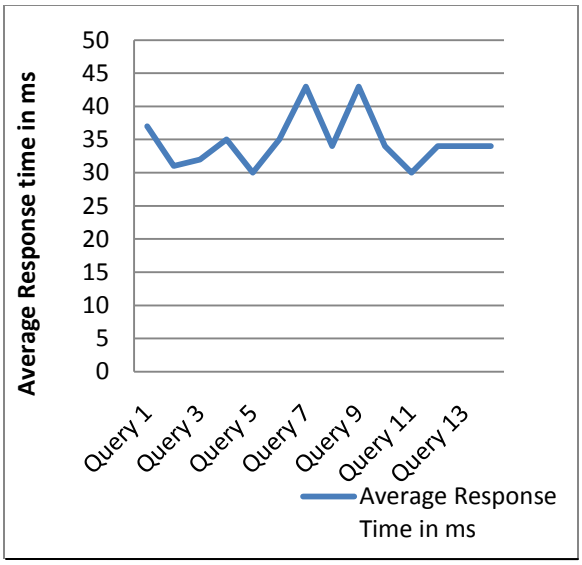

Figure 2: Average Response Time when headhunters are on resourceful devices.

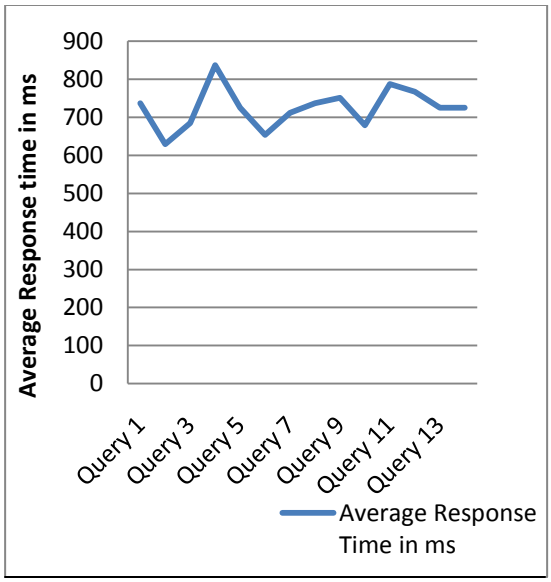

Figure 3: Average Response Time when headhunters are in the same domain

This base case is used for the comparison purposes when the headhunters are deployed on the PDAs. The average response time for this case, as shown in Figure 2, varies between 30 ms to 43 ms.

Scenario 1: Headhunters in home world

Figure 3, shows the average response time over a range of randomly selected queries when the headhunters are deployed on the PDAs and these results are obtained from a subset of headhunters which are in their home world. It is seen that the average response time for these queries varies between 630 ms and 830 ms, when the headhunters are in their home domain. This response time is more compared to the base case, because of the added time spent in communication with these devices, with wireless connectivity. This communication time adds on to overall response time.

Scenario 2: Headhunters in foreign world

The average response time, when the Headhunters are in the foreign world, for random queries, is indicated in Figure 4. Here, the average response time is 1200 ms.

The response time in this case increases due to the rerouting of the queries to the Foreign Agent across different domains and getting the results from the Headhunters.

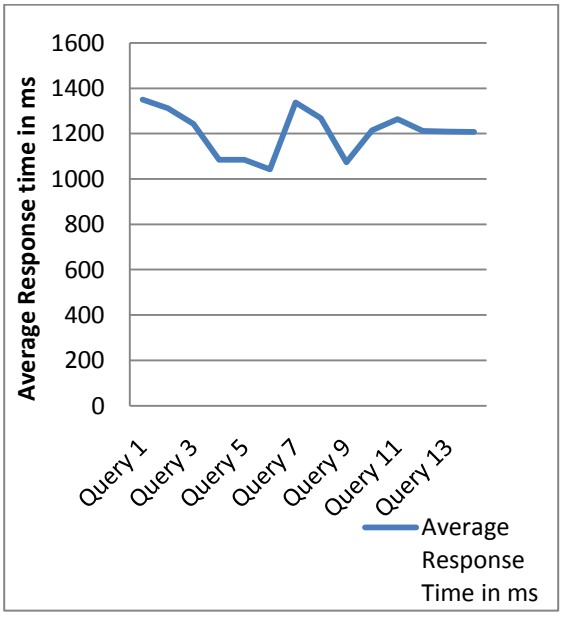

Figure 4: Average Response Time when headhunters are in the foreign domain.

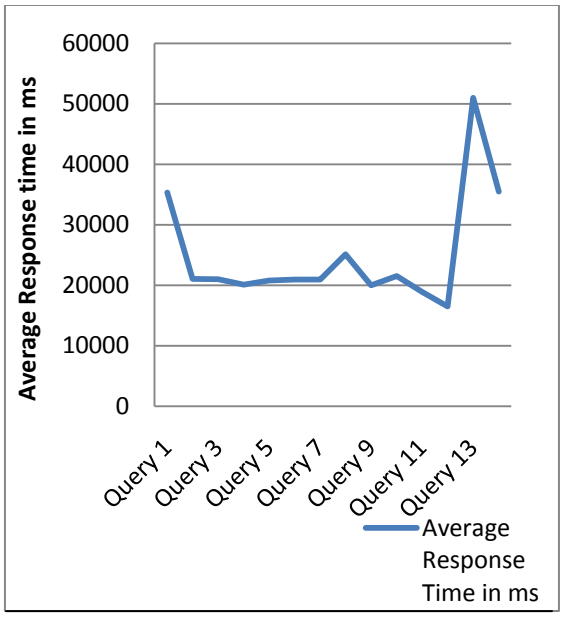

Figure 5: Average Response Time when headhunters are in transit

Scenario 3: Headhunters in transit

Figure 5 shows the case when one Headhunter, randomly selected, is moving across worlds and is not currently connected to any of the Foreign Agents. In such cases, the queries are buffered in a queue with its Home Agent and are sent to the Headhunter when it gets connected to a new world. As seen from the Figure 5, the response time in this case is much higher than the previous two cases, with certain values peaking even up to 50000 ms, which is due to the wait time involved until the Headhunter connects back either to its home world or a foreign world. This high wait time can be reduced by sending the queries across to different headhunters and getting the components, if the user of the discovery service does not expect a response from a specific headhunter.

Figure 6 indicates response times obtained when several headhunters are in transit. The response times in this case vary according to the availability of the other Headhunters in the system and also if they have the required components, thus exhibiting a random behavior. This behavior is because of the freedom given to the headhunters to move randomly across the worlds or not be connected to any world at all, and so at a time the number of headhunters available in the system may not be fixed and not all the headhunters may have the same components, so the query response times varies. The Query Manager has the task of finding a suitable headhunter and retrieving the desired components for the URDS client.

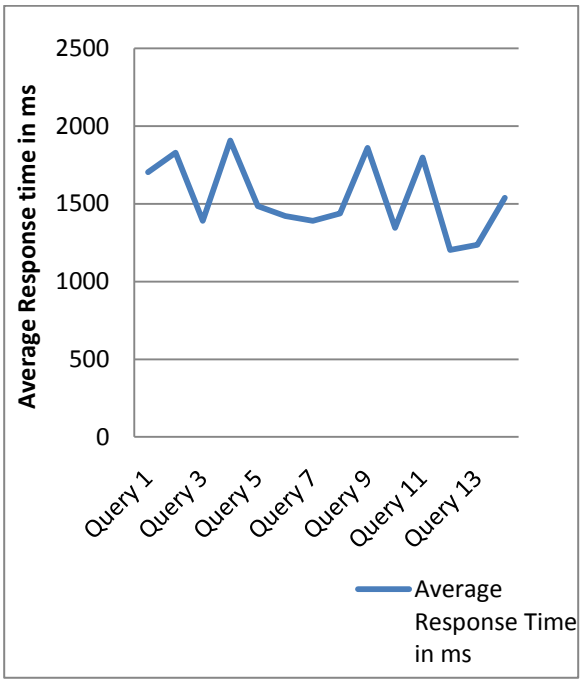

Figure 6: Average Response Time when several headhunters in case of the headhunter is in transit

We can conclude from our experiments that the average response time is higher using the resource-constrained devices as compared to resourceful devices. This is because of the additional time spent communicating with these devices using the pair of the Home Agent and Foreign Agent. Also, the response time increases when the headhunters are unavailable and the clients have to wait for them or query other available headhunters.

V. Conclusion and Future Work

In this paper, we have described the initial results obtained from incorporating resource-

constrained mobile devices in a service discovery system. An empirical validation of the MDE-URDS prototype indicates an increase in the average response time due to the re-routing of queries and results carried out by the Home and Foreign Agents when compared with the URDS containing resourceful devices. Such an increased response time may not be acceptable for designing real-time distributed applications such as wide-area tracking. In that case, then efforts must be made to reduce the response time. This is one area of future research that we will be pursuing. Another direction for our future work is to incorporate uncertainty of the context in the discovery process. Our current work can also be extended to include different levels of service matching which is an extension to the work described in recent reports[13].

References

[1] Sun Microsystems, "Jini™ Architectural Overview",http://pages.cs.wisc.edu/~lhl/cs740/papers/jini-overview.ps, 1999.

[2] Universal plug and Play (UPnP), http://www.upnp.org/, accessed on September 25th, 2009.

[3] Guttman E., Perkins C., Veizades J., and Day M., "Service Location Protocol", Version 2. IETF RFC 2608, 1999.

[4] Dabrowski C., Mills K., and Quirolgico, S., "A Model-based Analysis of First-Generation Service Discovery Systems", NIST, ational Institute of Standards and Technology, Special Publication 500-260, October 2005.

[5] Perkins, C., "Mobile IP", IEEE Communications Magazine, May 1997-Volume 35, Number 5.

[6] Siram, Nanditha N., Raje, R., Olson, A., Bryant, B., Burt, C., and Auguston, M., "An Architecture for the UniFrame Resource Discovery Service", Springer- Verlag Lecture Notes in Computer Science, Vol. 2596, 2003, pp. 20-35.

[7] Common Object Broker Architecture (CORBA), http://www.corba.org/, accessed on 25th September, 2009.

[8] UDDI Technical White Paper, 2000. http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf, accessed on 25th September, 2009.

[9] Sundramoorthy V., "FRODO High-level and Detailed Specifications", PhD Thesis, University of Twente, Enschede, the Netherlands, 2006.

[10] Lin,Y., and Chang, H., "Service Discovery in Location Based Services for Low-Powered Mobile Clients" Department of Computer Science and Information Engineering, National Chiayi University, Chiayi, Taiwan., 2004.

[11] Gandhamaneni, J., "UniFrame Mobile Agent Based Resource Discovery Service (MURDS)", M.S. Project TR-CIS-1122-03, Department of Computer & Information Science, IUPUI, June, 28, 2004.

[12] IBM's J9 JVM, http://brainmurmurs.com/products/timeout/docs/wm2003/j9Install.htm, accessed on 25th September, 2009.

[13] Raje, R., Katuri, P., Kumari, A., and Tilak, O., "Multi-level Matching of Distributed Software Components", Proceedings of the International Conference on Computer, Communication, and Instrumentation, Mumbai, India, 2009.