Virtual Backbone Generation and Maintenance in Ad Hoc Network Mobility Management

Ben Liang and Zygmunt J. Haas
School of Electrical Engineering, Cornell University, Ithaca, NY 14850
email: liang@ee.cornell.edu, haas@ee.cornell.edu

Abstract— **In this paper, we present the implementation issues of a virtual backbone that supports the operations of the Uniform Quorum System (UQS) and the Randomized Database Group (RDG) mobility management schemes in an ad hoc network. The virtual backbone comprises nodes that are dynamically selected to contain databases that store the location information of the network nodes. Together with the UQS and RDG schemes, the virtual backbone allows both dynamic database residence and dynamic database access, which provide high degree of location data availability and reliability. We introduce a Distributed Database Coverage Heuristic (DDCH), which is equivalent to the centralized greedy algorithm for virtual backbone generation, but only requires local information exchange and local computation. We show how DDCH can be employed to dynamically maintain the structure of the virtual backbone, along with database merging, as the network topology changes. We also provide means to maintain connectivity among the virtual backbone nodes. We discuss optimization issues of DDCH through simulations. Simulation results suggest that the cost of ad hoc mobility management with a virtual backbone can be far below that of the conventional link-state routing.**

I. Introduction

In the ad-hoc network architecture, there is no pre-existing fixed network infrastructure. Nodes of an ad-hoc network are mobile hosts with similar transmission power and computation capabilities. Direct communication between any two nodes is allowed when adequate radio propagation conditions and network channel assignment exist, in which case we say that these two nodes are neighbors. Otherwise the nodes communicate through multi-hop routing [1]. The lack of fixed infrastructure suggests that some network functions, otherwise handled by the wireline backbone, must now be maintained by the nomadic nodes in the ad-hoc network.

Most of the proposed and existing systems directly send data packets to a destination node through pre-determined routes, without using any specialized databases to store the mobile nodes' location. To achieve this, the initiating node must either already have an up-to-date routing table to all the nodes in the network (pro-active routing) or try to determine the route on demand (reactive) [2]-[4], or, as more recently proposed, a combination of both [1]. For a large network with many nodes and frequent topology change, direct routing potentially poses very high traffic and computational demands. Multi-level ad-hoc routing schemes [5]-[7] with similarity to the cellular wireline-wireless hierarchy were also proposed, in which all packets are sent from the initiating node to the destination through a set of *backbone* nodes which comprises a centralized subnet. Since every packet within the network must go through the subnet, these schemes impose very high demand of channel bandwidth and node stability on the backbone.

We have previously proposed ad-hoc mobility management schemes that utilize location databases. These location databases form a *virtual backbone* that is dynamically distributed among the network nodes [8][9] and serve only as containers for location storage and retrieval. Routing is carried out in the flat network structure (see reference [1]). That is, the routes do not necessarily go through the databases. However, the node location in the databases can provide important information to the routing protocol, so that route searching is more localized.

Nodes containing the location databases can dynamically detach and re-attach to the network at any time due to mobile movements or changes in the communications environment. However, the ups and downs of a database's connectivity to the network should have minimum effect on the other nodes' communication. This imposes significant challenge in the design and operation of an ad-hoc network. The traditional algorithms for mobility management, which rely on the base-stations and Mobile Switching Centers, are not applicable here. For example, in the HLR[1]-like schemes, which mimic the cellular networks and set up fixed association between mobiles and databases, the databases would too often fail to provide the sought mobile's location due to frequent disconnections of the databases.

One obvious extension to the IS-41 and the GSM HLR/VLR[2] schemes is to use multiple HLRs in ad-hoc networks. In such a case, unless all HLRs have failed (probability of such an event can be made small using large number of HLRs) the location information is available to the call initiating mobile host. However, such schemes suffer from the disadvantages of fixed association between databases and the mobiles that store these locations.

An improvement to this scheme is to dynamically assign network nodes to perform the HLR function depending on the time-varying network node stability. When a node that serves as an HLR disconnects from the rest of the network, a new active node should be chosen to generate and maintain a new HLR with the same identity as the disconnected one. Such a "virtual HLR" or "adaptive HLR" scheme severs the association between a location database and the mobile host where the database resides. However, the rigid association between a mobile and the databases where the mobile's location information is stored is still maintained.

In [8] and [9], we proposed two location database arrangement schemes, namely, Uniform Quorum System (UQS) and Randomized Database Group (RDG), that are doubly dis-

[1] Home Location Register
[2] Visitor Location Register

tributed. Both of them dynamically assign network nodes to contain the location databases through employment of a virtual backbone, which is assumed to have been dynamically formed, contingent upon the network node stability and traffic and mobility patterns. At the same time, they allow a mobile to access randomly chosen groups of databases, which provide more robust and reliable location data availability than the fixed HLR schemes.

At any instant, some location databases in an ad-hoc network may have failed or may be separated from the network. However, because of the dynamic nature of the mobile and database association in the UQS and RDG schemes, as long as at least one quorum or database group remains, location updating and location querying are still possible in the entire network. This is not true for the HLR-like schemes, where loss of some HLRs, even though small in number, can often disable part of the network. Moreover, we have shown that the "adaptive HLR" scheme is a limiting case of both the UQS and the RDG schemes, and is in general inferior.

Thus, the distribution of responsibility among the databases is the key benefit of employing the UQS and RDG structures of database arrangement over a virtual backbone for ad hoc network mobility management. The virtual backbone architecture is the foundation for allowing dynamic location database residence in the UQS and RDG schemes, which provides high degree of location data availability and reliability.

However, [8] and [9] do not specify how to generate and maintain the virtual backbone. Reference [5] introduces a distributed way of generating a backbone-like set of cluster-heads. In [6], the creation of a spine architecture is discussed. Both structures require that every node is within one hop away from a backbone node, which would lead to too many backbone nodes and location databases. Therefore, the applicability of these schemes to our work is limited.

We can consider the generation of a virtual backbone as a Minimum Set Covering (MSC) problem over a graph representing the ad hoc network. The MSC problem is known to be NP-hard, and the greedy algorithm provides the best known approximation to the optimal solution [10]. However, the greedy algorithm is a centralized one, which is not suitable for use here due to the lack of stability in the ad hoc environment.

In this work, we introduce means to dynamically maintain the virtual backbone in a distributed fashion. We present a Distributed Database Coverage Heuristic (DDCH) for virtual backbone generation. DDCH generates a virtual backbone with only local information exchange and local computation. Furthermore, it is shown to be an *equivalent*, distributed implementation of the centralized greedy algorithm. DDCH can also dynamically regenerate location databases, which update the virtual backbone, maintaining its structural integrity.

The rest of this paper is organized as follows. We first describe the network model in Section II, and define the scope of the virtual backbone structure in Section III. We then present two virtual backbone generation schemes, first the centralized greedy algorithm and second the equivalent DDCH, in Section IV. We then explain, in Section V, how the structure and connectivity of the virtual backbone can be dynamically maintained as the network topology changes over time. Performance evaluations and optimization issues are then discussed in Section VI, followed by conclusions in Section VII.

II. NETWORK MODEL

We consider an ad hoc network with similarly equipped nodes, which have the same range of transmission power and capability. Each node has a fixed transmission radius. We assume that the nodes find out about their neighbors through a *Neighborhood Discovery Protocol*, in which periodic self-identifying 'HELLO' beacons are broadcast by each node's transmitter. Therefore, two nodes are neighbors of each other, if they are within each other's transmission radius and hear each other's beacon.

Two nodes that are not neighbors communicate through multi-hop routing. We define the distance between two nodes as the number of hops in the shortest (minimum-hop-count) path between these two nodes.

We assume that packets are transmitted between nodes through unicast communication. We assume that the underlying CSMA/CA-like MAC layer protocols, such as MACAW [11] or DBTMA [12], provide reliable shared access to the unicast channel, and solve the problems of hidden and exposed terminals.

We define the location of a node in terms of its positional relationships between the node and the other nodes. Note that the particular definition is irrelevant to the mobility management scheme presented here. In particular, we will define the location of a mobile host as the ID number of its nearest location database.

We further assume that, although the network topology changes frequently, the virtual backbone generation and maintenance protocols can be carried out much faster than the topological changes.

III. THE VIRTUAL BACKBONE

The virtual backbone facilitates mobility management with the UQS or the RDG scheme of location database arrangement. It comprises nodes that contain the location databases. One location database resides in each virtual backbone node (we will refer to such a node as a *database* throughout this paper.) The databases are interconnected through multi-hop connections. When there is no graph partitioning of the network nodes, the database and the inter-connecting paths form a connected graph. The regular network nodes are connected with the databases, possibly through multi-hop connections. We further require that each network node is no more than r hops away from its nearest database.

Unlike in the physical hierarchical structures, where a backbone consists of high power nodes and high capacity links, the nodes and links within the virtual backbone possess the same capacity and stability as those of the regular nodes and links. The network nodes are dynamically selected to join or leave the virtual backbone based on the changing node movement and link topology. Therefore, any node can potentially serve as a database at some point during the time period that it remains in the network.

The virtual backbone and the non-backbone nodes form a *logical* two-level hierarchy. A connection can be a multi-hop path that spans both the backbone and non-backbone nodes. In particular, the virtual backbone nodes can communicate between each other through routes that pass by the non-backbone nodes. Furthermore, the multi-hop nature of database interconnection does not necessitate that the network routing traffic use the virtual backbone as a major artery. The flat network structure, where routing involves the regular nodes, should be employed, which allows balanced loading on the nodes and the links.

In the UQS and RDG schemes, the network nodes query and update selected groups of the location databases. There are many ways of accomplishing this. In one scheme, we could make available to every node the routes to all databases. In another, we could keep the routing information to the databases among the databases only, so that a non-database node needs to send its location update or query packet to a nearby database first, which then forwards the packet to the intended databases. In this work, we assume that the latter scheme is chosen. Thus, the virtual backbone nodes need to maintain inter-connection among themselves at all times. The procedure for virtual backbone connectivity maintenance is described in Section V.

As shown in [8] and [9], the number of databases required for the UQS and RDG structures in the virtual backbone is usually small compared to the size of the network. Furthermore, we will show later that the size of the virtual backbone with the optimal r value is also moderate. Therefore, for a large ad-hoc network, the cost of mobility management using the virtual backbone is typically very small compared to routing within the entire network. Besides mobility management, the virtual backbone can also perform other network functions, such as employing the database nodes as local congestion control centers for nearby nodes, or serving as the authentication entity for secure information exchange, on which we will not elaborate in this paper. Thus, the cost of virtual backbone maintenance can by far be offset by the benefit it gains.

IV. Virtual Backbone Generation

Given the network topology and a radius r, we would like to compute a virtual backbone of the minimum size. Since the communication environment in an ad hoc network changes rapidly, distribution and localization is key to the establishment of the virtual backbone. The database selection should be based on only local computation. At the same time, the locally computed virtual backbone should be small, such that the interconnectivity of databases is easy to maintain.

In this section, we will first introduce the centralized greedy algorithm, then describe its distributed equivalence, DDCH.

A. *Centralized Minimum Set Covering Computation through the Greedy Algorithm*

For each network node, we define an *r-zone*, which consists of the node itself and all nodes r or less hops away, where r is the guaranteed maximum hop count from a node to its nearest database. Then, this problem can be restated as: find a set of databases with minimum cardinality, such that every node in the network is in at least one database's r-zone. Namely, the virtual backbone "covers" the entire network.

Thus, the computation of a virtual backbone can be reduced to the following minimum set covering problem: given a set of objects V (i.e. nodes) and a collection E of sets of these objects (i.e. r-zones), find a subset C (i.e. r-zones induced by virtual backbone) $\subseteq E$ of minimum cardinality, such that every element $v \in V$ belongs to at least one of the sets in C.

The MSC problem is well known to be NP-hard. A comparative study of different approximation algorithms for this problem [10] has suggested that a randomized greedy algorithm with redundancy elimination gives the best performance (producing the smallest covering set) in many general situations. A greedy algorithm for MSC has the structure as follows:

```
C = φ
while ∪C ≠ V and E ≠ φ,
    X = arg max_{Y∈E}{|Y|}
    C = C + {X}
    E = E − {X}
    ∀Y ∈ E, Y = Y − X
end
```

In the above maximum cardinality computation, ties can be broken lexicographically or at random. For the randomized greedy algorithm, the above process is run multiple number of times, with different random seeds for tie-breaking, and the one that leads to the minimum solution is taken. It has been shown that this generally achieves a slightly better solution than a one-shot greedy implementation, with the penalty of substantial increase in running time [10].

The above procedure can terminate with redundant sets in C. A redundancy elimination procedure is then applied on the solution:

```
repeat,
    redundancy = max_{X∈C}{min_{v∈X}{|Z| : v ∈ Y, Y ∈ Z, Z ⊆ C}}
    if redundancy > 1,
        C = C − arg max_{X∈C}{min_{v∈X}{|Z| : v ∈ Y, Y ∈ Z, Z ⊆ C}}
    end
until redundancy = 1
```

The greedy algorithm for MSC (MSC_GR) can be easily computed in polynomial time, provided that a central controller is given the full information of V and E. However, in an ad hoc network, centralized control is generally not desirable, due to the lack of stability in the communication environment. Full network information exchange is prohibitive in an ad hoc network with large number of nodes and frequently changing topology.

Next, we will show that the greedy algorithm can be implemented in a distributive fashion, if the MSC problem is stated as zone covering in a graph. We will first discuss the connectivity issue of an r-zone.

B. *Local r-Zone Connectivity Maintenance*

An r-zone has similar function as the *intrazone* structure introduced in [1]. Each node constantly monitors the identity and connectivity of other nodes within its r-zone. The connectivity between a node and other nodes within its r-zone can be accomplished by any one of the several intrazone routing protocols suggested in [1].

Here, we assume that a link-state algorithm is employed, with the range of link update limited to r hops. When a link between two nodes is broken, this new piece of link-state information is propagated to all nodes within $r-1$ hops away. When a new link is established between two nodes, in addition to propagating this information to all nodes within $r-1$ hops away, each node also sends to its neighbors a link-state table that contains r-zone link information updates due to the new link [13]. The node identity and node state (see Section IV-C) are also sent along with a link-state update. Thus, a node always has the current identity, state, and routing information of all nodes within its r-zone. Since link-state exchange and routing table computation is localized, the r-zone connectivity maintenance is efficient and robust.

In this way, an r-zone is a basic unit of localized node connectivity maintenance. As we will see later, the optimal selection of r is one that balances the trade-off between the cost of local database computation and the cost of maintaining the virtual backbone connectivity. Note here that the cost of maintaining the connectivity of an r-zone is independent of the size of the network.

C. The Distributed Database Coverage Heuristic

In the ad hoc network under consideration, a node is either a database or a regular node. A regular node could be in one of the following three states: *normal*, *panic*, or *samaritan*.

With an ideally functioning virtual backbone, every node is within r hops from a database. In this case, all nodes are in the normal state. A node enters the panic state if there is no database within its r-zone. It stays in the panic state until, either it becomes a databases, or a database appears in its r-zone, at which time it either goes into the samaritan state or returns to the normal state, under conditions to be described. A node in the samaritan state is connected with a database within r hops, but it could become a database to cover the panic nodes within its r-zone, if certain conditions are met. A node in the normal state does not participate in updating the virtual backbone.

We define the *dependency number* of a node as the number of panic nodes that are within r hops from the node (including the node itself.) Thus, the dependency number of a node is the number of new nodes to be covered by the virtual backbone, if the node itself becomes a database.

The centralized MSC_GR algorithm can now be rewritten as (with slight abuse of notation):

```
C = φ
while ∪C ≠ V,
    find node v with the maximum dependency number
    C = C + {v}
    re-compute dependency number
end
```

Then, the proposed DDCH replaces MSC_GR, and generates a virtual backbone as follows. Initially there is no database in the network, and nodes are not connected with any database. Therefore all nodes are in the panic state. A node in the panic or samaritan state sends a *state packet* to all nodes within $2r$ hops with the following format

ID	state	dep_num	hop

where `ID` is the identity number of the node, `state` indicates whether the node is in the panic state or the samaritan state, `dep_num` is the dependency number, and `hop` is a hop counter that is initiated to $2r$ and is decremented by 1 each time the packet is forwarded. A node receiving a packet with `hop` $= 1$ will discard the packet.

A node in the panic or samaritan state collects the state packets and extract the dependency numbers from all other panic or samaritan nodes within $2r$ hops. If the node itself has the largest dependency number (with random tie-breaking), it becomes a database and joins the virtual backbone. Otherwise, three scenarios can occur to a panic node: if no new database appears in its r-zone within a time threshold specified by the maximum duration of new database generation after the panic messages, the node remains in panic, re-computes its dependency number, and sends out a new state packet. If a new database appears in its r-zone, and there is no panic node in the r-zone, the node returns to the normal state. If a new database appears in its r-zone and there are still panic nodes in the r-zone, the node changes its state to samaritan, re-computes its dependency number, and sends out a new state packet.

Two scenarios can occur to a samaritan node that does not have the maximum dependency number. After waiting for the maximum time of new database generation, if there is no panic node in its r-zone, the node returns to the normal state. If there are still panic nodes in its r-zone, the node remains in the samaritan state, re-computes its dependency number, and sends out a new state packet.

The above procedure is performed by all panic nodes in the network, until each one of them either, 1) becomes a database, or 2) finds a database within r hops. Therefore, at the end, no panic node is left in the network, and the virtual backbone covers all nodes.

Thus, the DDCH procedure for panic nodes and samaritan nodes can be compactly written as shown in Table I and II.

TABLE I
DDCH FOR PANIC NODES

```
while panic,
    send and receive state packets within 2r hops
    if self has maximum dependency number
        become database node
    else
        wait for maximum database generation time
        if no database appears in r-zone
            remain in panic
        elseif no panic nodes in r-zone
            set state to normal
        else
            set state to samaritan
```

Note that, although we have described the above virtual backbone generation procedure in a synchronous fashion, it can be

TABLE II

DDCH FOR SAMARITAN NODES

```
while samaritan,
    send and receive state packets within 2r hops
    if self has maximum dependency number
        become database node
    else
        wait for maximum database generation time
        if no panic nodes in r-zone
            set state to normal
        else
            remain samaritan
```

carried out asynchronously. In an asynchronous mode of operation, a panic/samaritan node constantly collects the dependency number from nodes within $2r$ hops, and periodically decides, based on the current dependency information at the instant, whether it will become a database or send out a new state packet. In this case, the procedure still guarantees that the virtual backbone covers all nodes in the network, but the size of the virtual backbone may not be optimal. Furthermore, since the exchange of dependency numbers is limited to a local area, assuming that the network nodes do not move too fast, when the periodic waiting time is long enough, it is possible to achieve near synchronous performance.

Theorem I: The DDCH algorithm, while carried out synchronously, generates the same virtual backbone as the centralized MSC_GR algorithm does.

Proof: Let C be the virtual backbone set computed by MSC_GR. We first prove that all databases selected by the DDCH are in C.

Let v be a panic or samaritan node turned into a database via DDCH. Then, at the time when v is still in panic/samaritan, but is going to become a database, every other panic/samaritan node in its $2r$ vicinity has a smaller dependency number. At the same time, the selection of other new databases outside of v's $2r$-hop zone affects the panic nodes only within the r-zone of those databases. Therefore, the new database selection outside of v's $2r$-hop zone only decreases the dependency numbers of the nodes $r + 1$ or more hops away, but does not change the dependency number of v. Thus, in either the DDCH or the MSC_GR algorithm, v will always have higher dependency number than the other panic/samaritan nodes within its $2r$ vicinity. No other node within v's $2r$-hop zone would become a database before v does. Since MSC_GR is guaranteed to terminate with all panic nodes (in particular, those within v's r-zone) covered by at least one database, v will eventually be chosen to become a database in MSC_GR.

Secondly, with random tie-breaking, a node with the maximum dependency number over all panic/samaritan nodes always exists. This node has the maximum dependency number among the panic/samaritan nodes in its $2r$ vicinity, and, therefore, will become a database. Thus, DDCH does not terminate, but keeps on generating new databases that are also in C, until all nodes are covered by the virtual backbone. ■

Thus, given a graph representing the network topology and a zone radius r, DDCH generates a virtual backbone that has the same databases as one computed by the centralized MSC_GR algorithm.

Redundant databases can be eliminated once the databases have established connections among themselves, through the connectivity maintenance procedure to be described in Section V-B. We define the *coverage multiplicity* of a node as the total number of databases in its r-zone. Then, the *redundancy number* of a database is the minimum coverage multiplicity of all nodes in its r-zone. As shown in Section IV-A, there is redundancy in the database assignment if there exists a database whose redundancy number is greater than one.

The redundancy elimination procedure can be carried out locally as follows. First, the redundancy numbers are exchanged between databases that are $2r$ or less hops apart. Then, every database that has the maximum redundancy number among all databases within its $2r$-hop zone deletes itself from the virtual backbone set. The above redundancy number exchange and database deletion continues until the redundancy number of every database is reduced to one. Similarly to Theorem I, it's easy to prove that this distributed procedure eliminates the same redundant databases as does the centralized one given in Section IV-A.

Therefore, through only local information exchange and local computation, DDCH computes the best known polynomial-time approximating algorithm of generating the minimum covering virtual backbone set. Next we will show how DDCH can be employed in an dynamic scheme that updates the virtual backbone in order to adapt to the changing network topology.

V. Dynamic Virtual Backbone Maintenance

In a mobile ad hoc network, the topology is constantly changing and the graph of the virtual backbone should be updated as the network topology changes. There are two choices here to accomplish this. One is to periodically regenerate the virtual backbone through DDCH over the entire network. However, more preferable is a dynamic scheme that updates the virtual backbone concurrently as the network topology changes. Topology changes in far away parts of the network should not affect the local databases. Thus, the distribution and local computational nature of the DDCH algorithm allows it to become an active part of the dynamic virtual backbone maintenance scheme.

The structure of the virtual backbone is maintained by DDCH, while the connectivity of the virtual backbone is maintained by local beacon exchanges and database mutual link-state updates.

A. Structural Maintenance

As the network topology changes, a database might move away from the neighborhood it used to occupied, or it might totally detach from the network altogether (possibly due to lost of radio contact, power failure, or jamming signals.) In either case, the nodes originally covered by this database will either: 1) find another database in their perspective r-zones, or 2) in the event that no other database is found, enter the panic state. Nodes in the panic state, along with the samaritan nodes induced by them, then start the DDCH algorithm to locally generate new databases, until they are covered again by the virtual backbone. Redundancy elimination is then performed. In order not to interrupt the operation of the original virtual backbone nodes, only the newly generated databases are allowed to be eliminated.

The above process repeats as the network topology changes over time. In order to prevent over-sizing of the virtual backbone, databases are erased in regions where there are too many of them. This can be accomplished by merging two databases that are within a threshold distance, D hops, of each other, where D is a design parameter.

We will see in Section V-B that the databases constantly monitor each other's location and availability through the virtual backbone connectivity process. When two databases, DB_1 and DB_2, are within D hops of each other, we first find the number of nodes in their respective r-zones, n_1 and n_2. If $n_1 < n_2$, the content of DB_1 (in our case, the location information of nodes who have updated in DB_1) is copied into DB_2, and then DB_1 deletes itself from the virtual backbone, and vice versa. With database merging, some nodes will be put into the panic state due to the deletion of the databases covering them. They will then trigger DDCH to update the virtual backbone.

Database movement, database detachment, and database deletion have the same effect, and, therefore, are transparent to the network nodes covered by the databases. When databases disappear from a region, the neighborhood nodes locally regenerates new databases to maintain connectivity to the virtual backbone.

For simplicity of presentation, we assume here that DDCH is carried out synchronously. Thus, we can view the system as having a globally synchronized timer, such that database updates are performed starting at some periodic database-updating instants by all nodes currently in the panic or samaritan state. We assume that the DDCH algorithm is carried out faster than the topological changes due to node movement. Thus, DDCH terminates when enough new databases have been generated to cover all previous panic nodes.

Given the zone radius r and the merging distance D, database merging and regeneration keep the size of the virtual backbone at a stable equilibrium over time. As we will show in Section VI, the size of the dynamically maintained virtual backbone is only slightly larger than the size of a virtual backbone periodically regenerated by applying DDCH or MSC_GR over the entire network.

B. Connectivity Maintenance

In order to facilitate uninterrupted location updates and queries in the UQS and RDG schemes, as explained in Section III, a virtual backbone node needs to have up-to-date routes to all other backbone nodes. Therefore, as the network topology changes, the databases need to maintain their inter-connectivity.

We approach this problem by first proving the following theorem.

Theorem II: In a connected network with the zone radius r, a database can always find at least one other database within $2r + 1$ hops away.

Proof: Use $|uv|$ to denote the distance between two network nodes u and v. Suppose there exists a database u_1, whose nearest database u_2 is $2r + 2$ or more hops away. Then, in a minimum-hop-count path between u_1 and u_2, there exists a non-database node v, such that $|u_1v| \geq r + 1$ and $|u_2v| \geq r + 1$. Furthermore, $|u_1u_2| = |u_1v| + |u_2v|$. Since all nodes are covered by the virtual backbone, there exists a database u_3, such that $|u_3v| \leq r$. Thus, $|u_1v| + |u_3v| < |u_1u_2|$. Since $|u_1u_3| \leq |u_1v| + |u_3v|$, this leads to $|u_1u_3| < |u_1u_2|$, which contradicts with the assumption that u_2 is the nearest database to u_1. ■

Thus, if we define *database adjacency* as the state of being within $2r + 1$ hops of distance, the virtual backbone is interconnected via paths between adjacent databases.

In order to maintain connectivity between adjacent databases, database connectivity packets are piggy-backed within the periodic neighborhood discovery beacons by all nodes. The packets are used by a node to inform its neighbors of the shortest route from the node to databases within $2r$ hops away. One packet is transmitted for each database, and it has the following format

ID	DB_ID	hop	path

where `ID` is the identity number of the node, `DB_ID` is the identity number of the database, `hop` is distance to the database, and `path` is the current route to the database.

In this way, a simple distance-vector protocol is in place to provide every node a route to databases $2r + 1$ hops away. In particular, the adjacent databases determine the routes between each other through listening to the beacons and extracting the database connectivity packets.

Then, any suitable routing protocol can be applied to maintain connectivity among all databases, treating the multi-hop connection between two adjacent databases as a regular radio link. In this work, we have chosen a link-state like protocol, in which a database pro-actively monitors the "adjacent-database-link-state" of the entire virtual backbone.

VI. Performance Evaluation and Optimization

In this work, we are most concerned with the bandwidth usage by the control messages in maintaining the virtual backbone. The cost of virtual backbone maintenance, in units of control packets per unit time, is a summation of the cost of r-zone connectivity maintenance, C_{zone}, the cost of DDCH database regeneration, C_{DDCH}, the cost of merging databases, C_{merge}, and the cost of virtual backbone connectivity maintenance, $C_{connect}$. The cost of redundancy elimination is omitted here, since it involves few databases and has negligible magnitude. Thus,

$$C_{total} = C_{zone} + C_{DDCH} + C_{merge} + C_{connect} \, .$$

Simulations are performed with a model ad hoc network with various values of node density and relative node velocity. Node density is reflected by the average number of neighbors, n, that a node has. Node velocity is modeled by a two-dimensional zero-mean Gauss-Markov process [14]

$$\begin{aligned} v_{i+1}^{x} &= \alpha v_i^x + \sigma\sqrt{1-\alpha^2}u_i^x \\ v_{i+1}^{y} &= \alpha v_i^y + \sigma\sqrt{1-\alpha^2}u_i^y \, , \end{aligned}$$

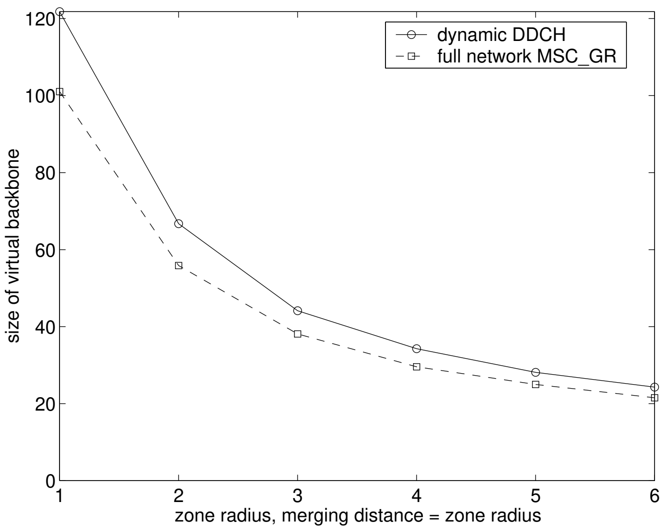

Fig. 1. Dynamic virtual backbone through DDCH

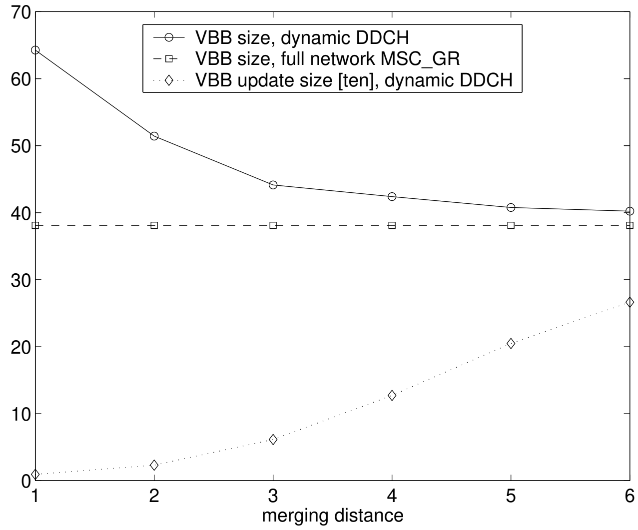

Fig. 2. Effect of merging distance selection

where α represents the amount of memory in a node's velocity between two time steps, σ is the standard deviation of the velocity, representing how fast a node moves, and u^x and u^y are i. i. d., zero mean, and unit variance, Gaussian random processes. The velocity in both directions has the units of transmission radius.

All simulations run with 500 nodes. A square coverage area is assumed. In order to keep the number of nodes in the coverage area constant, nodes that move beyond the square boundaries bounce back with symmetrical incident and reflective angles.

In computing C_{merge}, we have assumed that the size of a database is equal to the total number of nodes in the network. Therefore, merging a database into another means passing, along a path between this two databases, a packet 500 times as large as the control packets, such as the state packet sent out by the panic nodes. In computing C_{zone} and $C_{connect}$, we assume that a link-state updating table is transmitted via a packet whose size is equivalent to the length of the table times the size of a control packet.

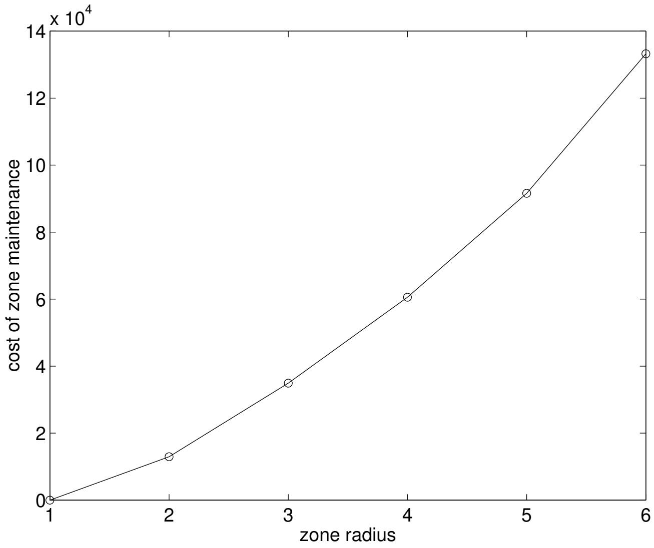

Fig. 3. Cost of r-zone connectivity maintenance

A. Dynamic DDCH vs. Full Network DDCH/MSC_GR

Dynamic virtual backbone maintenance, with DDCH and databases merging, induces a virtual backbone that is necessarily larger than that created by periodic virtual backbone generation with full network DDCH/MSC_GR. Figure 1 shows a side-by-side comparison of the average (over time) size of the virtual backbone induced by dynamic DDCH and full network DDCH/MSC_GR, as a function of the zone radius r, where r varies from 1 to 6. In every case, the merging distance D is set equal to r. The other parameters are $n = 6$, $\alpha = 0.5$, and $\sigma = 0.2$. We see here that the size of the dynamic virtual backbone is only about 15% larger than that created in the full network DDCH/MSC_GR, the best known approximation to the MSC problem.

Figure 2 shows the size of the dynamic virtual backbone vs. the merging distance D. Here $r = 3$, and D varies from 1 to twice the zone radius. The other parameters are $n = 6$, $\alpha = 0.5$, and $\sigma = 0.2$. Also plotted is the size of a virtual backbone by full network DDCH/MSC_GR, and the number of initial panic nodes right after the database merging (in units of ten).

We see here that as the merging distance increases, the size of the dynamic virtual backbone approaches the outcome of full network DDCH/MSC_GR. This is expected, since a longer merging distance implies that more databases are merged, which leads to more uncovered nodes, resulting in dynamic DDCH performing more like full network DDCH. The number of panic nodes due to merging of the databases at $D = 6$ is around 300, which is about 60% of the total number of nodes in the network.

Next, we consider the issues of cost trade-off in the zone radius and merging distance selection.

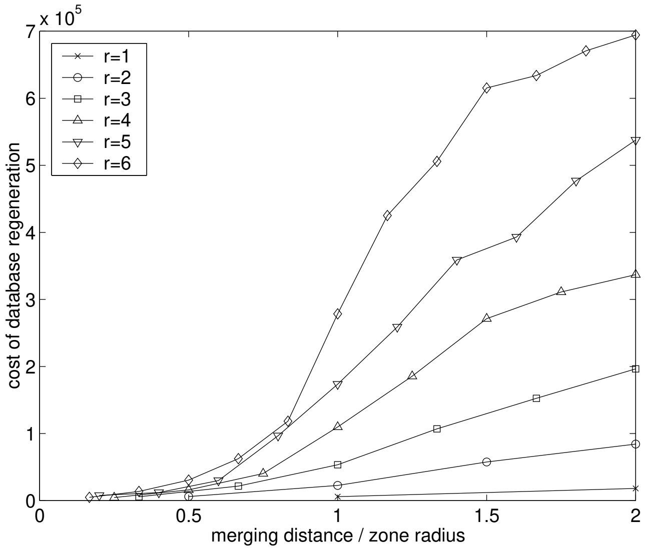

Fig. 4. Cost of dynamic DDCH database regeneration

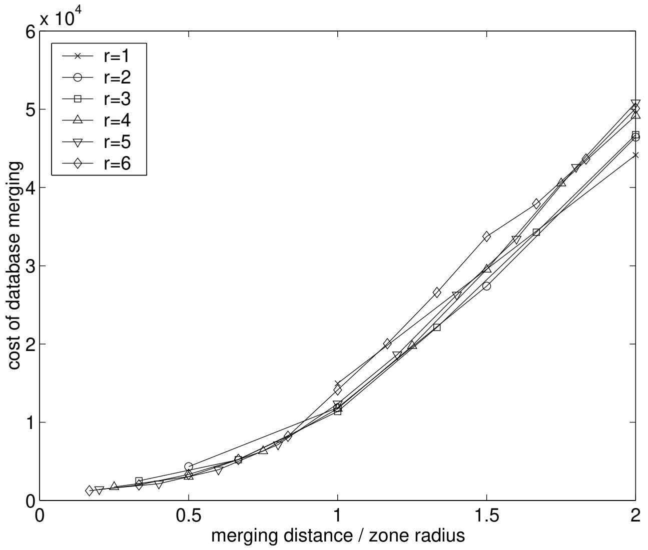

Fig. 5. Cost of merging databases

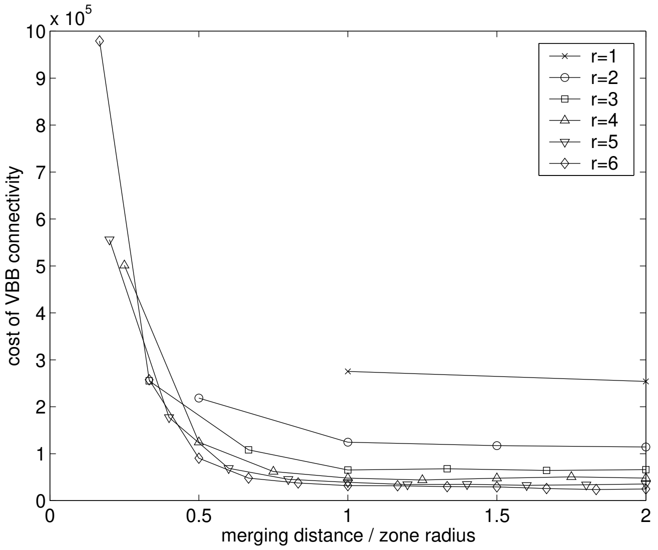

Fig. 6. Cost of virtual backbone connectivity maintenance

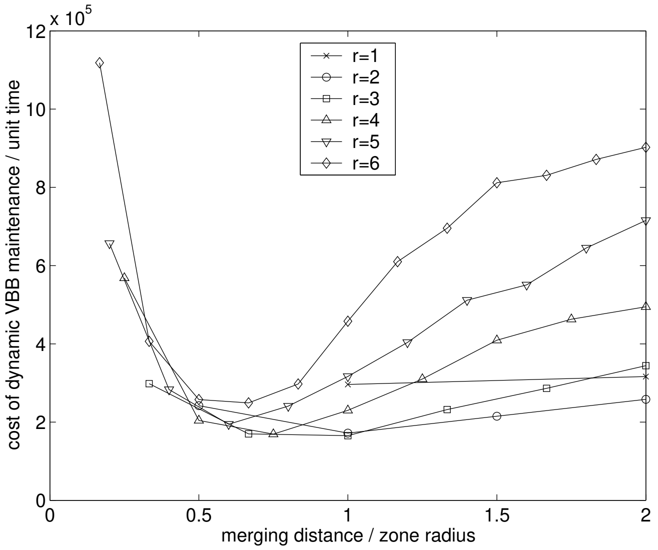

Fig. 7. Cost of dynamic virtual backbone maintenance

B. Optimal Zone Radius and Merging Distance Selection

As stated in Section IV, the zone radius is a parameter that balances the trade-off between the cost of local connectivity maintenance and global connectivity management. Figure 3 shows that C_{zone} is an exponentially increasing function of the zone radius. In Figures 4 - 6, we plot C_{DDCH}, C_{merge}, and $C_{connect}$ vs. the merging distance and the zone radius. In each of these figures, we have $n = 6$, $\alpha = 0.5$, $\sigma = 0.2$, the zone radius varies from 1 to 6, and D varies from 1 to twice the zone radius.

Figures 4 and 5 show that the cost of DDCH databases regeneration and the cost of database merging are both increasing function of the merging distance. In particular, the cost of DDCH becomes significant, when the merging distance is larger than the zone radius.

These figures also show that the DDCH cost is an increasing function of zone radius, which is expected, due to the exponential increase in the cost of zone connectivity maintenance. On the other hand, the database merging cost is not very sensitive to the zone radius change.

Figure 6 shows that the cost to maintain the virtual backbone connectivity falls sharply as the merging distance increases from one hop to r hops. For example, in the case of $r = 4$, as D goes from 1 to 4, $C_{connect}$ drops about 10 times from 5.0×10^5 to 4.8×10^4. However, when D is larger than the zone radius, $C_{connect}$ is mostly independent of D.

This figure also shows that $C_{connect}$ is a decreasing function of r.

Figure 7 shows C_{total} vs. r and D, which is a summation of the costs shown in Figures 3-6. This figure suggests the optimal zone size and merging distance for the given ad hoc network. From the graph, we see that a configuration of $r = 3$ and $D = 3$ incurs the minimum dynamic virtual backbone maintenance cost, which is about 1.7×10^5.

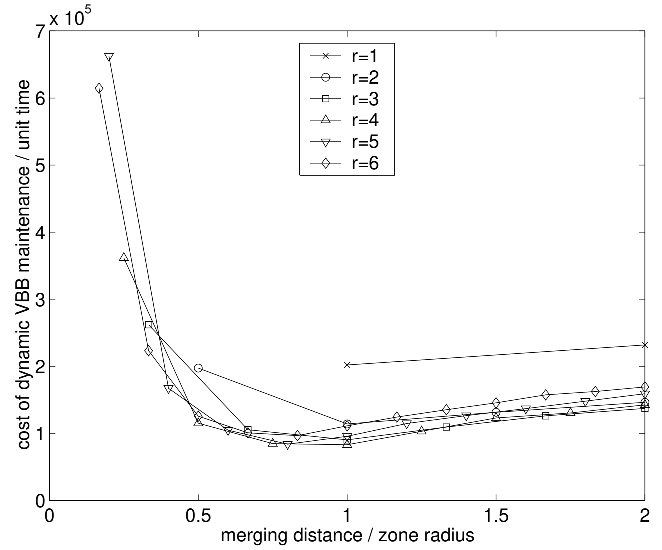

Fig. 8. Cost of dynamic virtual backbone maintenance for network with 4 neighbors per node

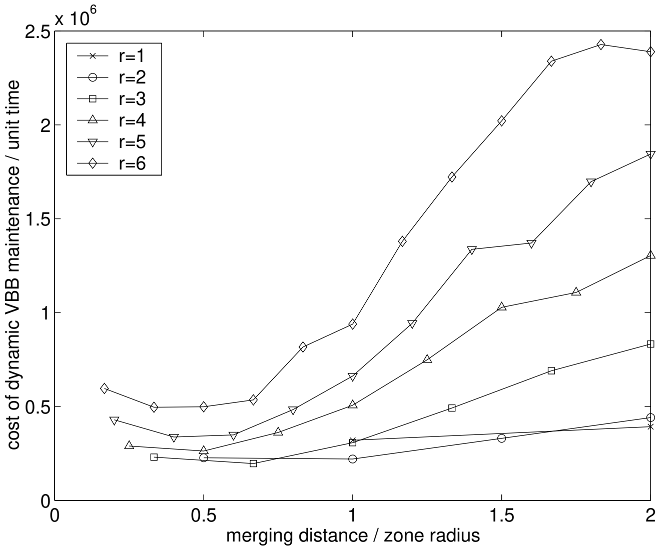

Fig. 9. Cost of dynamic virtual backbone maintenance for network with 8 neighbors per node

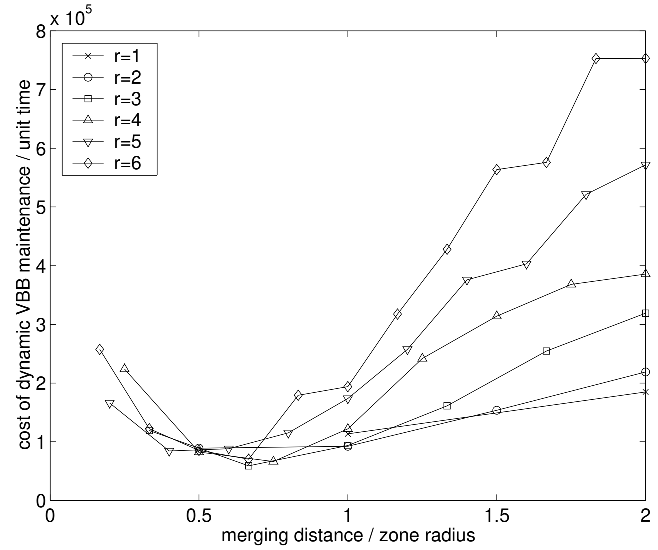

Fig. 10. Cost of dynamic virtual backbone maintenance for network with $\sigma = 0.05$ radius/unit time

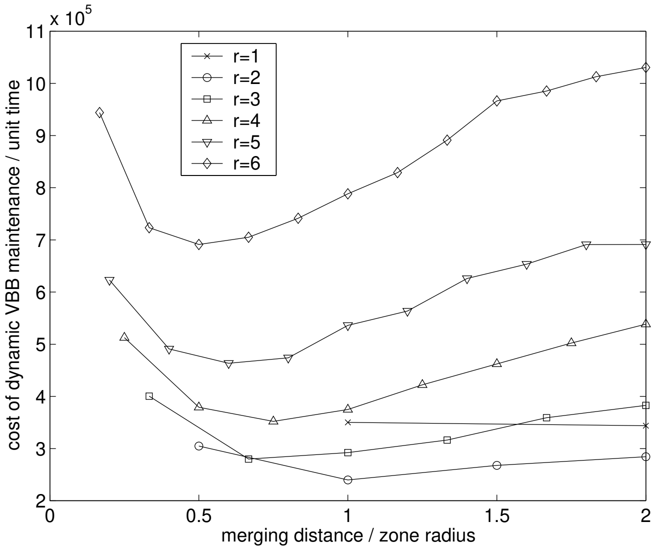

Fig. 11. Cost of dynamic virtual backbone maintenance for network with $\sigma = 0.8$ radius/unit time

In general, the optimal zone radius and merging distance vary with different network parameters. Figure 8 and 9 show C_{total} vs. r and D for a lower density network, with four neighbors for each node on the average, and a higher density network, with eight neighbors for each node on the average, respectively. Both have the same mobility model as in Figure 7. Figure 10 and 11 show C_{total} vs. r and D for a lower velocity network, with $\sigma = 0.05$ radius/unit time, and a higher velocity network, with $\sigma = 0.8$ radius/unit time, respectively. The other parameters in both are the same as in Figure 7. In particular, they have the same database-updating period as described in Section V-A.

From Figure 8, we see that a configuration of $r = 4$ and $D = 4$ incurs the minimum dynamic virtual backbone maintenance cost, which is about 8.3×10^4. From Figure 9, we see that a configuration of either $r = 2$ and $D = 2$, or $r = 3$ and $D = 2$ incurs the minimum dynamic virtual backbone maintenance cost, which is about 2.0×10^5. Therefore, the optimal zone radius decreases as the network node density increases. However, the optimal merging distance is almost always equal to the zone radius.

From Figure 10, we see that a configuration of either $r = 3$ and $D = 2$, or $r = 4$ and $D = 3$ incurs the minimum dynamic virtual backbone maintenance cost, which is about 5.9×10^4. From Figure 11, we see that a configuration of $r = 2$ and $D = 2$ incurs the minimum dynamic virtual backbone maintenance cost, which is about 2.4×10^5. Therefore, the optimal zone radius decreases as the network node velocity increases, but the optimal merging distance is almost always near the zone radius.

Note here that, although the cost of virtual backbone maintenance appears to increase slower than the increase in node velocity, in a system that is designed to match the database-updating frequency accordingly with the node velocity, the total

TABLE III

COST COMPARISON, WITH VARIOUS NODE DENSITY

n	4	6	8
virtual backbone	8.3×10^4	1.7×10^5	2.0×10^5
link-state	7.5×10^5	1.6×10^6	2.8×10^6

TABLE IV

COST COMPARISON, WITH VARIOUS NODE VELOCITY

σ	0.05	0.2	0.8
virtual backbone	5.9×10^4	1.7×10^5	2.4×10^5
link-state	4.1×10^5	1.6×10^6	5.5×10^6

cost should be linearly increasing with the node velocity.

C. Comparison with Pure Link-State Routing

Since we have used the link-state routing protocol in both r-zone and virtual backbone connectivity maintenance, it is interesting to compare the cost of the dynamic virtual backbone with the cost of link-state routing over the entire network. In Table III, we tabulate the cost of both schemes for three ad hoc networks with the same mobility pattern, $\alpha = 0.5$ and $\sigma = 0.2$, but different node densities, where $n = 4$, 6, and 8, respectively. In Table IV, we tabulate both costs for three ad hoc networks with the same node density, $n = 6$, but different mobility patterns, where $\alpha = 0.5$, and $\sigma = 0.05, 0.2$, and 0.8, respectively.

Here we see that dynamically maintaining the virtual backbone costs only about 10% as much as the link-state protocol. Of course, this is not a fair comparison, since maintaining the virtual backbone does not give a message initiating node a direct route to the destination node. However, this does provide a metric, against which the efficiency of the DDCH dynamic virtual backbone maintenance scheme is measured.

As shown in [8] and [9], with suitable database arrangement schemes such as UQS and RDG, the node location information stored in the databases are easily accessible. At the very least[3], after obtaining the identity number of the destination node's nearest database through database queries, a message initiating node can use the readily available routes between the databases to forward the message to the destination. Therefore, in this sense, maintaining the virtual backbone does provide a route between nodes, which justifies the above comparison.

More importantly, the cost of link-state is a heavier-than-linear increasing function of the node density, which is the case with many other direct routing protocols. However, the cost of virtual backbone maintenance seems to be an sub-linear function of the node density, suggesting that it is scalable to large and dense networks.

VII. CONCLUSIONS

In this paper, we have described the implementation details of a virtual backbone that supports the operations of the UQS and RDG schemes for mobility management in ad hoc networks. We have presented a locally computed Distributed Database Coverage Heuristic for virtual backbone generation, which is shown to be equivalent to the centralized greedy algorithm for computing a minimum covering database set. With DDCH, database merging, and a database connectivity maintenance protocol, the virtual backbone can be dynamically maintained as the network topology changes. Simulation results show that the dynamically maintained virtual backbone has the size of only 15% above that of the minimum set covering approximation provided by the greedy algorithm, which is the best known polynomial time algorithm for MSC.

Due to the local computation nature of the DDCH scheme, the adaptation to topological changes is fast, and the virtual backbone is scalable to various network size, density, and node velocity. With optimally chosen r-zone size and database merging distance, the cost of maintaining the virtual backbone is only about one tenth of the cost of a link-state protocol applied to the ad hoc network.

We have previously shown that proper utilization of location databases can lead to highly robust node location information access in ad hoc networks, even though the connectivity of such a network is inherently unstable due to its lack of fixed infrastructures. Used together with suitable database arrangements, such as the UQS and RDG scheme, the dynamically maintained virtual backbone can provide the efficiency and reliability sought in in ad hoc mobility management.

[3] However, we do not recommend this mode of operation. Routing should be carried out among all nodes and links, so as not to create congestion within the virtual backbone. The exact routing methods are not within the scope of this paper.

REFERENCES

[1] M. R. Pearlman and Z. J. Haas, "Determining the optimal configuration for the Zone Routing Protocol," *IEEE Journal on Selected Areas in Communications*, vol. 7, no. 8, pp. 1395-1414, August 1999.

[2] J. J. Garcia-Luna-Aceves et al, "Analysis of routing strategies for packet radio networks," *Proceedings of IEEE INFOCOM'85*, Washington, DC, March 1985.

[3] B. M. Leiner, D. L. Nielson, and F. A. Tobagi, "Issues in packet radio network design," *Proceedings of the IEEE*, vol. 75, pp. 6-20, January 1987.

[4] J. Jubin and J. D. Tornow, "The DARPA packet radio network protocols," *Proceedings of the IEEE, Special Issue on Packet Radio Networks*, vol. 75, pp. 21-32, January 1987.

[5] A. Ephremides, J. E. Wieselthier, and D. J. Baker, "A design concept for reliable mobile radio networks with frequency hopping signaling," *Proceedings of the IEEE*, vol.75, no.1, 1987.

[6] B. Das and V. Bharghavan, "Routing in ad-hoc networks using minimum connected dominating sets," *IEEE Int. Conf. on Communications*, June, 1997.

[7] J. Sharony, "A mobile radio network architecture with dynamically changing topology using virtual subnets," *MONET*, vol. 1, no. 1, pp. 75-86, 1996.

[8] Z. J. Haas and B. Liang, "Ad hoc mobility management with uniform quorum systems," *ACM/IEEE Transactions on Networking*, vol. 7, no. 2, pp. 228-240, April 1999.

[9] Z. J. Haas and B. Liang, "Ad hoc mobility management with randomized database groups," *Proceedings of IEEE ICC*, June 1999.

[10] T. Grossman and A. Wool, "Computational experience with approximation algorithms for the set covering problem," *European Journal of Operational Research*, no. 101, pp. 81-92, 1997.

[11] V. Bharghavan, S. Shenker, D. Demers, and L. Zhang, "MACAW: a medium access protocol for wireless LANs," *Proceedings of ACM SIGCOMM*, August 1994.

[12] Z.J. Haas and J. Deng, "Dual Busy Tone Multiple Access (DBTMA) - performance evaluation," *Proceedings of IEEE VTC*, May 1999

[13] Z. J. Haas and M. R. Pearlman, "The Zone Routing Protocol (ZRP) for ad Hoc networks," Internet Draft, draft-ietf-manet-zone-zrp-02.txt, June 1999.

[14] B. Liang and Z. J. Haas, "Predictive distance-based mobility management for PCS networks," *Proceedings of IEEE INFOCOM*, March 1999.