Time and Energy Efficient Service Discovery in Bluetooth

Igor Sedov[1], Stephan Preuß[2], Clemens Cap[3]
University of Rostock,
Dept. of Computer Science,
Chair[3] for Information and Communication Services,
Email: {igor, spr, cap}@informatik.uni-rostock.de

Marc Haase[1], Dirk Timmermann
University of Rostock,
Dept. of Electrical Engineering and Information Technology,
Institute of Applied Microelectronics and Computer Science,
Email: {marc.haase, dirk.timmermann}@etechnik.uni-rostock.de

***Abstract*— Key challenges in wireless mobile ad hoc networks are computational resource constraints, power limitations, and efficient service discovery techniques. The short range radio network technology Bluetooth suffers from long service discovery delays and high power consumption due to necessary connection establishment between discovering and discovered entity. For improving the efficiency of service discovery in Bluetooth networks we propose two new approaches. By leveraging the implicit broadcast capability of the Bluetooth inquiry procedure, we significantly reduce the service discovery time and hence lower the power consumption. Moreover, we enable the integration of resource limited devices which are incapable of taking part in service discovery themselves. Based on a performance evaluation and a comparison with the legacy Bluetooth Service Discovery Protocol (SDP), we present the benefits of our new approaches.**

I. Introduction and Motivation

Bluetooth is an emerging wireless technology for data and voice transmission within the 2.4 GHz band. One of the major application purposes of Bluetooth is the cable replacement for mobile and peripheral devices, e.g. PDAs, printers, smartphones, by establishing short range wireless ad hoc networks. The ad hoc network relationships are service based. Therefore the Bluetooth stack includes a service discovery protocol (SDP), proposed by the Bluetooth Special Interest Group (SIG) [Blu01]. SDP provides simple discovery mechanisms based on requesting service classes or service attributes successively from all devices in range. Moreover, the specification allows the usage of higher level service discovery protocols like Jini [Sun99], UPnP [UPn99], or Salutation [Sal02].

In environments with a high concentration of Bluetooth nodes the native Bluetooth SDP is inefficient due to the following reasons:

First, Bluetooth does only provide broadcast functionality after connection establishment between two or more devices within a piconet. That compromises the performance of the native Bluetooth SDP because in ad hoc environments moving nodes are changing the network topology all the time resulting in permanent discovery activities and hence in time consuming connection establishments. Routing is out of the scope of the Bluetooth specification, thus every network node is obliged to establish a connection with every other node in order to carry out service discovery. This leads to a high energy consumption and a significant time delay due to the connection establishments. Generic discovery protocols like Jini, UPnP, and Salutation are also affected. They usually rely on IP broadcasts or multicasts in order to find services or service brokers. IP over Bluetooth can be supplied by using PPP over RFCOMM or BNEP over L2CAP. Both approaches provide a broadcast capability that is unfortunately accompanied by a high communication overhead due the fact that a high level broadcast requires a properly established piconet. The current situation, characterized by long discovery delays, is unsatisfactory from a user's point of view and regarding power consumption.

Second, another drawback is that restricted nodes (with very limited memory and computational resources) cannot accomplish native SDP. For example, the oil sensor of a car engine collecting information about the oil volume, pressure, and temperature could propagate measured values to any requesting entity. However, because of the lack of a SDP server within the sensor only devices which exactly know the sensor's access parameters, e.g. the board computer, can communicate with it. All other devices are unable to retrieve the oil sensor service parameters.

The above mentioned disadvantages of native SDP in Bluetooth environments with high node concentrations leads us to the improvement of the existing native SDP. In this paper we will provide two different approaches for accelerating the service discovery procedure and hence reducing the energy consumption.

This paper is organized as follows. Section II briefly introduces the Bluetooth technology and reviews existing research activities. Section III presents our new service discovery models: *Coordinator-based Service Discovery* and *Peer-based Service Discovery*. Afterwards we analyze the benefits of our models and provide some initial results in Section IV. Section V concludes this paper.

II. Device and Service Discovery in Bluetooth

The Bluetooth specification includes a service discovery application profile which supports users of Bluetooth-enabled

[1]Supported by the German National Research Foundation DFG, program SPP 1079, *Sicherheit in der Informations- und Kommunikationstechnik.*

[2]Supported by the German National Research Foundation DFG, program Graduiertenkolleg 466, *Multimedia*.

[3]Supported by a grant of the *Heinz Nixdorf Stiftung*.

devices in finding communication partners with desired service characteristics. The service discovery process consists of several parts. First, information about available devices is collected using the *inquiry* procedure. Then, a peer-to-peer Bluetooth link (L2CAP) is established to a single node. Finally, SDP is utilized for searching desired services using a request/response scheme over the L2CAP transport protocol. Unlike other discovery protocols, native Bluetooth SDP does not provide enhanced discovery facilities like service advertisement or registration of service information with a central broker. If high-level discovery protocols, like Jini or Salutation, are deployed, native Bluetooth SDP must be executed in the first step in order to find the nodes which support the desired second level discovery method.

Current research activities focus on piconet and scatternet formation and communication scheduling algorithms inside scatternets as well as routing mechanisms between nodes. Law *et al.* [LMS02], Salonidis *et al.* [SBTL01], Marsan *et al.* [MCN+01] introduce new scatternet formation protocols. The main idea is to divide all nodes in independent piconets. These piconets are joined by shared slave nodes (bridges) to build a scatternet. However, the resulting network is a community of piconets, which are interconnected with each other. Concerning service discovery, the main drawback of this architecture is the necessity of a permanently connected piconet infrastructure with increased energy consumption for connection maintenance.

III. Enhanced Service Discovery Algorithms

In this section, we propose two new algorithms accelerating Bluetooth service discovery. The new schemes are backward-compatible to the Bluetooth specification version 1.1 including SDP version 1.0.

A. Common Basis

The major problem that prevents time and energy efficient service discovery among Bluetooth nodes is the missing broadcast capability in non-connected state. We overcome this problem by enhanced utilization of the *inquiry process*. Inquiry is used to find neighbor-devices. It is initiated by broadcasting an *ID packet*. All receivers respond with a *FHS packet* containing, among other values, the sender's Bluetooth device address and the *class of device (CoD)* value (see [Blu01]). The CoD value has a length of 24 bits and can be assigned by the application; it is neither hardware immanent nor fixed in the Bluetooth stack. It characterizes devices according to their functionality, in order to allow a pre-selection of potential communication peers prior to connection establishment.

The CoD value specification can be found in the Bluetooth assigned numbers specification [BTA02]. The CoD field structure is shortly depicted in figure 1. Although the CoD field format is defined, it is neither used in Bluetooth SDP nor for other real purposes.

The following two new proposals for service discovery make use of the CoD field in different ways.

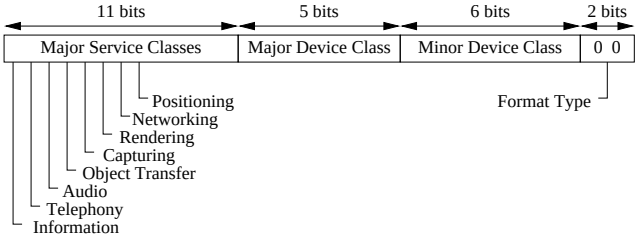

Fig. 1. Class of Device Field (first format type); taken from [BTA02].

B. Coordinator-based Service Discovery

1) Concept: In the first approach we rely on a central *coordinator node* which holds the complete service information of all Bluetooth nodes in range (community). This approach uses the CoD field for locating the coordinator. This method is most effective for scenarios with one central/fixed and many distributed/mobile nodes, e.g. a car's board computer or a technician's service terminal. Devices, which enter the coordinator's communication range, can discover the coordinator during the *inquiry process* by interpreting the CoD field. After knowing the coordinator node, they can request service information about all available devices from the coordinator or register their own services with the coordinator. The coordinator manages the complete set of service information of all devices in communication range. It has sufficient computational, power, and memory resources. This approach prevents mobile devices from accomplishing successive connection establishments and service discovery on all known devices. The costs for service discovery reduce to one connection establishment and a single service discovery process.

The algorithm consists of three phases: coordinator election, service registration and provision, as well as healing and error handling. Below we shortly describe each of this phases.

2) Coordinator election: Initially, assume there is no coordinator in range. This can happen during network initialization or when the previous coordinator disappeared. For electing a new coordinator, we propose to use the *Coordinator Election Algorithm* based on the lowest-ID algorithm and described in [SBTL01]. In the same manner as in [SBTL01], the winner of N-1 confrontations[1] will be the coordinator. In order to identify the coordinator, one bit of the reserved *major service classes* area of the CoD field is set (see figure 1). Each new node joining the community is now able to detect the coordinator during the inquiry procedure and can request service discovery information or can register its own services when staying longer in the community.

3) Service registration and provision: The coordinator collects service information from devices within the community. It uses native Bluetooth SDP to discover existing services and registers these in its own database. Thereafter, it provides the complete community service information.

If a device does not answer the request, the coordinator

[1]N is a number of all active nodes participating in the SDP procedure.

marks it as *unknown*. We enhance the existing Bluetooth SDP with an additional protocol data unit (PDU) called SDP_ServiceRegister PDU, to enable restricted nodes to register their services with the coordinator without own SDP server necessity. Figure 2 illustrates the format of the new PDU type. This PDU can be hard coded for restricted devices, so that they send always the same byte sequence for their services.

PDU Type	PDU ID	Parameters
SDP_ServiceRegister	08	AttributeListByteCount, AttributeLists, ContinuationState

Fig. 2. SDP_ServiceRegistration PDU

The parameters of the SDP_ServiceRegister PDU correspond to the SDP_ServiceSearchAttributeResponse PDU from the SDP specification [Blu01]. The *AttributeListByteCount* parameter contains the size of the *AttributeLists* parameters in bytes. *AttributeLists* parameter contains a standardized *Data Element Sequence* described in the part E of the [Blu01]. The *ContinuationState* parameter indicates, that this SDP_ServiceRegister request does not fit in a single registration PDU. Furthermore, we add a new Attribute ID *Remote BD_Addr*, so that service and service provider can be associated.

4) Healing and error handling: Self-healing and error handling are important requirements in wireless distributed systems. Nodes change the network topology all the time and in some circumstances it can occur, that there is no coordinator in the environment or there are two or more coordinator nodes at the same time. To solve this problem the native SDP must be enhanced with three new signaling parameters which are added to the SDP_ErrorResponse PDU. *SuccessfulRegistration* confirms the registration of remote services, *TooManyCoordinators* indicates, that, currently, there are two or more coordinators in this environment, and *NoCoordinator* informs other nodes about the absence of any coordinators.

Generally, nodes operate collaboratively. That means, if a node discovers a fault in the network, it informs corresponding devices about this event. For example, a node, which discovers the absence of any coordinator, and itself cannot be a coordinator, informs all other nodes with *NoCoordinator* signaling messages. All nodes, which received this message, switch into the coordinator election state for electing a new coordinator. The other error signaling mechanisms work similarly, so that the community is self-healing.

C. Peer-based Service Discovery

1) Concept: The second proposal for leveraging the CoD field in service discovery uses a new Format Type definition. While there is only one defined CoD format up to now, there is room for new ones (see format type field in figure 1 and figure 3).

In order to enable *fast* immediate discovery (without a central coordinator), we need a way to determine service information without executing SDP separately on each node.

Because it is impossible to broadcast application defined service requests so that nodes providing appropriate services can respond directly, service providers must be initiated to offer their service(s). The only possibility is the usage of the inquiry procedure. The inquiry request (ID packet) is the initiator message while the inquiry responses (FHS packets) contain the exact service types and further information in the CoD field.

Unfortunately the CoD field can carry at most 22 definable bits (see Figure 1) but type information in conjunction with other information easily exceeds that size. Hence, we have to compress service data. Assuming that the service information only has to be matched against the request and that it does not need any further processing, we can apply *hash or MAC (Message Authentication Code) functions* to create unique signatures of the service information. The used hash function does not need to fulfill any security requirements, but it should be *collision free*, in order to avoid mismatches. Applicable hash algorithms are for instance variants of MD5 (Message Digest), SHA1 (Secure Hash Algorithm), MAA (Message Authentication Algorithm) (see [Sch95]). All these algorithms produce signatures with a length of more than 22 bits, thus they have to be adapted to fit the size. Figure 3 shows the resulting CoD format. This approach has limitations: A service

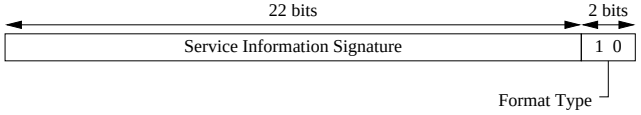

Fig. 3. Class of Device Field (second format type).

provider can only offer one service because a client usually searches only a single service per request; Service information must not need further processing at the client side because it cannot be rebuilt from the signature; The matching has to be exact because of the nature of the signature.

Example: A Bluetooth equipped centralized door locking (CDL) is the service provider, the board computer is the client. The CDL is configured to belong to a car with the license plate number HRO-XC007 so it sets up its service information to be of *type CDL* and of *scope HRO-XC007*. From that information it creates a signature and configures its Bluetooth stack with that value in the CoD field. The board computer starts to discover all its peripherals by inquiring the Bluetooth space. Because it stands in an underground car park there are lots of responses. Out of its license plate number and the desired type of peripheral, here CDL, it builds a signature and scans the CoD fields of all inquiry responses. This way, the board computer can discover all of its peripherals within a single inquiry procedure.

2) Application to Existing Discovery Protocols:

a) JESA Service Discovery Protocol: The *Java Enhanced Service Architecture (JESA)* [Pre01] is a service platform for spontaneous networking of resource limited devices. It relies on the *JESA Service Discovery Protocol (JSDP)* [Pre02] for finding network services. In *passive discovery*

mode JSDP-clients expect unsolicited announcements from service providers in the format shown in figure 4. Because

Field	Type
Version Number	`int`
Sender URI	`UTF8-String`
Service Type	`UTF8-String`
Service Count	`int`
Announced Service List	`UUID[]`
Scope Count	`int`
Discovery Scope List	`UTF8-String[]`

Fig. 4. JSDP Packet Format.

of the limitations of the Bluetooth peer discovery technique, it is impossible to transmit the service information in the first discovery phase using the CoD field. Thus the procedure is split up. In the first step, the fixed parts of the service information, which the client would have used in its request, are used to create a service signature at the provider side. Here apply the service type and the scope, but for the scope we have to limit the length of the scope list to a single entry. The client performs peer discovery and determines a list of potential service providers out of the inquiry responses based on the service signatures. Depending on the overall response count, the resulting list is much shorter; In the best case it has a length of one. With the matching providers, the client establishes a connection and sends a legacy JSDP request using a UDP unicast in order to get the full service information.

In order to apply peer discovery to JSDP, no changes are required in the original protocol. The provider only has to put its service signature into the CoD field, and the client has to perform an additional inquiry procedure.

b) Service Location Protocol: The *Service Location Protocol (SLP)* [GPVD99] supports service discovery immediately between client (User Agent) and provider (Service Agent) as well as mediated by a service broker (Directory Agent). Similar to the above JSDP approach we can speed up SLP's immediate mode in Bluetooth environments.

Usually, clients use UDP multicasts to send requests containing service type, scope, and a boolean expression for matching service attributes. In responses they get URLs characterizing type and location of a service (e.g. `service:ColorFax://cfax.informatik.uni-rostock.de:222`). It makes no sense to post the signature of a service URL in the inquiry response because a client cannot match it. Thus we build the signature from values the client has in its request - the service type and scope or only the type. Any other attribute information cannot be used because the provider has no idea about the client's request.

The discovery procedure is identical to the one used for JSDP. Starting with the pre-selection of potential providers, the client continues with legacy unicast based SLP discovery procedure with the providers matching its requirements in the first phase. SLP supports discovery via UDP or TCP unicasts.

c) Simple Service Discovery Protocol: The *Simple Service Discovery Protocol (SSDP)* [GCL+99] is used by *Universal Plug & Play (UPnP)* [UPn99] to locate service providers. SSDP can only operate in immediate mode. Requests and responses are sent using HTTP either over UDP or TCP.

SSDP Request

```
M-Search * HTTP/1.1
S: uuid:cdefghilk-01jan-2001-0099-1234567890xx
Host: 239.255.255.250:1900
Man: "ssdp:discover"
ST: ce:CDL
Scope: car:LPN/HRO-XC007
MX: 3
```

SSDP Response

```
HTTP/1.1 200 OK
S: uuid:cdefghilk-01jan-2001-0099-1234567890xx
Cache-Control: max-age = 5000
Ext:
ST: ce:CDL
USN: uuid:cdl123-01jan-2001-0099-12345678
Location: <http://hro-xc007.car/CDL>
```

Fig. 5. SSDP Request and Response Examples.

The lines tagged `ST` (Search Target) and `Scope` in the service request in figure 5 are the only search criteria in the first SSDP phase. Appropriate providers answer with the URL of the service description document (see `Location`-tag in figure 5) which can be retrieved in the second SSDP phase.

Depending on the actual purpose, we can use both, search target and scope, or only the search target to generate the service signature. The location of the service description has to be determined after potential service providers have been found during inquiry.

IV. Implementation and Performance Evaluation

A. Duration of Native Service Discovery Protocol

In order to verify the optimization achieved by our approach we measured the performance of the native SDP. Therefor we used two USB Bluetooth modules with CSR chip sets connected to Linux workstations. We measured the time for the connection establishment T_{Con} for a full service discovery process (except the inquiry procedure) and the service discovery time T_{NSDP} (SDP request transmission, SDP server request processing, SDP response transmission). Figure 6 shows the variation of discovery delays for distinct service counts (10 values at a time for 1, 5, 10, 20 services).

We observed, that the main time consumption resulted from the connection establishment. The peak value of T_{Con} reached 17 seconds. The processing time for service discovery T_{NSDP} is compared to T_{Con} negligibly small and nearly independent from the amount of transmitted services (see figure 6).

B. Gain of Time by Coordinator-based Service Discovery

The native SDP requires successive connection establishments to all devices within the environment. Thus, the full service discovery time for n devices will be the sum of the inquiry time T_{Inq}, the connection time T_{Con}, and native SDP time T_{NSDP}.

$$T = T_{Inq} + \sum_{i=1}^{n-1} (T_{Con_i} + T_{NSDP_i})$$

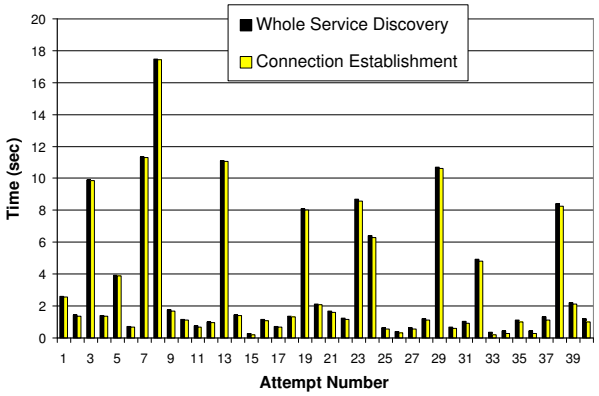

Fig. 6. Native SDP Time Delay

The coordinator model is based on the assumption, that the coordinator keeps the information about all available services in the environment, so that only one connection with the coordinator needs to be established. The service discovery time T_C in the coordinator based service discovery will be the sum of the inquiry time T_{Inq}, the connection time with a coordinator T_{Con_C}, and the coordinator SDP time T_{NSDP_C}.

$$T_C = T_{Inq} + T_{Con_C} + T_{NSDP_C}$$

Assume, the SDP processing time is negligibly small. Then, the time gain T_G by the coordinator based service discovery compared with the native SDP in percents will be:

$$T_G = \frac{T_{Con_C}}{\sum_{i=1}^{n-1} T_{Con_i}} * 100\%$$

C. Gain of Time by Peer-based Service Discovery

Using a broadcast-based second level discovery technology without broker (JSDP, SLP, SSDP) in a Bluetooth context results in a service discovery time T_{SD} including the inquiry time T_{Inq}, the time for Bluetooth connection establishment T_{Con}, the time Bluetooth SDP (T_{NSDP}) needs for determination whether the peer supports a certain second level service discovery technology, and the time required for the second level service discovery T_{SDP}. The maximum service discovery time $T_{SD_{max}}$ for a single service in a Bluetooth formation of n peers is:

$$T_{SD_{max}} = T_{Inq} + \sum_{i=1}^{n-1}(T_{Con_i} + T_{NSDP_i} + T_{SDP_i})$$

Applying the proposed approach leads to a drastic reduction of potential providers to be contacted. Being m the number of positive inquiry matches with $m < n$ the resulting maximum discovery time becomes:

$$T_{SD_{max}} = T_{Inq} + \sum_{i=1}^{m}(T_{Con_i} + T_{SDP_i})$$

The subtraction of both values results in the time gain T_G of the approach:

$$T_G = \sum_{i=1}^{n-1} T_{NSDP_i} + \sum_{j=1}^{n-m-1} (T_{Con_j} + T_{SDP_j})$$

Even for the case that there are only two Bluetooth devices, one being client, the other having the desired service ($n = 2, m = 1$) we have a time gain: $T_G = T_{NSDP}$.

V. Conclusion

In this paper, we analyzed specific features of the Bluetooth service discovery. We observed, that standard service discovery mechanisms are time and energy inefficient for ad hoc formations. Another disadvantage of the standard service discovery routine is the impossibility of using restricted nodes. Focusing on the time and energy consumption, we presented new discovery approaches based on the enhanced usage of the inquiry procedure. The presented results show that the proposed mechanisms are more time and energy efficient than the native SDP. The main drawback of our approach is the assumption, that all nodes are within range of each other. We plan to extend our approach in order to remove this limitation and in order to introduce security features.

References

[Blu01] Bluetooth SIG. Core. In *Specification of the Bluetooth System*, volume 1. Bluetooth SIG, 1.1 edition, February 2001. Available from World Wide Web: `http://www.bluetooth.com/pdf/Bluetooth_11_Specifications_Book.pdf`.

[BTA02] Bluetooth SIG. *Bluetooth Assigned Numbers*, June 2002. `http://www.bluetooth.org/assigned-numbers/baseband.htm`.

[GCL+99] Yaron Y. Goland, Ting Cai, Paul Leach, Ye Gu, and Shivaun Albright. Simple Service Discovery Protocol/1.0, October 1999. Available from World Wide Web: `http://www.upnp.org/download/draft_cai_ssdp_v1_03.txt`.

[GPVD99] E. Guttmann, C. Perkins, J. Veizades, and M. Day. Service Location Protocol, Version 2. IETF Internet Draft, RFC 2608, June 1999.

[LMS02] C. Law, A. Mehta, and K.-Y. Siu. A new Bluetooth Scatternet Formation Protocol. *ACM Mobile Networks and Applications Journa*, 2002.

[MCN+01] M. Marsan, C. F. Chiasserini, A. Nucci, G. Carello, and L. Giovanni. Optimizing the Topology of Bluetooth Wireless Personal Area Networks. *Infocom*, 2001.

[Pre01] Stephan Preuß. Java Enhanced Service Architecture [online]. 2001. Available from World Wide Web: `http://wwwiuk.informatik.uni-rostock.de/~spr/jesa/`.

[Pre02] Stephan Preuß. JESA Service Discovery Protocol. In E. Gregori, M. Conti, A. T. Campbell, G. Omidyar, and M. Zukerman, editors, *Proceedings of Networking 2002*, volume 2345 of *LNCS*, pages 1196–1201, Pisa, Italy, May 2002. Springer-Verlag.

[Sal02] The Salutation Consortium. *Salutation Technology*, 2002. `http://www.salutation.org/`.

[SBTL01] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire. Distributed Topology Construction of Bluetooth Personal Area Networks. *Infocom*, 2001.

[Sch95] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., 2 edition, October 1995.

[Sun99] Sun Microsystems, Inc. Jini Connection Technology [online]. 1999. Available from World Wide Web: `http://java.sun.com/products/jini/`.

[UPn99] UPnP Forum. Universal Plug and Play Connects Smart Devices [online]. 1999. Available from World Wide Web: `http://www.upnp.org/`.