

An Intelligent Agent based Approach for Service Discovery in Wireless Ad-hoc Networks

MINAS PERTSELAKIS AND NICOLAS TSAPATSOU LIS

Department of Electrical and Computer Engineering

National Technical University of Athens

9, Iroon Polytechniou, 15773 Zografou,

GREECE

E-mail: mper@cslab.ntua.gr, ntsap@image.ntua.gr

Abstract: - In this paper we propose an architecture for modeling the service discovery functionality of computing devices operating in pervasive computing environments. Forming wireless ad hoc networks requires service discovery protocols that support both proximity-based detection and abstract level service matching. In the proposed architecture, services, provided by sensors and actuators, are indirectly located through polling while composite service matching is based on the intelligent agent philosophy. In this way, our service discovery approach exploits the spread of proximity based device detection and the dynamics of the multi-agent environments.

Key-Words: - Intelligent Agents, Wireless Ad hoc Networks, Service Discovery Architectures

1 Introduction

Pervasive computing aims at the *availability* of software applications and multimedia information anywhere and anytime, and *invisibility* of computing; computing modules are hidden in multimedia information appliances which we use in our everyday life [1] [2]. Applications conforming to the pervasive computing paradigm are characterized by *interaction transparency* and *context-awareness* [3].

Interaction transparency means that the human users are not aware that there is a computing module embedded in the tool or device that they are using; it is definitely a non-intrusiveness process.

Context awareness refers to adaptation of the behavior of an application as a function of its current environment. This environment can be characterized as a physical location, an orientation or a user profile. In a mobile and wireless computing environment, changes of location and orientation are frequent. Context-aware applications can sense the environment and interpret the events that occur within it. Sensing their owner's identity and location is very important both for security reasons, since pervasive computing applications run in physical

devices that human users carry with them, and for adapting the provided services based on the user profile and location in an intelligent manner.

Context-aware applications face severe problems when both the service users and the service devices are mobile. These problems require dynamic forming of wireless ad hoc networks and on-the-fly system configuration. The dynamics of such system are complex because it requires not only system reconfiguration and low level configuration, e.g., multiple communication and security protocols, but also service detection and monitoring in order to provide the best available services [4].

Our proposal is an intelligent agent based approach for *service discovery* in wireless ad hoc networks. We consider that pervasive computing environments consist of three types of elements: computing devices (called *intelligent artefacts* in our terminology), sensors and actuators. Intelligent artefacts are devices with general-purpose computational logic, which allows them to be programmable by their users. In contrast, sensors and actuators cannot be programmed to implement general-purpose communication and service discovery protocols. Instead they support specific

protocols through which send / receive their data to and from conforming devices.

However, in the near future more and more sensors and actuators will support wireless communication protocols and will be Bluetooth enabled; this means that they will be detected by other Bluetooth enabled devices based on radio proximity. Our service discovery approach utilizes intelligent agents to implement high-level service discovery and Bluetooth SDP for device detection [5]. Thus, it manages to combine the characteristics of proximity based device detection and the advantages of the intelligent agent technology.

The paper is organized as follows: In Section 2 we discuss the deficiencies of existing Service Discovery Protocols, while in Section 3 we present the proposed architecture of the intelligent artefacts (our computing devices). In Section 4 we describe how wireless ad hoc networks, consisting of intelligent artefacts, sensors and actuators, are set up using this architecture. Finally implementation details are given in Section 5.

2 Deficiencies in the existing service discovery architectures

The most popular Service Discovery Protocols are Salutation, Service Location Protocol (SLP), Jini technology, Universal Plug and Play (UPnP) and Bluetooth Service Discovery Protocol (SDP) [5],[6],[7],[8],[9]. All the mentioned service discovery procedures and architectures have been developed to explore the service discovery issues in the context of distributed systems. While many of the architectures provide good base foundations for developing systems with distributed components in the networks, they do not adequately solve all the problems that may arise in a dynamic domain. Moreover, with the exception of Bluetooth, none of the others have been specifically designed for mobile environments.

Lack of rich representation. The existing service discovery infrastructures lack expressive languages, representations and tools that are good at representing a broad range of service descriptions and at reasoning about the functionalities and the capabilities of the services.

Lack of inexact matching. Most existing work supports an attribute-based discovery as well as a simple name lookup to locate a service. Usually there

are only a set of primitive attribute types, such as string and integer, to characterize a service. Thus, the service discovery process is primarily done by type matching, string comparison, or integer comparison. For representing real world objects such as network services, it is necessary to have more complex data structures to capture richer semantics.

3 The Proposed Architecture

In our consideration pervasive computing elements fall into three main categories: Intelligent Artefacts, Sensors and Actuators. These elements must be able to cooperate efficiently by exchanging data and services in order to accomplish their individual or collective tasks and define a place for pervasive computing applications.

Intelligent Artefact: An Intelligent Artefact is a mobile computing device able to offer high complexity services or to make decisions derived from the computational process and analysis of less composite services. These intelligent entities, acting as Clients, must be able to poll Sensors and Actuators, in order to use their services. The internal architecture of such an entity is based on the technology of intelligent agents and is described in detail later in this paper.

Sensors and Actuators. Sensors and actuators are considered to be small mobile wireless devices with a very specific area of operation, acting both as Primitive Service providers for their Clients (Artefacts). While Sensors' responsibility is to supply a computing device with a steady, unceasing flow of raw data (simple signals or features), Actuators are responsible for the conversion into action of any decision taken by the computing device. In other words, Sensors provide data services, while Actuators provide action services. If they are available, both types must be able to be polled by an Intelligent Artefact, since they cannot advertise themselves.

3.1 Artefact's Architecture

The proposed artefact agent-based architecture, as shown in Figure 1, is consisted of two basic components: the Agent Platform and the Communication Manager. The Agent Platform includes the Service Manager(s), which are the controllers of the system that coordinate the message passing protocol between Clients and Services, and

the Intelligent Unit(s), which perform intelligent tasks using the Primitive Services provided, acting also as providers of Composite Services to other Clients (e.g. other Intelligent Units). The Communication Manager(s) handle all the lower level communication with the Sensors and the Actuators, supporting a number of different protocols, since Artefacts must be capable of cooperating with heterogeneous network elements. A detailed description of each component is presented thereafter.

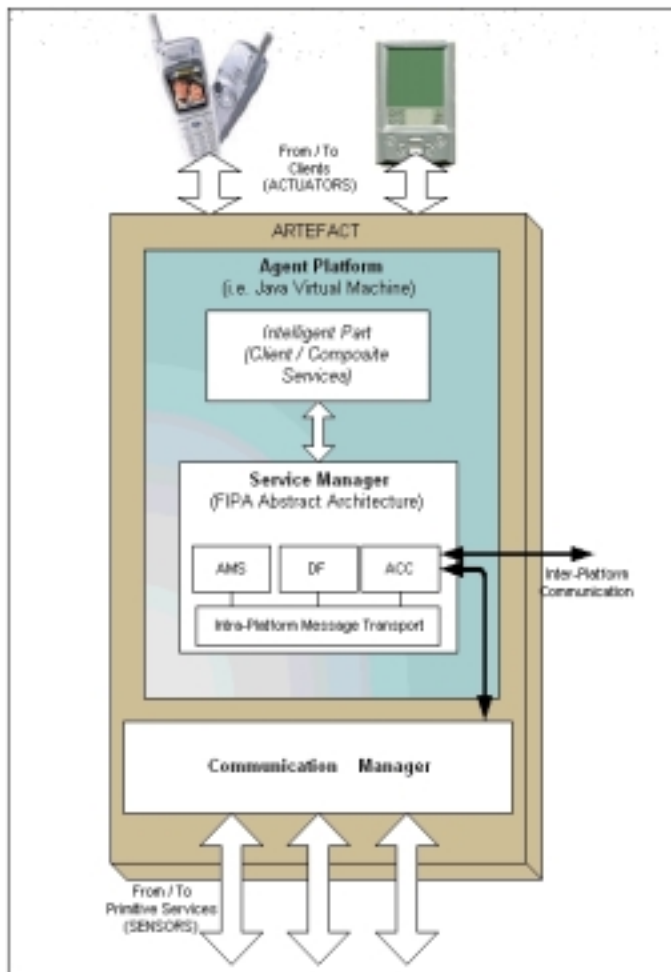


Fig.1: Artefact's Architecture

3.1.1 The Agent Platform

The Agent Platform (AP) provides an infrastructure in which agents can be deployed. An agent must be registered on a platform in order to interact with other agents on that platform or indeed other platforms. AP is divided in two basic parts of

operation: the Intelligent Unit and the Service Manager.

Intelligent Unit. The Intelligent Unit includes all the operations and the software needed to implement the intelligence of the artefact.

Its role is to act mainly as a Client requesting Services (Primitive or Composite) from other network elements through the Service Manager. When these Services become available, the IU processes them and produces other Composite Services, which can be just conclusions or even actions. Each Intelligent Unit may be implemented using intelligent agents and/or algorithms derived from computational intelligence (e.g. neural networks, fuzzy logic, etc. or a combination of them), but this issue is not addressed in this paper.

Service Manager. The Service Manager acts as a mediator between the Primitive Services or the Composite Services and the Clients. When a service starts up, it has to register itself to the SM, sending its Registration File (RF). This file contains its name, id and the interfaces it implements. When a new Client comes along, SM sends to it a Service List Object (SLO). This SLO changes dynamically, according to the services registered or deregistered within the Service Manager. So the Client always has the updated list of services. The Client can select a service, which causes the SM to send the appropriate RF for that service. The SM then updates its database to reflect that the specific Client is interested in the requested service. Whenever the Service Manager gets a status update of the service, it will send it to all interested Clients. The Client will continue to receive status reports from SM, until it deregisters itself. The Client sends the new RF to the Service Manager, after invoking the interfaces of the service. On receiving this RF, the SM validates the Client and the Registration File. If the Service is still available, the SM sends the Registration File to it; otherwise it is queued for sometime. Once this timeout expires, an error is returned to the Client.

In addition, SM is responsible for service discovery (polling) and leasing. It allows services to register themselves (in its platform or another) for a certain amount of time. If it does not receive any status update within that time, the registration is deleted. The SM implements an intelligent lookup for services, enabling the Clients to search for services that provide a certain kind of, or related function.

3.1.2 Service Manager Architecture

According to FIPA specifications [12], an Agent Platform should consist of three capability sets: an Agent Management System, an Agent Communication Channel and a default Directory Facilitator. In this architecture these types of agents are implemented into the Service Manager and their operations are summarized below:

Agent Management System (AMS). This agent is unique and mandatory for each Agent Platform. It controls and manages the AP and agent activities. More specifically, the AMS performs tasks such as script interpretation (Note: the AMS is expected to support multiple agent script languages such as Java or Tcl/Tk, as agents must be capable of operating in a heterogeneous network environment), agent management (e.g. assembly and disassembly of agents in accordance with the generic agent structure, registration and deregistration of agents in the Directory Facilitator), white-pages service (a list of agents identified by their GUID - the Global Unique Identifier that locates univocally the agent inside the community), resource management (i.e. controlling the system resources that are made available to individual agents) and handling of external messages (e.g. delivering incoming messages to the appropriate user application).

Functions associated with agent migration (e.g. suspension of script execution at an arbitrary point, capturing of state variables, etc.) and agent communications (e.g. establishing a connection to a remote platform) are implicitly handled as part of the script interpreter.

Agent Communication Channel (ACC). This agent is unique and mandatory for each Agent Platform. It plays the role of message router of its AP using the information provided by the Agent Management System. ACC can also be used either for the communication of agents inside a platform (intra-platform) or between agents of different platforms (inter-platform). Each agent must have access to at least one ACC.

Directory Facilitator (DF). A DF agent offers a yellow-pages service to other agents. Its key responsibility is to provide directory services (e.g. a listing of services and resources available at the artefact). All the agents registered at a given DF form an agent domain. Therefore each DF manages a different agent domain.

3.1.3 The Communication Manager

The Communication Manager (CM) is responsible for the communication between the Sensors or Actuators and the Service Manager. It could be implementing a number of different protocols by having different communication modules, for example, one that handles IR, another that works with Bluetooth, one that works with wireless LAN etc. The Communication Manager talks via a certain socket (ACC) to the Service Manager. This allows CMs and SMs to be on different systems.

When the Communication Manager receives information from a Primitive Service, it sends this information directly to the Service Manager through ACC. When it receives data from a Service Manager, the Communication Manager configures its status, validates the data and sends them thereafter to the appropriate network element.

4 Forming Wireless Ad hoc Networks

Using the inter-platform communication protocol and the agents' properties of mobility and interoperability, an 'ad hoc' network of Artefacts can be created in order to achieve more difficult and complex goals. For this reason an application level network communications protocol was designed to operate on top of the TCP/IP protocol stack. Given the importance of efficiently utilizing network resources, the protocol supports remote messaging between agents and platforms (e.g. instead of blindly transferring agents from one site to another, a "look before you leap" protocol mechanism is employed which allows an agent at a platform to query the status of another platform before migrating to the new artefact). In this way, mobile agents will only be transferred across the network if it is known beforehand that the remote artefact is capable of providing the desired services. Many of the network communications protocol messages can be directly invoked via the agent command language specified as part of the mobile agent based architecture.

4.1 Usual Operation

The usual operation of an Artefact is to create Composite Services and to perform specific tasks by requesting and using the Primitive Services provided by the Sensors or the Actuators. This operation includes the following steps Initially, the Intelligent Unit is assigned with a specific task. According to

this task, IU sends to the Service Manager a message to ask for the necessary services. The SM then, with the help of the Communication Manager, returns a Service List Object with all available Services, ids and states, thus allowing the IU to choose among the offered Services and start the Client-Service process. If the IU needs more services than these available, then the SM commences a service discovery process. When a requested not-available-before Service becomes available the SM updates its Service List and sends it again to the IU, which reacts accordingly. If a registered Service for some reason cease to exist or to function properly, the Communication Manager becomes aware of it immediately and informs the SM to update the registry. Whenever a Service List is updated, it is sent to all interested Clients, so the Clients always have the updated list of services.

4.2 Service Discovery Process

In case an Intelligent Artefact needs a Primitive Service that it is not in its registry, it is able to search for that service in another Artefact's service deposit. This is realized through the Inter-Platform Communication Protocol. In such case the artefact that requests services acts as a Client Artefact (representing the request of its Intelligent Unit), while the artefact that may provide the requesting services acts as a Server Artefact.

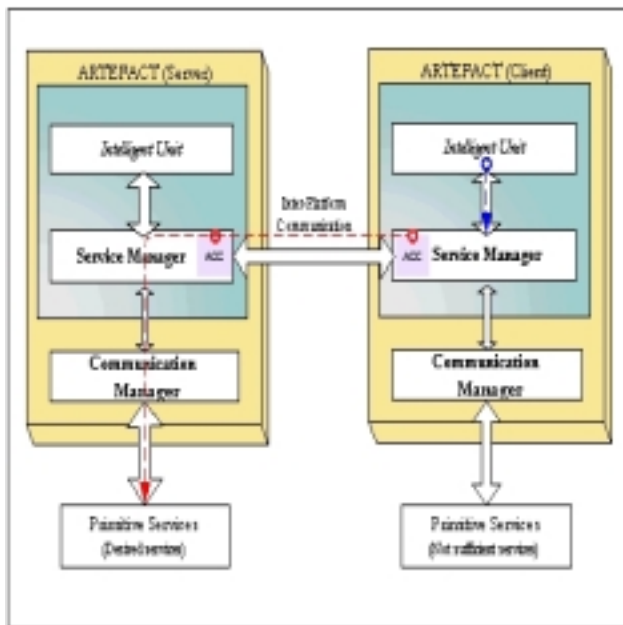


Fig.2: Using Primitive Services of other neighbor Artefacts

The Client Artefact sends a message to the Service Manager of the Server Artefact in order to query the status of the platform and the Services available, before sending a mobile agent. The Server's Service Manager responds by sending its Service List Object and a verification message that the Artefact is able to accept a new Client. Then, the Client is able to launch a mobile agent into the Server Artefact through the Agent Communication Channels, following the Inter-Platform Communication Protocol. After the execution of the mobile agent's script a connection is established in order for the Client Artefact to use the desired Primitive Service via the Server's Communication Manager, as it is shown in Figure 2.

In another case, an Artefact may be in need to use some of the Composite Services produced by another Artefact. In the same way, as described above, the Artefact no.2 can establish a connection through the Inter-Platform Communication Protocol and thus, become able to use the Composite Services of Artefact no.1, as shown in Figure 3.

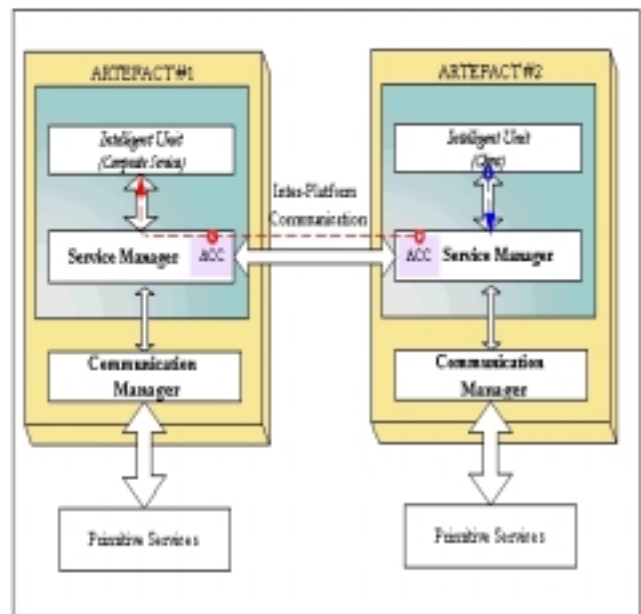


Fig.3: Using Composite Services of other neighbor Artefacts

5 Implementation Issues

For the implementation of the proposed architecture an ad-hoc network was created using the following components, as it shown in Figure 4.

A laptop computer was properly configured to play the role of the Intelligent Artefact, while one PDA and one cellular phone (both Bluetooth enabled) participated as an actuator and a sensor respectively. Using the Zeus Agent Building Toolkit [13], a Java-based multi-agent environment was created as an agent platform to host the Intelligence Unit and the Service Manager.



Fig.4: Prototype realization

In order to integrate Bluetooth capabilities for the Communication Manager, the XJB 100 Bluetooth Host Stack Protocol was implemented, which supports the serial port, generic access and service discovery application profiles for the host device [14]. To extend this network, we could place another laptop as a second Artefact connected to the first one via a LAN.

References:

- [1]. J. Birnbaum, "Pervasive information systems," *Communications of the ACM*, 40(2): 40-41, February 1997.
- [2]. M. Weiser, "Some computer science issues in ubiquitous computing," *Communications of the ACM*, 36(7): 75 - 84, July 1993.
- [3]. G. Abowd, "Software engineering and programming language considerations for ubiquitous computing," *ACM Computer Survey*, 28(4), December 1996.
- [4]. J. Waldo, "Alive and Well: Jini Technology Today," *IEEE Computer*, June 2000
- [5]. Bluetooth SIG, *Bluetooth Specification Part E: Service Discovery Protocol (SDP)*, November 1999, <http://www.bluetooth.com>
- [6]. Salutation Consortium, *Salutation Architecture: Overview*, 1998, <http://www.salutation.org>
- [7]. E. Hughes, D. McCormack, M. Barbeau and F. Bordeleau, "Service Recommendation using SLP," Carleton University, Canada, May 2001
- [8]. Sun Microsystems, *Technical White Paper: Jini Architectural Overview*, 1999, <http://www.sun.com/jini>
- [9]. Universal Plug and Play Forum, *Universal Plug and Play Device Architecture*, version 0.91, March 2000
- [10]. R. Grimm, T. Anderson, B. Bershad and D. Wetherall, *System Architecture for Pervasive Computing*, University of Washington, Seattle
- [11]. M. Panti, L. Penserini, L. Spalazzi, *A Critical Discussion about an Agent Platform based on FIPA Specification*, University of Ancona, Italy, 2000
- [12]. Foundation of Intelligent Physical Agents, *FIPA Abstract Architecture Specifications*, 2000 <http://www.fipa.org>
- [13]. British Telecom. plc, *The Zeus Agent Building Toolkit*, 2000, <http://193.113.209.147/projects/agents/zeus>
- [14]. Zucotto wireless, Inc. , *XJB 100 Bluetooth Host Stack Protocol*, 2000 <http://www.zucotto.com/products>