RC23270 (W0407-032) July 13, 2004
Computer Science

IBM Research Report

Enabling SIP-Based Session Setup in Ad Hoc Networks

Nilanjan Banerjee*, Arup Acharya, Sajal K. Das*
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

*CReWMaN
University of Texas at Arlington

Enabling SIP-Based Session Setup in Ad Hoc Networks

Nilanjan Banerjee
CReWMaN, University of Texas at Arlington
Arup Acharya
IBM T. J. Watson Research Center
Sajal K. Das
CReWMaN, University of Texas at Arlington

***Abstract*— The deployment of infrastructure-less ad hoc networks is suffering from the lack of applications inspite of active research over a decade. This problem can be alleviated by porting successful legacy Internet applications and protocols to the ad hoc network domain. Session Initiation Protocol (SIP) is designed to provide the signaling support to multimedia applications such as Voice over IP (VoIP), Instant Messaging etc. However, SIP relies on an infrastructure for service discovery as well as message routing, which is unavailable in ad hoc networks. In this paper, we propose two approaches to solve this problem and enable SIP-based session setup in ad hoc networks (i) a loosely coupled approach where the SIP service discovery is decoupled from the routing procedure and (ii) a tightly coupled approach which integrates the service discovery with a fully distributed cluster based routing protocol that builds a virtual topology for efficient routing. Simulation experiments show that the tightly coupled approach performs better for (relatively) static multihop wireless networks than the loosely coupled approach in terms of the latency. The loosely coupled approach, on the other hand, performs better in networks with random node mobility. The tightly coupled approach, however, has lower control overhead in both the cases.**

***Index Terms*— Session Initiation Protocol, Ad Hoc Networks, Ad Hoc Network Routing, Service Discovery**

I. Introduction

The rapid development of small, cheap and computationally powerful devices and major advancement in short range wireless communication technologies have increasingly made it possible to build scalable efficient ad hoc networks. The last few years have seen vigorous research primarily in ad hoc network routing protocols [11], [16], [23], [28], [29], [30], but the lack of applications in ad hoc domain has been a major impediment for the deployment and wide acceptance of ad hoc networks. One way to alleviate this problem is to support legacy Internet applications in the ad hoc domain along with the newly developed applications. Moreover, as the notion of ubiquitous computing [37] is gaining momentum with the increasingly pervasive nature of the mobile devices and wireless technology, the convergence of fixed mobile networks and infrastructure-less ad hoc networks [21] seems inevitable entailing the extension or adaptation of key legacy protocols of fixed mobile networks to ad hoc networks, as well.

Signaling protocols, developed for establishing multimedia sessions such as a VoIP with stringent resource requirements in the Internet, is one such important legacy protocol. Signaling protocols negotiate resources between the terminals and maintain them throughout the duration of a multimedia session. The two most prominent signaling protocols for IP based networks are H.323 [13] from International Telecommunication Union (ITU) and Session Initiation Protocol (SIP) [32] from IETF. It seems SIP is progressively gaining popularity over H.323, primarily because of its simplicity and flexibility. Moreover, some of the features of SIP, such as redirecting a call and proxying, can be potentially applied to wireless networks with mobile nodes. Recently SIP has found its application in the context of ubiquitous computing [4]. Ad hoc networks, being a key technology in ubiquitous computing, need to also support SIP to enable such applications. Apart from these, several useful SIP based services such as Instant Messaging, Presence have the potential of being utilized effectively in ad hoc networks.

However, SIP relies on an infrastructure heavily borrowed from the Internet infrastructure for SIP service discovery and thus cannot be used as is in ad hoc networks. Otherwise, SIP being an application layer protocol, follows true end-to-end semantics and can establish direct client-to-client sessions, provided the

This work was supported by NSF under the ORBIT testbed project, grant# NSF NRT Project #ANI-0335244 and by NSF ITR grant IIS-0326505.

clients can reach each other.

In this paper we investigate how SIP can be effectively used in ad hoc networks. Two possible approaches have been proposed, *viz.* a *loosely coupled approach* and a *tightly coupled approach*. In the former approach, SIP service discovery is decoupled from the underlying ad hoc routing protocol, whereas in the tightly coupled approach the service discovery is integrated with the routing protocol. While we use a simple expanding broadcast based scheme for the loosely coupled approach, we have proposed a distributed cluster based routing protocol for the purpose of integration with the service discovery in the tightly coupled approach. Simulation based experimental results indicate that the tightly coupled approach performs better for (relatively) static multihop wireless networks. However, in a network with random node mobility loosely coupled approach fares better.

The rest of the paper is organized as follows. Section II presents a brief overview of SIP. The issues related to supporting SIP in ad hoc networks are discussed in Section III. Section IV discusses the earlier research efforts to support SIP in ad hoc networks. The loosely coupled approach is described in Section V and the tightly coupled approach is described in Section VI. Some of the important properties of the routing protocol in the tightly coupled approach is described in Section VII. The comparative performance evaluation of the two approaches is presented in Section VIII along with related discussions in Section IX. Finally, Section X concludes the paper.

II. Overview of SIP

SIP is a control protocol that allows creation, modification and termination of sessions with one or more participants. SIP is used for voice and video calls either for point-to-point or multiparty sessions. It is independent of the media transport which for example, typically uses Real-time Transport Protocol (RTP) over UDP [33]. SIP is also used for Instant Messaging and presence detection [25]. It allows multiple end-points to establish media sessions with each other: this includes terminating the session, locating the end-points, establishing the session and then, after the media session has been completed. In recent times, SIP has gained widespread acceptance and deployment among wireline service providers for introducing new services such as VoIP; within the enterprises for Instant Messaging and collaboration; and amongst mobile carriers for push-to-talk service. Industry acceptance of SIP as the protocol of choice for converged communications over IP networks is thus highly likely. As shown in Figure 1, a SIP infrastructure consists of user agents, registration servers, location servers and SIP proxies deployed across a network. A user agent is a SIP endpoint that identifies services such as controlling session setup and media transfer. User agents are identified by SIP URIs (Uniform Resource Identifier), which is a unique HTTP-like URI of the form `sip:user@domain`. All user agents REGISTER its IP address with a SIP registrar server (which can be co-located with a SIP proxy). Details of the SIP protocol can be found in [32]. SIP defines a set of messages, such as INVITE, REFER etc., to setup sessions between user agents. These messages are routed through SIP proxies that are deployed in the network. DNS Service records help in finding SIP proxies responsible for the destination domain.

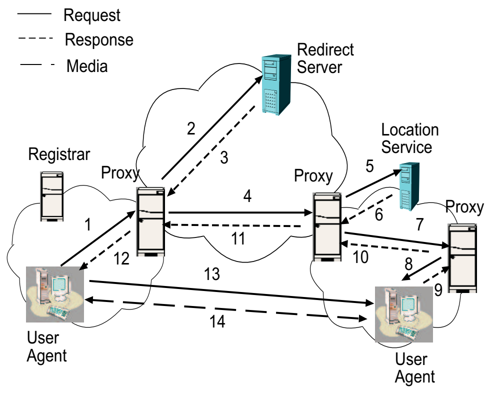


Fig. 1. SIP architecture

A session is setup between two user agents following a client-server interaction model, where the requesting user agent acts as the client and is known as the user agent client (UAC), interacting with the target user agent known as the user agent server (UAS) acting as server. All requests from an originating UAC, such as an INVITE are routed by the proxy to an appropriate target UAS, based on the target SIP URI included in the `Request-URI` field of the INVITE message. Proxies may query location and redirect servers for SIP service discovery or in order to determine the current bindings of the SIP URI. Signaling messages are exchanged between user agents, proxies and redirect/location servers to locate the appropriate services or endpoints for media exchange. For reasons of scalability, multiple proxies are used to distribute the signaling load [14]. A session is setup between two user agents through SIP signaling messages comprising of an INVITE (messages 1,2,4,7, and 8 in Figure 1), an OK response (messages 9-12 in Figure 1) and an ACK (message 13 in Figure 1) to the response [32]. The call setup is followed by media exchange using RTP. The session is torn down through an exchange of BYE and OK messages.

SIP distinguishes between the process of session establishment and the actual session. A basic tenet of SIP is the separation of signaling (control) from media. Signaling messages are usually routed through the proxies while the media path is end-to-end. The session setup messages like INVITE contain user parameters using Session Description Protocol (SDP) [12] in the message body. SDP provides information about the session such as parameters for media type, transport protocol, IP addresses and port numbers of endpoints. The IP address and port numbers exchanged through SDP is used for the actual data transmission (media path) for the session. Any of these parameters can be changed during an ongoing session through a RE-INVITE message, which is identical to the INVITE message except that it can occur within an existing session.

III. SIP IN AD HOC NETWORKS

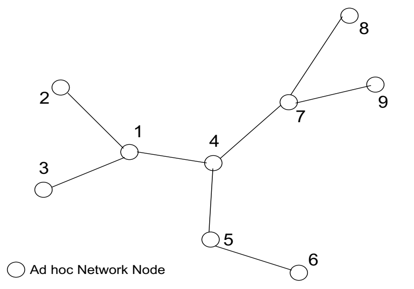


Fig. 2. An example of an ad hoc network

Ad hoc networks are in general considered to be a collection of wireless mobile nodes that dynamically forms an infrastructure-less network. Each node in such a network acts as a router and can forward or receive packets to nodes within the radio range. Figure 2 shows an example of an ad hoc network with 9 nodes where the links identify the pairs of nodes that are within each others radio range. The nodes are identified either by IP address or by internal address and the routing protocols take the responsibility of sending packets to these nodes once their addresses are known. As described earlier, SIP makes use of an infrastructure of registrars, location database and proxies to locate or discover the SIP end points and route the SIP messages for session setup. Unfortunately, this infrastructure is unavailable in ad hoc networks and an auxiliary mechanism is required to discover the SIP end points before the routing of SIP messages can be taken care of by the ad hoc routing protocols. In particular, the node address corresponding to the SIP URI of the `Request-URI` field in the INVITE message from a requesting UAC needs to be resolved before the routing of SIP messages can happen. There are potentially two approaches to do this: (i) *a loosely coupled approach* (LCA), where the SIP end point discovery is decoupled from the ad hoc routing protocol and (ii) *a tightly coupled approach* (TCA), where the SIP end point discovery is integrated with the ad hoc routing protocol. Figure 3 shows the functional diagrams of these two approaches.

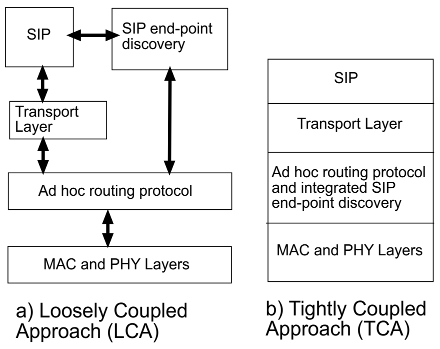

SIP
SIP end-point discovery
Transport Layer
Ad hoc routing protocol
MAC and PHY Layers
a) Loosely Coupled Approach (LCA)
SIP
Transport Layer
Ad hoc routing protocol and integrated SIP end-point discovery
MAC and PHY Layers
b) Tightly Coupled Approach (TCA)

Fig. 3. Loose and tight coupling of SIP end-point discovery with ad hoc network protocols

A. Routing Protocol Consideration

There are pre-dominantly two types of ad hoc routing protocols. They are *proactive routing strategy* and *reactive routing strategy*. In a proactive strategy, the results are computed based periodic advertisements and stored for future use. A reactive strategy, on the other hand, computes the routes when required by flooding the network with probe packets. A proactive strategy is capable of producing routes faster than the reactive strategy at the cost of maintaining pre-computed but sometimes redundant and spurious routes. Examples of proactive routing protocols are DSDV [29], OLSR [5] and that of reactive routing protocols are DSR [16], AODV [30]. Both the proposed LCA and TCA can be potentially integrated with either type of routing protocol. Several research studies [19], [15] have established the edge of reactive protocols over the proactive ones, particularly for highly dynamic networks. Proactive routing protocols in such networks suffer from high overheads and low convergence rates. However, reactive strategy can also suffer from prohibitive flooding traffic attributed to the redundancy factor associated with the “broadcast storm problem” [36] and unacceptable delay in route discovery process. A trade-off is generally done in such cases with cluster based routing [3], [6], [9], [18], [20]. In cluster based routing, several clusters are formed with the ad hoc nodes, each with a cluster head that is fully aware of all the other members of the respective cluster and is responsible for communication to them. Flooding of control packets and routing of data packets take place through the cluster heads only, thus restricting the flooding problem. Cluster based routing essentially creates a virtual topology with the cluster heads forming the backbone network. The virtual topology can be effectively used to provision for specialized SIP based services such as conferencing, which requires certain infrastructure entities like the conferencing server, proxies, etc. Since the advantages of the cluster based routing can be exploited only when the SIP end point discovery is integrated with the routing protocol, we have proposed an integrated fully distributed cluster based routing protocol for TCA and have used the reactive protocol, AODV as the underlying protocol for LCA.

IV. Related Works

SIP was originally intended for multimedia session setup in the Internet, hence not much work has been reported to support SIP in ad hoc networking domain. An early attempt was made [27] to extend SIP so that it can be used in ad hoc networks. A pro-active mapping of all the SIP clients in a network was maintained in each node by using a HELLO method, defined as an addendum to the already existing SIP methods. But, this proactive mapping may not be scalable and also incurs unnecessary control overhead.

SIP based mobility management in ad hoc networks was considered in [7]. But, the authors have assumed a hierarchy of nodes with gateways in the ad hoc network. Hence they did not exactly deal with the issues of SIP end point discovery and the following session setup in a purely infrastructure-less network. A truly ad hoc network was considered in [22] to evaluate SIP based mobility management, but a directory based SIP end point discovery procedure was proposed much like that of the pro-active scheme [27] and hence suffers from the same drawbacks of scalability and high control overhead.

SIP end point discovery is essentially similar to the service or peer discovery process in peer-to-peer (P2P) networks. A P2P network is generally constructed as an overlay network over the Internet and the service or peer discovery process involves the discovery of a particular service or the contact information of a peer without the use of any Internet routing infrastructure. There are three main approaches of such service or peer discovery in P2P networks: (i) a centralized approach, (ii) flooding based approach, and (iii) a distributed hash table based approach. The first one is a typical "phone book" approach where an indexing of all the services and the peers is maintained in a centralized server. Systems like Napster [24], Skype [34] followed this approach. These systems are often non-scalable and have a single point of failure. Systems such as Gnutella [8] use a flooding based approach where the requests are flooded through the neighboring peers in the networks until the service or peer discovery is done. As we shall see later, this approach is often non-scalable in terms of the number of messages. The distributed hash table approach, such as the Chord [35] protocol, creates a highly scalable structured overlay using hash tables to map the services and peers to the respective contact information. However, this approach is not particularly suitable for mobile ad hoc networks.

V. LCA: Loosely Coupled Approach

LCA employs a similar technique that ad hoc on demand distance vector (AODV) routing protocol uses to discover route to a given destination IP address. It defines two types of messages *viz.* SIPRREQ and SIPRREP messages, based on AODV RREQ and RREP messages respectively, to locate the node corresponding to a target SIP URI. The format of the SIPRREQ message is shown in Figure 4.

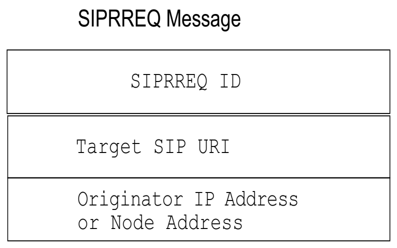


Fig. 4. Format of the SIPRREQ message

`SIPRREQ ID` is a sequence number uniquely identifying the particular SIPRREQ message when considered in conjunction with the originator IP address. The `Target SIP URI` is the SIP URI of the remote target with which the requesting party wants to setup a session.

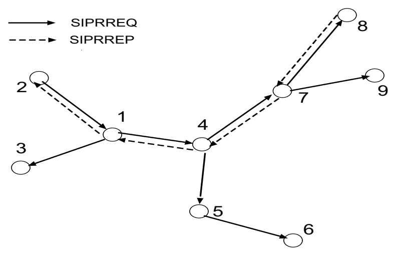


Fig. 5. The loosely coupled approach (LCA)

The requesting node disseminates a UDP based SIPRREQ message when it wants to discover a node address corresponding to the `Target SIP URI`. The `SIPRREQ ID` field is incremented by one from the last `SIPRREQ ID` used by the originating node. To prevent unnecessary network-wide dissemination of SIPRREQ and cause the "broadcast storm" problem, the originating node uses an expanding ring search technique. The time to live (TTL) for the SIPRREQ is initially set to TTL_START and then after a timeout period, called the RING_TRAVERSAL_TIME, if there is no response the TTL is incremented by TTL_INCREMENT. This is continued till TTL reaches TTL_THRESHOLD, when it is set to NET_DIAMETER. The retransmission of SIPRREQ is done following an exponential back-off algorithm to reduce congestion. If the node is not discovered within NET_TRAVERSAL_TIME, the originator node tries again to discover the node by broadcasting another SIPRREQ. Typical values of the TTL related parameters used in the discovery process can be obtained from the AODV recommendations [30]. The target node address can be determined when the SIPRREQ message reaches the target node or gets a "fresh enough" mapping of the SIP URI and the corresponding node address at an intermediate node. The discovered node address is then made available by unicasting a SIPRREP message back to the requesting node. The discovery process is illustrated in Figure 5.

The format of the SIPRREP message is shown in Figure 6. The `Target Node Address` is the node address corresponding to the node with the `Target SIP URI`. A node generates a SIPRREP message for either of the following two cases: (i) the node is itself the target or (ii) the node has a mapping of the `Target SIP URI` for a SIPRREQ message with same of higher `SIPRREQ ID` than that of the current request. When generating the SIPRREP, the node copies the `Target SIP URI`, the `SIPRREQ ID`, and the `Originator IP Address or Node Address` from the SIPRREQ message. The `SIPRREQ ID` is used by any intermediate node to keep a mapping for the `Target SIP URI`. Once created, the SIPRREP message is unicast using AODV to the next hop towards the originator of the SIPRREQ message. Thus when the SIPRREP reaches the originator, it knows the location of the target SIP URI and hands over the process of SIP message routing and subsequent media packet routing to AODV. However, potentially any ad hoc routing protocol can be used with this broadcast base4d discovery scheme.

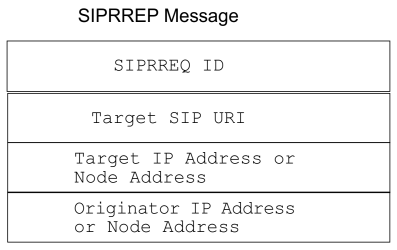

SIPRREP Message
SIPRREQ ID
Target SIP URI
Target IP Address or Node Address
Originator IP Address or Node Address

Fig. 6. Format of the SIPRREP message

VI. TCA: Tightly Coupled Approach

TCA is an integrated approach where the SIP end point discovery is coupled with a distributed cluster based routing protocol. The cluster based routing protocol creates a virtual topology with the cluster heads forming a backbone network, which is used in the routing of both SIP messages and data packets.

For convenience some of the terms and data structures used in the following protocol description are explained below.

- *Node*: An ad hoc network node with the extra capability of functioning as a SIP user agent, a registrar with a location service and a proxy server. However, all the functionalities are not required by the node at the same time.
- *Node ID*: A node ID is a string that uniquely identifies a node in the network. The internal node address or the IP address is generally used for this purpose.
- *Degree*: The degree of a node is the number of nodes adjacent to the given node.
- *Cluster*: A cluster is a group of nodes with a cluster head. The mechanism of forming a cluster and selecting the cluster head is described later in this section.
- *Cluster Head*: A cluster head is the node that elects itself as the leader for a cluster of nodes. A cluster node has all the information on the other members of the cluster and how to reach the nearest cluster heads of other clusters for forwarding packets.
- *Cluster Member*: Any node in a cluster which is not the cluster head is a cluster member.
- *Adjacency Table*: An adjacency table for a node contains a list of all the neighboring nodes along with their types, i.e., whether they are cluster heads or only members.
- *Cluster Adjacency Table*: A cluster adjacency table of a node contains the list of all cluster heads which are 2 hops away.

The protocol takes a fully distributed approach in the construction of clusters with nodes having higher degrees as the potential cluster heads and all other nodes are 1-hop away from their nearest cluster heads. The cluster heads connect with each other, either directly or through specially designated gateway nodes. As we shall prove later, the union of the cluster heads and the gateways form a fully connected backbone network topology. The union of the minimal number of cluster heads and gateways forming such a backbone is known as the Minimal Dominating Set (MDS). [1] Having minimal number of cluster heads is desirable since these are the most computationally intensive entities in the entire network, thus saving the total energy expenditure. The protocol builds such an MDS using local information only, which is gathered by means of periodic HELLO message that each node broadcasts to its 1-hop neighbors at an interval of HELLO_PERIOD.

Each node on receiving the HELLO message from its 1-hop neighbors computes its degree and uses it in the cluster head selection algorithm described next. A node selects itself as a cluster head if any of the following conditions are satisfied.

Condition 1: The node has the highest degree in its 1-hop neighborhood.

Condition 2: The node has the highest degree in the 1-hop neighborhood of any of its 1-hop neighbors.

A. Cluster Head Selection

On bootstrap, each node sends a HELLO message to its 1-hop neighbors. The format of the HELLO packet is shown in Figure 7. Initially, the `Degree` field is set to 0, the `Clusterheadflag` is turned off and both the adjacency tables are kept empty.

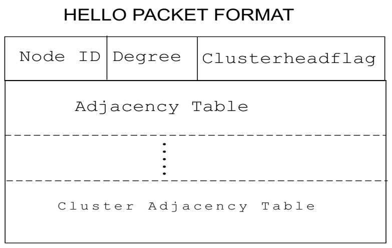


Fig. 7. Format of the HELLO message

Each node receives a HELLO packet from its 1-hop neighbors, computes its degree and populates the adjacency table. The format of the adjacency table is shown in Figure 8.

After the time period specified by HELLO_PERIOD, each node sends again the HELLO message, this time with the `Degree` field populated and the adjacency table completed. Note that the `Degree` fields of the adjacency table are still set to 0. It is with the next round of

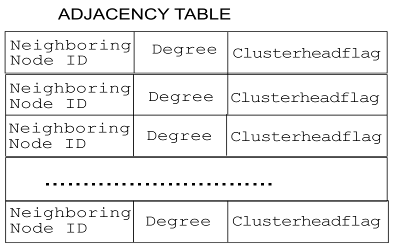


Fig. 8. Format of the adjacency table

HELLO message broadcast that each node gets to know the degree of its neighbors. Hence after three rounds of HELLO message exchange, each node can employ the **checkClusterhead** algorithm, shown in Figure 9, to decide whether it is a cluster head or not. The notations used in the algorithm are summarized in Table I.

TABLE I
NOTATIONS

$cluster(j)$	Cluster head corresponding to node j
$degree(j)$	Degree of node j
$type(j)$	Type of node j denoting whether it is a cluster head, cluster member or a gateway
$\mathcal{N}_1(j)$	Set of nodes 1-hop away from node j

```
1:  checkClusterhead($i$);
2:  for ($j \in \mathcal{N}_1(i) \cup \{i\}$) do
3:     $cluster(j) = j$;
4:     for ($k \in \mathcal{N}_1(j)$) do
5:        if ($degree(k) > degree(j)$) then
6:           $cluster(j) = k$;
7:        end if
8:     end for
9:     if ($cluster(j)$ == i) then
10:       $type(i) = clusterhead$;
11:    else
12:       $type(i) = clustermember$;
13:    end if
14: end for
```

Fig. 9. Cluster Head Selection Algorithm

Lines 4-6 of the **checkClusterhead** algorithm implements *Condition 1*. However, this condition alone cannot ensure even distribution of cluster heads and can result in too many cluster heads around the same set of nodes, which do not lead to an MDS. This is illustrated in the following example shown in Figure 10.

If the protocol had considered only *Condition 1*, then nodes 1, 2, 4, 6, 7, and 9 would have been selected as

[1]Finding a Minimum Dominating Set is, however, an NP-complete problem that can be mapped to the well-known set covering problem.

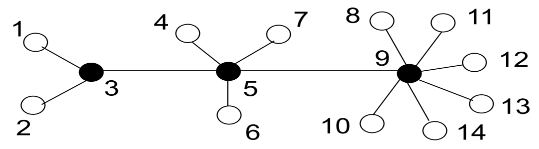


Fig. 10. A network topology showing the need for *Condition 2*

the cluster heads leaving out node 3 and 5 with higher degrees. Clearly this does not lead to an MDS. Ideally we would want nodes 3, 5 and 9 to be the cluster heads in this case. This is ensured by *Condition 2*.

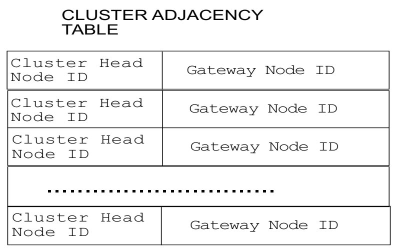
Once the cluster heads get selected, they assume the extra responsibility of acting as SIP proxies and registrars. In other words, they act as the inbound and outbound proxies for the SIP messages from the respective cluster members and keep a mapping of the SIP URIs and the node address of all of their respective cluster members. After a node gets selected as the cluster head, the `Clusterheadflag` filed of its HELLO message is turned on to let the neighbors know of their cluster head.

B. Cluster Formation

Once the cluster heads select themselves, they maintain connectivity to the neighboring cluster heads through the gateway selection process described in the following subsection. The remaining nodes or the cluster members get to know about them in the next round of HELLO message broadcast. We will prove later that each cluster member has atleast one neighboring 1-hop cluster head. The cluster member then sends a SIP REGISTER message to the 1-hop cluster head with highest degree and registers with the registrar service in there. In case of a tie, the cluster head with the lowest Node ID gets selected for registration.

C. Gateway Selection

Cluster heads form a virtual topology, where the routing of control and data packets take place through the cluster heads. Hence the cluster heads should be reachable from each other or, in other words, each cluster head should be aware of all the neighboring cluster heads. We shall shortly show that the cluster heads selected following the above procedure are either 2 or 3 hops away from their nearest neighbors. The HELLO message can detect the cluster heads which are 2 hops away but not those which are 3 hops away. For detecting the cluster heads 3 hops away, a cluster adjacency table is maintained at each node. Figure 11 shows the format of a cluster adjacency table.



CLUSTER ADJACENCY TABLE

Cluster Head Node ID	Gateway Node ID
Cluster Head Node ID	Gateway Node ID
Cluster Head Node ID	Gateway Node ID
....	
Cluster Head Node ID	Gateway Node ID

Fig. 11. Format of the cluster adjacency table

Each cluster member gets information about its 2-hop cluster heads from the HELLO messages. It creates its own cluster adjacency table for its 2-hop away cluster heads with the intermediate 1-hop neighboring node, relaying the HELLO message, as the *gateway* node. The cluster adjacency table is then appended to the HELLO message as an extension and sent to all the 1-hop neighbors. Any cluster head in its 1-hop neighbor gets to know about the cluster heads which are 3 hops away and identifies the cluster adjacency table relaying node as the gateway node. In either case, there may be more than one candidate for the gateway node. In those cases, the node with the lowest ID is selected as the gateway node.

Let us explain the gateway selection procedure by an example. Let a member node A gets to know about a 2-hop cluster head C from the HELLO message of an intermediate member node B. A then creates a cluster adjacency table with an entry having C as the cluster head and B as the gateway. Then it appends the cluster adjacency table to the HELLO message and send it to its immediate 1-hop neighbors. Let D be a cluster head in its 1-hop neighborhood. D adds to its routing table, the 3-hop cluster head C and the corresponding gateway as node A. Now D can reach C through the series of two gateway nodes, A and B. Thus each cluster head can reach to its 2-hop or 3-hop cluster heads through the designated gateways.

D. Function of SIP Registrar and Proxy Server

A cluster member on identifying its cluster head (from cluster heads HELLO message) registers with the corresponding SIP registrar by sending a SIP REGISTER message. The location service associated with the registrar in the cluster head keeps map of all the SIP URI and the node addresses of the cluster members. Because of the virtual topology induced by the cluster heads, the registration can be executed in exactly the same fashion as it takes place in an infrastructure based network [32].

E. *Routing Procedure*

1) Cluster Connectivity: The cluster head selection algorithm and the gateway selection procedure work in tandem to build a virtual topology where each cluster head can reach to its 2-hop and 3-hop neighboring cluster heads through the gateway nodes. This and the fact that each of the member nodes is atmost 1-hop away from a cluster head make it possible for any member node to reach any other member node through the cluster heads. In case a cluster head moves out of radio range, the local information based fully distributed operation of the cluster head selection algorithm ensures the selection of another appropriate node as the cluster head within a few subsequent rounds of HELLO message broadcast. If a cluster member moves, it can either itself become a cluster head or can remain a cluster member to a different cluster head. In the latter case, the cluster member again registers with the new cluster head's registrar service. Thus, the virtual topology and routing framework is maintained by the protocol in the face of node mobility.

2) Route Discovery: In our protocol, the cluster heads act as SIP proxies and as the forwarding nodes. Since only the cluster heads are responsible for forwarding the route discovery messages, the routing overhead is considerably reduced. When the SIP UAC in a cluster member node wants to establish a session with the SIP UAS of another cluster member node, it sends a SIP INVITE message with the `Request-URI` as the URI of the target SIP UAS. The INVITE message is sent to the corresponding proxy of the requesting node. The proxy then sends this message to the neighboring cluster heads or proxies in order to discover the route to the target node. In fact, the SIP call forking feature [32] can be used to achieve this. If any of the neighboring proxies has the target URI registered with itself, it sends the INVITE message to the target node, otherwise it forwards the message to its neighboring cluster heads after recording the proxy address in the `Record-Route` field of the SIP message. The target node on receiving the INVITE message sends back a SIP OK message via the reverse route specified by the list of traversing proxies in `Record-Route` header field. This is exactly the same as the typical proxy based routing of SIP messages [32]. The requesting node on receiving the SIP OK message, gets to know about the route to the target, which is used subsequently for both SIP session establishment and media packet delivery. The route to the destination is also stored in the intermediate cluster heads in a cache to reduce the overhead with subsequent route discoveries.

Figure 12 shows an example of the routing procedure for the same network that we considered to illustrate the operation of LCA. Initially all the nodes broadcast the HELLO message to their immediate neighbors. Thus node 1 gets intimation from nodes 2, 3, and 4. After the second round of HELLO message broadcast with the adjacency table, node 1 gets to know about the degrees of nodes 2, 3, and 4. In the next round of HELLO message, node 1 gets elected with subsequent formation of cluster I. Following the same steps as that of node 1, nodes 5 and 7 get selected as the cluster heads of cluster II, and III, respectively. The cluster members, on knowing their respective cluster heads in the next round of HELLO packet broadcast, register their URI with the registrar in the cluster heads, using the SIP REGISTER message. Also with the subsequent HELLO messages containing the cluster adjacency tables, node 4 get selected as the gateway node for communication between the cluster heads. Now if the UAC of node 2 wants to establish a session with the UAS of node 8, it will send a SIP INVITE message, with the `Request-URI` as the URI for node 8, to its designated cluster head i.e., node 1. Node 1 then selectively broadcasts the INVITE message to the neighboring cluster heads through the corresponding gateway nodes. The cluster head of cluster III, i.e., node 7 finds the `Request-URI` among the URIs of the nodes that have registered with it. Thus it forwards the INVITE message to node 8. Throughout the transmission of the INVITE message the path consisting of the series of traversing proxies is recorded in the `Record-Route` field. A SIP OK message is then sent back to node 2 following the list of proxies in the `Record-Route` field in the reverse order. Once the route is defined in this way between nodes 2 and 8, it is used for continuing with the session establishment and media packet delivery.

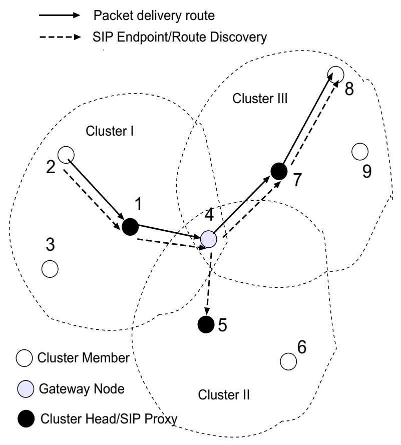


Fig. 12. Protocol Operation: Cluster Formation and Route discovery

F. Improving Routing Efficiency

The union of cluster heads and the gateway nodes define a fixed (or relatively static) connected multihop wireless network, where each of the nodes act as "wireless IP router" forwarding both the SIP messages and media packets. A forwarding node typically receives packets from the upstream nodes and then transmits them to the downstream nodes. Since the routing load is entirely focussed on the cluster heads and the gateway nodes, the packet forwarding efficiency of these nodes plays an important role in the overall efficiency of the cluster based protocol. However, multihop IEEE 802.11 wireless LAN, the most dominant of the present-day multihop networking technology, pose several challenges in terms of the available system throughput due to multihop routing inefficiency. The current 802.11 Distributed Coordination Function (DCF) MAC algorithm has been designed implicitly for either receiving or transmitting a packet, but not for a forwarding operation (i.e., receiving a packet from an upstream node and then immediately transmitting the packet to a downstream node as an atomic channel access operation). There are two key deficiencies:

- The forwarding node is involved in two separate RTS/CTS contention-based channel access attempts during the forwarding process: once to receive the packet (from the upstream node) and again to forward it (to the downstream node), and must thus suffer the contention resolution overhead twice.
- The same packet makes an unnecessary round-trip between the memory on the network interface card (NIC) and the hosts memory (accessed by the host software) to determine the next-hop MAC address. This round-trip not only loads the processor of the forwarding node, but also suffers from additional delays in transfers between the NIC and the host operating system.

A wireless IP forwarding architecture that uses MPLS [31] with modifications to IEEE 802.11 MAC has been proposed in [2] primarily to solve this problem and significantly improve the packet forwarding efficiency. The overheads of separate channel accesses is eliminated by defining the Data-Driven Cut-Through Medium Access (DCMA) protocol [2] as a simple extension of the 802.11 DCF. DCMA combines the Data ACK (to the upstream node) with the RTS (to the downstream node) in a single ACK/RTS packet that is sent to the MAC broadcast address. The problem of round-trip delay between the memory on the NIC and the hosts memory is solved by enabling the lookup for next hop within the NIC, without needing to perform the routing lookup in the host. MPLS, a well-known IP compatible technology has been used to perform next-hop lookup inside the NIC, by setting up labels that enables fast and scalable determination of the MAC address of the downstream node.

The system throughput of the IEEE 802.11 multihop networks can be further improved by increasing concurrent transmissions through better spatial reuse. The 802.11 MAC protocol and its variants are primarily designed for a single-hop wireless environment, where nodes typically form a clique and communication always takes place over a single wireless hop (often to a base station providing connectivity to the wired infrastructure). In such a single-cell environment, the 802.11 MAC contention resolution mechanism focuses primarily on ensuring that only a single sender-receiver node pair receives collision-free access to the channel at any single instant. The 802.11 MAC does not seek to exploit the spatial diversity inherent in multihop networks, where different sets of nodes are able to concurrently communicate with different sets of neighbors. This can be achieved potentially by three different methods: use of power control algorithms, use of directional antennas and modification of the MAC itself to relax some unduly harsh restrictions of the IEEE 802.11 MAC. One such 802.11-like protocol is called MACA-P [1] that provides synchronized parallel transmissions by allowing neighboring nodes to synchronize their reception periods, so that 1-hop neighbors switch between transmitting and receiving roles in unison at explicitly defined instants, and thus avoid the problem of packet collisions.

VII. Properties of the Cluster-based Protocol

Property 1: Every node is either a cluster head or a 1-hop neighbor of a cluster head.

Proof: The proof follows from the fact that the **checkClusterhead** algorithm runs at each node. A node is selected as a cluster head when it has the highest degree among its 1-hop neighbors or has a neighboring node with the highest degree among the 1-hop neighbors of the node. Thus each node is either a cluster head or a 1-hop neighbor of a cluster head.

Property 2: The maximum distance from any cluster head to another closest cluster head is 3.

Proof: The proof follows from *Property 1*. Since each of the member node is adjacent to atleast one cluster head, there can be atmost two neighboring cluster members between two closest cluster heads. Hence the result follows.

A. *Proof of Correctness*

1) First we shall prove that the cluster heads are all connected. When the cluster heads are 1-hop away, they are trivially connected with each other by a link. When the cluster heads are 2 or 3 hops away they are connected through a gateway or a pair of gateway nodes, which is ensured by the gateway selection algorithm described in sectionVI-C.
2) Now we shall prove that any node in the network can reach to any other node. Let the topology of the network be represented by the undirected graph $G = (V, E)$, where V is the set of nodes and $E \subseteq V \times V$ is the set of links between the nodes. Now let us verify whether any two nodes $v_0 \in V$ and $v_n \in E$ are connected or not. If v_0 and v_n are cluster heads then they are connected according to the first part of the proof. Otherwise, let v_0 and v_n are both cluster members. Then by *Condition 1*, for two cluster heads, v_1 and v_{n-1}, the following expression is true: $[(\exists v_1 \in V | (v_0, v_1) \in E) \wedge (\exists v_{n-1} \in V | (v_{n-1}, v_n) \in E)]$. Again, by the first part of the proof, v_1 and v_{n-1} are connected, hence v_0 and v_n are connected too. Note that the case when either one of v_0 or v_n is a cluster head is only a trivial subset of the previous case. Hence, any two nodes in the network are connected through the virtual topology created by the cluster heads.

VIII. Performance Evaluation

In this section we evaluate the performance of SIP based session setup for LCA and TCA. We compare the two approaches with respect to the following two important performance metrics: (i) the delay in discovering a SIP end point before establishing a SIP based session for static multihop wireless networks and dynamic wireless networks with random node mobility, and (ii) the control overhead, i.e., the number of control packets involved with either of the approaches.

A. *Simulation Experiments*

We have performed extensive simulation experiments with `ns2` [26]. For a static multihop wireless network, half the nodes are placed in a grid fashion within a $1000m \times 1000m$ square area to ensure connectivity, while the rest of the nodes are randomly distributed within the square area. In all the experiments, if not otherwise mentioned, the connections have been established between the farthest pair of nodes in the network. For dynamic networks, 15 nodes move randomly in a $650m \times 650m$ square area, following a random waypoint mobility model. The HELLO_PERIOD interval for TCA has been set to 5 secs. The TTL related parameter values for LCA have been taken to be the same as recommended by AODV specifications [30].

B. *Latency*

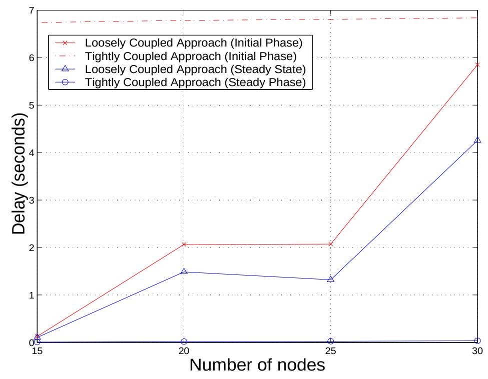


Fig. 13. Latency in SIP URI discovery in a static multihop network

1) Latency for static multihop networks: The latency involved in discovering a node with a particular SIP URI before establishing a session in a static multihop network is presented in Figure 13. Two phases of SIP end point discovery have been carried out to evaluate the effect of protocol convergence in the two approaches. We have observed that the latency in the initial phase of node discovery is much higher for the integrated approach than that of LCA. This is because the integrated approach takes some time to elect the cluster heads and form the clusters. But once the cluster formation is over, the discovery process takes much less time. This is evident from the latency involved with the second phase of SIP end point discovery. Also, in LCA the discovery message is incrementally broadcasted each time a SIP end point needs to be discovered, contributing to the latency. On the other hand, in the integrated approach selective broadcast is done only to the proxies or cluster heads, which essentially covers the entire network in one round of broadcasting. Of course, caching at each node may reduce the latency in locating an already discovered target for LCA, but we have presented results for different targets in each phase to illustrate the relative efficiencies of the two approaches.

2) Latency when the destination moves: Figure 14 shows the latency figure in discovering a SIP URI when the destination starts moving towards the source with a speed of 15m/s. It is observed that with the increase in the number of nodes in the network, the latency for LCA increases dramatically, whereas that for TCA is much

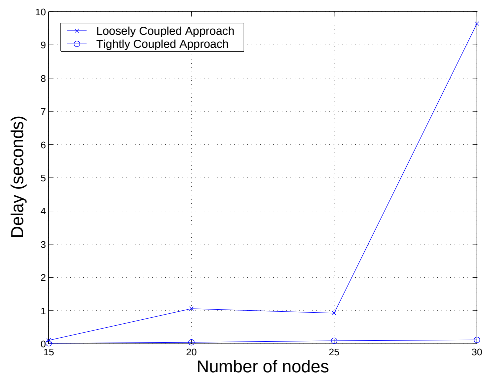


Fig. 14. Latency in SIP URI discovery when the destination moves

less and remains steady. This is because in TCA a virtual infrastructure is setup with the clusters, which account for the fast and scalable broadcast message transmission resulting in the faster discovery. This, however, happens as in this case we have a static multihop network backbone available for setting up the infrastructure, but as we will see later, it all changes as we have a highly dynamic wireless network with the randomly moving nodes.

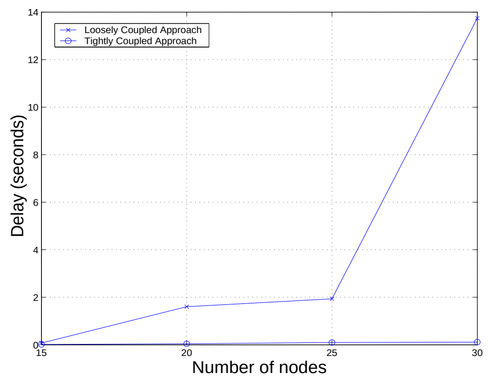


Fig. 15. Latency in SIP URI discovery when the proxies moves

3) Latency with proxy node movement: In TCA the cluster heads are configured as proxies and when a proxy moves, it may not remain a cluster head and may have to relinquish its role of a proxy. In that case the nodes affected get reconfigured to form new clusters with new cluster heads. We have measured the effect of the proxy movement on either approaches by allowing the proxies in TCA corresponding to the source and destination node to move randomly with a speed of 15m/s. For LCA, although there is no concept of proxies, the corresponding nodes are moved in the same random fashion. The delay for LCA vs. TCA is shown in Figure 15. In this case also, TCA shows considerable resilience and fares better than LCA in terms of the latency in SIP end-point discovery.

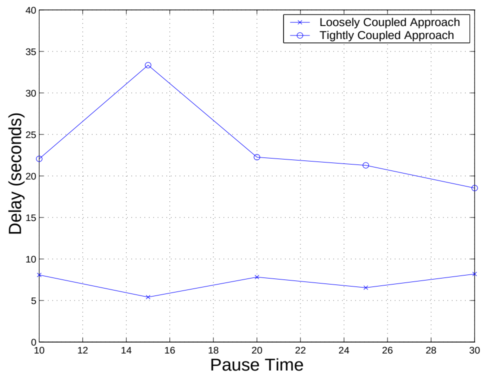


Fig. 16. Latency in SIP URI discovery with random node mobility

4) Latency with random node mobility: The performance scenario changes completely with random node movement. As mentioned above, with random node mobility all the nodes move following a random waypoint mobility model with speed 10m/s. The latency results for different pause time are averaged over 10 different random mobility scenarios and are shown in Figure 16. Due to high node mobility, TCA incurs significant delay in setting up and maintaining the virtual infrastructure resulting in high latency in discovering the SIP end point. In LCA, no such infrastructure is setup and broadcasting helps in finding the node directly through the shortest path, no matter how the nodes are moving, thus resulting in lower latency in finding the SIP end point.

C. Control Overhead

Another important performance metric is control overhead, measured in terms of the number of control packets exchanged in the network. This includes all the AODV messages along with the SIPRREQ and SIPRREP messages for the LCA and all the routing related messages for the TCA. Apart from contributing to the latency factor, the control overhead determines the scalability of a particular approach. It also affects the resulting throughput for data packets, as the control packets gets priority over the data packets in each node.

1) Control overhead for static multihop networks: The control overhead associated with the two approaches for the static multihop wireless network is shown in Figure 17. LCA has lower control overhead than that of TCA, initially. But, as the network grows larger, the overhead in LCA starts increasing rapidly and overshoots (for networks with more than 23 nodes) the corresponding overhead of TCA. This happens because the number

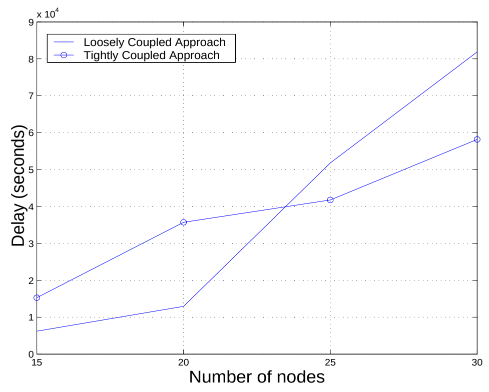


Fig. 17. Control overhead for static multihop networks

of message broadcasts increases drastically in LCA with the increase in the number of network nodes. In TCA, however, due to the restricted broadcast through the cluster heads, the control overhead remains scalable with the number of network nodes.

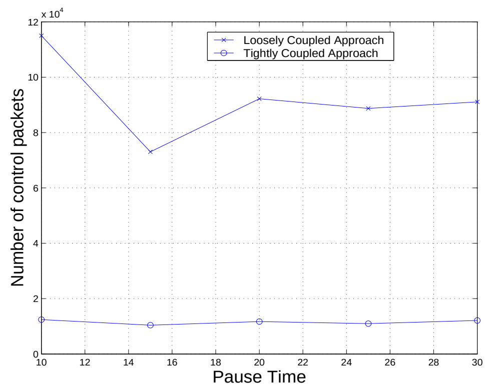


Fig. 18. Control overhead with random node mobility

2) Control overhead with random node mobility: Figure 18 presents the respective control overheads for the two approaches with random node mobility. Here also the speed has been taken to be 10m/s and the result has been shown with respect to different pause time averaged over 10 different random mobility scenarios. Although we have seen that LCA discovers the nodes faster than the integrated approach, it suffers from the "broadcast storm" problem in the face of random node mobility. This results in approximately an order of magnitude increase in the number of control messages for LCA over TCA.

IX. Discussions

The performance results reveal that TCA performs better than LCA for static multihop networks. Of course, there is a setup time for the virtual topology in TCA, but once it is setup, the node discovery process becomes much faster. TCA shows remarkable resilience in the face of isolated node movements, i.e., when the destination moves towards the source and when the proxies related to the source and the destination nodes move, albeit the latency in the later case is higher than that in the former case. However, in both the cases the latency for node discovery has been lower for TCA. This is because, despite one or two node movement, the overall virtual topology is maintained and a node can be quickly discovered using restricted broadcast through the virtual topology. On the other hand, in LCA, each time a node needs to be discovered, the expanding ring search technique is employed, whereby a considerable delay is incurred. The scenario, however, changes completely when all the nodes move randomly. LCA, in this case performs better than TCA in terms of latency, since the former approach does not entail the setting up and maintenance of a virtual topology in the face of random node mobility and can discover a node through the shortest path irrespective of the node movements.

The cluster based algorithm is designed with the objective of reducing the control overhead by restricting the broadcasting of the discovery messages. This is instantiated by the lower control overhead associated with TCA for a network with sufficient number of nodes (below which the overhead associated with the building of the virtual topology offsets the gain in broadcast overhead). In case of random node mobility also, LCA performs poorly when compared to TCA, due to the redundancy factor of the "broadcast storm problem".

So broadly speaking, the performance evaluation suggests that LCA should be adopted when it is required to setup sessions quickly in a network with high node mobility. Otherwise, TCA performs well for networks with low node mobility or static multihop networks. Also, if a little delay in initial session setup can be allowed, then TCA proves to be an attractive solution in the long run because of its low control overhead and consequently higher throughput in data packet transmission. Besides, TCA results in a virtual topology with SIP proxies and registrars, which can be most effectively used as anchor points for several specialized SIP based services, such as conference setup, SIP-based mobility management, etc.

X. Conclusion

In this paper, we have proposed two approaches to enable SIP session setup in ad hoc networks. One is a loosely coupled approach, where the SIP end point discovery is completely decoupled from the underlying routing protocol, while the other is a tightly coupled approach. We have proposed a cluster based routing algorithm integrated with the SIP end point discovery for the tightly coupled approach. Apart from having better performance in static multihop wireless networks with low node mobility, the cluster based routing protocol creates a virtual topology that can be effectively used to provision specialized SIP based services. For networks with highly mobile nodes, however, the loosely coupled approach has more desirable performance figures. We would like to incorporate the resource heterogeneity issue of the nodes in our cluster based algorithm. The feature of SIP enabling separation of signaling and media path can be potentially used in the context of load balancing in the cluster based approach where the overburdened gateway nodes and the cluster heads can be relieved by distributing the load appropriately among themselves. We would like to investigate into such load balancing schemes in our future work. Finally, we intend to analyze the performance gain that can be achieved in the cluster based algorithm by applying the routing enhancement schemes described in section VI-F.

References

[1] A. Acharya, A. Misra, and S. Bansal, “MACA-P : a MAC for concurrent transmissions in multi-hop wireless networks” *IEEE PerCom*, pp. 505–508, 2003.

[2] A. Acharya, A. Misra and S. Bansal, “A label-switching packet forwarding architecture for multi-hop wireless LANs”, *WoWMoM*, pp. 33-40, 2002.

[3] S. Basagni, “Finding a maximal weighted independent set in wireless networks,” *Telecommunication Systems, Special Issue on Mobile Computing and Wireless Networks*, 18(1/3), pp. 155-168, 2001.

[4] S. Berger, H. Schulzrinne, S. Sidiroglou, and X. Wu, “Ubiquitous computing using SIP”, *NOSSDAV*, pp. 82 - 89 , 2003.

[5] T. Clausen, and P. Jacquet, “ Optimized Link State Routing Protocol (OLSR)”, *IETF RFC 3626*, October 2003 .

[6] F. Dai and J. Wu, “An Extended Localized Algorithm for Connected Dominating Set Formation in Ad Hoc Wireless Networks,” *IEEE Transactions on Parallel and Distributed Systems*, Vol. 15, No. 10, 2004.

[7] A. Dutta, K. D. Wong, J. Burns, R. Jain, A. McAuley, K. Young, and H. Schulzrinne, *MILCOM 2002*, Vol. 1 , pp. 448 - 454, 2002.

[8] Gnutella webpage: http://gnutella.wego.com/.

[9] S. Guha and S. Khuller, “Approximation algorithms for connected dominating sets. Algorithmica,” Vol. 20, No. 4, pp. 374-387, 1998.

[10] E. Guttmann, C. Perkins, J. Veizades, M. Day, “Service Location Protocol, Version 2”, *RFC 2608*, 1999.

[11] Z. J. Haas and M. R. Pearlman, “The Zone Routing Protocol (ZRP) for ad hoc networks”, *IETF Internet Draft draft-ietf-manet-zonezrp- 01.txt* (1998).

[12] M. Handley, and V. Jacobson, “SDP: Session Description Protocol”, *IETF RFC 2327*, April 1998.

[13] International Telecommunication Union, “Packet based multimedia communications systems”, *Recommendation H.323, Telecommunication Standardization Sector of ITU*, Geneva, Switzerland, Feb. 1998.

[14] W. Jiang, J. Lennox, H. Schulzrinne and K. Singh, “Towards Junking the PBX: Deploying IP Telephony”, pp. 177-185 *NOSSDAV* 2001.

[15] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark, “Scenario-based performance analysis of routing protocols for mobile ad-hoc networks”, *Mobicom*, pp. 195 - 206, 1999.

[16] D. B. Johnson and D. A. Maltz, “The dynamic source routing in ad-hoc wireless networks”, *Mobile Computing, eds. T. Imielinski and H. Korth, chapter 5 (Kluwer, Dordrecht, 1996)* pp. 153 - 181.

[17] R. Koodli, C. E. Perkins, “Service Discovery in On-Demand Ad Hoc Networks”, *IETF Internet Draft, Manet Working Group, draft-koodli-manet-servicediscovery-00.txt*, October 2002. P.

[18] P. Krishna, N. Vaidya, M. Chatterjee, and D. Pradhan, “A cluster-based approach for routing in dynamic networks,” *ACM SIGCOMM Computer Communication Review*, pp. 49-65, April 1997.

[19] S.-J. Lee, M. Gerla, and C.-K. Toh, “A Simulation Study of Table-Driven and On-Demand Routing Protocols for Mobile Ad Hoc Networks”, *IEEE Network*, Vol. 13, No. 4, pp. 48-54, 1999.

[20] C. R. Lin and M. Gerla, “Adaptive clustering for mobile, wireless networks,” *Journal on Selected Areas of Communication*, Vol. 15, No. 7, 1997.

[21] H. Luo, R. Ramjee, P. Sinha, L. Li, and S. Lu, “UCAN: a unified cellular and ad-hoc network architecture,” *MOBICOM*, pp. 353-367, 2003.

[22] R. Matei, “JAIN SIP Approach in Ad-Hoc Networks Mobility Management” Ad Hoc Mobile Wireless Networks Research Seminar on Telecommunications Software, Autumn 2002. *http://www.tml.hut.fi/Studies/T-110.557/2002/papers/*

[23] S. Murthy and J. J. Garcia-Luna-Aceves, “An efficient routing protocol for wireless networks, ACM Mobile Networks and Applications”, *Special Issue on Routing in Mobile Communication Networks* pp. 183 - 197, 1996.

[24] Napster webpage: http://www.napster.com/.

[25] A. Niemi et al. “Session Initiation Protocol (SIP) Extension for Presence Publication” *SIMPLE Working Group Internet-Draft*, June 2003.

[26] “The network simulator”, available at `http://www.isi.edu/nsnam/ns`

[27] M. O’Doherty , “Pico SIP”, *IETF Internet Draft, draft-odoherty-pico-sip-00.txt*, January 2001.

[28] V. D. Park and M. S. Corson, “A highly adaptive distributed routing algorithm for mobile wireless networks”, *Proceedings of 1997 IEEE Conference on Computer Communications, INFOCOM97*, pp. 1405 - 1413, 1997.

[29] C. E. Perkins and P. Bhagwat, “Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers”, *Proceedings of ACM SIGCOMM*, pp. 234 - 244, 1994.

[30] C. Perkins, E. Belding-Royer, and S. Das “ Ad hoc On-Demand Distance Vector (AODV) Routing ”, *IETF RFC 3561*, July 2003.

[31] E. Rosen, A. Viswanathan, and R. Callon, “Multiprotocol Label Switching Architecture,” *IETF RFC 3031*, Jan. 2001.

[32] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, “SIP: Session Initiation Protocol”, *IETF RFC 3261*, June 2002.

[33] H. Schulzrinne et al. “RTP: A Transport Protocol for Real-Time Applications” *IETF RFC 1889* Jan 1996.

[34] Skype webpage: http://www.skype.com/.

[35] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek F. and H. Balakrishnan, “Chord: a scalable peer-to-peer lookup protocol for internet applications”, *IEEE/ACM Transactions on Networking (TON)*, Vol. 11, No. 1, pp. 17-32, 2003.

[36] Y.-C. Tseng, S.-Y. Ni, Yuh-Shyan Chen, and J.-P. Sheu, “The Broadcast Storm Problem in a Mobile Ad Hoc Network,” *ACM Wireless Networks*, Vol. 8, No. 2, pp. 153-167, 2002.

[37] M. Weiser, “The Computer for the Twenty-First Century,” *Scientific American*, 265(3), pp. 94-104, 1991.