A Classification of Service Discovery Protocols

Raluca Marin-Perianu Pieter Hartel Hans Scholten

June 2005

Abstract

Service discovery is an emerging field in the area of ubiquitous computing. There are various techniques and protocols (proposed or/and already implemented) particularly tailored to specific sets of objectives. This paper analyses the current state of the art and presents a taxonomy of service discovery protocols. Along with design issues, the diversity of solutions and open questions are systematically studied.

1 Introduction

1.1 Service Discovery definition

Service discovery paradigm arises in the context of the new field of self-organization in information systems. A service in the network can be any software or hardware entity that a user might be interested to utilize. A definition of service discovery is given by Wikipedia [30]: "Service discovery protocols are network protocols which allow automatic detection of devices and services offered by these devices on a computer network." In other words, service discovery is the action of finding a service provider for a requested service. When the location of the demanded service (typically the address of the service provider) is retrieved, the user may further access and use it.

1.2 Service discovery entities

Each service discovery protocol consists of at least two basic participating elements:

- *Client (or user)*: the entity that is interested in finding and using a service
- *Server*: the entity that offers the service

Protocols may use service repositories to facilitate service mappings. Therefore, it is common to find a third participating entity within service discovery protocols:

- *Directory (or server, broker, central, resolver)*: a node in the network that hosts partially or entirely the service description information

1.3 Service Discovery objectives

We identify the following objectives of service discovery protocols:

- *Discovery.* The ability to find a service provider if there is a service in the network with the properties described in the request. In order to achieve this goal, protocols implement the following functions:
  - *Use a description language.* To facilitate the discovery process, services are semantically described following a certain description language. Service requests are also expressed using this description language.
  - *Storage of service information.* Solutions for service information storage vary between two end-points: completely centralized and completely distributed (peer-to-peer) architectures. There is a wide range of hybrid solutions that have been proposed for achieving better results under specific circumstances or assumptions.
  - *Search for services.* Service requests are expressed using the description language and are addressed to the directory nodes or disseminated in the network.
- *No administration.* The network must organize and deliver information about its content without human intervention. This objective translates into the following functional issues:
  - *Maintenance against changes in service description.* Services may change their characteristics and the service discovery protocols should be aware of the new service descriptions available.
  - *Maintenance against topology changes.* Services may become unavailable or new services may be added to the network. Reliable information about the available services must be provided to the user while maintaining a consistent view of the available services.

1.4 Service Discovery additional issues

Depending on the characteristics of each protocol, other issues might be added to the overall design:

- *Functional issues.*
  - *Service selection.* Some protocols may implement automatic service selection, based on a set of metrics that is used to define the best service offer.
  - *Service usage.* Apart from performing service discovery, protocols may also offer methods for using the services.
  - *Security and privacy.* Because many entities interact with each other within a service discovery protocol, it is certain that security and privacy play an important role.
- *Performance issues.*
  - *Network scalability.* This requirement applies to protocols which are designed to manage large networks. Two goals have to be achieved in order to enable scalability:

    - *Load balancing.* Nodes being overloaded by registration or queries may delegate their work to other nodes.
    - *Query efficiency.* Queries have to be efficiently routed through the network in order to minimize the traffic.
  - *Fault tolerance.* Systems are designed to cope with the failure of servers, being able to run backup algorithms.
  - *Mobility support.* This feature applies to highly dynamic environments, in which nodes arbitrary may join, leave or change their position within the network. The information regarding available services need to rapidly adapt to these changes.
  - *Integration of resource-poor devices.* This issue concerns designing a lightweight protocol which can be run on PDAs, mobile phones, home appliances or even sensors that might be able to offer or use services.

2 Classification

An large number of discovery protocols have been proposed, focused on different applications. We classify the service discovery protocols by the network type, the functionality and the performance. Firstly, we group the service discovery protocols by the network category they are designed for. Secondly, we examine each functional issue separately (service description, storage, maintenance, search, service selection, service usage, security and privacy) and we point out the various implementation methods. Finally, we give a detailed description on how the major performance issues (such as network scalability, fault tolerance, mobility support and resource-awareness) are achieved.

2.1 Type of network

The design and performance of service discovery protocols are influenced by the characteristics of the network. We can identify the following relevant network features:

- *Size.* Network size may vary between small (a couple of devices) and very large (wide area).
- *Speed.* Speed may very from hundreds of Kilobits per second up to Gigabits per second.
- *Dynamics.* Networks can be static, such as the traditional wired local area networks, or very mobile, such as wireless ad-hoc networks.
- *Type of devices.* Devices participating in the network can vary from powerful servers to resource-poor devices, such as sensors.

Following these characteristics, we group networks in three classes: local area, wide area and ad hoc.

2.1.1 Local Area Networks

Service discovery protocols were first designed for this type of networks. Enterprise environments, home and office settings fall in this category. As the

total number of devices is limited to one administrative domain, it is assumed that LANs can provide services as DHCP. Generally, devices are considered to be resource-rich (e.g. computers, printers, faxes), but sometimes it is also necessary to integrate some resource-poor devices (as home appliances). Along with these characteristics, LANs have a backbone of reasonable network speed to permit heavy message communication between devices. In addition, service providers do not enter and leave the network very often and mobility is not a strong issue, so we expect a stable network configuration. In these conditions, service discovery can be very flexible in design and reliable in operation. Existent protocols in this area include the following: Jini [19], UPnP [11], SLP [13], FRODO [27].

2.1.2 Wide Area Networks

Wide area networks are characterised by a large number of devices and services that have to be managed. In addition, there are no broadcast or multicast mechanisms available. This results in the a set of challenging issues. First of all, scalability must be achieved, taking into consideration the size of the network. Second, the trade-off between consistency of stored information and network traffic must be well considered. Third, the threshold between the number of information replicas present in the network and the depth of search has to be evaluated. Examples of protocols developed in this area are the following: SSDS [15], CSP [1], INS/Twine [5], Superstring [2], GloServ [4].

2.1.3 Ad Hoc Networks

Although quite a large number of researchers have explored this area until now, more work has to be done in order to cover the diversity of the domain. Such a network is comprised of devices that gather in an improvised manner. The main characteristic of this type of network is the dynamics. The usual case is that every device acts as a router and has to maintain routing tables. Challenges imposed by this approach concern the instability of the network (mobility and node failures), very low or even inexistent caches and resource constraints (like energy). Some proposed solutions rely on the routing protocol to achieve both data forwarding and service discovery. Broadcast and multicast mechanisms are usually extensively used in these networks. Bluetooth SDP [9], LANES [16], Konark [14], Service Rings [17], GSD, [7], Allia, [23], DEAPspace [21] and cross-layer types of service discovery (such as Wu and Zitterbart [32], Cheng and Marsic [8] and Varshavsky et. al [29]) are examples of initiatives in this area.

The network characteristics are important in designing the type of storage, explained in section 2.2.2. A summary of the basic network features and storage types is given in Table 1.

2.2 Functional issues

2.2.1 Service description

Service discovery protocols rely on the existence of a service description. Users search for a particular service according to the fields and values in the service description.

Table 1: Comparative table network types

	LAN	**WAN**	**Ad Hoc**
Size	Medium (hundreds to thousand devices)	Large	Small
Speed	High	Medium	Low
Dynamics	Low	Medium	High
Type of devices	Powerful devices to home appliances	Powerful devices	Resource-poor devices
Types of storage	Centralized or unstructured distributed	Structured distributed	Unstructured distributed
Examples of protocols	Jini [19], UPnP [11], SLP [13], FRODO [27]	SSDS [15], CSP [1], INS/Twine [5], Superstring [2], GloServ [4]	Bluetooth SDP [9], LANES [16], Konark [14], Service Rings [17], GSD [7], Allia [23], DEAPspace[21], Wu and Zitterbart [32], Cheng and Marsic [8], Varshavsky et. al [29])

The most widely used description format is the attribute-value structure. For instance, SLP uses service templates registered with IANA, whose attributes are defined in a template document that is readable by people and machines. In Bluetooth SDP, all the information about a service is contained within a service record, which consists entirely of a list of attributes. Some protocols use hierarchical attribute-value pairs, such as INS name-specifiers, while many of them rely on XML. Ontologies based on XML enable a better classification and make use of schemas for attribute definitions. For instance, GloServ [4] utilizes RDF for service descriptions, whereas GSD [7] chooses DAML+OIL.

Other protocols make exception from using simple text attribute-value schemes for service descriptions. In Jini [19], attributes are Java objects that can be associated with services to provide descriptive information about them.

Searching for services described in user requests or queries comes along with service matching. User queries which are comprised of attribute-value pairs require that matched content simultaneously satisfy all the pairs present in the request. If one attribute is not specified, it is generally considered that it can take any value. One alternative is to use wild-card matching (INS [3]). In Jini, finding an appropriate service requires passing a template that is used to match and filter the set of existing services.

Some protocols enable multi-criteria selection, which results in a collection of nodes that satisfy more than one condition. SSDS [15] supports this feature by implementing a method called filtered query flooding, where summaries of nodes contents are used as filters for the queries.

2.2.2 Storage of Service Information

Information retrieval on available services relies on the storage system. Servers are aware of the data they provide, but in order to be easily and rapidly accessed, several additional storage mechanisms have been developed. Network size and type is very important in designing the storage system. In ad hoc networks storage may be inexistent due to increased mobility, whereas in wide

area networks it is compulsory to have intermediate storage, although this can increase substantially the design complexity.

1. Centralized This approach is optimal in terms of rapid access to data and low traffic, although it creates a single point of failure. SLP with a Directory Agent [13] and Jini [19] are examples that have a central server for information storage. They use the server only for information retrieval, the actual communication between the client and the server being done in a peer-to-peer manner.

2. Unstructured Distributed In unstructured distributed storage systems, communication is based on broadcast or multicast mechanisms. This technique is common for protocols designed to work in local area networks and ad hoc networks.

Several methods are used to obtain and maintain the service data:

- service providers flood the network with service advertisements
- clients flood the network with discovery messages
- nodes cache the service advertisements
- nodes overhear in the network traffic and cache the interesting data

UPnP [11], Konark [14], DEAPspace[21], Wu and Zitterbart [32], Varshavsky et. al [29], GSD [7] are examples of protocols in which every interested node maintains it own view of services and devices in the network.

Another important issue is how complex the view that each device should maintain is. While protocols such as UPnP [11] delegate each node to have a consistent view of all devices and services present in the network, GSD [7] proposes that a node caches the service descriptions available in a maximum number of hops (the diameter of its knowledge).

3. Structured Distributed Structured distributed storage methods are commonly found in the context of large networks, where the previously mentioned storage solutions do not scale well. Directory nodes organize themselves in an overlay structure that allow them to route the discovery messages in a limited number of hops. We classify the structured distributed systems into three categories: hierarchical, flat and hybrid.

Hierarchical This type of storage follows the DNS model. Information, advertisements and queries are propagated up and down through the hierarchy. Parents store information of their children and thus it is possible that the root node becomes a bottleneck. If the size of network is huge – in the case of wide area networks – a compression method for stored information is necessary. Protocols implement continuous aggregation of descriptions while they travel up to the root node. Examples of protocols which implement hierarchical storage are SSDS [15] and CSP [1].

Flat Protocols that fall in this category rely heavily on peer-to-peer overlay networks, constructed by means of *distributed hash tables (DHT)*, such as CAN [22], Chord [26], Pastry [24] and Tapestry [33]. DHTs are used to store key-value pairs on designated nodes. Messages are input to a hash function and routed in a bounded number of hops to the nodes responsible for the resulting key.

Each node maintains a routing table with identifiers and network addresses of other nodes. The major advantage of the DHT protocols is the efficient lookup mechanism, which normally is performed within O(log(N)) hops, where N is the number of nodes in the overlay network.

Protocols of this type can use theoretically any existing DHT, but they often chose one of them for implementation. INS/Twine protocol [5] relies on Chord for hash-based partitioning of resource descriptions among a set of symmetric peer resolvers. One Ring Rules Them All [6] uses structured P2P overlays as a platform for service discovery and chooses Pastry as an example. The infrastructure proposed relies on a universal ring that all participating nodes are expected to join. Other examples are Superstring [2] and CDS [12], which distribute the cost of storage and query resolution among several nodes (detailed in section 2.3.1).

Hybrid Hybrid solutions combine ideas from the above hierarchical and flat storage mechanisms with additional optimization techniques. Some of them rely on hierarchical models to organize groups of nodes. Others try to overcome the disadvantages of the DHT approaches (e.g. the cost of maintaining a consistent distributed index), while preserving their benefits (e.g. the efficient lookup mechanism).

Service Rings [17] deal with a hierarchical ring architecture. A ring is a group of devices that are physically close to each other and offer similar services. Each service ring has a designated service access point (SAP), which stores information about all the services offered by the ring. SAPs can be organized in higher-level rings, which also have SAPs that store summaries of services they provide. Rings permanently monitor the network traffic and make decisions for optimizing their structure.

LANES [16] builds up an overlay network inspired by CAN. The two-dimensional CAN structure is optimized by constructing loosely coupled lanes of nodes. Service announcements are propagated through a lane in a top-bottom manner. As a consequence, an arbitrary node from the lane has full information of the services offered by that lane. Service requests are propagated to another lane using anycast routing. On average, this algorithm distributes the descriptions to $\sqrt{N}$ nodes, where N is the total number of nodes in the network.

The Project JXTA protocols [28] establish a virtual network overlay on top of existing physical network infrastructure. This approach combines DHT methods with a random walker that checks for non-synchronized indices. The resolver nodes are not required to maintain a consistent distributed hash index, in order to avoid expensive network traffic. Instead, a limited-range walker is used to walk the rendezvous from the initial DHT target.

The resolver network in INS [3] is comprised of nodes organized in a spanning tree. To handle excessive lookup loads, resolver nodes (INRs) spawn instances on other candidate resolvers. There is a central component in the network, named Domain Space Resolver (DSR), which maintains a list of active and candidate INRs. Each new node contacts first the DSR to get the list of resolvers and then picks one (based on the round-trip latency) to establish a neighbour relationship.

Overlay networks can be built also by making use of backbone selection algorithms. Kozat and Tassiulas [18] tackle the solution of a distributed directory by creating and maintaining a network backbone. Selected nodes are considered to form a relatively stable dominating set. They receive and process service requests, acting as service discovery agents (DA).

2.2.3 Maintenance

Maintenance is regarded as a permanent adjustment of the service information stored in directory nodes. However, if no service information is maintained outside the server nodes, we can speak of *implicit* service information update, by means of active discovery ([8]) or simple request/response discovery (Bluetooth SDP [9]). In the case that service information is stored on directory nodes, protocols need to implement two types of maintenance:

- *Maintenance against changes in service description.* The storage system needs to maintain consistency against the changes in the service characteristics. There are two methods for achieving this objective:
  - *Advertise the service with a different description.* Nodes adjust the service information according to the new advertised description. This technique is mostly used in unstructured distributed systems, where all the nodes receive the service advertisements, either they are interested in the service or not.
  - *Event notification or publish/subscribe.* Services publish their offer and interested clients subscribe for events with the directory nodes or directly with services. Service updates are received only by the nodes which have subscribed for the service. For example, in Salutation, an event notification requested by a client is called a "long-term request". Eventing in UPnP consists of services publishing updates and control points subscribing to receive this information. FRODO [27] uses a mechanism that allows subscribing Control devices to receive notifying messages about any change in the state of services. Cheng and Marsic [8] propose that any appliance interested in a service sends a "join reply" packet, which is a registration for further reception of updated service advertisements.
- *Maintenance against topology changes.* Services can be added or deleted from the network or network topology may change, while central nodes have to preserve a consistent view of the services being available. The following state management schemes are used to control service availability:
  - *Soft state.* Services have to periodically re-register with the directories in order to restate their availability; if within a certain amount of time no service confirmation is received, cache entries are deleted.
  - *Hard state.* Records do not expire in a specific amount of time; they remain unchanged until they are explicitly deleted. Deletion may occur when services leave the network and send explicit deregistration messages. To overcome the problem that nodes can fail without prior deregistration, several schemes can be used, such as pinging the service for checking its availability.
  - *Hybrid state.* Some protocols may implement hybrid state management for combining management-simplicity of soft state to low-bandwidth requirements of hard-state. For example, with INS/Twine protocol [5], resolvers at the edge of the network refresh state information at a higher frequency than resolvers in the network core. When a departure without prior de-registration occurs, the first resolver that detects it sends explicit remove messages to the other resolvers.

Structured distributed systems that build overlay networks have to follow certain structural conditions and therefore implement algorithms for preserving consistency in the case of nodes entering or leaving the network, and in the situation of network partition and reintegration. Protocols dealing with this problem include Service Rings [17] and LANES [16].

2.2.4 Search methods

Depending on what type of storage each protocol chooses, different discovery mechanisms can be identified. The object of discovery can be:

1. *Directory node* – in centralized and structured distributed storage environments, nodes need to discover the directory node for sending their advertisements and requests.

2. *Directory nodes in the overlay structure* – in structured distributed storage systems, directory nodes need to route service discovery messages to other directory nodes in the overlay structure.

3. *Services* – in unstructured distributed storage systems, nodes have to manage themselves for finding appropriate services (possible by flooding)

Regarding the search for services, it is important to know that the extension of the search is conditioned by the dispersion degree of the information in the network. We must always take into account the threshold between initial dissemination of service descriptions and the extension of the following queries. More organized and distributed information translates into less search effort. On the contrary, complex storage mechanisms make the information consistency difficult to maintain. That is why, in very mobile networks, flooding may be the only option for service lookup.

There are mainly two types of obtaining information about servers and their offer:

1. *Passive Discovery (or Push Model)* – services or brokers announce their presence

2. *Active Discovery (or Pull Model)* – an application needs information about services or brokers and sends discovery messages

In general, protocols implement both methods. Advertised service descriptions can be the local ones (as in UPnP), or the entire local database, including services offered by others (as in DEAPspace [21]). Complex search is effectuated within distributed algorithms. In this case, information flows and is stored in the overlay network based on the rules specific to each protocol. For example, in the LANES protocol [16], service announcements are propagated throughout a lane, whereas service requests are sent to other lanes. In hierarchical approaches, data flow up and down the hierarchy.

2.2.5 Service selection

After submitting a query for a certain service, it is often the case that multiple servers can offer the specified service. The best service can be chosen manually by the user, or by means of an optimization algorithm implemented on the client's side, or can be selected by one of the directory agents present in the

system. When considering the last two cases, a very important issue is the metric used to define the best offer. For example, Varshavsky et al. [29], consider that the application running on the client chooses the server with the lowest hop count. Similarly, CDS [12] allows the client to make use of a query optimization algorithm, which selects the best Rendezvous Points (RP). Two criteria can be taken into account: the RP with the smallest response time or the RP with the smallest database.

CSP [1] introduces the concepts of static and dynamic context-attributes. Static attributes are the ones that keep fixed values set at the time of announcements, while the value of a dynamic attribute is determined at the time of lookups. Examples of context attributes are: distance to server, server load and channel conditions. The evaluation of context attributes is done by the Broker Agent (BA) which is responsible for the service registry.

2.2.6 Service usage

As already mentioned, service discovery concerns the process of searching for service providers according to certain criteria. Therefore, further service usage is a separate issue, although several protocols also address this problem.

After discovery, the client usually connects directly to the device offering the service. Commands can be given using RPC, Java RMI, Authenticated RMI, SOAP. A protocol based on HTTP for communication and on XML for descriptions may use SOAP for service control. Jini utilizes Java RMI for communication between services, which allows not only data to be passed from object to object around the network but also full objects, including code.

2.2.7 Security and privacy

Security is a widespread problem that has not been overlooked by service discovery protocols. Sometimes this issue is so important that is part of the main design strategies. Major security constraints include authentication, authorization, trust, confidentiality, integrity, and non-repudiation. The following definitions are taken from RFC2828 [25].

- *Authentication* − Authentication is the process of verifying an identity claimed by or for a system entity. Entities may be services, clients or central directories. Clients may have to authenticate in order to use a service (in Salutation [10]) or Service Agents and Directory Agents have to prove to clients that they are what they pretend to be (SLP [13]). Other protocols, such as SSDS [15], specify that all endpoints have to authenticate to each other. Options for achieving authentication include username and password methods (Salutation [10]) and stronger ones such as digital signatures.

- *Authorization* − Authorization is the right or permission that is granted to an entity to access a system resource. Options for achieving authorization include the usage of capabilities (SSDS [15]) or access control lists (Jini [19]). Credentials can also be used for establishing either a claimed identity or the authorizations of a system entity (JXTA [28]).

- *Trust* − Trust means the extent to which someone who relies on a system can have confidence that the system meets its specifications. A complex trust mechanism has been developed by SuperstringRep [31] using the

reputation of an entity to influence the choice of a service. The reputation system keeps track of what other nodes have experienced while using a certain service. When a new client wants to make use of a particular entity, it first queries the reputation system and obtains a service score. Other protocols (Splendor [34]) use certificate authority-based trust models that provide public key infrastructures to secure transactions.

- *Confidentiality* – Confidentiality is the property that information is not made available or disclosed to an unauthorized entity. This goal is usually achieved through encryption. For example, in SSDS [15], public key and symmetric key encryption are used for confidentiality.

- *Integrity* – Integrity is the property that data has not been changed, destroyed, or lost in an unauthorized or accidental manner. Message integrity can be achieved through verification mechanisms, such as hashes (Splendor [34]).

- *Non-repudiation* – Non-repudiation is a security service that provides protection against false denial of involvement in a communication. In Splendor [34], for non-repudiation purpose, data is hashed and hashes are signed before encryption.

- *Privacy* – Privacy is the right of an entity to determine the degree to which it will interact with its environment, including the degree to which the entity is willing to share information about itself with others. For instance, in Splendor [34], the location information of users is kept private until they want to reveal their position.

2.3 Performance issues

2.3.1 Network scalability

Scalability is one issue tightly related to load balancing and query efficiency. Considering delay as the primary performance metrics for network scalability, protocols implement algorithms to avoid circumstances in which servers are burdened with huge storage of service information and overwhelmed by registration messages and queries. Moreover, delay is strictly related to the efficiency in finding service information in large networks. We analyse the methods that have been proposed for achieving scalability, considering both load balancing and query efficiency issues:

1. *More directory nodes and service grouping* – Protocols designed for local area networks may address the scalability issue by enabling service grouping and designating multiple directory nodes to handle service registrations (possibly assigned to specific groups). For instance, in SLP, services are grouped together using 'scopes', which are strings that identify services. Scopes are the primary means for deployments of large networks in SLP. Discovery and lookup can be performed within a certain scope, administrated by a designated Directory Agent. Likewise, in Jini, services can be configured to register with lookups that are part of particular groups.

2. *Caching used to avoid overloading nodes* – This technique prevents the directory nodes responsible for storing information associated to a service from being overloaded if the popularity of the service increases. One Ring to Rule Them All [6] uses this method for load balancing.

3. *Distribute load to new servers* – If one directory node is overwhelmed by service registration and queries, it might delegate other nodes to join the distributed directory and to take over a part of their effort. In SSDS [15], if a particular server is overloaded, a new one can be spawned to a nearby machine, assigned to be a child of the overloaded server.

   The Superstring service discovery protocol [2] distributes the cost of storage and query resolution among several nodes. Descriptions are disseminated through a hierarchy that precisely matches the service description hierarchy. The root stores the key of the top level component of the description and hashes each of the following elements, passing them to its neighbours (that become its children). Thus, the computational cost for any query is shared among many resolvers.

   CDS ([12]) tackles the load concentration problem by constructing a load balancing matrix (LBM), which is designed to share the information otherwise assigned to a single node. The matrix grows and shrinks dynamically, as a result of the registration and query load. Each column contains a partition of the content and nodes in the same column are replicas of each other. When more registrations are encountered, new partitions are added (more columns), and when more queries are made, new replicas are added to the matrix (more lines).

4. *Hierarchical structures* – DNS-like hierarchical structures enable network scaling to a large number of nodes. Parents store information about their children and queries pass up through the hierarchy until a possible match is encountered. After that, requests are routed down to the child offering the specified service. SSDS [15], CSP [1] and Service Rings [17] are protocols that use hierarchical structure to achieve scalability. Protocols based on hierarchical storage handle better the overloading problem of directories. Typically, when a node is overwhelmed by the amount of information it is supposed to store or is flooded with queries, it can easily spawn the load on a nearby machine, assigned to be its child. On the contrary, if a node considers that it is insufficiently burdened, it can delegate the load to its parent.

5. *Distributed hash indexing* – DHT approaches provide an efficient and scalable index lookup mechanism. Messages are routed in O (log N) hops, where N is the number of resolvers in the network. Examples of protocols that use DHT are INS/Twine [5], Superstring [2], JXTA [28], One Ring to Rule Them All [6].

6. *Overlay structure optimization* – To improve efficiency in overlay networks, some protocols adapt to network changes by optimizing network structure. For instance, LANES [16] deals with inefficient inner lane connections and lanes splitting and merging to achieve the optimal size. Another example is Service Rings protocol [17], which implements algorithms for ring splitting and merging and also ring restructuring for efficient overlay links.

7. *Aggregation of descriptions* – In the case that the network is large, some protocols with hierarchical directories utilize compression methods for storing information. For instance, SSDS [15] provides an aggregation of service descriptions as they travel farther from their source through a filtering scheme named Bloom-filtered crossed terminals (BCT). The result is a bit vector that summarizes its collection of resources. One should be

aware of the fact that due to compression mechanisms, directories cannot always issue the correct answer because of false hits and in consequence, backtracking methods have to be used in order to reach the expected result. Another example is CSP [1], which is based on the Centroid aggregation method. This approach exploits the word frequency pattern followed by the Zipf distribution. CSP specifies that seldom used words need to be discriminated in favour of more frequently used ones at the upper layers of the index tree. The attributes that are not very spread, encounter aggressive aggregation that forces service information to quickly converge to a smaller size set.

2.3.2 Fault tolerance

In this section, a fault is referred to as the failure of directory nodes. For enforcing continuous operation of the network when faults occur, several solutions have been envisaged:

- *Redundancy.*
  - *For centralized storage.* Maintain a backup of the central directory node (FRODO [27]). When the central fails, the backup takes over the brokering function.
  - *For distributed storage.* Maintain multiple copies of the service information on several resolver nodes. When one of the resolver nodes fails, the remaining ones can still respond to the query. (INS [3], INS/Twine [5], JXTA [28], One Ring Rule Them All [6], LANES [16])
- *Reinitialization.* Re-election of the central node and re-registration of services (FRODO [27])
- *Multi modal functioning.* Optional peer-to-peer operation for the protocol (SLP [13])

2.3.3 Mobility support

Weak mobility support is implemented by most protocols along with maintenance (see section 2.2.3). However, in networks where dynamics is the primary characteristic, regular maintenance techniques are not sufficient for keeping information up-to-date.

Two methods for solving this problem have been proposed:

1. *Reactive method*, where information changes according to events in the network (for example, the route to server is no longer available)
2. *Proactive method*, where nodes maintain a consistent view by periodically exchanging update messages

Cross-layer discovery protocols implement these methods by using routing information for making decisions of service availability or by sniffing the network traffic and process it in order to have information on additional services.

For example, Wu and Zitterbart [32] stipulate three alternatives for updating service information: implicit service confirmation, reception of a service poll reply and reception of a “no service” indication. Implicit service confirmation is

based on snooping into bypassing data packets and on extracting useful information from routing updates. A service poll reply is stored by all intermediate nodes, so it is used for updating caches, whereas "no service" indication explicitly stipulates that a service is no longer available.

Varshavsky et al. [29] use both proactive and reactive methods for rediscovery. The latter one relies on available routing information for making decisions. Applications may choose between the eager and the lazy policies. The eager policy triggers rediscovery as soon as the active route to the current server breaks. The lazy policy delays active rediscovery until it has tried all known routes to all known servers that implement a service. Rediscovery is usually followed by reselection. There is a spectrum of possibilities for doing reselection. On one extreme lies performing reselection as a result to any change to the service table. The other extreme is to do reselection only when there is no valid route to the current server.

Certain protocols use mobility information for adjusting the service advertisement rate and the diameter of announcements. For example, GSD [7] implements a small advertisement time interval for highly dynamic environments, opposed to a larger value for rather stable networks. The extent of each advertisement (the number of hops that it passes) is also regulated depending on different mobility situations. Similarly, Allia [23] controls the frequency of advertisements and the diameter of the alliance taking into account the mobility of nodes.

2.3.4 Resource awareness

Protocols that support integration of resource-poor devices usually *delegate the work load to additional devices*. One example is Surrogate Jini [20], where a small device uses a host-capable machine that has the computational resources to execute code written in Java programming language on behalf of the device, and can provide the necessary resources that such code might need. In FRODO [27], 3D devices that cannot stand alone, delegate some of their work load more powerful 300D devices. Splendor [34] uses proxies for achieving privacy for service providers, offloading computational work and enabling mobile services to do authentication and authorization easily.

3 Conclusion and open issues

We propose a classification according to three main criteria:

1. *Network type.* This criteria is derived from the application domain the protocols are designed for. Being initially conceived for local area networks, protocols evolve towards wide-area and ad-hoc networks. For wide-area networks, scalability is the main research challenge. Ad-hoc networks face primarily the mobility problem.

2. *Functionality.* We identify the main functionality issues that service discovery protocols implement, such as service description, storage, maintenance, search methods, service selection, service usage, security and privacy.

3. *Performance.* We analyse the performance capabilities that protocols are able to achieve, such as scalability, fault tolerance, mobility, resource awareness.

We notice a distinctive trend of evolution in the field of service discovery. Protocols tend to increase the performance of ad-hoc environments with scalability and resource awareness. Mobile devices increase in number and decrease in capabilities, while service discovery protocols need to preserve the reachability issue. Moreover, functionalities such as service selection and security are constantly added and improved. However, security is often neglected in the design phase of service discovery protocols. Many solutions add security in a following step, which is a practice that do not deliver the expected secure protocol.

As future challenges, we mention the following:

- Scalability of mobile ad hoc networks one problem that needs to be further explored. Ubiquitous computing offers straightforward applicability to this issue. One example is the public environment, where many heterogeneous devices come together and have to discover and use a variety of services.
- Regarding resource-awareness, delegation of work load to powerful nodes might not be a solution when these devices are scarce and hard to reach. There is a need for lightweight protocols that are able to run on small and resource-poor devices. Sensor networks are a possible direction for developing new service discovery protocols.
- Security has to be included from the very beginning in the design of any service discovery protocol.
- Another research challenge is the inter-operability among service discovery protocols. Considering the effort that has been done in this area and the multitude of protocols that have arisen, it becomes very important to have a support for collaboration, that will permit discovery across heterogeneous networks which run different protocols.

We conclude that service discovery is still an open field, where many improvements can be added to the existing solutions and new protocols have to be developed to manage large scale environments and resource constraints of present and future devices.

References

[1] *A Multi-Tier Ubiquitous Service Discovery Protocol for Mobile Clients*, Montréal, Canada, 2003. Proceedings of the 2003 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2003).

[2] *Superstring: A Scalable Service Discovery Protocol for the Wide Area Pervasive Environment*, Sydney, September 2003. Proc. of the 11th IEEE International Conference on Networks.

[3] William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley. The design and implementation of an intentional naming system. In *Symposium on Operating Systems Principles*, pages 186–201, Charleston, SC, December 1999.

[4] Knarig Arabshian and Henning Schulzrinne. GloServ: Global service discovery architecture. In *MobiQuitous*, pages 319–325. IEEE Computer Society, June 2004.

[5] Magdalena Balazinska, Hari Balakrishnan, and David Karger. INS/Twine: A scalable peer-to-peer architecture for intentional resource discovery. In *International Conference on Pervasive Computing 2002*, August 2002.

[6] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, and Antony Rowstron. One ring to rule them all: Service discovery and binding in structured peer-to-peer overlay networks. In *Proceedings of the SIGOPS European Workshop*, Saint-Emilion, France, September 2002.

[7] Dipanjan Chakraborty, Anupam Joshi, Tim Finin, and Yelena Yesha. GSD: A novel group-based service discovery protocol for MANETs. In *4th IEEE Conference on Mobile and Wireless Communications Networks (MWCN)*. IEEE, September 2002.

[8] L. Cheng and I. Marsic. Service discovery and invocation for mobile ad hoc networked appliances, December 2000.

[9] Bluetooth Consortium. Specification of the bluetooth system core version 1.0 b: Part e, service discovery protocol (SDP), November 1999.

[10] The Salutation Consortium. Salutation architecture specification version 2.0c, June 1999. available online at http://www.salutation.org/.

[11] UPnP Forum. UPnP device architecture version 1.0, June 2000. available online at http://www.upnp.org/.

[12] Jun Gao and Peter Steenkiste. Rendezvous points-based scalable content discovery with load balancing. In *Networked Group Communication*, pages 71–78, 2002.

[13] E. Guttman and C. Perkins. Service location protocol, version, June 1999.

[14] S. Helal, N. Desai, V. Verma, and C. Lee. Konark – a service discovery and delivery protocol for ad-hoc networks, March 2003.

[15] Todd D. Hodes, Steven E. Czerwinski, Ben Y. Zhao, Anthony D. Joseph, and Randy H. Katz. An architecture for secure wide-area service discovery. *Wireless Networks*, 8(2/3):213–230, 2002.

[16] Michael Klein, Birgitta Konig-Ries, and Philipp Obreiter. Lanes - a lightweight overlay for service discovery in mobile ad hoc networks. Technical Report 2003-6, University of Karlsruhe, 2003.

[17] Michael Klein, Birgitta Konig-Ries, and Philipp Obreiter. Service rings - a semantic overlay for service discovery in ad hoc networks. In *DEXA Workshops*, pages 180–185, 2003.

[18] Ulas C. Kozat and Leandros Tassiulas. Service discovery in mobile ad hoc networks: An overall perspective on architectural choices and network layer support issues. *Ad Hoc Networks*, 2(1):23–44, June 2003.

[19] Sun Microsystems. Jini architecture specification version 2.0, June 2003.

[20] Sun Microsystems. Jini technology surrogate architecture specification, October 2003.

[21] Michael Nidd. Service discovery in DEAPspace. *ieee-pcm*, 8(4):39–45, August 2001.

[22] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard M. Karp, and Scott Schenker. A scalable content-addressable network. Technical Report TR-00-010, University of California, Berkeley, Berkeley, CA, August 2001.

[23] Olga Vladi Ratsimor, Dipanjan Chakraborty, Sovrin Tolia, Deepali Khushraj, Anugeetha Kunjithapatham, Anupam Joshi, Tim Finin, and Yelena Yesha. Allia: Alliance-based service discovery for ad-hoc environments. In *ACM Mobile Commerce Workshop*, September 2002.

[24] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–350, November 2001.

[25] R. Shirey. Request for comments: 2828, May 2000. Internet Security Glossary.

[26] Ion Stoica, Robert Morris, David R. Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, August 2001.

[27] V. Sundramoorthy, J. Scholten, P. G. Jansen, and P. H. Hartel. Service discovery at home. In *4th Int. Conf. On Information, Communications & Signal Processing and 4th IEEE Pacific-Rim Conf. On Multimedia (ICICS/PCM)*. IEEE Computer Society Press, December 2003.

[28] Bernard Traversat, Mohamed Abdelaziz, and Eric Pouyoul. Project JXTA: A loosely-consistent DHT rendezvous walker, May 2003.

[29] Alex Varshavsky, Bradley Reid, and Eyal de Lara. The need for cross-layer service discovery in MANETs, January 2004.

[30] Wikipedia. Service discovery — wikipedia, the free encyclopedia, 2005.

[31] Ryan Wishart, Ricky Robinson, and Jadwiga Indulska. SuperstringRep: Reputation-enhanced service discovery. In *ACSC*, 2005.

[32] J. Wu and M. Zitterbart. Service awareness in mobile ad hoc networks. Boulder, Colorado, USA, March 2001. Paper Digest of the 11th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN).

[33] 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.

[34] Feng Zhu, Matt W. Mutka, and Lionel M. Ni. Splendor: A secure, private, and location-aware service discovery protocol supporting mobile services. In *PerCom*, pages 235–242, New York, NY, USA, 2003. ACM Press.

Appendix - Comparative table on service discovery protocols

	Type of network	**Architecture**	**Storage of service information**	**Search methods**	**Event notification**	**Service description**	**Service selection and usage**	**Fault tolerance and mobility support**	**Network scalability**	**Security**
Jini [19]	Enterprise network	Centralized	On Lookup Service	Both active and passive discovery for finding the Lookup Service	Distributed events	Java proxy objects	Usage - Java RMI via proxy objects	Lease mechanism for granting access to services	-Service grouping -The use of federation	- Authentication - Authorization (access control lists) - Confidentiality -Integrity
UPnP [11]	Enterprise network	P2P	On every control point	Both active and passive discovery for finding services	Eventing mechanism	XML description, based on UPnP template language	Usage - messages encapsulated in SOAP	-Expiry time for advertisements -"Device unavailable" notification	-	-
SLP [13]	Enterprise network	1. Centralized (with DA) 2. P2P (without DA) *DA = Directory Agent	1. On DA 2. On UA and SA *UA = User Agent SA = Service Agent	1. both active and passive discovery 2. active discovery of services	-	Service templates registered with IANA	-	- Lifetime for service registrations	-More DA -Scope mechanism for service grouping	Optional authentication of DA and SA (using digital signatures)
Bluetooth SDP [9]	Small, maximum 8 devices that can be low-cost and low-power	Client-Server	On every server	Request / Response messages between client and server	-	Service attributes (ID and value)	-	Implicit (no caches are maintained)	-	Determined in the Bluetooth connection negotiation phase
Salutation [10]	Any network	Flexible (can be P2P or centralized)	Service Registry on every Salutation Manager	Salutation Manager Protocol between two SLMs	Long-term requests	Service description records	Usage - through RPC	Periodic availability check of services	-	User authentication (username and password)
INS/Twine [5]	Large and dynamic environments	Overlay network of resolvers which form a DHT	Each resolver holds a range of keys and their values	Service discovery messages are routed in O(log N) hops	-	Hierarchies of attribute-value pairs	-	-Each strand is stored on multiple nodes -Hybrid state management scheme	Hash-based partitioning of resource descriptions among resolvers	-

Continued on next page

	Type of network	Architecture	Storage of service information	Search methods	Event notification	Service description	Service selection and usage	Fault tolerance and mobility support	Network scalability	Security
SSDS [15]	Wide-area	Hierarchical structure of SDS servers	SDS servers	- passive discovery of SDS servers - client queries are routed through the hierarchy	-	XML; aggregation of service descriptions using Bloom-filtered crossed terminals (BCT)	-	Soft state of service announcements	-Multiple shared hierarchies -Shedding server load by spawning on a nearby machine	Authentication of endpoints via digital signatures, privacy via encryption and access rights via capabilities
CSP [1]	Wide-area	Three-tier: proximity, domain and global; hierarchical structure of servers	Servers	Exploring the hierarchy of servers	-	Attribute-value; aggregation of service descriptions using Centroid method	Selection – through context attribute	Frequent updates for intra-domain movements; no updates in the global network	-Hierarchical structure -Minimal overhead associated with unpopular services	-
INS [3]	Dynamic and mobile	Spanning-tree overlay network	Overlay network formed by INRs	-Domain Space Resolver for INR discovery -Passive discovery for services -Early and late binding	-	Name-specifiers, consisting of attributes and values	-	-Replication of service information among resolvers -Soft-state name exchanges and updates	Spawning INRs to candidates resolvers	-
Superstring [2]	Wide-area	DHT overlay network	Resolution hierarchy that matches the service description	Queries are resolved through the routing process	-	Hierarchical description model	-	-	Distribution of the storage and queries among several nodes	SuperstringRep: trust problem addressed by means of a reputation system
JXTA [28]	Wide-area P2P	Virtual network overlay	Rendezvous peers	-Loosely-consistent DHT combined with a limited-range rendezvous walker (used for out-of-sync indices)	-	XML	-	-Index replication among resolvers -Periodic random updates and heartbeat among rendezvous	-Scalable distributed hash indexing - no frequent updates as DHT is loosely consistent	-Certificate authority-based trust models –Credential mechanism for message validation -Encryption -Client and server authentication via digital signatures

Continued on next page

	Type of network	Architecture	Storage of service information	Search methods	Event notification	Service description	Service selection and usage	Fault tolerance and mobility support	Network scalability	Security
CDS [12]	Hundreds or thousands of mobile devices	Hash-based overlay network	Small set of rendezvous points for each content name	Applying the hash function to AV pairs in the query	-	Attribute-value pairs	Selection - query optimization algorithm: node with the smallest response time or the node with smallest database is selected	Names are stored in a soft state fashion	Load Balancing Matrix to share registration and query load	-
GloServ [4]	Local-area and wide-area	Hierarchical	Name Servers manage information of services	UA gets the service hierarchy and queries the name server using RQL	Event subscription through event agent	RDF schemas and descriptions	-	Registration is periodically renewed	Hierarchical structure	Authentication of service providers
One Ring to Rule Them All [6]	Large-scale P2P	Universal ring	Persistent store through a DHT overlay network	-Discovery of contact node at bootstrap -Distributed search engine for finding services	Persistent queries for notification of new services	Textual description	Usage - service implementations stored on the overlay network	File replication over multiple nodes	-Scalable distributed hash indexing -Caching used to avoid overloading nodes	- Authentication with digital signatures
Splendor [34]	Mobile ad-hoc networks – public environments	Four components: clients, services, directories and proxies	Directories	-Active and passive discovery of directories -Clients query directories for services	-	-	-	-Soft state storage of service information -Hard-state storage of services represented by proxies	Aggregation and filtering of service registrations	- authentication - authorization - confidentiality -integrity -non-repudiation
Service Rings [17]	Mobile ad-hoc networks	Overlay structure of hierarchical rings that groups services which are geographically and semantically close	Service Access Points present on every ring	Queries are routed through the ring hierarchy	-	-Languages that support “dist” and “sum” functions - Descriptions are summarised at SAP points		-RingCheck message periodically checking for consistency -Algorithms for broken rings and network partition and reintegration	-Scaling through the hierarchical structure -Algorithms for ring restructuring	-

Continued on next page

	Type of network	Architecture	Storage of service information	Search methods	Event notification	Service description	Service selection and usage	Fault tolerance and mobility support	Network scalability	Security
LANES [16]	Mobile ad-hoc networks	Loosely-coupled lanes overlay structure	Nodes within a lane share the same service information	Discovery messages are transmitted from lane to lane using anycast routing	-	-	-	-Proactive maintenance -Algorithms for node login/logoff, broken connections, network partition and reintegration	Algorithms for optimizing the lanes structure	-
Kozat and Tassiulas [18]	Mobile ad-hoc networks	A subset of the network nodes form a dominating set (virtual backbone)	Distributed directory located on the virtual backbone	Source based multicast tree algorithm	-	-	-	-Periodic service registration -Algorithms for backbone maintenance	Through the use of multiple service broker nodes	-
FRODO [27]	Home appliances -3C(poor), 3D(medium) and 300D(powerful) devices	Centralized	Central acts as a repository for service information	Clients ask the central for services	Subscribers are notified about any change in the state of a service	-	Selection - Devices can request for the best matched service Usage - through XML parsing	-Central election and re-election -Recovering from Central/Backup failure -Soft state for 300D devices, periodic poll of 3D devices	-	-
DEAPspace [21]	Wireless ad-hoc single-hop	P2P	Entire world view on every device	Modified passive search	-	Includes a TTL and the address	-	Nodes broadcast their entire view - rapid convergence	-	-
Konark [14]	Wireless ad-hoc multi-hop	P2P	Service registry on every device	Both active and passive discovery	Clients may subscribe to events	XML	Usage – SOAP	Servers periodically announce their services	-	-
Allia [23]	Ad-hoc	P2P, nodes are organized in alliances	Every node caches advertisements from nodes in its alliance	-Passive discovery -Active discovery by multicast or broadcast service requests	-	-	-	Adjustment of advertisement rates and alliance diameter based on mobility of nodes	-	Policies for access rights and credential verification

Continued on next page

	Type of network	**Architecture**	**Storage of service information**	**Search methods**	**Event notification**	**Service description**	**Service selection and usage**	**Fault tolerance and mobility support**	**Network scalability**	**Security**
GSD [7]	Mobile ad-hoc	P2P	Every node caches advertisements to maximum N hops away (advertisement diameter)	-Passive discovery -Active discovery by selectively forwarding the request to a set of nodes based on semantic information	-	- DAML+OIL -facilitates grouping of services based on service functionality	-	Adjustment of the advertisement time interval and advertisement diameter based on mobility of nodes	-	-
Wu and Zitterbart [32]	Mobile ad-hoc	P2P; based on DSR routing protocol	Every node caches advertisements	Both active and passive discovery	-	-	-	Reactive: -implicit service confirmation -reception of a service poll reply -reception of a "no service" indication	-	-
Varshavsky et al. [29]	Mobile ad-hoc	P2P; based on DSR and DSDV routing protocols	Each nodes maintains a service table	Both active and passive discovery	-	-	Selection - Clients choose the service with the lowest hop count	-Reselection: none, route breaks, any change - Rediscovery: proactive, reactive (route breaks, no route to server, no route to any server)	-	-
Cheng and Marsic [8]	Mobile ad-hoc appliances	P2P; based on ODMRP routing protocol	On every node which is interested in the service	-Active discovery -Passive discovery in the bootstrap	Nodes subscribe for receiving service updates	-	Usage - service invocation through mobile objects	Implicit through active discovery	-	-