A distributed component-based middleware for the discovery of information appliances

François Lévesque[1] and Michel Barbeau[2]

Abstract— **This paper is about a middleware for the discovery of services provided by information appliances from Personal Digital Assistants (PDAs). Our middleware makes possible automatic configuration of applications on PDAs such that they can access services offered over networks to which they are attached. The architecture of such a system is distributed. In our approach, information appliances are abstracted as CORBA servant components. This approach makes applications independent of implementation of information appliances and provides uniform interfaces to them.**

Keywords— **Pervasive Information System, Resource Discovery, Service Location, Protocol, Distributed component, Distributed object.**

I. Introduction

The concept of *Pervasive Information System* is defined by Birnbaum as computers hidden in so called information appliances such as washing machines, furnaces, and refrigerators [1]. The pervasive information system exploits small, specialized, and embedded processors with standardized interfaces of a client-server model. In this vision, the qualifying *pervasive* is used to stress the fact that computers are so commonplace and embedded in the environment that they are more noticeable by their absence than their presence. The big surprise is to uncover the absence of a processor embedded within an appliance rather than the opposite.

Many new technologies are required to make the concept of pervasive information system a reality. Numerous problems are related to wireless telecommunications and development of middleware. This paper addresses an aspect called service discovery. In particular, we consider a situation in which users with Personal Digital Assistants (PDAs) dynamically discover services provided by information appliances in their environment.

In this paper, we present a middleware based on distributed components with parts running on PDAs for the discovery of services and parts running on IP networks for the advertisement of services of information appliances.

In Section II, we review the service discovery problem, how it is currently addressed by IP networks, and some recent developments. In Section III, we present our distributed component-based middleware for service discovery and advertisement. Its efficiency is analyzed in Section IV. We conclude with Section V.

1. Communications Research Centre, 3701 Carling avenue, P.O. Box 11490, Terminal H, Nepean, Ontario, Canada K2H 8S2
2. School of Computer Science, Carleton University, 1125 Colonel By Drive, Ottawa, Ontario, Canada K1S 5B6

II. Current Practice and recent developments

Two basic architectures of systems of entities discovering and advertising services are identified by Kempf and St. Pierre [2]. The first architecture involves only Service Agents (SAs), providers of services, and User Agents (UAs), users of services communicating directly with each other. With the aim of reducing traffic, a second architecture is proposed involving SAs, UAs, and Directory Agents (DAs) acting as central sources of information and to which SAs register services and UAs enquire about services.

For machine-to-machine connection on the Internet, the IP address of the computer of an SA and port number on which it is listening are required by a UA. Hence, there is the problem of mapping service names to machine names and port numbers on which services are offered.

For mapping service names to names of machines offering services, the Internet currently relies on naming conventions for machines offering services (e.g. www.scs.carleton.ca) and registered associations with a DNS of standard machine name and real machine address (e.g. www.scs.carleton.ca is mapped to fusion.scs.carleton.ca).

However, there is a limitation to this approach. Only one machine per DNS can offer a service under a standardized name. A solution to this problem has been proposed [3]. It is called DNS SRV Resource Records. It allows mapping a service name (as defined in file /etc/services, e.g. FTP) to names of machines offering the service. Machine names then have to be mapped to machine addresses. It is based both on a local name server (DNS) and a local file listing associations of machine name and IP address (file /etc/hosts on Unix). For mapping service names to port numbers on which services are offered, port numbers are standardized. Associations of service name to port number and transport protocol name are stored in a local file (e.g. /etc/services on Unix).

The advantage of the current practice in IP networks is that there are no requirements for special infrastructure for service discovery and advertisement. It is supported by in-place naming services. There is, however, the impossibility of advertising multiple machines offering the same service under the same name (unless DNS SRV Resource Records are used). UAs must be aware of naming conventions. Search by attributes is not supported. Machine names and port numbers are discovered using different mechanisms. Updates need to be performed at two different places. Introduction of new types of services requires standardization of new names. Information may not be up-to-date. Finally, there is the absence of a generic solution for all kinds of hardware and software.

The Service Location Protocol (SLP) [4] has recently been introduced as a general protocol for advertisement and location of network services at the scale of an enterprise IP network. In contrast to the aforementioned mechanisms, advertisement and location of services is by name and by characteristics.

SLP support two models of interaction: a model not involving DAs and a model involving DAs. Without DAs, UAs send using UDP over IP multicast or broadcast message Service Request to SAs. SAs are listening and when they find a match between a requested service and a service they offer, they reply to the UAs using unicast. DAs, UAs, and SAs are members of scopes, which are groups of agents. UAs and SAs communicate with DAs supporting their scope by using unicast UDP or TCP. SAs register their offer of service with DAs in their scope(s). UAs sends queries to the DAs supporting their scope(s). If a DA finds a match between a requested service and a registered service, it replies to the UA.

It is important to stress that the de facto standardized mechanisms used within the Internet and SLP solely address the service discovery and advertisement aspects. The use of a service provided by a SA requires the knowledge by a UA of a protocol specific for communication with the service (e.g. *lpr* for a printer service). Knowledge and support of several protocols are required whenever several services are used. Our approach, described in the next section, unifies the protocols used for discovery of services and communication with services.

III. Distributed component-based middleware for service discovery and advertisement

We introduce the architecture of an original middleware for the advertisement of services on IP networks and their discovery on PDAs. Our middleware makes PDAs automatically aware of the information appliances in their environment. A network administrator can register, update, and deregister services. Applications on PDAs specify the services they need, and the information appliances providing them are discovered automatically. The architecture is based on CORBA distributed components [5].

We first present a conceptualization of the discovery problem through an example. Then, we discuss the architecture of our middleware.

A. Conceptualization

An example is pictured in Figure 1. The UA is a PDA, the SA is an electronic projector, and there is a DA (server at the bottom). The SA is registered with the DA (1). The UA sends a request to the DA specifying the properties of a service, i.e. a projector (2). The DA returns a service access point which is the information required to communicate with the SA (3). Finally, the UA communicates and uses the service provided by the SA (4).

From a user interface point of view, a window is presented containing a list enumerating names and properties of services discovered during previous discovery processes (see Figure 2).

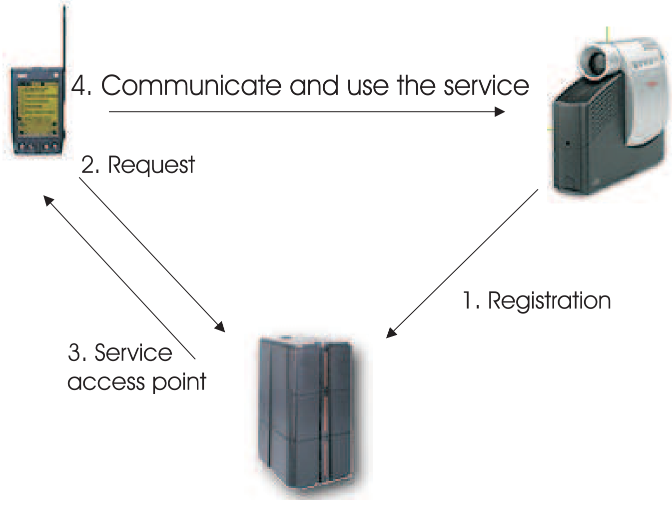

Fig. 1. A service discovery example.

The content of the list can be updated and a new discovery process can be launched. The user must first specify the type name and properties of the required service (see Figure 3).

B. Architecture and implementation

The architecture of our middleware is pictured in Figure 4. A client application, which acts as a UA, runs on a PDA. This application discovers and displays the services to a user. Through a management application, a network manager registers CORBA servant components modeling information appliances. Servants represent services and hide their implementation. When registered, the DA stores a serialized representation of them.

A DA registers itself with a CORBA naming service and then becomes visible to the UAs.

When a UA launches a discovery process, a request containing a type name and values of attributes of a service is sent to a DA. The DA finds matches and returns object references (i.e. addresses) in response to the UA.

When a UA requests a service, an object adapter is responsible for activating and initializing the CORBA servant component modeling the information appliance.

All interactions go through the CORBA Object Request Broker (ORB), a platform independent communication bus. The ORB is used both for service discovery and service use. Hence, a single uniform ORB interface is used in both cases and needs to be supported by the UAs.

IV. Efficiency

Pervasive Information Systems rely on wireless networks which bandwidth is a scare resource that must be used efficiently. The equations below characterize the bandwidth usage of our middleware.

Our CORBA based system uses the IIOP protocol for the communication between a DA, UAs, and SAs. Be-

Fig. 2. Enumeration of services.

Fig. 3. Specification of services.

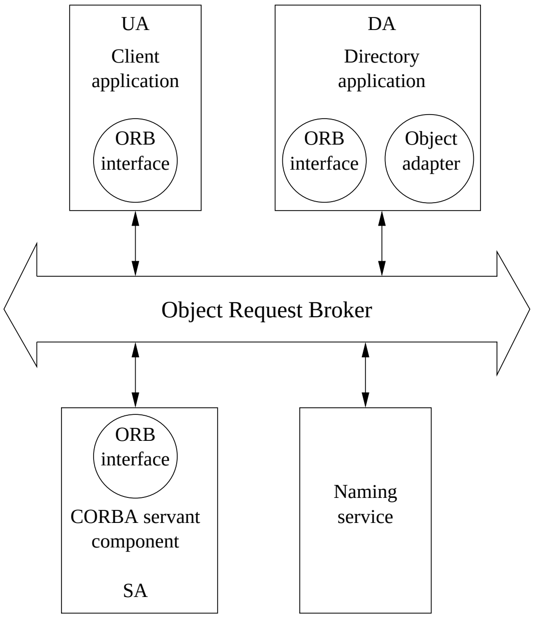

Fig. 4. Architecture.

cause applications on the PDA do not use a full CORBA implementation but only the IIOP part, we use a custom Naming Service (NS) through which clients get an Interoperable Object Reference (IOR) of a DA.

In our system, UAs create an object reference to a DA with the aim of sending service requests to the DA. The creation of an object reference implies that a request (NSReq) is sent to an NS for a specific DA and that the NS returns the IOR (NSRep) of the DA. A TCP connection is established with the NS before making the request and is released afterwards. Once a UA has the DA object reference, it sends a service request (SrvReq) to the DA. The DA replies with a message (SrvRep) that contains the list of all SA IORs providing the requested service.

IIOP is over TCP, thus messages exchanged over IIOP share a common header and are encapsulated within a TCP header, an IP header, and a network header (e.g. Ethernet header). Since using TCP requires establishing or releasing a connection, the variables ETCPCONN and ETCPCLOSE represent the number of bytes associated with these actions. In our model, variables ENSRQST, ENSREP, ESRVRQST, ESRVREP, and IOR respectively represent the size of an encapsulated NSReq, encapsulated NSRep, encapsulated SrvReq, encapsulated SrvRep, and IOR.

Let N be the number of SAs, M be the number of UAs, R the reference request period to a DA, and L the service request period. Over a duration of T seconds, in the presence of a single DA, the amount of traffic, in bytes, generated for obtaining the reference to a DA is given by the expression:

$$\begin{aligned} Ref &= M * T/R * (ETCPCONN \\ &+ ENSRQST + ENSREP \\ &+ ETCPCLOSE + ETCPCONN) \end{aligned}$$

The amount of traffic generated for obtaining the reference to a DA equals the number of UAs time the frequency of reference requests times the size of the overall data, which consists of the number of bytes required to establish a TCP connection with the NS plus the size of an encapsulated NSReq plus the size of an encapsulated NSRep plus the number of bytes exchanged to release the connection with the NS.

The amount of bandwidth due to the SrvReq messages is expressed by the following formula:

$$Req = M * T/L * ESRVRQST$$

The amount of bandwidth due to the SrvReq messages equals the number of UAs times the service request frequency times the size of an encapsulated SrvReq.

All IORs of SAs offering to the required service are packed in a single SrvRep message. The amount of bandwidth used by the SrvRep messages corresponds to:

$$Rep = M * T/L * (ESRVREP + N * IOR)$$

The amount of bandwidth used by the SrvRep messages equals the number of UAs times the service request frequency times the size of an encapsulated SrvRep along with N IORs.

The overall amount of bandwidth generated in a configuration of N SAs and M UAs over a period of T seconds is modelled by the following equation:

$$BU = Ref + Req + Rep$$

The overall amount of bandwidth represented in this equation equals the traffic due to the retrieval of a DA reference by the UAs along with the traffic generated by the SrvReq messages and the SrvRep messages returned by the DA.

Figure 5 plots the usage of bandwidth, by the messages SrvRep and SrvReq, if T is set to one hour and L to 15 minutes (abstraction is made of the extraction of retrieval of the DA reference, data elements exchanged in messages are assigned representative values). The bandwidth usage is more sensitive to the number of UAs than to the number of SAs. More than the half of the usage of bandwidth is due to TCP connection traffic and the large size of the IORs.

A bandwidth usage analysis for SLP is presented in Ref. [6] for a configuration with a DA, with N SAs, and M UAs. For comparison purposes, Figure 6 plots the usage of bandwidth by SLP with an equivalent configuration for the service request and service replies to and from a DA. With respect to Figure 5, the amount of bandwidth used is four times lower.

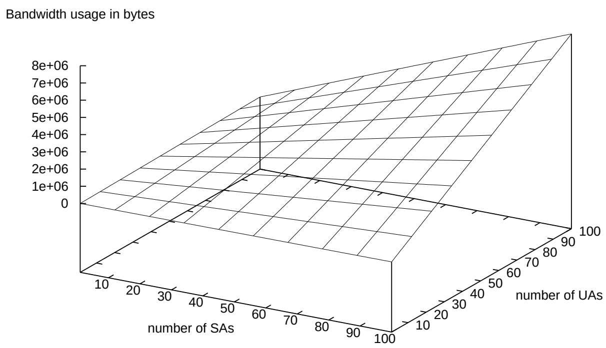

Fig. 5. Bandwidth usage of our middleware.

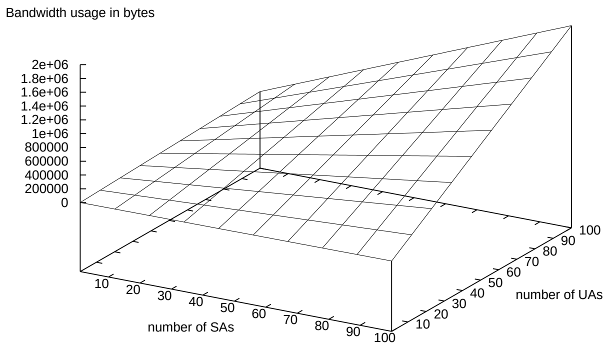

Fig. 6. Bandwidth usage of SLP (Service requests and replies only).

V. Conclusion

We have presented the architecture of a middleware for the advertisement over networks of information appliances abstracted as CORBA components and their discovery on PDAs through an ORB interface.

Approaches proposed for this type of problem require, in addition to a discovery and advertisement protocol, other protocols for communication and the use of services. This means that several protocols need to be supported within PDAs. An advantage of our approach is that a single protocol is required to handle both discovery of services and communication with services. No service specific communication protocol is required. Information appliances abstracted as distributed components have a uniform interface with hidden implementation details.

The architecture of our middleware is comparable, for instance, to an SLP architecture in which a DA is involved. In contrast to SLP which addresses only the advertisement and discovery of SAs, we address as well communication with SAs having a single uniform interface. On the one hand, usage of bandwidth is higher with our approach than with SLP. On the other hand, a single protocol needs to be supported in the PDA, the ORB interface. This sets the footprint of applications on PDAs to a fixed amount of memory (for at least the resource discovery and usage parts). In addition, our approach uses TCP, in contrast to SLP that uses UDP by default. Reliability is better with TCP.

References

[1] J Birnbaum, "Pervasive information systems," *Communications of the ACM*, vol. 40, no. 2, pp. 40–41, February 1997.
[2] J. Kempf and P. St. Pierre, *Service Location Protocol for Enterprise Networks*, Wiley, 1999.
[3] A. Gulbrandsen and P. Vixie, "A DNS RR for specifying the location of services (dns srv)," IETF Request for Comments: 2052, October 1996.
[4] E. Guttman, C. Perkins, J. Veizades, and M. Day, "Service location protocol, version 2," IETF Request for Comments: 2608, June 1999.
[5] Object Management Group, "The Common Object Request Broker: Architecture and specification," ftp.omg.org, 1999.
[6] M. Barbeau, "Bandwidth usage analysis of Service Location Protocol," Submitted for publication, 2000.

François Lévesque is a researcher in Satellite Software Communications at the Communications Research Centre in Ottawa, Canada. His present research activity focuses on improving satellite network resource management and on software radio architecture for satellite communications. Mr. Lévesque received a BSc in Computer Science in 1996 from the Uniservité de Sherbrooke. In 2000, he received an MSc in Computer Science from the Uniservité de Sherbrooke specializing in resource management and location on personal digital assistant handheld device.

Michel Barbeau holds a Bachelor's degree in Computer Science from Uniservité de Sherbrooke (1985), a Master's degree in Computer Science from the Uniservité de Montréal (1987), and a PhD in Computer Science from the Uniservité de Montreal (1991). During his doctoral studies, he was awarded Governor General's academic gold medal in recognition of exceptional scholastic achievement.

Since 2000, Michel Barbeau has been an associate professor at Carleton University (Ottawa). He was a professor at the Uniservité de Sherbooke from 1991 to 1999, a visiting researcher of at the University of Aizu (Japan) from 1998 to 1999, and a research assistant at INRS-Telecommunications (Montréal) from 1987 to 1998.

Dr. Barbeau's main areas of research are computer networks, wireless computer networks, telecommunications protocols, satellite telecommunications, software engineering, and real time systems.