JESA Service Discovery Protocol

Efficient Service Discovery in Ad-Hoc Networks

Stephan Preuß

University of Rostock; Dept. of Computer Science;
Chair for Information and Communication Services*

<mailto:spr@informatik.uni-rostock.de>

<http://wwiuk.informatik.uni-rostock.de/staff/spr.html>

Abstract. Pervasive computing requires management techniques allowing efficient service handling in volatile contexts. The Java Enhanced Service Architecture (JESA) is a service platform addressing this issue for resource limited devices. A major problem in dynamic service networks is the discovery of desired services. The *JESA Service Discovery Protocol (JSDP)*, one of JESA's core components, is a lightweight, platform independent service discovery protocol for ad-hoc networks. JSDP features transparent operation with or without central service brokers providing scalability from point-to-point connections to larger structured networks.

Keywords: service discovery, pervasive computing, ad-hoc networking

Classification (CR 1998): C.2.2, C.2.3, C.2.4

1 Introduction and Motivation

In ad-hoc networking, client nodes enter an initially unknown territory for using network services. During the *service discovery* process, they gather information about their surrounding service context avoiding the inflexible use of only well known or preconfigured services. A discovery technology meeting a broad range of application areas is characterized by: *scalability concerning service count and resource usage, platform independence concerning computing and network, low complexity for easy application development, and application transparency*. The existing discovery technologies (e.g. SLP [1], Ninja SDS [2], SSDP [3], Jini [4], Salutation [5]) do not meet all of the above characteristics.

Especially, the need for a service platform applicable to resource limited embedded or mobile devices as well as to desktop and server systems led to the development of the *Java Enhanced Service Architecture (JESA)* [6] – a lightweight, Java-based middleware for spontaneous service discovery and usage. It is mainly intended to be used in industry and home automation as well as mobile computing. One of its core components is the *JESA Service Discovery Protocol (JSDP)* – an efficient discovery mechanism offering operation modes of different complexity to be applicable to a wide variety of device classes.

* Supported by a grant of the Heinz Nixdorf Stiftung.

2 Service Discovery

The establishment of service relations in volatile network environments requires the nodes to become aware of their service context and context changes. *The information necessary for using a specific service is gathered in the service discovery process.* Service discovery comprises at least one of the following items: *locating the service provider, acquiring additional service or provider information, retrieving the provider's access interface (proxy, stub, etc.).*

Service discovery can be classified according to the level of initial knowledge, the relation, the count, and the activity of the involved entities. The distinct discovery classes build a hierarchical structure as laid out in Fig. 1. Service dis-

Fig. 1. Service Discovery Categories.

covery is split up into two top-level categories. *Preconfigured* discovering entities know either about the desired service provider or whom to ask for that information. In contrast, *non-configured* entities are innocent regarding the service context. This is the typical situation for ad-hoc networks. Further subdivision relates to the number of involved entities. *Location-aware* and *immediate* discovery classes are characterized by direct relation between client and provider. The provider itself supplies the client with all necessary information. In *mediated* modes, service brokers deal discovery information on behalf of providers. At the bottom-level, five discovery categories are distinguished: *Location-aware* clients know the logical location of their desired service, hence discovery reduces to getting additional information like service access interface or service characteristics; In *active* mode, a client initiates a request response procedure by broadcasting a request for a certain service. Appropriate providers respond at least with the location data of their service; *Passive* mode, in contrast, releases clients from inquiry and obliges providers to announce their services; Both *mediated* modes work with central information brokers. For proper operation providers have to register their service data with a broker. For that purpose, a broker may be treated as a special service provider. Regular providers will be clients of the broker for the registration period and have to discover the broker service by any of the discovery means discussed here. *Transparent* and *non-transparent* mode differ in the client's awareness of the broker. If a client intentionally used a broker for finding services it discovers *non-transparently*. Whereas, if the client believed to interact directly with a provider but in fact it was a broker, the client discovers *transparently*.

3 Related Technologies

There is a number of ad-hoc service platforms deploying different discovery schemes: *SLP* offers a comprehensive message based discovery system for TCP/IP networks; *SSDP* is the discovery protocol used in *UPnP* [7], it mainly bases on UDP, HTTP, and XML; The discovery protocol of the Ninja project, *SDS*, enables secure service discovery with an infrastructure of service, accounting, and certificate directories; *Salutation* provides generic network independent service discovery and access by the use of network and service managers; *Jini* features Java-based, broker-centric discovery and service usage with proxies.

Table 1. Discovery Classes Supported by Current Protocols.

	Location-aware	Non-transparent	Transparent	Active	Passive
SLP		+		+	+
SSDP	+			+	+
SDS		+			
Salutation		+		+	+
Jini		+			
JSDP		(+)	+	+	+

Table 1 presents an overview of the protocols' discovery class conformance, according to the classification given in Section 2. Furthermore, it contains a forecast of the JSDP functionality.

4 JESA Service Discovery Protocol

JSDP has been designed as a lightweight discovery protocol for embedded and mobile systems. It provides generic functionality: *locating of service providers, retrieval of service proxies and service attributes*. Almost any discovery feature can be added leveraging service attributes. Integral security mechanisms have been left off the protocol to keep it small. A major goal is JSDP's ability to work transparently *with or without* a central service broker (see Section 2) thus offering the possibility of peer-to-peer communities of limited devices on the one side and large scalable service networks on the other side.

4.1 Discovery Strategies

JSDP incorporates three major discovery modes (see Fig. 2). The *immediate* ones (*active, passive*) are intended for short range discovery in the local network segment. The *transparent* mode enables long distance discovery across segment boundaries by using a broker hierarchy. Brokers may either forward service requests or share the registered service information. With the existing means, *non-transparent* discovery can be realized by special broker services having not

Fig. 2. JSDP Operation Modes: a) passive; b) active; c) transparent.

only a registration interface but a query interface, too. This architecture would be similar to Jini.

Service discovery is performed in two steps. In the first step, provider location information is gathered using the *Provider Location Protocol (PLP)*. If a client decides to further examine or to use a certain service, in the second step, proxy and/or attribute information will be retrieved by the *Proxy/Attribute Request Protocol (PARP)*.

4.2 Provider Location Protocol

The *Provider Location Protocol (PLP)* can be used in request response mode (active discovery) or in announcement mode (passive discovery) for acquiring service provider locations. For simplicity, it defines only a single message format for requests and announcements (see Fig. 3). Service queries or announcements contain only the service *type* because this is the most important selection criterion. If a client needs more information about a service it will continue with the attribute retrieval and come to a decision according to the service characteristics. The *group list* can be used to limit the service matches to certain administrative groups. A query will match a service if version and type are identical and the service is member of at least one listed group. The *ID list* can contain a set of *Universal Unique Identifiers (UUID)* which specify services a client does not want to get announced. A provider fills in this list a single UUID which is used by clients for further transactions. Although PLP is designed in

Version	Sender URI	Type	ID Count	ID List	Group Count	Group List
Int32	UTF8 String	UTF8 String	Int32	[Int128]	Int32	[UTF8 String]

Fig. 3. Service Request / Announcement Message.

a Java environment, it is not Java-specific. Hence, it can be used for service discovery in non-Java environments as well. The current PLP implementation bases on UDP. Requests or unsolicited announcements are delivered with UDP multicasts; alternatively, link-local broadcasts can be used. Solicited announcements (service responses) are sent by UDP unicasts. The message flow for active discovery accords to Fig. 2b. A client uses the *sender URI* field to tell potential responders where to send the answer. The sender URI field in the answer codes the continuation point for the next discovery steps, in fact where to contact the provider to get the attributes or the service access interface.

Transparent discovery mode uses the same message exchange procedure. From the client’s point of view nothing changes. Providers have to register their service data (location information and access interface) with a broker and stop answering requests or announcing services. Brokers should be implemented as regular JESA services discoverable by JSDP means. Now, the broker answers requests for registered services and delivers their proxies or attributes on request. The unsolicited announcement of registered services (passive discovery) is possible but not encouraged.

4.3 Proxy/Attribute Request Protocol

After successful completion of PLP, a client knows about at least one provider’s or broker’s location and can fetch a service proxy or service attributes using the *Proxy/Attribute Request Protocol (PARP)*. In contrast to PLP, PARP is more dedicated to a Java environment because it ships serialized Java objects (JESA service proxies and attributes [8,6]) across the network. PARP uses a stream connection, currently it is TCP, to request and transmit service data. The Request message (see Fig. 4) indicates in the tag field whether proxy or

Tag	Service UUID
int32	Int128

Fig. 4. PARP Request.

attribute list is to be transmitted. The service UUID field exactly identifies the target service and has been retrieved in a PLP response. In immediate modes, directly talking to a provider, the UUID would be almost superfluous. It is possible that in the time between PLP and PARP execution a certain provider leaves the community and a new one enters with the same logical location. Here, the UUID avoids the delivery of incorrect service data. In transparent mode, using a broker, the UUID is used to select the appropriate information out of the pool of registered service data. If a service matching the UUID is available service data, according to the tag field, will be returned to the client in form of serialized Java-objects.

For robustness, transparent operation mode can be split up. During PLP, the broker does not deliver its own URI for further discovery but the one of the original provider. Hence, proxy or attributes will be fetched from the real provider. Successful completion of this step ensures a properly working provider.

Clients should not cache service data over long periods and reuse them because there is no guarantee that a service remains available and does not change its logical location or characteristics. For most cases a “discover-use-forget” strategy will be appropriate.

5 Conclusion and Future Work

JSDP has been developed as integral discovery component of the Java Enhanced Service Architecture. It focuses on core discovery tasks and allows arbitrary extensions. By the seamless integration of immediate and mediated discovery modes JSDP scales with the service count. The current implementation is fully functional on top of TCP/IP networks. JSDP itself comes with a memory footprint of about 30K. Together with the NetObjects technology [8] for service execution, a properly working JESA system requires about 80K. JESA and hence JSDP require a Java1.1 compliant runtime environment and will fit to embedded or mobile systems running PersonalJava, Kaffe, Jeode, J2MECDC.

Ongoing developments will produce: 1) *Java Abstract Network (JANet)* which decouples JESA from TCP/IP, a reference implementation will run with the CAN-bus; 2) An OSGi [9] interface which automatically transforms OSGi services into JESA services discoverable by JSDP; 3) An integration of the Bluetooth SDP [10] into JSDP to avoid multiple discovery levels when applying JESA to Bluetooth devices.

References

1. Guttman, E., Perkins, C., Veizades, J., Day, M.: Service Location Protocol, Version 2. IETF Internet Draft, RFC 2608 (1999)
2. Czerwinski, S.E., Zhao, B.Y., Hodes, T.D., Joseph, A.D., Katz, R.H.: An Architecture for a Secure Service Discovery Service. In: Proceedings of the Mobicom 99, Seattle, Washington, USA, ACM (1999) 24–35
3. Goland, Y.Y., Cai, T., Leach, P., Gu, Y., Albright, S.: Simple Service Discovery Protocol/1.0. http://www.upnp.org/download/draft_cai_ssdp_v1_03.txt (1999)
4. Arnold, K., Wollrath, A., O'Sullivan, B., Scheifler, R., Waldo, J.: The Jini Specification. Addison-Wesley (1999)
5. Salutation Consortium Inc.: Salutation Architecture Specification V2.1. <ftp://ftp.salutation.org/salute/s21a1a21.pdf> (1998)
6. Preuß, S.: Java Enhanced Service Architecture. <http://wwwiuk.informatik.uni-rostock.de/~spr/jesa/> (2001)
7. UPnP Forum: Universal Plug and Play Connects Smart Devices. <http://www.upnp.org/> (1999)
8. Preuß, S.: NetObjects - Dynamische Proxy-Architektur für Jini. In: Proceedings of Net.ObjectDays 2000, Erfurt, c/o tranSIT GmbH (2000) 146–155
9. The Open Services Gateway Initiative: OSGi Service Gateway Specification. <http://www.osgi.org/> (2000)
10. SIG, B.: Core. In: Specification of the Bluetooth System. Volume 1. Bluetooth SIG (February 2001)