



UNIVERSITY OF
CALGARY

A Survey on Application of Agent-based Technology in Pervasive Computing

Provided by: M. Moussavi

This tutorial has been provided as part of the coursework for:

SENG 609.22
Agent-based Software Engineering

Abstract:

In the recent years the attention of wireless technology has been increasingly focused on pervasive systems such as cell phones and PDAs with limited communication and processing power. One of the major challenges in this area is to find an appropriate technique to enable these devices on an ad hoc wireless network to find the available services offered by other nodes. In this tutorial, which is part of the coursework for SENG 609.22, first, the mechanisms of the existing protocols for service discovery is briefly studied, and finally an agent-based application presented by Zhou Wang, Jochen Seitz in their paper entitled “An Agent-based Distributed Service Model for Nomadic Users”, published by Institute of Telematics, University of Karlsruhe D-76128 Karlsruhe, Germany, is briefly explained.

Overview:

Collaboration and communication between small devices has been one of the most challenging areas of wireless technology in recent years. These type systems with limited processing and communication power are called pervasive systems. Examples of pervasive systems are PDAs (Personal Digital Assistants), cell phones, that each one offers a specific service to the user.

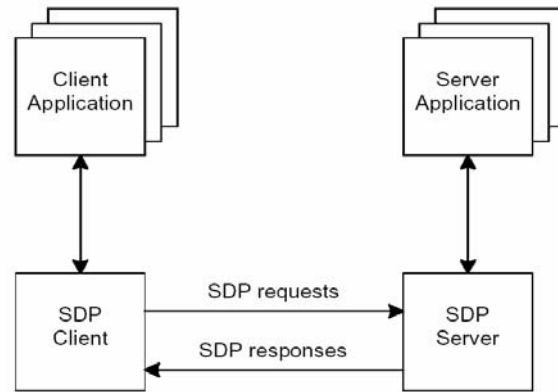
One of the areas of interest in pervasive technology is to make these devices collaborative with each other to make more services available to each unit [19]. For example, the collaboration between your cell phone and the air conditioning system in your car that allows you to adjust the car’s air temperature before leaving your home or your office, is a simple example of this type of collaboration. In other words your car’s air conditioning system provides a service (adjusting air temperature), and makes it accessible to other devices on an ad-hoc network system. In order to achieve this type collaboration a mechanism should be in place that allows the devices dynamically to discover each other and share their available services with other [19].

Today, many service discovery protocols (SDPs) for ad-hoc networks have been introduced. The existing protocols have been classified in two categories, proactive like IEEE802.11 LAN and HiperLAN, and reactive like Bluetooth. A proactive SDP that is also called a push-based SDP uses a broadcast mechanism, where each node advertises the services it can offer to the rest of the nodes in the network. In the case of reactive SDPs that is also known as pull-based, each node queries other nodes for the available services.

Background and the Study of Existing Protocols:

In a network-centric model of computing systems, the first problem for getting access to various kinds of information is discovering these available services on the network [6]. In general a service discovery mechanism should provide the means for a client to

discover the existence of services provided by a server, and other detailed information such as the attributes of those services. For example, the following figure shows the Bluetooth's service discovery configuration [6], which involves communication between an SDP server agent and an SDP client agent. Where a service can be considered as entity (object) that can provide information, perform an action, or control a resource on behalf of another entity [6]. A service may be implemented as software, hardware, or a combination of hardware and software [6]. Bluetooth specification part E expresses this process based on the following on the following figure:



As the computer networking community realized the need for service discovery several companies started researches in this field. The most well known among them includes:

- ❑ Service Location Protocol (SLP), developed by the IETF;
- ❑ Jini, which is Sun's Java-based approach to service discovery;
- ❑ Salutation;
- ❑ Microsoft's Universal Plug and Play (UPnP);
- ❑ Bluetooth

Service Location Protocol (SLP)

SLP is a decentralized, lightweight, scale and extensible protocol for service discovery within a site [12]. SLP defines *Service URL* that defines service type and address for the service. There are three agents in SLP: *User Agent (UA)*, *Service Agent (SA)*, and *Directory Agent (DA)*. UA is a piece of software that sends a service request to the other nodes. SA is another piece of software that advertises the services, and finally DA builds a directory of the advertisements. SA advertises itself by registering with a DA, and its registration message contains the URL for the advertised service, lifetime for the service, and other descriptive information. SA periodically refreshes the registration with DA. For If network is small there will be no directory agent (DA), and in this case UA's service request message is directly sent to SAs [2].

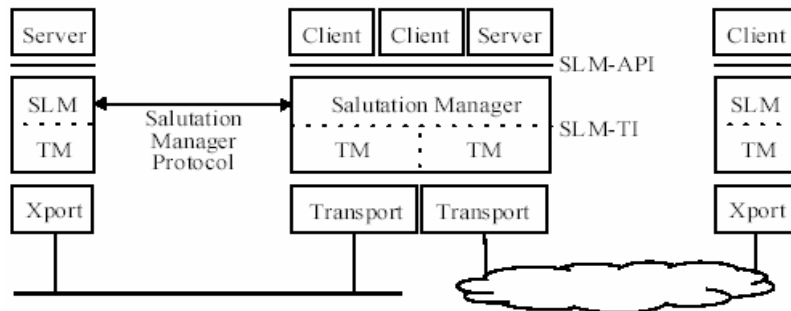
Jini

The purpose of the Jini architecture is to federate groups of devices and software components into a single, dynamic distributed system [2]. Jini technology is an extension

of the programming language Java and has been developed by Sun Microsystems. Each Jini device is assumed to have a Java Virtual Machine (JVM) running on it. The Jini architecture principle is similar to that of SLP [8]. Devices and applications register with a Jini network using a process called *Discovery and Join*. To join a Jini network, a device or application places itself into the *Lookup Table* on a lookup server, which is a database for all services on the network (similar to the DA in SLP). Services may upload device drivers, an interface, and other programs that help the user to access the service. When a client wants to utilize the service, the object code is downloaded from the Lookup Table to the JVM of the client [2]. Jini system operates on three protocols called *discovery*, *join*, and *lookup*. A pair of protocols discovery/join occurs when a device is plugged in, discovery occurs when a service is looking for a lookup service with which to register, and Join occurs when a service has located a lookup service and wishes to join it [2]. Access to services in the Jini system is granted on lease basis: A service is requested for a time period and, then, granted for negotiated period between the service user and provider. This lease must be renewed before its expiration. Otherwise, the resources associated with the services are released [2].

Salutation

The Salutation architecture has been developed by an open industry consortium, called the Salutation Consortium[1]. As shown in the following figure, the Salutation architecture is composed of two major components [9]: *Salutation Manager* and *Transport Manager*. Services register with an SLM, and clients query the SLM when they need a service. If the required service is found, clients are able to request the utilization of the service. The Salutation’s architecture provides a standard method for applications, services and devices to describe and to advertise their capabilities to other applications, services and devices. It also enables application, services and devices to search other applications, services or devices for a particular capability [2][13].



Universal Plug and Play (UPnP)

UPnP was developed by an industry consortium. It is mainly used for small office or home computer network [1]. UPnP's does not have mechanism such as central service register like DA in SLP. It uses a discovery protocol called Simple Service Discovery Protocol (SSDP) [1]. SSDP uses HTTP over UDP and it's designed for usage in IP networks [14]. Universal Plug and Play (UPnP) is an architecture for pervasive peer-to-

peer network connectivity of intelligent appliances, wireless devices, and PCs of all form [1]. In UPnP, a device can dynamically join a network, obtain an IP address, convey its capabilities upon request, and learn about the presence and capabilities of other devices or similarly leave the network [1]. SSDP uses HTTP over multicast and unicast UDP that are referred to as HTTPMU and HTTPU, and XML is used to describe device features and capabilities [2].

Each advertisement message contains a URL, pointing to an XML file in the network. This file describes the UPnP device's capability [2]. Other devices can retrieve this XML file, and inspect the features of this device and decide whether it fits their purposes [2].

Bluetooth Service Discovery Protocol

Bluetooth is a new short-range wireless transmission technology. The Bluetooth protocol stack contains the Service Discovery Protocol (SDP), which is used to locate services provided by or available via a Bluetooth device. SDP is described in the Bluetooth specification part E [6]. It is based on the Piano platform by Motorola and has been modified to suit the dynamic nature of ad hoc communications, and supports the following inquiries: search for services by service type; search for services by service attributes; and service browsing without *a priori* knowledge of the service characteristics. SDP does not include functionality for accessing services. Once services are discovered with SDP, they can be selected, accessed, and used by other mechanisms such as SLP, etc.

Since Bluetooth SDP is designed specially for Bluetooth environments, it supports limited functionality, compared to other service discovery protocols. Service discovery application profile [15] defines protocols and procedures used by a service discovery application to locate services in other devices. Bluetooth SDP runs on a predefined connection-oriented channel of L2CAP, and there's no event notification when services become unavailable [2]. Therefore, other service discovery protocol might be used to complement these lacks [2].

Evaluation of Different Protocols

Although all protocols are using similar architectures there are some differences between them. In this section these differences have been summarized:

SLP is a standardized and well-documented protocol that works on TCP/IP networks, and allows the user agents to select the most appropriate service from among services on the network. Since SLP is able to operate with or without a Directory Agent (DA), it is suitable for networks of different sizes, ranging from very small ad hoc connectivity to large enterprise networks [1] [2].

UPnP also is designed for TCP/IP networks, and one of its important features its XML use for service/device description. UPnP has been supported by many companies, and it seems to be one of the successful protocol, using centralized approach [2].

The Salutation is also a well-defined protocol, but it suffers from the lack of remote event notification, which is a useful feature in distributed environment [2]. Since it uses Service on a higher layer, and the transport layer is not specified, it can operate in any network [2]. Salutation like SLP is also independent on the programming language [1].

Jini is mainly based on Java. Since Java is a platform independent, in a way this is an advantage to Jini, but also can be considered as a constraint [2]. Another advantage of Jini is its possibility to move the code (device driver to the client application) [2]. Also its service proxy concept is one of strongest features not found in other protocols [2].

The service discovery protocols mentioned above use a centralized information server that listens for broadcast or multicast packets on a well-known address [2]. Ad hoc groups cannot be expected to provide such infrastructure. Bluetooth SDP is designed to suit the dynamic nature of ad hoc communications. On the other hand, since Bluetooth SDP is designed specially for Bluetooth environments, it supports limited functionality, compared to other service discovery protocols [2].

An Agent-based Service Model for Nomadic Users

This section presents a brief description studies by: Wang and Seitz, in the Institute of Telematics, University of Karlsruhe in Germany [20]. This study focuses on providing a generalized infrastructure to enable efficient nomadic service access, and suggest a model that supports asynchronous service lookup, dynamic service invocation and flexible adaptation to environments that improves the service availability and service quality, and reduces network bandwidth consumption.

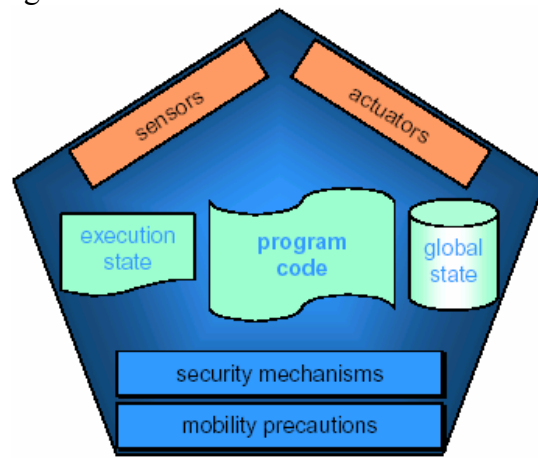
This model consider

- Support for disconnected users: Because of the characteristics of the wireless link, users may be unexpectedly disconnected from time to time. This should not lead to fatal communication failures.
- Application of mobile code: From our point of view, the device heterogeneity and user mobility can be facilitated by mobile code.
- Extension of standardized frameworks: To achieve far-reaching acceptance, a solution for a distributed service model for nomadic users should consider standardized frameworks and should try to extend these.
- Consideration of scarce resource on the end systems: As already pointed out, mobile end systems suffer from resource shortage that has to be handled in the middleware for distributed applications.
- Generality: Finally, the architecture should be generally applicable supposing that the mobile users may want to use the same resources and applications on the mobile end system as in a fixed network.

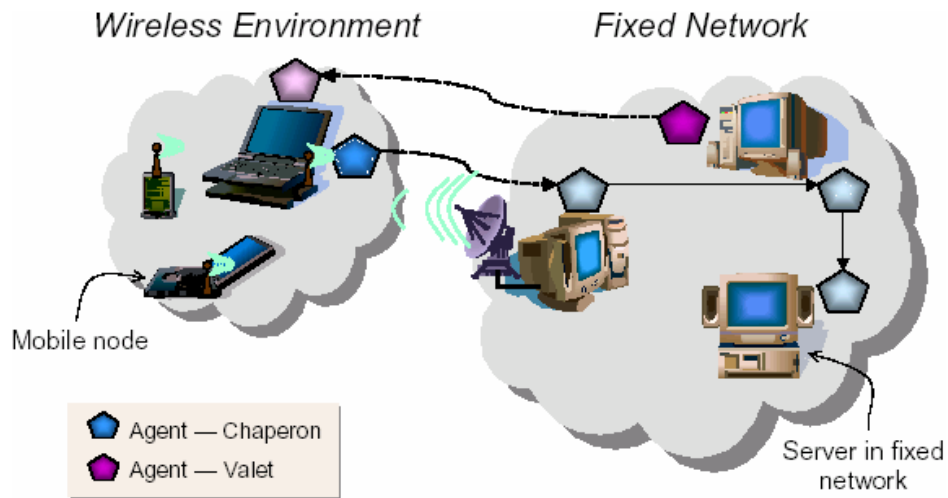
The main components of this model are two types of mobile agents which are called “chaperon” and “valet”. These agents act as active and moveable proxies to support service access in mobile and wireless environments.

Wang and Seitz [20], has adopted mobile agents approach to build their model.

Mobile agents are not reliant upon the agent platform and are able to carry data with them so they can preserve state information and precede a computation while migrating through a number of nodes. Wang and Seitz [20], in their paper explain a mobile agent based on the following figure:



Wang and Seitz, in their paper add that, the main goal of their model is to provide an efficient access the remote services by simplifying the lookup and service selection procedures in a wireless environments. The following figure presented by Wang and Seitz in their paper, shows schematic presentation of their model:



This figure shows, how the mobile element connects to the fixed network via a particular access point. The selection of an appropriate access point and hand-offs between different access points is handled in the data link layer and should not be considered in the middleware to support mobile users. Once the client requests services in fixed networks, it sends a mobile agent called chaperon to the fixed network via the access point. The task of the chaperon is to act on behalf of the mobile client to retrieve the requested services. Therefore, the chaperon contains a description of the required service types, service-specific attributes as well as user-specific preferences, and is able to personalize the services for the mobile user[20].

Another agent called valet prepared by the service provider according to the user's needs is then sent back to the mobile client [20]. Running on the mobile nodes, a valet can be regarded as the proxy for accessing the remote service. According to Wang and Seitz, the valet either implements the whole functionality of the requested service (which supports disconnected users), or contains the service interfaces that the client can invoke (similar to the JINI architecture) [20]. That is in this case the service can only be used during connection. The valet then takes over the further communication with the service provider in fixed network. Then, service data is transferred from the service provider to the client applications via the valet[20]. Whenever there is a resource bottleneck or a problem in the communication, the valet can adapt the communication flow to the given conditions [20]. Wang and Seitz also add:

- in order to allow nomadic users to roam, the network infrastructure has to provide horizontal or even better vertical network hand-offs. This hand-off will take place automatically as it is handled in the data link layer.
- in order to serve the mobile user further on, the chaperon is notified about the hand-off and migrates into the area of the new access node, accompanying the mobile node's movement.

References:

- [1] *Bettstetter C. and C. Renner*, A Comparison of Service Discovery Protocols and Implementation of the Service Location Protocol, Technische Universität München (TUM), Institute of Communication Networks, D-80290 Munich, Germany
- [2] *Lee C., and S. Helal*, Protocols for Service Discovery in Dynamic Mobile Networks, *International Journal of Computer Research* ISSN 1535-6698, Volume 11, Number 1, pp. 1-12, 2002 Nova Science Publishers, Inc.
- [3] Sun Microsystems, .Jini Connection Technology, <http://www.sun.com/jini>.
- [4] Sun Microsystems, .Jini Community Resources: Jini Technology Architectural Overview,. January 1999. <http://www.sun.com/jini/whitepapers/architecture.html>.
- [5] Sun Microsystems, .Jini Community Resources: Jini Specification v1.0.1. www.sun.com/jini/specs.
- [6] Bluetooth Specification Part E. Service Discovery Protocol (SDP). <http://www.bluetooth.com>, November 1999.
- [7] Brent Miller and Robert Pascoe. Mapping Salutation Architecture APIs to Bluetooth Service Discovery Layer. <http://www.bluetooth.com>, July 1999.
- [8] Sun. Technical White Paper: Jini Architectural Overview. <http://www.sun.com/jini/>, 1999.
- [9] Salutation Consortium. White Paper: Salutation Architecture: Overview. <http://www.salutation.org/whitepaper/originalwp.pdf> , 1998.
- [10] E. Guttman, C. Perkins, J. Veizades, and M. Day, .Service Location Protocol, Version 2,. IETF RFC 2608, June 1999. <http://www.ietf.org/rfc/rfc2608.txt>

- [11]Rekesh John, .UPnP, Jini and Salutation . A look at some popular coordination frameworks for future networked devices,. California Software Labs, June 17, 1999. <http://www.cswl.com/whiteppr/tech/upnp.html>
- [12] James Kempf, Ryan Moats, and Pete St. Pierre. Conversion of LDAP Schemas to and from SLP Templates. Internet Draft, October 1999.
- [13] Salutation Consortium, .Salutation Architecture Specification Version 2.0c . The Salutation Consortium, June 1, 1999. <http://www.salutation.org>
- [14] Universal Plug and Play Forum. Universal Plug and Play Device Architecture. Version 0.91, March 2000.
- [15] Microsoft Corporation, .Universal Plug and Play Device Architecture Version 1.0,. June 8, 2000. http://www.upnp.org/UpnPDevice_Architecture_1.0.htm
- [16] Bluetooth Consortium. Specification of the Bluetooth System Profiles Version 1.0 B: Part K:2, Service Discovery Application Profile,. Dec 1, 1999. <http://www.bluetooth.com/developer/specification/specification.asp>
- [17] Nidd M. Service Discovery in DEAPspace, IEEE Personal Communications, August 2001.
- [18]Sun Microsystems, .Java 2 Platform Midcro Edition (J2ME) Technology for Creating Mobile Devices,. White Paper, May 19, 2000. <http://java.sun.com/products/cldc/wp/KVMwp.pdf>
- [19] Celeste Campo, Service Discovery in Pervasive MultiAgent Systems Celeste. Dept. Telematic Engineering Universidad Carlos III de Madrid Avda. Universidad 30 Legan'es (Madrid), Spain celeste@it.uc3m.es. Workshop on Ubiquitous Agents on embedded, wearable, and mobile devices 2002 Bolonia, Italy.
- [20] Zhou Wang, Jochen Seitz, An Agent-based Distributed Service Model for Nomadic Users, Institute of Telematics, University of Karlsruhe D-76128 Karlsruhe, Germany
e-mail: (zhouwang,seitz)@telematik.informatik.uni-karlsruhe.
- [21] A.D. Joseph, M. F. Kaashoek, "Building Reliable Mobile-Aware Applications using the Rover Toolkit", Proceedings of the 2nd ACM International Conference on Mobile. Computing and Networking, New York, November 1996, pp. 117-129.
Proxy Architecture for Mobile Multimedia Applications", Proceedings of the 2nd IFIP/IEEE International Conference on Management of Multimedia Networks and Services '98, Versailles, France, November 1998