Multi-Level Hierarchies for Scalable Ad hoc Routing

Elizabeth M. Belding-Royer
Department of Computer Science
University of California, Santa Barbara, CA 93106
ebelding@cs.ucsb.edu

Abstract

Ad hoc networks have the notable capability of enabling spontaneous networks. These networks are self-initializing, self-configuring, and self-maintaining, even though the underlying topology is often continually changing. Because research has only begun to scratch the surface of the potential applications of this technology, it is important to prepare for the widespread use of these networks. In anticipation of their ubiquity, the protocols designed for these networks must be scalable. This includes scaling to both networks with many nodes, and networks with rapidly changing topologies. This paper presents two hierarchical clustering protocols that improve the scalability of ad hoc routing protocols. The Adaptive Routing using Clusters (ARC) protocol creates a one-level clustered hierarchy across an ad hoc network, while the Adaptive Routing using Clustered Hierarchies (ARCH) protocol creates a multi-level hierarchy which is able to dynamically adjust the depth of the hierarchy in response to the changing network topology. It is experimentally shown that these protocols, when coupled with an ad hoc routing protocol, produce throughput improvements of up to 80% over the ad hoc routing protocol alone.

1 Introduction

Mobile ad hoc networking has become increasingly popular in recent years as a way to provide instant networking between groups of people that may not all be within transmission range of one another. Applications for these types of networks range from campus and conference scenarios, to emergency operations, to military scenarios. The projected number of users in each of these applications ranges from a handful of people in some emergency situations, to tens and hundreds of people in campus and conference scenarios, to thousands and tens of thousands of people in military applications. While the popularity of these networks as a research topic has risen drastically in the recent years, these networks still remain an essentially untapped resource. Due to the lack of commercial applications for this technology, it is impossible to predict what the "killer app" will be, and the scale in which it will be used. Because of these reasons, the networking technology developed today must be scalable to accommodate a potentially large number of users in the near future. Due to the potential applications of ad hoc networks to traffic scenarios [6], the technology must also be scalable to networks of rapidly moving nodes.

To provide multi-hop communication in these networks, numerous routing protocols have been developed [10, 14, 18, 21, 22, 25, 26]. Many of these protocols can be placed into one of two classes [29]: proactive (table-driven) approaches, and reactive (on-demand) approaches. Proactive protocols are derived from the traditional distance vector and link state protocols commonly used in wired networks. These protocols maintain routes between each pair of nodes throughout the lifetime of the network. While this approach has the benefit that a route is generally available the moment it is needed, proactive protocols have poor scaling properties due to their $O(n^2)$ overhead. Additionally, previous work has shown these protocols to not perform as well as reactive routing protocols in most scenarios [7]. Reactive protocols, on the other hand, only establish routes *on-demand*, or when needed. These protocols thereby only incur overhead for route construction

and maintenance when those routes are actually needed, since they do not maintain routes that are not utilized. The drawback to these protocols is that they introduce a *route acquisition latency*, or a period of waiting to acquire a route after the route is needed. These protocols have been shown to also have limited scalability, due to their route discovery and maintenance procedures [19].

One alternative to these protocols for improving scalability is clustering, or hierarchical, routing protocols. Hierarchical protocols place nodes into groups, often called clusters. These groups may have some sort of cluster leader that is responsible for route maintenance within its cluster and between other clusters. The motivation for implementing hierarchical routing algorithms is that they tend to be more scalable, due to their intrinsic characteristics. Section 2 explains these characteristics in more detail.

This paper describes two solutions for hierarchical routing in ad hoc mobile networks. The first protocol, the Adaptive Routing using Clusters (ARC) protocol, creates a one-level clustered hierarchy on a network of nodes. ARC is loosely based on the linked cluster architecture (LCA) presented in [2], in that similar cluster structures are created. Clusters are created with one cluster leader per cluster, and each node within the cluster must be within direct transmission range of the cluster leader. Inter-cluster communication is accomplished through the utilization of gateway nodes. While based on the LCA clustering scheme, there are some key differences between the two approaches. The method by which cluster leaders are elected in the two schemes differs, and ARC allows for multiple communication paths between clusters, whereas the LCA protocol is limited to just one such path. Further, ARC implements a new algorithm for cluster leader maintenance that avoids the *rippling effect* that is characteristic of other clustering protocols. The rippling effect results when one cluster leader change results in additional leadership changes in the network. While leadership changes are necessary so that the number of cluster leaders does not grow over time, these changes are expensive in terms of communication overhead and processing power. Hence they should be minimized.

A one-level hierarchy can achieve significant performance improvements in an ad hoc network, primarily in terms of increasing route robustness and therefore achieving higher data throughput. However, its benefit is still limited when networks grow to thousands or tens of thousands of nodes. Therefore, this paper also presents the Adaptive Routing using Cluster Hierarchies (ARCH) protocol. ARCH builds upon the foundations of ARC to create a multi-level hierarchy that is able to dynamically adjust its depth in response to the changing conditions of the network.

The remainder of this paper is organized as follows. Section 2 explains the motivation for hierarchical routing protocols and provides examples of how these protocols can be used to increase network scalability. Section 3 describes related work in the area of hierarchical routing protocols. Next, two protocols for hierarchical routing are presented. The Adaptive Routing using Clusters (ARC) protocol is described in section 4, and the Adaptive Routing using Cluster Hierarchies (ARCH) protocol is described in section 5. Section 6 presents an evaluation of the hierarchy created by the protocols, and compares ARC's performance against that of a reactive routing protocol. Section 7 provides further analysis of the results, and finally section 8 concludes the paper.

2 Hierarchical Routing Motivation

Numerous routing protocols [14, 22, 25, 26] have been developed for discovering and maintaining routes in ad hoc networks. Each of these protocols utilizes what is called a *flat routing* scheme. In a flat routing scheme, each node on a route records the physical next hop towards the destination as its next hop for that route. For example, in figure 1(a), node A would record node B as its next hop towards destination node D. Flat routing works well in small networks but has scalability limitations that degrade performance in larger, or rapidly moving, networks. With a flat routing scheme, whenever there is a link break in a route, the route is then broken until that link is repaired. After a link break, routing protocols typically either discover a new, viable route [22, 26], or else consult their route cache to determine whether an alternative route is known [14].

(a) Flat Routing

(b) Hierarchical Routing

Figure 1: Example of Routing.

However, even if another route exists in the route cache, there is no guarantee that route is still viable; it may have similarly become broken since the time that it was discovered. In large networks (i.e., over 500 nodes), paths can easily be on the order of 15 or more hops [19]. With routes of this length, link breaks can occur quite frequently. Similarly, in networks with rapidly moving nodes, link breaks are also highly likely to occur. Because some action must be taken each time a link break occurs, these protocols have limited scalability in large and/or rapidly moving networks.

The alternative to these flat routing solutions are hierarchical routing protocols. Hierarchical routing schemes are typically utilized by clustering protocols. Figure 1(b) illustrates an example of a typical clustering topology, which has been placed upon the network illustrated in figure 1(a). In this figure, clusters are created based on the proximity of the nodes. Typically, each cluster has a cluster leader, and all nodes in a cluster are within direct transmission range of the cluster leader. Nodes that lie within the transmission range of more than one cluster leader are called *gateways*, and can be used by the cluster leaders to route between the clusters. Nodes G_1 and G_2 are examples of gateways, and CL_1 and CL_2 are cluster leaders.

Instead of recording routes hop-by-hop as in the flat routing scheme, hierarchical routing protocols can record routes between clusters. Hence, in the figure, CL_1's next hop for the destination D is CL_2, *not* G_1. By recording CL_2 as its next hop along that route, CL_1 can use *any* gateway lying between the two clusters as an intermediate node to reach CL_2. For instance, it may select gateway G_1. The advantage of this solution is that if G_1 wanders out of the vicinity of either CL_1 or CL_2 and can no longer be used as a gateway, there is still another gateway, G_2, that can be used instead. Hence no route reconstructions need to occur, and data packet transmissions can continue without any disruption, i.e., without rebuilding the route. Further, as the network nodes continue to move, it is likely that other nodes, such as N_1 or N_2 will move into the overlapping area between the clusters and can then be used as gateways as well. Hence the robustness of the route is increased due to the greater routing flexibility. In routes with longer path lengths, the increase in robustness is much greater.

Hierarchical clustering has the additional advantage that it allows nodes to be addressed based on their position in the hierarchy. For instance, a node whose base address is z that is a member of level-1 cluster y and level-2 cluster x could have address $x.y.z$. If the node is a member of multiple clusters, it could feasibly have multiple addresses. The multiplicity of addresses allows for ease in locating nodes as they move through the network, since it is likely that one of the addresses is valid at any given time. However, this addressing scheme has the drawback that, if a node is a member of n different hierarchies, the amount of network resources required to distribute and store routing information increases by n times over that in a single hierarchy [30].

3 Related Work

There are numerous proposals for clustering and hierarchical routing schemes. This section presents a small sampling of those protocols, including those that are most closely related to the ARC and ARCH protocols. One-level clustering schemes are presented first, and multi-level hierarchical solutions follow. A comprehensive overview of different clustering strategies is presented in [32].

The linked cluster architecture (LCA) [2], developed for packet radio networks, is one of the earliest clustering protocols for these networks. The LCA protocol organizes nodes into clusters based on node proximity. Each cluster has a clusterhead, and all nodes within a cluster are within direct transmission range of the clusterhead. Clusterhead election is based on node identifiers, where the node with the largest identifier in a given area becomes the clusterhead. Gateways in the overlapping region between clusters are used to connect the clusters. LCA specifies that there most be only one such gateway designated to interconnect the clusters at any given time. Pairs of nodes can also be used to connect clusters if there are no nodes in the overlapping region.

Gerla and his team at UCLA have developed numerous clustering protocols [8, 11, 12, 20, 23]. Of these, the Clusterhead Gateway Switch Routing (CGSR) protocol described in [8] is most similar to ARC. In this protocol, packets are routed alternately between cluster leaders and gateways. The authors define various extensions that can be added to CGSR, such as priority token scheduling and gateway code scheduling, in order to control channel access. Additionally, they define a Least Cluster Change algorithm (LCC), which is designed to reduce the number of cluster leader changes, because such changes can result in a substantial amount of overhead.

A different approach to clustering is taken by Basagni in [3], which presents two clustering algorithms. The first of these is the Distributed Clustering Algorithm (DCA), which is intended for "quasi-static" networks in which nodes are slow moving, if moving at all. The other algorithm is designed for higher mobility and is called the Distributed and Mobility-Adaptive Clustering algorithm (DMAC). Both DCA and DMAC assign nodes different *weights* and assume that each node is aware of its respective weight. The weights are used in the determination of the cluster leaders. In the DMAC protocol, if two cluster leaders come into contact, the one with the smaller weight must revoke its leader status.

An additional approach is that taken by the Core Extraction Distributed Ad Hoc Routing (CEDAR) algorithm [31]. Instead of creating a cluster topology, CEDAR builds a set of nodes (i.e., a *core*) to perform route computation. Using the local state information, a minimum dominating set of the network is approximated to form the core. CEDAR establishes QoS routes that satisfy bandwidth requirements using the directionality of the core path. Link state and bandwidth availability is exchanged to maintain important information for computing QoS routes.

Kleinrock was an early pioneer of hierarchical routing schemes for static networks. In [15], Kleinrock and Kamoun investigated a hierarchical routing scheme with the goal of reducing routing table size. The authors determined that the length of the routing table is a strict function of the clustering structure. Clustering generally has the unwanted result of an increase in path length, and so the goal was to find an optimal clustering scheme that minimized this path length increase. It was determined that the number of entries in a node's forwarding table is minimized when the number of level-i clusters in each level-$(i+1)$ cluster is e, and the number of levels in the clustering hierarchy equals $\ln N$. In this case, the forwarding table contains $e \ln N$ entries.

The Landmark Routing technique [33] is a novel approach to building a hierarchy as it is based on landmarks, as opposed to transmission ranges. A landmark is a router whose location is known by its neighboring routers up to some radius. All routers within that radius know how to reach the landmark. A hierarchy of such landmarks is built by increasing the radius of some of the routers. Nodes have hierarchical addresses based on the landmarks with which they are associated. A source node routes to a destination by sending the packet to the lowest level landmark with which both nodes are associated. As the packet approaches the destination, the

granularity of routing knowledge about that destination improves, and so the packet can be accurately routed to the destination.

A more recent approach to hierarchical clustering is the Multihop Mobile Wireless Networks (MMWN) system [27]. The system divides nodes into switches and endpoints, where only switches can serve as packet routers. An ad hoc network is a special case where all nodes are switches. Each endpoint is associated with exactly one switch, and the switches are grouped together to form clusters. The clusters are in turn placed into hierarchical clusters, and so on. The MMWN approach requires all clusters to be disjoint. Cluster merges occur when the number of switches in a cluster falls below some merging threshold, N_{merge}. Similarly, cluster splits occur when the number of switches becomes greater than the splitting threshold, N_{split}.

4 One-Level Hierarchical Clusters

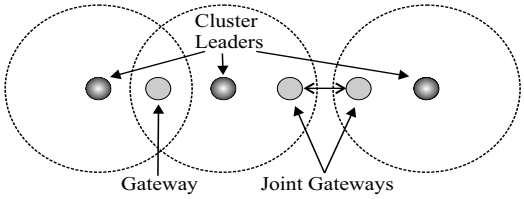


Figure 2: Cluster Gateways.

A one-level hierarchical clustering scheme can be used to create an overlay of clusters across a network. When a clustering scheme is used, routes can be recorded hierarchically, across clusters or cluster leaders, thereby adding more routing flexibility and greater robustness to network routes. As discussed in the previous section, there are many variations on the creation of clusters. The Adaptive Routing using Clusters (ARC) protocol is an example of such a scheme. ARC groups nodes into clusters based on the nodes' proximity to one another. Each cluster has a cluster leader. Nodes within a cluster are one-hop neighbors of the cluster leader. A node that is only one hop from more than one cluster leader is called a *gateway*, and can be used by the cluster leaders to route between the clusters. In a densely connected network, it is likely that two clusters will have multiple gateways between them. It is also possible for pairs of nodes to be *joint gateways*. Joint gateways are nodes that lie in separate clusters, but that are within transmission range of one another. Such pairs of nodes can also be used to route between clusters. The difference between gateways and joint gateways is illustrated in figure 2. Whenever both a gateway and a pair of joint gateways exists to connect two clusters, the single gateway is favored because it is one fewer hop.

Every node in the network has an associated status level that is set according to whether the node is a cluster leader, and if not how many clusters leaders with which it is within transmission range. Nodes can have the status level of either `cldr` if they are a cluster leader, `gateway` if they are a gateway, or `node` if they are an ordinary node (i.e., not a cluster leader or a gateway). Joint gateways also have status level `gateway`. Every node in the network must either be a cluster leader itself, or else it must be a member of a cluster, and therefore have a cluster leader.

The ARC algorithm uses a new method for determining when a cluster leader should revoke its leader status. This method is based on a subset property, whereby when one cluster becomes a subset of another cluster, the cluster leader of the subset cluster gives up its leader status and joins the superset cluster. This method results in minimal cluster leader changes during the lifetime of the network and prevents the *rippling effect*, as discussed in section 4.3.

4.1 Initialization

The cluster topology is initialized and maintained through the periodic transmission of *Hello* messages by each node. The Hello messages serve the purpose of announcing the presence of nodes and their current status level, as well as indicating connectivity information of the non-leader nodes. Hello messages from non-leader nodes include a list of those cluster leaders' IP addresses to whose cluster the node belongs. The information contained in these cluster leader lists help cluster leaders learn the identity of their neighboring leaders. For each cluster leader included in the cluster leader list, there is an associated field indicating whether the gateway can reach that cluster leader directly, or whether a joint gateway is needed. Hello messages are local broadcasts, i.e., they have a time to live (TTL) of one. These messages are never retransmitted by neighboring nodes.

When a node is first initialized, it has an undefined status level. To set its status level, it must determine whether it is within the bounds of any currently defined clusters. The node broadcasts a Hello message to its neighbors to announce its presence and to search for neighboring cluster leaders. It then sets a timer to wait for the reception of Hello messages. During this time, the node records the IP address of any cluster leaders from which it receives a Hello message. At the end of the initialization period, if the node has heard from exactly one cluster leader, it becomes an ordinary `node` and becomes a member of that cluster. If it has heard from more than one cluster leader, it becomes a `gateway` for the different clusters. On the other hand, if the node has not heard from any cluster leaders, then it becomes a leader (`cldr`) itself. The node sets its status level depending upon which of these conditions is satisfied. It then transmits a Hello indicating its new status, and reporting those cluster leaders with which it is able to communicate, if any.

In the event that all nodes initialize simultaneously, the nodes broadcast Hello messages looking for neighboring cluster leaders at approximately the same time. They then wait the discovery period for the reception of Hello messages. Because the nodes are initialized at the same time, many of these nodes may become cluster leaders, since there are not yet any cluster leaders in the surrounding area. In this situation, it is possible for nodes that are within direct communication range of one another to become cluster leaders. If this is the case, the cluster leader revocation algorithm, described in section 4.3, aids in the reduction of the number of cluster leaders.

4.2 Discovering the Topology

Nodes learn of the local hierarchical topology through the reception of Hello messages from neighboring nodes. A Hello message transmitted by a non-leader node contains a list of the cluster leader IP addresses to whose cluster the node belongs. For instance, in figure 1(b), a Hello transmitted by node G_1 would contain the IP address of leaders CL_1 and CL_2, since G_1 is a member of both of those leader's clusters. A Hello message transmitted by node N_2 would indicate that CL_2 is the only cluster leader with which N_2 can communicate directly.

A cluster leader that receives a Hello message from a non-leader node learns of a node that is a member of its cluster. The Hello message from that node indicates the other cluster leaders with which that node is able to communicate, if any. By receiving Hello messages from nodes that are in its cluster, a cluster leader learns the ID of its neighboring cluster leaders, as well as the ID of member nodes that can be used as gateways to reach those cluster leaders.

Non-cluster leaders also receive Hello message transmissions from their neighboring non-leaders. On reception of these messages, a non-leader learns with which cluster leaders its neighbor is able to directly communicate. The node can use that information to determine for which neighboring cluster leaders it can serve as a joint gateway. For instance, if the node is within the cluster of leader CL_1, and its neighboring node is within the cluster of leader CL_2, that pair of non-leaders can serve as joint gateways for cluster leaders CL_1 and CL_2, thereby establishing a communication path between the two clusters.

When a cluster leader receives a Hello message directly from a neighboring cluster leader, it learns that it is now within transmission range of that cluster leader. Two leaders that are within direct transmission range can utilize that link and eliminate the need for a gateway. In this scenario, the cluster leader then determines whether it should give up its leader status, according to the algorithm described in section 4.3.

4.3 Status Changes

When a non-leader node receives a Hello message from a cluster leader, it knows that it is within that leader's cluster. This is because the reception of a Hello message transmission from a cluster leader implies the two nodes are within direct communication range. If the node is a member of only one cluster, and hence has status `node`, and then it moves into the communication range of another cluster leader, the node changes its status level to `gateway` since it is now within the boundary of multiple clusters. The node can now be used as a gateway to route between the clusters. The opposite situation can also occur. A node that is a gateway for multiple clusters can eventually move so that it is only within direct transmission range of a single cluster leader. In this case, it changes it status level to `node`.

The most interesting status changes occur when either an ordinary node becomes a cluster leader, or when a cluster leader gives up its leader status to become an ordinary node. The following sections describe these occurrences in detail.

Becoming a Cluster Leader

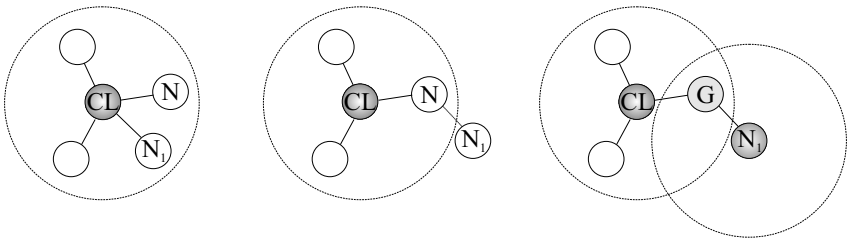


Figure 3: A Node Becomes a Cluster Leader.

Every node must either be a member of a cluster, or else be a cluster leader. Hence if a node is not within the transmission range of any cluster leaders, it should be a leader itself. When a node moves to the periphery of a network, it is possible that it will move out of direct transmission range of all other cluster leaders. When a node loses contact with all cluster leaders, it transmits a Hello message to look for other cluster leaders, to verify that it is not within the transmission range of any leaders. It then sets a timer to wait for the reception of a Hello message from a leader. If it receives such a message before the timeout, it joins that leader's cluster. Otherwise, if the node does not receive a Hello message from another leader before its timer expires, the node becomes a leader itself. In that case, it updates its status level and broadcasts a Hello message announcing this information.

Figure 3 illustrates an example of this process. In the figure, node N_1 wanders beyond the transmission range of its cluster leader. While it is still within transmission range of other non-leaders, it is no longer able to communicate directly with any leaders (figure 3(b)). After it broadcasts a Hello message to look for leaders and does not receive a response, the node becomes a leader itself, as indicated in figure 3(c).

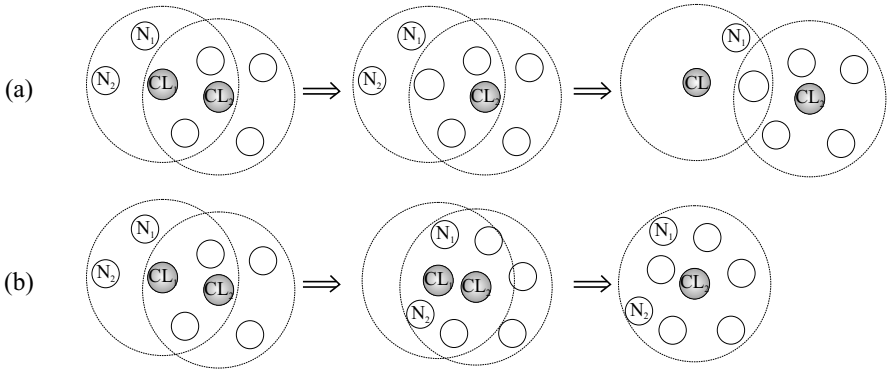


Figure 4: Cluster Merge.

Becoming an Ordinary Node

As nodes wander to the periphery of a network, is it likely that they will become cluster leaders because they will be out of transmission range of all other leaders. To prevent a continual growth in the number of cluster leaders, there must be a mechanism for cluster leaders to revoke their leader status and become non-leaders. Many clustering protocols [3, 8, 13] have the requirement that when two cluster leaders come within direct transmission range of one another, one leader must give up its leader status. The drawback with this approach, however, is that it can result in further leader changes within the network. An example is given in figure 4(a). In this figure, suppose leaders CL_1 and CL_2 come within transmission range of one another. The protocol for deciding which leader should revoke its leader status can be based on lowest IP address [8], a weighted algorithm [3], or other techniques. Suppose a least-ID algorithm is being used, whereby the leader with the lower ID gives up its status. In the example, CL_1 would then give up its leader status and join CL_2's cluster. The problem, though, is that nodes N_1 and N_2 are then left without a leader. In this case, one of them must now become a leader. This results in a *rippling effect*, whereby one leader change has resulted in additional changes within the network. Cluster leader changes are expensive due to the changes in routing paths which occur as result. In networks where routes are recorded between cluster leaders, leadership changes result in routing changes, and hence may generate routing overhead as well. The rippling effect, therefore, can have a detrimental performance impact on a network and should be avoided. Hence leader changes should be minimized, and when they occur, they should be isolated from further leadership changes.

The ARC protocol's mechanism for indicating when a cluster leader change should occur is based on the *subset* property. Instead of requiring a leadership change whenever two leaders come within transmission range of one another, the only time a leader change occurs is when one cluster becomes a subset of another cluster. By requiring the subset property to hold, further cluster leader changes are prevented. Formally, let C_1 be the set of nodes in CL_1's cluster, and let C_2 be the set of nodes in CL_2's cluster. Then,

if $C_1 \subset C_2$, i.e. $\forall n$, if $n \in C_1$, then $n \in C_2$,

a leadership change can occur. In the case $C_1 = C_2$, the leader with the lower ID becomes a non-leader, and the leader with the greater ID remains the overall leader. An example is given in figure 4(b).

In figure 4(b), cluster leaders CL_1 and CL_2 have come within transmission range of one another. However, because neither cluster is a subset of the other, both CL_1 and CL_2 remain leaders. As the nodes move, however, each of the nodes within CL_1's cluster becomes a member of CL_2's cluster as well. Because the cluster leaders are moving towards each other, the bounds of their clusters are also moving towards each other. Hence, one cluster is likely to become a subset of the other. Once this occurs, CL_1 can safely give up its leader status and join CL_2's cluster, and no further changes are necessary. Nodes N_1 and N_2 are a part of CL_2's cluster, and hence are not left without a leader. In the event that two cluster leaders only momentarily come

within transmission range of each other, a leadership change is unlikely to occur, and hence the overhead of the change is saved.

5 Multi-Level Hierarchical Clusters

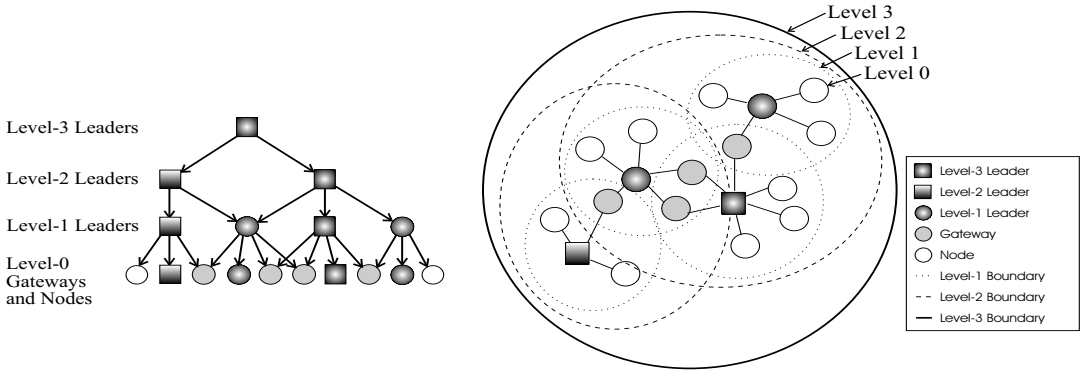


Figure 5: Logical Structure of a Cluster Hierarchy.

The clustering algorithm described in the previous section can be extended to a multi-level hierarchy such that the cluster hierarchy is able to dynamically adapt to the topology of the network. A one-level clustering system increases scalability and robustness of routes, but its benefit is limited when networks grow to thousands or tens of thousands of nodes. Creating a multi-level hierarchy allows the network to adapt to the number of nodes in the network, thereby further increasing the scaling ability of the system.

In a hierarchical system, each m_{th} level cluster is defined recursively as a set of $m - 1_{st}$ level clusters. Each m_{th} level cluster is also a node at level m in the tree representation. Further, each level-m cluster has associated with it a level-m leader. Any level-m leader is also a leader of each level-$(m-k)$ cluster of which it is a member, where $0 \leq k \leq m$. Figure 5 illustrates an example of a cluster hierarchy. Nodes in the figure are at level-0. Nodes are grouped into clusters, where each cluster has a cluster leader. Cluster leaders are at level-1. The cluster leaders are in turn placed into level-2 groups, each of which has its own group leader. Finally, the group leaders themselves are placed into a level-3 supergroup, and there is an overall supergroup leader of the entire network. A hierarchical classification scheme easily lends itself to a tree representation [17], as shown in figure 5. The tree diagram of the figure illustrates that a level-2 leader is also a level-1 cluster leader for its cluster, and is also a node at level-0.

Routes in a hierarchically clustered network can be recorded logically across the hierarchical tree. By recording the routes hierarchically, the robustness gain is greater than with a one-level clustering system. For instance, consider the network given in figure 6. Suppose node A wishes to send a message to node I. The physical path of the route, as determined by a flat routing protocol, could be

$$A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow F \rightarrow G \rightarrow H \rightarrow I.$$

However, routing the packet according to the hierarchical tree, A's packet is routed to its cluster leader, node B, which in turn sends the packet to its leader, node D. Node D then sends the packet to node H. Node H routes the packet down the hierarchical tree, and then delivers the packet to its destination I. Hence, the hierarchical route would be recorded as

$$A \rightarrow B \rightarrow D \rightarrow H \rightarrow I.$$

The following section presents a multi-level hierarchical clustering protocol called the Adaptive Routing using Cluster Hierarchies (ARCH) protocol. ARCH extends the clustering algorithm of ARC to build a

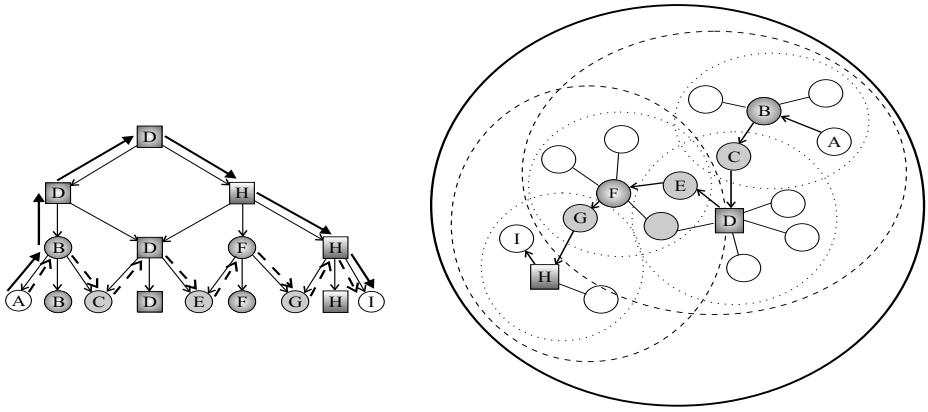


Figure 6: Hierarchical Routing.

multi-level hierarchy. ARCH creates the cluster hierarchy through the periodic exchange of *Hello* messages between neighboring nodes; no other message types are needed. The hierarchy built and maintained by ARCH is able to dynamically adapt to the changing conditions of the network. As the network becomes larger, hierarchical levels are added according to the algorithm given in section 5.3, so that an overall hierarchical leader is maintained. Conversely, as nodes leave the network, hierarchical levels are removed. Through these mechanisms, a hierarchy appropriate to the number of nodes and topology of the network is maintained. The hierarchical structure created by ARCH is evaluated in sections 6 and 7. For each hierarchical tree created, there is always a root of the tree which is the overall hierarchical leader for that network.

5.1 Status Levels

The status levels of the nodes and the leaders are monotonically increasing numbers. Nodes that do not have a status level are assigned the status value `undef`, which does not have an associated numeric value. Non-leader nodes (`gateway` and `node`) have status level zero. Level-1 leaders (`cldr`) have status level one, and level-2 leaders have status level two. In general, a level-m leader has status level m; the numerical value of the status level is equal to the hierarchical level of the leader.

5.2 Hierarchy Initialization and Discovery

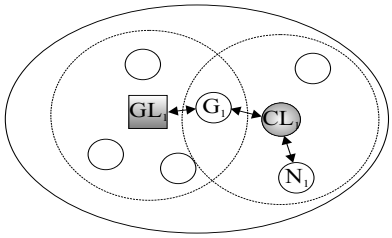


Figure 7: Example Hierarchical Topology.

Similar to the topology discovery procedure of ARC described in section 4.2, nodes learn the characteristics of their surrounding neighbors though the periodic exchange of Hello messages. The Hello messages contain the same information as the ARC Hello messages, including the list of cluster leaders associated with non-leader nodes. ARCH Hello messages also have additional fields. For each leader listed in the cluster leader list, the leader's status level is also included. A leader at status level-m is also a level-1 cluster leader, as indicated

in figure 5. As such, it is important to identify the highest hierarchical level m of each leader. ARCH Hello messages also contain a hierarchy list, whereby nodes include the IP address of all leaders in the hierarchy above them, within whose hierarchical cluster they reside. The hierarchy list is a linked list of structures that contains, for each entry, a leader's IP address, its status level, and the hopcount to that leader in *cluster hops*. This new metric is defined due to the varying number of physical hops that can be used to connect two clusters. Cluster leaders can be within direct transmission range of each other (a distance of one hop), can be connected through a gateway (two hops), or can be connected through a pair of joint gateways (three hops). To normalize these values, the *cluster hop* distance for each of these three cases is one.

Similar to the ARC protocol, the hierarchical structure of the network is created through the use of *Hello* messages. At initialization, a node's status level is undefined and it broadcasts a Hello message in search of neighboring cluster leaders so that it may join a cluster. If any cluster leaders are within the transmission range of the new node, they respond by transmitting a Hello message to that node. At the end of the discovery period, the node initializes its status level according to from how many cluster leaders it has heard. Simultaneous initializations are handled as described in section 4.1.

Nodes learn of their hierarchical leaders through the propagation of hierarchy information in Hello messages. An example is given in figure 7. When GL_1 transmits a Hello, the status level information in the Hello indicates that GL_1 is a level-2 leader. When node G_1, a gateway node, transmits its Hello message, it indicates that nodes GL_1 and CL_1 are its cluster leaders. Included with this information is that fact that GL_1's status level is 2, while that of CL_1 is 1. In this way, when CL_1 receives the Hello message from G_1, it learns that its neighboring leader is a level-2 leader, and that this node is its hierarchical parent. When it sends its own Hello message, CL_1 includes node GL_1 in its hierarchy list, and indicates that the status level of GL_1 is level-2. When node N_1 receives this Hello message, it now knows that it also has a level-2 leader, namely GL_1. Thereafter, when N_1 transmits its own Hello message, it will include CL_1 in its cluster leader list and GL_1 in its hierarchy list. The method by which the hierarchical structure is built is described in the following section.

5.3 Building and Maintaining the Hierarchy

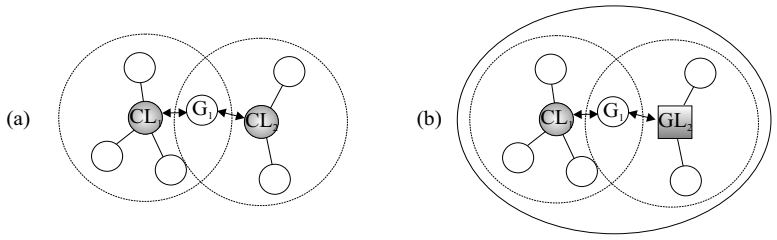


Figure 8: Creation of Cluster Hierarchy.

The information contained in the Hello messages is used to build the hierarchy. Cluster leaders are initially established through the same mechanism as in ARC, as described in section 5.2. Cluster leaders are level-1 leaders. Once there exists two neighboring cluster leaders, one of these leaders must become a level-2 leader, so that there is an overall leader. Suppose the scenario in figure 8(a) exists, where nodes CL_1 and CL_2 have just become cluster leaders. Node G_1 eventually receives Hello messages from each of these leaders, and hence becomes a gateway for the two clusters. When it broadcasts its own Hello message, it indicates in its cluster leader list that both nodes CL_1 and CL_2 are its cluster leaders, and that both nodes are have status level 1. After receiving this message, CL_1 and CL_2 now know that they each have a neighboring cluster leader, and so one of them must increase its hierarchical status level to become the overall leader. This could be accomplished in any number of ways, i.e. a weight-based algorithm, the greater number of cluster members,

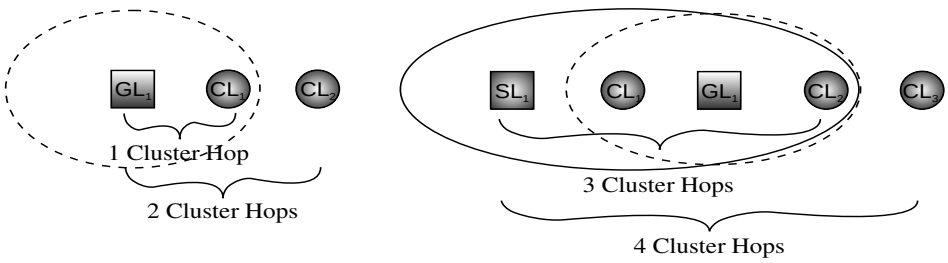


Figure 9: Maximum Distance Between Hierarchical Levels.

etc. The approach ARCH uses is an ID-based algorithm. The leader with the greater ID (i.e., IP address) becomes the overall leader. Hence, in the scenario in figure 8(a), node CL_2 increases its status level and becomes the overall leader, as indicated in figure 8(b).

In general, a cluster leader determines whether it should increment its status level each time it a Hello message from one of its member nodes. The leader examines the hierarchical information contained in the hierarchy list of the Hello message. Each of the following conditions must hold true for a leader to increment its status level:

(i) The leader does not already have a hierarchical parent.
(ii) The leader is the same status level as another leader in either the hierarchy list or the cluster leader list.
(iii) The leader's IP address is greater than that of the other leader with the same status level.

If each of these conditions holds, the leader increments its status level by one. Otherwise it remains at the same level.

The leader can also use the hierarchy list to determine whether it has any parents of which it is not already aware. It traverses the list, examining the status level and hopcount values for each entry. For a leader in this list to be a parent of the leader receiving the Hello, the following formula must hold true:

$$2^{(sl-1)} - 1 \geq hc$$

where sl is the status level of the parent node, and hc is the distance, in cluster hops, of the leader from the possible parent. $2^{(sl-1)} - 1$ represents a maximum bound on the distance from a leader to one of its children.

Figure 9 illustrates an example of determining hierarchical parents. In the three node network, CL_1 is only one cluster hop away from GL_1, and so it can be a member of that node's group. However, CL_2 is two cluster hops away from GL_1, and so it is too far to be a part of that group. Hence, it needs to become a level-2 leader itself. Similarly, the five-node network is an example of a level-3 group (supergroup). Leaders CL_1, GL_1, and CL_2 are within three cluster hops from SL_1. The status level of SL_1 is three, and since $2^{(3-1)} - 1 \geq 3$, each of those nodes are a part of SL_1's supergroup. However, CL_3 is four cluster hops away, and so it is too far to be a member of that supergroup. Hence, SL_1 is not a hierarchical parent of CL_3.

After the hierarchy has been built, there must also be a mechanism for decreasing the status level of nodes so that the hierarchical tree does not grow without bound. A leader decrements its status level in one of two situations: (i) when its cluster becomes a subset of another cluster, or (ii) when it determines that it no longer needs to be at as high a level.

When two leaders come within direct transmission range of each other, it is possible that one of them will give up its leader status. The only time a leader gives up its leadership is when its cluster becomes a subset of another cluster, as described for the ARC protocol in Section 4.3, *and* its status level is less than or equal to that of the other leader's. If both these conditions do not hold, then the leader does not give up its leader status. Formally, let CL_1 and GL_2 be the two leaders which have come into transmission range, and let sl_1

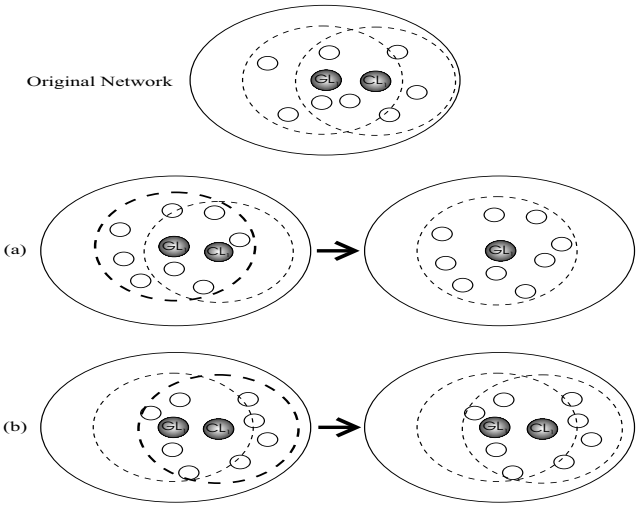


Figure 10: Possibilities for Decreasing Hierarchical Levels.

and sl_2 be their respective status levels. Also, let C_1 be the set of nodes in CL_1's cluster, and let G_2 be the set of nodes in GL_2's cluster. Then, CL_1 only gives up its leader status if the following two conditions hold:

(i) $C_1 \subseteq G_2$, i.e. $\forall n$, if $n \in C_1$, then $n \in G_2$, and

(ii) $sl_1 \leq sl_2$.

Figure 10 illustrates two possible scenarios for a given network configuration. In the original network, CL_1 and GL_1 come within transmission range of each other. However, each still has members of its cluster that are not in the other leader's cluster. As the nodes move, one possible scenario is that all the nodes in CL_1's cluster will also be a part of GL_1's cluster. In this case, since the status level of CL_1 is one, while that of GL_1 is two, CL_1 can give up its leader status. This scenario is shown in Figure 10(a). The other possibility is that all the nodes in GL_1's cluster move within the boundaries of CL_1's cluster. The subset property then holds, but because GL_1's status level is greater than that of CL_1, it does not reduce its hierarchical level. Figure 10(b) illustrates this scenario.

A leader may also decrement its status level whenever its status level is greater than `cldr`, and it determines that it no longer needs to be at that high a level. A leader performs this check whenever it receives a Hello from a non-leader node. The leader at level-m must determine whether it has any neighboring leaders at status level-$(m-1)$. As it checks the status level of each of its neighboring leaders, if it does not find a neighbor at a status level $m-1$, or if it does find such a neighbor and

$$2^{(sl-1)} - 2 < hc$$

where sl is the status level of the neighboring leader and hc is the distance, in cluster hops, between the two leaders, then the leader must decrement its status level by one, to $m-1$. Otherwise, it retains its current status level. If the distance of the leader at the lower status level satisfies the above formula, then that leader is likely to have another leader at a higher level, and so this node does not need to retain its status. However, if the neighboring leader is closer than $2^{(sl-1)} - 2$, then this node must continue to be its hierarchical leader.

6 Performance Evaluation

To evaluate the performance of the ARC and ARCH protocols, three sets of simulations have been performed. The first set of simulations is used to evaluate the hierarchical topology that each of the protocols creates. The

Number of Nodes	50	100	250	500	750
Room Size	1000m×1000m	1500m×1500m	2400m×2400m	3450m×3450m	4300m×4300m

Table 1: Room Sizes for Network Simulations

general characteristics of the hierarchies are evaluated by examining the number of nodes at each hierarchical level, as well as by the determination of the inter-cluster connectivity pattern. For instance, the mean number of gateways for a neighboring cluster leader provides insight into the robustness of the route. Similarly, the number of reachable cluster leaders per non-leader indicates the utility of each gateway node.

In the second set of simulations, the performance of the subset algorithm for cluster leaders to revoke their leader status is evaluated. The algorithm is compared against two popular alternatives for leader revocation. The first is the least ID algorithm. In this approach, when two leaders come within transmission range of each other, one of the leaders must give up its leader status. The leader to give up its leader status is selected based on IDs. The leader with the least ID becomes an ordinary node, while the leader with the greater ID remains a leader. This is the Least Cluster Change (LCC) algorithm implemented by the CGSR protocol [8]. The other approach is to use a weight-based metric. With this algorithm, when two leaders come within direct transmission range of each other, a weight-based metric is used to determine which leader retains its leader status. The leader with the greatest weight value remains a leader, while the leader with the lower weight value becomes an ordinary node. This is the metric utilized by the DMAC protocol [3]. In [3], the basis for the weight metric is not specified. A different work [5], however, investigates a variety of weight metrics and their impact on cluster stability. For the simulations presented here, the number of cluster members is used as the metric. The weight for each leader is assigned to be the number of members of the cluster. The leader with the most cluster members remains a leader whenever two leaders come within transmission range. In the case of a tie, the least ID algorithm is used to select the remaining leader. By examining the number of cluster leader and status level changes during the lifetime of the network, the overall stability of the cluster topology can be evaluated. Each of set of experiments was performed for both the ARC and ARCH protocols.

The third set of simulations investigates the performance benefits of clustering for scaling in ad hoc networks. In these experiments, the performance of ARC with an ad hoc routing protocol is compared against the performance of that ad hoc routing protocol by itself. The mechanism for the integration of the two protocols is explained in section 6.3.

Each of the experiments was performed using the GloMoSim network simulator [1] developed at UCLA. The IEEE 802.11 MAC layer protocol [9] is used for channel access in each simulation. The mobility model used is the random direction model [28]. With this model, nodes are randomly distributed throughout the simulation area at the start of the simulation. Each node then selects a degree in which to travel from a uniform distribution between 0 and 360. After selecting the direction for travel, the node begins moving in this direction at the specified speed. Once the node reaches the boundary of the simulation area, it rests for 30 seconds. It then selects a new direction, this time between 0 and 180 degrees, relative to the boundary wall on which the node is located. The node then resumes travel in this new direction.

In the first set of simulations, the number of nodes is varied between 50 and 750. The room size corresponding to each number of nodes is given in table 1. As more nodes are added into the simulation, the room size is increased so that the overall network density is kept approximately constant. This allows the evaluation of the protocols in networks of increasing size. For each of the simulations, the transmission range of each node is 250m, and the average number of neighbors per node is six [16]. Each simulation models 300 seconds of real time, and each time point represents an average of ten simulation runs.

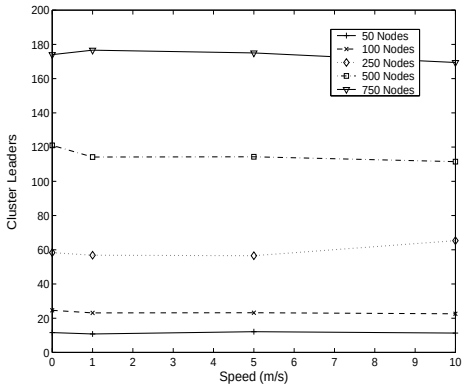


Figure 11: Total Cluster Leaders.

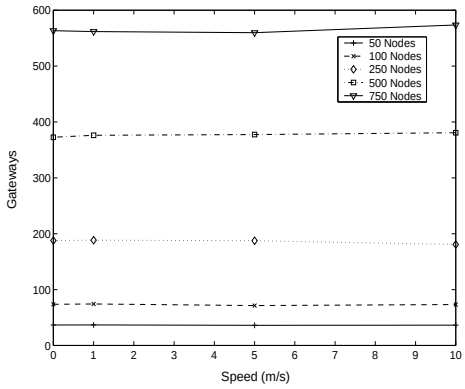


Figure 12: Total Gateways.

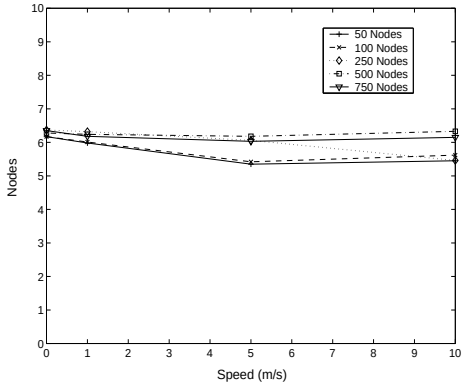


Figure 13: Nodes per Cluster.

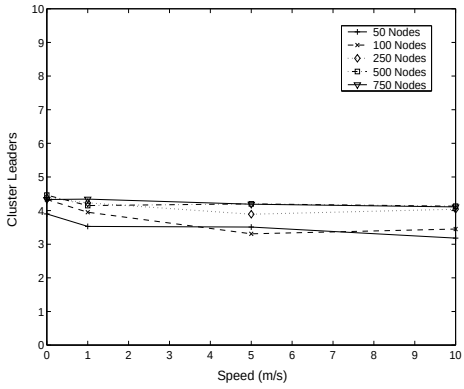


Figure 14: Reachable Cluster Leaders per Node.

6.1 ARC Results

Figures 11 and 12 show the average number of cluster leaders and gateways, respectively, at the simulation end for each of the simulated scenarios. The graphs show that the number of nodes at each status level is not particularly affected by the mobility of the nodes. Given that the node density is held constant, the primary determinant of the number of nodes at each status level is the number of nodes in the network. For the given network density, the number of cluster leaders in each simulation is consistently approximately 25% of the number of nodes in the network. The number of cluster leaders in the networks is directly related to the transmission radius of the network nodes. If the nodes had a larger transmission radius, there would be fewer cluster leaders, more nodes per cluster, and more gateways between clusters. The reverse is true for a smaller transmission radius.

One other observation to note from the figures is the number of gateways in the network relative to the number of ordinary nodes. The ordinary nodes (i.e., nodes that are neither leaders nor gateways) typically comprised less than 2% of the network nodes. The vast majority of the non-leaders is able to communicate with two or more cluster leaders. This leads to high inter-cluster connectivity and increased route robustness, since there are multiple gateways between each cluster.

Figure 13 illustrates the average number of nodes per cluster in the different simulations. The figure shows that for a 250m transmission radius, the average number of nodes per cluster remains nearly constant at six, regardless of the number of nodes in the network or mobility speed of the nodes.

The total number of reachable cluster leaders per non-leader is shown in figure 14. A reachable cluster

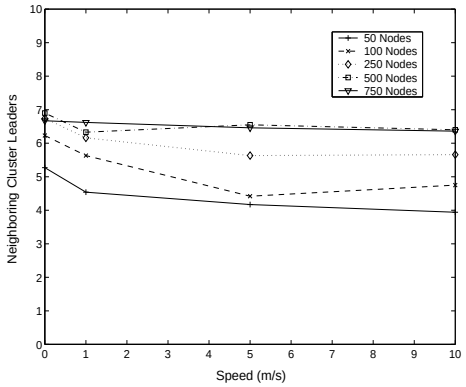


Figure 15: Neighbors per Cluster Leader.

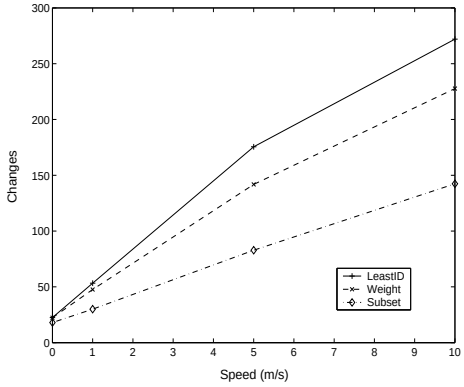


Figure 16: Total Node-Leader and Leader-Node Changes.

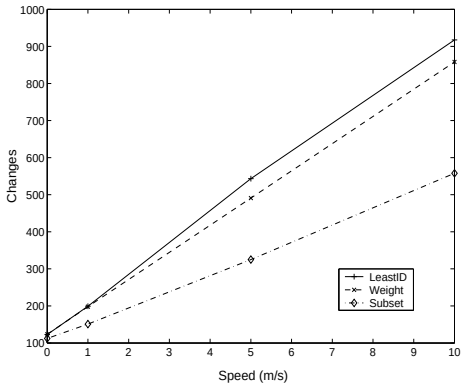


Figure 17: Total Status Level Changes.

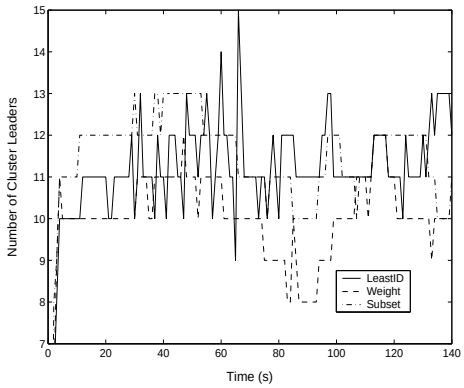


Figure 18: Number of Cluster Leaders during Simulation.

leader is a leader with which a gateway or ordinary node can communicate, either directly or through a joint gateway. The figure shows that nodes are able to communicate with, on average, four cluster leaders. Of these four cluster leaders, two of the leaders are typically reachable through direct transmission, while the other two are reachable through the use of a joint gateway.

While not shown, there is an average of three to four gateways between clusters. With this average, the robustness of routes is greatly increased, because if one gateway moves out of range, there are likely to be many others capable of maintaining the connection. The number of gateways is dependent upon the transmission range and network density, and so the degree of robustness similarly depends upon the overall network density. This is further examined in section 6.3.

Figure 15 illustrates the high inter-cluster connectivity levels achieved by the protocol. The figure indicates the average number of cluster leaders that are reachable by a given cluster leader through its gateways. The number of neighboring cluster leaders remains relatively constant over the varying mobilities. This number increases slightly for the increasing network sizes, due to the larger number of cluster leaders in these networks. The figure indicates a high connectivity level for all networks, with the average cluster leader being able to reach between four and seven neighboring cluster leaders through its gateways.

Figures 16 through 18 represent results from the second set of experiments, in which the effectiveness of the subset algorithm for leader revocation is evaluated. For this set of experiments, 50 nodes are modeled in a room of size 1000m$\times$1000m. The transmission range of each node is 250m. All the other parameters are the

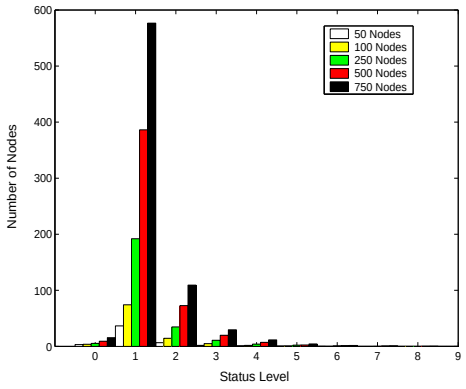


Figure 19: Nodes at Each Status Level.

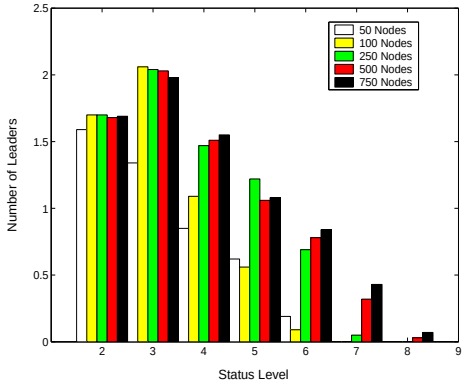


Figure 20: Leaders per Node at Each Status Level.

same as the previous set of simulations.

The number of cluster-leader-to-node changes plus the node-to-cluster-leader changes is given in figure 16. As the average node speed increases, the number of such changes rises. At higher node speeds, nodes change neighbors more rapidly, and so the probability that two cluster leaders come within transmission range of each other increases. The figure shows that the LeastID algorithm results in the greatest number of cluster leader changes during the simulation. The algorithm is followed by the Weight approach. The Subset algorithm yields the fewest number of cluster leader changes. This is primarily due to the more strict requirement for a node to give up its leader status. The LeastID approach results in more than twice as many cluster leader changes as the Subset algorithm throughout the simulations.

Figure 17 represents the total number of status changes in the simulations. This includes gateway-to-node and node-to-gateway status changes. This figure is consistent with the previous results, in that the LeastID and Weight algorithms produce the greatest number of status level changes, while the Subset algorithm produces the fewest.

The previous three graphs alone do not give an indication of the relative stabilities of the three algorithms. The stability of the cluster topologies is more closely reflected by figure 18. This figure represents the number of cluster leaders at each second, during the first 140 seconds of the simulations. The figure indicates that the Subset algorithm results in a fairly stable number of cluster leaders that fluctuates between 10 and 12 cluster leaders, after initially fluctuating to 13 cluster leaders. The LeastID and Weight algorithms, on the other hand, show much wider variance. The LeastID approach fluctuates rapidly between 9 and 15 cluster leaders, while the Weight algorithm oscillates between 8 and 12 cluster leaders. In particular, the LeastID approach varies between 9 and 15 cluster leaders on a period of only one second. Such a high number of cluster leader changes results in high overhead in terms of routing updates and cluster reconfigurations, and leads to an unstable hierarchical topology.

6.2 ARCH Results

Figures 19 and 20 represent the hierarchical topology created by the ARCH protocol. In figure 19, the number of nodes at each status level, for networks ranging from 50 to 750 nodes, is shown. For the purposes of the figures, status level zero corresponds to an ordinary node, while level one indicates a gateway. Cluster leaders have status level two; the hierarchical levels build upwards from level two. The figure indicates that the majority of nodes in each network are gateways, concurring with the results shown for ARC in section 6.1. Few nodes are ordinary nodes, only able to communicate with a single cluster leader. The number of leaders is greatest at status level two, the cluster leaders. The number of leaders in the levels above this level gradually

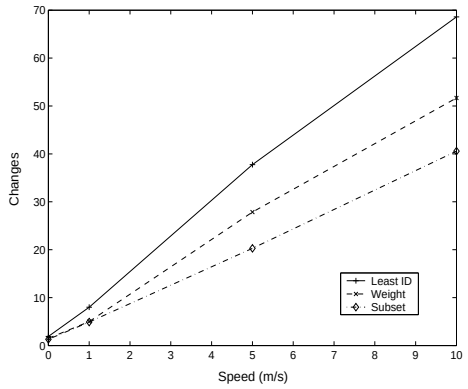


Figure 21: Total Cluster Leader to Node Changes.

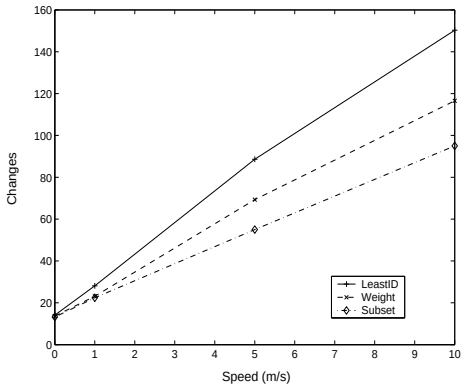


Figure 22: Total Node-Leader and Leader-Node Changes.

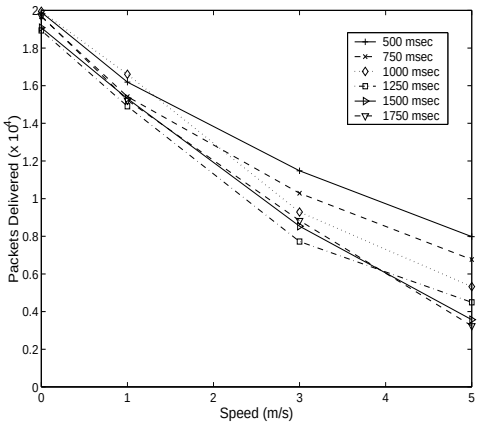


Figure 23: Number of Packets Delivered for Varying Hello Frequency.

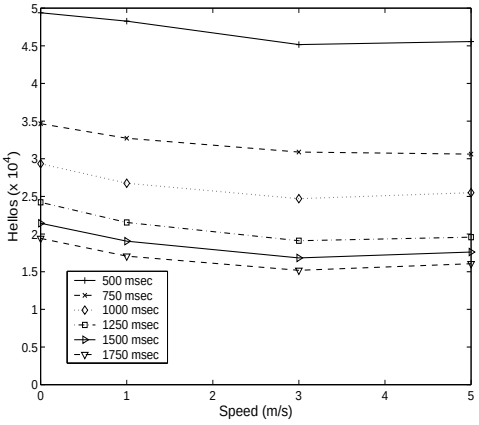


Figure 24: Number of Hellos Transmitted for Varying Hello Frequency.

tapers off. The largest networks have leaders at the highest levels, as would be expected. Larger networks have more clusters, and hence require greater hierarchical levels in order to create the hierarchy.

Figure 20 represents the number of leaders per node at each hierarchical level. Because the cluster leaders at level two are at the lowest leader level, the data begins at this level. The values are calculated by summing the number of leaders at a particular level for each of the nodes at lower hierarchical levels. For instance, the number of leaders per node at level three is found by adding the number of level three leaders for the ordinary nodes, gateways, and level two cluster leaders. The hierarchy characteristics are further evaluated in section 7.

The effectiveness of the subset algorithm for maintaining cluster stability is evaluated in figures 21 and 22. The graphs for ARCH follow the same trend as the respective graphs for ARC, given in section 6.1. In each of the figures, the LeastID algorithm results in the greatest number of cluster leader changes. The Weight approach is next, followed by the Subset algorithm with the fewest number of status level changes.

6.3 ARC with AODV

The utility of a hierarchical protocol is best exemplified by the scalability gains achieved when compared with a flat routing protocol. The ARC and ARCH protocols can be run underneath virtually any on-demand routing

protocol for ad hoc networks. The protocols serve as an interface between the routing protocol and the IP layer. In this section, the ARC protocol is combined with the Ad hoc On-Demand Distance Vector (AODV) routing protocol [26] to demonstrate the performance gains achieved over the AODV protocol operation without the ARC protocol. AODV is a reactive routing protocol that discovers and maintains routes only when those routes are needed by a source node. Route finding is accomplished through a route request/route reply discovery cycle. Further details on the AODV protocol can be found in [24, 26].

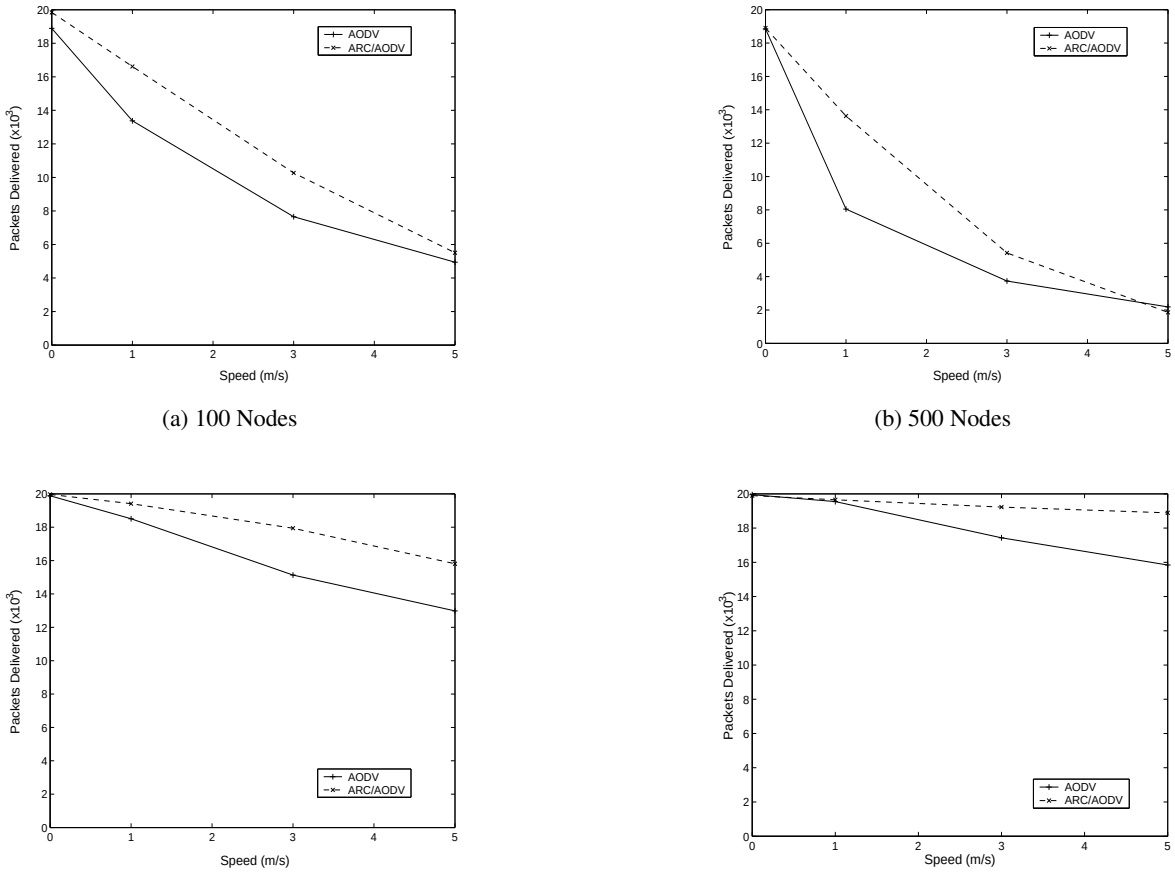
A cluster leader serves as a representative for the nodes in its cluster and processes routing packets on the node's behalf. To this end, the ARC protocol intercepts all AODV control packets before they reach the AODV protocol. If the node is a cluster leader, the packet is passed up to the AODV protocol for processing. Otherwise, the packet is either rebroadcast by the non-leader if it is a broadcast packet, or it is unicast to the next hop along the path to its destination.

Because non-cluster leaders do not process AODV control packets, they do not learn of routes through route requests and replies. It is the responsibility of the cluster leader to inform gateway nodes of routes of which they should be aware. This is accomplished through a packet type called the *Route Activation* packet (RTAct). When a cluster leader requires a gateway to serve as a forwarding node to a neighboring leader along some route, the cluster leader sends that gateway a RTAct packet containing the destination address, as well as the address of the next-hop cluster leader towards this destination. With this information, when a gateway receives data packets for that destination, it knows to which neighboring cluster leader those data packets should be forwarded. When that gateway moves out of range of either cluster leader, the upstream cluster leader can simply select a new gateway between the two cluster leaders, and transmit to it an RTAct packet. In this way, the route can be continually locally repaired, and the source node does not need to reinitiate route discovery to repair the route. Due to space limitations, further explanation of how AODV can work with the ARC protocol has been omitted from this paper. However, a more thorough explanation can be found in [4].

To demonstrate the interoperation of ARC and AODV, two sets of simulations have been performed. The first evaluates the ARC/AODV combination in networks of increasing number of nodes and network size, so that the effect of the increased path length between sources and destinations can be evaluated. The average network density in these simulations is held relatively constant at six neighbors per node. In the 100 node network, a room size of 1500m$\times$1500m is used, while in the 500 node network, a room size of 3450m$\times$3450m is used. The transmission range for all nodes is 250m. In the second set of simulations, the simulation area is held constant while the transmission range of the nodes is modified. These simulations evaluate the effect of varying density on the protocols. In these simulations, the number of nodes is 100 and the simulation area is 1500m$\times$1500m. The transmission range of the nodes is varied from 350m to 450m. This varies the network density from 13 to 20 neighbors per node. A transmission range of 250m in the same size network is used in the previous experiment.

Each simulation data set was run for ten different initial configurations. The results of these simulations were averaged together to produce the resulting graphs. The data channel is 2 MB/s, and data packets are 64 bytes. There are 20 randomly selected source/destination pairs, and each source transmits four packets per second. Each simulation models 300 seconds of real time.

The sending frequency of hello messages represents a tradeoff between bandwidth utilization and power consumption, versus the most up-to-date routing information. Figure 23 represents the number of packets able to be delivered by ARC running over AODV in a network of 100 nodes. The hello frequency of cluster leaders is held constant at one hello per second, while that of the non-cluster leaders is varied. The lines in the graph indicate the different hello sending frequencies of the non-cluster leaders. For example, the line labeled “500 msec” indicates a hello message sending frequency of one hello every 500 msec for non-leader nodes. As was shown in figure 11, cluster leaders comprise approximately 25% of the network nodes. Figure 23 indicates that the difference in the number of delivered data packets increases as the rate of mobility increases. For slow mobilities, a sending interval of 1000 msec produces the greatest number of delivered packets, while for faster mobility the highest frequency produces the best results. The drawback to the increased sending rate,



(a) 100 Nodes

(b) 500 Nodes

(c) 350m Range

(d) 450m Range

Figure 25: Number of Packets Delivered.

however, is shown in figure 24 as the total number of hello messages transmitted by all network nodes during the simulation. To balance the number of delivered data packets with the number of hello messages produced, a sending interval of 1000 msec is used for the remainder of the simulations.

The number of data packets delivered to their destinations in the 100 and 500 node networks is shown in figure 25(a) and (b). The graphs show that ARC is able to deliver more data packets to the destinations than AODV by itself. A performance improvement of 80% is achieved in some scenarios. The performance differential is particularly great for the moderate mobility speed (1-3 m/s), and then decreases as the speed of the nodes increases. This is due to the frequency at which Hello messages are transmitted. As nodes move faster, topology information becomes stale more rapidly. If the Hello sending rate is increased, the local topology information maintained by each node is more accurate, and hence cluster leaders have more up-to-date information about the gateways to the other clusters. This results in a greater performance differential between the AODV and ARC/AODV combination, particularly at the higher mobilities. The expense, however, is greater overhead.

Figure 25(c) and (d) represents the data packets delivered in the 100 node networks with varying density. The trend witnessed in these graphs is the opposite of that in the previous graphs. As the density of the network increases, more nodes lie within a given cluster. Cluster leaders have more gateways per cluster, and therefore greater routing flexibility. Hence, routes need to be repaired less frequently. As the nodes move faster, the

Number of Nodes (N)	Theoretical Optimum ($h = lnN$)	Leaders at Level $\lfloor h \rfloor$	Leaders at Level $\lceil h \rceil$
50	3.91	1.8	1.0
100	4.61	1.8	0.8
250	5.52	2.2	0.9
500	6.21	1.3	0.8
750	6.62	1.7	1.1

Table 2: Average Maximum Hierarchical Level.

links break more rapidly and more packets are dropped by AODV. With ARC/AODV, however, there are increasingly more routing options between any two clusters, and so routes rarely need to be repaired; fewer packets are dropped and more packets can be delivered to their destinations.

7 Observations

The hierarchical structure obtained by ARCH can be compared to the theoretical optimum derived by Kleinrock and Kamoun in their 1977 paper on hierarchical routing [15]. In this paper, the authors investigated a hierarchical routing scheme with the goal of reducing routing table size. The authors determined that the length of the routing table is a strict function of the clustering structure. It was determined that the number of entries in a node's forwarding table is minimized when the number of level-i clusters in each level-$(i+1)$ cluster is e, and the number of levels in the clustering hierarchy equals $\ln N$. In this case, the forwarding table contains $e \ln N$ entries.

Table 2 indicates the depth of the hierarchy created by ARCH. Kleinrock's theoretical optimum h for each network size is given in the second column of the table. The third and fourth columns indicate the average number of leaders at level-$\lfloor h \rfloor$ and level-$\lceil h \rceil$, respectively. The table shows that, on average, there is one leader per network at level-$\lceil h \rceil$. In the majority of simulations, this is the greatest hierarchical level in each of those networks. ARCH conforms to the maximum hierarchical depths proven to be theoretical optimum. As such, the protocol lends itself well to hierarchical addressing structures. When used with hierarchical addressing, it will be extremely beneficial for reducing routing table sizes. Further discussion on hierarchical addressing is beyond the scope of this paper.

The simulation results show that the subset algorithm performs better than its predecessors in reducing the number of status level changes during the network lifetime. As shown in figure 18, the subset algorithm creates a hierarchical topology which is more stable during node movement. The reduction in the number of status level changes does not result in an unmitigated growth in cluster leaders. On the contrary, the rippling effect of leadership changes is eliminated, and the number of cluster leaders shows significantly less fluctuation than in the least ID and weight-based approaches. The reduction in leadership changes results in fewer routing path changes, which translates into less routing control overhead.

The integration of ARC with AODV demonstrates the improvement in data packet delivery that can be achieved through hierarchical routing. The performance improvement is due to the increase in route robustness gained by having multiple routing options between clusters. Previous results [4] have shown that the primary benefit of ARC is the increase in data packet delivery. The drawback, however, is the increase in control overhead generated by the periodic messaging requirement. It would be beneficial to investigate clustering strategies that eliminate this requirement, or else that can operate with a much longer hello interval. Adaptive clustering, in which nodes can modify the frequency of the hello messages to the rate of movement of surrounding neighbors, would allow a reduction in overhead during periods of little movement, while at the

same time providing increased accuracy of neighborhood information during periods of rapid mobility [11].

One additional disadvantage of the clustering protocols is that the centralization of routes through cluster leaders may unfairly tax the battery power of those nodes, resulting in reduced battery lifetimes. A more distributed approach to routing would be beneficial to the protocol, so that the cluster leaders could help establish paths, but then let the member nodes within the cluster do the actual forwarding of the data packets. In her chapter on cluster-based networks, Steenstrup discusses some methods for distributed routing in these networks [32].

The integration of the ARCH protocol with AODV is an area of future work. It is expected that performance improvements will continue to be achieved as the network size is further increased.

8 Conclusion

Hierarchical routing increases the robustness of routes by allowing greater flexibility in routing decisions between nodes on a given path. The increase in robustness directly translates into longer-lived routes, resulting in fewer route repairs and higher data delivery rates. Two hierarchical routing protocols have been presented. The Adaptive Routing using Clusters (ARC) protocol creates a one-level cluster topology on a network of nodes, while the Adaptive Routing using Cluster Hierarchies (ARCH) protocol creates a multi-level hierarchy that dynamically adjusts its depth in response to the topology of the network. The strengths of the two protocols lie in their hierarchical routing structure for improving the scaling potential of ad hoc routing protocols, their avoidance of the rippling effect during cluster leader changes, and their increase in data packet delivery for both large networks and networks with rapidly moving nodes. Through simulation, it has been shown that the hierarchical topologies created by the two protocols are more stable than in other approaches. Further, when combined with a flat routing protocol, the data packet delivery is improved by as much as 80%.

9 Acknowledgments

Although this work has a single author, the work was partially completed under the direction of the author's doctoral advisor, P. Michael Melliar-Smith. This work was supported in part by a UC Core grant with Cubic Defense Systems.

References

[1] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, and M. Gerla. GlomoSim: A Scalable Network Simulation Environment. Technical Report CSD Technical Report, #990027, UCLA, 1997.

[2] Dennis J. Baker and Anthony Ephremides. The Architectural Organization of a Mobile Radio Network via a Distributed Algorithm. *IEEE Transactions on Communications*, 29(11):1694–1701, November 1981.

[3] Stefano Basagni. Distributed Clustering for Ad Hoc Networks. In *Proceedings of the 1999 International Symposium on Parallel Architectures, Algorithms, and Networks*, pages 310–315, Australia, June 1999.

[4] Elizabeth M. Belding-Royer. Hierarchical Routing in Ad hoc Mobile Networks. *To appear in the Wireless Communications and Mobile Computing*, 2002.

[5] Christian Bettstetter and Roland Krausser. Scenario-Based Stability Analysis of the Distributed Mobility-Adaptive Clustering (DMAC) Algorithm. In *Proceedings of the* 2_{nd} *Annual Symposium on Mobile Ad hoc Networking and Computing*, Long Beach, California, October 2001.

[6] Linda Briesemeister and Gunter Hommel. Role-Based Multicast in Highly Mobile but Sparsely Connected Ad hoc Networks. In *Proceedings of the* 1^{st} *Annual Workshop on Mobile and Ad hoc Networking and Computer (MobiHOC)*, pages 45–50, Boston, MA, August 2000.

[7] Josh Broch, David A. Maltz, David Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the* 4^{th} *Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 85–97, Dallas, Texas, October 1998.

[8] Ching-Chuan Chiang, Hsiao-Kuang Wu, Winston Liu, and Mario Gerla. Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel. In *Proceedings of IEEE Singapore International Conference on Networks (SICON)*, pages 197–211, April 1997.

[9] IEEE Standards Department. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Standard 802.11–1997*, 1994.

[10] J.J. Garcia-Luna-Aceves and M. Spohn. Source-Tree Routing in Wireless Networks. In *Proceedings of the* 7^{th} *International Conference on Network Protocols (ICNP)*, Toronto, Canada, November 1999.

[11] M. Gerla, T.J. Kwon, and G. Pei. On Demand Routing in Large Ad hoc Wireless Networks with Passive Clustering. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, Spetember 2000.

[12] Mario Gerla and Jack Tzu-Chieh Tsai. Multicluster, Mobile, Multimedia Radio Network. *Wireless Networks*, 1(3):255–265, March 1995.

[13] Mingliang Jiang, Jinyang Li, and Yong Chiang Tay. Cluster Based Routing Protocol (CBRP) Functional Specification. *IETF Internet Draft, draft-ietf-manet-cbrp-spec-00.txt*, August 1998. (Work in Progress).

[14] David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Tomasz Imielinski and Hank Korth, editors, *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[15] Leonard Kleinrock and F. Kamoun. Hierarchical Routing for Large Networks. *Computer Networks*, 1(1):155–174, 1977.

[16] Leonard Kleinrock and F. Kamoun. Optimal Clustering Structures for Hierarchical Topological Design of Large Computer Networks. *Networks*, 10(3):221–248, 1980.

[17] D. E. Knuth. *The Art of Computer Programming*, volume 1. Addison-Wesley, 1969.

[18] Young-Bae Ko and Nitin H. Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. In *Proceedings of the* 4^{th} *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 66–75, Dallas, Texas, October 1998.

[19] Sung-Ju Lee, Elizabeth M. Royer, and Charles E. Perkins. Ad hoc Routing Protocol Scalability. To appear in the International Journal on Network Management, 2002.

[20] Chunhung Richard Lin and Mario Gerla. Adaptive Clustering for Mobile Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 15(7):1265–1275, September 1997.

[21] Shree Murthy and J. J. Garcia-Luna-Aceves. An Efficient Routing Protocol for Wireless Networks. *ACM/Baltzer Mobile Networks and Applications*, 1(2):183–197, October 1996.

[22] Vincent D. Park and M. Scott Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 1405–1413, Kobe, Japan, April 1997.

[23] Guangyu Pei, Mario Gerla, Xiaoyan Hong, and Ching-Chuan Chiang. A Wireless Hierarchical Routing Protocol with Group Mobility. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1538–1542, New Orleans, LA, September 1999.

[24] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir R. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. *IETF Internet Draft, draft-ietf-manet-aodv-10.txt*, March 2002. (Work in Progress).

[25] Charles E. Perkins and Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. *SIGCOMM '94: Computer Communications Review*, 24(4):234–244, October 1994.

[26] Charles E. Perkins and Elizabeth M. Royer. The Ad hoc On-Demand Distance Vector Protocol. In Charles E. Perkins, editor, *Ad hoc Networking*, pages 173–219. Addison-Wesley, 2000.

[27] Ram Ramanathan and Martha Steenstrup. Hierarchically-organized, Multihop Mobile Wireless Networks for Quality-of-Service Support. *ACM/Baltzer Mobile Networks and Applications*, 3(1):101–118, 1998.

[28] Elizabeth M. Royer, P. Michael Melliar-Smith, and Louise E. Moser. An Analysis of the Optimum Node Density for Ad hoc Mobile Networks. In *Proceedings of the IEEE International Conference on Communications*, Helsinki, Finland, June 2001.

[29] Elizabeth M. Royer and C.-K. Toh. A Review of Current Routing Protocols for Ad-Hoc Mobile Networks. *IEEE Personal Communications*, 6(2):46–55, April 1999.

[30] N. Shacham. Organization of Dynamic Radio Networks by Overlapping Clusters: Architecture Considerations and Optimization. In *Performance '84*, pages 435–447, December 1984.

[31] Prasun Sinha, Raghupathy Sivakumar, and Vaduvur Bharghavan. CEDAR: a Core-Extraction Distributed Ad hoc Routing algorithm. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, pages 202–209, New York, NY, March 1999.

[32] Martha Steenstrup. Cluster-Based Networks. In Charles E. Perkins, editor, *Ad Hoc Networking*, chapter 4. Addison-Wesley Publishers, 2000.

[33] Paul F. Tsuchiya. The Landmark Hierarchy: A New Hierarchy for Routing in Very Large Networks. In *Computer Communication Review*, pages 35–42, Stanford, CA, August 1988.