Asynchronous Service Discovery in Mobile Ad-hoc Networks

Khaled Nagi

Computer Science Department.
Faculty of Engineering.
Alexandria University
El-Shatby, Alexandria, Egypt.
nagi@ipd.uni-karlsruhe.de

Birgitta König-Ries

Institute for Program Structures and Data Organization.
Universität Karlsruhe.
D-76128 Karlsruhe, Germany.
koenig@ipd.uni-karlsruhe.de

Abstract: The wide availability of mobile devices equipped with wireless communication devices paves the way for building highly dynamic communities of ad-hoc networks. For an asynchronous service discovery in this environment, we suggest the use of mobile agents. Here, we concentrate on file sharing and exchanging services. We adopt a new asynchronous approach for discovering and locating these files, since the availability of files in this particular domain will change significantly over time when nodes constantly join and leave the network. In this paper, we present a typical application scenario, highlight the main components of the system architecture and discuss the technical challenges facing us while trying to port main infrastructural features of traditional database management systems to our mobile light-weight software agents.

1. Introduction

Nowadays, almost all new mobile devices are equipped with wireless communication devices. A consequence of this availability is that users - moving within their enterprise or institution - can easily build highly dynamic ad-hoc networks and can exchange information and services among these devices on the basis of mutual collaboration. In large institutions with several thousands of members, e.g., students in a university or employees in a big company, the need for this informal exchange of information and digital services is born. The ideal pattern for cooperation in these situations would be similar to the Peer-To-Peer (P2P) network paradigm that originally emerged on the Internet for exchanging files (e.g., mp3 files in the music exchange community) and later digital services (e.g., web services in e-Commerce) and multi-agent systems in research labs.

Common to all these P2P networks is the need for service discovery in a non-monolithic environment. This can be accomplished using centralized yellow/white pages services as in CORBA [1], FIPA [2] or completely decentralized as in Gnutella [3].

However, the environment of mobile ad-hoc networks differs in two important aspects from these: First, users disconnect themselves more often and remain offline for much longer times. Second, ad-hoc networks are characterized by the lack of an underlying infrastructure; thus, a centralized approach to service discovery is not feasible. Most of the work done in the area of service discovery in ad-hoc networks is to locate resources on the network, such as connectivity to a broader area network, access to printers, scanners, screen projectors, etc. This is usually done in the network layer by modifying the routing protocols [4] or is integrated in the application layer [5]. In some scenarios, such as file sharing and exchanging, it is also sufficient for them to receive the requested service in a later point in time. This requires the service discovery process to be *asynchronous*; a new approach for service discovery. This has also been noticed in the SOUL project [6]. There, users send off mobile agents with service requests and offers which meet in a *market place*, a geographic area with a high device density, negotiate there and then return to their owners. In contrast to our approach introduced below, SOUL does not combine synchronous and asynchronous service discovery and does not support complex service descriptions.

The rest of the paper is organized as follows. In Section 2, we present the application scenario drawn from everyday situation facing our students. Section 3 presents our asynchronous approach for service discovery using mobile agents and highlights the advantages of using mobile agents in this scenario. The system architecture and the implementation platform are described in Section 4. A comparison of our approach with approaches, which are purely based on static DBMS on mobile devices, is highlighted in Section 5. Finally, Section 6 summarizes the paper and outlines the future work.

2. Application Scenario

The following application scenario from the DIANE project [7] illustrates the need for asynchronous service discovery.

Anna is a computer science student. She has to pass her database exam in a two-week's time. That is why she is looking for solutions to all SQL exercises and unofficial summaries of lectures made by her colleagues. In this academic environment, students are willing to exchange their knowledge but there is no platform for this exchange and hence there cannot be a centralized approach for service discovery. Anna might first try a synchronous service search, i.e., she will propagate her service request in the ad-hoc network using the mechanisms provided by the DIANE platform [8]. However, it is quite likely that at any given point in time not everybody who would be able to provide some information to her will be logged on to the net. Thus, in order to obtain a full set of results, Anna should use asynchronous service discovery.

In our scenario, Anna is supposed to state her requirements "I need all notices about SQL lectures and exercises. Deadline: 15th of January 2003. Contact information: I am daily at the library from 14:00 to 16:00, or you can mail me the document at anna@student.myUni.edu". This information is saved on her PDA and she needs to propagate it to all students she comes across in order to increase her chance of getting as many documents as possible before the exam. She would transfer this information to Bob's PDA upon meeting him at lunch break, which, in turn, will autonomously forward this request to Allen's and Marc's PDA during their usual evening get-together. If Marc has a relevant document, he already has Anna's contact information; he would send her the document.

However, answering these many requests manually for people he does not know does not sound to be appealing. Marc would rather have a piece of software that handles the request, autonomously identifies the required document and then asks him for permission to send the document. If the request arrives after the 15th of January, the software will automatically remove this pending request and performs the necessary garbage collection. If it arrives before the 15th of January, it would be ready for propagation to the next PDA with the same hosting platform.

3. Asynchronous Service Discovery Using Mobile Agents

A search operation, in its most general case, is composed of data about the request and the code implementing the search. For the search to be carried out, the data *must* be propagated through the network, whereas propagating the code is very dependent on the nature of the application. Migrating both data and code is the basic definition of *mobile agents*. In our work, we make use of mobile agents to propagate search requests without loss of generality.

3.1 The degree of agency

Choosing the right degree of *agency* in the system depends on the following factors.

- *The non-monolithic nature of the system:* Recently, more and more developers are moving away from providing large monolithic systems [9] A non-monolithic system has the following implications for our application: The power of the code performing the navigation to find the required document has to depend on the node that dispatches this request not on the nodes on the course of navigation or even the node that owns the requested documents. This promises more flexibility and a better selectivity of the search request. Moreover, different nodes on the network have heterogeneous capabilities. They have different power and storage constraints. Each node can install only the capabilities that satisfy its needs. Having a non-monolithic system means that changing the system, e.g., upgrading the search algorithm, in one node does not eventually lead to upgrading the system on all

nodes that are joining or even may join the network in the future. This does not pose any further constraint on nodes joining the network to install special software to exchange files with other nodes. On the other hand, it is harder to tune the overall performance of fully decentralized non-monolithic systems.

- *The need for dynamic code adaptation**:*** Mobile agents can sense their execution environment and react autonomously to its changes by adapting their code dynamically. In our application scenario agents propagate through the network changing their search plan according to the intermediate search results they find. For example, an intelligent agent may autonomously change its plan during the course of the search switching it from an exact match search to relevance search, if it suspects the presence of very relevant documents not included in the original search request. Another example would be an agent deciding to search for a distiller after finding the requested document in postscript format, only.

- *Definition of search plans:* A search plan can be formulated as a partially ordered graph of simple actions. If these actions are somehow well defined and standardized, their code can be installed on all nodes, resulting in a rather monolithic system. In this case, only the graph of the search plan must be propagated as part of the data of the search request. However, for smaller search requests, developers – in our case computer science students- tend to prefer writing simple search scripts over defining declarative and rather formal search plans. This, in turn, favors a more agent-oriented approach. In our application scenario (and also all similar scenarios without a centralized instance to undertake the development and the coordination of the system), we assume that the users would tend to the latter approach.

3.2 Advantages of mobile agents

Furthermore, there are several typical characteristics of the mobile agent paradigm that make it most appealing for this application environment.

- Mobility: we use code migration to overcome the continuous disconnections in the ad-hoc network. In this application scenario, being connected to a neighboring PDA occurs for short periods followed by long disconnections.

- Agent cloning: together with code migration, agent cloning enables the user to discover this virtual ad-hoc network in an asynchronous manner.

- Persistence: mobile devices have a severe physical limitation, which is their sparse power supply. In order to save power, they are in hibernating mode for long period of times. The persistence and self-invocation nature of mobile agents enable them to overcome this limitation autonomously.

- Decentralization: Mobile agents are by nature decentralized and do not rely on a central device or an infrastructural component as every agent is treated in the same way.

- Autonomy: minimizes the need for system management from the part of the hosting system under the assumption that the mobile agents have no harmful intentions.

- Intelligence: the paradigm enables the users themselves to enhance the search capabilities of their mobile agent by individually developing more intelligent algorithms to deal with the inherent heterogeneity of the service description.

3.3 Difference to typical Peer-To-Peer applications

The main difference between our approach and typical Peer-To-Peer (P2P) systems used in the Internet is the *asynchronous* behavior. The asynchronous nature of mobile agent extends the search to nodes that may join the network at different times or even within several days. Furthermore, a direct implementation of these P2P file exchange systems over ad-hoc networks is *not* possible for the following reasons:

- *The highly dynamic topology of ad-hoc networks*: Nodes can move arbitrarily. This implies that nodes that are physically close, e.g., in one-hop distance at one point of time may well be at different sides of the network at another point of time. This makes it hard to maintain efficient overlay structures. Also, nodes are connecting and disconnecting from the network all the time, so that it is not possible to rely on the existence of any given node.

- *The limited resources of the nodes* in terms of power and memory capacity as most of the hosts rely on batteries for supplying power. Since sending is very energy demanding, this restricts severely the amount of messages that can be send. The lack of memory prevents the extensive replication of information across the nodes.

- *The limited bandwidth:* In a wireless environment, bandwidth is usually restricted and therefore the traffic that is needed for the maintenance of the network as well as the traffic caused by applications must be kept minimal.

4. System Architecture

4.1 Main components of a mobile agent

Figure 1 illustrates the main components of the mobile agent. The *service discoverer* resides at the heart of the agent. Here, the search intelligence is implemented. Since each node has its own semantics concerning the representation of meta-information about the documents, we use a representation of a domain specific ontology. We base this representation on work done within the DIANE project [10]. Using the description of the

required service and a cached subset of the domain-specific ontology, the agent communicates with the hosting environment for service discovery through the *interaction manager*. Having discovered an interesting service, the contact information is sent to the hosting environment to establish the Peer-To-Peer communication and the information interchange (either synchronous through file transfer using the wireless communication interface or asynchronous using mail). The *persistence manager* is responsible for serializing the agent before entering the hibernation mode or before migrating to another device. In general, it is responsible for managing the life-cycle of the agent, killing itself upon completion of its task or cloning itself before the migration. The *migration manager* is responsible for establishing the negotiation before code cloning and transfer whenever a new device is in the neighborhood.

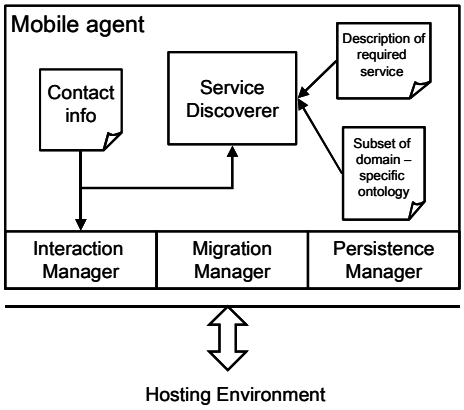

Figure 1. The main components of a mobile agent

4.2 Architecture of the hosting environment

Figure 2 illustrates the main components of the hosting platform and the life-cycle of an agent in this platform. The hosting platform encapsulates the *runtime environment* for the agents to execute. It maintains a *directory* of information services that the owner of the mobile device is willing to publish and share with other users in the ad-hoc network community. For the matching process, a *domain ontology* must be always present on the hosting platform to support the search process [10]. The *query processor* is responsible for answering the search requests coming through the interaction manager of the mobile agent. The *communication manager* interacts with the outside world. It receives migrating agents into and out of the system coming from the transport layer. It also interfaces with the email system for establishing the offline contact to the service requester.

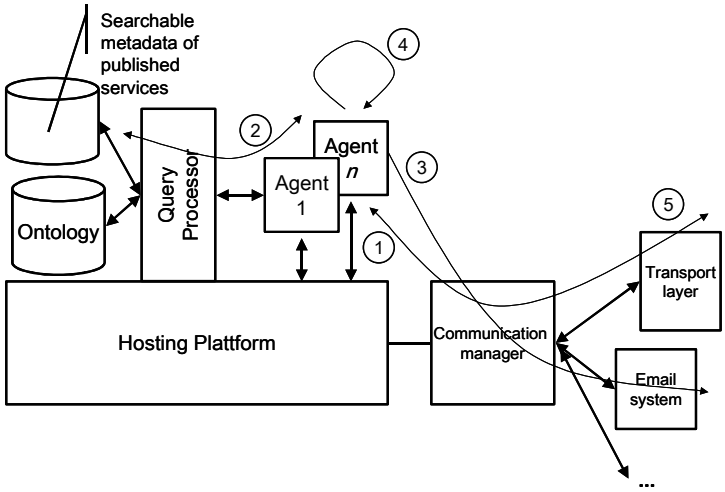

Figure 2. Hosting Platform and typical life-cycle of a mobile agent

Using these components in the hosting platform, a typical life-cycle of a mobile agent can be summarized as follows.

The migrating agent arrives from the transport layer through the communication layer (step 1). Then, it is allowed to reside in the hosting platform and perform its search for information by querying the directory of published services (step 2). If an interesting service is found, the user is asked for permission before establishing contact with the information requester, usually via email (step 3). Following deadline and resource consumption constraints, the agent may clone itself (step 4) and migrate to other devices upon their presence in the physical neighborhood (step 5).

4.3 Implementation platform

Our mobile agents will be hoping between different platforms: notebooks, PDAs, tabletPCs, smartphones, etc. in order to guarantee total interoperability, we sought a hosting environment, which is available on all their hardware platforms, and operating system. Our decision was to use Java. The Java platform is becoming the standard platform for deploying agents as compared to earlier LISP and other typical AI languages. Several Java-based platforms, such as FIPA-OS [11] and JADE [12], are becoming a defacto standard in the world of static agents. MicroFIPA-OS [13] and LEAP [14] are their counterpart for PDA and other light weight devices.

The Java 2 Micro Edition (J2ME) [15] is now available on almost all the lightweight platforms and is compatible with the Java 2 Standard Edition (J2SE) found on more powerful notebooks. J2ME provides a light version of the most popular Java libraries. It defines standard interfaces for exchanging messages over wireless networks, telephony systems, lightweight graphics, etc. To facilitate development, several emulators are available for programming and testing on the standard PC.

5. Database-like Infrastructural System Requirements

In contrast to an approach purely based on static DBMS on mobile devices, our proposed solution for asynchronous service discovery is certainly a very flexible one and promises a wider search potential. However, this approach poses lots of design challenges. In order not to end up with a virus-like system spreading across the mobile devices and consuming their scarce resources, almost every infrastructural support offered by traditional DBMS and distributed systems in general must be revisited, embedded in our mobile agents or hosting platform, and optimized for use in this new paradigm. Here, we outline some of these requirements and compare them to standard approaches used in Distributed Database Management Systems and Distributed Systems.

- *Persistence services:* we definitely need low cost persistence services for serializing the mobile agents before hibernations. This infrastructural support is present since the early days of object-oriented database management systems and seems to be one of its few components that commercially survived.

- *Agent replication:* we need replication management for cloned agents. The algorithms for managing cloning have their roots in the distributed database systems. However, since the consistency constraints are far more relaxed, these algorithms must be, in turn, simplified.

- *Agent coordination:* another usage of agent cloning is to assign different agents subtasks in a distributed information retrieval process. The coordination of these spawned agents can be easily modeled using open nested transactions. In our work done on static agents [16], we deomsntrated the feasibility and the beinfits of using a variation of the open-nested transaction model to model agent coordination and govern both their cooperative and competitive behavior.

- *Ontology modeling:* in this highly decentralized system with no standardizing instance, ontology modeling becomes a difficult task as opposed to well-defined database schemes. Here, we base our solution on work previously done within the DIANE project [10].

- *Security:* Security issues in the field of mobile agents are much more complex than in the the field of distributed database management systems. Here, it is a two-fold problem. Taking the side of the hosting nodes, executing the foreign code of a mobile agent certainly represents a security threat. Although user authentication and authorization as in DBMS seem to be an appealing solution, it is practically very difficult to achieve due to the large flexibility and execution possibilities of a mobile agent as compared to simple SQL statements. In the past few years, many research efforts concentrated on protecting the hosting platform from malicious agents. This protection varies from the use of authentication and authorization to a complete monitoring of the execution of the agent. Considering the side of the mobile agent, revealing its plans by expressing it in a declarative way to the hosting platform for execution is the security threat. Malicious nodes can manipulate the honest agent turning it to a virus-like piece of software spreading through the network. In our scenario, we assume the honesty of both agents and hosting nodes; a realistic assumption for a community of students or employees of the same organization having a common interest of sharing information among them.

- *Scalability:* is the most crucial design challenge in this scenario. "How many agents can reside in the mobile device with its limited resources" is the decisive question, whose answer will tune the operating parameters of the system. Simulation models extracted from the world of distributed information systems form the basis for our simulation study.

- *Predictability:* is a very difficult problem on the level of the macro agent society. Similar to scalability, simulation is our only means to improve the predictability of the system. Our experience [16] in this area, shows that here too, many simulation parameters have their roots in the simulation models of distributed information systems.

6. Summary and future work

Mobile ad-hoc networks offer new possibilities for users to share information. In such environments, service discovery cannot always be done synchronously. In this paper, we present an approach for asynchronous service discovery based on mobile agents. We present the reasons for choosing mobile agents for our approach. Then, we highlight the basic architectural components of the system and present the required infrastructure that must be adopted from the well-established database technology.

Beside the developement of a functional protytype based on the J2ME, we are currently working on an extensive simulation model to analyze the behavior of the system. In our analysis, we concentrate on performance metrics, such as the degree of propagation of agents within the network as compared to the purely synchronous approach. We also investigate the impact of each of the control parameters on the performance of the mobile agents and the resource consumption in the network in terms of superfluous migrations. Here, we target some good settings of the control parameters before the actual deployment.

References

[1] CORBA Trading Object Service http://www.omg.org/technology/documents/formal/trading_object_service.htm

[2] The Foundation for Physical Agents. http://www.fipa.org

[3] Gnutella File Sharing. http://gnutella.wego.com

[4] R. Koodli and C. Perkins. Service Discovery in On-Demand Ad-Hoc Networks, MANET Working Group Internet Draft, October, 2002.

[5] J. Wu and M. Zitterbart. Service Awareness and its Challenges in Mobile Ad-Hoc Networks. In: *Proc of the GI Jahrestagung 2001, Volume 1*, Vienna, Austria, 2001.

[6] SOUL Project. Self-Organized Ubiquitous Learning. http://bowmore.uni-trier.de/soul/publikationen.htm

[7] DIANE: Dienste in Ad-hoc-Netzen. http://www.ipd.uni-karlsruhe.de/DIANE

[8] M. Klein, B. König-Ries. Multi-Layer Clusters in Ad-hoc Networks - An Approach to Service Discovery. In: *Proc. of the International Workshop on Peer-To-Peer Computing*. Pisa, Italy, May 2002.

[9] D. Kotz. Future Directions for Mobile Agent Research, available at: http://dsonline.computer.org/0208/f/kot.htm

[10] B. König-Ries, M. Klein. Information Services to Support E-Learning in Ad-hoc Networks In: *Proc. of. First International Workshop on Wireless Information Systems (WIS2002)*, Ciudad Real (Spain), April 2002.

[11] The FIPA-OS platform, http://fipa-os.sourceforge.net/

[12] The JADE (Java Agent DEvelopment Framework), http://sharon.cselt.it/projects/jade/

[13] Supporting software agents on small devices. In Proc. of the first international joint conference on Autonomous agents and multiagent systems, Bologna, Italy, 2002.

[14] The LEAP (Lightweight Extensible Agent Platform), http://leap.crm-paris.com/

[15] Java 2 Micro Edition, http://java.sun.com/j2me/

[16] K. Nagi. Transactional Agents: Towards a Robust Multi-Agent System. *Lecture Notes on Computer Science (LNCS 2249), Springer-Verlag.* 2001.