

# Enabling SIP-Based Sessions in Ad Hoc Networks

Nilanjan Banerjee <sup>§ †</sup>

Motorola India Research Labs,  
66/1, Plot 5, Bagmane TechPark, C. V. Raman Nagar Post,  
Bangalore - 560093, India.

Email: nilanjan@motorola.com

Phone: +91 80 2601 6065; Fax: +91 80 2534 3100

Arup Acharya

3N-C12, IBM T. J. Watson Research Center,  
19 Skyline Drive, Hawthorne, NY 10532, USA.

Email: arup@us.ibm.com

Phone: +1 914 784 7481

Sajal K. Das

CReWMaN, Department of Computer Science and Engineering,  
The University of Texas at Arlington,  
Box 19015, 416 Yates St., Room 300, Nedderman Hall,  
Arlington, TX 76019-0015, USA.

Email: das@cse.uta.edu

Phone: +1 817 272 7405

<sup>§</sup> Corresponding Author; <sup>†</sup> This work was partially done while the author was a graduate student in CReWMaN, University of Texas at Arlington.

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

# Enabling SIP-Based Sessions in Ad Hoc Networks

Nilanjan Banerjee <sup>§ †</sup>

Motorola India Research Labs,  
66/1, Plot 5, Bagmane TechPark, C. V. Raman Nagar Post,  
Bangalore - 560093, India.

Email: nilanjan@motorola.com

Phone: +91 80 2601 6065; Fax: +91 80 2534 3100

Arup Acharya

3N-C12, IBM T. J. Watson Research Center,  
19 Skyline Drive, Hawthorne, NY 10532, USA.

Email: arup@us.ibm.com

Phone: +1 914 784 7481

Sajal K. Das

CReWMaN, Department of Computer Science and Engineering,  
The University of Texas at Arlington,  
Box 19015, 416 Yates St., Room 300, Nedderman Hall,  
Arlington, TX 76019-0015, USA.

Email: das@cse.uta.edu

Phone: +1 817 272 7405

## Abstract

The deployment of infrastructure-less ad hoc networks is suffering from the lack of applications in spite of active research over a decade. This problem can be solved to a certain extent by porting successful legacy Internet applications and protocols to the ad hoc network

<sup>§</sup> Corresponding Author; <sup>†</sup> This work was partially done while the author was a graduate student in CReWMaN, University of Texas at Arlington.

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

domain. Session Initiation Protocol (SIP) is designed to provide the signaling support for multimedia applications such as Internet telephony, Instant Messaging, Presence etc. SIP relies on the infrastructure of the Internet and an overlay of centralized SIP servers to enable the SIP endpoints discover each other and establish a session by exchanging SIP messages. However, such an infrastructure 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 endpoint discovery is decoupled from the routing procedure and (ii) a tightly coupled approach, which integrates the endpoint 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 in SIP session setup. The loosely coupled approach, on the other hand, generally 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 (SIP), Ad Hoc Networks.

## 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 and efficient ad hoc networks. The last few years have seen vigorous research primarily in ad hoc network routing protocols [16], [24], [38], [42], [43], [44], but the lack of applications in ad hoc domain has been a major impediment for the wide deployment and acceptance of ad hoc networks. One approach to solve this problem is to extend the legacy Internet protocol based applications to the ad hoc domain. The immediate advantage is that we have had enough knowledge on the design principles and functional operations of these applications, which could enable us to adapt them to the ad hoc domain easily. Moreover, as the notion of ubiquitous computing [61] 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 [34] seems

inevitable, entailing the extension or adaptation of key legacy protocols of fixed mobile networks to ad hoc networks, as well.

However, apart from individual adaptation of legacy applications to the ad hoc domain for optimal performance, there are some basic functional building blocks such as the standard Internet protocols on which the applications are built that need to be modified to make the applications work in ad hoc networks. For multimedia applications, signaling protocol is one such functional building block.

Signaling protocols have been developed for establishing multimedia sessions such as a Voice over IP (VoIP) with stringent resource requirements in the Internet. Signaling protocols negotiate resources between the terminals and establish a multimedia session. The two most prominent signaling protocols for IP based networks are H.323 [19] from International Telecommunication Union (ITU) and Session Initiation Protocol (SIP) [48] from IETF. SIP is progressively gaining popularity over H.323, primarily because of its simplicity and flexibility. Moreover, some of the features of SIP, such as re-directing a call and proxying, can be potentially applied to wireless networks with mobile nodes. 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. SIP has also found its application in the context of ubiquitous computing [3]. Ad hoc networks, being a key technology in ubiquitous computing, need to also support SIP to enable such applications. Besides, SIP can also act as the facilitating protocol for fixed-ad hoc network convergence. SIP has been extended by IETF SIMPLE (SIP for Instant Messaging and Presence Leveraging Extensions) Working Group (WG) [54] to enhance the basic protocol with Instant Messaging and Presence (IMP) functionalities. A presence system allows users to subscribe to each other and be notified of changes in state. Instant Messaging (IM), on the other hand, enables the exchange of content between a set of participants in near real time.

Although SIP being an application layer protocol, follows true end-to-end semantics and can establish direct client-to-client sessions provided the clients can reach each other, SIP relies on the Internet routing infrastructure and some centralized SIP servers for SIP

endpoint discovery (explained with further details in Section III) and subsequent routing of SIP messages for session establishment. Hence SIP cannot work as it is in an infrastructure-less network such as peer-to-peer (P2P) networks or ad hoc networks. Attempts have been made only recently to enable SIP operation in P2P networks [4], [25], [37], [51], [52], where it has been integrated with an underlying P2P protocol such as Chord [56]. However, P2P protocols are not suitable for ad hoc networks, which is radically different from P2P networks because of the node mobility. P2P protocols are incapable of supporting node mobility and are designed for static P2P nodes. Moreover, there has been enough indication from the previous studies [12], [29] that the service and resource discovery protocols in ad hoc networks, which are functionally similar to the problem we are solving here, need to be integrated with the underlying routing layer to handle the dynamism of ad hoc networks.

In this paper, we propose two possible approaches for the SIP endpoint discovery, *viz.* a *loosely coupled approach* and a *tightly coupled approach*. In the former approach, SIP endpoint discovery is decoupled from the underlying ad hoc routing protocol, whereas in the tightly coupled approach the SIP endpoint discovery is integrated with the routing protocol. While we use a simple expanding broadcast based scheme for the loosely coupled approach, we have used a distributed cluster based routing protocol for the purpose of integration with the SIP endpoint 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 generally fares better. The major contributions of this paper can be listed as follows:

- We establish the problem associated with basic SIP deployment over ad hoc networks. Then we address the issue of integration of SIP with ad hoc routing protocols. Note that, here we are not addressing the issues related to SIP based application deployment or development for ad hoc networks. Such issues need to be addressed separately for each individual applications. For example, a SIP-based conferencing application need special architectural solution for operation in ad hoc networks [11] in addition to the basic SIP-based session setup support. Such special application considerations are outside the scope of the paper. Here, by solving the problem of deploying SIP in ad hoc networks, we are merely providing the basic SIP-based session setup support

to the applications.

- We evaluate the two major approaches to ad hoc routing protocols *viz.* reactive and proactive routing protocols in the above context.

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 and related research activities. The loosely coupled approach is described in Section V and the tightly coupled approach is described in Section VI. The comparative performance evaluation of the two approaches is presented in Section VII along with related discussions in Section VIII. Finally, Section IX 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 [53]. It allows Internet endpoints with SIP services (the SIP endpoints) to establish media sessions with each other: this includes locating the endpoints, establishing the session and then terminating the session after the media session has been completed. Note that each individual application using SIP as the signaling protocol needs to be designed separately on top of SIP after considering their individual requirements, which are outside the scope of a signaling protocol. For example, in a SIP based multi-party conferencing session the conferencing architecture is independent of the SIP signaling architecture [28].

As shown in Figure 1, a SIP architecture consists of user agents, registration servers, location servers and SIP proxies deployed across a network. The logical entities at the SIP endpoints, which implement the SIP functionality of controlling session setup are called the user agents. The User agents are identified by SIP URIs (Uniform Resource Identifier), which is a unique HTTP-like URI of the form `sip:user@domain`. A SIP user is identified by an address-of-record (AOR). An address-of-record (AOR) is a SIP or SIPS URI that points to a domain with a location service that can map the URI to the contact IP address or to another URI, where the user might be available. All user REGISTER their contact information (e.g.,

IP address) with a SIP registrar server (which can be co-located with a SIP proxy). The association between the SIP AOR and the contact information is called the *binding*.

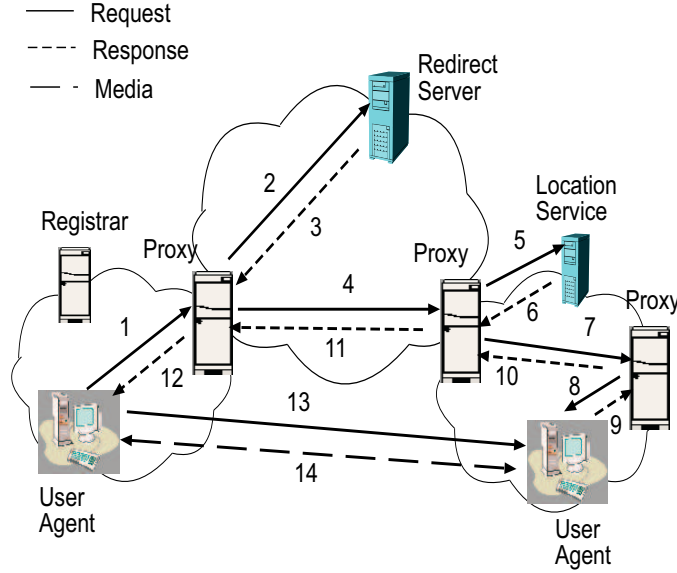


Fig. 1. SIP architecture

A session is setup between two user agents, each corresponding to a SIP endpoint, 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, REGISTER, REFER, etc., are routed by the proxy to an appropriate target UAS, based on the target SIP AOR included in the **Request-URI** field of the request messages. Proxies query location and redirect servers for SIP endpoint discovery corresponding to the target SIP AOR. SIP endpoint discovery and message routing has been explained with more details in Section III.

For reasons of scalability, multiple proxies are used to distribute the signaling load [21]. 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 [48]. 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) [17] 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, which are exchanged through SDP after the completion of SIP endpoint discovery, are 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.

SIP provides a general framework for event notifications [47], whose purpose is to allow SIP endpoints to receive notifications from remote endpoints indicating that an event has occurred. Two SIP methods, SUBSCRIBE and NOTIFY, are used in the framework. SIP is also used for Instant Messaging and Presence (IMP) applications [54]. According to the definition, an IMP system allows users to *subscribe* to each other and be *notified* of changes in state, and for users to send each other short instant messages in real-time. When one user wishes to send an instant message to another, the sender formulates and issues a SIP request using the MESSAGE method defined as an extension. However, this extension allows messages to be exchanged outside a session, independently each-other, and it is better suited for short messages exchanges. SIMPLE uses Message Session Relay Protocol (MSRP) for session-mode messaging [6], where instant messages are exchanged in the context of a session. The MSRP protocol is rather simple, as it uses only two primitives: (i) SEND: for sending instant messages between endpoints (ii) VISIT: for establishing MSRP sessions. Interested readers can refer to [48] for further details on SIP.

Without going into further details of the above procedures and protocols, it can be axiomatically assumed that the SIMPLE messages such as SUBSCRIBE, NOTIFY, MESSAGE, SEND, are typically request messages and are semantically similar to the SIP INVITE message. In other words, all these messages use the same mechanism as that of the INVITE message to reach the destination. For convenience, in the rest of the paper we will use the SIP INVITE message to represent the entire suite of SIP request messages.

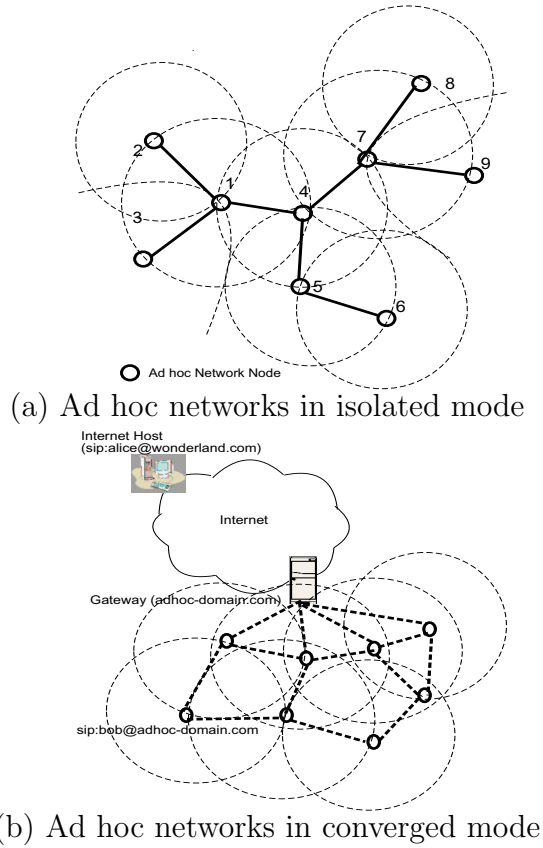


Fig. 2. Examples of ad hoc networks

### III. SIP IN AD HOC NETWORKS: THE PROBLEM STATEMENT

As mentioned before, SIP uses an overlay infrastructure of the Internet and centralized SIP servers for the discovery of SIP endpoints and the subsequent establishment of a session between the endpoints. Each user registers the AOR and the corresponding contact information of the SIP endpoint with a registrar server. Usually, the contact information is IP address or DNS resolvable SIP URIs. However, other types of contact information, such as telephone numbers, can also be registered. This association between the SIP AOR and the contact information is called the binding. During the SIP endpoint discovery by the proxies (the proxy functionalities are explained later on) SIP registrars make use of an abstract location service and return the binding information for the SIP AOR. SIP standard presumes the SIP AOR of the called party is known beforehand. However, discovery of an unknown AOR corresponding to a desired user or device can also be done with the help of service discovery protocols such as Service Location Protocol (SLP) [60] and Jini Network

Technology (Jini) [22].

SIP relies on some intermediate entities called the proxy servers to process the SIP requests, extract the SIP AOR and send them to the appropriate registrar server to retrieve the current binding information. The registrar and the proxies are logical entities, which are either pre-configured or are determined using DNS services.

A basic SIP session setup involves the calling SIP endpoint or UAC contacting a proxy server (typically, a UA is manually configured with an outbound proxy, or can learn about one through auto-configuration protocols) and sending through it the SIP request messages to the proxy server corresponding to the callee's SIP endpoint or UAS. The callee side proxy server then contacts the local location service to resolve the current bindings and eventually delivers the request message to the intended SIP UA. The intermediate proxies perform a processing based on the Route header field of the SIP request messages, which can be summarized as follows.

- The proxy inspects the **Request-URI** field of the SIP request message to extract the target SIP AOR. If it indicates a resource owned by this proxy, the proxy replaces it with the results of running a location service. Otherwise, the proxy does not change the **Request-URI**.
- The proxy inspects the URI in the topmost **Route** header field value. If it indicates this proxy, the proxy removes it from the **Route** header field.
- The proxy forwards the request to the resource indicated by the URI in the topmost **Route** header field value or in the **Request-URI** if no **Route** header field is present. The proxy determines the address, port and transport to use when forwarding the request by applying the DNS procedures described in [49] to that URI.

Ad hoc networks, as defined by the MANET IETF WG [35], are in general considered to be a collection of wireless mobile nodes that dynamically form 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. The main characteristics of ad hoc networks are as follows: (i) There is no fixed infrastructure. (ii) The topology changes unpredictably as nodes move. (iii) The wireless links are of low bandwidth, high error rates, and are possibly asynchronous in nature. (iv) The nodes are portable, limited in power with limited range and limited computing capacity. The issues arising out of the above characteristics are discussed in [46].

Ad hoc networks can operate primarily in two modes - either in *isolated* mode or in *converged* mode. In the former case, nodes form a self-sustained network without any connection to an external network. In the latter case, a gateway acts as the interface between the Internet (or an external network) and the ad hoc network, and any communication between these two networks takes place through the gateway.<sup>1</sup> The gateway assumes the responsibility of locating the desired ad hoc network node for the purpose of establishing a connection with the Internet.

Typically such an ad hoc network constitute a separate domain (e.g., `adhoc-domain.com`) and in the converged mode the gateway provides the DNS and the registrar services for the domain. Potentially, in the converged mode any Internet node can move into this domain and attach itself to a SIP URI corresponding belonging to this domain (e.g., `sip:visiting@adhoc-domain.com`), which serves as the current contact address.

Figure 2(a) shows an example of an ad hoc network in an isolated mode 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 addresses or by an internal addressing scheme and the ad hoc network routing protocols [8], [24], [31], [43], [44] take the responsibility of sending packets to these nodes once their addresses are known. Similarly, Figure 2(b) shows an example of an ad hoc network in the converged mode. A gateway connects the ad hoc network as an independent domain, *viz.* `adhoc-domain.com` in the Internet and hides all the internal details of the ad hoc network from the Internet. Thus if a SIP endpoint in the Internet *viz.* `sip:alice@wonderland.com` wants to establish a session with `sip:bob@adhoc-domain.com` then the gateway act as a proxy in the session setup procedure and locates the SIP endpoint corresponding to `sip:bob@adhoc-domain.com` followed by message routing to the appropriate SIP endpoint.

In both the cases however, the normal procedure of SIP AOR binding resolution and the routing of SIP request messages based on some centralized and often pre-configured entities, cannot be applied. The associated problems are described below:

- The resolution of the AOR binding is not possible with a centralized registrar as in an ad hoc networks all the nodes are mobile. The only option is to enquire about

<sup>1</sup> Note that the Internet gateway selection procedure in an ad hoc network is a challenging problem by itself [50], [57], [62] entailing the solutions to a number of problems such as gateway discovery, load balancing, traffic engineering, etc., and is outside the scope of this paper.

the binding information to the node directly responsible for the concerned SIP AOR. Since the nodes are all mobile, locating and enquiring a particular node in an ad hoc network becomes a challenging problem.

- The routing of SIP messages through an overlay of some fixed server (proxy) nodes is also not possible in an ad hoc network due to node mobility. The solution to this problem is to use the underlying ad hoc network routing protocol for SIP message routing, either in an integrated fashion or in an independent way.

It can be argued, however, that in the converged mode the SIP AOR binding resolution can always be done by the static gateway provided all the mobile nodes register with the gateway. But such a strategy leaves the problem of routing of SIP messages unaddressed. Moreover, such a mechanism would be very inefficient or non-scalable for internal end-to-end connections i.e. a connections between mobile nodes as in this case the gateway needs to be contacted for every SIP endpoint update and resolution operation.

In this paper, we address the above mentioned issues. We adopt two approaches: (i) *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.

#### A. Routing Protocol Considerations

There are pre-dominantly two types of ad hoc routing protocols, which either adopt a *proactive routing strategy* or a *reactive routing strategy*. In a proactive strategy, the results are computed based on 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 [43], OLSR [8] and that of reactive routing protocols are DSR [24], AODV [44]. Both the proposed LCA and TCA can be potentially integrated with either type of routing protocol. However, several research studies [23], [31] 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

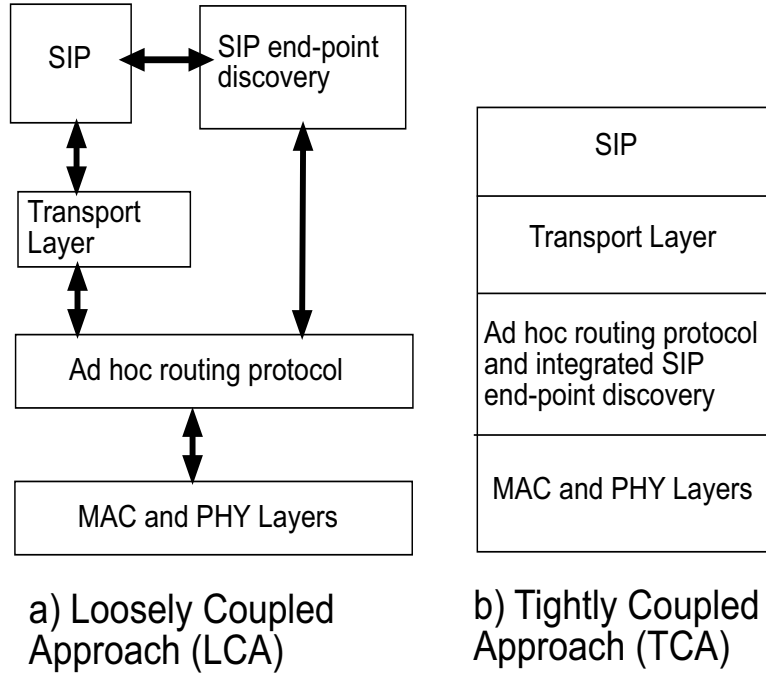


Fig. 3. Loose and tight coupling of SIP end-point discovery with ad hoc network protocols

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” [59] and unacceptable delay in route discovery process. A trade-off is generally done in such cases with cluster based routing [2], [9], [14], [20], [30], [33]. 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 traffic. Cluster based routing essentially creates a virtual topology with the cluster heads forming the backbone network. Additionally, cluster based routing protocols inherently supports the rendezvous functionality by the virtue of its virtual topology. The advantages of the cluster based routing can be fully exploited only when the SIP end point discovery is integrated with the routing protocol. Hence, we have proposed the integration of a fully distributed cluster based routing protocol, similar to other distributed clustering approaches [1], [20], with SIP under the TCA. Note, that in LCA, the selection of underlying protocol is not important as the SIP endpoint

discovery is completely independent of it. For performance comparison, however, we have used a reactive routing protocol, AODV [44], and a cluster-based routing protocol, Cluster Based Routing Protocol (CBRP) [20], with LCA.

#### IV. RELATED WORK

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 [41] to extend SIP so that it could 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 is not scalable for large ad hoc networks and also incurs unnecessary control overhead.

SIP based mobility management in ad hoc networks was considered in [10]. But, the authors have assumed a hierarchy of nodes with gateways within 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 [36] 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 [41] and hence suffers from the same drawbacks of scalability and high control overhead.

Although the fully distributed deployment of SIP for ad hoc networks proposed in [32], solves some of the above mentioned problems specific to ad hoc networks as well as the issue of legacy application portability to the ad hoc network domain, it fails to address the issue of scalability. The binding resolution is done by maintaining at each node a local cache, containing the binding information of all other nodes. However, the solution has been proposed only for small single-hop ad hoc networks and hence does not scale for large ad hoc networks envisioned by the IETF MANET WG.

Another attempt of enabling SIP in ad hoc network has been made in [11] in the context of implementing an application layer signaling system in ad hoc networks for managing multiparty sessions. JXTA technology [26] has been used to develop a middleware responsible for peer discovery and providing transport service to the signaling layer i.e., SIP. But there are at least two problems with this particular effort. First of all, as indicated in [26], JXTA is not suitable for dynamic networks such as ad hoc networks. Secondly, the details of the

integration between the SIP endpoint discovery and the middleware peer discovery are not furnished.

SIP end point discovery in an ad hoc network is semantically similar to the service or peer discovery process in 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 [39] followed this approach. These systems are often non-scalable and have a single point of failure, which makes them unfit for ad hoc networking environments. Systems such as Gnutella [13], Skype [55] 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 [56] protocol, creates a highly scalable structured overlay using hash tables to map the services and peers to the respective contact information. Motivated by the similarity in the problem domain, efforts have been made recently to empower SIP with Chord protocol [51] and enable it operate in P2P networks. Other similar proposals [4], [25], [37], [52] also use some kind of P2P protocols for the same purpose. However, Chord and other P2P protocols cannot be applied to independent node addressing scheme and does not support random node mobility as is the case with mobile ad hoc networks. Moreover, it does not consider the underlying routing topology, which is an important criterion for ensuring routing efficiency, particularly in the context of ad hoc networks. Hence, this approach is not particularly suitable for mobile ad hoc networks.

The problem of SIP endpoint discovery in ad hoc networks is also functionally similar to the service [7], [12], [18], [29] or resource [45] discovery problem in distributed environments such as that in ad hoc networks, mobile agent networks, grids. Several protocols have been proposed, targeted to each of these environments, while the basic design approaches remain the same i.e. either they are based on a centralized entity [58] or they employ some

king of flooding mechanism for the discovery process [45]. Some protocols [7], [18] have been proposed to solve these problems by using some kind of application level clustering or grouping. However, the possibility of integration with the routing layer has been ignored by all of them. This possibility has been identified as a need for ad hoc networks in several later studies [12], [29].

### V. LCA: LOOSELY COUPLED APPROACH

LCA works on top of the ad hoc routing protocol and 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, derived from AODV RREQ and RREP messages respectively, to locate the node corresponding to a target SIP AOR. The format of the SIPRREQ message is shown in Figure 4.

#### SIPRREQ Message

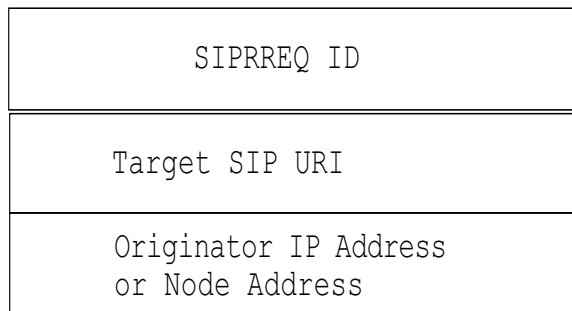


Fig. 4. Format of the SIPRREQ message

**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 AOR of the remote target with which the requesting party wants to setup a session.

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 the consequent “broadcast storm” problem, the originating node uses an expanding ring search technique. The time

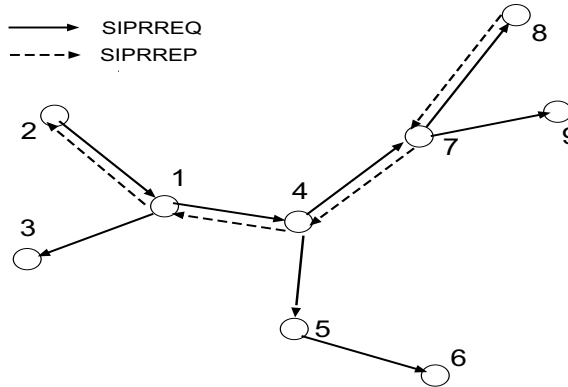


Fig. 5. The loosely coupled approach (LCA)

to live (TTL) for the SIPREQ 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 SIPREQ 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 SIPREQ. Typical values of the TTL related parameters used in the discovery process can be obtained from the AODV recommendations [44]. The target node address can be determined when the SIPREQ message reaches the target node or gets a “fresh enough” mapping of the SIP AOR 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 resolved 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 SIPREQ message with same or higher SIPREQ ID than that of the current request. When generating the SIPRREP, the node copies the **Target SIP URI**, the **SIPREQ ID**, and the **Originator IP Address** or **Node Address** from the SIPREQ message. The SIPREQ 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 SIPREQ

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.

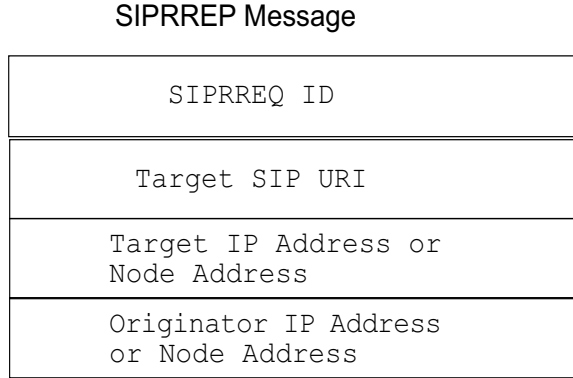


Fig. 6. Format of the SIPRREP message

## VI. TCA:TIGHTLY COUPLED APPROACH

TCA is an integrated approach where the SIP endpoint 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. Essentially, the protocol is not fundamentally different from the existing maximum degree based cluster head selection approach [1], [20], [33]. However, here we have focused on the integration of the cluster based routing protocol with SIP and its performance evaluation in SIP context. For the purpose of the thorough understanding of the integration procedure we have explained the protocol details of the cluster based routing protocol. We have also extended the routing protocol to accommodate the resource heterogeneity of the ad hoc network nodes.

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. We shall show later in the appendix that 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).<sup>2</sup> 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 in the network. Moreover, in our proposed integration scheme, the cluster heads host the SIP registrars and proxys and it is desired to have minimum number of such entities in the network as they tend to increase the number of hops for SIP messages besides increasing the energy expenditure at the nodes. We shall show later in the appendix that the protocol builds such an MDS using a greedy strategy of selecting cluster heads with maximum degree based on local information such as node degrees of neighbors only. The local information is gathered by means of periodic HELLO message that each node broadcasts to its 1-hop neighbors at an interval of HELLO\_PERIOD.

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 roles 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 a 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

<sup>2</sup> Finding a Minimum Dominating Set is, however, an NP-complete problem that can be mapped to the well-known set covering problem.

cluster heads which are 2 hops away.

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` and the `Clusterheadflag` fields are set to 0, and both the adjacency tables are kept empty.

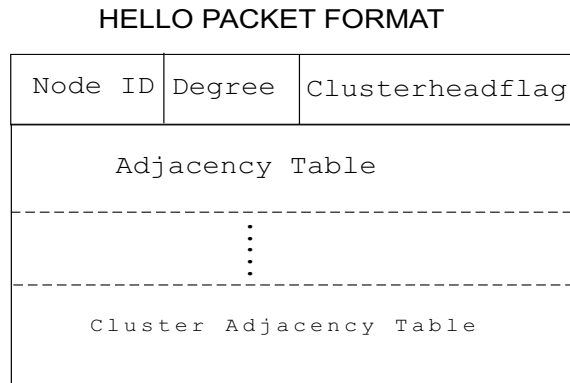


Fig. 7. Format of the HELLO message

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 each 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, after the second round of HELLO message, are still set to 0. It is with the next round of HELLO message broadcast that each node gets to know the degree of its neighbors. Hence after receiving at least three HELLO messages, 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.

ADJACENCY TABLE

Neighboring Node ID	Degree	Clusterheadflag
Neighboring Node ID	Degree	Clusterheadflag
Neighboring Node ID	Degree	Clusterheadflag
.....		
Neighboring Node ID	Degree	Clusterheadflag

Fig. 8. Format of the adjacency table

TABLE I  
NOTATIONS

$cluster(j)$	Cluster head corresponding to node $j$
$degree(j)$	Degree of node $j$
$clusterheadflag(j)$	Clusterheadflag of node $j$ denoting whether it is a cluster head or not; 1 denotes a cluster head; 0 denotes either a cluster member or not yet assigned; -1 denotes the node has the highest degree in the neighborhood but not sufficient resource;
$\mathcal{N}_1(j)$	Set of nodes 1-hop away from node $j$

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 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*.

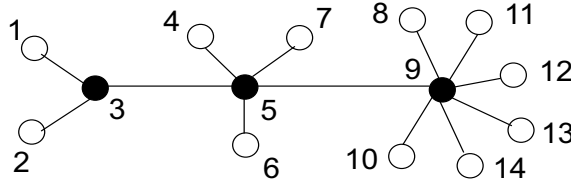
Once the cluster heads get selected, they assume the extra responsibility of acting as SIP proxies and registrars. In other words, they act as the inbound and outbound proxies for the SIP messages from the respective cluster members and keep a mapping of the SIP URIs and the node address of all of their respective cluster members. After a node gets selected as the cluster head, the **Clusterheadflag** field of its HELLO message is set to 1 to let the neighbors know of their cluster head.

```

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:     $clusterheadflag(i) = 1$ ;
11:   end if
12: end for

```

Fig. 9. Cluster Head Selection Algorithm

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

### 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 at least 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 sends 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.

```

1: Clustering Protocol;
2: for each network node  $i$  do
3:   Send HELLO packet to neighbors and update the ad-
   jacency table;
4:   Sleep for HELLO_PERIOD time period;
5:   Send HELLO packet with updated adjacency table;
   {The degree field of each entry assigned to 0 during
   bootstrapping};
6:   Sleep for HELLO_PERIOD time period;
7:   The degree field of the adjacency table is updated
   based on the previous round of HELLO packets and
   sent to the neighbors;
8:   The adjacency table is updated; {At this point each
   node has the degree information of its 2-hop neigh-
   bors};
9:   Sleep for HELLO_PERIOD time period;
10:  checkClusterhead( $i$ );
11:  Sleep for HELLO_PERIOD time period;
12:  Select gateway nodes between two cluster heads,
   which are 3-hops away from each other with the help
   of the cluster adjacency table;
13:  for each HELLO_PERIOD time interval do
14:    if there is a change in adjacency table and cluster
    adjacency table due to node movements then
15:      Execute each of the above steps;
16:    end if
17:  end for
18: end for

```

Fig. 12. Distributed cluster formation protocol

#### D. Function of SIP Registrars and Proxy Servers

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 [48].

## *E. Routing Procedure*

### E.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 at most 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. The algorithm for the cluster formation and maintenance protocol is presented in Figure 12.

### E.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 AOR 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 [48] can potentially be used to achieve this. If any of the neighboring proxies has the target AOR 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 [48]. 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.

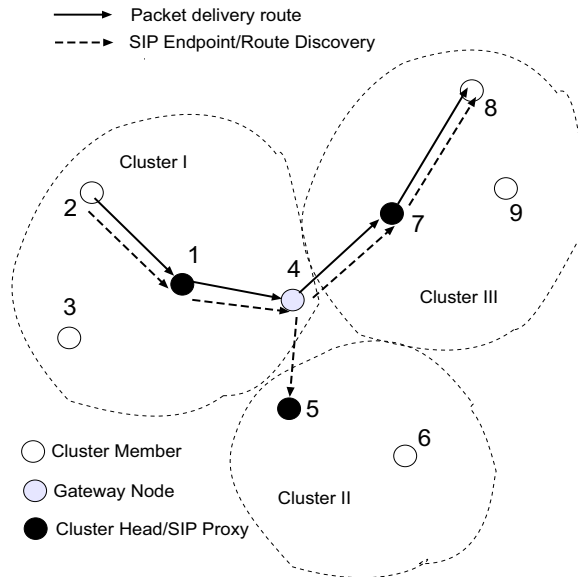


Fig. 13. Protocol Operation: Cluster Formation and Route discovery

Figure 13 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 AOR 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.

#### *F. Cluster-Based Protocol with Node Resource Constraint*

Typical resource constraints affecting the cluster formation procedure are residual battery power left in the nodes, computing power of the nodes etc. For example, a node elected as the cluster head may not have sufficient resource to become the cluster head and host the SIP registrar and the proxy. In that case a new election procedure is required to elect a new capable cluster head. Now, there are potentially two approaches to consider these constraints in the cluster formation procedure. First, these information is collated in each node and sent to the neighbors by adding them to the HELLO message. But, it is very difficult to define these parameters in generic terms so as to incorporate them in the protocol message format. Moreover, variable number of resource parameters would render the HELLO message transmission non-scalable. The second approach is that the potential cluster heads in terms of node degree takes a decision locally on whether it is going to elect itself as a cluster head or not based on its resource. In this case, the cluster formation protocol shown in Figure 12, remains the same but the **checkClusterhead** procedure needs to be modified as shown in Figure 14.

## VII. PERFORMANCE EVALUATION

In this section we evaluate the performance of SIP based session setup for LCA and TCA. For LCA, we have considered AODV [44] and CBRP [20] as the underlying routing protocol. When LCA is used on top of CBRP, the inherent cluster-based routing mechanism implicitly

```

1: checkClusterhead( $i$ );
2: for ( $j \in \mathcal{N}_1(i) \cup \{i\}$ ) do
3:    $cluster(j) = j$ ;
4:   for ( $k \in \mathcal{N}_1(j)$ ) && ( $clusterheadflag(k) \neq -1$ ) 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:    if (Resource constraint for node  $j$  is satisfied based on the
11:     number of neighboring nodes) then
12:       $clusterheadflag(j) = 1$ ;
13:    else
14:       $clusterheadflag(j) = -1$ ;
15:    end if
16:  end if
17: end for

```

Fig. 14. Cluster Head Selection Algorithm

restricts the broadcast of SIP endpoint discovery message defined in LCA, so that they are communicated only through the neighboring cluster heads, which then get broadcasted to the corresponding cluster members.<sup>3</sup> We compare the two approaches with respect to the following two important performance metrics: (i) the delay or latency in discovering a SIP end point and 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 the two approaches.

#### A. Simulation Experiments

We have performed extensive simulation experiments with ns2 [40]. 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

<sup>3</sup> Note, that in LCA with CBRP the SIP endpoint discovery layer is still decoupled from the underlying routing layer. Although LCA with CBRP solves the broadcast storm problem, like that in TCA for some cases (see Section C), it is incapable of exploiting the service discovery benefits of the routing protocol as is done by the latter. Consequently, as we shall see in Section B, there is negligible performance gain with respect to latency for SIP session establishment with LCA using CBRP.

nodes move randomly in a  $650m \times 650m$  square area, following a random waypoint mobility model [24]. 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 [44]. The resource constraints for the nodes have not been considered in the distributed cluster-based routing protocol.

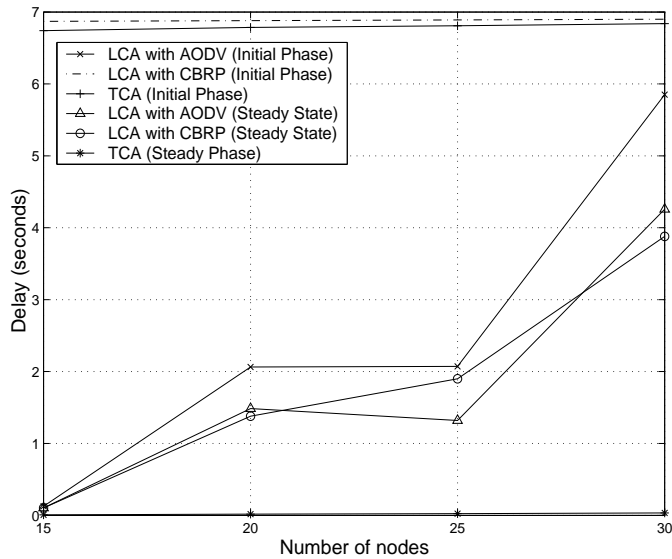


Fig. 15. Latency in SIP URI discovery in a static multihop network

## B. Latency

### B.1 Latency for static multihop networks

The latency involved in discovering a node with a particular SIP AOR and establishing a session in a static multihop network is presented in Figure 15. Two phases of SIP endpoint 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 or TCA than that of LCA with AODV. 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 endpoint discovery. Also, in LCA with AODV the discovery message is incrementally broadcasted each time a

SIP endpoint 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. The latency in the initial phase for LCA with CBRP, however, is comparable with TCA because the transmission of SIP messages following the SIP endpoint discovery process takes as much time as in TCA, both being cluster based routing protocols. In the second phase also there is no significant improvement in latency when LCA is used with CBRP. This is because, the SIP endpoint discovery mechanism in LCA is independent of the underlying routing layer, and the discovery messages are broadcasted much in the similar fashion as with AODV, albeit exploiting the clusters formed in CBRP. In fact, for some cases the latency for LCA with CBRP could be more than LCA with AODV since in the former case the discovery messages have to follow the route through cluster heads only and immediate neighbors (in terms of physical proximity) might have to take a longer route to communicate among each other. 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. In addition to the above factors in favor of TCA, the integration of SIP endpoint discovery messages with the SIP messages in TCA further improves the latency. In contrast, the latency is comparatively more in LCA as the discovery process is decoupled from the SIP session establishment process and additional delay is incurred during the session establishment procedure.

## B.2 Latency when the destination moves

Figure 16 shows the latency figure in discovering a SIP AOR 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 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. Here also, because of the above mentioned reasons, the latency does not improve

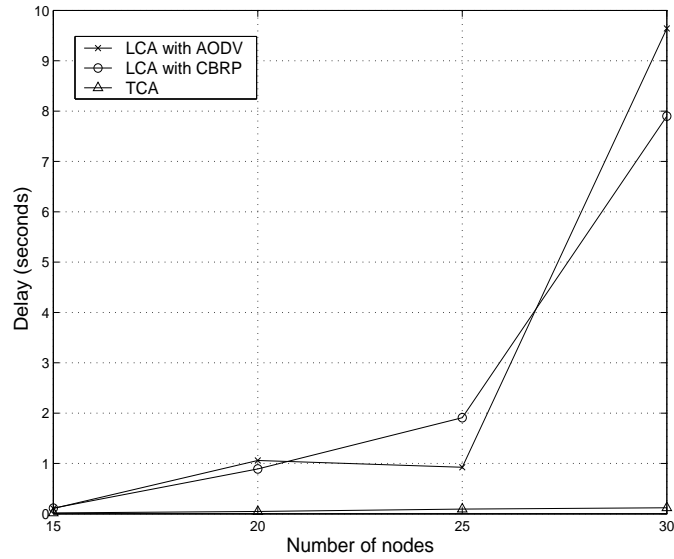


Fig. 16. Latency in SIP URI discovery when the destination moves

even when LCA is used with CBRP.

### B.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 17. In this case also, TCA shows considerable resilience and fares better than LCA in terms of the latency in SIP endpoint discovery. In this case also, because of the decoupling of routing layer and the SIP endpoint discovery process in LCA, the latency does not improve even when CBRP is used for LCA.

### B.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

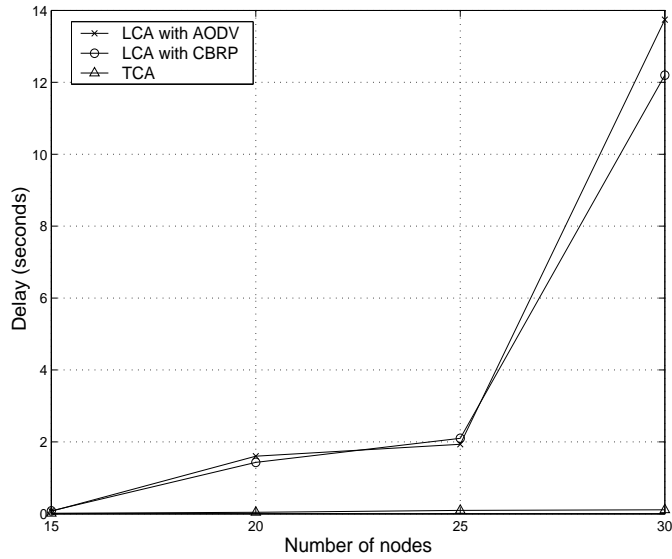


Fig. 17. Latency in SIP URI discovery when the proxies moves

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 18. 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 with AODV, 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 endpoint. For LCA with CBRP, however, the results are somewhat different from the previous cases with static nodes or with isolated node movements. The reason is that the discovery procedure and the subsequent transmission of SIP messages get delayed due to the re-arrangement of the virtual routing infrastructure of CBRP because of random node mobility.

### 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 and CBRP messages along with the SIPRREQ and SIPRREP messages for the LCA and all the routing related messages for the TCA. For TCA, the routing related messages as well

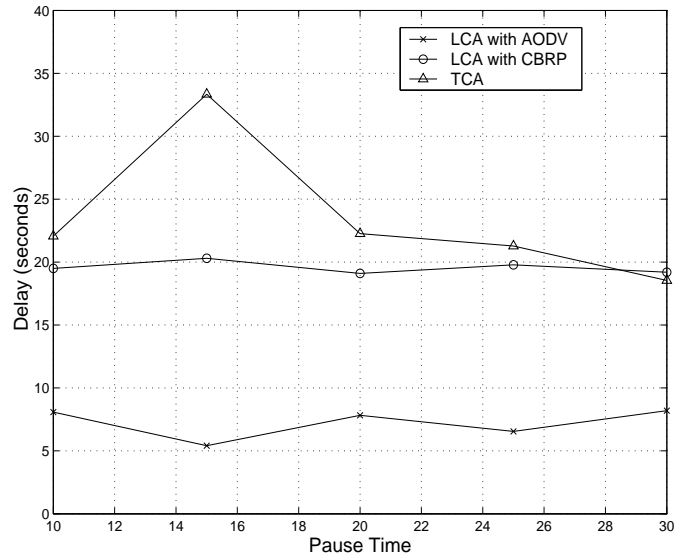


Fig. 18. Latency in SIP URI discovery with random node mobility

as the SIP request messages take part in the SIP endpoint discovery and are considered to constitute the control overhead. 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.

### C.1 Control overhead for static multihop networks

The control overheads associated with the two approaches for the static multihop wireless network are shown in Figure 19. LCA with AODV has lower control overhead than that of TCA, initially. But, as the network grows larger, the overhead in LCA with AODV starts increasing rapidly and overshoots (for networks with more than 23 nodes) the corresponding overhead of TCA. This happens because the number of message broadcasts increases drastically in LCA with AODV with the increase in the number of network nodes. For LCA with CBRP, however, the broadcast remains scalable because of the underlying clustering in the routing layer. The control overhead remains scalable in TCA also for the same reason.

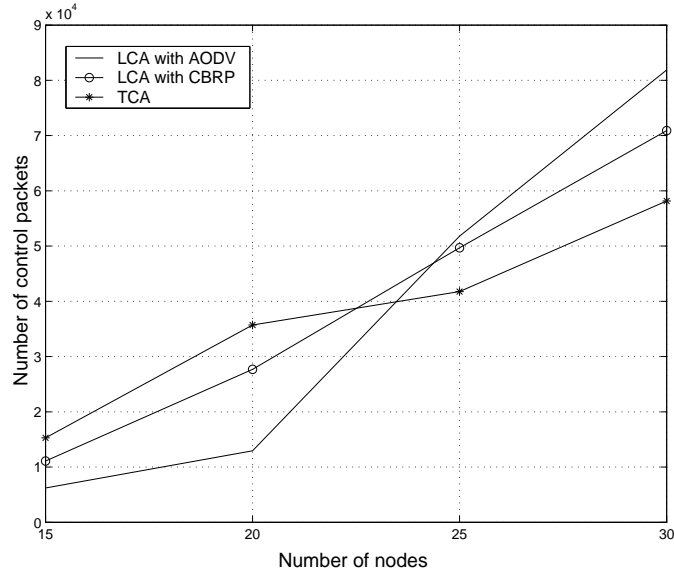


Fig. 19. Control overhead for static multihop networks

## C.2 Control overhead with random node mobility

Figure 20 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. Note that, in this case the “broadcast problem” persists even for LCA with CBRP. This happens because of the drastic changes in the virtual infrastructure in the face of random mobility resulting in the SIP endpoint discovery, depending on the expanding ring search based discovery procedure defined in LCA, facing the same broadcast storm problem.

## VIII. 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 set up, 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

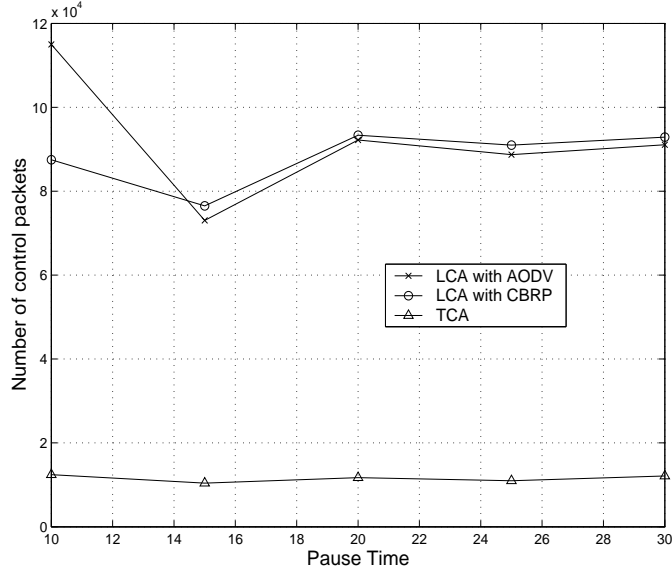


Fig. 20. Control overhead with random node mobility

when the proxies related to the source and the destination nodes move, although 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 over 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. Besides the subsequent routing of SIP messages takes its own time for session establishment. Although scalable broadcast of SIP endpoint discovery message is possible in LCA with CBRP as the underlying routing protocol, the latency does not improve because of the decoupling of the discovery procedure with the routing layer, and the expanding ring search technique employed by LCA. The scenario with respect to latency, however, changes completely when all the nodes move randomly. LCA, in this case performs better or as good as 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”. This problem is not even solved when CBRP is used with LCA.

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 relatively 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.

## IX. CONCLUSION

In this paper, we have proposed two approaches to enable SIP applications in ad hoc networks. One is a loosely coupled approach, where the SIP endpoint discovery is completely decoupled from the underlying routing protocol, while the other is a tightly coupled approach. We have used a cluster based routing algorithm integrated with the SIP endpoint discovery for the tightly coupled approach. Apart from having better performance in relatively 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 generally has more desirable performance figures. We have also presented a variation of the cluster based routing algorithm considering node resource constraints. 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 and the design and deployment of specialized SIP based applications in our future work.

## APPENDIX

**A. 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.

**B. Proof of Correctness for the Cluster-based Protocol:**

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 sectionC.
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  $v_n \in V$  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.

### C. Lemmas:

**Lemma 1:** Selection of cluster heads based on maximum degree criteria yields a minimal dominating set (MDS).

**Proof:** Before going into the proof let us define a *dominating set*. In a graph  $G$ , a set  $S \subseteq V(G)$  is a dominating set if every vertex not in  $S$  has a neighbor in  $S$ .

Now, let us assume a process where in each step we select a node (or vertex in the graph representation of the network) as the cluster heads or include it in the dominating set, based on some criteria, followed by the selection of the set of nodes neighboring it. According to our cluster head selection procedure the criterion that we have considered is a greedy criterion based on maximum node degree. Now, let there be  $\beta$  non-selected nodes after a certain step of the process. Then, in the next step after the selection of a node  $x$  as a cluster head, there will be  $(\beta - (j + 1)/n)$  non-selected nodes, where  $j$  is the degree of node  $x$  and  $n$  is the total number of nodes in the network. Evidently, if  $j$  is maximum, the number of remaining non-selected nodes is minimum, by the virtue of which we can claim that the number of steps in the selection procedure for cluster heads is also minimized. Since the number of steps are minimized, we can infer that the number of nodes considered in the set of cluster heads is also minimized yielding a minimal dominating set.

**Lemma 2:** A cluster head gets elected in at most  $O(|V|)$  steps under the modified protocol operation with resource constraint, where  $V$  denotes the set of the nodes.

**Proof:** The `checkClusterhead` algorithm ensures that at each instance of its call at least one node is eliminated (i.e., `clusterheadflag` of the node is assigned  $-1$ ) given that there are one or more nodes with highest degree among its neighbors, but with insufficient resource to become the cluster head. Evidently this elimination process can go on for at most  $O(|V|)$  times before at least one cluster head is finally elected and at least one cluster is formed.

## REFERENCES

- [1] D. J. Baker and A. Ephremides, "The Architectural Organization of a Mobile Radio Network via a Distributed Algorithm," *IEEE Trans. Comm.*, Vol. 29, No. 11, pp. 1694-

- 1701, 1981.
- [2] 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.
  - [3] S. Berger, H. Schulzrinne, S. Sidiroglou, and X. Wu, "Ubiquitous computing using SIP", *NOSSDAV*, pp. 82 - 89 , 2003.
  - [4] D. A. Bryan and C. Jennings, "A P2P Approach to SIP Registration," *Internet Draft draft-bryan-sipping-p2p-00.txt*, January 2005.
  - [5] B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging". *RFC 3428*, December 2002.
  - [6] B. Campbell, R. Mahy, C. Jennings , "The Message Session Relay Protocol". *Internet Draft (Work in Progress) draft-ietf-simple-message-sessions-07.txt* , July 2004.
  - [7] D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha, "GSD: A Novel Group-based Service Discovery Protocol for MANETs," *IEEE Conference on Mobile and Wireless Communications Networks (MWCN)*, 2002.
  - [8] T. Clausen, and P. Jacquet, " Optimized Link State Routing Protocol (OLSR)", *IETF RFC 3626*, October 2003.
  - [9] 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.
  - [10] A. Dutta, K. D. Wong, J. Burns, R. Jain, A. McAuley, K. Young, and H. Schulzrinne, *MILCOM 2002*, Vol. 1 , pp. 448 - 454, 2002.
  - [11] C. Fu; R. H. Glitho, R. Dssouli, "A novel signaling system for multiparty sessions in peer-to-peer ad hoc networks", *IEEE Wireless Communications and Networking Conference*,

- Vol. 4, Page(s):2287 - 2292, 2005.
- [12] J. A. Garcia-Macias and D. A. Torres, "Service Discovery in Mobile Ad-Hoc Networks: Better at the Network Layer?," *International Conference on Parallel Processing Workshops (ICPPW05)*, Page(s):452 - 457, 2005.
- [13] Gnutella webpage: <http://gnutella.wego.com/>.
- [14] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets. Algorithmica," Vol. 20, No. 4, pp. 374-387, 1998.
- [15] E. Guttman, C. Perkins, J. Veizades, M. Day, "Service Location Protocol, Version 2", *RFC 2608*, 1999.
- [16] 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).
- [17] M. Handley, and V. Jacobson, "SDP: Session Description Protocol", *IETF RFC 2327*, April 1998.
- [18] S. Helal, N. Desai, V. Verma, and C. Lee, "Konark – A Service Discovery and Delivery Protocol for Ad-hoc Networks," *IEEE Conference on Wireless Communication Networks (WCNC)*, Vol. 3, Page(s): 2107 - 2113 , 2002.
- [19] International Telecommunication Union, "Packet based multimedia communications systems", *Recommendation H.323, Telecommunication Standardization Sector of ITU*, Geneva, Switzerland, Feb. 1998.
- [20] M. Jiang, J. Li and Y. C. Tay, "Cluster Based Routing Protocol (CBRP)," *IETF Internet Draft draft-ietf-manet-cbrp-spec-01.txt*, August 1999.
- [21] W. Jiang, J. Lennox, H. Schulzrinne and K. Singh, "Towards Junking the PBX: Deploying IP Telephony", pp. 177-185 *NOSSDAV* 2001.
- [22] Jini Network Technology, <http://www.sun.com/software/jini/>

- [23] 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.
- [24] 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.
- [25] A. Johnston, "SIP, P2P and Internet Communications," *IETF Internet Draft draft-johnston-sipping-p2p-ipcom-00.txt*, January 2005.
- [26] JXTA Technology, <http://www.sun.com/software/jxta/>
- [27] 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.
- [28] P. Koskelainen, H. Schulzrinne, and X. Wu, "A SIP-based conference control framework," *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, pp. 53 - 61, 2002.
- [29] U. C. Kozat, L. Tassiulas, "Network layer support for service discovery in mobile ad hoc networks," *INFOCOM*. Vol. 3, Page(s):1965 - 1975, 2003.
- [30] 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.
- [31] 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.
- [32] S. Leggio, J. Manner, A. Hulkkonen, K. Raatikainen, "Session Initiation Protocol De-

- ployment in Ad-Hoc Networks: a Decentralized Approach,” *2nd International Workshop on Wireless Ad-hoc Networks (IWVAN)*, London, May 2005.
- [33] C. R. Lin and M. Gerla, “Adaptive clustering for mobile, wireless networks,” *Journal on Selected Areas of Communication*, Vol. 15, No. 7, 1997.
- [34] 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.
- [35] Mobile Ad-hoc Networks (manet), <http://www.ietf.org/html.charters/manet-charter.html>
- [36] 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/>
- [37] P. Matthews and B. Poustchi, “Industrial-Strength P2P SIP,” *IETF Internet Draft draft-matthews-sipping-p2p-industrial-strength-00.txt*, February 2005.
- [38] 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.
- [39] Napster webpage: <http://www.napster.com/>.
- [40] “The network simulator”, available at <http://www.isi.edu/nsnam/ns>
- [41] M. O’Doherty , “Pico SIP” , *IETF Internet Draft, draft-odoherty-pico-sip-00.txt*, January 2001.
- [42] 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.
- [43] 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.
- [44] C. Perkins, E. Belding-Royer, and S. Das “ Ad hoc On-Demand Distance Vector (AODV) Routing ”, *IETF RFC 3561*, July 2003.
- [45] G. P. Picco, A. L. Murphy, and G-C. Roman, “LIME: Linda Meets Mobility,” *ICSE 1999*, Page(s): 368-377, 1999.
- [46] R. Ramanathan and J. Redi, “A Brief Overview of Ad hoc Networks: Challenges and Directions”, *IEEE Communications Magazine*, Vol. 40, No. 5, Page(s): 20 - 22, 2002.
- [47] A. B. Roach, “Session Initiation Protocol (SIP) Specific Event Notification”, *RFC 3265*, June 2002.
- [48] 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.
- [49] J. Rosenberg and H. Schulzrinne, “SIP: Locating SIP Servers”, *IETF RFC 3263*, June 2002.
- [50] P. M. Ruiz and A. F. Gomez-Skarmeta, “Enhanced Internet connectivity for hybrid ad hoc networks through adaptive gateway discovery,” *IEEE International Conference on Local Computer Networks*, Page(s):370 - 377, 2004.
- [51] K. Singh and H. Schulzrinne, “Peer-to-peer Internet Telephony using SIP,” *Columbia University Technical Report CUCS-044-04*, Oct 2004.
- [52] Earthlink SIPShare, SIP-based P2P Content Sharing Prototype, <http://www.research.earthlink.net/p2p/>.
- [53] H. Schulzrinne et al. “RTP: A Transport Protocol for Real-Time Applications” *IETF RFC 1889* Jan 1996.

- [54] SIP for Instant Messaging and Presence Leveraging Extensions (SIMPLE) IETF Working Group. <http://www.ietf.org/html.charters/simple-charter.html>
- [55] Skype webpage: <http://www.skype.com/>.
- [56] 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.
- [57] A. Striegel, R. Ramanujan, J. Bonney, "A protocol independent Internet gateway for ad hoc wireless networks," *IEEE International Conference on Local Computer Networks*, Page(s):92 - 101, 2001.
- [58] S. Tarkoma, M. Laukkanen, "Supporting software agents on small devices," *First International Joint Conference on Autonomous Agents and Multiagent Systems*, Page(s): 565566, 2002.
- [59] 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.
- [60] J. Veizades, E. Guttman, C. Perkins, S. Kaplan, "Service Location Protocol," *IETF RFC 2165*, June 1997.
- [61] M. Weiser, "The Computer for the Twenty-First Century," *Scientific American*, 265(3), pp. 94-104, 1991.
- [62] J. H. Zhao, X. Z. Yang, H. W. Liu, "Load-balancing Strategy of Multi-gateway for Ad hoc Internet Connectivity," *International Conference on Information Technology: Coding and Computing*, Vol. II, Page(s): 592-596, 2005.