SIPHoc: Efficient SIP Middleware for Ad Hoc Networks

Patrick Stuedi Marcel Bihr Alain Remund Gustavo Alonso

Department of Computer Science, ETH Zurich
8092 Zurich, Switzerland
Contact Authors: {stuedip, alonso}@inf.ethz.ch

Abstract. Mobile Ad Hoc Networks (MANETs) offer a flexible way to connect mobile devices to build complex infrastructures. A key issue in MANETs is session set up and management since, unlike in conventional networks, there is no centralized component to provide such a service. Yet, session set up is necessary to provide any form of communication beyond unreliable, single message communication. In this paper we describe SIPHoc, a middleware infrastructure for session set up and management in MANETs. SIPHoc provides the same interface as the SIP standard but its implementation is fully decentralized. Moreover, SIP session establishment to and from the Internet is possible as soon as a single node in the MANET has Internet access. The paper presents the architecture and implementation of SIPHoc and evaluates its performance. The experiments show that SIPHoc is message efficient and provides a low dial-to-ring delay. SIPHoc allows SIP based applications to be used in MANETs without modification. In the paper, this is demonstrated by showing how SIPHoc supports VoIP conversations within a MANET and between the MANET and end-points on the Internet.

1 Introduction

Multi hop, mobile ad-hoc networks (MANETs) offer the opportunity to extend networks beyond the reach of fixed infrastructures. They also allow to build complex distributed systems using mobile devices. A key problem with ad-hoc networks is that their very nature makes it difficult to implement any form of communication requiring *sessions* rather than just best-effort dissemination of independent packets.

Session establishment and management are key procedures in conventional networks. There is even a standard, the *Session Initiation Protocol* (SIP) [20], that is used as a signaling protocol in applications such as Internet conferencing, telephony, and instant messaging. Being able to use SIP in MANETs would be a significant step towards turning MANETs into seamless extensions of conventional networks. The difficulty in doing so lies on the highly centralized architecture of SIP and the need to provide a seamless session connection from the MANET to the Internet. Numerous attempts have been made at adapting SIP to MANETs [16, 5, 12] and there are studies on the cost of running SIP on MANETs [2]. Unfortunately, none of these proposals has been implemented and often they work only on either isolated MANETs or MANETs permanently connected to the Internet. Moreover, all existing solutions impose limitations on the network topology and/or the routing protocol.

In this paper we present SIPHoc, a middleware platform for session establishment and management in MANETs that does not suffer from any of the limitations of previous proposals and has been implemented. SIPHoc does not require any centralized components, is message efficient (through routing message piggyibacking), and independent of the routing protocol (currently, SIPHoc supports both AODV [18] and OLSR [4]). SIPHoc does not impose any topology, and allows seamless interaction with the Internet (by treating nodes connected to the Internet as *gateway services* and making this information known across the MANET as a distributed service). Unlike previous work, SIPHoc is compatible with the SIP standard.

To demonstrate the potential of SIPHoc and its feasibility, we have used SIPHoc to provide a *Voice-over-IP* (VoIP) solution that supports VoIP conversations within a MANET and between the MANET and end-points on the Internet. As far as we are aware, SIPHoc is the only platform that allows VoIP applications to run unchanged whether on the Internet or over a MANET. In the paper we also use the VoIP application to study the performance of SIPHoc and show that the resulting overhead is close to optimal and comparable to that of standard operations on MANETs.

2 Related Work and Contributions

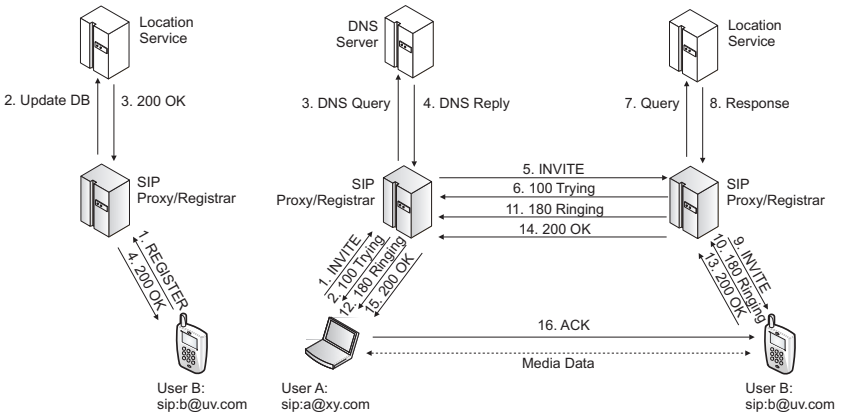
The problem of running SIP on MANETs has many facets. In this section we describe SIP, the design constraints for SIPHoc, and how SIPHoc differs from related work.

2.1 SIP Overview

SIP is a protocol for session initiation and tear-down. SIP works by building an overlay network on top of a regular IP network using a set of SIP entities such as proxies and registrar servers. A *SIP proxy* is an intermediary entity primarily in charge of routing: its job is to ensure that a request is sent to another entity 'closer' to the targeted user. A *SIP registrar* is used by SIP applications to register their current location. A *SIP location service* is used by registrars to store user location information and by SIP proxies to query user location information.

A SIP application first registers with the system by communicating its *SIP user name* and its current location to a *registrar* (Figure 1a). The registrar to be used is either determined using DNS or it is statically configured (through the so called *outbound-proxy*). Registrars and proxies are logical entities and it is not uncommon to have them co-located on the same node. When the registrar hears from a node, it builds a *binding*, i.e., an association between a SIP user name and the corresponding contact address (typically an IP address or resolvable name).

To establish a session with another user whose current location is unknown, a *SIP INVITE message* is sent to the proxy/registrar (step 1, Figure 1b). The proxy responds with a *100 Trying message* (step 2). The 100 Trying response indicates that the INVITE message has been received and that the proxy is trying to route the INVITE message to the final destination. Since the outbound proxy/registrar does not know the location of user B, it uses a DNS server to locate the proxy of the destination node and forwards the INVITE message accordingly (steps 3 to 6). The receiving proxy/registrar uses the



(a) SIP registration (b) SIP session setup

Fig. 1. Basic SIP mechanism

previously registered binding information of the user to locate the destination (steps 7 and 8) and finally delivers the INVITE message to the intended recipient (step 9). The recipient responds with a *180 Ringing message*, which is routed back through the two proxies in the reverse direction (step 10 to 12). If user B decides to establish the session with user A, it responds with a *200 OK* (step 13 to 15). Finally, user A sends an ACK message to confirm the reception of the 200 OK message (step 16). At this stage, the two users have learned each others' contact address through the INVITE/200 OK messages and from then on they communicate directly, bypassing the two proxies.

2.2 SIP Compatibility, Decentralization and Message Efficiency

In a MANET, SIP must operate in a fully distributed manner while maintaining the same interface and procedures. This is a challenge because several of the problems caused by the decentralized nature of MANETS can be solved by ignoring SIP procedures. For instance, registrars can be eliminated if all applications pro-actively announce their contact addresses. This can be done using either a HELLO method [16] or REGISTER broadcast messages [12]. Such approaches create a significant message overhead and introduces incompatibilities with session mobility and tear-down. As an optimization, [12] suggests to use the Service Location Protocol (SLP) [9] to discover the SIP bindings. Unfortunately, SLP is also centralized and very inefficient in MANETs due to the heavy use of multicast [2]. Existing work to make SLP more efficient in MANETs is highly routing protocol specific [13].

In SIPHoc we strictly follow the SIP interface and message flow. We have tested the compatibility with SIP by running KPhone (sourceforge.net/projects/kphone), Twinkle (www.twinklephone.com), Jain SIP Communicator (snad.ncsl.nist.gov/proj/iptel) and Linphone (www.linphone.org) on top of SIPHoc without any modifications. SIPHoc addresses the lack of centralized registrars by using SLP to dynamically discover the SIP outbound proxy. However, the SLP service we use is adapted to MANETs: it only

sends message by piggybacking them to the MANET routing messages. At the same time, SIPHoc does not depend on a particular routing protocol, rather new routing protocols can be easily incorporated as plug-ins to the platform.

2.3 Topology independence

SIP relies on DNS to locate nodes (steps 3/4 in Figure 1b). A way to bypass the lack of a DNS infrastructure in MANETs is to restrict the network topology and assign specialized roles to given nodes [5]. This approach makes the endpoint discovery process easier and eliminates the need for (part of) the registrar since the position and communication paths to all nodes are known in advance. This approach does not introduce message overhead but imposes strong restrictions on the routing protocols. It is also very difficult to efficiently maintain a fixed routing topology in mobile settings. An alternative approach to restricted topologies is to form node clusters that act as DNS surrogates within the MANET [22, 7]. However, electing and maintaining specialized nodes is difficult and expensive in MANETs. In fact, it has been shown [7] that in terms of transmitted messages, a fully distributed service discovery like the one used in SIPHoc outperforms the per-cluster approach, especially in mobile scenarios.

SIPHoc has been designed to be independent of the underlying network topology and, thus, it supports both static and mobile MANETs. As a result, SIPHoc avoids the problem of having to elect nodes for specialized tasks and replacing them when conditions change (e.g., when the node is switched off).

2.4 Connecting to and from the Internet

Connecting the MANET to the Internet poses several problems. First, the proxy does not know whether the target node is in the MANET or outside. Second, establishing sessions between nodes in and outside the MANET requires stable IP addresses. This can be done by using NAT [6] but it also requires additional mechanism like STUN [21]. If no NAT is available, extra mechanisms are needed to maintain consistent network addresses across connections of the MANET to the Internet.

A way to avoid such problems is to assume the MANET is permanently connected to the Internet. Then one can impose a fixed network topology leading to the gateway [3]. This approach does not work for establishing SIP sessions in MANETs not connected to the Internet and re-introduces the problems of fixed topologies. Fixed topologies can be avoided through dynamic gateway discovery [15]. There are also many proposals for routing traffic to the Internet from a MANET [23, 19]. Unfortunately, all such proposals are routing protocol specific and cannot be generalized. Routing independent approaches exist [14] but require gateway nodes to be located according to a fixed hierarchy. A more flexible design without topology assumptions is described in [10]. Similarly, there are proposals based on IPv6 [15] and Mobile IP [10, 23, 19]. Unfortunately, neither IPv6 nor Mobile IP are widely used today and are of no help in MANETs disconnected form the Internet.

SIPHoc has been designed such that seamless and transparent communication with the Internet is available across the MANET as soon as one node in the MANET has Internet connectivity. To maintain generality and to be applicable today, SIPHoc does

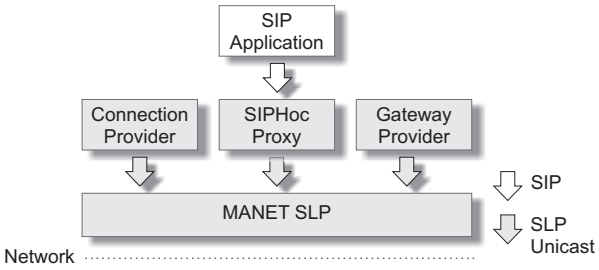


Fig. 2. Architecture Overview: *SIPHoc* Processes on a MANET node

not assume NAT, IPv6 or MobileIP. Moreover, any node in the network may act as a potential gateway: in SIPHoc gateways are established and discovered dynamically.

3 Architecture and System Components of SIPHoc

The SIPHoc architecture is shown in Figure 2. It is based on four components running as independent operating system processes within a node in the MANET. The components are divided into two groups, one for implementing SIP in the MANET, and one for connecting to the Internet.

- SIP in MANET components:
  - A *MANET SLP* layer providing a regular SLP (Service Location Protocol) interface but implementing efficient and decentralized service lookup functionality. It runs only on nodes where one of the other three components is present.
  - A *SIPHoc Proxy* with a standard SIP interface but implementing MANET specific functionality. It only runs on nodes with an end user applications. Each *SIPHoc Proxy* serves as an outbound SIP proxy for local SIP applications.
- Internet connectivity components:
  - A *Gateway Provider* that turns the node into a gateway if the node has Internet access. It is started on nodes willing to act as a gateway once they have access to the Internet.
  - A *Connection Provider* that manages connections of the node to the Internet when there is a gateway in the MANET. It is started on nodes willing to become Internet connected if chance comes up.

The *SIPHoc Proxy* is accessed by the application through the standard SIP protocol. The *Distributed Service Locator* process is accessed using SLP. All processes shown in Figure 2 can be started and stopped independently of each other.

4 SIPHoc components for running SIP in MANETs

The two key problems to solve in this section are how to replace the centralized location service by a distributed solution, and how to find other proxies without relying on DNS.

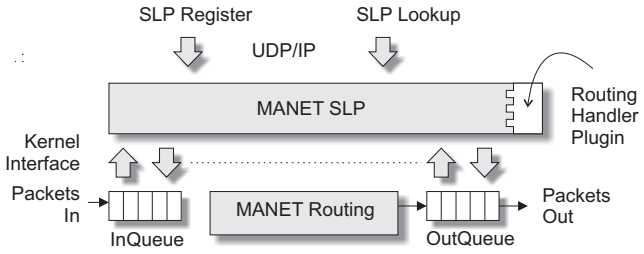


Fig. 3. MANET SLP architecture

4.1 MANET SLP

MANET SLP is a fully distributed service discovery platform for MANETs. MANET SLP provides a regular SLP interface over UDP for service registration and lookup (Figure 3). Services can be registered using the SLP *REGISTER* interface and looked up using the SLP *LOOKUP* interface. All services are *soft-state*, meaning that they expire after a specified period of time. The lifetime for locally stored services can be defined upon startup.

MANET SLP works by piggybacking sevice information onto routing messages (as also suggested in [24],[8]). This is done by capturing routing messages (using the *libipq* [1] library under linux) and extending them with service information. As pointed out above, this idea is not new. What is unique in MANET SLP is that the routing specific functionality is encapsulated within a *routing handler*. The routing handler is a software module that receives raw routing packets as input and generates altered packets that include the piggybacked service information. Whether the routing handler acts proactively and constantly disseminates information or only on demand (when a service is needed) depends on the underlying routing protocol. Which routing handler to use is decided at system startup. An additional advantage of this design is that, unlike in [13], routing protocols do not have to be modified to be used in SIPHoc.

4.2 SIPHoc Proxy

A standard SIP proxy/registrar accepts SIP registrations of a collection of users from certain domains. A SIPHoc proxy typically only accepts SIP registrations from users (applications) on that particular device. In addition to storing these registrations in its local location service table, each SIPHoc proxy uses MANET SLP to advertise itself as the contact address for these registered users (e.g., as the outbound proxy for those users). If a SIPHoc proxy receives an INVITE message and cannot find the target in its local location service table, it consults the MANET SLP layer and forwards the INVITE message to the proxy which was advertised as outbound proxy for this user (we call this procedure *dynamic outbound proxy selection*).

An alternative approach would have been to implement a distributed storage solution using MANET SLP where each proxy knows about all other SIP users in the

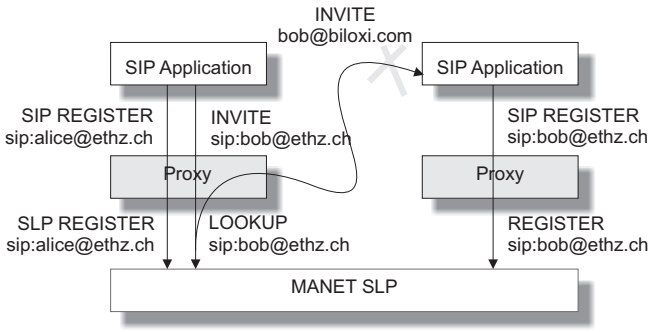


Fig. 4. Violation of the SIP message flow

MANET. This is very efficient but creates the same problems as the use of broadcast messages to register SIP applications across the MANET [16, 12] (Figure 4). If each proxy can find out the location of the final destination of the session by itself (through MANET SLP), SIP bindings are resolved directly at the caller's proxy and INVITE messages do not pass the callee's proxy. This violates the standard SIP message flow, where INVITE messages always pass through the proxy where the target is registered (Figure 1b). This violation creates problems with session mobility and session tear down.

In contrast to this, through dynamic outbound proxy selection, SIPHoc complies with the traditional SIP message flow. Moreover, the use of a proxy at each device creates an infrastructure that is by design fully decentralized.

4.3 SIPHoc Proxy Example

How the *SIPHoc Proxy* works can be best understood with an example (5). The example maps the standard SIP message flow shown in Figure 1 to the SIPHoc proxy and MANET SLP components just described. We assume two users Alice and Bob. The IP addresses of the two machines of Alice and Bob are $192.168.220.1$ and $192.168.220.2$ respectively. Each user runs a *SIPHoc Proxy* on port 5060 and a SIP application on port 5062. Both machines are in the MANET, within an arbitrary hop-distance from each other. *ProxyA* is the proxy used by Alice and *ProxyB* that of Bob.

To register with SIPHoc, both users send their URI and contact address to their proxies (1 and 5, Figure 5)): *sip:alice@ethz.ch* and ($192.168.220.1$:5062 for Alice; *sip:bob@ethz.ch* and $192.168.220.2$:5062 for Bob. The local SIPHoc proxy for each one of them will then store the corresponding entry in the local location service table (2,6). It will also contact the underlying MANET SLP and register the entry so that it is advertised by the MANET SLP module (3,4,7,8). Note that the entry in the local location service table differs from the entry sent to the MANET SLP in that the latter contains the contact address of the proxy (port 5060) rather than the one of the user (port 5062). To establish a SIPHoc session, assume Alice contacts Bob, Alice sends an INVITE message to *sip:bob@ethz.ch* (9). *ProxyA* checks whether the target SIP URI

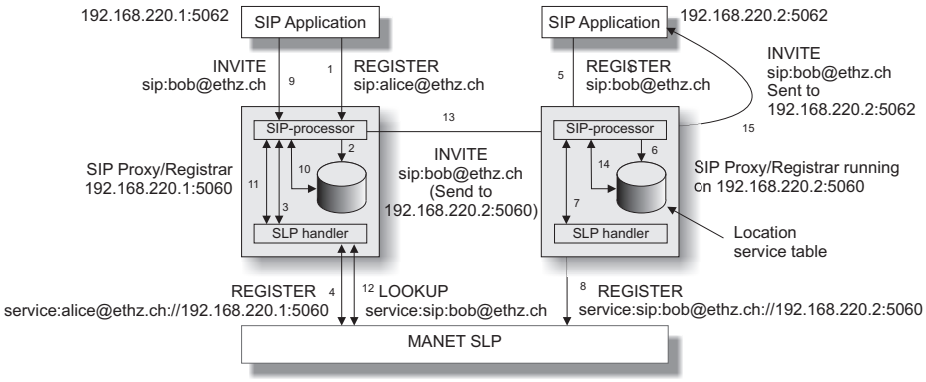


Fig. 5. SIPHoc Proxy/Registrar: dynamic outbound-proxy-selection

is in the local location service table (10). If that is not case, then it requests the entry from the MANET SLP layer (11-12). Once the contact address of the proxy of the user *sip:bob@ethz.ch* is found, the INVITE message is forwarded (13). There is no difference in whether a SIP proxy receives an INVITE message from the local user or from another proxy over the network. Hence, *ProxyB*, upon receiving the INVITE message, checks whether the requested SIP URI is available in its local location service table (14). In this case it will find the entry (registered by Bob as 192.168.220.2:5062), and can then forward the INVITE message to Bob (15).

4.4 SIPHoc over AODV and OLSR

When the SIPHoc proxy contacts the MANET SLP module, what happens depends on the underlying routing protocol. This is because, depending on how routing takes place, the MANET SLP will be either pro-active or act on demand. Currently we have implemented two different routing handlers. One for AODV [17] (on demand) and one for OLSR [4] (pro-active). As an example of the differences in terms of how services are found depending on the routing protocol, the message flow for session establishment in the case of AODV is shown in Figure 6 and for the case of OLSR in Figure 7. In an on demand routing handler (AODV), a lookup request from the SIPHoc proxy results in a *route request* from the routing protocol with the SIP information piggybacked to it [11]. In a pro-active routing handler (OLSR), the routing messages are exploited to constantly disseminate and maintain information across all MANET SLP modules. The advantage of the SIPHoc architecture (Figure 2) is that all these network properties are abstracted through the routing handler plug-in used in the MANET SLP layer and thus neither the SIPHoc proxy nor the SIP application are concerned with the routing protocol.

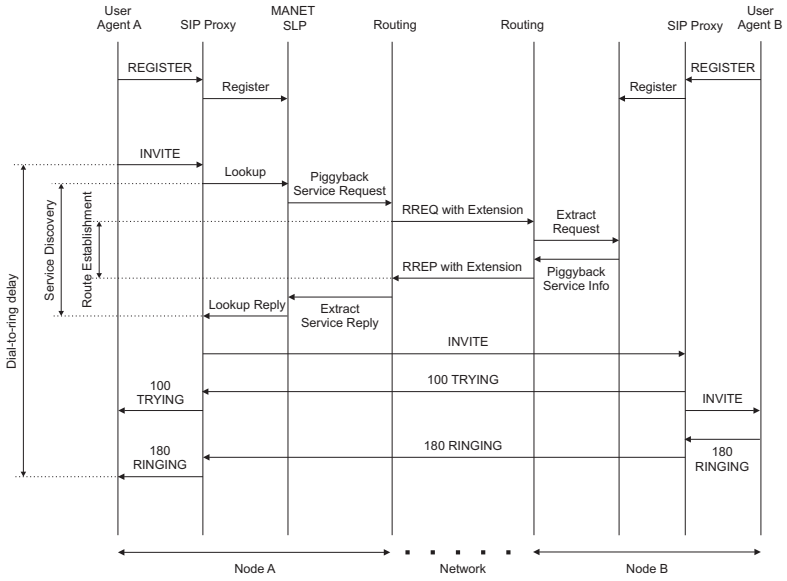


Fig. 6. SIP registration and call setup in a MANET using AODV

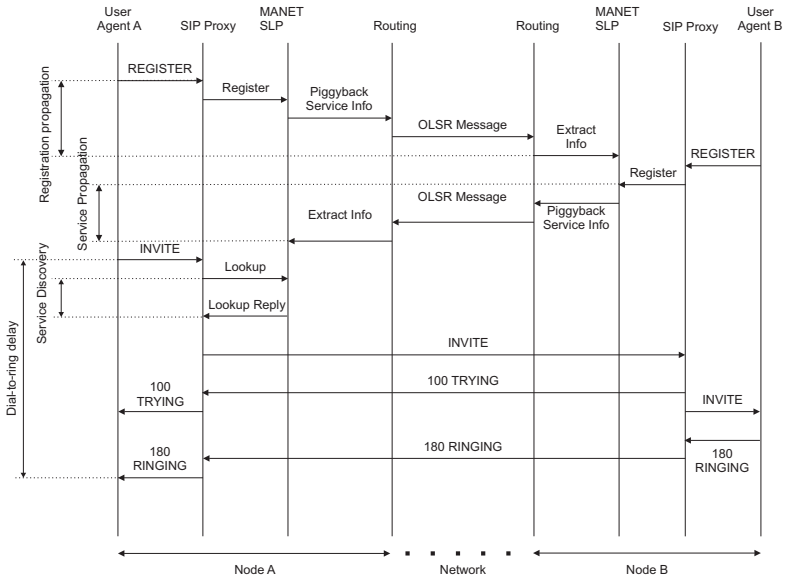


Fig. 7. SIP registrations and call setup in a MANET using OLSR

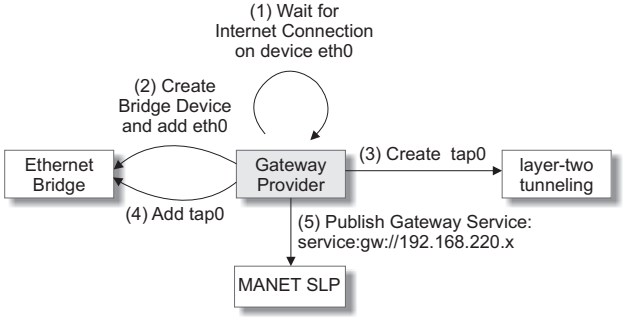


Fig. 8. Gateway Provider

5 Internet-connected MANETs

In this section we enhance the architecture described in the previous Section to support Internet connectivity.

5.1 Gateway Provider

A *Gateway Provider* is a process that can set up a node to become a *gateway* in case the node has Internet connection, and removes the *gateway* functionality in case the Internet connection is lost. *Gateway Provider* processes are started on nodes willing to act as gateways. A *gateway* is a node that is directly connected to the Internet and configured to provide Internet access to all the nodes within the MANET. Typically, these are nodes with multiple interfaces, since one interface is configured for communication with the MANET, and another interface is dynamically attached to the Internet. How a *Gateway Provider* works is shown in Figure 8. Once started, the *Gateway Provider* process keeps waiting for an Internet connection to become available (Figure 8, step 1). Detecting whether a node has Internet connection or not is done using a special *InternetDetectionAPI*. The idea is to exploit system support to efficiently detect a possible Internet connection. Our current implementation makes use of the operating system routing table. If an Internet connection has successfully been detected, say on interface *eth0*, the *Gateway Provider* process then creates a *bridge* device on that node and immediately adds interface *eth0* to the *bridge* (step 2). In a following step, a *layer-two tunnelling* device *tap0* is created (step 3) and also added to the *bridge* (step 4). The device *tap0* allows any node within the MANET to set up a *layer-two tunnel* connection to the gateway node. Since the tunnel device *tap0* is part of the *bridge*, traffic received on *tap0* is directly forwarded to the Internet via interface *eth0*. To provide MANET nodes with such gateway functionality, a *Gateway Provider* must however first register itself as a *gateway* service using the underlying *MANET SLP* service (step 5). Once a gateway service is registered, any node within the MANET may look up the gateway's location and connect to it. If the Internet connection is lost the service will be de-registered or will timeout, and both the tunnel endpoint *tap0* and the *bridge* will be removed.

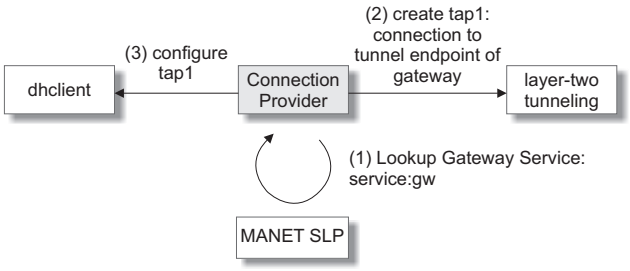


Fig. 9. Connection Provider

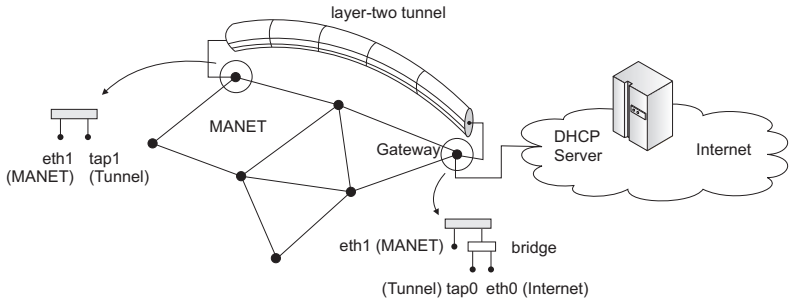


Fig. 10. Network state after the *Connection Provider* has successfully configured a node for Internet access

5.2 Connection Provider

A *Connection Provider* is a process that sets up an *Internet connection* if a gateway can be found. *Connection Provider* processes are started on nodes who want an Internet connection. How a *Connection Provider* works is shown in Figure 9. Once started, the *Connection Provider* process periodically searches for a gateway service by performing an SLP lookup request (Figure 9, step 1). If a gateway service can be found[1], a *layer-two tunnel* connection to the gateway is established (step 2). To finally configure the node for Internet access, a DHCP request is triggered on that newly established *tap1* interface (step 3). Since the *tap0* interface at the gateway node is bridged towards the Internet, the DHCP request will eventually be answered by the DHCP server that is reachable from the gateway node. The mechanism of IP configuration is encapsulated in an *IPConfiguration* module with a well defined interface. This allows the *Connection Provider* to easily adapt to other ways of IP configuration such as, e.g., IPv6 auto-addressing or MobileIP. After the IP configuration on the *tap1* device is done, the corresponding node is not only able to communicate with any node in the Internet, but nodes from the Internet may also transparently connect to that node within the MANET. A more detailed

[1] Currently, if multiple gateway services are found, the list of all gateways is passed to the tunnelling component

perspective on how components such as *bridge* and *layer-two tunnel* interfaces interact with each other is given in Figure 10.

The proposed mechanism differs from previous work for MANET-Internet connectivity in that it combines both a dynamic approach (through the use of a Gateway-Provider and a Connection-Provider) with a routing independent approach (through layer-two tunneling) while still being message efficient (due to MANET SLP).

5.3 Enhancing the SIPHoc Proxy

The ultimate vision of a SIP infrastructure that works in both isolated and Internet connected MANETs is that clients can use their officially registered SIP accounts[2] transparently in the MANETs. In other words, the only change should be the scope of the corresponding SIP URI which will vary depending on whether the MANET is currently connected to the Internet or not. For instance, assume again a user Bob in the MANET with SIP URI *sip:bob@ethz.ch*. Given that Bob's SIP account is officially associated with the SIP provider at *ethz.ch*, we would like calls to and from the Internet to become possible as soon as the MANET is connected. On the other hand, Bob should always be able to call any SIP user within the MANET – and vice versa – even if the MANET is currently disconnected from the Internet. To implement this vision of transparent and seamless SIP connectivity, two problems have to be solved: First, the proxy has to know whether the target node is in the MANET or outside. Second, the proxy has to make sure the contact address used during SIP session establishment is in fact reachable by the target user.

Let us first see how the proxy determines the SIP target. Suppose the MANET is currently connected to the Internet[3]. Besides registering users as described in section 4.2, the proxy now additionally forwards SIP REGISTER messages to the Internet if a responsible proxy for the specified domain is available. Upon receiving an INVITE message, the proxy immediately forwards the request to the Internet. If the INVITE process succeeds, the call is considered to be established. If the INVITE process fails – because either the target SIP URI is not registered with any proxy in the Internet, or the user is not online – the INVITE message will simply be forwarded to the responsible proxy in the MANET, if available. To minimize the call-to-ring delay, looking up the outbound-proxy is done concurrently with the forwarding of the INVITE message to the Internet. The workflow of a SIPHoc proxy as described is illustrated in Figure 11. The activities, highlighted grey and labeled "Continue as usual", refer to the procedures described in section 4.2

Let us now see how the proxy copes with SIP user contact addresses when having an Internet connection. Typically, nodes with Internet connection have multiple IP addresses assigned (see section 5.2). Let's call the IP address used in the MANET *internal* and the one used for communicating to the Internet *external*. SIP applications implement static binding, meaning that they use the IP address determined at startup time to be included in the contact address field of any SIP message sent towards the

[2] A SIP account associated with some official SIP provider in the Internet

[3] The SIPHoc proxy detects an Internet connection using the InternetDetectionAPI described in section 5.1

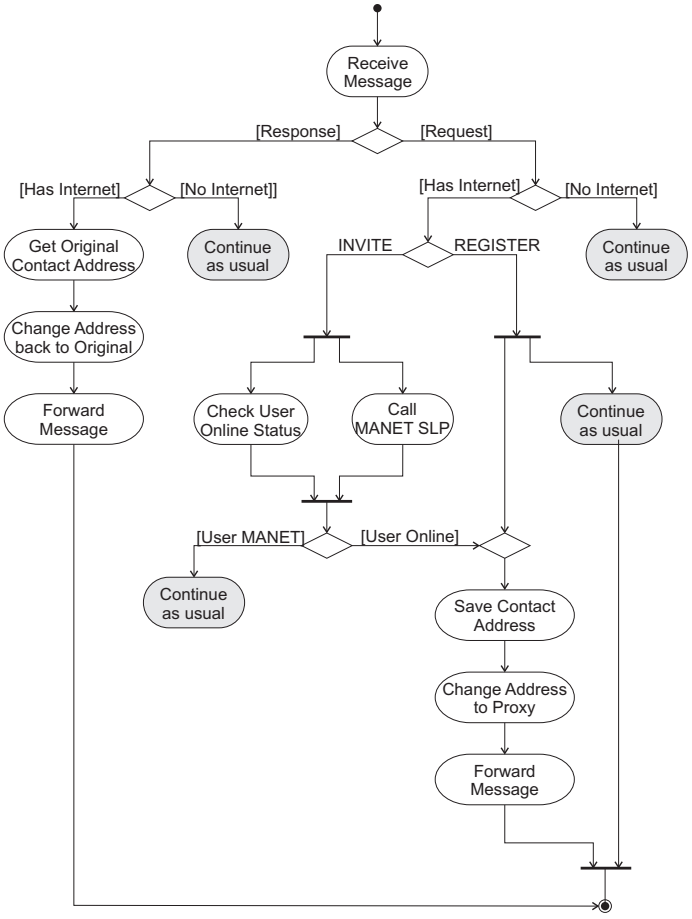


Fig. 11. Activity Diagram of the *SIPHoc Proxy* (SIP-Processor Component)

proxy (REGISTER, INVITE). Since the *external* IP address is configured dynamically when Internet connectivity is available, SIP messages would carry contact addresses pointing to an *internal* IP address. In practice, for connections to and from the Internet, one would however like to include the *external* IP address in the contact header of the SIP message because these addresses are used later by the application to establish the actual session. For the application to know which address to include in the contact header, it would have to know about its connection state (Internet, MANET) and the one of the target. Since we want out-of-the-box SIP applications to run transparently in heterogeneous MANETs without the application having to be modified, we propose a concept called **dynamic contact-address selection** to be implemented in the proxy. In Figure 11 this is illustrated by two additional activities before the final message forwarding. In the case of a SIP request (INVITE, REGISTER), the old contact address is

first saved. Then, the contact address is changed to the *external* IP address of the proxy. Once a response for a given request is received, the contact address is changed back to the original address that is used by the local SIP application (Figure 11). This keeps the application totally unaware of whether it communicates over the Internet or only within the MANET.

6 Case Study: VoIP in MANETs

6.1 VoIP

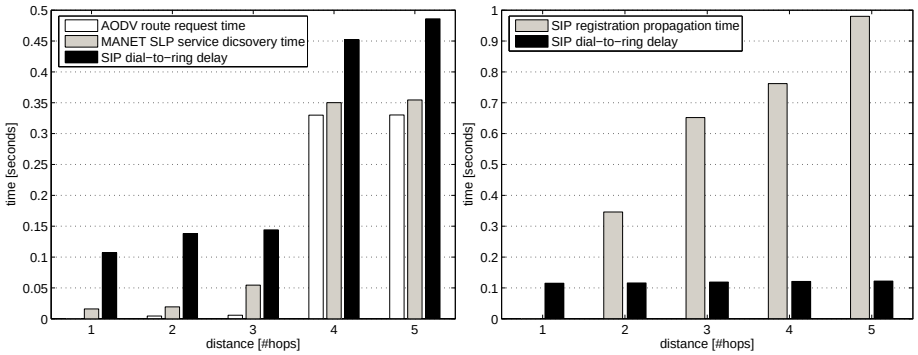
Since SIPHoc is strictly SIP compatible, it allows out-of-the-box SIP based VoIP applications to run transparently in MANETs. In this section we use Kphone as a VoIP application to evaluate our SIP infrastructure. However, we also have successfully tested SIPHoc with various other Softphones such as Linphone, Twinkle or Ekiga. From a user perspective, the metric of interest is the *dial-to-ring* delay, i.e., the time elapsed between the caller clicking the button on the calling terminal and the time the called party hears the ringing. This call setup includes a set of SIP messages (INVITE, TRYING, RINGING) as well as the associated routing messages with their service extensions embedded. The duration of the call setup depends on the current locations of caller and callee (MANET, Internet) and on the routing protocol used (AODV, OLSR). The following two sections evaluate the session setup time for a) pure Ad Hoc environments and b) Internet-connected Ad Hoc networks.

6.2 Experimental Setup

The measurements were done using 6 notebook computers running Debian 3.1 (Sarge). Five of them had a 2.0Ghz Mobile Pentium 4 and were equipped with an integrated 11Mbit/s IEEE802.11b wireless network interface card and were running kernel 2.6.8. The sixth laptop had a Pentium M processor with 1.73GHz combined with a 54Mbit/s IEEE802.11g wireless network interface card (used at 11Mbit/s through configuration) and was running kernel 2.6.11. Because it would be difficult to find a spatial separation of the notebooks which would have required multihop communication between them, an artificial separation using packet filter rules was used: They were only allowed to communicate with their direct neighbors , all other traffic was dropped (by the default policy). All results are averages over a set of 10 tests. The routing performance was measured by restarting the AODV daemon and issuing an ICMP packet to the target host afterwards, so the time required for the route discovery could be calculated from entries in the log of the AODV daemon. The same goes for the service lookup times where the SLP daemons were restarted after every lookup.

6.3 Performance in a MANET

In Figure 12a we study the session setup time for the AODV case and relate the results to both the route establishment and the service discovery time. The x-axis in Figure 12a refers to the number of nodes on the path (hop distance) between the caller



(a) AODV (b) OLSR

Fig. 12. Performance of SIPHoc in MANETs

and the callee. A first observation from Figure 12a is that looking up a service using MANET SLP only takes a few milliseconds longer than a simple route request for the same hop distance. Therefore, the price for piggybacking service information into routing messages is minimal. At the same time, MANET SLP reduces the time necessary to access a given service, since service lookup and route discovery take place simultaneously. As a direct conclusion from this, the SIP dial-to-ring delay is kept very low, only a few milliseconds more than the MANET SLP service discovery time. The overhead comes from the three additional messages involved in a SIP session setup (INVITE, Trying, RINGING). Please note that even if the SIPHoc proxy on the caller part would resolve the requested SIP URI by some magic oracle in zero time, the dial-to-ring delay would still be at least the sum of these 3 messages plus the route establishment time. Thus the measurements prove that SIPHoc reduces the dial-to-ring delay almost to the lowest value possible. The gap in time between three and four hops in Figure 12a is due to AODV's *expanded ring search* technique [17]. In an *expanding ring search*, the originating node initially uses a TTL equal 2 in the RREQ packet IP header and sets a timeout for receiving a RREP. If the RREQ times out without corresponding RREP, the originator broadcasts the RREQ again with the TTL incremented by 2 until a TTL threshold is reached[4]. In Figure 12a, nodes within a distance of three hops can be reached with a TTL value of 2 (one RREQ message), nodes within a distance of 4 and 5 hops, however, can only be reached with a TTL value of 4 (two RREQ messages) which leads to a gap in the route and discovery time.

While for the AODV case most of the overhead goes into dynamically looking up the outbound proxy during session establishment, the major overhead for the OLSR case is during the SIP registration phase. In Figure 12b we study the time needed for a SIP registration to be propagated across the network using OLSR. The x-axis of Figure 12b indicates the hop distance between the local proxy receiving the registration (source proxy) and the target. We see that even nodes being 5 hops apart from the source proxy receive the service information in less a second. Thus, already one second after a user

[4] Initial TTL value and increment can also be configured differently

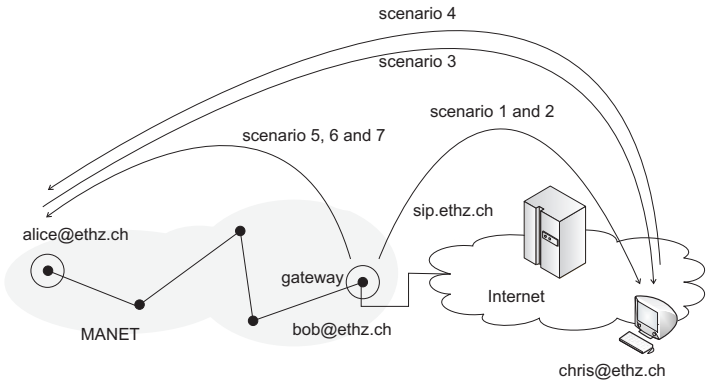


Fig. 13. Evaluation setup

has registered with its local proxy, all nodes within a hop distance of 5 nodes have the SIP binding of the given user accessible in their local MANET SLP process. Once the SIP binding of a user is available locally, the SIP dial-to-ring delay reduces to the time needed to send and receive the messages involved in a call setup (INVITE, Trying, RINGING), which is about 100ms (Figure 12b).

The conclusion from these results, both for AODV and OLSR, is that the *dial-to-ring* delay in an ad hoc network using *SIPHoc* is comparable to the route discovery time which is close to the lowest possible value. In this sense, SIPHoc is close to optimal.

6.4 Internet-connected environment: scenarios

The next experiment evaluates the dial-to-ring delay for Internet connections. The setup consists of 5 laptops[5] arranged to form a linear 4-hop network. On the MANET side, there is user Alice with no direct Internet access. User Bob is on the other extreme of the MANET and is on a node that – in some experiments – acts as gateway and has Internet access. A third user, Chris, is located in the Internet. We also assume a SIP proxy on *sip.ethz.ch* located in the Internet. We have evaluated the *dial-to-ring* delay in 7 scenarios as illustrated in Figure 13: (1) Bob calling Chris using an unmodified out-of-the-box SIP proxy[6] as a forwarding proxy for Bob; (2) Bob calling Chris using SIPHoc; (3) Alice calling Chris using Bob as a gateway; (4) Chris calling Alice using Bob as a gateway; (5) Bob calling Alice when Bob is disconnected from the Internet; (6) Bob calling Alice when Bob is connected to the Internet; and (7) Bob calling Alice when Bob is connected to the Internet and Alice is connected to the Internet using the gateway provided by Bob (therefore the INVITE message is routed through the Internet). We also include the cost of an AODV route request (8) for comparison.

[5] DELL Latitude, 2Ghz Intel Pentium IV, 256 MB RAM

[6] We have used the JAIN Proxy: http://snad.ncsl.nist.gov/proj/iptel

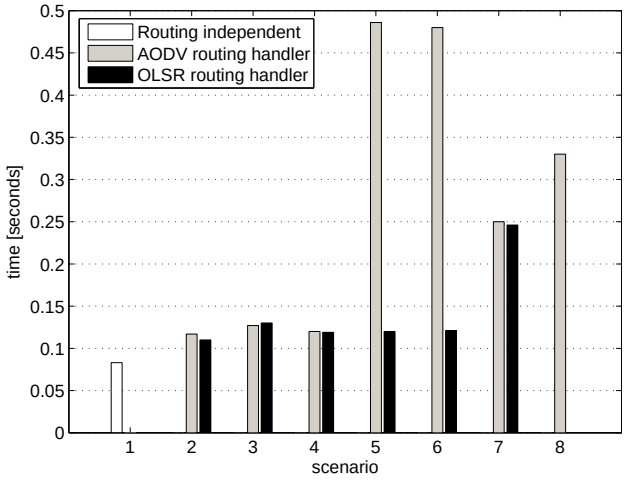


Fig. 14. *dial-to-ring* delay

6.5 Internet-connected environment: experiments

Each set of bars in Figure 14 corresponds to the result of one scenario, with the x-axis representing the scenario identifier. For all scenarios except the first one, we consider both the case where AODV is used as a routing protocol and the case where OLSR is used. Scenarios 1 and 2 are SIP connections over the Internet. They allow to determine the cost of SIPHoc over a plain SIP infrastructure. The results show that the overhead of SIPHoc (scenario 2) is small compared with the cost of using a plain SIP proxy (scenario 1). The overhead comes from the *SIPHoc Proxy* that performs various checks (looking up whether the node has Internet connection or not, preparing MANET SLP lookup) before forwarding the actual INVITE message to *sip.ethz.ch*. Since the target user Chris in scenario 1 is located in the Internet, the SIPHoc proxy invite processing succeeds without the MANET SLP layer being involved (Figure 11). Thus, there is no difference between the dial-to-ring delay for AODV and OLSR in scenario 1. Scenarios 3 and 4 are in the range of scenario 2 indicating that the 4-hop latency in the MANET is rather negligible compared to the whole session setup time. Furthermore, scenarios 3 and 4 confirm that both directions, MANET-Internet and Internet-MANET, perform similarly. The reason why there is no significant difference between the AODV and the OLSR case is that the route between Alice and the gateway has been established during the gateway discovery phase which took place asynchronously in a separate process (connection provider, Section 5.2) before the actual session establishment. In scenario 5, the target user cannot be found in the Internet, thus calls to the MANET SLP layer are necessary. If AODV and the corresponding MANET SLP routing handler is used, the dynamic discovery of the outbound-proxy (section 4.2) results in additional messages sent across the network, which increases the overall dial-to-ring time. The overhead is nevertheless in the range of a route request (scenario 8). This is because the outbound-

proxy lookup and the route discovery take place simultaneously. If OLSR is used in combination with the corresponding MANET SLP routing handler, no overhead at all is observed when comparing scenarios 5 and 6 with scenarios 3 and 4. This is because the SIP registration information has been propagated across the network pro-actively, piggybacked on routing messages. In scenario 6, note that the MANET is considered to be connected to the Internet. However, since Alice uses an official SIP account, but is not connected to the Internet, the *SIPHoc Proxy* receives an offline response from *sip.ethz.ch* and then proceeds as in scenario 5, leading to a similar result. Scenario 7 has an impressive performance considering that the call goes through 4 hops in the MANET, the gateway and the Internet consecutively. The call is faster than the calls in scenarios 5 and 6 since no MANET SLP lookup is involved, but takes longer than scenarios 3 and 4 because both endpoints use a *SIPHoc Proxy*. The results for AODV and OLSR again match quite well since the SIPHoc proxy succeeds immediately to establish a session over the Internet and no MANET SLP call is necessary.

The results have shown that the overhead for calls to and from the Internet is within an acceptable range. This is mainly due to the strategy described in section 5, after which the SIPHoc proxy always first tries to establish a SIP session over the Internet and only uses dynamic outbound-proxy discovery if necessary. Since the overhead of calls to the Internet is negligible compared to the overhead within the MANET, the overhead in case a given user cannot be found in the Internet is almost zero.

6.6 Gateway service

One important last question is how fast the gateway service can be discovered in the MANET. The answer depends on the routing protocol used. For the AODV case, the gateway service discovery time corresponds to the values for service discovery shown in Figure 12a. For the OLSR case, the gateway discovery time corresponds to the registration propagation time shown in Figure 12b.

To see whether tunnel maintenance affects the performance of the gateway, we have measured *dial-to-ring* delays for various setups with up to 1000 additional (idle) tunnel connections, without observing any recognizable slowdown.

We have also studied the packet overhead caused by the fact that the gateway is accessed through a layer 2 tunnel which wraps the packet and adds its own Ethernet and IP headers. The Ethernet header uses 14 bytes and the IP header 20 bytes. *openvpn*, used as tunneling application in our setting, sends its packets using UDP which adds another 8 bytes. This results in an overall overhead of 42 bytes per packet. Compared with the typical MTU of 1500 bytes per packet in Ethernet, an overhead of 42 bytes is almost irrelevant. For voice data, however, the audio data contained in a UDP packet is typically in the range of 160-172 bytes. A voice data packet on the wire without the tunneling overhead would therefore have a size of 214 bytes. With the tunnel header, the size of each packet increases to 256 bytes. This is an overhead of 20%. If only a few nodes in the network communicate through the gateway using tunneling this will not have a big effect. If the network is big and most of the users communicate through the gateway to the Internet, an overhead of 20% may decrease the available capacity.

6.7 Discussion

We have shown in sections 6.3 and 6.5 that SIPHoc provides a very efficient SIP middleware infrastructure for VoIP applications in both isolated and Internet connected MANETs. Our experiments illustrate that SIPHoc is able to establish sessions within a few hundred milliseconds, regardless whether the other party is located in the MANET or in the Internet. Due to the efficient architecture of SIPHoc, the overhead in the dial-to-ring delay stays in the order of a route discovery time, which is the lowest value possible since the cost of a route discovery has to be paid anyway.

7 Conclusion

In this paper we have presented *SIPHoc*, a middleware providing SIP-compatible session establishment in both isolated and Internet-connected MANETs. The advantages over existing work are that *SIPHoc* does not impose any network topology, does not involve any centralized components, does not require the modification of existing standards, is message efficient, and has an overhead comparable to MANET route requests. In the paper we have used VoIP to show how SIPHoc can be used by SIP-compatible applications to transparently establish sessions with other parties either located within the MANET or in the Internet. To the best of our knowledge, *SIPHoc* is the first complete implementation of a SIP infrastructure for MANETs and the only one that resolves all the limitations of existing work.

Acknowledgement

The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

References

1. The Netfilter/Iptables Project, June 2001. http://www.netfilter.org.
2. H.-G. S. amd Ki-Hyung Kim, W.-D. Jung, and J.-S. Park. Performance of service location protocols in manet based on reactive routing protocols. In *Proceedings of 4th International Conference on Networking (ICN'05)*, 2005.
3. J.-J. Chen, Y.-L. Cheng, Y.-C. Tseng, and Q. Wu. A push-based voip service for an internet-enabled mobile ad hoc network. In *Proceedings of 3rd IEEE VTS Asia Pacific Wireless Comm. Symposium (APWCS'06)*, 2006.
4. T. Clausen, P. J. (editors), C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L.Viennot. Optimized link state routing protocol (olsr). RFC 3626, October 2003. Network Working Group.
5. A. Dutta, R. Jain, K. Wong, J. Burns, K. Young, and H. Schulzrinne. Multilayered mobility management for survivable network. In *Proceedings of MILCOM*, 2001.
6. K. Egevang and P. Francis. The IP Network Address Translator (NAT). RFC 1631 (Informational), May 1994. Obsoleted by RFC 3022.

7. P. Engelstad and Y. Zheng. Evaluation of service discovery architectures for mobile ad hoc network. In *WONS '05: Proceedings of the Second Annual Conference on Wireless On-demand Network Systems and Services*. IEEE, 2005.
8. C. Frank and H. Karl. Consistency challenges of service discovery in mobile ad hoc networks. In *MSWiM '04: Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 105–114, New York, NY, USA, 2004. ACM Press.
9. E. Guttman, C. Perkins, J. Veizades, and M. Day. Service Location Protocol, Version 2 . RFC 2608 (Proposed Standard), June 1999. Updated by RFC 3224.
10. U. Jönsson, F. Alriksson, T. Larsson, P. Johansson, and J. Gerald Q. Maguire. Mipmanet: mobile ip for mobile ad hoc networks. In *MobiHoc '00: Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, pages 75–85, Piscataway, NJ, USA, 2000. IEEE Press.
11. U. 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):23–44, January 2004.
12. S. Leggio, J. Manner, A. Hulkkonen, and K. Raatikainen. Session initiation protocol deployment in ad-hoc networks: a decentralized approach. In *2nd International Workshop on Wireless Ad-hoc Networks (IWWAN)*, 2005.
13. L. Li and L. Lamont. A lightweight service discovery mechanism for mobile ad hoc pervasive environment using cross-layer design. In *Percom'05: Proceeding of the 3rd International Conference on Pervasive Computing and Communications Workshops*, 2005.
14. M. Michalak and T. Braun. Common gateway architecture for mobile ad-hoc networks. In *WONS'05: Second Annual Conference on Wireless On demand Network Systems and Services*, 2005.
15. A. Nilsson, C. E. Perkins, A. J. Tuominen, R. Wakikawa, and J. T. Malinen. Aodv and ipv6 internet access for ad hoc networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(3):102–103, 2002.
16. M. O'Doherty. Pico sip. Internet Draft, February 2001.
17. C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Experimental), July 2003.
18. C. Perkins and E. Royer. Ad hoc on-demand distance vector routing. *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, Feb 1999.
19. C. E. Perkins. Mobile-ip, ad-hoc networking, and nomadicity. In *Proceedings of International IEEE Computer Software and Applications Conference (COMPSAC'96)*, volume 00, page 0472, Los Alamitos, CA, USA, 1996. IEEE Computer Society.
20. J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261 (Proposed Standard), June 2002. Updated by RFCs 3265, 3853.
21. J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). RFC 3489 (Proposed Standard), Mar. 2003.
22. F. Sailhan and V. Issarny. Scalable service discovery for manet. In *Proceedings of the 3rd IEEE international Conference on Pervasive Computing and Communications (PerCom'05)*. IEEE, 2005.
23. Y. Sun, E. M. Belding-Royer, and C. E. Perkins. Internet connectivity for ad hoc mobile networks. *International Journal of Wireless Information Networks special issue on "Mobile Ad Hoc Networks (MANETs): Standards, Research, Applications"*, 9(2), April 2002.
24. C. Ververidis and G. Polyzos. Extended zrp: a routing layer based service discovery protocol for mobile ad hoc networks. In *Mobiquitous '05: Proceedings of the Second Annual International Conference on Mobile and Ubiquitous Systems*. IEEE, 2005.