ICICS-PCM 2003
15-18 December 2003
Singapore

Service Discovery At Home

Vasughi Sundramoorthy, Hans Scholten, Pierre Jansen and Pieter Hartel
University of Twente, the Netherlands
{vasughi, scholten, jansen, pieter}@cs.utwente.nl

Abstract

Service discovery is a fairly new field that kicked off since the advent of ubiquitous computing and has been found essential in the making of intelligent networks by implementing automated discovery and remote control between devices. This paper provides an overview and comparison of several prominent service discovery mechanisms currently available. It also introduces the At Home Anywhere Service Discovery Protocol *(SDP@HA) design which improves on the current state of the art by accommodating resource lean devices, implementing a* dynamic leader election *for a central cataloguing device* and *embedding robustness to the service discovery architecture as an important criterion.*

Keywords: service discovery, ubiquitous computing, home networking.

1. Introduction

The objective of a service discovery mechanism is to develop a highly dynamic infrastructure where clients would be able to seek particular services of interest (e.g. printing, displaying) and devices providing those services (e.g. printer, laptop) would be able to announce or advertise their capabilities to the network without manual configuration and device driver installation. Furthermore, service discovery allows the network to be self-healing by automatic detection of services which have become unavailable. Resource lean devices could also take part in service discovery activities by being able to delegate some of their load to more powerful devices. Once services have been discovered, devices in the network could remotely control each other by adhering to some standard of communication. As a result, data redundancy is also eliminated as data stored in one place could be accessed by any other devices without having to be copied over.

This paper presents several state of the art in this field and introduces the Service Discovery Protocol for the [1]At Home Anywhere project (SDP@HA) which focuses on making the service discovery mechanism more *resource aware* and *robust*. The mechanisms for service search and control of remote devices are not explicitly detailed in this paper due to insufficient space.

[1] This work is sponsored by the Netherlands Organization for Scientific Research (NWO) under grant number 612.060.111, and by the IBM Equinox program.

2. State of the Art

Service discovery mechanisms [1-5] such as Service Location Protocol (SLP), Universal Plug and Play (UPnP), Jini and Bluetooth SDP have similar characteristics, but defer widely in implementation. Basically, they provide the mechanisms to allow *discovery* of services according to certain device/service types and *advertisement* of services if it is a peer-to-peer architecture, such as in UPnP and Bluetooth SDP, or *registration* of services to a central *catalogue*, such as in SLP and Jini, which enlists all available services in the network. Meanwhile, service usage time is restricted by an expiry time or *lease* to disallow indefinite use of a service and ensure garbage collection.

2.1 Service Location Protocol

The Service Location Protocol (SLP) was developed by the IETF SvrLoc working group. There are three major software entities in SLP: *User Agents (UA), Service Agents (SA)* and *Directory Agents (DA)*. UAs discover the location and attributes of the services that the devices they represent are requesting, while the SAs advertise the location and attributes of the services they represent. The existence of DA gives a cataloging facility of all services in the network. There are two types of systems in SLP: (1) with DA, where SAs register information on the services they represent to a DA, so that UAs could send a query to the DA and be returned with the contact information of the service they are looking for, and (2) without DA where a UA will multicast search requests to the network, and an SA matching the service types requested will unicast a reply.

In SLP, the architecture with DAs causes the system to be vulnerable to a single point of failure, leading to information loss and breakdown in service discovery. Furthermore, it adds an additional component to be administered. Nonetheless, SLP provides good search facilities with filters that allows attribute and predicate string search. However, it only provides the location and contact information of the remote service while the access of the service itself is left up to the implementer.

2.2 Universal Plug and Play

Microsoft's Universal Plug and Play (UPnP) relies heavily on IP and web technologies such as XML. There are three major components in UPnP: *control device*, *device* and *services*. The *control device* searches by means of multicast for interesting devices whose services it wants to use. The *device* is any appliance or device that has some

0-7803-8185-8/03/$17.00 © 2003 IEEE

services to offer. It might contain embedded devices, and could behave as a control device. Devices advertise their capabilities using multicast. *Services* meanwhile contain (1) a state table that is updated whenever the service state changes, (2) a control server that receives action requests from the control devices, executes them, updates the state table and sends responses and (3) an event server that sends notifications to service subscribers whenever a state change occurs. Service discovery in UPnP is through the Simple Service Discovery Protocol (SSDP). XML documents provide device/service descriptions along with URLs to the user interface of the remote device. Control of remote services are through SOAP [6] and XML parsing of action requests, while GENA [7] is used to publish notifications to subscribers when a service's state changes.

UPnP consume heavier resources than SLP to support GENA and SOAP web servers, and XML parsing. Although it is not vulnerable to a single point of failure like SLP, its purely peer-to-peer architecture increases network traffic due to extensive use of multicast messaging. Furthermore, resource starved devices would not be able to handle such extensive processing. UPnP's service search capability is inferior to SLP as it is restricted to only device/service type searches.

2.3 Jini

Jini was developed by Sun Microsystems and implemented using Java. All members of the network are known as *services*. The most important aspect of Jini is the *lookup service*, where every service will have to register with at least one lookup service. Although there is a *peer lookup* mechanism, where clients could search for services through multicast and receive replies from matching services, this is not the norm. Once the lookup service of a suitable group has been located, the service provider will upload its service proxy object. A client service in search of a certain service type will contact the lookup service to download the proxy object.

The proxy object could be the complete implementation of the service, in which the client executes the service entirely by itself, or it could be an RMI stub, which, when invoked by the client, will cause some action to be executed by the remote service. A private communication method can also be implemented, without the Jini client needing to know how the actual communication takes place with the remote service.

Using Java to implement Jini allows code mobility and services to be represented as Java class interfaces, but the downside is that every device in the Jini network has to run the Java Virtual Machine (JVM), which is a very demanding requirement for resource lean devices. Furthermore, a detailed standardization process is necessary if services are to be represented in Java classes.

2.4 Bluetooth Service Discovery Protocol

Bluetooth, developed by the Bluetooth Special Interest Group, is meant for low power, short range (10m), wireless radio system devices operating in the 2.4GHz ISM band. Bluetooth devices periodically sniff for nearby Bluetooth devices and form a personal area network called *piconet* which has a maximum of 8 members. The member that initiates communication becomes the master of the piconet. Groups of piconets communicating with each other are called *scatternets*. The Bluetooth Service Discovery Protocol (Bluetooth SDP) requires an *SDP server* running in any device that is capable of providing services. The server maintains a set of *service records* that contain a list of *attributes* which represent different *service classes*. Each service is represented by one service record. A Bluetooth device wanting to use a service is called an *SDP client*. The SDP client sends a request message, which includes the list of service classes it is looking for and the SDP server checks if a match occurs and responds with a *service handle* that will subsequently be used to learn the values of the service's attributes.

Like SLP, Bluetooth SDP does not provide a mechanism to access the remote service. However, the service provider does provide information on the appropriate communication protocol that should be used by the SDP client to gain access to its service. Another drawback is devices are restricted to short range service discovery. Furthermore, as in UPnP, the Bluetooth SDP has a purely peer-to-peer architecture without a cataloguing facility that is not suited for resource lean devices.

3. @HA Service Discovery Protocol

In the At Home Anywhere project (@HA) [8], home appliances are divided into three classes to distinguish their resource capacity, mainly memory and processing power:

- ***3C (3+ cent) appliance:*** simple devices that implement only a network stack to connect to the system.
- ***3D (3+ dollar) appliance:*** medium complex devices that implement a network stack and service discovery layer.
- ***300D (300+ dollar) appliance:*** powerful devices, controlled by a complex embedded computer. These devices may have 3C devices embedded in them. Their memory requirements are high (>1MB).

The @HA service discovery layer will only know of the existence of 3D and 300D devices as they will encapsulate the 3C devices as part of their services. By distinguishing devices into such classifications, the functions of the service discovery protocol could be partitioned according to their resource capabilities.

3.1 @HA Service Discovery Requirement

The *@HA Service Discovery Protocol (SDP@HA)* addresses these major issues which are lacking in the current state of the art:

- Participation of resource poor devices - the protocol for the @HA network is *device-aware*. Current technologies are suited only for devices in the 300D category. Thus every feature of the service discovery protocol must be considered whether it can support resource lean devices.

- Delegation of work load - since resource lean 3D devices might not be able to stand alone, workload has to be delegated to more powerful 300D devices.
- Robust architecture – it is vital to recognize that the home environment is unlike the professional office environment, where a system administrator attends to network maintenance. It is essential that the system is robust and able to recover from network errors automatically. Existing technologies do not give this area much prominence.

The SDP@HA makes several assumptions, including that the network has broadcast capability, the device/service type is preset according to some naming standard, the physical location of devices is known so user could search for services according to the rooms in his/her home and there is a reliable gateway for security authentication that ensures integrity of the devices in the network.

3.2 @HA Service Discovery Design

The architecture of the SDP@HA is a hybrid of client-server and peer-to-peer. The client-server model is incorporated so that there is a server-like device called the Central that reduces communication cost as search and service advertisement messages are addressed to one particular device instead of broadcasted to the entire network. It also provides aid to resource lean devices so their memory, processing power and energy usage are reduced and thus able to prolong their services. The client-server environment requires at least one 300D device to be available. It is assumed that a home has at least one appliance with more than 1MB of memory available, as in TV, PC, VCR, etc. The most powerful 300D device is elected to become the Central, which acts as a repository for service information in the network. When a Central is not available (no 300D device available, or a leader election process is being carried out, with no Central selected yet), devices requiring a service will enter *Peer Search* mode as explained later in this paper. The workings of SDP@HA are as explained in the following sections.

Central election: The Central election algorithm in SDP@HA is based on the resources of 300D devices. This algorithm allows any 300D device to be a potential Central, instead of making the Central a separate entity to be added to the network. Thus cost of additional administration is eliminated. Wireless 300D devices are given lower priority in Central election. This is because of energy constraints and possible higher mobility which make them unreliable candidates for the position of Central. A wireless 300D device could still become the Central in the absence of their wired peers because it could still provide service cataloguing facility for 3D devices and reduce the burden on their resources.

When a 300D device initializes, it will broadcast a MyResource message. If the 300D device does not receive any reply from an existing Central after a timeout, it assumes that Central is unavailable and elects itself as the Central and informs the network. If it does receive MyResource messages from other 300D devices, a Central election algorithm is started. Four parameters are taken into consideration as basis for Central election:

i. device orientation , e ={-1, 0, 1} where if:
   e = −1, wireless device (lowest priority)
   e = 0, wired devices with wireless capability
   e = 1, wired devices (highest priority)
i. processing power, p
ii. memory size, m
iii. unique device identifier, u

A 300D device (d) checks the parameters of a remote device (d') contained in the MyResource message received to determine whether it is superior. The values of e and p are given more priority thus will be compared first. Values of m and u will only be considered to break a tie between two devices with the same device orientation and processing power, because any device with a minimum available memory requirement of 1MB (300D) is capable of handling the job required of the Central. The unique device identifier, u, is universally unique (eg. MAC address), and thus can be used to break a tie between two devices of the same type by comparing which device has the larger identifier. For the algorithm, we need the following definitions:

Definition 1. *For any two tuples of equal length, $d = (x_1, \ldots, x_n)$ and $d' = (x'_1, \ldots, x'_n)$, the operator > on d and d' is defined such that one of the following is true:*
(1) $x_1 > x'_1$
(2) $\exists i \in \{2,\ldots n\}$ such that $x_i > x'_i$, and $x_j = x'_j, \forall j \in \{1, \ldots i\text{-}1\}$

Definition 2. *For any two tuples of equal length, $d = (x_1, \ldots, x_n)$ and $d' = (x'_1, \ldots, x'_n)$, the operator < on d and d' is defined such that one of the following is true:*
(1) $x_1 < x'_1$
(2) $\exists i \in \{2,\ldots n\}$ such that $x_i < x'_i$, and $x_j = x'_j, \forall j \in \{1, \ldots i\text{-}1\}$

Algorithm 1: For the Central election algorithm, the parameters of the local device, $d = (e, p, m, u)$ and remote device, $d' = (e', p', m', u')$ are compared and result in 2 scenarios:
Case 1: if (d < d'): Drop out of Central election
Case 2: if (d > d'): Create RankList in descending order

Finally, only one device should have a complete RankList with itself on top which becomes the Central. The Central will broadcast its status at least twice to ensure every member receives this message due to the unreliable nature of the broadcast network.

Communication failure could occur where not all MyResource messages are received. This would result in more than one device finding itself on top of its RankList. Negotiation of the Central position will be done between these devices, and one of them will be elected (not detailed in this paper).

Device and Service Registration: Once a Central has been selected, it will appoint the next device in the RankList as its Backup. The RankList will be backed up by the Backup device. Central then broadcasts a ServiceRegistration-Request, as shown in Figure 1. A device receiving this message sends a ServiceRegistrationReply containing the list of services it provides (represented by ServiceTypes), their attributes, and a lease period for each service. Central

will enlist them in the ServiceLookUp table, and generate a unique ServiceID for each of the service types. The ServiceLookUp table will also be back-ed up in the Backup device.

The lease period for each 300D service has to be renewed before its expiry time. If not, Central purges the data, assuming the device has gone offline. In the case of 3D devices, Central will poll them periodically, as these devices will not be able to send lease renewals constantly. Central sends updates to the Backup devices whenever a change occurs in the ServiceLookup table.

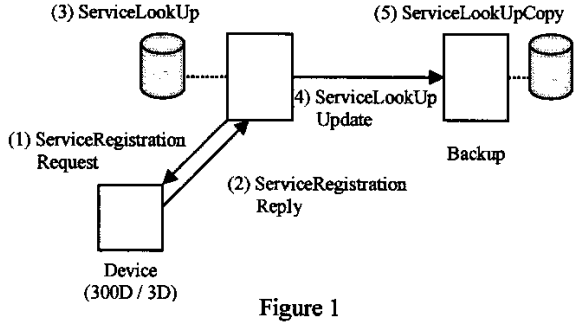

Figure 1

Search, Subscription and Control: When a device wants to look for a particular ServiceType with certain attributes, it sends a ServiceSearch message to Central. Central checks the ServiceLookUp table if a match occurs, and replies with a ServiceReply, containing the list of ServiceTypes that match. Here, the device can request for only the best match out of several possibilities. Once the best matched service is chosen, the device (now known as the Control device) can send an ActionRequest to the remote service. The remote service (Service Provider) executes the action and returns an ActionResponse, to indicate whether the action has been successful or not.

Control devices cache the best matched service and its lease period with the Central for future use. Optionally, they can also go into the Subscription mode for specific services. Subscription allows the Control device to reuse the same service without going through the Search process via the Central. Control devices become Subscribers by sending a SubscriptionRequest with a certain lease period to the remote service. Any change in the state of services will be notified to the Subscribers through event messages.

If the Service Provider is a 3D device, it might not be able to maintain a SubscriptionTable necessary to administer subscriptions. In which case, the Central will administer a CentralSubscriptionTable for all 3D Service Providers. SubscriptionRequests will be made to the Central, which, after receiving service updates from the 3D Service Provider when a state change occurs, will notify all its Subscribers.

Robust Architecture: The SDP@HA is robust because it (1) detects (through polling) and recovers from Central/ Backup failure, (2) allows any device to detect loss of services and inform Central, (3) allows any device to request Central election when Central is found missing, (4) hand over of Central/Backup functions to a more powerful new device, and (5) preserves service lifetime of resource lean devices.

Polling is used to ensure the online status of Central and Backup. Central polls the Backup periodically, to which, the Backup replies. Retransmission is used for reliability. When Central finds the Backup not responding, it will remove Backup from its ServiceLookUp table and RankList. The next device in RankList becomes the new Backup. On the other hand if the Backup finds the Central is not polling, it will elect itself as the new Central and inform the network of its new status. The rest of the members will reroute their messages to this new Central.

As mentioned earlier, service registration leases are renewed to indicate the online status of a 300D device, while 3D devices are polled periodically by Central. If an action requested by a Control device is not executed, it will send a notice to the Central. Central will poll the failing device to check if it is still alive. Meanwhile if a device finds the Central missing after a timeout, it may request for a Central election. All these steps lead towards detecting a missing service and add to the robustness of the system.

There are 2 scenarios when new devices are detected:

- *300D device detected*: the broadcasted MyResource message is used. When Central receives this message, it has to determine if the new device is superior in resources than itself or the Backup. Algorithm 2 shows how this is done:

Algorithm 2: For the Central/Backup handover, the parameters of the Central, $d = (e, p)$, the new device $d' = (e', p')$ and the Backup $d'' = (e'', p'')$ are compared. x is the threshold set to ensure that Central handover takes place only when the new device's processing power is sufficiently higher than the Central's. The *Definitions* used Central election are applied here.
Case 1: if (d<d') and (p'>x): Central handover to new device and Backup takeover by current Central
Case 2: if (d<d') and (p'<x): Backup handover to new device
Case 3: if (d>d') and (d'>d''): Backup handover to new device

- *3D device detected*: the 3D device broadcasts a Small-DeviceAnnounce message when it initializes. Upon receiving this, the Central recognizes it as a 3D device and provides the necessary support (polling for online status, subscription, etc). If a 3D device does not receive any response from Central, it resends the message every w period according to a delay of: w = 2 x (initial wait period) until maximum w, before resetting. This preserves resources especially for energy deficient wireless 3D devices. If during this time, the 3D device needs a service, it takes part in *Peer Search*, by broad-casting a ServiceSearch message to the network, thus allowing 3D devices to search for services even when no 300D devices are around, and allows 300D devices to search for services during Central election.

Table 1 compares critical features of the existing state of the art against the SDP@HA. Although the existence of the Central may be similar to SLP's architecture with DA, it is important to note that the Central is not simply a service broker, nor is it an additional entity to be installed in the

Table 1: Comparison of state of the art with SDP@HA

Feature	SLP	UPnP	Jini	Bluetooth SDP	SDP@HA
Architecture	Client-server / Peer-to-peer	Peer-to-peer	Client-server / Peer-to-peer *(peer lookup)*	Peer-to-peer	Client-server / Peer-to-peer *(Peer Search)*
Catalogue service	Directory Agent	No	Lookup service	No	Central
Leasing concept	Yes	Yes	Yes	No	Yes
Remote control	No	Yes	Yes	No	Yes
Scope of search	Service type Attributes String	Device type Service type	Service type ServiceID Attributes	Service type Attributes	Device type Service type Attributes
Robust	No	No	No	No	Yes *(with Backup)*
Resource awareness	No	No	No	No	Yes
Work load delegation	No	No	No	No	Yes

network. Any 300D device is a potential Central that can monitor the network for new devices according to their resources. With a dynamic backup mechanism available, the loss of the Central is not critical. Even if the backup mechanism fails, the network can reinitiate a new Central discovery without external configuration.

The SDP@HA is currently being simulated using *Rapide* [9]. Rapide supports constraint language and analysis tools that allow testing of the outputs of partially-ordered sets of events (POSETs) for violation of defined constraints and also test against consistency conditions. The simulation gives an overall view of the performance of the protocol, at different scenarios, and test how SDP@HA fares against communications and node failures.

4. Conclusion

The novel concept of searching, discovering and remotely controlling services introduces a highly dynamic behavior in today's networks. This paper provides a brief overview of existing service discovery mechanisms and introduced a more resource aware and robust SDP@HA design, which allows participation of resource lean devices and incorporates better error recovery mechanisms. Future work in this area will include performance analysis measurements against existing models [10] for benchmarking using Rapide and implementation of the protocol on top of @HA network layer [8].

References

[1] Guttman, E. Perkins, C., Veizades, J., and Day, M. "*Service Location Protocol, V.2*", Internet Engineering Task Force (IETF), RFC 2608.

[2] Microsoft. "*Universal Plug and Play Architecture, V1.0*", Jun 8, 2000.

[3] Ken Arnold et al. "*The Jini Specification, V1.0*", Addison-Wesley 1999. Latest version is 1.1.

[4] The Bluetooth SIG, Inc. "*Specification of the Bluetooth System, Core, Vol. 1*", Version 1.1, Feb 22, 2001, 1999.

[5] Christian Bettstetter and Christoph Renner. "*A Comparison Of Service Discovery Protocols and Implementation of The Service Location*" Proceedings of 6th EUNICE Open European Summer School: Innovative Internet Applications (EUNICE), Twente, Netherlands, Sept 13-15, 2000.

[6] Don Box, et al., "*Simple Object Access Protocol (SOAP) 1.1*", W3C Note; http://www.w3.org/TR/SOAP.

[7] Cohen, J., Aggarwal, S., and Goland, Y. "*General Event Notification Architecture Base: Client to Arbiter*" http://www.upnp.org/draft-cohen-gena-client-01.txt

[8] F.T.Y. Hanssen, P.H. Hartel, T. Hattink, P.G. Jansen, J. Scholten and J. Wijnberg. "*A Real-Time Ethernet Network at Home*", published by Real-Time Systems Group, Vienna Univ. of Technology, Vienna, Austria, Jun 2002.

[9] Luckham, D. "*Rapide: A Language and Toolset for Simulation of Distributed Systems by Partial Ordering of Events*", http://anna.stanford.edu/rapide, Aug, 1996.

[10] Dabrowski, C. and Mills, K. "*Analyzing Properties and Behavior of Service Discovery Protocols Using an Architecture-Based Approach*", Proceedings of Working Conference on Complex and Dynamic Systems Architecture, Brisbane, Australia, Dec 2001.