Service Discovery for Mobile Ad Hoc Networks: A Survey of Issues and Techniques[♦]

Christopher N. Ververidis[¤] and George C. Polyzos[❖], *Member, IEEE*

***Abstract*- This paper surveys research in service advertising, discovery and selection for mobile ad hoc networks (MANETs) and related issues. We include a categorization of service discovery architectures for MANETs and their modes of operation, presenting their merits and drawbacks. We pay particular attention to cross-layer service discovery a special class of efficient service discovery approaches for MANETs. We also present security issues and discuss service description options, service selection mechanisms and service state maintenance techniques. We conclude with a summary, an outlook and directions for future research in this area.**

***Index Terms*— Service Discovery, Advertisement, Selection, Description, Cross layer**

I. Introduction

MOBILE Ad Hoc Networks (MANETs) are networks comprised of mobile nodes (e.g. portable computers, PDAs etc.) equipped with wireless interfaces and communicating with each other without relying on any infrastructure. In these networks each mobile node may act as a client, a server and a router. MANETs have emerged to fulfill the need for communication of mobile users in locations where deploying a network infrastructure is impossible, or too expensive, or simply is not available at that time. Characteristic scenarios for MANETs are disaster relief operations, battlefields and locations where infrastructure-based WLAN coverage (also called hotspots) is not provided and wireless WANs (e.g. GPRS/UMTS) are too expensive to use or too slow.

Most of the research on MANETs has focused on issues dealing with the connectivity between mobile nodes in order to cope with the dynamism of such networks and the arising problems thereof. This dynamism is due to the mobility of nodes, the wireless channel's adverse conditions and the energy limitations of mobile nodes, all of which lead to frequent disconnections and/or node failures. These research efforts have led to the creation of a sound technical basis for dealing with the aforementioned problems regarding node connectivity in MANETs (mainly through routing protocols, link layer protocols etc.)

However, solving the problems of connectivity alone is not sufficient for the adoption of MANETs. Since their basic role is to allow mobile users to exchange data and use each other's services, there is also a need for architectures, mechanisms and protocols for Service Discovery. Service Discovery is defined as a process allowing networked entities to:

- Advertise their services.
- Query about services provided by other entities.
- Select the most appropriately matched services.
- Invoke the services.

Service Discovery has been mainly addressed in the context of wired networks in the past. However, in the context of MANETs new challenges arise:

- Node mobility, affecting service availability.
- Frequent disconnections of the server, the client or intermediate nodes breaking or changing the path and the service selection parameters.
- Channel variability, leading to significant communication characteristics variability (data rate, delay etc.)

Despite the existence of a number of survey papers regarding service discovery protocols ([48], [89], [90], [91]) we believe that a comprehensive overview of techniques and open issues for service discovery in MANETs would be useful. It is the purpose of this paper to provide a comprehensive review on the state of the art regarding service discovery approaches for MANETs. In the following we will try to make a categorization of service discovery approaches according to the mechanisms they utilize and their features. We will also highlight features that are not fully developed yet and require further research.

The structure of the remainder of the paper is as follows: Section II describes the basic service discovery architectures, Section III presents the possible modes for service discovery, Section IV highlights approaches based on cross-layer optimizations, Section V discusses special features for service discovery such as service description options, service selection mechanisms and service state maintenance techniques, Section VI comments on security, trust and

[♦] This research has been supported in part by the project Autonomic Network Architecture (ANA, #IST-27489), which is funded by the IST FET Program of the European Commission.

[¤] C. N. Ververidis is with the Computer Science Department, Athens University of Economics and Business, Athens, 10434 GREECE (phone: +30-210-8203646; fax: +30-210-8203860; e-mail: chris@aueb.gr).

[❖] G. C. Polyzos is with the Computer Science Department, Athens University of Economics and Business, Athens, 10434 GREECE (e-mail: polyzos@aueb.gr).

privacy issues for service discovery, Section VII identifies areas requiring further research and Section VIII concludes the paper.

Before continuing to the presentation of research results regarding service discovery it is worth briefly presenting the pioneering service discovery approaches developed and adopted by the industry, namely Jini[1] [1], Salutation[2] [2], UPnP[3] [3], Bluetooth SDP[4] [5], SLP[5] [6] and Bonjour[6] [7].

Jini

Jini is a service discovery architecture specifying how service discovery and service invocation is to be performed among Java-enabled devices (a Java Virtual Machine is mandatory). A central component of a Jini deployment is a Lookup server. Lookup servers act as directories. They store services published by service providers and also they reply to client queries. Lookup servers announce their presence in response to requests multicasted by service providers or clients. Service providers register their services with Lookup servers by sending service objects along with their attributes. Service objects are actually proxies written in Java and serve as interfaces for clients to access a remote service. Clients receive those proxies (usually RMI stubs) by Lookup servers upon successful match of their requests. Requests may include the type of the requested service as well as other attributes. Jini also supports leases, which means that services are registered for a specific amount of time and if they do not get updated they are erased from Lookup servers. Another characteristic of Jini is that through Java remote events, clients can be notified upon changes in the status of a remote service. Finally, Jini provides security through the Jini Security Framework (see section VI).

Salutation

Salutation was primarily designed for home and enterprise environments. Its architecture allows devices, services and applications to advertise their capabilities, discover and access each other. Capabilities are expressed as attribute sets. A basic component of the architecture is Salutation Managers (SLM), who are responsible for storing attribute sets. Every device has a local SLM with descriptions of its own services. However SLMs at different devices communicate with each other via the Salutation Manager Protocol in order to discover services available in other devices. Communication protocol independence is achieved through a transport independent layer between an SLM and the Salutation Transport Manager (TM), which implements the transport functionality. One SLM may have many TMs in order to operate over different network technologies (e.g. IR, Bluetooth etc.). Service availability can be checked by setting a local SLM to periodically query a remote SLM about the needed service. Regarding security Salutation supports only password-based authentication. For small footprint devices a less demanding version of Salutation, Salutation-Lite [58], has also been developed.

UPnP

Like Salutation UPnP was also proposed for use in small office and home environments and mainly targets device and service discovery. Through UPnP, devices first advertise their presence in a network and upon request they also present their capabilities using XML for service descriptions. The basic entities considered in UPnP are control points (acting as service directories) and devices. Control points are optional so if there is no control point, devices may also listen to service advertisements directly. Service discovery in UPnP is based on the Simple Service Discovery Protocol (SSDS [59]), which operates using HTTP over multicast and unicast UDP. It is worth noting that UPnP can be deployed only over TCP/IP networks and generally operates better over reliable networks. A special feature is that through AutoIP, UPnP devices automatically receive an IP address even when a DHCP server is absent. Unfortunately due to the extensive use of multicasting (multicasting is used both for service advertisements and service requests) UPnP cannot scale well. Also it does not support attribute-based querying for services. UPnP provides many mechanisms for securing service discovery and access through UPnP Security [50] (see section VI).

Bluetooth SDP

Bluetooth SDP is a service discovery protocol for Bluetooth enabled devices. Bluetooth SDP addresses only service discovery and does not address service advertising, service caching in registries or service access. Every service is described by a service record consisting of a set of attribute-value pairs each of which describes a service characteristic. Bluetooth SDP defines two methods for discovering services, namely ‘service searching’ and ‘service browsing’. With the first method a client formulates a query containing desired service attributes and those are matched against service records at the provider and the result is returned. The latter method allows a client to send a generic query and get a list of all services of a specific provider. We should note that Bluetooth SDP supports only 1-hop discovery and hence its discovery capability is limited to the immediate proximity of a

[1] Sun Microsystems.

[2] **Salutation Consortium members:** Canon Inc., Consumer Electronics Association (CEA), Continental Automated Buildings Association (CABA), Fuji Xerox Co., Ltd., Hewlett Packard, Infrared Data Association (IrDA), Institute of Certified E-Commerce Consultants (ICECC), International Business Machines, Konica Minolta Holdings, Inc., Kyocera Mita Corporation, National Institute of Standards and Technology (NIST), Oki Data Corp., Ricoh Company, Ltd., Seiko Epson Corp, SISCO, Sun Microsystems.

[3] **UPnP Forum’s Steering Committee members:** Broadcom Corporation, Cable Television Laboratories, Inc., Intel Corporation, LG Electronics, Microsoft Corporation, Motorola, Inc., Nokia Corporation, Panasonic, Philips Consumer Electronics, Pioneer Research, Ricoh Company, Ltd., Samsung Electronics Company, Ltd., Siemens AG, Sony Corporation, Thomson Inc.

[4] **Bluetooth Special Interest Group members:** Agere Systems, Ericsson Technology Licensing AB, Intel Corporation, Lenovo, Microsoft Corporation, Motorola, Inc., Nokia, and Toshiba Corporation.

[5] Internet Engineering Task Force (IETF) standard.

[6] Apple Inc.

device.

SLP

The Service Location Protocol (SLP) is an IETF standard and has been embedded in many commercial products (by Hewlett Packard, IBM etc.). SLP addresses only service discovery and leaves service invocation unspecified. Service descriptions consist of unique URLs (for locating the service in the network) and a set of attribute-value pairs. Clients may query for services using their type or some combination of their attributes utilizing SLP's capability of substring matching. SLP also allows grouping services in scopes. Service browsing is also allowed if a client requests to see all available services. SLP can work totally distributed using only User Agents (UA) on client devices and Service Agents (SA) on service providers. Communication among them takes place through multicasting. If directories exist, then they are represented by Directory Agents (DA). In the directory-based operation of SLP when a DA enters the network it multicasts a beacon and any SA that hears it must register its service to this DA. UAs that hear this message unicast their queries to the DA. If no DA is present UAs multicast queries and all receiving SAs with matching descriptions respond using unicasts. Regarding security, SLP provides only a PKI-based mechanism for signing service advertisements (see section VI). Finally, we should note that a more lightweight version of SLP, called SLPManet, was proposed in [79], excluding features like optional SLP messages, DAs and authentication.

Bonjour

Bonjour is a technology developed by Apple to provide service and device discovery among computers, electronic appliances and other networked devices (e.g. printers, faxes etc.). Bonjour runs over the IP protocol and also has the capability of automatically assigning IP addresses to networked devices, even without the help of a DHCP server. Bonjour's core is a service discovery protocol entirely based on the Multicast DNS Service Discovery – MDNS-SD. Actually MDNS-SD extends MDNS [59] so that hosts in an ad hoc network can resolve, in addition to host names also service names to IP addresses without relying on DNS servers. In MDNS-SD clients multicast their DNS-like queries specifying (defined in DNS-SD [60]) the service type they are looking for, the domain where the service resides and the preferred communication protocol. Service providers respond to those queries by DNS service records. However, a new provider coming into the network may make a multicast announcement so that other devices become aware of its presence. The service records are cached on client devices for a limited time and if not updated (by querying again) they are deleted. However, multicasting everything creates a significant amount of traffic. Bonjour tries to address this by employing "exponential back-off" for increasing the gap between queries and announcements in order to minimize traffic while keeping the user's view as fresh as possible [7].

All the aforementioned approaches were mainly designed for administered networks (even if ad hoc), some requiring fixed-well known directories, others making extensive use of broadcasting and multicasting (hence not scaling for large ad hoc networks) and others not supporting mobility. However, they have served as a solid base and source of inspiration for developing new protocols oriented to pure ad hoc environments.

II. Service Discovery Architectures

Regarding service information dissemination, there are three basic architectures that a service discovery approach may adopt (see Fig.1). We will present each of them and refer to representative approaches found in the literature:

A. *Directory-based Architectures*

In this architecture there are three possible roles for a mobile node. A node can be a server (service provider, offering one or more services to other nodes), a client (service requestor, requesting services from other nodes) or a service directory (facilitating communication between providers and clients). Service providers register their services to service directories and service requestors are informed about the available services in the network only through these directory nodes.

A directory can be implemented as *centralized* (hosted by a single node) or can be *distributed* among several nodes. Centralized approaches were primarily adopted by service discovery protocols in wired networks or in wireless local area networks where one or more fixed hosts take up the role of a directory (e.g. UDDI [4]). A simple centralized directory, however, is not a good solution for an ad hoc network, since no node is always reachable. A centralized directory also represents a single point of failure and is not well suited for such volatile environments. *Scalability* is also another problem, since in MANETs the nodes are resource-poor and a single node acting as a directory would not be able to handle responses for a large number of nodes. Distributed directories are thus more suitable for MANETS.

A basic question is whether global service discovery is to be provided (i.e. to be possible for every node to learn and invoke any service provided in the ad hoc network). One approach is to use full *replication* for directory nodes in order for every directory to store all services available in the MANET, irrespectively of their location.

A classic distributed directory approach is Jini, where a few nodes, named Lookup servers, act as directories. However, there is no communication among Lookup servers and it is at the discretion of service providers to publish their service to more than one directory node and keep them updated. In this case, that automatic replication is not provided, a service may be known only locally, around the directory node that originally hosts it and remote MANET nodes will not be able to easily discover it. Global discovery is hence not supported, since services are advertised only in the area where the Lookup servers reside.

In more elaborate distributed directory approaches, nodes acting as directories are in constant communication with each other to disseminate and also replicate service information among them. Such approaches are based on protocols that create and maintain a *backbone* of directory-enabled nodes. For example in [28] a backbone of directory nodes is formed using a Minimum Dominating Set algorithm. Servers advertise their services to one or more members of the backbone. However, despite the fact that service replication is not inherently provided, global discovery is possible since backbone members disseminate to each other service discovery requests that could not be satisfied locally. This way a service requestor and a service provider connected to the opposite edges of the formed backbone can still discover each other and communicate. A better way to forward requests to neighboring backbone members (instead of doing it randomly) was proposed in [30]. There, backbone members frequently exchange directory profiles guaranteeing that service requests are forwarded to nodes that are likely to cache the description of the requested service.

An alternative to the aforementioned backbone based approaches for implementing distributed directories are *clustering approaches*. A representative approach is "Service Rings" [22]. In Service Rings a number of clusters are formed. Each cluster (called a ring) of service providers is formed based on physical proximity and semantic proximity of the descriptions of provided services. Every ring has its own Service Access Point (SAP), which is responsible for handling service registrations and service requests (operating as a directory). SAPs also communicate with each other and exchange summaries about all the services they are aware of in their own ring. This way higher-level rings are also formed iteratively. Global discovery is possible since if a node's request cannot be satisfied by its local SAP, then this SAP forwards the request to neighboring SAPs (and eventually to higher level SAPs) that are possibly capable of satisfying the request based on the service summaries they have previously sent. A similar approach is also adopted in [16] with the difference that the hierarchy of clusters is strictly dependent on a common service ontology. At the bottom level of this hierarchy, clusters group devices offering services described by the same leaf term of the ontology and being within radio range of each other. Moving up the hierarchy, every level consists of groups of clusters of their respective lower level. The higher the level, the more general the semantic descriptions become (always in alignment with the generalization of categories performed as we move up the ontology tree). In [11] each cluster groups nodes with similar mobility patterns. In each cluster one of the nodes (called clusterhead) stays awake permanently and answers discovery requests. The rest of the nodes periodically wake up to provide the actual services and also to inform the clusterhead about their presence and services. The clusterheads are re-elected periodically to avoid draining a single node's battery.

Another solution to the global discovery problem, when replication is not provided (either by servers or directories), is to use *Distributed Hash Table* based techniques along with *location information*. Such approaches are described in [21] and [87]. The network topology is divided into geographical regions, where each region is responsible for a set of keys representing the services of interest. Each key is mapped to a region based on a hash-table like mapping scheme. A few elected nodes within each region are responsible for storing these keys (in [87] all nodes inside the region store these keys), thus acting as directories. Global discovery is possible since a node requesting a service, uses the same hashing function (as the one used by service providers) and finds the directory-location where its description is stored. The service request is then routed, using this location information, towards that directory. Location information is also used in [32] for creating clusters of nodes based on physical proximity. Every cluster has a gateway, which is responsible for handling routing and discovery requests and for storing service descriptions from nodes located within its region. Inter-gateway request forwarding is also possible for global service discovery and is done on a region-covering basis (actually requests are routed to neighboring regions, where another gateway will be present and will try to answer them). This approach however differs from other backbone or cluster based approaches in the sense that cluster leaders (gateways in this case) are elected or de-selected automatically based on location information and do not need to keep contact with each other.

The *election of nodes* for taking the role of a directory or for participating in a directory structure (backbone) is a very crucial issue for a directory-based service discovery approach. In [16] and [22] an election mechanism is not provided, while in [21] a directory node election is performed randomly. In more elaborate approaches, like [28] and [30], criteria like average packet loss rate, effective degree (for connectivity to neighbors) and capacity are used to choose one among a set of candidate nodes.

It is important to note here that directory based approaches imply additional communication costs in the network for maintaining the directory structure and also for exchanging data among the members of a distributed directory for preserving service consistency and for replicating service information. If maintenance and consistency procedures are not well tuned, then either too much traffic will be generated, causing congestion and hence rendering the whole MANET useless, or inconsistencies in service information and directory structure (due to insufficient updating) will degrade the performance of the service discovery process.

B. Directory-less Architectures

This type of architecture differs from the previous in that there are no service directories to mediate communication between service providers and service requestors. It is much simpler from directory-based architectures since there is no need for directory selection and maintenance mechanisms. Service providers broadcast service advertisements and service requestors broadcast service requests. Both processes may take place at the same time in the network. In the early approaches of this type only servers could reply to service requests. Later intermediate nodes (located along the paths between servers and requestors) were also allowed to reply to service requests based on the information they had cached locally by overhearing past server replies.

A basic problem in those non directory-based approaches is how to determine the frequency of service advertisements in order to reduce network load and avoid redundant transmissions. *Scheduling and prioritization* was one of the first techniques proposed to deal with the problem. For example in [9] servers periodically broadcast service advertisements to their 1-hop neighbors. These advertisements contain services provided locally by the sending node and also services that the sending node has learned from its neighbors, which are then stored as service records in the receiving node's local cache. Servers, whose services are about to expire[7] or have expired, are assigned a greater probability to make the next broadcast. An exponential back-off algorithm regulates the periodicity of broadcasts depending on server priority and changes in the network (i.e. new servers). A similar approach is employed in [82] where providers periodically advertise their services along with services that they became aware of by neighboring service providers. A provider postpones advertising its services and back-offs for a fixed amount of time if it receives and advertisement that contains its own services (this means that nodes in its vicinity are aware of its service and up-to-date). Very close to this concept is also the mechanism proposed in [83] where a provider listens to other's broadcasts and when it is its turn to broadcast, it only broadcasts service information (if any) that has not expired and has not been seen recently in previous broadcasts.

Another way for lowering the load imposed on the network by broadcasting for service discovery is to use *multicasting*. In [10] servers multicast their advertisements on a fixed multicast group and so do service requestors. In contrast to [9] where 1-hop discovery is performed, in [10] the messages cover the whole network.

Covering the whole network using either broadcasting or even multicasting techniques is very costly. This is why many approaches use various other techniques:

- *Advertisement range bounding/scoping*
- *Selective, probabilistic and intelligent (advertisement/request) forwarding*
- *Peer-to-peer (P2P) information caching*
- *Intermediate node responding to service requests.*

Several approaches taking advantage of these ideas and techniques are described next.

Many approaches use an advertisement range measured in number of hops specifying when the advertisement message will be dropped. In the Group-based Service Discovery (GSD) protocol [18] and the Alliance-based Service Discovery (Allia) protocol [57] such a technique is adopted. However, in order to allow most of the nodes in the network to eventually become aware of the advertised services, these two approaches also include a technique called *peer-to-peer (P2P) information caching* for nodes to merge services heard by others and re-advertise them (using again a range) along with their own services. Eventually, most nodes will become aware of all services in the network, but at a lower cost since service merging is performed.

The two approaches mentioned above also employ *selective forwarding* of service requests to further reduce the load of service discovery. Selective forwarding means that a node receiving a service request that it cannot fulfill will forward the request only to those of its neighbors that are known to host the requested service, or similar services. Besides selective forwarding [80] and [81] propose that the overhead of GSD can be further reduced by an additional mechanism called Broadcast Simulated Unicast (BSU). Instead of forwarding the same query in unicasted packets towards selected neighbors, with BSU the message is forwarded once using broadcast. Only the selected neighbors will further process this packet since it contains a list with the intended recipients. If a neighboring node receives such a packet and does not find itself in the receiver's list, it will just discard the packet. However, a significant amount of bandwidth will have been saved.

The P2P information caching technique is also used in [29] along with *probabilistic forwarding* instead of selective forwarding. In this case a node receiving a service request that it cannot fulfill, forwards it with a probability that decreases with the number of hops that the request has already traveled.

Intelligent forwarding can also be used for spreading service advertisements, as done in [35]. In [35] every node continuously monitors its 2-hop neighborhood. In order to avoid duplicate packet forwarding and also to cover every node, each server initially sends its advertisements only to those nodes in its 1-hop neighborhood (called brokers) through which all its 2-hop neighbors can be reached. In the next advertising round new brokers will be formed for the 2-hop neighbors of the originating server thereby expanding the service coverage in the same way (by forwarding the advertisement only to a subset of their 1-hop neighbors through which all their 2-hop neighbors can be reached). Costly broadcasting is hence replaced by a few unicasts in every advertising round.

Another way to reduce the load imposed by service discovery requests and advertisements is to allow *intermediate nodes to respond to service requests*. Intermediate nodes may have been informed about the existence of some services either by receiving and forwarding service advertisements or because they themselves have requested these services in the past. Hence a service request may not need to travel all the way to the service provider, since it can be answered by an intermediate node located closer to the service requestor.

In [17] intermediate nodes are allowed to answer service

[7] Each service record has a Time To Live (TTL) field. This TTL continuously decreases with time until it reaches zero (except if a new advertisement is received and the TTL is refreshed). When the TTL becomes zero the corresponding service is considered expired and its record should be deleted from the node's cache.

requests. However in order not to decrease the number of discovered services the authors propose that intermediate nodes must be informed of all the services matching the issued requests. This is because dropping requests at intermediate nodes that already know one out of many matching services may decrease the service discoverability of the protocol. It is proposed that when answers come to a service requestor from different servers and different paths, intermediate nodes and servers along those paths are updated to become aware of all the services that were returned to the requestor. Thus, when they receive another request for the same kind of service from another node, any server or intermediate node will be able to reply with all the matching services they became aware of by informing each other in previous requests.

Finally in order to totally avoid broadcasting or multicasting and the associated costs, the use of *location information* has also been proposed in [23] for sending service advertisements and service requests. In this protocol, servers periodically send their advertisements along cross-shaped trajectories. At each node in the trajectory, a backwards pointer is set up establishing paths leading to the service provider. Any service requestor need simply send a query along a path that intersects with the advertisement path. Requests are answered by nodes at the intersection (intermediates) of the advertising and requesting trajectories.

C. Hybrid Architectures

In these architectures service providers register their services with service directories if they locate any in their vicinity (if not they simply broadcast service advertisements). Service requestors send their queries to the service directories they are aware of. If they are not aware of any service directory, they broadcast them to the whole network. Service replies may come both from service providers and service directories.

D. Comparisons

Despite the multitude of publications on each of the service discovery architectures described in this section, researchers have not come into a general consensus on which architecture is better. The basic criteria for evaluating the effectiveness of service discovery architectures are *service availability*, *messaging overhead* and *latency*. The reason making it difficult to come to a general conclusion on which architecture is more suitable for a MANET is that it depends on many factors, some of which relate to the MANET's characteristics (e.g. server and client density, node mobility and service request frequency) and others to tunable parameters of the discovery architecture employed (e.g. flooding / broadcasting scopes, directory node density, service registration and announcement frequency). For example, for a MANET with a high degree of mobility and a low service request frequency a distributed architecture without caching could prove to be more efficient than a directory-based architecture, since the

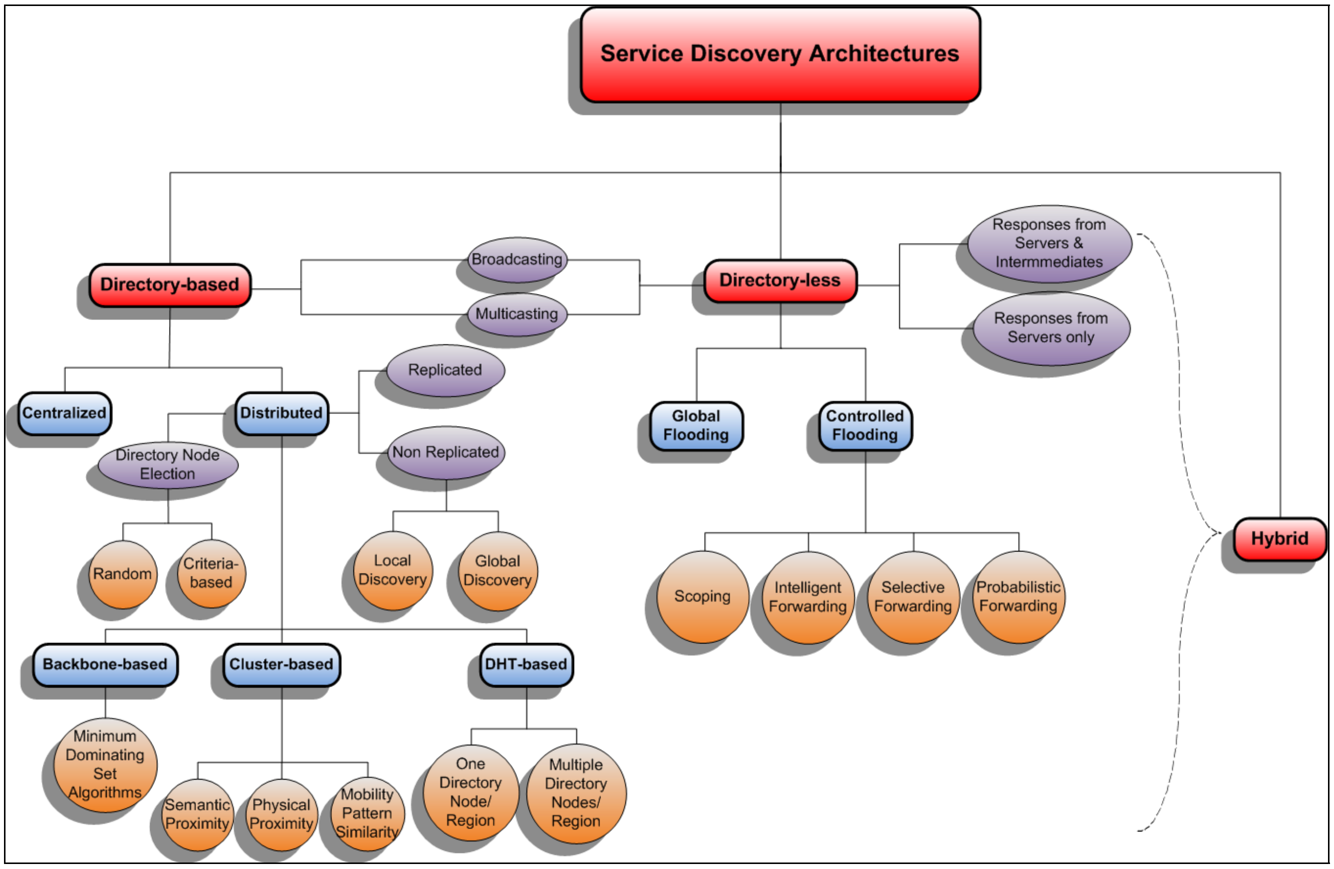

Figure 1: Service Discovery Architectures

latter would either suffer from stale service information in directories, or would demand much overhead for maintaining service information integrity and coping with mobility. However, if the same MANET of the previous example faced a very high service request frequency a directory-based architecture could be more efficient. In this case, a directory-less architecture would demand that clients frequently flood the whole network with their queries. This traffic would most probably outweigh the traffic created in a directory-based architecture for maintaining consistent directories and for unicasting queries to directories only, instead of flooding them to the whole network. The above would hold in the general case where both architectures' parameters are tuned similarly. However, there exist certain values for the tunable parameters that could affect an architecture's performance so severely, that a seemingly better/matching architecture could prove to be worse than the other.

Generally, none of the three architectures can outperform the other two in all of the above mentioned performance criteria. Even the underlying routing protocol (especially when integrated with the service discovery process) may have an impact on the performance of a service discovery architecture as shown in [36] and [31]. In [36] simulations show that in proactively routed MANETs the hybrid architecture outperforms in terms of service availability the other two architectures. However, the directory-less architecture outperforms the other two architectures in messaging overhead. A more recent work [31] for reactively routed MANETs, however, shows by simulations that a directory-less architecture may outperform a hybrid one, both in terms of higher service availability and lower message overhead, while having almost the same delays.

It would be interesting to have a flexible/autonomic architecture with the ability to self tune its parameters and change operational modes from directory-based to directory-less or hybrid, based on a MANET's dynamic characteristics.

III. Service Discovery Modes

Irrespectively of the service discovery architecture there are three possible ways/modes of operation for a service requestor to acquire service information:

A. *Reactive*

In this mode a service requestor issues a query in an on demand basis to directory nodes or directly to service providers. There are many variations for this mode, some of which have been discussed in the service discovery architectures section of this paper. To name a few options for service requestors, they may choose to set a limited TTL so that they do not flood the whole network when there are no directories. They may expand their search step-by-step by gradually increasing the hops that a service request is allowed to travel. They may utilize mechanisms to selectively forward their requests to specific neighbors only, instead of sending them to every neighbor. They also may unicast, multicast or broadcast a query to one or more directories or to one or more servers.

B. *Proactive*

In this mode service providers advertise their services (either to service directories or directly to potential service requestors) on discrete time intervals. The same holds for advertisements originating from directory nodes. Servers and directories have also the option to use ranges for the advertisements instead of flooding the whole network. A basic tunable parameter is how frequently those advertisements should be sent, since it greatly depends on the level of dynamism of the MANET (mobility, failures, congestion).

C. *Hybrid*

In this mode both proactive and reactive communication between service requestors, service providers and service directories is possible. For example servers may proactively advertise their services to service directories, but clients may issue requests to service directories only reactively (on demand). As explained in [24], several strategies can be employed by clients and servers in order to discover services. For example in a "greedy" strategy all servers may advertise services to all nodes and all clients query all nodes in the network in order to discover services, while in a "conservative" strategy servers may advertise services to a random set of nodes and clients may also query only a random set of nodes. Other more complex strategies include incremental increase of the advertisement and querying sets and memorizing previously queried nodes in order to avoid querying them again in next rounds, for the case that a service has not been discovered in the previous round. As expected, authors conclude that "greedy" strategies offer higher success rates and lower delays than "conservative" strategies, but produce much higher overheads. However, they also note that depending on factors such as success rate requirements, delay tolerance, overhead tolerance, node memory constraints, network dynamism (expressed as mobility and underlying routing protocol - proactive or reactive) the preferred strategy is different.

D. *Comparisons*

Mohan et al. in [33] present a simulation analysis of the proactive and reactive modes in their simplest form, which involves global flooding. According to these results the proactive mode outperforms in terms of latency and overhead the reactive mode when the number of servers is significantly lower than the number of clients. The opposite happens when the available servers are significantly more than the clients in the network. A hybrid scheme is proposed to give on the average better results in terms of overhead and latency for most combinations of number of servers to number of clients. This hybrid scheme is enhanced by a mechanism allowing servers (respectively clients) to determine network congestion before deciding to send an advertisement (respectively a query). If the congestion is over a given threshold, the senders (either clients or servers) exponentially back-off in order to avoid congesting the network further, causing delays, retransmissions etc. However, careful selection of this threshold is difficult in such a dynamic environment as a MANET.

In a MANET with a proportion of clients to servers close to 50% the preferred approach depends on the actual demand for discovering services. It is intuitive that in such MANETs, if service discovery requests are rather rare, a reactive approach would be more efficient (at least in terms of control overhead) than a proactive or hybrid approach. Of course in cases where service discovery is performed frequently, a proactive scheme would prove to be preferable (provided that services are advertised in appropriate time intervals, matching the demand). This is also backed by experimental results in [46], where the authors provide a thorough analysis (both theoretical and experimental), on the performance of reactive and proactive service discovery modes investigating the impact of several factors (mobility, traffic patterns, message aggregation, use of caching). They conclude that the actual service context is what determines which mode is most efficient and that a hybrid mechanism able to adapt to service demand is the preferred choice. They especially investigate the impact of the underlying routing protocol and also its coupling with the service discovery process on the performance of each service discovery mode. This coupling leads to a special case of service discovery protocols, namely the cross layer service discovery protocols, which we examine in the following section.

IV. Cross Layer Service Discovery

In contrast to traditional application layer based service discovery, there are many approaches that employ *cross layer* techniques in order to benefit from information available at lower layers of the protocol stack. Most of these approaches are based on integrating the routing process with the service discovery process. The motivation for integrating routing and service discovery stems from the fact that any service discovery protocol implemented above the routing layer will always require the existence of some kind of routing protocol for its own use. Hence, two message-producing processes must coexist: the first one communicates service information among service providers and service requestors; the second one communicates routing information among them. As a result, a node is forced to perform multiple times the battery-draining operation of receiving and transmitting (control) packets. Cross layer service discovery exploits the capability of acquiring service information along with routing information (from the same message) by piggybacking service information onto routing messages. This way, redundant transmissions of service discovery packets at the application layer are avoided and energy is saved. Henceforth, we will refer to those cross layer service discovery protocols as integrated protocols.

The idea of providing routing layer support for service discovery was first introduced by Koodli and Perkins in [13]. They argue that for proactively routed MANETs, a service reply extension added to topology updating routing messages is enough for providing both service discovery and route discovery concurrently. In reactively (or on-demand) routed MANETs, the service discovery process follows the traditional route discovery process by using its message formats for route requests (RREQ packets) and route replies (RREP packets) extended to carry also a service request or a service reply respectively. In [41] the authors have extended the Ad hoc On-Demand Distance Vector (AODV) routing protocol with service discovery functionality and have experimentally compared it with NOM [15] (a pure application based service discovery protocol). Their findings show that the integrated protocol produces 30% to 50% less control overhead and has 2 to 7 times lower service acquisition latency than the application layer based protocol (depending on simulation parameters). Authors of [68] and [69] have provided additional extensions to the integrated AODV protocol to also support QoS aware service selection (see Section V. B).

In [61] AODV and Dynamic Source Routing (DSR) are extended (named SD-AODV and SD-DSR) to support service discovery and are compared in terms of traffic overhead against an application layer service discovery protocol based on the Service Location Protocol (SLP) [6] and also against a protocol with global knowledge. The global knowledge protocol uses an oracle to determine which service providers are available in the network and to select the closest one for communication. Once again it is experimentally shown that both integrated protocols outperform the SLP-based approach under any node density, request frequency and speed. SD-DSR is also shown to be more efficient than SD-AODV, since it allows its nodes to update their routing information and maintain a consistent view of routes and services by overhearing other nodes' transmissions. Regarding the global knowledge application layer service discovery protocol, when using AODV at the network layer, SD-AODV presents comparable performance, but requires that services are cached for short periods of time so that stale service information (e.g. due to node movement) is erased and service provider selection is nearly optimal (i.e. the closest server must be selected). SD-DSR compared to the global knowledge protocol with DSR at the network layer performs slightly better or worse depending on network conditions. The global knowledge approach always tries to contact the closest provider. If the path to the closest provider is unreliable the global knowledge will keep trying several times before choosing the second nearest provider. In DSR a provider is tried only once and if there is no response the second nearest provider is contacted.

A similar study on DSR and AODV integrated protocols was conducted in [62], where authors propose the use of a module at the link layer, which is responsible for assembling and disassembling packets, to embed service information from an application layer service discovery protocol and at the same time routing information from a network layer protocol. This way routing protocols are not extended or modified in any way. The authors experimentally show that cross layer service discovery using the link layer module and AODV (respectively DSR) produces about 15% (respectively 90%) less traffic than classic service discovery without the module and using AODV (respectively DSR) at the network layer.

In [25] in order to compare a *reactive* routing and service discovery protocol and a *proactive* routing and service discovery protocol, DSR and the Destination-Sequenced Distance Vector protocol (DSDV) are extended to provide service discovery functionality. Those approaches are compared against SLP, implemented at the application layer. The extended DSR protocol proves to have the least messaging overhead among the three, with second best the extended DSDV protocol. DSDV is not the only proactive routing protocol extended with service discovery functionality. In [63] and [64] researchers have also extended the Optimized Link State Routing (OLSR) proactive routing protocol to support service discovery, but no comparisons with other integrated or application layer protocols are presented.

Service discovery extensions have also been introduced in *hybrid routing protocols* in [12] and [40] where energy consumption is also considered. In hybrid routing protocols each node proactively advertises the routes and services it is aware of by sending control messages to its neighbors up to a fixed number of hops away (this is called the node's zone). Information for routes or services outside this zone may be gathered only upon request (reactively). Experimental results show that those integrated protocols also clearly outperform application layer based service discovery protocols demonstrating energy savings of 30% to 95%.

Comparisons between a hybrid integrated protocol and an on-demand integrated protocol (based on the AODV routing protocol) are given in [27], [65] and [66]. The proposed hybrid integrated protocol in [27] resembles the one proposed in [40] but adds the functionality of dynamically adjusting the size of a node's zone depending on service usage frequency. The higher the popularity of a node's services, the larger the zone where proactive announcements should be propagated. By simulations it is shown that the hybrid protocol has 25% less control overhead and is 2 to 10 times faster in service acquisition than the on-demand AODV-based integrated protocol. In [65] the same concept is followed, with the difference that the zone size determination is based on the transmission power/range selected by a node. Paper [66] presents another hybrid integrated protocol (SPIZ), where an autonomous and adaptive zone radius determination mechanism (based on multiple criteria such as call rate, mobility, service popularity etc.) is provided. SPIZ is compared against an AODV-based integrated protocol, a ZRP-based integrated protocol and application layer based service discovery protocols implementing pull and push methods for service discovery. SPIZ saves 20% to 65% of the control traffic for service discovery when compared to those approaches, with ZRP being the second best and AODV following. Those savings are attributed to SPIZ's capability to adapting to the network's characteristics using the zone determination mechanism (e.g. larger zones are selected for high call rates and low mobility, but also providers with popular services operate more efficiently with larger zones). The performance of the pull based service discovery protocol is worse than all aforementioned integrated protocols and even worst for the push based service discovery protocol.

Another category of routing protocols, namely multicast routing protocols has been used for service discovery. The authors in [19] and [20] extend the On-Demand Multicast Routing Protocol – (ODMRP) to support service discovery functionality. According to this approach each server and its possible clients form a multicast group. Each server multicasts an advertisement encapsulated in an ODMRP join query packet. Any client, interested in the advertised services, stores the advertisement and sends a service awareness reply encapsulated in an ODMRP join reply packet. Once the multicast group between a server and all interested clients has been formed, the server will re-send advertisements only if its service changes. Otherwise it waits for explicit queries from clients. In [28] authors show that an AODV-based integrated protocol performing service discovery using anycasting[8] is much better in terms of delay and control packet overhead compared to an ODMRP-based integrated protocol constructing requestor-based multicast trees for performing service discovery. However, the ODMRP-based integrated protocol has a significantly higher service hit ratio especially in highly mobile environments.

A more radical approach is adopted in [67], where the authors do not integrate service discovery with a well-known protocol, but build their own multicast routing protocol named HESED, which also supports service discovery. In HESED multicast routing is used both for service requests and service responses. Intermediate nodes locally cache service reply information but do not use it to reply to requests. When requesting a service, a client first searches its local cache, and if it finds a matching service record, it calculates the probability that the path to the service is still valid. The routing part of HESED uses a beaconing mechanism allowing nodes to know their 2-hop neighbors. Depending on the change rate of their neighbors, nodes calculate the probability that a route to a server is valid. The proposed protocol shows significant gains (up to 80% less delay and control packet overhead) over a flooding based protocol implemented at the

[8] In anycasting, a virtual server node is defined that is uniquely identified by the IP anycast address, for which only the actual server nodes have routing entries. In the anycast-1 service discovery scheme, every node receives the service advertisements from the different service instances and stores only one single entry in its routing table, the one towards the neighbor which sent the advertisement with the smallest hop count value. Therefore, a query is always sent to the neighbor that is the closest to any service instance. The major drawback of this simple anycast implementation is its lack of robustness. Due to its single entry per service, it fails to deliver a query when any of the links on the path to the service becomes unavailable.

application layer. This is especially true for high-density scenarios, since HESED employs an intelligent forwarding mechanism similar to the one proposed in [35].

Finally, another benefit provided by cross layer approaches is the exploitation of routing information for restoring service sessions, or making handovers from provider to provider. This idea was implemented in [77], where the authors integrated the GSD protocol with routing. The integrated protocol additionally provides automatic redirection to another service provider when the route to the selected service provider fails. Comparisons of the integrated protocol with the simple GSD protocol over AODV showed increased service success ratio of up to 50% for the integrated protocol.

All the aforementioned approaches target higher efficiency in terms of energy consumption, capacity-scalability and control packet overhead. However, they share a common disadvantage: they require that the logical separation among protocol stack layers is broken (in order for application layer logic to be integrated into routing or link layer logic) and as a consequence minor or even major modifications to well-established routing (or in case of [62] link layer) protocols are done. A promising approach would be to piggyback service discovery messages into the data section of the routing protocol and leave the routing headers intact. This would require only packet interceptors for investigating the data part and executing the service discovery logic specified at the application layer. Very close to this concept, lies the approach proposed in [70], however the example provided (using AODV) requires that the underlying routing protocol's headers have already been service-extended. Intelligent methods for matching service discovery policies with routing policies also have to be defined (e.g. what if push-based service discovery is to be run over a reactive routing protocol?).

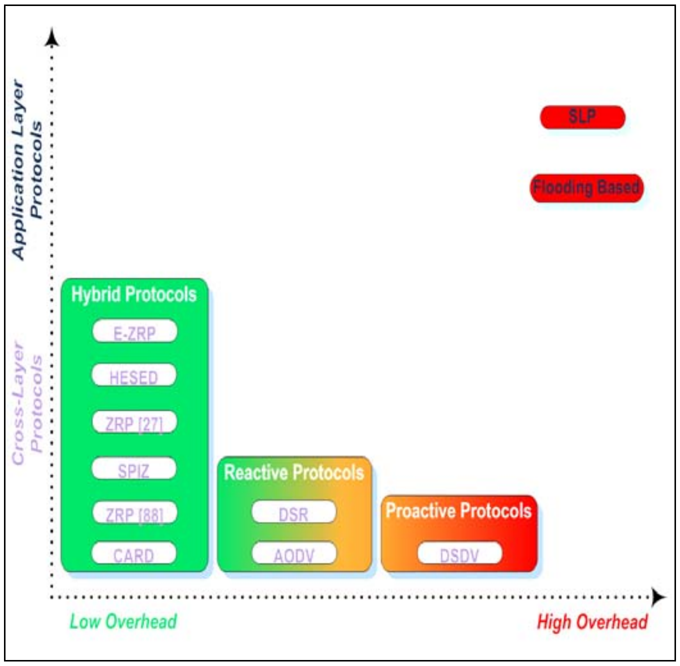

Figure 2: Overhead comparison of Cross Layer vs. Application Layer Service Discovery Protocols

Since direct comparisons are difficult due to lack of specific compatible performance data for the various protocols, a rough categorization regarding which type of routing protocol (reactive, proactive or hybrid) is more efficient (in terms of messaging overhead) when integrated with a service discovery protocol is given in Fig.2, based on simulation results collected and combined from [12], [25], [27], [28], [40], [61], [62], [66], and [88].

V. Special Features

A. Service Description Options

A simple method for describing services is to use *Unique Universal Identifiers* (UUIDs). Such a method was adopted in [5] and [25]. The use of UUIDs is mostly recommended in environments where there is not much service heterogeneity and resources such as energy and bandwidth are very scarce. They also fit well in network layer based service discovery protocols, where there is a need to enhance routing messages with service information without increasing the length of those messages too much. The routing process should not be affected and hence small service-aware routing messages are preferable. Moreover, keeping those messages short leads to low energy consumption for sending and receiving them. It is worth noting that UUIDs could also be combined with anycast or multicast addresses to facilitate the discovery process.

However, UUIDs should be a-priori known to all the nodes participating in a MANET. In addition, UUIDs cannot capture service attributes in order to help users make more informed decisions upon selecting among similar services. In [14] a two-level discovery protocol is proposed, adding service attribute support to a method based on UUIDs. There, services are first discovered using the classic UUID method, but then users are given the opportunity to request service attributes that will further specify the discovered service's capabilities.

Attributes or attribute lists implemented as text-based keyword-value pairs have been extensively used as service descriptions. In [3] such descriptions are written in the Extensible Markup Language (XML). Similar descriptions are also used in [10] where an XML-like language for describing services is defined, accompanied by a tree-like hierarchy for service categorization. In contrast to the approach in [3], where each new description for a new service must go through standardization, in [10] the tree-like hierarchy of services can be seamlessly extended with new categories and services.

All the aforementioned approaches are generally based on *exact keyword matching–syntactic matching*. This implies that service providers and service requestors have agreed on the exact keywords that should be used for each service. Especially when a single keyword is used to discover a service, it is very difficult to select it properly, since a very broad keyword will result in a lot of irrelevant services, while a keyword that is too specific will lead to the exclusion of many, possibly useful services. In [34] the authors propose a *multi-keyword service discovery protocol* to tackle this issue. Their protocol, through a training process, adapts to keyword usage patterns and complements a user's requests with additional keywords automatically. Their performance evaluation when using 1 to 4 such additional keywords shows a 25% to 70% hit rate improvement compared to traditional single keyword matching.

Another class of protocols, however, relies on the use of *ontologies* so that *semantic matching* is possible and keyword similarity can be taken into account when searching for services. GSD [18] for example exploits the semantic capabilities offered by the Web Ontology Language (OWL) to describe services and service requests. OWL is also used to define an ontology for the services in the MANET. OWL provides rules for describing further constraints and relationships among services, which can be exploited to create the ontology of services and service groups. However, it should be noted that in this class of protocols the ontology must be common for all nodes, and hence a priori agreed to among them.

Let us mention here that service descriptions concern not only the functional characteristics of a service. Service descriptions can be rich enough to also provide *context awareness, scope awareness* and *QoS awareness* regarding a service.

B. Service Selection Mechanisms

A basic feature for service discovery approaches, which has been many times underestimated, is service selection. Service selection is the phase that comes after service replies have been gathered by the service requestor.

Service selection can be categorized into *automatic* and user *assisted*. Some early work regarding both manual service discovery (with the help of a service browser implemented as an application) and automated service selection (based on a service ranking system with the ranking function being formulated by the user) is cited in [37].

A rather simple approach for automatic service selection is to select the *best match* according to the similarity degree between the keywords supplied in the service request and those describing the service. This however implies that a distance function applied to keywords for determining their degree of similarity is available.

Generally a selection algorithm is based on certain criteria or metrics. These metrics can be either *route* (e.g. hop-count, bandwidth, delay) or *service* (e.g. server load, remaining energy, capacity) specific. In [25] and [26] the service with the *lowest hop-count* is automatically selected. Paper [25] demonstrates through simulations that the achieved localization of communications leads to improved network performance. It also shows that triggering *reselection* of servers after detecting changes in network topology is very effective in lowering congestion and delays. In [38] the proposed discovery protocol selects a service instance based on two metrics, the *hop-count* between service requestor and service provider and also the *capacity of service (CoS)*. The CoS metric expresses the nominal capacity of a service instance. The service selection algorithm is automatic and does not involve interaction with the user.

A more complex approach is adopted in [8], where *mobile agents* are used to perform service selection after receiving the list of available services. Users can customize their selection algorithms and embed them in the mobile agent. Context information potentially useful for performing the selection is *current system user load, actual bandwidth available, actual packet drop rates and the velocity of the provider*. Agents are transferred to the service providers' devices and compute a rank based on the specified metrics. They then send these ranks back to the requestor and based on a local policy the desired service is selected (e.g. first rank received, or best rank). This way bandwidth is also saved since only a value is sent back instead of the whole context information.

All the aforementioned approaches, consider the impact of service selection mechanism on the client side (delays, hit ratio, QoS). In [68] the authors investigate the impact of two basic and easy to implement service selection strategies on the lifetime of mobile servers and of the whole network. Using the first strategy, a client always selects the nearest server (in hops), while using the second strategy a client always selects the server with the maximum remaining energy. Through simulations it is shown that selecting providers based on their remaining energy yields on the average 5% to 10% better performance in terms of service and network lifetimes, service success and service discoverability ratios, compared to selecting the closest server. A similar work in [69] also evaluates the performance of the closest server selection strategy, but against a strategy considering the available bandwidth of the path to the service provider. According to the latter strategy, the path with the maximum minimum bandwidth among all candidate paths towards matching service providers is selected. Early simulations show that both strategies have the same service discoverability ratios, leading to the conclusion that different route selection criteria do not seem to have significant impact. It should be noted however, that both [68] and [69] use an AODV-based service discovery protocol, meaning that route selection strategies are severely affected by the expanding ring search (ERS) mechanism of AODV. Using ERS clients may only discover a few (or even only one) service providers located near them and hence path-based (e.g. bandwidth) or server-based (e.g. energy) selection policies may actually yield the same selection as the closest server selection policy. Route selection criteria would be more important for non ERS-based protocols, which can potentially discover all possible paths to all possible service providers.

C. Service State Maintenance Techniques

The issue of service state maintenance certainly cannot be neglected when designing a service discovery protocol. It is a challenge to maintain accurate and valid service information and service state especially in MANETs where the inherent dynamism leads to frequent changes in service availability. One approach is to maintain a *hard state* of services where a provider must de-register its service/s before leaving the ad hoc network. However, in MANETs where unpredictable disconnections occur (due to mobility, path loss, congestion and node failures), assuming that a provider will be able to de-register its service before disconnecting is not realistic. The opposite approach is to maintain a *soft state* of services. In this case each service record is associated with a Time-To-Live (TTL) counter, upon the expiry of which the service record is automatically deleted. It is the job of the service provider to periodically refresh that counter by re-advertising the service in order for the service to stay 'alive' in the caches of nodes and directories. These approaches mainly relate to the availability of a service and not its state.

For state maintenance one can use two mechanisms. The first mechanism is *polling*. Polling is used by service requestors to ask a server about the current state of a service (frequently, QoS related metrics are involved). Polling can be performed proactively or on-demand. The second mechanism, namely *notifications*, is the inverse mechanism of polling. Notifications are sent by service providers (or directories) to inform clients about service state changes. In this mechanism clients have to register to the server (or to the directory) their interest in receiving service state updates.

In [39] the authors prove that notifications are less efficient in terms of produced control messages than periodic polling, especially when the status of services in the MANET changes very frequently. However, they also show that polling is a less responsive mechanism than update notifications because of the dependency on the period of polling.

VI. Security, Privacy and Trust in Service Discovery

Recently, in [47], a security requirements model specific to service-oriented architectures has been clearly defined. According to this model there are four sets of security requirements, regarding:

- Service registration and deregistration: during this phase mutual authentication between directories and providers must be ensured. Also upon service registration, service integrity must be kept until deregistration from the directory.
- Service discovery: only authorized clients must be allowed to discover services, and only those services for which they have access rights. Moreover service requests and replies must be kept confidential so that an attacker/eavesdropper cannot perform an inventory of available services and devices.
- Service delivery: during delivery, the service must be protected against malicious tampering or accidental modifications by intermediaries.
- Service availability: the system must be able to handle denial of service attacks, including denying service discovery to illegitimate clients.

In addition, the authors of [49] arrive at similar requirements for secure service discovery, by identifying the possible types of threats that could arise from various types of node misbehavior. According to their threat model, nodes can be categorized as: failed (nodes-victims of attacks), selfish (participating in the service discovery process only when it is convenient for them) and malicious (nodes trying to disrupt the service discovery process).

Unfortunately service discovery approaches fail to address most of the above requirements and often overlook the problems of security, privacy and trust. These problems are important, especially when service discovery protocols are employed in public ad hoc environments (e.g. shopping malls, concerts, conferences). On the contrary, in private ad hoc environments (e.g. home or enterprise networks) devices are more or less trusted and managed by one administrator (their owner) and access to (support by) fixed trusted service registries, servers and authorities can be assumed. In the remainder of this section we briefly present and comment on how proposed service discovery approaches deal with security, categorizing them according to the environment they mostly fit in:

A. *Private/"Managed" Ad Hoc Environments*

In this sub-section we briefly report and discuss the security features of service discovery approaches designed mainly with private ad hoc environments in mind, whose security models fit better to such "managed" environments (a detailed security analysis for those protocols can be found in [47] and [48]).

Jini enhanced with the Jini Security Framework provides authentication, integrity and confidentiality. Jini is based on the use of service objects (proxies) downloaded to client devices in order to access a specific service. The security framework allows a service provider to dictate which clients are allowed to download and execute a service proxy. Also it allows clients to impose constraints on the functionality of a service object once downloaded. It is however cumbersome to verify that the service proxy can be trusted (at code level) and that it will closely follow the client's restrictions when contacting remote servers.

In UPnP enhanced with UPnP security [50], every device has a public key. Many different mechanisms are supported for authorization, authentication, confidentiality and integrity. However, they are all based on centralized architectures implying/requiring the existence of certificate validation authorities and authorization servers storing access control lists.

SLPv2 also provides authentication and data integrity based on private and public keys that have to be distributed to devices/users by an administrator, something that may impose scalability problems for large networks. Moreover, the authentication provided is only one-way, i.e. only users can authenticate service providers and directories and not the opposite.

Bluetooth SDP relies on security mechanisms described in the Bluetooth specification. According to the specifications unidirectional or mutual authentication and encryption is supported. These features are based on a secret link key that has to be exchanged by two devices when discovering each other. A protocol for secret key distribution however is not provided, leading to possible inefficiencies for large networks consisting of unknown devices.

SSDS [51] provides authentication for clients and service providers through certificates. It also provides data privacy and confidentiality through symmetric key encryption. However, it assumes that clients and service providers always trust directories, where service providers register their services and which respond to clients' service discovery requests.

Salutation provides only a rather basic authentication scheme based on usernames and passwords to control service access.

Similarly to Salutation, the service provision model proposed in [52] is based on service registries (either centralized or decentralized) protected with a password. Also services inside registries may be protected with their own passwords for increased access permissions granularity. However, this scheme also requires password distribution and management (possibly by an administrator), which affects the system's scalability.

Finally, in [56] directories (i.e. registries) use a multicast address to periodically announce their unicast address and certificate. Servers register their service to those directories. Also, directories respond to client requests for services. Communications are secured by a PKI infrastructure. What is new is that clients are supposed to have, besides ad hoc communication capabilities, access to the Internet over a side channel. Through this channel clients and servers communicate with their proxies, which handle registration, authentication, authorization, and key management for them. Those proxies upon authenticating the involved entities also send them a session key that is used for encrypting their communications over the ad hoc network. Unfortunately this solution cannot work in ad hoc environments without the help of side channels for accessing proxies over the Internet.

B. Public/ "Pure" Ad Hoc Environments

Trust among certain entities (e.g. directories and service providers) and central administration cannot be assumed in a public environment however.

In [42, 53] the authors propose a lightweight distributed privacy and security aware service discovery approach. Users are potential service discovery clients and also providers. Each provider may administer his own directories, which can be portable (e.g. on a PDA). For every directory the provider creates and manages user identities (in the form of passwords or certificates) for controlling access to them. In order to access services from a foreign directory, physical contact with its administrator is required in order to get the appropriate access credentials. Clients are represented by user agents responsible for managing access credentials to avoid distracting the user every time access to a specific service is needed. Moreover, to enhance user privacy and confidentiality service requests and replies do not contain plain text, but are transferred in the form of Bloom filters. Those Bloom filters also include random bits set to 1 (for making it even harder to an attacker to infer who discovers what service) and are also digitally signed. Unfortunately, hash functions producing those Bloom filters have to be agreed upon beforehand among all possible members of the network and there is no automatic way for acquiring them. Furthermore, despite the fact that this work is a step forward towards secure service discovery in "pure" ad hoc environments, physical contact to obtain access rights and controlling access only at the directory level and not at the service level, render this solution inflexible.

Another promising direction for securing service discovery in "pure" ad hoc environments is to use distributed trust models that do not rely on centralized Certification Authorities. For example SPDP [44] is such a protocol based on an anarchic trust model. In SPDP there are no directories. Service clients and providers gradually built trust relationships based on past transactions and recommendations obtained by other trusted entities. Mutual authentication between devices is based on a challenge mechanism. Actually, each device acts as a Certification Authority for the services it provides. Data integrity and confidentiality for service requests and replies is guaranteed with IPSec. A basic shortcoming is that the protocol cannot defend against a malicious device that joins the network for the first time. Its initial trust-score will be set to ignorance (0.5 in a scale from 0 to 1) and clients or providers may interact with it at their own risk. This problem becomes even more severe in the case that devices are able to easily change their identity.

In order to protect service providers from this problem, SSRD+ [45, 54] proposes a similar trust mechanism but enhances it with a risk assessment mechanism. Using this mechanism, every service provider ranks the risk for providing each of its services to unknown clients based on past service usage patterns. In order to calculate the risk it collects information about the number of times the service access was granted and denied, the average trust values of the devices that requested the service, service time etc. A similar mechanism could also be used by clients for ranking services and be protected from unknown service providers (or malicious providers continuously changing identities).

C. Privacy Beyond Service Discovery and Open Issues

Even if every entity participating in the service discovery process is authenticated and communications are encrypted, service providers become the holders of sensitive user information such as user location, service usage patterns, interaction history etc. Although out of the scope of this paper, it is very challenging to provide ways for clients to interact with service providers without "leaving" sensitive information. For a review of current approaches dealing with such issues, the interested reader may refer to [55].

Finally the following issues regarding privacy in service discovery remain open: balancing security with user interaction. Most approaches require considerable user intervention e.g. for installing certificates, contacting administrators or service owners for acquiring passwords, for physical contact in order to authenticate to a device offering a desired service [43] etc. Balancing security with device capabilities (many pervasive devices may not be capable of supporting the security features due to limited memory storage, battery power and computational capability) is also a reasonable requirement [56].

VII. Discussion And Open Issues

In the following subsections we identify open issues related to service discovery in mobile ad hoc networks that present an open field for further investigation and research:

A. *Adaptation and Flexibility*

Besides a few exceptions, most service discovery protocols do not adapt their mechanisms based on context. Context-awareness has been primarily used for augmenting service selection rather than for self-tuning of the discovery protocol.

Depending on the sophistication of the service discovery protocol self-tuning can be radical or conservative. Conservative tuning refers to changing the values of the protocol's basic operational parameters (e.g. the maximum number of hops an advertisement is allowed to travel), while radical tuning regards changing the method of operation (e.g. from push-based to pull-based). Most protocols employ predefined fixed parameters for their operation (e.g. discovery scopes, advertisement frequency, time intervals for repeating a failed query, cached service lifetimes, cache sizes etc.). This monolithic protocol design cannot cope well with the dynamism of ad hoc networks, where node speed, node density and channel conditions may vary a lot.

For example, a proactive service discovery protocol may need to decrease the service advertising frequency when severe congestion is detected. This way it will not aggravate more the congestion and will also allow directories or servers to save energy avoiding useless transmissions (i.e. a transmission that would probably result in more collisions).

In a highly dense environment a pull-based service information dissemination method may be more efficient than a push based one (assuming that a protocol supports both). Hence, there is a need for developing protocols and architectures with more autonomic features [78] allowing their optimization and self-adaptation on-the fly depending on the network's conditions. In this direction, one of the most difficult issues is how to coordinate self-adapting discovery agents so that service discovery can be realized effectively. Election mechanisms (e.g. for deciding on the preferred dissemination method) and incentive schemes (e.g. for conforming to majority decisions regarding the service discovery process) may prove valuable tools in this effort.

B. *Interoperability*

Considering the multitude of service discovery standards, architectures and protocols and taking also into account the ubiquitous and pervasive nature of future environments, interoperability in service discovery will be a major issue requiring attention (to avoid building a 'Tower of Babel'). It is clear that requiring all devices to support all service discovery protocols is far from being realistic. To the contrary interoperation seems to be the way forward. Despite a few efforts [71-76] much remains to be done towards this. It is out of the scope of this paper to analyze in detail the approaches proposed for service discovery protocol interoperability; however, it is worth outlining their basic characteristics and weaknesses. Some of the approaches try to make direct translations from one protocol to another, while others try to translate all protocols to a common protocol. It is obvious that not every protocol provides the same functionality and some mappings are simply not achievable (e.g. UPnP service status notifications cannot be mapped to any SLP function). On the other hand defining a common protocol which can support all possible functionality ranging from service description methods to service invocation to security provision to context-awareness etc. is too optimistic if not impossible. Furthermore, some approaches, called explicit, require that client applications make calls to the common protocol implemented as middleware. Other approaches, called transparent, implement middleware that accepts any service discovery protocol call issued by legacy clients and transforms it appropriately depending on the protocol provided in the network. However, all of those approaches (transparent or explicit) require that translation modules for all possible protocols are available in the middleware, or that nodes are always available to make translations (either fixed bridges or mobile clients), or that thin client devices can deal with the complexity of the code required for identifying the used service discovery protocols and for making protocol translations. Considering the above, further work is needed for developing truly scalable interoperability solutions for service discovery, matching the requirements as well as the restrictions posed by MANET environments.

C. *Benchmarking*

One of the major problems in the research area of service discovery for MANETs is that little attention has been given in standardizing the evaluation of service discovery protocols. In effect discovery protocols found in the literature are often incomparable since different settings and assumptions have been made during their evaluation.

Developing a universal evaluation framework for service discovery protocols would allow fair and direct comparisons among protocols. The foundations of such a framework for simulation-based evaluation have been set in [84], where authors presented BenchMANET. This benchmark specifies a number of tests associated with realistic service discovery applications and scenarios for MANETs. Each of these tests defines different configurations of basic parameters that affect service discovery (service provider population, client population, network size, area size, mobility model, services per node, service advertisement lifetime). However, this framework still lacks other equally important parameters e.g. related to frequency of advertisement and querying. Moreover, the specifics of the underlying routing protocol are not considered.

Besides simulation an evaluation can use analytical models too. Regarding analytical modeling and evaluation of service discovery protocols there have been proposed two models in [85] and [86]. The model developed in [85] uses a M/G/c/c queue to model and predict the behavior of the service cache on a node. Using the model the average timeout of a service description can be determined given the protocol's parameters (e.g., advertisement frequency, service cache size etc.). Also the optimal timeout for service descriptions can be calculated for achieving a non-fluctuating average number of services discovered (equilibrium). Unfortunately the aforementioned model, being abstract, cannot take into account radio link behavior and node mobility as well as of other specifics of the service discovery protocol (e.g. forwarding policies). Moreover the model was designed only for evaluating proactive directory-less service discovery. Directory-based service discovery was modeled in [86], where a queuing based model for service caches was also developed. However, this model is more elaborate since it accounts for node movement and link failures. Through this model the optimal service advertisement update rate can be determined in order to optimize system performance in terms of success rate and network overhead. Such analytical models can prove to be valuable tools in the evaluation and optimization of developed service discovery protocols, but require more sophistication to cover (i.e., be adaptable to) a broader range of service discovery approaches (e.g. hybrid discovery architectures).

VIII. Conclusions

MANETs have attracted extraordinary attention from the research community in recent years, yet civilian, mass applications remain elusive. Efficient service discovery is one of the key issues that need to be resolved for the acceptance of MANETs.

In this article we surveyed the literature on service advertisement, discovery and selection schemes for MANETs, presenting the most representative approaches. In most traditional approaches, these three aspects are intertwined and typically an integrated scheme is proposed.

We discussed the fundamental architectures for service discovery, explaining the basic ideas for each architecture and commenting on their merits and drawbacks. Then we classified the basic discovery modes into three categories, based on the way a client (requestor) acquires service information. We discussed the performance for each mode in MANETs with different characteristics. Our analysis led us in identifying the need for having autonomic service discovery protocols capable of flexibly adapting their operation (in terms of selected architecture, discovery mode and values for tunable parameters) to the actual context and service demand specific to the ad hoc network in which the protocol is used.

We have also paid particular attention to a special class of efficient service discovery approaches using cross-layer optimization. The integration of service discovery with several (if not all) types of routing protocols (reactive, proactive, hybrid, multicast) was analyzed and the advantages and disadvantages for each integrated protocol have been presented. Analyzing simulations results found in the literature we conclude that hybrid integrated discovery protocols perform best in terms of overhead. Second best are reactive integrated protocols, leaving the third place to proactive integrated protocols. However, the worst performance in terms of overhead is experienced by application layer-based service discovery protocols (for SLP-based and flooding based protocols).

We have also reported on special features of service discovery and particularly on service description options, service selection mechanisms, service state maintenance techniques and also security issues. Especially regarding security we underlined the need for more flexible secure service discovery protocols that can balance security with user interaction and device capabilities.

Finally we have identified three issues that require further research, namely, service discovery flexibility and adaptation, interoperability and benchmarking.

References

[1] Sun Microsystems, “JINI Architecture Specification,” Nov. 1999.

[2] Salutation Consortium, “Salutation Architecture Specification,” Available: http://web.archive.org/web/20030623193812/www.salutation.org/, 1999. (The Salutation Consortium was disbanded on 30 June 2005).

[3] Microsoft Corporation, “Universal Plug and Play: Background,” Available: http://www.upnp.org/resources/UPnPbkgnd.htm, 1999.

[4] “Universal Description Discovery and Integration Platform,” Available: http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf, Sept. 2000.

[5] “Specification of the Bluetooth System,” Available: http://www.bluetooth.com, December 1999.

[6] E. Guttman, C. Perkins, J. Veizades, and M. Day. “Service Location Protocol, Version 2,” IETF RFC 2608, June 1999.

[7] Apple Inc, “Bonjour Technology White Paper,” Available: http://images.apple.com/macosx/pdf/MacOSX_Bonjour_TB.pdfT, 2007

[8] J. Tyan and Q. H. Mahmoud, “A network layer based architecture for service discovery in mobile ad hoc networks,” Proc. IEEE CCECE 2004, Niagara Falls, May 2004.

[9] M. Nidd, “Service Discovery in DEAPspace,” IEEE Personal Communications, August 2001, pp. 39-45.

[10] S. Helal, N. Desai, V. Verma, and C. Lee, “Konark - A Service Discovery and Delivery Protocol for Ad hoc Networks,” Proc. 3rd IEEE Conference on Wireless Communication Networks (WCNC), New Orleans, Louisiana, March 2003.

[11] G. Schiele, C. Becker and K. Rothermel, “Energy-Efficient Cluster-based Service Discovery for Ubiquitous Computing,” Proc. 11th ACM SIGOPS European Workshop, Leuven, Belgium, September 2004.

[12] A. Helmy, S. Garg, N. Nahata and P.Pamu, “CARD: A contact-based Architecture for Resource Discovery in Wireless Ad hoc Networks,” Mobile Networks and Applications (MONET), Vol. 10, 2005, pp.99-113.

[13] R. Koodli and C. E. Perkins, “Service discovery in on-demand ad hoc networks,” IETF Internet Draft, draft-koodli-manet-servicediscovery-00.txt, October 2002.

[14] S. Preu, “JESA Service Discovery Protocol,” Proc. Networking 2002, Lecture Notes in Computer Science Series, vol. 2345, 2002, pp. 1196-1201.

[15] D. Doval and D. O'Mahony, “Nom: Resource Location and Discovery for Ad hoc Mobile Networks,” Proc. 1st Annual Mediterranean Ad hoc Networking Workshop, Med-hoc-Net 2002, Sardegna, Italy, Sept 4-6, 2002.

[16] M. Klein and B. König-Ries, “Multi-layer clusters in ad hoc networks - an approach to service discovery,” Proc. 1st International Workshop on Peer-to-Peer Computing (Co-Located with Networking 2002), Pisa, Italy 2002, pp. 187–201.

[17] S. Motegi, K. Yoshihara and H. Horiuchi, “Service discovery for wireless ad hoc networks,” Proc. 5th International Symposium on Wireless Personal Multimedia Communications, vol. 1, 2002, pp. 232 - 236.

[18] D. Chakraborty, A. Joshi Y. Yesha, and T. Finin, “Toward Distributed Service Discovery in Pervasive Computing Environments,” IEEE

Transactions on Mobile Computing, Volume 5, Number 2, February 2006, pp. 97-112.

[19] Liang Cheng, “Service advertisement and discovery in mobile ad hoc networks,” Workshop on Ad hoc Communications and Collaboration in Ubiquitous Computing Environments, in conjunction with the ACM 2002 Conference on Computer Supported Cooperative Work, New Orleans, November 16-20, 2002.

[20] Wenbin Ma, Baoning Wu, Wei Zhang, and Liang Cheng, “Implementation of a lightweight service advertisement and discovery protocol for mobile ad hoc networks,” IEEE Globecom 2003, San Francisco, CA, Dec 1-5, 2003, pp. 1023-1027

[21] K. Seada and A. Helmy, “Rendezvous Regions: A Scalable Architecture for Service Location and Data-Centric Storage in Large-Scale Wireless Networks,” ACM Mobicom 2003, San Diego, CA, September 2003. (Refereed Poster)

[22] M. Klein, B. König-Ries, P. Obreiter,” Service Rings - A Semantic Overlay for Service Discovery in Ad hoc Networks,” Proc. 6th Intl. Workshop on Network-Based Information Systems (NBIS 2003) at DEXA 2003, Prague, September 2003.

[23] J. Tchakarov and N. Vaidya, “Efficient Content Location in Wireless Ad Hoc Networks,” IEEE International Conference on Mobile Data Management (MDM), January 2004.

[24] Honghui Luo and M. Barbeau, ”Performance Evaluation of Service Discovery Strategies in Ad Hoc Networks,” Proc. 2nd Annual Conference on Communication Networks and Services Research, CNSR 2004, Fredericton, N.B., Canada, May 19-21, 2004, pp. 61-68.

[25] Engelstad, P.E., Zheng, Y., Koodli, R., Perkins, C.E., *"Service Discovery Architectures for On-Demand Ad Hoc Networks"*, International Journal of Ad Hoc and Sensor Wireless Networks, Old City Publishing (OCP Science), Vol. 2. Number 1, March 2006, pp. 27-58

[26] C. Frank and H. Karl, “Consistency Challenges of Service Discovery in Mobile Ad Hoc Networks,” Proc. 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), Venice, Italy, October 2004, pp. 105-114.

[27] Chang-Seok Oh, Young-Bae Ko and Young-Sung Roh, “An Integrated Approach for Efficient Routing and Service Discovery in Mobile Ad Hoc Networks,” Proc. IEEE Consumer Communications and Networking Conference (CCNC'05), Las Vegas, Nevada, Jan. 2005.

[28] U. C. Kozat and L. Tassiulas, “Service discovery in mobile ad hoc networks: an overall perspective on architectural choices and network layer support issues,” Ad Hoc Networks 2(1), 2004, pp. 23-44.

[29] Zhen-guo Gao, Xiao-Zong Yang, Tian-yi Ma, Shao-Bin Cai, “RICFFP: An Efficient Service Discovery Protocol for MANETs,” Proc. International Conference on Embedded and Ubiquitous Computing, EUC-04, Aizu, Japan, 26-28 August 2004, pp. 786-795.

[30] F. Sailhan and V. Issarny, ” Scalable Service Discovery for MANET,” Proc. 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom'2005), Kauai Island, Hawaii 8–12 March 2005.

[31] P.E Engelstad and Zheng, Y., “Evaluation of Service Discovery Architectures for Mobile Ad Hoc Networks,” Proc. 2nd annual conference on Wireless On-demand Networks and Services (WONS 2005), St.Moritz, Switzerland, Jan. 19-21, 2005.

[32] J. Tyan and Q. H. Mahmoud, “A Comprehensive Service Discovery Solution for Mobile Ad Hoc Networks,” MONET 10(4), 2005, pp. 423-434.

[33] U. Mohan, K. C. Almeroth and E. M. Belding-Royer, “Scalable Service Discovery in Mobile Ad Hoc Networks,” NETWORKING 2004, Athens, Greece, 2004, pp. 137-149.

[34] Hen-I Yang and R. Bose, “A Multi-level Multi-keyword Service Discovery Protocol for Mobile Ad hoc Networks,” Department of Computer and Information Science and Engineering, University of Florida. http://www.cise.ufl.edu/class/cen5531fa05/files/hyang_rbose.pdf

[35] A. Nedos, K. Singh and S. Clarke, “Service*: Distributed Service Advertisement for Multi-Service, Multi-Hop MANET Environments,” Proc. 7th IFIP International Conference on Mobile and Wireless Communication Networks (MWCN'05), Springer-Verlag, Marrakech, Morocco, 2005.

[36] G. E. Güichal, “Service Location Architectures for Mobile Ad hoc Networks,” Master Thesis, Georgia Institute of Technology, July 2001.

[37] M. Barbeau, “Service discovery protocols for ad hoc networking,” CASCON 2000, Workshop on Ad Hoc Communications, Mississauga, Ontario, Canada, 2000.

[38] V. Lenders, M. May and B. Plattner, “Service Discovery in Mobile Ad Hoc Networks: A Field Theoretic Approach,” Elsevier Journal on Pervasive and Mobile Computing (PMC), Volume 1, Issue 3, September 2005, pp. 343-370.

[39] C.Dabrowski, K.Mills, and J.Elder, “Understanding consistency maintenance in service discovery architectures during communication failure,” Proc. 3rd International Workshop on Software and Performance, ACM Press, Rome, Italy, July 2002, pp. 168–178.

[40] C. N. Ververidis and G. C. Polyzos, “Extended ZRP: Performance Evaluation of a Routing Layer Based Service Discovery Protocol for Mobile Ad Hoc Networks,” Proc. 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous’05), San Diego, California, July 2005, pp. 114-123.

[41] J. Antonio García-Macías and Dante Arias Torres, “ Service Discovery in Mobile Ad hoc Networks: Better at the Network Layer?,” Proc. IEEE Intl. Workshop on Wireless and Sensor Networks (WSNET'05), Oslo, Norway, June 2005, pp. 452-457.

[42] Feng Zhu, Mutka M. and Ni L., “PrudentExposure: a private and user-centric service discovery protocol,” Proc. 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004 (PerCom 2004), New York, USA¸ March 2004, pp. 329-338.

[43] H. Kopp, U. Lucke and D. Tavangarian, “Security Architecture for Service-based Mobile Environments,” Proc. 3rd IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2005 (PerCom 2005), Kauai Island, Hawaii, March 2005, pp. 199-203.

[44] F. Almenarez, and C. Campo, “SPDP: A Secure Service Discovery protocol for Ad hoc networks,” 9th open European summer school and IFIP workshop on next generation networks (EUNICE 2003), Hungary, Sep 2003.

[45] M. Sharmin, S. Ahmed, S. I. Ahamed, and H. Li, “SSRD+: A Privacy-aware Trust and Security Model for Resource Discovery in Pervasive Computing Environment,” Proc. 30th Annual International Computer Software and Applications Conference (COMPSAC 2006), IEEE CS Press, Chicago, USA, September 17-21, 2006, pp. 66-70.

[46] J. Hoebeke, I. Moerman, B. Dhoedt and P.Demeester, “Analysis of Decentralized Resource and Service Discovery Mechanisms in Wireless Multi-hop Networks,” Proc. 3rd International Conference on Wired/Wireless Internet Communications (WWIC 2005), Xanthi, Greece, May 2005, pp. 117-127.

[47] D. Cotroneo, A. Graziano, and S. Russo, “Security Requirements in Service Oriented Architectures for Ubiquitous Computing,” Proc. 2nd Workshop on Middleware For Pervasive and Ad hoc Computing, ACM Press, New York, NY, pp. 172-177.

[48] F. Zhu, M. Mutka, L. M. Ni, "Service Discovery in Pervasive Computing Environments", Pervasive Computing, IEEE Volume 4, Issue 4, Oct. Dec. 2005, pp. 81–90.

[49] A. Leung and C. J. Mitchell, “A service discovery threat model for ad hoc networks,” Proc. International Conference on Security and Cryptography (SECRYPT 2006), Setubal, Portugal, August 7-10, 2006, INSTICC Press, 2006, pp.167-174.

[50] C. Ellison, “Home Network Security,” Intel Technology Journal, vol. 6, 2002, pp. 37-48.

[51] S. Czerwinski, B. Y. Zhao, T. Hodes, A. Joseph, and R.Katz, “An Architecture for a Secure Service Discovery Service,” Proc. 5th Annual International Conference on Mobile Computing and Networks (MobiCom '99), Seattle, WA, 1999.

[52] R. Handorean and G.C. Roman, “Secure service provision in ad hoc networks,” Proc. 1st International Conference on Service Oriented Computing (ICSOC 03), number 2910 in Lecture Notes in Computer Science, 2003, pp. 367–383.

[53] Feng Zhu, Matt Mutka, and Lionel Ni, “A Private, Secure and User-centric Information Exposure Model for Service Discovery Protocols,” IEEE Transactions on Mobile Computing, vol. 5, No. 4, 2006, pp. 418-429.

[54] M. Sharmin, S. Ahmed, and S. I. Ahamed, “An Adaptive Lightweight Trust Reliant Secure Resource Discovery for Pervasive Computing Environments”, Proc. 4th Annual IEEE International Conference on Pervasive Computer and Communications (PerCom 2006), Italy, Mar 06.

[55] P. Bhaskar and S. I. Ahamed, "Privacy in Pervasive Computing and Open Issues," Proc 2nd International Conference on Availability, Reliability and Security (ARES'07) pp. 147-154

[56] F. Zhu, M. Mutka and L. Ni, "Facilitating Secure Ad hoc Service Discovery in Public Environments," Journal of Systems and Software 76 (2005), pp. 45-54.

[57] O. Ratsimor, D. Chakraborty, S. Tolia, D. Kushraj, A. Kunjithapatham, G. Gupta, A. Joshi, T. Finin, "Allia: Alliance-based Service Discovery for Ad hoc Environments," Proc. 2nd ACM Mobile Commerce Workshop, September 2002.

[58] Bob Pascoe, "Salutation-Lite: Find-and-Bind Technologies for Mobile Devices," Salutation Consortium, Available: http://web.archive.org/web/20031023072133/www.salutation.org/whitepaper/Sal-Lite.pdf, June 6, 1999.

[59] S. Cheshire and M. Krochmal, "Multicast DNS (Internet Draft)," Apple Computer, Inc. http://files.multicastdns.org/draft-cheshire-dnsext-multicastdns.txt, 2004.

[60] S. Cheshire and M. Krochmal, "DNS-Based Service Discovery," Internet-Draft (work in progress), http://files.dns-sd.org/draft-cheshire-dnsext-dns-sd.txt, August 2006.

[61] G. P. Halkes, A. Baggio and K. G. Langendoen, " A Simulation Study of Integrated Service Discovery," EUROSSC 2006, LNCS 4272, 2006, pp.39-53.

[62] V. Atanasovski and L. Gavrilovska, "Efficient Service Discovery Schemes in Wireless Ad Hoc Networks Implementing Cross-Layer System Design," Proc. 27th International Conference on Information Technology Interfaces (ITI 2005), Cavtat, Croatia, June 20-23, 2005.

[63] J. L. Jodra, M. Vara, J. M. Cabero, J. Bagazgoitia and J. L. Jodra, "Service Discovery Mechanism Over OLSR for Mobile Ad hoc Networks," Proc. 20th International Conference on Advanced Information Networking and Applications - Volume 2 (AINA'06), 2006, pp. 534-542.

[64] L. Li and L. Lamont, "A Lightweight Service Discovery Mechanism for Mobile Ad Hoc Pervasive Environment Using Cross-Layer Design," Proc. 3rd IEEE International Conference on Pervasive Computing and communications Workshops, March2005, pp.55-59.

[65] R. Harbird, S.Halies and C.Mascolo, "Adaptive resource discovery for ubiquitous computing," Proc. 2nd Workshop on Middleware for Pervasive and Ad hoc Computing, Toronto, Canada, October 2004, pp.155-160.

[66] D. Noh and H. Shin, "SPIZ: An Effective Service Discovery Protocol for Mobile Ad Hoc Networks," EURASIP Journal on Wireless Communications and Networking, vol. 2007, Article ID 25167, 13 pages, 2007, doi: 10.1155/2007/25167.

[67] H. Hassanein, Y. Yang and A. Mawji, "A new approach to service discovery in wireless mobile ad hoc networks," International Journal of Sensor Networks, Volume 2, Number 1-2 / 2007, pp.135-145.

[68] S. Athanaileas, C. N. Ververidis and G. C. Polyzos, "Optimized Service Selection for MANETs using an AODV-based Service Discovery Protocol", accepted for publication in the 6th Annual Mediterranean Ad Hoc Networking Workshop (MEDHOCNET 2006), Corfu, Greece, June 2007.

[69] Z.Fan and E. G. Ho, "Service Discovery in Ad Hoc Networks: Performance Evaluation and QoS Enhancement", Wireless Personal Communications: An International Journal, Volume 40, Issue 2, Springer, January 2007, pp.215 – 231.

[70] P. Stuedi, M. Graber and G. Alonso, "Exploiting Synergies between Service Discovery and Routing in Wireless Multihop Networks," Technical Report 510, ETH Zürich, 2006.

[71] J. Allard, V. Chinta, S.Gundala and G. Richard, "Jini Meets UPnP: An Architecture for Jini/UPnP Interoperability," Proc. International Symposium on Applications and the Internet (SAINT- 2003), Orlando, Florida (USA), January 2003.

[72] A. Friday, N. Davies, N. Wallbank, E. Catterall, and S. Pink, "Supporting service discovery, querying and interaction in ubiquitous computing environments," Wireless Networks, vol. 10, no. 6, 2004, pp. 631–641.

[73] Paul Grace, Gordon S. Blair and Sam Samuel. "ReMMoC: A Reflective Middleware to Support Mobile Client Interoperability," Proc. International Symposium on Distributed Objects and Applications (DOA), Catania, Sicily, Italy, November 2003.

[74] P.G. Raverdy, V. Issarny, A. de La Chapelle, and R. Chibout, "A multi-protocol approach to service discovery and access in pervasive environments," Proc. 3rd Annual International Conference on Mobile & Ubiquitous Systems: Networks & Services, (Mobiquitous 2006), San Jose, California, USA, July 2006.

[75] Y. D. Bromberg and V. Issarny, "INDISS: Interoperable Discovery System for Networked Services," Proc. 6th International Middleware Conference, Grenoble, France, November/December 2005.

[76] T. Koponen and T. Virtanen, "A Service Discovery: A Service Broker Approach," Proc. 37th Annual Hawaii International Conference on System Sciences (HICSS'04), Hawai, January 2004.

[77] D. Chakraborty, A. Joshi and Y. Yesha, "Integrating service discovery with routing and session management for ad hoc networks," Ad Hoc Networks, Volume 4, Issue 2, March 2006, pp. 204-224.

[78] George C. Polyzos, Christopher N. Ververidis and Elias C. Efstathiou, "Service Discovery and Provision for Autonomic Mobile Computing," Proc. 2nd IFIP Workshop on Autonomic Communication (WAC 2005), Springer Lecture Notes in Computer Science Vol. 3854 (LNCS), Athens, Greece, pp 226-236.

[79] M.A. El Saoud, T. Kunz, and Samy Mahmoud, "SLPManet: Service location protocol for MANET," Proc. International Wireless Communications and Mobile Computing Conference (IWCMC 2006), Vancouver, Canada, Jul 2006.

[80] Z. Gao, L. Wang, M. Yang and X. Yang, "CNPGSDP: An efficient group-based service discovery protocol for MANETs," Computer Networks, Volume 50, Issue 16, 14 November 2006, pp. 3165-3182

[81] Z. Gao, L. Wang, M. Yang and D. Wen, "PCPGSD: An enhanced GSD service discovery protocol for MANETs," Computer Communications, Volume 29, Issue 12, 4 August 2006, pp. 2433-2445.

[82] C. Campo, M. Munoz, J. C. Perea, A. Marın, C. Garcıa-Rubio: PDP and GSDL: a new service discovery middleware to support spontaneous interactions in pervasive systems," Proc. 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom'2005 Workshops), Kauai Island, Hawaii 8–12 March 2005.

[83] C. Lee and S.Helal, "Gossip-Based Service Discovery in Mobile Ad Hoc Networks," IEICE-Transactions on Communications Volume E89-B, Number 9, September 2006, pp. 2621-2624.

[84] M. A. El Saoud, T. Kunz, and S. Mahmoud, "BENCHManet: An evaluation framework for service discovery protocols in MANET," Proc. 3rd Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2006), Volume 3, Reston, Virginia, USA, June 2006, pp. 860-865.

[85] D. Chakraborty, A. Shenoi, A. Joshi, and Y. Yesha, "A Queuing Theoretic Approach for Service Discovery in Ad hoc Networks," Proc. Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS), San Diego, California, USA, January 2004.

[86] T. Wu and G.S. Kuo, "An Analytical Model for Centralized Service Discovery Architecture in Wireless Networks," Proc. IEEE 64th Vehicular Technology Conference, 2006, VTC-2006 Fall, September 2006, pp. 1-5.

[87] S. Sivavakeesar, O.F. Gonzalez, G. Pavlou, "Service Discovery Strategies in Ubiquitous Communication Environments," IEEE Communications, special issue on Advances in Service Platform Technologies for Next Generation Mobile Systems, Vol. 44, No. 9, September 2006, pp. 106-113.

[88] A. Helmy, "Contact-extended zone-based transactions routing for energy-constrained wireless ad hoc networks," IEEE Transactions on Vehicular Technology, vol. 54, no. 1, 2005, pp. 307–319.

[89] C. Bettstetter and C. Renner, "A Comparison of Service Discovery Protocols and Implementation of the Service Location Protocol," Proc. EUNICE Open European Summer School, Twente, Netherlands, Sept 13-15, 2000.

[90] Helal, S., "Standards for service discovery and delivery," IEEE Pervasive Computing, Volume 1, Issue 3, 2002, pp. 95-100.

[91] G.G. Richard, "Service advertisement and discovery: enabling universal device cooperation," IEEE Internet Computing, Volume 4, Issue 5, Sep/Oct 2000, pp. 18-26.

AUTHOR BIOGRAPHIES

Christopher N. Ververidis received his Master of Science in Information Systems from the Computer Science Department of Athens University of Economics and Business (AUEB). His Master's thesis on Location Based Services was awarded with the Ericsson Award of Excellence in Telecommunications by Ericsson Hellas S.A. He is currently working as a Ph.D. student and research assistant at the Mobile Multimedia Laboratory (MMLab), AUEB. He has participated in several national and international research programs in the areas of networking and telecommunication services. His current research interests include ubiquitous computing, wireless networks and service discovery protocols for mobile ad hoc networks. He has published 12 papers in refereed workshops and conferences. During his PhD studies he has served as a reviewer for various conferences, workshops and journals (IEEE Infocom, ACM Mobicom, IEEE VTC, IEEE ISCC, IEEE WoWMoM, IEEE PIMRC, WCMC Journal). He is on the Technical Program Committee of IEEE WCNC 2008. He is on the Technical Program Committee of IEEE WCNC 2008.

George C. Polyzos, Professor of Computer Science at AUEB since 1999, is leading the Mobile Multimedia Laboratory. Previously, he was Professor of Computer Science and Engineering at the University of California, San Diego, where he was co-director of the Computer Systems Laboratory, member of the Steering Committee of the UCSD Center for Wireless Communications and Senior Fellow of the San Diego Supercomputer Center. He has received his Dipl. in EE from the National Technical University in Athens, Greece (1982) and his M.A.Sc. in EE (1985) and Ph.D. in Computer Science (1989) from the University of Toronto. His current research interests include mobile multimedia communications, ubiquitous computing, wireless networks, security, Internet protocols, distributed multimedia, and performance analysis of computer and communications systems. Prof. Polyzos is on the editorial board of the journal *Wireless Communications and Mobile Computing* and has been a guest editor for: *IEEE Personal Communications*, ACM/Springer *Mobile Networking*, *IEEE Journal on Selected Areas in Communications*, and *Computer Networks*. He has been on the Program Committees of many conferences and workshops, co-chaired the 1999 IEEE International Workshop on Mobile Multimedia Communications and has been a reviewer for the US NSF, the California MICRO program, the European Commission, the Greek Secretariat of Research and Technology and many scientific journals.