A Unified Service Discovery Architecture for Wireless Mesh Networks

Martin Krebs, Karl-Heinz Krempels, and Markus Kucay

Department of Computer Science, Informatik 4
RWTH Aachen University, Germany
Ahornstrasse 55, 52074 Aachen
{krebs, krempels, kucay}@cs.rwth-aachen.de

Abstract. This paper proposes a unified architecture for service discovery in Wireless Mesh Networks. In this architecture, routing clients and non-routing clients, which connect via a mesh gateway with *Service Proxy (SP)*, can also participate seamlessly in the service discovery process. Therefore, the multicast DNS (mDNS) protocol is encapsulated in Optimized Link State Routing (OLSR) messages to make mDNS multi hop capable. *Service Caches (SC)* on wireless mesh routers are added for efficiency reasons. It is discussed that simple flooding of service discovery messages is not efficient and that inspection of messages at the application layer increases the efficiency of message propagation. We present a plug-in for an OLSR-daemon which is widely used in real world deployments. Finally, the measurements performed in the department's wireless mesh testbed are discussed with results and conclusions.

Keywords: Service Discovery, Wireless Mesh Networks, Cross-Layer Design, OLSR

1 Introduction

Wireless Mesh Networks (WMNs) [1] are an emerging technology in the direction of future wireless networks. They are a flexible technology to provide wireless high-speed broadband coverage to areas where the installation of wires is not possible or too costly. Example deployment scenarios are community, emergency, disaster recovery or municipal networks. From the user point of view, automatic service discovery and zero configuration will be a central task in WMNs.
Current solutions and proposals for service discovery in ad-hoc networks like SLP [2] or UPnP [3] use simple flooding which is unacceptable in wireless networks. Many solutions presume an existence of well defined multicast groups which are also unrealistic in ad-hoc or mesh networks. A more efficient approach are distributed hash tables (DHT) for P2P networks which can be used to create an overlay network, but this is inappropriate for highly dynamic wireless networks. For complex service discovery operations it is not clear whether any meaningful hash value could even be calculated. It can not be assumed that a querying client

knows the hash value of a service in advance.
In this paper we present a complete service discovery approach for WMNs using DNS-SD in combination with the advantages of the OLSR message distribution mechanism. We use caching techniques on the wireless mesh nodes realized through an OLSRD [4] plug-in, providing a cross-layer design, where the service discovery protocol benefits from routing layer information such as hop count or the ETX metric for the services.
This paper is organized as follows: In section 2 we review related work and existing protocols for service discovery. In section 3 we discuss service discovery in Wireless Mesh Networks, before our architecture is presented in Section 4. In section 5 we state the results from our implementation in the wireless mesh testbed. Section 6 comprises the conclusion.

2 Related Work

On the protocol level a lot of work has been done for service discovery in the internet, e.g. the Service Location Protocol (SLP) [2], Simple Service Discovery Protocol (SSDP) [5], DNS-based Service Discovery (DNS-SD) [6] together with multicast DNS (mDNS) [7] or Universal Plug and Play (UPnP) [3]. On the application level are the Java-based Jini [8] or UDDI (Universal Description, Discovery and Integration) for web services. These protocols are designed for wired infrastructure networks and are not well suited for Mobile Ad-hoc Networks (MANETs) or even Wireless Mesh Networks (WMNs), since these approaches are directory or simple flooding based.
A lot of work has also been done for service discovery in ad-hoc networks. Konark [9] is a service discovery mechanism on the application level for ad-hoc networks. mSLP [10] is an enhancement of SLP, where SLP Directory Agents (DAs) setup up a fully meshed structure and exchange service registration states. However, this approach does not scale well in WMNs, because service registration states must be replicated between all servers. This replication causes a high network load.
Another promising approach seems to be the integration of service discovery mechanisms in a routing protocol. In many cases the service requests are piggybacked on route request messages. Koodli and Perkins [11] propose a solution for service discovery in ad-hoc networks embedded in a reactive routing algorithm. They describe a service discovery mechanism in on-demand ad-hoc networks along with discovery routes to the service, for example with AODV or DSR.
However, most of the work in cross-layer design has been done for reactive routing protocols for efficiency reasons. On the level of proactive routing protocols are approaches for cross-layering routing and service discovery like [12] where the authors enable the Session Initiation Protocol (SIP) for MANETs with OLSR. The authors propose an approach called OLSR with Service Location Extension which is implemented in a simulator where servers regularly advertise their location or clients can query for server locations.

3 Service Discovery

In general, there are two architecture approaches to perform service discovery: directory based and non-directory based approaches. In case of the non-directory based approach, broadcasting or multicasting is used. Flooding techniques are only suited for small ad-hoc networks or networks with a reliable transport medium where broadcasting does not bother at all. The advantage of this architecture is that no administration is needed and that it is not dependent on infrastructure components like a central server.
In large infrastructure networks a local directory server is used for performance reasons. However, a directory server solution is also not suggested in WMNs, because all operations rely on a single point of failure. Service discovery operations are not possible if the central server or its wireless links are not available.
A widely used technique to optimize standard protocol effectiveness in wireless networks is called *cross-layer* design. When service discovery mechanisms are integrated within OLSR two approaches are possible: In the first case service announcements/replies are piggybacked on Topology Control information (TC) messages which are broadcasted in a regular interval. This is similar to the proactive routing approach. Here, a service record is immediately available and no service query needs to be initiated. Backward compatibility is not given, because all OLSR nodes must be capable to parse the modified TC messages.
In the second case a service discovery protocol is tunneled through OLSR. There is no piggybacking, because a new OLSR message type is defined for the service discovery messages. The normal query/advertisement phases of the original service discovery protocol remain and are separated from TC updates. Backward compatibility is given, because there is no change to the original protocols. OLSR is only needed to distribute and deliver the messages. This approach is well suited for a WMN where routing and non-routing clients seamlessly can operate service discovery. The approach of mDNS messages encapsulated in a new OLSR message type is discussed in the following.

3.1 DNS-based Service Discovery

DNS-based Service Discovery (DNS-SD) [6] offers clients the opportunity to discover a list of named instances of the desired service using only standard DNS-messages. DNS-SD can be used in combination with multicast, which is then called multicast DNS (mDNS) [7] or it can be used with any existing DNS server. Moreover, clients can register their services at a DNS server if the DNS server allows dynamic updates. DNS-SD is a widely used technique which is implemented in Apple's Bonjour Protocol [13].
Instead of requesting for "SRV" (Service) resource records, clients query for a "PTR" (pointer from one domain to another in the DNS namespace) record. In a second step, if the client selects a service instance, a query for the corresponding "SRV" record is processed. For DNS efficiency a server may place additional information in the answers, even if the client originally did not request it. This

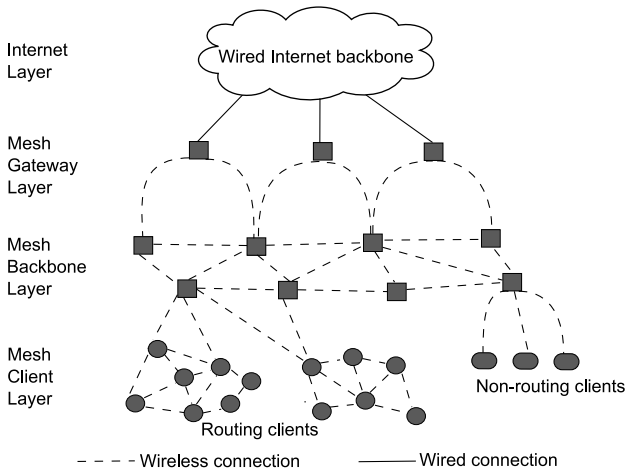


Fig. 1. Wireless Mesh Network architecture

may help to suppress the client's following request message. This response message includes the requested "PTR" record, the additional "SRV", "TXT" and all address records. A client request for `<Service>.<Domain>` will result in a list of available services like `Instance Name = <Instance>.<Service>.<Domain>`. The major advantage of using DNS-SD is that everything already exists. So there is no need to define a new protocol. The combination of DNS-SD and multicast is called Multicast DNS which is designed to be used in ad-hoc networks or wired infrastructure LANs, where no servers or manual configuration are needed.

3.2 Multicast DNS

Multicast DNS (mDNS) [7] is an IETF protocol which allows DNS operations on the local link without a DNS server. For this purpose mDNS uses a part of the DNS namespace, which is available for local use. mDNS needs little setup and even works if no infrastructure is available. Furthermore, mDNS does not change the existing standardized DNS message types such as Operation or Response Codes. The most important difference to classical unicast DNS is that the client sends its DNS message to a multicast address. mDNS works only on the local link, i.e. not beyond routing borders, since multicast is not forwarded by routers by default. The advantage of mDNS is the nature of multicast messages, i.e. all other clients can overhear the message and are able to detect possible conflicts.

4 Proposed Architecture

Our proposed architecture consists of wireless mesh routers, routing mesh clients and non-routing clients (see Fig. 1). Therefore, we propose an architecture which supports two different client access and service discovery modes: In the first mode

(described in Fig. 2) non-routing clients connect to the wireless mesh network through an access point. The clients do not implement any mesh routing protocol and they are running a downwardly compatible mDNS service discovery application. The second mode (described in Fig. 3) provides service discovery support for routing clients which implement OLSR and a mDNS service discovery application. The multicast DNS (mDNS) protocol is encapsulated in Optimized Link State Routing (OLSR) messages to make mDNS multi hop capable. When using OLSR for message propagation there is no need to setup and maintain multicast routing, like constructing multicast routing trees [14]. The presented application layer inspection of service discovery messages improves efficient message propagation. In this approach we use distributed DNS-SD caches called *Service Caches (SC)* on wireless mesh routers.
Since Optimized Link State Routing [15] (OLSR) is the most popular proactive routing protocol, we are using the OLSR implementation OLSRD [4] from UNiK for the WMN testbed and implementation.

4.1 Non-Routing Clients

Non-routing clients connect to the mesh network through an access point. They do not have access to information from the routing layer, because they are not implementing a routing algorithm. We distinguish between the legacy and the modified mDNS application. In the modified application, the client is able to control the Time-To-Live (TTL) value by using the first octet of the transaction ID of the DNS header to omit mesh network-wide flooding and traffic flow distortions caused thereby. This is possible, because the transaction ID is set to zero in the original mDNS protocol. For legacy clients who do not set the Time To Live value, the corresponding access point sets the value to 255. Clients can also advertise services, but service announcements are not routed and broadcasted into the mesh network. However, they are stored on the corresponding mesh gateway node with a *Service Proxy (SP)*.

4.2 Routing Clients

Routing clients are running OLSR and can directly access the internal tables from OLSR. They can extract metrics like hop count or *Expected Transmission Count (ETX)* for desired services. This is an advantage compared to a non cross-layering approach. Before a client uses a service, it can choose the most suitable service (*service selection*). This can be either the nearest service with respect to hop count or the service with the best ETX. A routing client can also advertise and browse for services.
Moreover, advertisements are possible for any service lease time within a flooding diameter ≤ 255. Mobile clients should only be allowed to advertise a service with a small lease time for their own services. For mobile nodes the lease time of the service must be kept small, because otherwise records of mobile nodes which already left the network are still stored. If a node turns off normally it sends an update with resource record live set to zero, which means that the resource

record is deleted. Mobile clients are limited in capacity and therefore have a smaller cache for services.

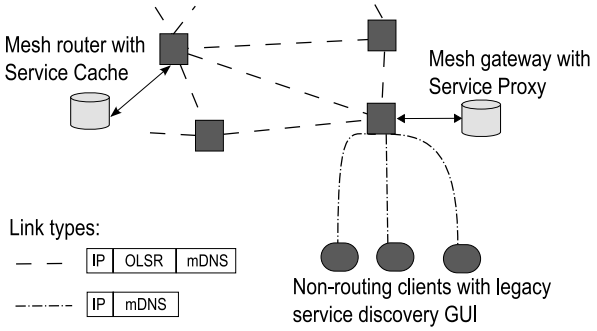


Fig. 2. Scenario A: service discovery support for non-routing clients

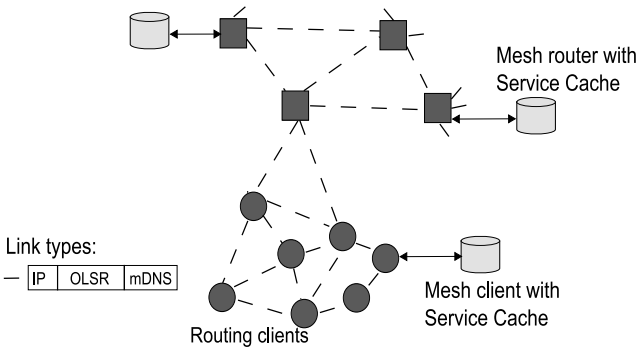


Fig. 3. Scenario B: service discovery support for routing clients

4.3 Mesh Router

When the plug-in is activated on a mesh router, the router can act as *Service Cache (SC)*. Mesh routers which do not run the plug-in forward the messages defined by OLSRD's default forwarding strategy. Instead of being complete directory servers, the SCs are self-organizing and do not need administrative effort. It is recommended to run the caches on fixed mesh nodes with a continuous power supply and enough memory and storage. If OLSR receives encapsulated mDNS messages, the plug-in extracts the service records and stores them in the cache. The corresponding service entry is deleted after the lease time of the service

record has expired.
A limitation of mDNS is that exact matching or complex queries like in SLP are not possible. A query is always domain based, which means that for example a cache responds with all 'printer' services and not only with the desired 'color printer'.
One important architectural decision concerns the answer behavior of the caches. In the original mDNS protocol only authoritative answers are allowed. This means that a cache must not answer with service records which are not running locally and for which the mDNS stack is not authoritative. For our architecture we decided to allow *non- authoritative* answering.
The strength of the non-authoritative answering behavior is that more services can be discovered within a smaller hop range. This is beneficial for mesh gateways with a service proxy. Clients connected to the mesh gateway can receive all necessary services within one hop.
We are also aware of cache replacement strategies [16]. However, this is not the focus of this paper and should not be considered further. We decided to use a simple cache replacement strategy: If the cache is full, the oldest entry is discarded.

4.4 Protocol Definition

In our approach we use the OLSR header to make mDNS multi hop capable. Therefore, every standard mDNS message gets a new OLSR header and its message handling is controlled by OLSRD. We now can also make use of OLSR's advanced flooding techniques called *Multi Point Relaying*. In the following we describe some OLSR message fields and their importance for our approach:

- **Message Type** is defined as *222*.
- **Time-To-Live (TTL)** states how many hops the message is allowed to be forwarded. The value is decremented by one every time the message is forwarded. If the value is zero, the message is discarded. A client sets this value to control the network search depth for discovering services.
- **Hop Count** is incremented by one every time the message is forwarded. This value is also displayed to the client together with the service to indicate how many hops the service is away.
- **Message Sequence Number** is incremented by one every time a node generates a message. This field is also used to discard duplicated messages.

We also need a hop control in non encapsulated mDNS messages which are created by non-routing clients. Therefore, we use the transaction ID field from the original DNS header. The transaction ID field is set to zero in the original mDNS protocol. For our approach we need a TTL field already in DNS, because otherwise the user application can not control the number of hops the message is allowed to be forwarded. This modification is only necessary for scenario A (see Fig. 2) where non-routing clients connect to an access point. If the service discovery application is directly integrated within OLSR (see scenario B), this is not needed. Routing clients can directly set the Time To Live field value of the OLSR header.

4.5 Message Propagation Strategies

In our architecture we implemented two message propagation strategies: The *Simple Flooding* mechanism is a straight forward way to propagate service discovery messages. When a client sends a query it can limit its query with the TTL message field of OLSR to a certain number of hops to avoid flooding of the whole network [1]. A receiving node answers with its services from the cache. Duplicate information about services is always forwarded and never discarded. Matching answers from the local cache are always sent regardless of any equivalent answers which were already forwarded.
In the *Discarding Duplicates* strategy each node holds an additional forward history hashtable indicating the last time the service record was forwarded. After the *HashValidity* time τ the entry is deleted from the table. All new incoming messages are discarded as long as there is an entry in the hashtable for the corresponding service. Answers from the local node are always sent with no respect for already seen answers from other nodes. Whereas OLSR discards only duplicate packets [15], our strategy is an application layer routing which inspects the message body and discards duplicate information which was sent by multiple nodes. The discarding strategy is only used on fixed mesh routers and not on mobile clients.

5 Results

5.1 Testbed

For our measurements we use our 36 nodes wireless mesh testbed. The testbed is located at the Department of Computer Science at RWTH Aachen University. The Department complex consists of one four- and two three-story buildings. The wireless mesh routers are distributed over different offices and floors inside the buildings. The mesh routers are single board computer (SBC) based on the WRAP.2E board by PC Engines [18] running on a minimal Ubuntu Linux. Each router consists of two WLAN IEEE 802.11a/b/g interfaces which are built on Atheros AR5213 XR chips, and two omnidirectional antennas. The first WLAN interface is tuned to channel 1 running in ahdemo mode [19] to connect to the mesh network. The second WLAN interface is used for client access and can be tuned to different channels dependent on interferences with other WLAN access points. All routers are also connected with an Ethernet interface for management reasons. Currently, we are running the pro-active and table-driven UniK OLSR daemon (OLSRD) [4] in version 0.5.4 as routing algorithm. Table 1 shows the classification of the wireless mesh routers of our testbed to scenario classes. The topology and the resulting number of neighbors for each router are given by the placement of the routers in the buildings. For more details and performance measurements about our testbed see [20].

[1] We assume that the TTL is set by the user manually or by some adaptive algorithm which proposes a feasible value, e.g. depending on the topology.

Table 1. Classification of the testbed mesh routers to scenario classes

Scenario class	Number of neighbors	Number of mesh routers
Sparse	$n < 5$	6
Medium	$5 \leq n \leq 10$	19
Dense	$n > 10$	11

5.2 Measurements

In the following measurements with OLSRD and our service discovery plug-in we are using hysteresis, which adds more robustness to the link sensing but delays neighbor registration. We are also using Multipoint Relaying (MPR) and default RFC [15] values for the message interval. We started the measurement after the warm-up phase of OLSR after the routes became mostly stable.

In our first measurement we want to investigate with which Time To Live a client has to send a query to discover as many services as possible. Therefore, we installed our plug-in on all mesh routers of the testbed. Every node loads a unique service at start-up which has to be discovered by our client. We investigated two scenarios: First, queries are initiated from within a dense topology and in a second measurement queries are initiated from within a sparse topology.

In the next step we started the mDNS client application on a laptop which sends its DNS-SD queries with unicast to OLSR on the same machine. The query is then encapsulated as OLSR packet and sent to the mesh network.

As Fig. 4 shows, the client discovers around 11 services within one hop in the dense scenario. This high number of services results from the high number of direct one-hop neighbors of the client. The measurement shows that about 3 to 4 hops are necessary to discover about 90% of our services in the testbed querying from a node which is located in a dense region. In the sparse topology scenario the client discovers only a few services within a small number of hops, because of a small number of direct neighbors. More services are discovered with a higher TTL. The result shows that the number of discovered services varies for the same number of hops within different measurement runs. Even though our mesh routers are fixed, the routes are dynamic and can change very often due to link quality issues. However, the number of discovered services per hop also depends on the topology from where the query is initiated.

In the second measurement we focus on the service discovery message overhead after a cold start and after a warm-up phase. In the first scenario (see Fig. 5(a)) the caches on 36 nodes perform a cold start where they load their unique local service. In the second step a laptop client sends a service request message which triggers all nodes to respond with cached services.

We now compare the Simple Flooding strategy against the Discarding Duplicates strategy with a HashValidity time $\tau = 10s$: With the Simple Flooding strategy the mesh routers have to handle a significantly higher number of messages than with the Discarding Duplicates strategy.

In the second scenario (see Fig. 5(b)) we directly continue working with the current state of the first scenario. Due to propagation, every cache has now all

36 services cached. This means all router caches have the same information. A simple query for the corresponding service class is leading to very high message overhead using the Simple Flooding strategy, because every node propagates its matching cache content through the whole mesh network.

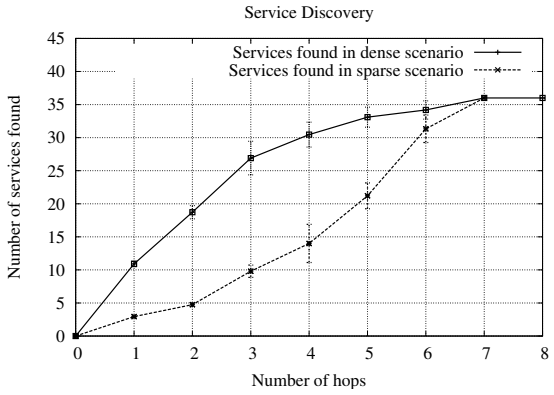
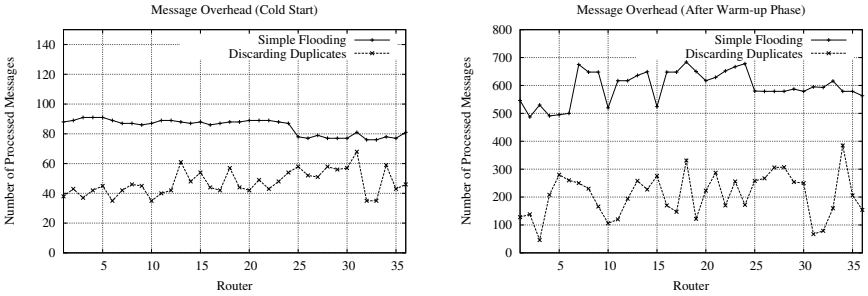


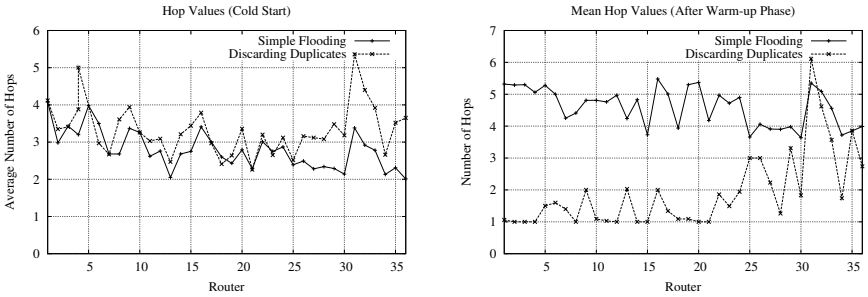
Fig. 4. Service discovery measurement



(a) After a cold start (b) After a warm-up phase

Fig. 5. Message overhead of all mesh routers

Figure 6(a) and 6(b) show the average hop count value of all received messages after a cold start and after the warm-up phase. The Discarding Duplicates strategy leads to a lower number of average hop count value after the warm-up phase, because all nodes receive answers now from their neighbors. In general, service discovery messages are not forwarded over many hops, because they are discarded if the same information was already sent in the last 10 seconds. For overall performance it is more efficient to receive service response message from



(a) After a cold start (b) After a warm-up phase

Fig. 6. Mean hop count values of received messages of all mesh routers

near SCs rather than from far away.
The Discarding Duplicates strategy improves the number of messages processed by every node compared to Simple Flooding. For further overhead reduction other techniques need to be applied.

6 Conclusion

In this paper we presented a unified approach for service discovery in Wireless Mesh Networks. Furthermore, we implemented this as a plug-in for OLSRD and presented our measurement results from our testbed. We encapsulated the messages from mDNS in OLSR packets. With this technique it is possible to use the multicast based service discovery protocol mDNS in a Wireless Mesh Network without a multicast routing protocol or the need to define a new protocol. This approach benefits from advanced OLSR flooding techniques like Multi Point Relaying or message sequence numbers. We showed that simple flooding of service discovery messages is not efficient and that the presented application layer inspection of messages can further improve message propagation with regard to message overhead and hop count.

Acknowledgment

This work was supported by the German National Science Foundation (DFG) within the research excellence cluster Ultra High-Speed Mobile Information and Communication (UMIC).

References

1. Akyildiz, I.F., Wang, X., Wang, W.: Wireless mesh networks: a survey. Computer Networks **47**(4) (2005) 445-487

2. Guttman, E., Perkins, C., Veizades, J., Day, M.: Service Location Protocol, Version 2. RFC Standard 2608 (1999)
3. UPnP Forum: Universal Plug and Play (UPnP) http://www.upnp.org/.
4. OLSRD: UniK OLSR Daemon (OLSRD) http://www.olsr.org/.
5. Cai, T., Leach, P., Gu, Y., Goland, Y.Y., Albright, S.: Simple Service Discovery Protocol/1.0. IETF Internet Draft (April 1999) work in progress.
6. Cheshire, S., Krochmal, M.: DNS-Based Service Discovery. IETF Internet Draft (August 2006) work in progress.
7. Cheshire, S., Krochmal, M.: Multicast DNS. IETF Internet Draft (August 2006) work in progress.
8. Sun Microsystems Inc.: Jini network technology http://sun.com/jini/.
9. Helal, S., Desaii, N., V.Verma, Lee, C.: Konark: A Service Discovery and Delivery Protocol for Ad-Hoc Networks. In: Proc. IEEE Wireless Communications and Networking Conference (WCNC 2003). (March 2003)
10. Zhao, W., Guttman, E.: mSLP - Mesh Enhanced Service Location Protocol Internet Draft draft-zhao-slp-da-interaction-07.txt.
11. Koodli, R., Perkins, C.E.: Service Discovery in On-Demand Ad Hoc Networks. IETF Internet Draft (October 2002) work in progress.
12. Li, L., Lamont, L.: Service Discovery for Support of Real-time Multimedia SIP Applications over OLSR Manets. OLSR Interop & Workshop 2004, San Diego, California (August 2004)
13. ZeroConf: Zero Configuration Networking http://www.zeroconf.org/.
14. Jia, W., Cheng, L., Xu, G.: Efficient multicast routing algorithms on mesh networks. ICA3PP'02: Proceedings of the Fifth International Conference on Algorithms and Architectures for Parallel Processing (2002)
15. Clausen, T., Jacquet, P.: Optimized Link State Routing Protocol (OLSR). IETF Experimental RFC 3626 (October 2003) http://rfc.net/rfc3626.txt.
16. Hu, Y.C., Johnson, D.B.: Caching Strategies in On-Demand Routing Protocols for Wireless Ad Hoc Networks. In: MobiCom'00: Proceedings of the 6th annual international conference on Mobile computing and networking
17. Chakraborty, D., Joshi, A., Yesha, Y.: Integrating service discovery with routing and session management for ad-hoc networks. Ad Hoc Networks **4**(1) (2006) 204-224
18. PCEngines: WRAP-Wireless Router Application Platform http://www.pcengines.ch/wrap.htm.
19. MADWiFi: Multiband Atheros Driver for WiFi (2007) http://madwifi.org/.
20. Zimmermann, A., Guenes, M., Wenig, M., Makram, S.A., Meis, U., Faber, M.: Performance Evaluation of a hybrid Testbed for Wireless Mesh Networks. In Proceedings of the 4st IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'07) (October 2007)