TKN Telecommunication Networks Group

Technical University Berlin

Telecommunication Networks Group

Service Discovery in Wireless Sensor Networks

Christian Frank, Vlado Handziski, Holger Karl

chfrank@inf.ethz.ch, handzisk@tkn.tu-berlin.de, hkarl@ieee.org

Berlin, March 2004

TKN Technical Report TKN-04-006

TKN Technical Reports Series

Editor: Prof. Dr.-Ing. Adam Wolisz

Abstract

This document provides a preliminary analysis of the mechanisms for service and resource discovery in the Wireless Sensor Network (WSN) domain. These Service Discovery (SD) protocols form the cornerstone of the reconfiguration capability of the architecture and enable its effective management. We detail two proposals for incorporating this functionality in the Energy-efficient Sensor Network (EYES) framework.

The EYES project (IST-2001-34734) is a three years European research project on self-organizing and collaborative energy-efficient sensor networks. It will address the convergence of distributed information processing, wireless communications, and mobile computing.

The goal of the project is to develop the architecture and the technology which enables the creation of a new generation of sensors that can effectively network together so as to provide a flexible platform for the support of a large variety of mobile sensor network applications.

The document was submitted to the European Comission as EYES Deliverable 3.4: Service Discovery.

Contents

Chapter 1

Service discovery architectures

This chapter introduces the basic interaction patterns between the entities in a SD task and provides an overview of the relevant research. It is a slightly modified version of the material appearing in [14] and is organized as follows.

In Section 1.2 we give an overview of the traditional service discovery protocols, many of which have been designed primarily for wired local area networks within a single domain.

Systems providing discovery for wide-area networks including the discovery of Internet services are presented in Section 1.3. Section 1.4 deals with the recently very active research topic of distributed hash tables which are implemented in peer to peer systems.

Section 1.5 deals with service discovery protocols especially designed for mobile ad hoc networks and Section 1.6 describes work done to implement internet connectivity in ad hoc networks through extensions to the *Mobile IP* protocol.

Other related work ranging from *Sun's Project JXTA (JXTA)* architecture over architectures studying the smart forwarding of queries to research in pervasive computing systems will be presented in Section 1.7.

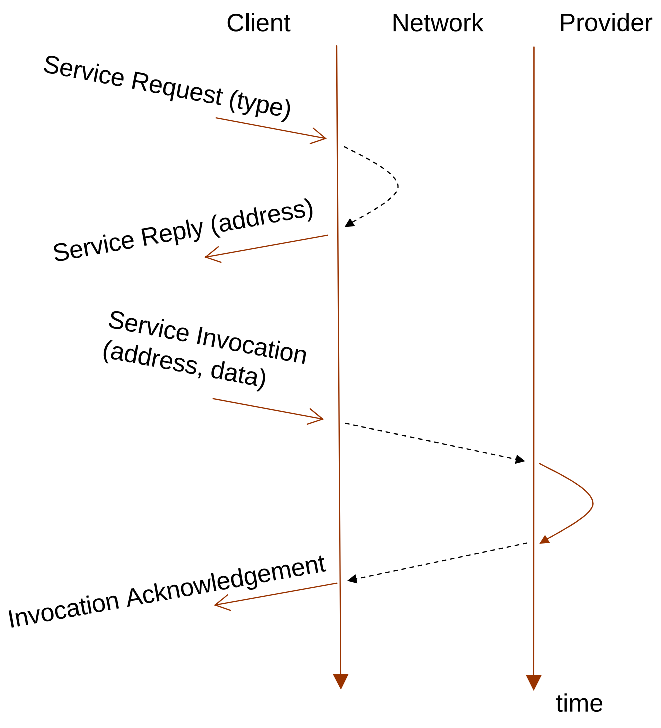
1.1 Entities and interaction patterns

This section contains a short visual introduction to the basic interaction pattern between requesters and providers of services during a typical SD task. The discovery process implicitly defines the necessary interfaces that have to be exposed to the application layer on both sides.

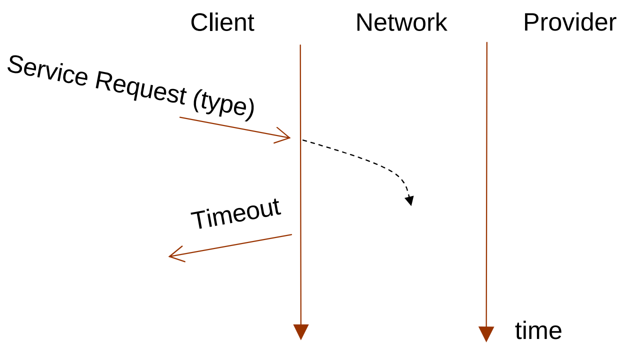
1.1.1 Requesting a service

The basic interface provided to a SD client is shown in Figure 1.1. The discovery process is simple: The client application issues a request to the node's lower layers and expects a service reply (within a given timeout).

In the case that a provider is found a reply is issued is shown in the Scenario 1.1(a). The reply includes the network address of the provider, to be used for subsequent communication between client and provider. After a successful discovery phase, additional communication between client and provider will be necessary in most cases. This information exchange is modeled by a *Service Invocation* and *Invocation Acknowledgment* handshake. Naturally, more or other data messages can be exchanged. End to end routing of data messages between client and provider will be needed.



(a) Replied Scenario



(b) Timed-out Scenario

Figure 1.1: Client Use Case

In the case that no provider is available, no reply will be issued by the network. In this case the client will have to wait for a timeout as shown in Figure 1.1(b).

1.1.2 Providing a service

The Provider node interface is also simple: Applications shall be able to register and deregister their services with the service discovery layers as seen in Figure 1.1.2.

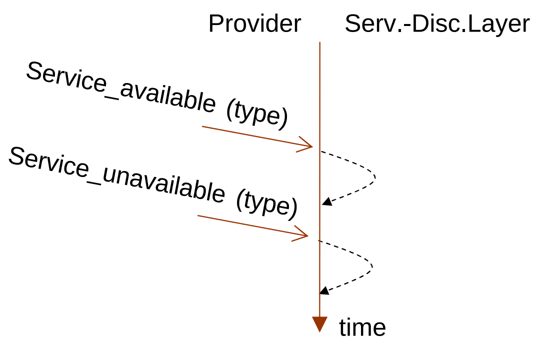


Figure 1.2: Provider Use Case

1.2 Classical Service Discovery Protocols

Discovery protocols provide features to spontaneously lookup services with a low communication overhead. Note that the connotation of *service* in the traditional context refers to a network service like a printer or *ssh* server; The traditional view of *service discovery* within a local area network is different from the usage scenarios in multi-hop wireless networks we discuss here. Overviews are given in [4, 23, 28], these include *SLP*, *Salutation*, *Jini*, *UPnP* and the service directory service *UDDI*. The protocols originally address one local administrative domain although extensions are sometimes discussed.

Service Lookup Protocol (SLP) [19] is a proposed standard for service discovery in TCP/IP networks. It provides a dynamic configuration mechanism for applications on local area networks. It automatically selects matching services specified by a client query. It may be implemented with or without the use of *directory agents*. If implemented without *directory agents*, local multicast to all service providers is used to discover services.

Sun's Project Jini (Jini) [11] is a *Java*-based framework for heterogeneous networks and software components. Jini benefits from Java's class loading capabilities to enable very flexible interface specifications for services. Service discovery is provided by the *Jini Lookup Service* which resides on a designated node to which other nodes may send advertisements or lookup messages. Cascading of multiple lookup services is possible to enable discovery outside the local domain. In addition to employing the central role of *Lookup Services* Jini imposes several constraints on the participants as it relies on *Java* and *Remote Method Invocation (RMI)*. Although *Jini* has often been termed an ad-hoc networking protocol because of its ability to spontaneously integrate new services and applications, it does not provide a suitable support for wireless sensor networks, because of its heavyweight RMI based protocols and its centralized structure.

Salutation [42, 43] is addressing the use of appliances and equipment in a highly connected network. Distinctive features are its transport independence and the support of code mobility: Devices store drivers, which enable their use, for a variety of operating systems in a repository, so clients may obtain the code to utilize the device. Clients and servers access their local Salutation Manager (SM) to discover or advertise services. Communication among SMs is independent of underlying transport protocols, although to communicate two SMs must use the same protocol. In a network without SMs, local broadcast is used.

Universal Plug-and-Play (UPnP) [51, 52] is an extension of conventional Plug and Play to encompass remote devices in IP networks. The protocol interaction is very similar to using SLP without directory agents. Discovery requests are sent to a designated local multicast address upon which replies are unicast from matching service providers.

None of the protocols described above is really suited for the mobile ad hoc networks environment. They are either based on aggregating information into central directories which would impose single points of failure or on maintaining a network wide multicast tree used for periodic service advertisements. The latter is a proactive measure which is inadequate in a bandwidth-scarce networks.

1.3 Wide-Area Discovery

Many of the protocols of the above Section 1.2 consider one single administrative domain in which all nodes participate. Some may be extended to multiple administrative domains, e.g. cascading Jini lookup services [11], to cover more than one domain. Similar extensions have been proposed for Service Lookup Protocol (SLP). A number of protocols address the specific functionality of multiple administrative domains of wide area networks to potentially cover the whole Internet.

Universal Description, Discovery and Integration Protocol (UDDI) is a recent industry initiative to provide a global directory for internet web services [49]. The directory may be replicated among several servers, so-called *node operators*. Changes should propagate to all servers within a set amount of time. The architecture is intended for relatively static and highly available service providers like e-commerce sites.

Other systems providing wide-area discovery are *Twine* [2], *Terradir* [45] and the *Secure Discovery Service* [10] which will be discussed in Sections 1.4, 1.7.2 and 1.7.3 respectively.

1.4 Peer-to-Peer Distributed Lookup Systems

Several recent systems provide a Distributed Lookup Service (DLS) often referred to as Distributed Hash Tables (DHTs). Such systems are CAN [39], Chord [46], Tapestry [56], and Pastry [41] and are targeted at large-scale peer-to-peer applications in the Internet environment.

1.4.1 Common Characteristics

The essential capability of all of those systems is to map a given key onto a network node and to route a message to this node in an efficient way. The resulting node could for example store a value for that key, making the system a distributed hash table. Several important issues are addressed: Decentralized operation, scalability, fault tolerance, and load balancing. *Insert* and *Lookup* operations and state-per-node information are efficient. More information on these data-centric architectures can be found in [20, Sect. 1.2.1].

1.4.2 Employing DLS systems for ad hoc service discovery

Each Distributed Lookup Service (DLS) system is based on maintaining an overlay space of node keys which is the basis of an application layer overlay network. Each node is assigned a *(hash-)*key when joining the system based on the node's identifier. From this point on a lookup message for a given key will be routed to a node whose key is *closest* to the sought key in a logarithmic number of overlay network hops. CAN is an exception here trading in this bound for a constant routing table size. *Close* meaning:

- numerically close in *Tapestry*, *Pastry*, and *Chord*, i.e. in a one dimensional ring space
- close in an d-dimensional coordinate space onto which each key is mapped in *CAN*

One may consider storing pairs of (*service-type*, *provider-address*) in a DLS system, as done, e.g., in [21]. The overlay network concept of those systems is disadvantageous for our purpose: A node joining the network will have to exchange state information with its closest neighbor(s) in the overlay space. As one overlay network hop may mean an arbitrary number of real topology hops, the closest overlay neighbor might be located arbitrarily far-away even at the opposite side of the wireless network, introducing high latency. Heuristics to improve performance based on real topology are described in [8] for the *Pastry* system or in [55]. Yet the proactive communication for maintaining a consistent overlay structure (especially in the face of frequent node joins and leaves) is unnecessary overhead if a service provider could be found within three or four hops using simple expanding-ring flooding of the request.

Wide-area service discovery based on a DLS system is provided in [2]: The so called *Twine* architecture additionally provides semantic resolution of partial queries for a given resource, e.g. one may query for a ”multimedia device” or a ”digital camera” or a certain model. It employs a set of so called *resolvers* which are organized in a DLS network. Resources are described in a semantic tree fashion with so called *Attribute-Value-Tree (AVTree)* defining a *strand* as a partial path from the root to any node of the AVTree. All given possible *strands* of an *AVTree* are mapped onto the full resource description. A client may now query any resolver using a full or partial service description, and queries are load distributed among the participating *resolvers*.

1.5 Ad Hoc Discovery Protocols

TANG et al. [48] and CHENG [9] both describe a multicast based service discovery approach for ad hoc networks. The latter adds extensions to the On-Demand Multicast Routing Protocol (ODMRP) [54] to provide service discovery, i.e. integrates routing and discovery like in our proposals. Both let services/resources be advertised to a known multicast address which is joined by nodes interested in advertisements. In this, the protocol operations are similar to SLP without directory agents.

The Service Discovery Protocol (SDP) used in Bluetooth devices [5] is a very simple protocol enabling applications on a device to request services provided by a Bluetooth peer. The peer will reply whether it provides the requested service. SDP was not intended for larger networks (as one needs to subsequently address every peer) and additional constraints are imposed by the *Picconet* concept in Bluetooth. AVANCHA et al. [1] discuss providing more sophisticated matching mechanisms in presence of resource-rich devices like notebooks.

LAWRENCE [27] describes work currently under way to adapt an existing multi-agent platform (JADE) to the ad hoc network environment. The agent platform runs ontop of existing ad hoc routing and basic service discovery technologies like Bluetooth SDP or JXTA. More sophisticated application level discovery is provided by the agents themselves but does not inherently take location into account.

NAHATA et al. [30] present a hybrid resource discovery approach in which proactive routing and announcements are performed within a small neighborhood area. Discovery outside the neighborhood is initiated reactively through remote *contact* nodes which then reply the query from their own neighborhood or forward it to their contacts. The work is not adaptive to the request pattern and induces a significant proactive routing overhead within the neighborhood areas in the case that network traffic is very infrequent.

WU and ZITTERBART describe a discovery and advertisement scheme for services in mobile adhoc networks in [53]. Discovery (client initiated) and advertisements (provider initiated) both happen in a network wide flooding fashion. By snooping into data packets information on whether a route to the provider is still available is maintained. Discovery floods may be avoided by *service polls* piggybacked into the IP option field of outbound data messages.

Service discovery in ad-hoc networks using *anycasting* [33] lets providers of a given service type all advertise routes to a *virtual node* representing the service type. Routing will then forward routing queries for the *virtual node* correctly to *one* of the service providers. Anycast extensions are described for link-state, distance-vector and link-reversal routing types are described. Link-state routing protocols need to disseminate the whole topology to all network nodes and are not applicable to the scenarios that we are examining. Anycasting for distance-vector routing is discussed for basic approaches like the asynchronous Bellman-Ford algorithm [3]. But for Destination-Sequenced Distance-Vector Routing (DSDV) or Ad hoc On-Demand Distance Vector Routing (AODV) [35] the article explicitly states that further work is needed for coordinating sequence numbers of *virtual nodes*.

In [25] a backbone (clustering) based approach is chosen. In this context arguments are given that using directories in designated nodes is in fact feasible and a good candidate for service discovery in MANETs. Two protocols are given, one for the creation and maintenance of the virtual backbone and a second one for service resolution. The described *Distributed Service Discovery Protocol (DSDP)* is compared to a multicast approach based on ODMRP [54], and to *anycasting* variants of AODV and Dynamic Source Routing (DSR) [22]).

1.6 Internet Connectivity for MANETs

Approaching the discovery problem from a different viewpoint are protocols providing internet connectivity to ad hoc networks and thus integrating a routing component with the Mobile IP protocol [40]. Mobile IP provides mobility to mobile users with fixed IP addresses in the face of hierarchical IP routing: A mobile user arriving at a remote local area network cannot use its own IP address (because of the invalid subnet mask), but must register with a so called *foreign agent* in its current network. The *foreign agent* then establishes a tunnel to the mobile host's home site and forwards all packets to and from the mobile host via the tunnel. The work in this section pertains to enable Mobile

IP functionality in ad-hoc networks that surround *foreign agents* and thus extend the functionality of the administrated network core.

The resource to be discovered in an ad-hoc network in this case are the *foreign agents* of Mobile IP which are hooked onto the wired Internet. Prominent work in this area is done by SUN et al. [47]. The focus is on integrating AODV and Mobile IP protocols. Foreign agent advertisements are flooded periodically by intermediate nodes. The analysis focuses on variating mobility and the announcement interval with one or two foreign agents.

RATANCHANDANI and KRAVETS [38] describe a hybrid approach between reactive discovery and periodical advertisements of foreign agents. The reactive discovery is integrated into AODV route discovery. Agent advertisements are limited in scope and also establish routes.

1.7 Other Related Work

1.7.1 JXTA Peer-to-Peer Framework

Sun's Project JXTA (JXTA) [17] is a set of protocols for peer to peer systems also including service discovery. Protocol requests and responses are Extensible Markup Language (XML) messages and thus independent of programming language and network transport.[13] specifies JXTA request and reply messages but the way in which queries for a service are forwarded is left open. A proposal of how to forward requests in a JXTA based internet search engine is given in [6]. Peers are distinguished by being *ordinary* or *hub peers* (sometimes also referred to as *rendezvous peers*. Ordinary peers would only advertise the services they offer while *hub peers* would act as proxies for service advertisements of other peers.

1.7.2 Query Routing in Resource Discovery

Resource discovery has also been a very active topic in *Multi-Agent Systems (MAS)* research. The term *Query Routing* stands for an informed distributed search used for resource discovery [16]. When a client performs a search for a resource, it queries a single server which then forwards the query towards the servers which hold the desired resource. The forwarding process employs summarized knowledge about the capabilities of other servers of the network, called *forward knowledge*.

GIBBINS and HALL analyze communication and time complexities of query routing in a number of network topology types. This includes hierarchical and complete networks, but also topologies as created by distance vector and link state routing. The article delineates a general model of query routing which applies to all network topologies upon which it bases the complexity analysis. Except for ordered networks whose topologies are either hierarchical or clustered with designated nodes representing the cluster, the routing table size is O($totalnodes$) as propagation of *forward knowledge* for each node is needed.

PRINKEY [36] gives a similar approach using bitmasks at each node as forward knowledge for query routing in tree topologies. A bit is set when relevant information on a certain indexed item is available in the subbranch of the tree induced by the node. The bitmask of a parent node is an OR operation of all child nodes thus the bitmasks do not grow. A query is routed to a branch of the tree if all bits of the query are set in this branch. However, propagating such bitmasks along the hierarchy is very expensive if the index space is large. [26] and *Terradir* [45] describe a network which is based on hierarchical classification of content and query forwarding according to the hierarchy. However, the first does not address *forward knowledge* dissemination, the later how nodes are made responsible for a part of the hierarchy.

MOREAU describe an application level algorithm providing for two features: A distributed directory service and a message router that uses the directory for message delivery. The algorithm is designed for mobile agents moving across hosts which allow agents to migrate and execute called *sites* and is described on the application level.

1.7.3 Pervasive Computing

A large base of research work is undertaken in the domain of *Smart Spaces*, *Smart Homes* and *Pervasive Computing*. One of the definitions of *Pervasive Computing* is: ”Numerous, casually accessible, often invisible computing devices [...] connected to an increasingly ubiquitous network infrastructure composed of a wired core and wireless edges” [31]. As the definition states, the systems rely on the wired core of the network providing service discovery through a hierarchical structure of servers..

The *Secure Service Discovery Service (SDS)* [10] is a part of the Berkeley *Ninja* Project [18]. SDS is based on a hierarchical structure of domains with a server architecture which propagates service descriptions along the hierarchy from room-level to internet level. In Ninja services run in a well-engineered cluster of workstations on the Internet which is contrarily different to the ad hoc node approach taken here.

Other projects in this area are *Centaurus* and *Centaurus2* [50]. University of Washington’s *Portolano* [12] project describes a ”data centric approach” to disseminate information in the wired core. KICIMAN and FOX [24] deal with application level interactions and data conversion between ad hoc network participants in *Interactive Workspaces*.

Chapter 2

Service discovery in WSNs

This chapter discusses the service discovery functionality in the specific context of WSNs. Here we analyze the most prevailing usage scenarios for service/resource discovery and their distinct requirements.

2.1 General properties

The WSNs have several characteristics that set them apart from the "traditional" wired and wireless networks that are the usual targets of the current Service Discovery (SD) protocols and architectures discussed in the previous chapter. These differences make the large body of research in "classical" SD protocols not directly applicable to the problem at hand.

One of the basic properties of the WSNs that makes them so different is their *application specific* nature. This is to say that the nodes in a WSN cooperate together in order to jointly solve one or more clearly defined tasks or applications. This is in a a contrast to the traditional networking architectures (wireless ad hoc networks included) that provide a general communication infrastructure that is more or less *application agnostic*.

The application specific nature of the WSN make the obtained *data* the primary object of interest. The identity of the node that produced the data is either completely irrelevant, or only marginally relevant as a proxy value for some other semantic property like location, hop distance etc. This has a significant impact on the SD functionality that traditionally has to create a binding between the requested service/resource description and the Identification (ID) of the node(s) in the network that provide(s) such a service. The semantic routing protocols that operate without globally unique IDs make such a direct binding unnecessary. Here the main task of the SD component would be to support the reconfiguration capability of the network and to build repositories of the available semantic attributes and other concepts. Internally, these repositories will facilitate the cooperation between the nodes in the network. On the edge, they will support the interaction with the human operator(s) and the interfacing with the external networks.

Nevertheless, there are some particular tasks in a WSN that require an ID-centric communication. Network management and actuator control are such examples. In these cases, a SD exchange is necessary to match the providers and the consumers of the service/resource. Only after this step is completed can the individual entities start with the real interaction that is mostly *client/server* and *request/reply* in nature. As this is close to the standard SD functionality, we believe that a modified

SD protocol proposal from the Mobile Ad-Hoc Network (MANET) area might be a adequate solution here.

Regardless of the data-centric or ID-centric nature, the SD implementation has to be streamlined and should contribute to the overall energy-efficiency of the protocol stack. The number of generated messages is the main factor determining the scalability of the protocol. Thus, every attempt should be made to reduce it. The tight integration between the routing and the SD functionality has been suggested as a promising approach to this end [37]. The sharing of the available state information and the lowered control overhead (due to the piggybacking) can significantly increase the scalability of the implementation making it appropriate for the resource constrained WSN setting.

2.2 Usage scenarios

In this section we motivate several usage scenarios for the SD functionality in the light of the specific conditions outlined above.

2.2.1 Inside the network

The main task for the SD component inside the network is to provide support for the instantiation and reconfiguration of the nodes. When a new node is introduced in the network it has to learn about the capabilities and services of the other nodes. This knowledge enables optimal division of work and maximizes the efficiency of the cooperative problem solving. Because of the dynamic nature of WSNs (form introduction of new nodes, depletion of the existing ones, mobility, communication errors, etc.) this resource discovery has to be periodically repeated during the whole lifetime of the network.

As discussed in the previous section, these activities entail both address-centric and data-centric parts. In the first group are the discovery mechanisms that tend to establish a link between the requested service/resource and the unique identity of the node that offers this service. One typical example for this type is the control of the acting elements in the network, i.e. actuator control. This communication is mostly done in a *request/reply* fashion, because the return information about the successfulness of the request can have significant impact on the future actions. This creates tight identity and temporal coupling between the entities and is usually addressed with and address-centric discovery scheme. The same conditions occur in the network management task, when the operator needs to target his commands to specific nodes: turning particular on or off, selective reprogramming of the nodes, gateway selection, etc.

The second group of discovery tasks are the ones that do not require that a specific binding is created between the unique identity of the providers and the services they offer. If the network supports data-centric routing of messages, this binding is not necessary as the entities can communicate between each-other even without knowing the unique ID of the other side. For example, on instantiation, the temperature sensing node might issue a resource discovery request for other temperature nodes in his vicinity so that he can determine its sleeping schedule. After overhearing the replies in the neighborhood the node computes the optimal schedule and then rebroadcasts its decision back.

We can see that the underlying data-centric routing implicitly performs the same function as the classical service discovery task. This means that the focus of the SD capability is shifted towards providing some additional services on top of this basic functionality. One such service might be to optimize the discovery task in the case of frequent requests for fixed or slowly changing services by caching the information on the data-centric routing level. Another one is the creation and maintenance of a metadata repository of the attributes used in the network that will guarantee a consistent use of the data-centric naming both inside and outside the network. We will discuss this specific scenario in more detail when we describe our Content-Based Publish/Subscribe (CBPS) based solution in Section 3.2.

2.2.2 Bridging

We believe that this will be a very relevant use-case once WSNs become more popular and widely deployed. In such circumstances, it is easy to envision a situation when one posseses or uses several WSNs from different vendors. For example, one might have a power metering/controlling WSN form the power utility, and a temperature/humidity monitoring network from another vendor.

When used in a bridging scenario, the SD component would have the additional responsibility to mediate between the service/resource discovery exchanges between the separate networks so as to enable joint use of their capabilities. On the above example, this might be the task of turning on the heating element (controlled by the power utility WSN) when the temperature nodes detect a temperature below the predefined threshold.

On the data-centric plane, this mainly maps to the problem of translating between the attribute and concept repositories in the two networks. The complexity of this operation will to a large extent depend on the existence or nonexistence of a standardized attribute and service specification templates. In the worst case, this can be done explicitly, i.e. the owner/operator can explicitly define the equivalence relations between the attributes in the two domains.

On the address-centric plane, this mainly maps to the problems of looking up the bridging node(s), establishing common network management, and supporting cross-domain actuator control.

2.2.3 Gatewaying

The SD functionality also plays an important role in the interfacing between the WSN and outside entities. Regardless whether this *gatewaying* is towards the operator or towards external non-WSN networks, the SD component must provide a coherent view to the services/resources available in the network and expose them in the most suitable way for the external user. We provide a short motivating scenario for this use-case:

For example, we have a new user entering a building that has WSN based Heating, Ventilation, and Air Conditioning (HVAC) monitoring and control. The network configuration is unknown to the user. He needs information on the available sensors, actuators, their operational state, etc. In other words, he needs a global or partial view of the network and its capabilities so that he can issue meaningful service invocation commands. The SD component has a crucial role in providing this easy access to the available services of the WSN.

First, the low-level communication between the user's Personal Digital Assistant (PDA) and the WSN has to be resolved, for example by equipping it with a suitable networking interface and the low-level protocols used on the WSN. Then, a list of the available services/resources in the network has to be generated, as discussed in the previous use-cases.

However, because of efficiency reasons, inside the WSN, the SD will probably operate with customized representations of the services. In general these representations are incompatible with the ones in the standardized, Internet Protocol (IP)-based, protocols that run on the user's PDA or on

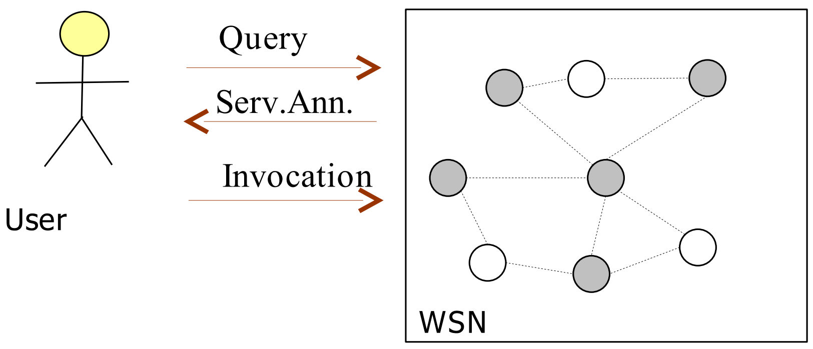


Figure 2.1: Interaction between an external user and a WSN

the external network (like SLP, JXTA, UPnP, etc.). This makes the translation between the protocols and the service representations a major issue in this use-case. It is important to notice that this is not just a problem of mechanical translation between different “string” representations of the services. Sometimes this entails “transformation” between different interaction patterns. For example, if the WSN operates in a data-centric way using CBPS, the SD component at the gateway has to transform between the *request/reply* nature of the external SD commands, and the *publish/subscribe* interaction inside the network.

Chapter 3

Service discovery in the EYES framework

Having in mind the specific requirements for the SD capability discussed in the previous chapter, in this one we sketch the details of two viable solutions to the problem. Both solutions are based on a tight integration between the routing and SD components. In this way, the control overhead is minimized and the SD functionality can be further enhanced by the information available on the routing level.

The first proposal is aimed at the "classical" SD functions where the exact identity of the service provider is required. In other words, it provides the basis for a future one-to-one, client/server based exchange between the service provider and service requester. It can be used for network management, actuator control or similar ID-centric tasks.

The second approach is an extension of the data-centric communication model presented in[20]. It will be used for building a metadata repository that maintains a list of the capabilities of the network, thus enabling consistent use of the concepts (in this case CBPS attributes and operations) inside the network and also on the network edge when interfacing with external networks.

3.1 Integrated with an ad hoc routing protocol

The main idea of [34] is to attach a service discovery header to control packets used by routing. However, it appears reasonable to extend it further with functionality that reflects the dynamic nature of *service availability*, which is independent from the question of whether a service provider is *reachable* from a pure routing point of view – the main problem is that cached information on availability might be outdated or staled inside the network while routing information is still accurate.

Hence, we choose [34] as our starting point and extend it with some extensions that reflect this service dynamism. The material in this section is an abbreviated version of the content appearing in [15].

3.1.1 Implementation of the basic functionality

In this section we describe the implementation of the basic protocol.

Service request and reply packets

PERKINS and KOODLI assume that the routing component follows a request/reply cycle: A route request packet containing the sought destination is broadcast and is replied by a node knowing a route to the destination. This sequence is part of most reactive routing protocols.

A service request packet (SREQ) is shown in Figure 3.1. It comprises a route request packet (RREQ) plus an additional header describing the sought service, the so-called *Service Request Extension* or SRE. The SRE contains information for identifying a given service type (*service selector*). In [34] this information may be either a URL or a port number; in our context it is modeled by an integer referring to a predefined service type. If the search should be enhanced with semantic attributes (which is not implemented), these attributes would also be located in the SRE.

Similarly, a service reply packet (SREP) will be a route reply packet including additional information on the service found, accordingly termed a *Service Reply Extension* or SRpE. It includes the service lifetime, which may be different for various providers.

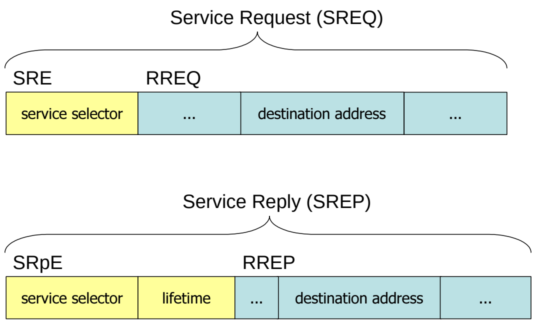


Figure 3.1: An additional header is attached in front of routing control messages

The depicted *destination address* field in RREQ and RREP messages refers to the sought destination for which route discovery is performed. Service discovery will use it to store the provider address, see below.

Service and route resolution

At the time a client issues a request, it creates a route request packet and adds an SRE with the sought service type. At this time the destination address of the RREQ packet is unknown and will be set to zero (other fields may need to be set compatibly, e.g. if using AODV routing the destination sequence number must be set to *unknown*). The destination address of the RREQ will later contain a matching provider address.

The SREQ will propagate through the network like an ordinary RREQ. A node which receives an SREQ will fill-in the missing destination address if it knows a matching provider or if it is a provider itself. Each node will store so-called *service bindings*, a binding associating a service type to addresses of provider nodes. The provider node will update its bindings locally. Other nodes will cache service information from SREPs they forward. Note that multiple providers may be registered for a given *service type*. The table additionally stores a cache lifetime up to which the service information is valid.

Let us now look at the request process. As mentioned, the client node broadcasts a service discovery request with an empty destination address. The handling of an incoming SREQ is shown in Figure 3.2.

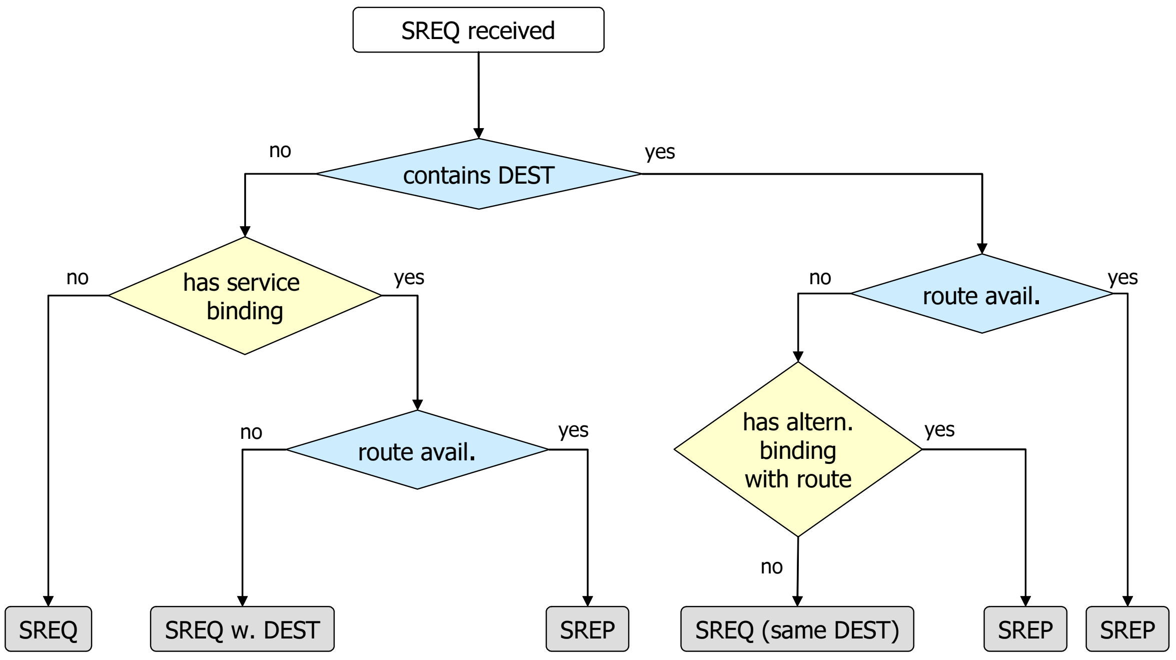


Figure 3.2: Handling an incoming SREQ

The node at first checks whether the packet contains a destination (the destination field is non-zero). If the packet *does not* contain a destination, there are two possibilities:

1. If the node does not have a service binding for the given service type it will rebroadcast the SREQ unchanged.
2. If the node has a service binding including a route, it will send an SREP packet back towards the client. If it has a service binding but no route, it will just fill-in the destination address of the binding with the least number of hops and rebroadcast an SREQ with an updated destination field.

If the packet *does* contain a destination there are also two cases:

1. If the node knows a route to the destination, it will send a RREP packet with a SRpE header containing the service type and lifetime in its service binding table.
2. If the node does not know a route to the destination, it will look up whether it has alternative service bindings (including a route to the provider address) for the given service type. If so, it sends an SREP including service time and the lifetime stored in its table, if not it just rebroadcasts the SREQ.

An SREP will be unicast back along the spanning tree established by the SREQ flood. The intermediate node will add or update its own service binding if the information in the message is more recent than its own. In the opposite case, the lifetime of the message is updated to the node's own service binding entry. The RREP part of the message will always be processed by the corresponding routing component and thus routes will be constructed via the message as well. If the SREP is directed at the local node, a service reply is indicated to the requesting application and the message is not forwarded.

3.1.2 Scalability enhancements

Proactive enhancements

The route and service search by PERKINS and KOODLI and its integration with AODV routing contribute the valuable concept of simultaneously discovering routes and services. It is not clear, however, how these mechanisms behave especially under heavy load on the network and under frequently changing provider availability.

We evaluated possible proactive enhancements that are able to decouple protocol effort from the number of requests in areas of high traffic. Naturally, the proposed caching of service bindings is able to improve performance in high load scenarios (Figure 3.3). It is unclear, however, what a suitable caching lifetime would be in such scenarios. Although caching may reduce protocol effort, we expect that it yields a significant amount of incorrect replies, especially when providers become unavailable and in presence of many requests. Also, caching may compromise our goal of hop-wise optimality, when a choice among several providers is made.

Generally, proactivity is activated by a provider becoming either newly available or unavailable. In both cases, the time during which the network stores inconsistent information should be as short as possible. In the reactive case (with caching), a node learns about a new provider only after it has overheard or processed a request/reply pair to this node. This happens soon at high request frequencies. Periodic announcements, on the other hand, limit the time until the new provider is known to the announcement period. To match the reactive protocol's performance in high-load cases, frequent announcements would be necessary, which are just wasted overhead in other cases. Additionally, the period of inconsistency for providers that have become unavailable remains. Hence, the benefits of periodic announcements would depend on a careful tuning of announcement periods; therefore, periodic announcements are questionable and will not be considered further.

Negative announcements

When a provider becomes unavailable, inconsistencies in the network have a bigger impact and should be corrected sooner, lest requests are incorrectly replied. Here, the disadvantages of caching are clear, both for reactive and for periodically announcing systems. To limit incorrect answers, short cache lifetimes would be necessary, nullifying its effectiveness. Therefore, explicitly removing cached entries by negative announcements would allow to maintain caching benefits without, hopefully, imposing too large an administrative burden (Figure 3.1.2). Such negative announcements are easy to implement for providers *actively* switching state. This active transition to unavailable status, triggered by the service application is a crucial aspect, in which our scenario is different from scenarios used in most related work.

In the case that the provider becomes inadvertently disconnected through route changes, the case is different: The first request will result in a route error, yet the the provider address is still valid infor-

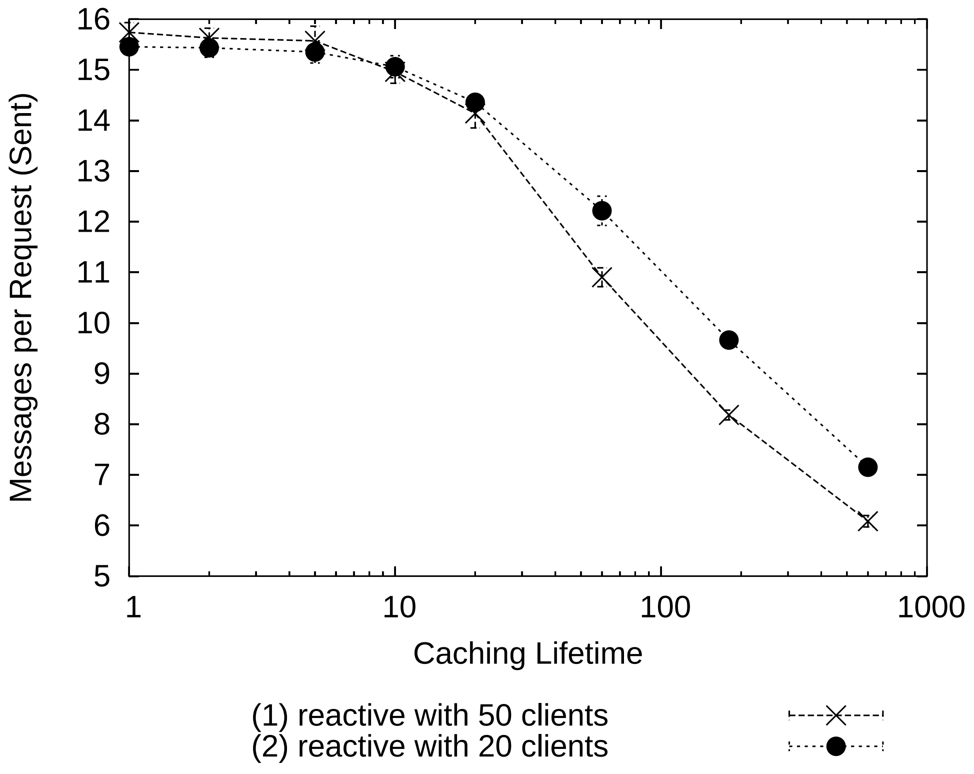


Figure 3.3: Benefits from increasing the cache lifetime for a long-lived service

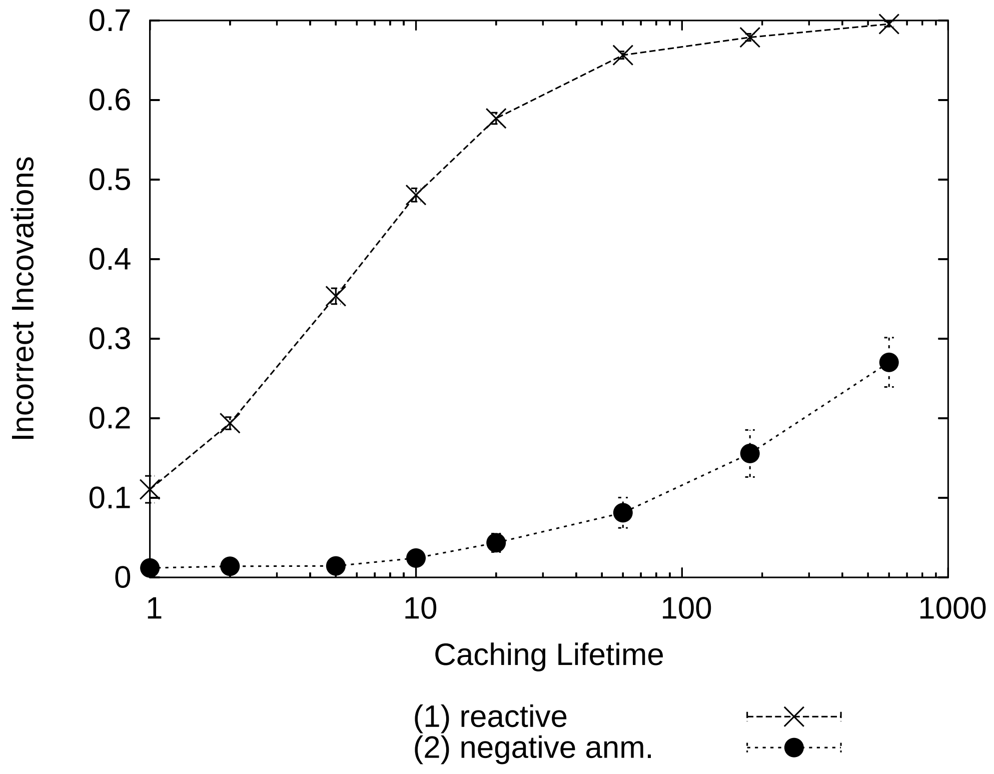


Figure 3.4: Incorrectly invoked request as a result of caching for a dynamic service. Negative announcements significantly improve the consistency.

mation and useful information: If no other service provider is known at this node, a new route to the provider will have to be discovered by the routing component. The route error becomes relevant only when choosing among several providers: As no hop count for the route to the unavailable provider other will be preferred choices.

Once inconsistencies are dealt with, the possibility to maintain longer service lifetimes is reintroduced, as the entry will be erased when the provider actively changes state and routes will be kept up to date by the routing protocol.

Similarly, positive announcements upon provider state changes may also be beneficial, but preliminary evaluations showed that they increase inconsistency for providers which frequently change state: The effort required to delete information spread via positive announcements plus the additional effort to send the announcement in the first place is larger than their benefit. Due to space considerations, we largely omit them from further discussion.

In conclusion, the main mechanisms to be considered are: reactive with or without caching and caching with negative announcements.

3.2 Based on the semantic addressing architecture

The rapid growth of the distributed systems in the last decade has prompted the development of more flexible communication schemes that closely mirror the emerging dynamic and decoupled nature of the applications. The *publish/subscribe* is one such scheme that supports a form of loose interaction between the entities in a distributed system [32]. It is event-based in nature, that is very different from the *request/reply* behavior of the "classical" *point-to-point* and *synchronized* protocols.

Instead of specifying the identities of the communicating parties, in the publish/subscribe model, the parties specify the nature of the data/events that they are prepared to provide or consume. The abstract *brokering* service that sits between them coordinates the interaction, making sure that the matching data is delivered to the interested parties.

The receivers declare their interest in certain types of data by subscribing. This is done by sending a *subscribe* message that codifies the nature of the requested data. From this point on, all the produced data that is matching the subscription is delivered to the subscriber. This continues until the subscriber declares that he is no longer interested by unsubscribing with an *unsubscribe* message that codifies the nature of the data he does not want to receive anymore. The providers disseminate the data by *publishing* it in the form of notification messages. In addition, they can issue *advertisement* messages in which they declare the nature of their future notifications. These messages are then used to improve the brokering performance and to inform the subscribers about the availability of new information types.

One may argue on the need of a separate *service discovery* component in a network with a CBPS middleware. From a functionality point of view, the naming part (Section 3.2.2) provides support for almost all of the standard service discovery tasks. Nevertheless, spreading new subscriptions to all nodes each time one wants to learn about the available services may not be the optimal solution. If these requests are going to be frequent enough, it makes sense to collect the necessary service description information for easy future access.

A natural and energy-efficient way of approaching the problem is to make use of information that is already present in the network. As stated, the *advertisements* are important tool for routing optimization in the CBPS systems. But the same knowledge can also be exploited to build a distributed repository of node capabilities that forms the basis for quite energy-efficient *service discovery* functionality. This represents the core idea of our approach.

To increase clarity we will use the conventions of the Service Lookup Protocol (SLP), as specified in [19] when describing the matching between the SD and CBPS entities and their messages.

3.2.1 Entities and interaction patterns

Just like in the "classical" case, there the two basic types of nodes:

Service Providers (SPs) that provide services/resources to the other nodes.

Lookup Clients (LCs) that request the use of some service/resource offered by the SP nodes.

The underlying CBPS middleware already provides the basic discovery functionality between these types of entities. In order to provide additional functionality, we introduce another type of nodes:

Lookup Servers (LSs) that increase the performance of the SD using soft caching. The cached data also serves as a metadata repository for the available CBPS attributes and concepts.

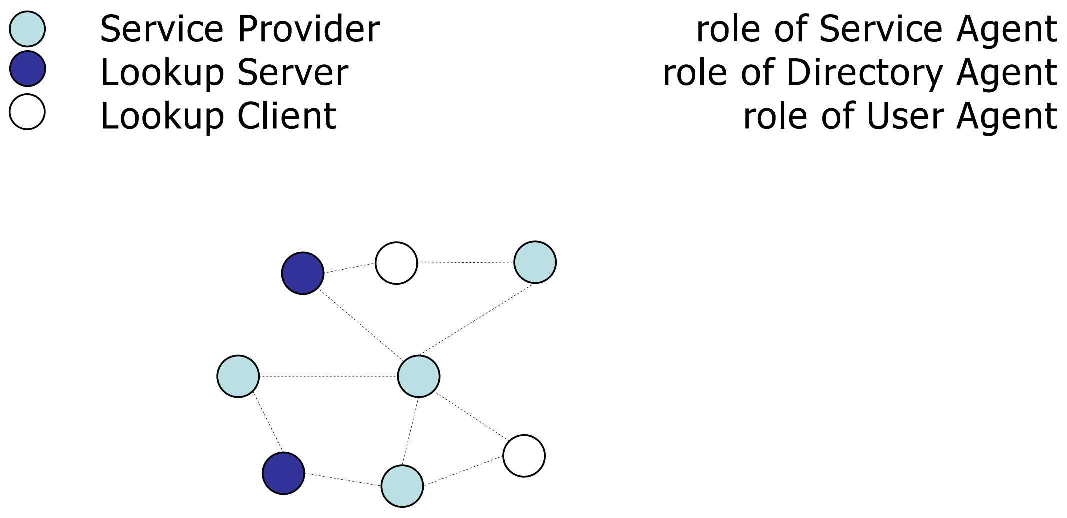


Figure 3.5: Entities in SD based on CBPS. Relation with the SLP conventions

The communication between the above entities is performed exclusively using the primitives of the CBPS layer. The design is very intuitive:

- The Service Providers (SPs) *publish* service announcement messages.
- The Lookup Servers (LSs) *subscribe* to service announcement messages and cache them.
- The Lookup Servers (LSs) *publish* service lookup information messages (summarized service announcement messages).

- The Lookup Clients (LCs) *subscribe* to service lookup information messages.
- The Lookup Clients (LCs) *publish* service invocation messages
- The Service Providers (SPs) *subscribe* to service invocation messages

The LS nodes provide decoupling between the providers and the requesters which would have communicated directly otherwise (Figure 3.6). This decoupling provides a relief for the Service Provider (SP) batteries under heavy load and long-term service configurations. It also resolves some of the problems arising when dealing with powered-off and standby SP nodes. By using the capabilities of the CBPS layer, the service announcement traffic can be aggregated thus increasing the energy-efficiency. Most importantly, the setup enables a gracious fall-back to the basic discovery services of the CBPS middleware in the case of LS failure.

The consistency of the cached information in the face of the network architecture and service dynamics is going to be maintained using the mechanisms described in detail in Section 3.1.2

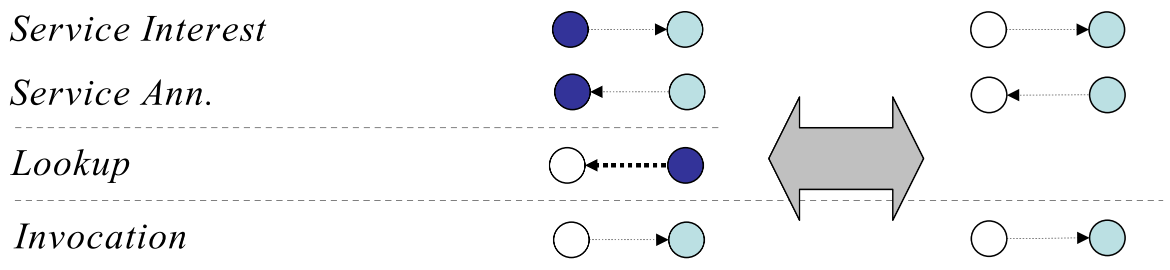


Figure 3.6: Changes in the communication sequence due to the LS nodes

Before we explain our proposal more closely, we first give a short description of the CBPS naming scheme used in EYES. Further details about the data-centric architecture can be found in the deliverable [20].

3.2.2 Content-based naming

One of the most widely used content-based naming scheme is the so called *attribute* naming system [7]. In this type of naming, the elementary subscriptions are defined as *conjunction* of simple attribute constraints in the form of (*attribute*, *operator*, *value*) tuples. The event notifications, on the other hand, are conjunction of (*attribute*, *value*) tuples.

The various parts of the attribute filter have the following meaning:

Attribute This part of the *content-based address* specifies the property of the notification that will be subjected to inspection during the filtering process. The usage of a preselected set of attributes does limit the expressiveness to a degree, but the complexity of the matching and forwarding tasks is largely simplified.

The attributes are usually typed. Most of the implementations line-up the attribute types to those available in the used programming language. In addition to the basic number and boolean types, almost all of the implementations support string attributes and provide means for extending in the form of User Defined Types (UDTs).

Value The value is selected from the allowed range for the given attribute type and has to be aligned with the operation part of the name, i.e it has to belong to its valid domain. Some of the operations also allow the use of values with *special* meaning like “`ANY`”, “`ALL`”, etc.

Operator The last part of the name tuple contains the binary predicate operator used for the filtering. Primarily, it is one of the common equality and ordering relations (EQ, GT, GE, LT, LE). For the more complex attributes, type-specific operators can be defined (e.g. *substring*, *prefix*, *suffix* operators for strings).

3.2.3 Message types

The similarity between a CBPS interest and a SD service lookup is obvious. Both types of messages declare the interest of the issuer in a specified property of the target (in this case a service that it provides). Because of this similarity the lookup messages can be very naturally expressed as CBPS subscriptions.

For example, using the CBPS naming described above, the LS nodes can issue an interest message for the available services in their vicinity:

```
CLASS IS interest
TYPE IS service search
SERVICE_KEY EQ_ANY
LATITUDE_KEY GE own_lat-1
LATITUDE_KEY LE own_lat+1
LONGITUDE_KEY GE own_long-0.5
LONGITUDE_KEY LE own_long+0.5
```

The issued interest propagates in the network and forms gradient state that will be used by the matching service data messages on their way from the SP nodes back to the LS.

Upon receiving such an interest, the SP nodes respond with a service announcement message that describes the offered service and its parameters (together with an invocation Application Programming Interface (API) when appropriate). This response is in the form of a CBPS advertisement when the service setup is long-lasting and fixed, or a normal CBPS notification for the more dynamic cases.

For example, a light dimmer actuator that is in the vicinity of the above LS can issue such an announcement:

```
CLASS IS advertisement
TYPE IS "service announcement"
SERVICE_KEY IS "light dimmer"
LATITUDE_KEY IS own_lat
LONGITUDE_KEY IS own_long
i_LIGHT_LEVEL IS (0,20,40,60,80,100)
```

These announcements are collected and cached by the LS nodes. After this they are ready to answer the service lookup queries from the LC nodes.

For example a node that looks for a light dimmer in its vicinity:

```
CLASS IS interest
TYPE IS "service lookup"
LOOKUP_ID IS <RAND_NR>
SERVICE_KEY EQ "light dimmer"
LATITUDE_KEY GE 20
LATITUDE_KEY LE 22
LONGITUDE_KEY GE 30.5
LONGITUDE_KEY LE 31.0
```

After receiving a reply from a given LS node, the LC can invoke the service (turn the light on):

```
CLASS IS interest
TYPE IS "service invocation"
SERVICE_KEY EQ "light dimmer"
LATITUDE_KEY EQ 21
LATITUDE_KEY EQ 30.7
i_LIGHT_LEVEL IS 100
```

With this, the typical SD lookup and invocation task is completed. The eventual subsequent exchanges between the LC and the SP are out of the scope of this document.

3.2.4 Metadata repository

In Section 2.1 we argued for the need of a metadata repository service that will maintain a list of the capabilities of the network. In the case of the CBPS based SD proposal, it is clear that this metadata repository should be geared towards the components of the CBPS middleware. At the very minimum, the metadata repository should keep a record of the available CBPS attributes and operators. The concrete implementation of the metadata repository will closely follow the design of the Central Repository Service (CRS) as proposed by SGROI et al. [44]. The records in the repository will be updated in two ways:

Automatically when a node that introduces new attributes is added to the network. The metadata service will query the attributes stored in the node's service description record and add the new attributes to the repository. Same procedure applies for automatic removal of the attributes.

Explicitly by an application that calls the repository API primitives discussed below.

The application or the external users (Section 2.2.3 can interact with the repository via the following primitives:

`char* MRGetAttributeList()` that returns a list of the attributes currently present in the repository.

`int MRCheckAttribute(char* attribute)]` that checks whether the specified attribute already exists in the repository.

`int MRAddAttribute(char* attribute)` that adds new attribute record in to the repository.

`int MRDeleteAttribute(char* attribute)` that deletes the specified attribute from the repository.

Similar API will be used for registering and querying the CBPS operators. The consistency of the state will be maintained using the same mechanisms discussed in Section 3.1.2.

Chapter 4

Conclusion

Although with a different focus then in the traditional case, the SD-like capability, is still an essential component for any reconfigurable, dynamic and multi-aplication WSN.

As shown in Chapter 2, the specific properties of the WSNs exert diverging demands on the SD functionality. Trough several generic use-cases, we have tried to uncover the full range of different services that the SD component has to provide to the upper protocol layers.

Because of its particular nature, the implementation of the SD functionality in the WSNs is constrained by conflicting requirements. From one side, the bridging and the gatewaying use-cases require a standard conforming solutions that will enable easy integration of the network with the external world. On the other side, the resource constraints in the in-network use-case ask for customized implementatons that minimize the communication overhead.

In this light, we are concentrating our interests on two possible solutions for the problem. Both of them rely on a tight integration between the routing and the SD functionality as a necessary precondition for an energy-efficient implementation that fits the tight resource budget of a typical WSN.

The first approach (Section 3.1) is based on extending the traditional ad hoc on-demand routing protocols with a service discovery capability as proposed in [34]. Their proposal was implemented and significantly extended with several optimizaton schemes: negative announcements, intelegent scoping and closest-first anycasting.

The second approach (Section 3.2) is a lightweight solution that uses the EYES semantic addresing architecture [20] as basis for the implementation. The reuse of the CBPS subscriptions as *service lookups* and CBPS advertisements as *service announcements* enables a service discovery functionality at a very low price in energy terms.

In Section 3.1.2 we have explored the use of *caching* as a basic mechanizm for increasing the scalability of the above solutions. While it is clearly benefitial in lowering the amount of messages, caching also creates state maintenance problems that must be adequatly addressed. We argue that the appropriate solution to the consistency problem depends both on the service types and on the networking architecture.

Bibliography

[1] S. Avancha, A. Joshi, and T. Finin. Enhancing the Bluetooth Service Discovery Protocol. Technical report, Computer Science and Eletrical Engineering, University of Maryland Baltimore County, Aug 2001.

[2] M. Balazinska, H. Balakrishnan, and D. Karger. INS/Twine: A Scalable Peer-to-Peer Architecture for Intentional Resource Discovery. Pervasive 2002 - International Conference on Pervasive Computing, August 2002.

[3] D Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, 1992.

[4] C. Bettstetter and C. Renner. A comparison of service discovery protocols and implementation of the service location protocol. In *Proc. of EUNICE*, Twente, Netherlands, September 2000.

[5] Bluetooth Specification v.1.1 - Part E Service Discovery Protocol, February 2001. `www.bluetooth.com`.

[6] S. Botros and S. Waterhouse. Search in JXTA and Other Distributed Networks. In *Proc. 2001 Int'l Conf. Peer-to-Peer Computing*, Lingköping, Sweden, August 2001. `http://people.jxta.org/stevew/BotrosWaterhouse2001.pdf`.

[7] Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems (TOCS)*, 19(3): 332–383, 2001.

[8] M. Castro, P. Druschel, Y. Hu, and A. Rowstron. Exploiting Network Proximity in Distributed Hash Tables. In Ozalp Babaoglu, Ken Birman, and Keith Marzullo, editors, *International Workshop on Future Directions in Distributed Computing (FuDiCo)*, pages 52–55, June 2002.

[9] Liang Cheng. Service Advertisement and Discovery in Mobile Ad hoc Networks. In *Conference on Computer Supported Cooperative Work (CSCW 2002)*, New Orleans, LA, USA, November 2002.

[10] S. E. Czerwinski, B. Y. Zhao, T. D. Hodes, A. D. Joseph, and R. H. Katz. An Architecture for a Secure Service Discovery Service. In *Mobile Computing and Networking (MobiCOM '99)*, pages 24–35, 1999.

[11] K. Edwards and T. Rodden. *Jini Example by Example*. Prentice Hall PTR, June 2001.

[12] M. Esler, J. Hightower, T. Anderson, and G. Borriello. Next Century Challenges: Data-Centric Networking for Invisible Computing. In *Mobile Computing and Networking*, pages 256–262, 1999.

[13] S. Waterhouse et al. JXTA Search Protocol Specification 1.0 (DRAFT). Technical report, Sun Microsystems, June 2001.

[14] Christian Frank. A Hybrid Service Discovery Approach for Mobile Ad-Hoc Networks. Diplomarbeit, Technische Universität Berlin, September 2003.

[15] Christian Frank and Holger Karl. Consistency Challenges of Service Discovery in Mobile Ad Hoc Networks. Unpublished paper, 2004.

[16] N. Gibbins and W. Hall. Scalability Issues for Query Routing Service Discovery. In *Proceedings of the Second Workshop on Infrastructure for Agents, MAS and Scalable MAS at the Fourth International Conference on Autonomous Agents (ICMAS2001)*, pages 209–217, 2001.

[17] L. Gong. Project JXTA: A Technology Overview, 2001.

[18] S. D. Gribble, M. Welsh, J. R. von Behren, E. A. Brewer, D. E. Culler, N. Borisov, S. E. Czerwinski, R. Gummadi, J. R. Hill, A. D. Joseph, R. H. Katz, Z. M. Mao, S. Ross, and B. Y. Zhao. The Ninja architecture for robust Internet-scale systems and services. *Computer Networks*, 35 (4):473–497, 2001.

[19] E. Guttman, J. Veizades, C. Perkins, and M. Day. RFC 2608: Service Location Protocol, Version 2, June 1999.

[20] Vlado Handziski, Andreas Köpke, Christian Frank, and Holger Karl. EYES Deliverable D3.3: ”Semantic Addressing”, August 2003.

[21] H.-J. Hof, E.-O. Blaß, T. Fuhrmann, and M. Zitterbart. Design of a Secure Distributed Service Directory for Wireless Sensor Networks. In *1st European Workshop on Wireless Sensor Networks (EWSN)*, pages 172–187, Berlin, Germany, January 2004. Springer-Verlag.

[22] D. Johnson, D. Maltz, Y-C. Hu, and J. Jetcheva. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR). IETF Internet Draft, April 2003. Work in progress, expires Oct 15. 2003.

[23] R. José. *An Open Architecture for Location-Based Services in Heterogeneous Mobile Environments*. PhD thesis, Computing Department, Lancaster University, England, April 2001.

[24] E. Kiciman and A. Fox. Using Dynamic Mediation to Integrate COTS Entities in a Ubiquitous Computing Environment. In *Second International Symposium on Handheld and Ubiquitous Computing (Lecture Notes in Computer Science)*, Bristol, England, September 2000. Springer Verlag.

[25] U. C. Kozat and L. Tassiulas. Network Layer Support for Service Discovery in Mobile Ad Hoc Networks. In *Proc. IEEE INFOCOM*, San Francisco, CA, USA, March 2003.

[26] H. T. Kung and et al. MotusNet: A Content Network. Submitted for publication `http://www.eecs.harvard.edu/~htk/publication/2001-cs244-content-network%.pdf`, 2001.

[27] J. Lawrence. LEAP into Ad-Hoc Networks. Workshop on Ubiquitous Agents on embedded, wearable, and mobile devices, July 2002.

[28] R. McGrath. Discovery and Its Discontents: Discovery Protocols for Ubiquitous computing. Technical Report UIUCDCS-R-99-2132, University of Illinois, Urbana-Champaign, April 200.

[29] L. Moreau. Distributed Directory Service and Message Router for Mobile Agents. *Science of Computer Programming*, 39(2-3):249–272, 2001.

[30] N. Nahata, P. Pamu, S. Garg, and A. Helmy. Efficient Resource Discovery for Large Scale Ad hoc Networks using Contacts. *ACM SIGCOMM Computer Communications Review*, 32(3):32, 2002.

[31] About Pervasive Computing. Pervasive Computing, 2001. `http://www.nist.gov/pc2001/about_pervasive.html`.

[32] Brian Oki, Manfred Pfluegl, Alex Siegel, and Dale Skeen. The Information Bus: an architecture for extensible distributed systems. In *Proceedings of the fourteenth ACM symposium on Operating systems principles*, pages 58–68. ACM Press, 1993.

[33] V. Park and J. Macker. Anycast Routing for Mobile Services. In *Conference on Information Sciences and Systems (CISS)*, March 1999.

[34] C. Perkins and R. Koodli. Service Discovery in On-Demand Ad Hoc Networks. IETF Internet Draft, October 2002. Work in progress.

[35] C. E. Perkins, E. M. Belding-Royer, and S. Das. Ad Hoc On Demand Distance Vector (AODV) Routing. IETF Internet Draft, February 2003. Work in progress.

[36] M. T. Prinkey. An Efficient Scheme for Query Processing on Peer-to-Peer Networks. Internet Draft `http://aeolusres.homestead.com/files/`, 2001.

[37] Bhaskaran Raman, Pravin Bhagwat, and Srinivasan Seshan. Arguments for Cross-Layer Optimizations in Bluetooth Scatternets. In *Proceedings of the 2001 Symposium on Applications and the Internet (SAINT 2001)*, page 176. IEEE Computer Society, 2001.

[38] P. Ratanchandani and R. Kravets. A Hybrid Approach for Internet Connectivity for Mobile Ad Hoc Networks. In *IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, LA, USA, March 2003. IEEE.

[39] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proc. ACM SIGCOMM*, 2001.

[40] RFC 3344: IP Mobility Support for IPv4, August 2002.

[41] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001.

[42] Salutation Architecture: Overview. The Salutation Consortium, 1998.

[43] Salutation Architecture Specification V2.0c (Part-1). The Salutation Consortium, June 1999.

[44] Marco Sgroi, Adam Wolisz, Alberto Sangiovanni-Vincentelli, and Jan M. Rabaey. A Service-Based Universal Application Interface for Ad-hoc Wireless Sensor Networks. Unpublished white paper, 2004.

[45] B. Silaghi, S. Bhattacharjee, and P. Keleher. Query Routing in the TerraDir Distributed Directory. In *SPIE ITCOM'02*, 2002.

[46] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Sclable Peer-to-peer Lookup Service for Internet Applications. In *Proc. ACM SIGCOMM*, pages 149–160, San Diego, CA, August 2001. `http://www.pdos.lcs.mit.edu/papers/chord:sigcomm01/`.

[47] Y. Sun, E. M. Belding-Royer, and C. E. Perkins. Internet Connectivity for Ad hoc Mobile Networks. *Intl. J. of Wireless Information Networks (special Issue on Mobile Ad hoc Networks)*, 9(2), 2002.

[48] D. Tang, C.-Y. Chang, K. Tanaka, and M. Baker. Resource Discovery in Ad hoc Networks. Technical Report CSL-TR-98-769, Stanford University, Aug 1998.

[49] UDDI Version 3.0 Published Specification, July 2002. `www.uddi.org`.

[50] J. Undercoffer, F. Perich, A. Cedilnik, L. Kagal, and A. Joshi. A Secure Infrastructure for Service Discovery and Access in Pervasive Computing. *ACM MONET: Special Issue on Security in Mobile Computing Environments*, 2002.

[51] Understanding Universal Plug and Play: A White Paper. Universal Plug and Play Forum, June 2000.

[52] Universal Plug and Play Device Architecture. UPnP Forum, June 2000. Version 1.0.

[53] J. Wu and M. Zitterbart. Service Awareness in Mobile Ad Hoc Networks. In U. Killat and W. Lamersdorf, editors, *Kommunikation in Verteilten Systemen (KiVS)*. Springer, February 2001.

[54] Y. Yi and S. Lee. On-Demand Multicast Routing Protocol (ODMRP) for Ad-Hoc Networks. IETF Internet Draft, November 2002. Work in progress.

[55] B. Y. Zhao, Y. Duan, L. Huang, A. D. Joseph, and J. D. Kubiatowicz. Brocade: Landmark Routing on Overlay Networks. In *First International Workshop on Peer-to-Peer Systems (IPTPS 2002)*, Cambridge, MA, March 2002.

[56] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. Technical Report UCB/CSD-01-1141, UC Berkeley, April 2001.

Acronyms

AODV	Ad hoc On-Demand Distance Vector Routing
API	Application Programming Interface
AVTree	Attribute-Value-Tree
CBPS	Content-Based Publish/Subscribe
CRS	Central Repository Service
DCR	Data Centric Routing
DHT	Distributed Hash Table
DLS	Distributed Lookup Service
DSDP	Distributed Service Discovery Protocol
DSDV	Destination-Sequenced Distance-Vector Routing
DSR	Dynamic Source Routing
EYES	Energy-efficient Sensor Network
GHT	Geographic Hash Table
HVAC	Heating, Ventilation, and Air Conditioning
ID	Identification
IP	Internet Protocol
IST	Information Society Technologies
JXTA	Sun's Project JXTA
Jini	Sun's Project Jini
LC	Lookup Client
LS	Lookup Server
MAC	Medium Access Control

MANET	Mobile Ad-Hoc Network
MAS	Multi-Agent Systems
ODMRP	On-Demand Multicast Routing Protocol
P2P	Peer-to-Peer
PDA	Personal Digital Assistant
RMI	Remote Method Invocation
SDP	Service Discovery Protocol
SD	Service Discovery
SLP	Service Lookup Protocol
SM	Salutation Manager
SP	Service Provider
TCP/IP	Transmission Control Protocol/Internet Protocol
UDDI	Universal Description, Discovery and Integration Protocol
UDT	User Defined Type
UPnP	Universal Plug-and-Play
WSN	Wireless Sensor Network
XML	Extensible Markup Language