Hybrid Routing in Ad Hoc Networks with a Dynamic Virtual Backbone

Ben Liang, *Member, IEEE* and Zygmunt J. Haas, *Senior Member, IEEE*

***Abstract*— Virtual Backbone Routing (VBR) is a scalable *hybrid* routing framework for ad hoc networks, which combines local proactive and global reactive routing components over a *variable-sized* zone hierarchy. The zone hierarchy is maintained through a novel distributed virtual backbone maintenance scheme, termed the Distributed Database Coverage Heuristic (DDCH), also presented in this paper. Borrowing from the design philosophy of the Zone Routing Protocol, VBR limits the proactive link information exchange to the local routing zones only. Furthermore, the reactive component of VBR restricts the route queries to within the virtual backbone only, thus improving the overall routing efficiency. Our numerical results suggest that the cost of the hybrid VBR scheme can be a small fraction of that of either one of the purely proactive or purely reactive protocols, with or without route caching. Since the data routes do not necessarily pass through the virtual backbone nodes, traffic congestion is considerably reduced. Yet, the average length of the VBR routes tends to be close to optimal. Compared with the traditional one-hop hierarchical protocols, our results indicate that, for a network of moderate to large size, VBR with an optimal zone radius larger than one can significantly reduce the routing traffic. Furthermore, we demonstrate VBR's improved scalability through analysis and simulations.**

***Index Terms*— Ad hoc network, virtual backbone, hybrid routing, zone routing, clustering, scalability.**

I. Introduction

In the ad hoc networking environment, node mobility and topological instability limit the applicability of routing protocols previously developed for wireline networks[1][2][3][4][5]. Routing protocols that are designed for ad hoc networks include *flat* and the *hierarchical* protocols. In a flat routing protocol, all nodes serve the same set of routing functions, while in a hierarchical routing protocol, nodes are differentiated by location or association.

The flat protocols can be roughly divided into three categories: *proactive*, *reactive*, and *hybrid*. A proactive routing protocol, also called a *table-driven* protocol, requires that each node maintains an up-to-date routing table, such that a route is readily available when data packets need to be send out. Routing protocols, such as DSDV[6], FSR[7], OLSR[8], STAR[9], TBRPF[10], and WRP[11], are examples of proactive protocols. In reactive routing protocols, also called *on-demand* protocols, a node is not required to maintain a routing table (although route caches may be kept), but instead a route query process is initiated whenever it is needed. Routing protocols such as ABR[12], AODV[13], DSR[14], and TORA[15] are examples of reactive protocols. A reactive protocol avoids the control message overhead incurred in building routing tables, which may never be used before the routes change due to the variation in node connectivity overtime. However, when routes are requested, nodes need to send out route query messages into a large part of the network, which could lead to the delay of route response and potentially a large penalty in network resources. A hybrid flat protocol, the Zone Routing Protocol (ZRP), was proposed by Haas and Pearlman[16] to combine the benefit of both approaches. In this protocol, a node proactively maintains only the link information of nodes within a variable-sized local neighborhood called the *routing zone*, and it reactively sends out route queries to faraway destinations through an efficient bordercasting algorithm.

In this work, we are concerned with routing protocols that are scalable to network size. The key to a scalable routing protocol is the *selective representation* of topology details. First, it is usually not necessary to store in each node the entire set of network links. This is so, since a node seldom utilizes the links in the faraway parts of the network and the acquisition of the status of faraway links requires extraneous overhead. Second, route queries should be limited to a small portion of the network that has high probability of returning a usable route, since it is expensive for a source node to send route queries to every node in the network. In ad hoc network routing, selective topology representation is particularly important, since there is frequent link breakage and link establishment in all parts of the network, and since the network bandwidth and other resources are scarce. Many of the aforementioned flat routing protocols carry out some degree of selective topology representation. These techniques include limited-scope link updating[7][8][16], source-tree-limited link storage[9], expanding ring search[13], and early query termination through route caching[14][15][17].

Besides these techniques, an important form of selective topology representation is naturally realized in the hierarchical routing protocols. Hierarchical routing was employed in the precursor of today's ad hoc networks, the DARPA packet radio networks[3][18], such as PRNet and SURAN. More recent contributions include LCA[19], CBRP[20], Spine[21], CEDAR[22], HSR[7], LANMAR[23], MMWN[24], NTDR[25], and ZHLS[26]. Hierarchical routing

Manuscript received August 28, 2001; revised March 2, 2004; accepted July 3, 2005. The associate editor coordinating the review of this paper and approving it for publication was B. Li. This work was supported in part by the National Science Foundation grant numbers ANI-0329905, ANI-0081357, and CNS-9980521, by the DoD MURI (Multidisciplinary University Research Initiative) Program administered by the Air Force Office of Scientific Research (AFOSR) under contract F49620-02-1-0217, and by the Natural Sciences and Engineering Research Council of Canada.

Ben Liang is with the Department of Electrical and Computer Engineering, University of Toronto (email: liang@comm.utoronto.ca). Zygmunt J. Haas is with the School of Electrical and Computer Engineering, Cornell University (email: haas@ece.cornell.edu).

protocols hide the topology details of the faraway parts of the network by organizing the nodes into hierarchical layers. Nodes within pre-defined proximity are grouped into *supernodes*, which could be furthered grouped into higher level super-supernodes. A node is given knowledge of faraway nodes only in terms of their higher level associations.

In this paper, we present the Virtual Backbone Routing (VBR) scheme, a hierarchical routing protocol that is hybrid in nature. It combines hierarchical and zone routing in a simple, yet effective form. It is based on the ZRP's concept of combining proactive local zone routing with reactive global route queries. However, unlike ZRP, VBR utilizes the notion of a *virtual backbone (VB)* to efficiently direct the route querying control traffic.

VBR is unlike CBRP, CEDAR, HSR, or NTDR, which requires that each node is at most one hop away from the head of a supernode. Instead, VBR employs the novel concept of *Distributed Database Coverage Heuristic (DDCH)* for virtual backbone generation and maintenance that supports variable-sized routing zones, using only local link information. VBR is unlike LANMAR, where nodes are pre-assigned to local groups before network deployment, or ZHLS, which requires the usage of Global Positioning System to define its geographical zones. Instead, VBR employs DDCH for dynamic local-zone construction and maintenance as the network topology changes. MMWN is geared toward multimedia communications over a mixture of ad hoc networks and wireline networks, and hence employs non-overlapping location areas and a strict location registration/de-registration mechanism. Such a mechanism is suitable for establishing virtual-circuits to guarantee the quality-of-service during a multimedia session, but could also incur large communication overhead. In VBR, the routing zones can overlap, and a more efficient reactive route querying mechanism is employed. Furthermore, as shown in Section V-C, VBR can employ both direct route caching and VB path caching to minimize routing delay and reduce routing overhead.

By applying selective topology representation in both the *zone-limited* link information maintenance and the *VB-constrained* route queries, VBR achieves higher level of efficiency and scalability than purely proactive or purely reactive routing. Our numerical results show that the cost of VBR can be a small fraction of that of either one of its component protocols. Furthermore, since the data routes do not necessarily go through the virtual backbone, congestion is reduced and the length of VBR generated routes are near optimal. Finally, simulation results suggest that in a network with more than 200 nodes and moderate route request rate, VBR with an optimal zone radius (generally larger than 2) can reduce the overhead of similar 1-hop protocols by 40% or more.

The rest of this paper is organized as follows. In Section II, we provide an overview of VBR. In Section III, we introduce the network model assumptions. In Section IV, we present the DDCH, a scheme to dynamically maintain the virtual backbone. In Section V, we explain how to use the virtual backbone hierarchy to achieve efficient routing. The performance evaluations and optimization issues are discussed in Section VI. Finally, further discussions of related work and the concluding remarks are given in Sections VIII and VII.

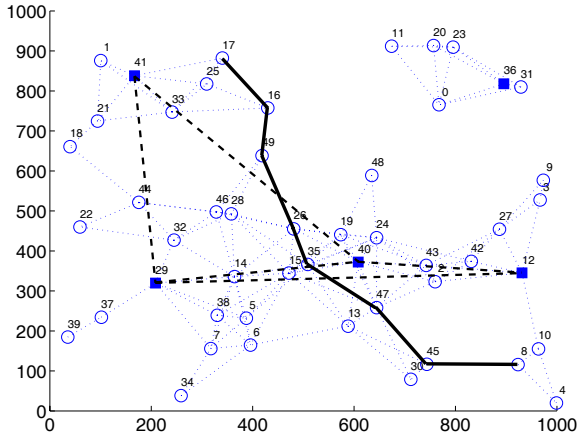

Fig. 1. Snapshot of VBR simulation with 50 nodes and $r = 2$. *Legend* - **circles**: regular nodes; **solid squares**: VB nodes/databases; **thin dotted lines**: radio links; **dashed lines**: multi-hop virtual links; **solid line**: the actual data route between node 8 and node 17.

II. Virtual Backbone Routing

A. Virtual Backbone Routing

The VBR framework consists of the local proactive and the global reactive components. As in ZRP, we define a local *routing zone* for each network node. The zone size is determined not geographically but by its radius, r, in hops, such that the routing zone of a node consists of the node itself and all nodes r or less hops away. However, since VBR relies on the virtual backbone to efficiently disseminate the route query packets, the *proactive component* of VBR consists of the VB maintenance in addition to the local routing-zone connectivity maintenance.

For local connectivity maintenance, each node monitors the state of all nodes and the status of all links within its routing zone through a proactive routing protocol. For the VB maintenance, a set of VB nodes is dynamically selected and refreshed through a Distributed Database Coverage Heuristic (DDCH), such that each node either belongs to the set of VB nodes or is at most r hops away from one. The VB has the following additional properties:

- The network nodes join or leave the VB as a result of node movement and changes in link topology. Therefore, any node can potentially serve as a VB node at some point during the time period that it is part of the network.
- The VB nodes in two neighboring zones establish a virtual link that spans multiple regular nodes. Thus, in a connected network, the VB nodes are interconnected through these virtual links.
- During the global routing stage, a VB node serves as a database queried by the source nodes for the link-state information of all nodes within the VB node's routing zone. Therefore, in this paper, we use the phrase "VB node" and the word "database" interchangeably.

The *reactive component* of VBR is applied when a source node needs to find a route to a destination node outside of its routing zone. The source first sends a route query to its

nearest database (unless the source itself is a database). If this database does not see a route to the destination, it broadcasts the route query to all other databases through the virtual links between the VB nodes. After receiving the route query, a database whose routing zone contains the destination, sends a route reply, through the reversed VB path, back to the query originating database. Each database along the way computes the best route segments that it can see within its zone, and appends them in the route reply packet. Thus the compiled route does not have to be contained within the VB. Finally the query originating database selects and establishes the best route for the source node.

Figure 1 shows a sample snapshot of VBR simulation in OPNET™, with 50 nodes and $r = 2$. As demonstrated in this figure, the data routes do not necessarily pass through the VB nodes. Also note that the VBR protocol functions properly even when there is network partitioning. Each partition maintains its own VB and uses it to set up routes within the partition.

In VBR, since only the local connectivity information is proactively maintained, the routing traffic scales well as the network size increases. Also, since the route query packets are only transmitted over the VB virtual links, the global route querying traffic is localized and directed. Furthermore, local and global routing traffic can be tuned to minimize and balance the traffic load. The local storage can also be reduced, since only the local routes and the virtual links need to be stored.

Next, we provide an analytical framework that demonstrates the scalability of VBR.

B. Scalability of VBR

Let N be the number of nodes in the network and λ be the route request rate per node. We define the cost of a routing protocol as the amount of routing control packets per node per unit time. Suppose we are given a proactive routing protocol that has routing cost $O(N^x)$ per node per unit time, and a reactive routing protocol that has routing cost $O(\lambda N^y)$ per unit time.[1] We would like to estimate the routing cost of VBR, based on a combination of these two protocols.

The routing cost of VBR is divided into three parts: the cost of local connectivity maintenance, C_{local}, the cost of maintaining the VB, C_{VB}, and the cost of route queries, C_{query}. Suppose each node proactively maintains the link-state information within a routing zone with R nodes. Then

$$C_{local} = O(R^x) \ . \tag{1}$$

As shown in Section IV, the VB can be maintained by a distributed algorithm using only the local connectivity information. Therefore, we have

$$C_{VB} = O(R^x) \ . \tag{2}$$

Next, we determine C_{query}. Since each routing zone around a VB node has R nodes, the virtual link between each pair of nearby VB nodes has length $O(\sqrt{R})$. Furthermore, assuming uniform node distribution and optimal VB node selection, the number of nodes in the VB is $\frac{N}{R}$, and the total number of virtual links is $O(\frac{N}{R})$. Obviously, at the arrival of a route request, no query cost is incurred if the destination node is within the source node's zone. If the destination node is not within the source node's zone, *in the worst case*, each route query is forwarded through all virtual links in the VB. Therefore, we have

$$\begin{aligned} C_{query} &= \left(1 - \frac{R}{N}\right) O\left(\lambda \left(\sqrt{R} \times \frac{N}{R}\right)^y\right) \\ &= \left(1 - \frac{R}{N}\right) O\left(\lambda N^y R^{-\frac{y}{2}}\right) \ . \end{aligned} \tag{3}$$

When $R << N$, we have

$$C_{query} = O\left(\lambda N^y R^{-\frac{y}{2}}\right) \ . \tag{4}$$

Thus, when the zone size is not too large, the aggregate routing cost of VBR can by approximated by

$$C_{VBR} = O(R^x) + O\left(\lambda N^y R^{-\frac{y}{2}}\right) \ . \tag{5}$$

From (5), we obtain that C_{VBR} is minimized when

$$R = O\left(\lambda^{\frac{2}{2x+y}} N^{\frac{2y}{2x+y}}\right) \ . \tag{6}$$

This suggests that (4) and (5) are valid when λ is not too large. In this case, the minimum C_{VBR} is

$$\min_R C_{VBR} = O\left(N^{\frac{2xy}{2x+y}}\right) \ . \tag{7}$$

Furthermore, it is clear that when λ is large, the optimal R should be large, such that the cost of VBR approaches that of the proactive routing protocol.

It is easy to see that, given any pair of proactive and reactive protocols, as long as $y < 2x$, we have $\frac{2xy}{2x+y} < x$ and $\frac{2xy}{2x+y} < y$. Therefore, by combining proactive routing and reactive routing into a hierarchy, VBR can achieve improved scalability compared with either a purely proactive protocol or a purely reactive protocol. As a point of reference, suppose WRP is used for local connectivity maintenance, and AODV or DSR are used for VB assisted route querying. From the reference [27], all of these protocols have routing cost $O(N)$ (i.e., $x = y = 1$). Combining them in VBR, we can achieve $C_{VBR} = O(N^{\frac{2}{3}})$.

The scalability of VBR comes at a price, however. As in any hierarchical protocols, implementing the generation and maintenance of the hierarchy requires extra processing at both the VB and non-VB nodes. Other disadvantages include sub-optimal routes and the potential occurrence of traffic hot-spots.

However, VBR is also able to alleviate these problems significantly. In Sections IV and V, we explain how the VB can be distributively maintained and employed to assist routing, so that the VB nodes are not overly burdened. In Section VI, we further study the length of data routes generated by VBR and show that it is near optimal. But first, in the next section, we present the network model assumptions.

[1]The values of x and y for some ad hoc routing protocols are given in [27]

III. Network Model

We consider an ad hoc network with nodes of equal range of transmission power and processing capability. Each node has a fixed transmission radius. We assume that the nodes find out about their neighbors through a *Neighborhood Discovery Protocol* (NDP), in which periodic, self-identifying 'HELLO' beacons are broadcast by each node's transmitter. Two nodes are neighbors of each other if they are within each other's transmission radius and hear each other's beacons. The routing control packets are assumed to be transmitted between nodes through unicast communication. It is also assumed that the underlying MAC layer protocol provides relatively reliable shared access to the unicast channel.

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. Although the proposed scheme can be easily modified to accommodate quality-of-service constraints, this is not of primary concern in this paper and is left for future work.

IV. Dynamic Virtual Backbone Maintenance

Since the communication environment in an ad hoc network changes rapidly, distribution and localization is the key to the establishment of the virtual backbone. As the network topology changes frequently, the graph of the virtual backbone should be updated based on only local information exchange. At the same time, the locally computed virtual backbone should be small (without redundancy), such that the interconnectivity of databases requires little network resources to maintain.

The DDCH is suitable for dynamic VB maintenance due to its distributive nature. In the rest of this section, we first describe the connectivity maintenance of a local routing zone, then introduce a centralized greedy algorithm for VB generation, and finally prove that the DDCH is a distributed equivalence of the centralized greedy algorithm.

A. Local Zone State and Connectivity Maintenance

The network nodes are partitioned into *database* and non-database nodes. The non-database nodes can be at any time in any one of the following three states: *panic*, *samaritan*, or *normal*. With an ideally functioning VB, 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 routing zone. 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 routing zone, if certain conditions are met.

A node monitors the link-state information and the state of all nodes within its local routing zone. For this purpose, we have chosen a simple link-state protocol as follows.[2] Each node periodically sends a *connectivity packet* to all of its neighbors, which, in turn, forward the packet to their neighbors, until the packet reaches all nodes within the sender's routing zone. The connectivity packets have the following format:

ID	seq_num	state	neigh_list	DB_num	hop

where `ID` is the identity number of the sending node, `seq_num` is a sequence number that increases each time a new connectivity packet is sent out by the node, `state` indicates the node's current state, `neigh_list` contains a list of the node's neighbors, `DB_num` is the number of databases within the node's routing zone (the use of which will become apparent in Section IV-E), and `hop` is a hop counter that is initiated to r and is decremented by 1 each time the packet is forwarded. A node receiving a packet with $\texttt{hop} = 1$ discards the packet. A node also discards a packet if it has already seen the packet, as indicated by the combination of ID and the sequence number fields.

B. The Centralized Greedy Algorithm for Minimum Set Covering

Given the network topology and a radius r, we would like to find a set of databases with minimum cardinality, such that every node in the network is at least in one database's r-hop zone. Namely, the VB "covers" the entire network. Thus, the computation of a VB can be reduced to the following Minimum Set Covering (MSC) problem:

Given a set of objects V (i.e., nodes) and a collection Q of sets of these objects (i.e., r-hop zones), find a subset C (i.e., r-hop zones induced by the VB) $\subseteq Q$ 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 [28] has suggested that a randomized greedy algorithm with redundancy elimination is the best known polynomial-time algorithm (producing the smallest covering set) in many general cases.

A greedy algorithm for MSC has the structure as follows:

$C = \phi$
while $\cup C \neq V$ and $Q \neq \phi$,
  $X = \arg\max_{Y \in Q}\{|Y|\}$
  $C = C + \{X\}$
  $Q = Q - \{X\}$
  $\forall Y \in Q,\ Y = Y - X$
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 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 [28].

[2]For clarity in illustrating the advantage of combining proactive routing and reactive routing in a hierarchy, we have chosen to implement this simplified protocol for local zone maintenance. Obviously, it is also possible to employ one of the many previously proposed, more sophisticated, protocols here. However, that is outside the scope of this paper.

In VBR, 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 VB, if the node itself becomes a database. Then, the greedy algorithm for minimum VB generation can be rewritten as:

$VB = \phi$
while VB does not cover all nodes,
    1. find node v with the max dependency number
    2. add v to VB
    3. re-compute dependency number
end

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 the graph topology. However, in an ad hoc network, centralized control is generally not desirable, due to the possible intermittent connectivity of nodes. Full network information exchange in an ad hoc network with large number of nodes and frequently changing topology is prohibitively expensive.

Next, we will show how this greedy algorithm can be implemented in a distributive fashion, using what we have termed the Distributed Database Coverage Heuristic (DDCH).

C. The Distributed Database Coverage Heuristic for VB Generation

Initially there is no database in the network, and nodes are not connected to any database. Therefore all nodes are in the panic state. A node in the panic or samaritan state periodically sends *state packets* to all nodes within $2r$ hops with the following format:

ID	seq_num	state	dep_num	hop

where `ID` is the identity number of the node, `seq_num` is a sequence number that increases each time a new state packet is sent out by 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 $\texttt{hop} = 1$ discards the packet. A node also discards a packet if it has already seen the packet, as indicated by the combination of the ID and the sequence number fields.

A node in the panic or samaritan state collects the state packets and extracts the dependency numbers from all other panic or samaritan nodes within $2r$ hops. If the node itself has the largest dependency number (with lexicographical tie-breaking), it becomes a database and joins the VB.

Otherwise, there are three possible actions by a panic node:

- If no new database appears in its routing zone within a time threshold, the node remains in panic, re-computes its dependency number, and sends out a new state packet.
- If a new database appears in its routing zone, and there is no panic node in the routing zone, the node returns to the normal state.
- If a new database appears in its routing zone and there are still panic nodes in the routing zone, the node changes its state to samaritan, re-computes its dependency number, and sends out a new state packet.

There are two possible actions by a samaritan node that does not have the maximum dependency number:

- If there is no panic node in its routing zone, the node returns to the normal state.
- If there are still panic nodes in its routing 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 becomes a database or finds a database within its r hops. Therefore, at the end, no panic node is left in the network, and the VB covers all nodes.

It can be shown that, given a graph representing the network topology and a zone radius r, DDCH generates a VB that has the same databases as one computed by the centralized greedy MSC (MSC_GR) algorithm. Therefore, through only local information exchange and local computation, DDCH computes the best known polynomial-time approximation to the minimum covering virtual backbone set. More precisely, we have the following:

Theorem: Given an ad hoc network represented by a graph $G(V, E)$, let B_D be the set of VB nodes selected by DDCH, and let B_G be the set of VB nodes selected by MSC_GR. Suppose the same dependency-number tie-breaking mechanism is used in both algorithms. Then, $B_D = B_G$.

Proof: Let B_1 be the set of VB nodes selected in the first round of DDCH. We first introduce two lemmas, whose proofs are given in the Appendix.

Lemma 1: $B_1 \subseteq B_G$.

Lemma 2: Furthermore, suppose all nodes in B_1 are first chosen as VB nodes. If we carry out MSC_GR on the set of panic and samaritan nodes in $(V - B_1)$ and obtain an additional set of VB nodes B_2, then $B_G = B_1 \cup B_2$.

These two lemmas imply that, at the completion of each round of VB selection by DDCH, if MSC_GR is applied for the remaining nodes, a same set of new VB nodes will be selected as if MSC_GR instead of DDCH was applied before the current round of VB selection. Thus, applying DDCH over $G(V, E)$ is equivalent to recursively replacing the MSC_GR operation in Lemma 2 by one round of DDCH until no panic node is left. Hence we have $B_D = B_G$.

■

D. Asynchronous Implementation of DDCH

Although we have described the above VB generation procedure in a synchronous fashion, it needs to be implemented asynchronously in an ad hoc network. In the asynchronous mode of operation, each panic or samaritan node sends out periodic state packets, independently of the other nodes. It also constantly collects the dependency numbers originated within its $2r$-hop zone. Immediately before each state packet is sent, it decides, based on the collected dependency numbers since the last time that it sent out a state packet, whether it will become a database or send out a new state packet.

Clearly, the asynchronous procedure still guarantees that the VB eventually covers all the nodes in the network, but

the size of the VB may not be as optimal as in MSC_GR. The asynchronous DDCH is implemented in the simulation described in Section VI.

Next we show how the DDCH can be employed in an dynamic scheme that updates the VB in order to adapt to the changing network topology.

E. Dynamic VB Structural Maintenance

There are two possible ways to update the VB as the network topology changes. One is to periodically regenerate the VB through DDCH over the entire network. The other is a dynamic scheme that updates the VB concurrently as the network topology changes. Obviously, the latter one is more preferable, since topology changes in faraway parts of the network should not affect the local databases. Furthermore, the frequency of changes may be time-dependent. Due to its distributive and local computation nature, the DDCH algorithm is applied to dynamically maintain the VB.

A database may move away from the neighborhood it previously occupied, or it may totally detach itself 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 respective routing 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.

The above process repeats as the network topology changes over time. In order to prevent over-sizing of the VB, databases are eliminated in regions where there are too many of them. This is accomplished by deleting redundant databases.

As shown in Section IV-A, each node writes `DB_num`, the number of DBs within its r hops, in the connectivity packet. Then the *redundancy* of a database is defined as the minimum `DB_num` among all nodes within the database's routing zone. A database is *redundant* if its redundancy is greater than or equal to 2.

A distributed procedure similar to DDCH is carried out to exchange the redundancy number among nearby databases, and to eliminate the redundant ones in a distributed-greedy manner. In particular, as shown in Section V-A, each database is made aware of the presence of all databases within $2r+1$ hops from itself through the *gateway* nodes. Furthermore, it can be shown that the redundancy number of a database is not affected by the generation or elimination of databases more than $2r+1$ hops away. Therefore, effective (in the greedy sense) redundant database elimination can be achieved by allowing a database to resign, if it has the maximum redundancy number (with lexicographical tie-breaking) among all databases within $2r+1$ hops.[3]

Database migration, database detachment, and database resignation have the same effect on the network nodes covered by the databases. When a database disappears from a region, the neighborhood nodes locally regenerate new a database to maintain connectivity to the virtual backbone. Given the zone radius r, database resignation and regeneration keep the size of the virtual backbone at a stable equilibrium over time. Our simulation results suggest that 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[29].

[3]In the actual implementation, a database resigns if its redundancy number has remained maximum for a threshold amount of time. In our simulation, that threshold is set to twice the connectivity-packet inter-arrival interval.

V. Routing with a Virtual Backbone

With a dynamically maintained VB in place, the routes between the source-destination node pairs can be determined by utilizing the location information stored in the databases. Although VB nodes serve as the coordinators in directing route queries and selecting routes, the routes do not necessarily need to pass through the VB. But before the VB can assist in establishing routes for other nodes, it needs to maintain the connectivity among its own members. In the next subsection, we present the VB connectivity maintaining procedure.

A. Dynamic VB Connectivity Maintenance

It is easy to show that, in a connected network with the zone radius r, a database can always find at least one other database within $2r+1$ hops away. Thus, if we define *database adjacency* as the state of being within $2r+1$ hops of distance away, the VB is inter-connected via the multi-hop virtual links between adjacent databases. Figure 1 in Section II shows an example of the virtual links in a network with $r=2$.

In this work, we extend the definition of *gateway* nodes used in the single-hop Linked Cluster Architecture in [19]. We define a node as a gateway from database DB_1 to database DB_2 if it is within r hops from DB_1 (*upstream DB*) and it is exactly $r+1$ hops from DB_2 (*downstream DB*).

Each time that a node r hops away from a DB receives the connectivity packet from the DB, it sends a *gateway notification packet* to all of its neighbors. The set of its neighbors that are not $r+1$ hops away from the DB ignores the packet, but those that are gateways update their record of the DB. Thus, a gateway node monitors the connectivity to its downstream databases, and reports any changes of downstream database membership to all upstream databases within r hops.

Therefore, through its routing zone link-state table and the gateway nodes, a database always has an up-to-date, shortest route to the databases within $2r+1$ hops. Then, any suitable routing protocol can be applied to maintain connectivity among all databases, treating the multi-hop virtual link between two adjacent databases as if it were a single-hop link.

Furthermore, due to the multi-hop nature of our zone definition, often there are multiple gateways between two adjacent databases. Therefore, multipath and alternate-path routing can be employed to achieve robust connectivity between databases and between source-destination pairs [30][31], alleviating routing congestion at the gateways. We will see in Section VI that the multiple gateways lead to selections of VBR-created routes that are near optimal.

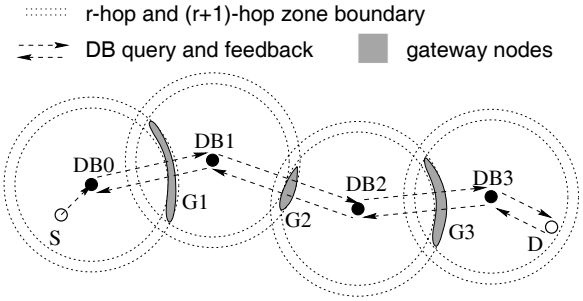

Fig. 2. Route query and feedback.

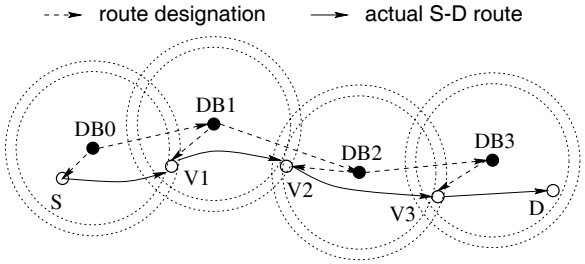

Fig. 3. Route designation.

Next, we show how the connectivity information contained in the VB can be utilized to facilitate routing among regular network nodes.

B. Virtual Backbone Assisted Route Determination

When a source node is within r hops from the destination node, it routes all data packets to the destination using its local link-state table. Otherwise, the VB routing is carried out through three rounds of control packet transmissions: *route query*, *route feedback*, and *route designation*.

For convenience of illustrating the advantage of combining proactive routing and reactive routing in a hierarchy, we first describe a simplified protocol for the reactive part of VB routing. Obviously, it is also possible to employ one of the many previously proposed techniques here, such as route caching[17] or expanding-ring search[4]. We describe a procedure to for route caching with VBR in Section V-C.

1) Route Query: In the route query step, we employ a VB flooding protocol as follows.

A message-initiating source node, S, sends a query for the location of the destination node, D, to a nearest database DB_0. DB_0 is then responsible for finding a route for S. If a distributed database-group location storage and query scheme is employed, such as those described in [32] and [33], DB_0 will then query a selected group of databases for the location of D. In this work, for simplicity of presentation, we assume that the location of D is known only to the databases within $r+1$ hops from it. In this case, if the destination is not in DB_0's routing zone, it forwards the query to all of its adjacent databases, which in turn will forward, over the virtual links, the query to the other databases in the entire network. The route query control packet has the following format:[4]

S	D	seq_num	DB_path

where DB_path is the accumulated path containing the identity of all databases that have forwarded the packet. As with the other control packets, the seq_num is used to limit the number of redundant transmissions.

As shown in Figure 2, suppose DB_m ($m = 3$ in this example) is the first database to receive the route query packet who has node D within its $r+1$ hops. Then in the query packet, the field DB_path contains DB_1, DB_2, ..., DB_{m-1}. If DB_m itself is not the destination node, DB_m queries D for its availability and readiness to receive the message from S. If D is available, it replies to DB_m, and the route feedback process starts. Otherwise, DB_m forwards the unavailability of D back to S, through the reverse of DB_path. Although not necessary, we assume that, in either case, D will no longer respond to the same query from other databases.

2) Route Feedback: Let G_i be the set of gateway nodes from DB_i to DB_{i-1}, where $i = 1, 2, \ldots, m$, as defined in Section V-A. Then, in the route feedback step, DB_m and each database in the reverse DB_path successively determines the best (shortest) route from G_i to D.

First, DB_m determines the shortest route from every node in G_m to D, using only links that it can see (i.e., links between nodes within r hops from DB_m and links between r-hop and $(r+1)$-hop nodes from DB_m). It then sends the length of these routes, one for each node in G_m, together with the identity of the respective starting nodes in G_m, to DB_{m-1}.

In turn, DB_{m-1}, knowing all possible routes from G_{m-1} to G_m with links that it knows about, determines the shortest route from every node in G_{m-1} to D, with the constraint that each route must go through one node in G_m. It then records this information for future use, and sends to DB_{m-2} the length of these G_{m-1}-to-D routes, one for each node in G_{m-1}.

This procedure continues until DB_0 receives the packet from DB_1 containing the constrained shortest route from every node in G_1 to D. Using the routes it knows about between S and G_1, DB_0 finds a node, V_1, in G_1, through which the S-to-D route is the shortest. It then sends the selected S-to-V_1 route to S, and passes along the identity of V_1 to DB_1. This starts the route designation process.

Figure 2 illustrates the processes of route query and route feedback with $m = 3$. Figure 3 shows an example of the chosen intermediate gateway nodes $\{V_1, V_2, V_3\}$, and the corresponding S-to-D route in the route designation process.

3) Route Designation: In the route designation step, the databases pass down segments of the selected constrained best route along DB_path. As illustrated in Figure 3, for each $i = 1, 2, \ldots, m-1$, DB_i receives the identity of the selected gateway node, V_i, from the previous database, DB_{i-1}. It then sends the identity of the next selected gateway node, V_{i+1}, to DB_{i+1}, and, at the same time, sends its selection of the constrained best V_i-to-V_{i+1} route to V_i. Finally, DB_m receives the identity of V_m, and then sends the constrained best V_m-to-D route to V_m.

In doing so, we have created a multi-stage route from S to D, which contains the intermediate gateway nodes V_1, V_2, ...,

[4]The Quality-of-Service requirements can be specified here, if necessary.

V_m. Each gateway node V_i along the chosen S-to-D route has a record of the route segment from itself to the next gateway (or D if $i = m$) and is responsible for forwarding the data packets toward their final destination.

Note that, although we have used minimum hop count as the measure of route optimality, the above routing method is not limited to this criterion. For example, each link can be assigned a value indicating its current capacity or delay, and these metrics could be used in routing with quality-of-service constraints.

C. Routing and VB Path Caching Enhancement to VBR

In an reactive routing protocol, and similarly in the reactive component of VBR, a newly discovered route could be cached, so that it may be reused the next time that the same route is requested[17]. A properly tuned route caching strategy can considerably reduce the time delay and traffic overhead of a routing protocol. The VBR routing framework allows two types of route caching: *direct route caching* and *VB path caching*.

Direct route caching in VBR is similar to route caching in many existing reactive routing protocols[5]. Two cases of such route caching exists. In the basic case, a source node caches routes so that a route is available when an application, running within the same node, demands it. We call this source route caching. As an extension to the above, some reactive routing protocols, such as AODV and DSR, allow an intermediate node (a non-destination node that has received a copy of the source node's route request) that has a cached route to the destination reply to the source with the cached route. We call this intermediate route caching. VBR can support both.

VB path caching is a unique feature of VBR. Whenever an VB assisted route query is successful, all VB nodes on the VB path that receives the route feedback stores the destination node's ID, its associated VB node, and the VB path that leads to that VB node. When a future route query to the destination node is sent to a VB node that contains the VB path cache to the destination's associated VB node, the store VB path cache can be used to direct the route query, instead of the usual VB flooding protocol.

In the route-caching enhanced version of VBR, a source node first tries to use its direct route cache to its destination node. If such cache does not exist, or if it is found to be invalid, it sends a route query to its nearest VB node. That VB node, in turn, tries to use its VB path cache to the destination node's associated VB node. If such cache does not exist, or if it is found to be invalid, the usual route querying procedure is performed via VB flooding.

VI. Performance Evaluation and Optimization

In this section, we evaluate the performance of VBR based on our discussion in Section II of its benefits and disadvantages. We first study the optimality of the data routes created by VBR, and show that they are near optimal in length. We then investigate the optimization of the zone radius r. We show that often the optimal zone radius is greater than one and, therefore, the variable-sized local routing zones of VBR is more optimal for ad hoc routing than the one-hop clusters used in many other hierarchical protocols. Finally, we compare the message overhead of VBR to that of purely proactive and purely reactive protocols, over various route request rates, to show the benefit of hybrid routing.

We use the OPNET™ simulator to evaluate the performance of VBR with DDCH in the asynchronous mode. Three sets of simulations are carried out with $N = 50$ nodes, $N = 100$ nodes, and $N = 300$ nodes. In each set of simulations, the radio transmission range of a node is defined by a circle of 230 meters in radius, and the system coverage area is defined as a square with side-length 1000 meters, 1450 meters, or 2500 meters, such that the average node degree is approximately 8 neighbors per node. For each N, VBR with different zone radii r is simulated.

Initially, all nodes are uniformly distributed within the coverage area. Each node moves with a constant velocity, with magnitude uniformly distributed between zero and 2 meters/second and direction uniformly distributed between zero and 2π. This can be viewed as a special case of the Gauss-Markov mobility model[34]. In order to eliminate the edge effect, the walls of the square coverage area are assumed to wrap around from left to right and from top to bottom, and the movement and transmission range of a node are adjusted accordingly.

Each node sends out a local connectivity packet every 1 second.[5] Each panic or samaritan node sends out a state packet every 5 seconds. Route requests are generated by a source node as a Poisson stream. The destinations are uniformly chosen among all nodes. For the 50-node case, the network is allowed to first "warm up" for 50 seconds in order to eliminate transient effects. Then 100 seconds of network operations are simulated, during which 5000 route requests are made. For both the 100-node and 300-node cases, the "warm-up" period is 20 seconds and then 20 more seconds are simulated, with 2000 route requests.

At the end of each VBR route query, a correct route is indicated by the successful relaying of a data packet from the source to the destination. If no correct route is found, possibly due to node movement or inaccurate topology information, the source node sends out the data packet by flooding the network (similarly to flooding the connectivity packets except without the r-hop limit), and the message overhead of this flooding is accounted for in the cost of VBR. We observe that, in our simulation with the above parameters, such flooding occurs in less than 5% of the VBR route queries.

For each set of network parameters, four simulation runs are carried out, and their average is shown for the analysis in Sections VI-A to VI-D.

A. DDCH Virtual Backbone Selection Optimality

VBR achieves topology abstraction by representing a large network by a small virtual backbone. In Figure 4, we plot the

[5]In reality, the frequency of local connectivity packets is a design parameter to be optimized based on the node mobility, required QoS, and other factors such as the underlying MAC protocol. These implementation considerations are outside the scope of this paper.

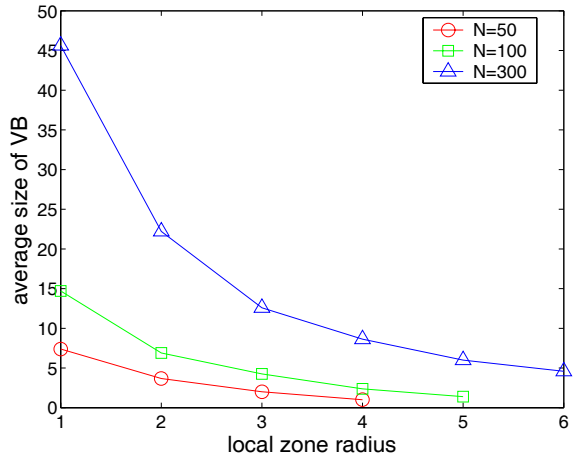

Fig. 4. Average VB size for different N and r.

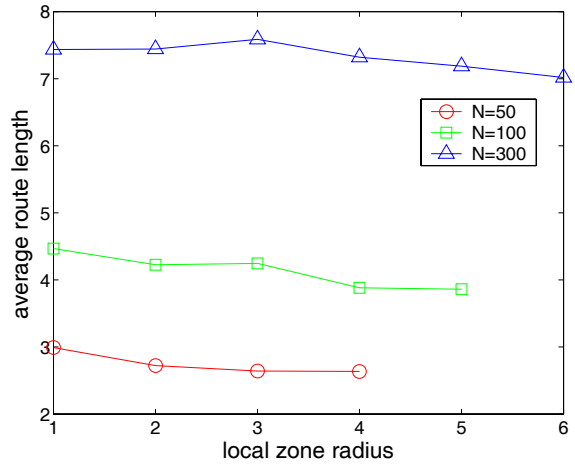

Fig. 5. Average route length for different N and r.

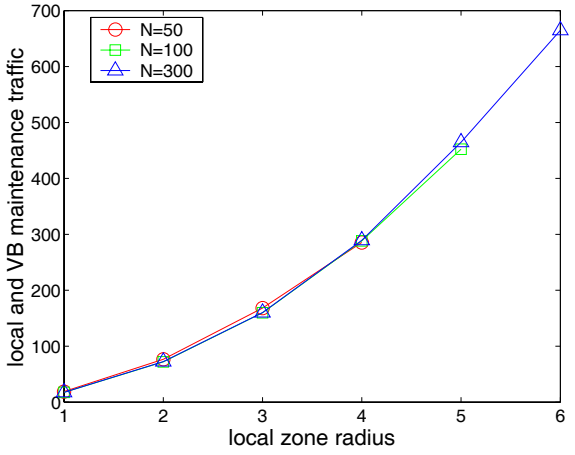

Fig. 6. Cost of local zone and VB maintenance vs. N and r.

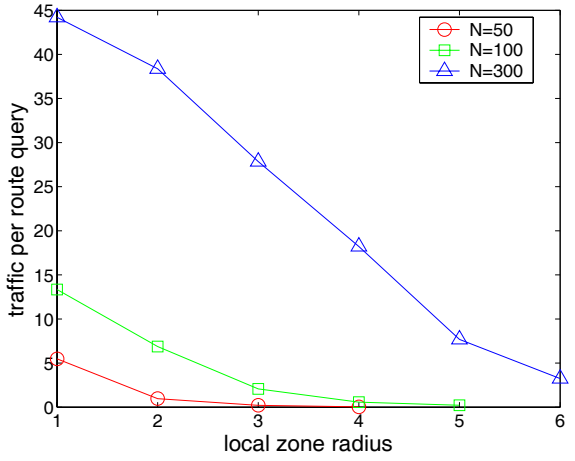

Fig. 7. Cost per route query vs. N and r.

average size of VBs generated and maintained by DDCH over the duration of the simulation runs. The zone radius varies from $r = 1$ to $r = 6$. Clearly, DDCH is very effective in controlling the size of the VB. For example, for a 300-node network, with zone radius 2, the VB size is kept around 22 on average, and with zone radius 6, a VB of size 5 can cover, and efficiently assist routing in, the entire network.

B. VBR Route Optimality

In Figure 5, we plot the average length of the VBR routes (without route caching) for networks with different N and r values. Note that, for N=50, 100, and 300, the average route length of *ideal* routing is 2.6 hops, 3.9 hops, and 6.8 hops, respectively. Therefore, these plots show that, for the networks under consideration, the VB assisted routing results in routes that are within 15% in length compared with the shortest ones. Furthermore, for many cases with larger values of r, the VBR route length is within 5% of the shortest distance. This suggests that the VB does provide sufficient directional guidance in routing. This also reflects the fact that the multiple gateways between the local routing zones around the databases provide sufficient alternate paths to choose from, which leads to near optimal route lengths.

Furthermore, these plots show that the route length is a decreasing function of the zone radius. Intuitively, this is so. As the zone radius increases, there are less intermediate database stages between the source and the destination. Therefore, it is not surprising that a VBR generated route approaches the optimal route as the zone radius increases.

Since the routes are near optimal for a wide range of network sizes and zone radii, in the following optimization of r, we assume that the procedure returns optimal (shortest) routes and we consider only the overhead of control messages.

C. Optimal Zone Radius Selection

As alluded to in Section II-B, the cost of VBR, in units of control packets per node per second, comprises the cost of proactivity, namely, the cost of local zone and VB maintenance, and the cost of reactivity, namely, the cost of route querying.

In Figure 6, we plot the proactive cost of VBR vs. N and r. Recall that, in this implementation of the VBR, $x = 1$ in Equations (1) and (2). Since the zone size $R = O(r^2)$, the proactive cost is a quadratic function of r, and it is independent of N. These plots help verify that conclusion.

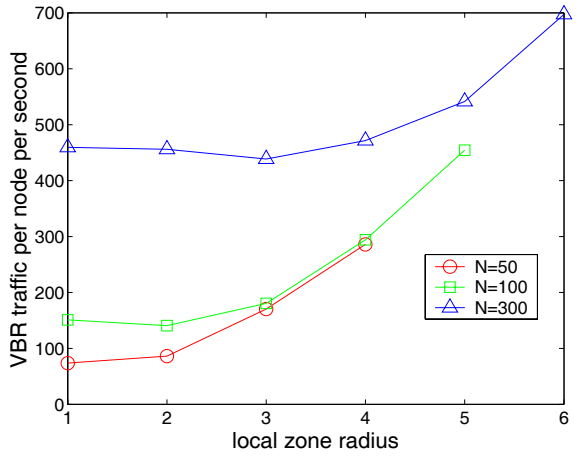

Fig. 8. Cost of VBR vs. N and r for $\lambda = 1$.

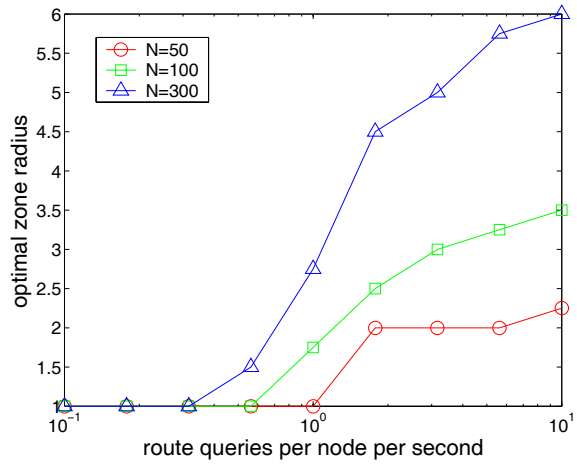

Fig. 9. Optimal zone radius vs. N and λ.

In Figure 7, we plot the reactive cost, per route query, of VBR without route caching vs. N and r. From Equation (4) with $y = 1$, the cost per route query is proportional to N and inversely proportional to r. These plots help verify that conclusion. Furthermore, we see here that the cost of route query quickly goes to zero as the zone radius increases. This is because no actual route query is necessary when the destination node is within the routing zone of the source node, which occurs with larger probability as the size of the zone increases.

From the above two figures, one can draw the conclusion that the zone radius is an important parameter that balances the trade-off between the cost of proactive local connectivity maintenance and the cost of reactive global connectivity information gathering. As an example, Figure 8 plots the total cost of VBR vs. N and r, for a route query rate $\lambda = 1$ per second. For this case, the minimum cost is achieved when r =1, 2, and 3, for N =50, 100, 300, respectively. As the network increases in size, VBR supports variable-sized local routing zones, and hence it is more scalable than other hierarchical routing protocols with fixed zone sizes.

Figure 9 presents the average of the optimal values of r vs. λ for different N values. These plots confirm our earlier conclusion in Section II-B that the optimal r is an increasing function of both N and λ.

D. Performance Advantage of Hybrid Hierarchical Routing

The trade-off between a reactive protocol and a proactive protocol is reflected by the amount of overhead in collecting network connectivity information *a prior* and the penalty of not having the information when a route is requested. VBR's selective topology representation in both the proactive and the reactive parts of a hybrid and hierarchical protocol can substantially reduce the amount of control message overhead. By adjusting the operating parameter r, one can optimize the trade-off between proactivity and reactivity in order to reduce the cost of routing.

In our example implementation of VBR, we have used a simple link-state protocol in the local zone routing, and a simple flooding mechanism in forwarding route queries over the VB. Therefore, it is justifiable to compare VBR with a simple link-state or flooding protocol. As suggested by Equation (7), even if a more efficient proactive protocol and a more efficient reactive protocol were considered, by employing them in local zone routing and route querying respectively, VBR would still continue to compare favorably relatively to either one of these protocols alone.

As such, we compare in more detail the cost of VBR, C^*_{VBR}, optimized over r, with that of simple flooding, C_{flood}, and simple link-state, C_{LS}, for different network sizes and route request rates. Here, the flooding protocol for routing in the entire network is similar to what is used in sending the VBR connectivity packets, except no hop limit is applied. In the link-state protocol, each node floods the entire network (using the aforementioned flooding protocol) with a list of its neighbors once every 1 second (same frequency as in VBR). As in VBR, we assume that a node resends the data packet via flooding if no correct route is found in the link-state protocol.

The cost of a purely reactive protocol linearly increases with λ. On the other hand, the cost of the purely proactive protocol is much less affected by λ. Therefore, the performance gain of VBR over flooding, C_{flood}/C^*_{VBR}, increases, and the performance gain of VBR over link-state, C_{LS}/C^*_{VBR}, decreases, as λ increases.

In Figure 10, we plot the performance gain of VBR (without route caching) vs. λ, for N =50, 100, and 300. These curves show that VBR out-performs both the proactive and reactive components that it employs, for a wide range of node route query rate. For example, even for a small network with $N = 100$, when $\lambda = 1$, using the VBR can reduce the cost of either flooding or link-state by 80%.

As alluded to earlier, the increment and decrement of the performance gain is due to the dominance of C_{flood} and C_{LS} for different ranges of λ. As λ becomes very small, there is no need to maintain link information even for a small neighborhood, and therefore, $r = 0$ gives the best performance. As λ becomes very large, it is worthwhile to constantly update the status of each link throughout the entire network, and therefore, large r ($r \to \infty$, in the limit) gives the best performance. Thus, VBR is designed to allow graceful

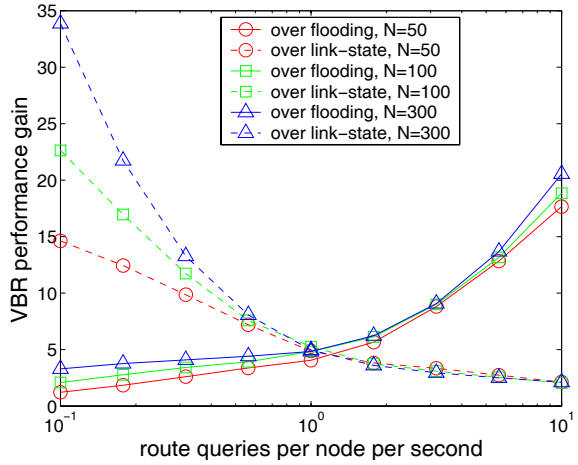

Fig. 10. Performance gain C_{flood}/C^*_{VBR} and C_{LS}/C^*_{VBR} vs. N and λ.

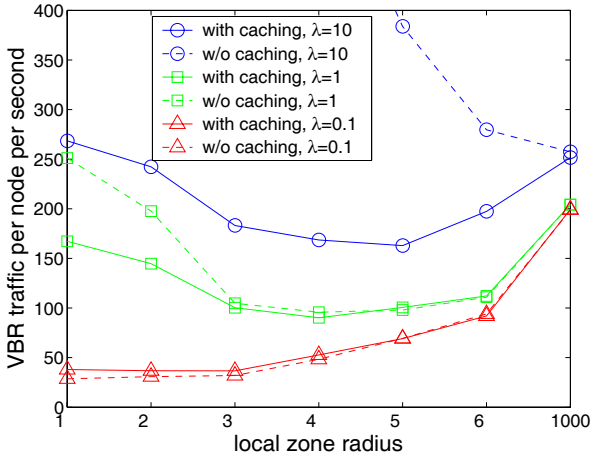

Fig. 11. Route caching enhancement. VBR with zone radius 1000 is equivalent to a pure link-state protocol.

degradation into either a reactive protocol or a proactive protocol, adaptive to the network operating parameters.

More importantly, comparing the right hand side of Equation (7) with $O(N^x)$ and $O(N^y)$, one observes that the performance gain of VBR increases as N increases. Figure 10 confirms this observation. This leads to the conclusion that the VBR is more scalable to larger networks.

E. Effect of Route Caching in VBR

Direct route caching and VB path caching can further improve the performance of VBR. In Figure 11, we plot the traffic overhead per node per second of a network with 200 nodes in a square of side-length 2050 meters. Data are collected for different zone radii and route request rates, comparing VBR with and without route caching. For proper comparison with route caching in some of the popular reactive routing protocols (e.g., AODV and DSR), a broadcast channel is assumed for each node. Since intermediate-node route caching does not significantly reduce traffic overhead[17], we have shown results with source-node route caching only, in addition to VB path caching.

These plots illustrate that route caching can significantly improve the performance of VBR, especially when the route request rate is large (e.g., $\lambda \geq 1$). But when the route request rate is very small (e.g., $\lambda \leq 0.1$), route caching can degrade the performance of routing. This matches with expectation and confirms the observations made with regard to route caching in general.

More interestingly, these plots also demonstrate how route caching can change the optimal zone radius. Since route caching is another means for combining proactivity with a reactive routing protocol, it reduces the optimal zone radius in general. As an example, when $\lambda = 10$, for VBR without caching, the optimal zone radius is larger than 6, but for VBR with caching, the optimal zone radius is 5. Note also that the curves are relatively flat around the optimal value of r. Therefore, the system can achieve near optimal performance even if the optimal r is not determined or set exactly.

In addition, Figure 11 shows that, even with route caching, the traditional protocols such as AODV and DSR can be improved by the VBR framework with an optimal zone radius. For example, when $\lambda = 1$, VBR with zone radius 3 has traffic overhead 100 packets per node per second, while a reactive protocol (i.e., $r = 0$), even with route caching, has traffic overhead more than 170 packets per node per second. This suggests that VBR can achieve a cost reduction of more than 40% over a reactive protocol with route caching, even for a network of moderate size.

VII. Related Work and Discussions on Combining Hierarchical and Zone Routing

VBR is designed based on the experiences of many previously proposed ad hoc routing protocols. It is inspired by the per-node zone concept proposed in ZRP. It also shares similarities with various forms of other hierarchical routing, as described in Section I, which have existed since the beginning of packet radio network research. Thus, the novelty and contribution of this work are the *combination* of hierarchical and zone routing into a simple, yet effective framework, and the introduction of a novel method for the dynamic and distributed maintenance of a *virtual backbone* that supports *variable-sized* local routing zones (the DDCH). We have also provided theoretic analysis results and the OPNET implementations that prove the efficiency of the proposed methods.

Section I has briefly summarized the main differences between VBR and the other routing protocols. Our theoretic and simulation results has further shown that the variable-sized zones give VBR the advantage over the other cluster-based hierarchical protocols, such as NTDR, CBRP, HSR, LCA, and CEDAR, which utilize what is equivalent to VBR's 1-hop routing zones. For example, in a network with 200 nodes and with the system parameters as shown in Figure 11, with moderate route request rate ($\lambda = 1$), VBR with zone radius 3 can reduce the routing overhead of the similar 1-hop hierarchical protocol by 40%.

VBR utilizes a local zone connectivity maintenance protocol that is similar to ZRP's. The difference between these two protocols lies in how they handle on-demand route queries

to faraway destinations. ZRP utilizes a Bordercast Resolution Protocol, where a source node multicasts route querying packets to selected peripheral nodes of its local routing zone, which, in turn, disseminate the route query in the same manner to their peripheral nodes. This achieves efficient routing over the *flat* network structure. VBR, on the other hand, relies upon the virtual backbone hierarchy to disseminate its route querying packets. Hence, VBR pays the up-front penalty of having to invest in the maintenance of the virtual backbone. However, by constraining the route querying packets within the virtual backbone, VBR reduces the amount of overhead in flooding the route querying packets, in addition to other query reduction mechanisms. Therefore, the VBR framework can be advantageous in networks with high route request rates.

VIII. Conclusions

In this paper, we have presented a novel Virtual Backbone Routing framework for ad hoc networks, which combines local proactive routing and global reactive routing over a two-level hierarchy. The efficiency of this protocol is achieved by applying selective topology representation in both the *zone-limited* link information maintenance and the *VB-constrained* route queries. Through mathematical analysis and network simulation, we have shown that VBR is efficient and scalable to large networks.

The local routing zones of VBR are variable-sized and adaptable to the network operating parameters. This is shown to lead to a higher degree of scalability than purely proactive or reactive protocols, or hierarchical protocols with one-hop clusters. In addition, VBR can be further enhanced by direct route caching and VB path caching, incurring less routing traffic compared with the traditional reactive protocols with route caching.

Unlike other hierarchical routing protocols, the data routes of VBR do not necessarily go through the VB. Therefore congestion is reduced. Furthermore, the length of VBR-generated routes are shown to be generally within 5% of the optimal value. Finally, VBR provides proper routing service even in the event of network partitioning.

Given a required zone radius, the proposed distributed VB generation algorithm, DDCH, computes the same VB set as the centralized greedy algorithm, which is the best known polynomial-time approximation to the minimum covering VB set. Due to the local computation nature of DDCH in dynamic VB maintenance and the reactive nature of the global route establishment process, VBR's adaptation to topological changes is fast and efficient. Under extreme conditions, it also allows graceful degradation into either a purely proactive or purely reactive protocol.

VBR provides a dynamic framework for the efficient combination of proactive and reactive routing protocols. In this paper, we have used as examples a simple link-state protocol for local route maintenance and a simple flooding protocol over the VB for global route query, and have correspondingly given performance comparisons against these two protocols. As shown in Section II-B, the efficiency of VBR is closely tied with the efficiency of the underlying local zone routing and global route querying protocols. Therefore, VBR can achieve even higher degrees of scalability by allowing the dynamic combination of proactive and reactive protocols that are proven to be robust and efficient.

Appendix

Given an ad hoc network represented by a graph $G(V, E)$, let B_D be the set of VB nodes selected by DDCH, and let B_G be the set of VB nodes selected by MSC_GR. Suppose the same dependency-number tie-breaking mechanism is used in both algorithms. The following two lemmas are used in Section IV to prove that $B_D = B_G$.

Lemma 1: Let B_1 be the set of VB nodes selected in the first round of DDCH. Then $B_1 \subseteq B_G$.

Proof: Let $B_G = \{b_1, b_2, \ldots, b_{|B_G|}\}$, where b_i, $1 \leq i \leq |B_G|$, is the VB node selected by MSC_GR at the i-th step. Let $D(v, i)$ be the dependency number of node v at the beginning of the i-th step of MSC_GR.[6] Then, by the construction of MSC_GR,

$$D(b_i, i) = \max_{v \in V} D(v, i) \ . \tag{8}$$

For each $v \in B_1$, we use induction to show that,

$$D(v, i) = D(v, 1) \ , \quad \text{for } 1 \leq i \leq i_{VB}(v) \ , \tag{9}$$

where $i_{VB}(v)$ is the step in MSC_GR when v becomes either a VB node or a normal node and hence no longer takes part in the generation of new VB nodes. It will be clear later that, at the $i_{VB}(v)$-th step, v can only become a VB node.

The initial step of the induction is trivially true. Further more, for each $v \in B_1$, v is chosen in the first round of DDCH because it has the highest dependency number among all nodes within $2r$ hops before any VB node selection. This still holds in MSC_GR. Namely,

$$D(v, 1) = \max_{u \in Z(v, 2r)} D(u, 1) \ , \tag{10}$$

where $Z(v, x)$ denotes the set of nodes within x hops from v. Suppose, at the beginning of the k-th step of MSC_GR, $D(v, k) = D(v, 1)$. Then, since

$$D(u, i) \geq D(u, j) \ , \quad \text{for all } u \text{ and } i > j \ , \tag{11}$$

we have

$$D(v, k) = \max_{u \in Z(v, 2r)} D(u, k) \ . \tag{12}$$

If $D(v, k) = \max_{u \in V} D(u, k)$, then from Eqn. (8), $b_k = v$, and $i_{VB}(v) = k$. Otherwise, $D(b_k, k) > D(v, k)$. Then, from Eqn. (12), $b_k \notin Z(v, 2r)$, in which case, the selection of b_k into the VB does not affect the dependency number of v. Hence, we have

$$D(v, k+1) = D(v, k) = D(v, 1) \ . \tag{13}$$

This completes the induction.

Therefore, during the entire duration that MSC_GR is executed, no node within $Z(v, 2r)$ has dependency number greater

[6] All dependency numbers are assumed to reflect the proper tie-breaking mechanism (e.g., lexicographical), such that $D(u, i) \neq D(v, i)$ for all $u \neq v$, and i.

than v, and hence, no node within $Z(v,2r)$ becomes a VB node before v does. Since, after $|B_G|$ steps of MSC_GR, there is no more panic node in the network, v must be a VB node. Thus, $B_1 \subseteq B_G$.

■

Lemma 2: Suppose all nodes in B_1 are first chosen as VB nodes. If we carry out MSC_GR on the set of panic and samaritan nodes in $(V - B_1)$ and obtain an additional set of VB nodes B_2, then $B_G = B_1 \cup B_2$.

Proof: Since $B_1 \subseteq B_G$, let $B_1 = \{b_{h_1}, b_{h_2}, \ldots, b_{h_{|B_1|}}\}$, where, without loss of generality, $1 \leq h_1 < h_2 < \cdots < h_{|B_1|} \leq |B_G|$. Thus,

$$D(b_{h_1}, h_1) > D(b_{h_2}, h_2) > \cdots > D(b_{h_{|B_1|}}, h_{|B_1|}) \ . \quad (14)$$

From (9), it follows that

$$D(b_{h_1}, 1) > D(b_{h_2}, 1) > \cdots > D(b_{h_{|B_1|}}, 1) \ . \quad (15)$$

Let $B_2 = \{b'_1, b'_2, \ldots, b'_{|B_2|}\}$, where b'_i, $1 \leq i \leq |B_2|$, is the VB node selected by MSC_GR over $(V - B_1)$ at the i-th step. We use induction to prove that, for $1 \leq i \leq |B_G|$,

$$\{b_1, b_2, \ldots, b_i\} = \{b_{h_1}, b_{h_2}, \ldots, b_{h_\zeta}\} \cup \{b'_1, b'_2, \ldots, b'_\eta\} \ , \quad (16)$$

where $i = \zeta + \eta$.

Obviously, $D(b_1, 1) = \max_{v \in V} D(v, 1) = \max_{v \in Z(b_1, 2r)} D(v, 1)$. Therefore, $b_1 \in B_1$. Furthermore, from Eqn. (15), we have $b_1 = b_{h_1}$. Suppose, after the k-th step of MSC_GR over V,

$$\{b_1, b_2, \ldots, b_k\} = \{b_{h_1}, b_{h_2}, \ldots, b_{h_\alpha}\} \cup \{b'_1, b'_2, \ldots, b'_\beta\} \ , \quad (17)$$

where $k = \alpha + \beta$. At this time, if $D(b_{h_{\alpha+1}}, k+1) = \max_{v \in V} D(v, k+1)$, then $b_{k+1} = b_{h_{\alpha+1}}$. Otherwise, we have $D(b_{k+1}, k+1) > D(b_{h_{\alpha+1}}, k+1)$ and $h_{\alpha+1} > k+1$. Therefore, from (9), $D(b_{k+1}, k+1) > D(b_{h_{\alpha+1}}, 1)$. Combining this with (11) and (15), we have

$$D(b_{k+1}, 1) > D(b_{h_i}, 1) \ , \quad \text{for } \alpha + 1 \leq i \leq |B_1| \ . \quad (18)$$

Then, from (10),

$$b_{k+1} \notin \bigcup_{\alpha+1 \leq i \leq |B_1|} Z(b_{h_i}, 2r) \ . \quad (19)$$

In the generation of B_2, at the beginning of the $(\beta+1)$-th step of MSC_GR over $(V - B_1)$, the VB consists of the following nodes:

$$B_1 \cup \{b'_1, b'_2, \ldots, b'_\beta\} = \{b_1, b_2, \ldots, b_k\} \cup \{b_{h_{\alpha+1}}, b_{h_{\alpha+2}}, b_{h_{|B_1|}}\} \ . \quad (20)$$

Let $D'(v, i)$ be the dependency number of node v at the beginning of the i-th step of MSC_GR over $(V - B_1)$. From (19), the selection of b_{h_i}, $\alpha + 1 \leq i \leq |B_1|$, into the VB does not affect the dependency number of b_{k+1}. Therefore,

$$D'(b_{k+1}, \beta + 1) = D(b_{k+1}, k+1) \ . \quad (21)$$

Since $D(b_{k+1}, k+1) = \max_{v \in V} D(v, k+1)$, and the addition of b_{h_i}, $\alpha + 1 \leq i \leq |B_1|$, to the VB only decreases or does not affect the dependency number of any node in the network, we have

$$D'(b_{k+1}, \beta + 1) = \max_{v \in V} D'(v, \beta + 1) \ , \quad (22)$$

and hence $b'_{\beta+1} = b_{k+1}$.

Therefore, after the $(k+1)$-th step of MSC_GR over V,

$$\{b_1, b_2, \ldots, b_k, b_{k+1}\} = \{b_{h_1}, b_{h_2}, \ldots, b_{h_\alpha}\} \cup \{b'_1, b'_2, \ldots, b'_\beta\} \cup \{b_{h_{\alpha+1}} \quad \text{or} \quad b'_{\beta+1}\} \ . \quad (23)$$

This completes the induction.

Eqn. (16) suggests that, after $|B_G| - |B_1|$ steps of MSC_GR over $(V - B_1)$,

$$B_G = B_1 \cup \{b'_1, b'_2, \ldots, b'_{|B_G| - |B_1|}\} \ . \quad (24)$$

Since there is no panic node left in the network, MSC_GR terminates at $|B_2| = |B_G| - |B_1|$. Therefore, $B_G = B_1 \cup B_2$.

■

REFERENCES

[1] J. J. Garcia-Luna-Aceves *et al.*, "Analysis of routing strategies for packet radio networks," in *Proc. of IEEE INFOCOM*, March 1985.

[2] B. M. Leiner, D. L. Nielson, and F. A. Tobagi, "Issues in packet radio network design," *Proc. IEEE*, vol. 75, no. 1, pp. 6–20, January 1987.

[3] J. Jubin and J. D. Tornow, "The DARPA packet radio network protocols," *Proc. IEEE*, vol. 75, no. 1, pp. 21–32, January 1987.

[4] C. E. Perkins, Ed., *Ad Hoc Networking*. Addison-Wesley Longman, 2001.

[5] Z. J. Haas, J. Deng, B. Liang, P. Papadimitratos, and S. Sajama, "Wireless Ad Hoc Networks," in *Wiley Encyclopedia of Telecommunications*, J. G. Proakis, Ed. John Wiley & Sons, 2002.

[6] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," in *ACM SIGCOMM: Computer Communications Review*, vol. 24, no. 4, October 1994, pp. 234–244.

[7] A. Iwata, C.-C. Chiang, G. Pei, M. Gerla, and T.-W. Chen, "Scalable routing strategies for ad hoc wireless networks," *IEEE J. Select. Areas Commun.*, vol. 17, no. 8, pp. 1369–1379, August 1999.

[8] P. Jacquet, P. Muhlethaler, A. Qayyum, A. Lanouiti, L. Viennot, and T. Clausen, "Internet draft," draft-ietf-manet-olsr-02.txt, July 2000.

[9] J. J. Garcia-Luna-Aceves and M. Spohn, "Source-tree routing in wireless networks," in *Proc. of IEEE ICNP*, October 1999.

[10] B. Bellur and R. G. Ogier, "A reliable, efficient topology broadcast protocol for dynamic networks," in *Proc. of IEEE INFOCOM*, March 1999.

[11] S. Murthy and J. J. Garcia-Luna-Aceves, "An efficient routing protocol for wireless networks," *ACM Mobile Networks and Applications Journal*, October 1996.

[12] C.-K. Toh, *Wireless ATM and Ad Hoc Networks: Protocols and Architectures*. Kluwer Academic Press, 1997.

[13] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Proc. of IEEE Workshop on Mobile Computing Systems and Applications*, February 1999, pp. 90–100.

[14] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, T. Imielinski and H. Korth, Eds. Kluwer Academic Publishers, 1996, ch. 5, pp. 153–181.

[15] V. Park and M. S. Corson, "Internet draft," draft-ietf-manet-tora-spec-03.txt, November 2000.

[16] Z. J. Haas and M. R. Pearlman, "Providing ad-hoc connectivity with the reconfigurable wireless networks," in *Proc. of ACM SIGCOMM*, September 1998.

[17] B. Liang and Z. J. Haas, "Optimizing route-cache lifetime in ad hoc networks," in *Proc. IEEE INFOCOM*, Apr. 2003, pp. 281 – 291.

[18] N. Shacham and J. Westcott, "Future directions in packet radio architectures and protocols," *Proc. IEEE*, vol. 75, no. 1, pp. 83–99, January 1987.

[19] D. J. Baker, A. Ephremides, and J. A. Flynn, "The architectural organization of a mobile radio network with distributed control," *IEEE J. Select. Areas Commun.*, vol. SAC-2, pp. 226–237, January 1984.

[20] M. Jiang, J. Li, and Y. C. Tay, "Internet draft," draft-ietf-manet-cbrp-spec-01.txt, August 1999.

[21] B. Das and V. Bharghavan, "Routing in Ad-Hoc Networks Using Minimum Connected Dominating Sets," in *Proc. of IEEE ICC*, June 1997.

[22] R. Sivakumar, P. Sinha, and V. Bharghavan, "CEDAR: A core-extraction distributed ad hoc routing algorithm," *IEEE J. Select. Areas Commun.*, vol. 17, no. 8, pp. 1454–1465, August 1999.
[23] G. Pei, M. Gerla, and X. Hong, "LANMAR: Landmark routing for large scale wireless ad hoc networks with group mobility," in *Proc. of IEEE/ACM (MobiHOC)*, August 2000.
[24] R. Ramanathan and M. Steenstrup, "Hierarchically-organized, multihop mobile wireless networks for quality-of-service support," *ACM/Baltzer Mobile Networks and Applications*, vol. 3, no. 1, pp. 101–119, 1998.
[25] R. Ruppe and S. Griswald, "Near term digital radio (ntdr) system," in *Proc. of IEEE MILCOM*, vol. 3, November 1997, pp. 1282–1287.
[26] M. Joa-Ng and I.-T. Lu, "A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks," *IEEE J. Select. Areas Commun.*, vol. 17, no. 8, pp. 1415–1425, August 1999.
[27] E. M. Royer and C.-K. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," *IEEE Personal Commun. Mag.*, pp. 46–55, April 1999.
[28] 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.
[29] B. Liang and Z. J. Haas, "Virtual backbone generation and maintenance in ad hoc network mobility management," in *Proc. IEEE INFOCOM*, March 2000, pp. 1293 – 1302.
[30] M. R. Pearlman, Z. J. Haas, P. Scholander, and S. S. Tabrizi, "On the impact of alternate path routing for load balancing in mobile ad hoc networks," in *ACM MobiHOC*, Aug. 2000.
[31] A. Tsirigos and Z. J. Haas, "Analysis of multipath routing - Part I: The effect on the packet delivery ratio," *IEEE Trans. Wireless Commun.*, vol. 3, no. 1, pp. 138–146, Jan. 2004.
[32] Z. J. Haas and B. Liang, "Ad hoc mobility management with uniform quorum systems," *IEEE/ACM Trans. Networking*, vol. 7, no. 2, pp. 228–240, April 1999.
[33] ——, "Ad hoc mobility management with randomized database groups," in *Proc. of IEEE ICC*, June 1999.
[34] B. Liang and Z. J. Haas, "Predictive distance-based mobility management for multi-dimensional PCS networks," *IEEE/ACM Trans. Networking*, vol. 11, no. 5, pp. 718 – 732, Oct. 2003.

Ben Liang received honors simultaneous B.Sc. (valedictorian) and M.Sc. degrees in electrical engineering from Polytechnic University in Brooklyn, New York, in 1997 and the Ph.D. degree in electrical engineering with computer science minor from Cornell University in Ithaca, New York, in 2001. In the 2001 academic year, he was a visiting lecturer and post-doctoral research associate at Cornell University. He joined the Department of Electrical and Computer Engineering at the University of Toronto as an assistant professor in 2002. His current research interests are in mobile networking and wireless multimedia systems. He won an Intel Foundation Graduate Fellowship in 2000 to complete his dissertation on wireless network mobility management and the Best Paper Award at the IFIP Networking conference in 2005 for his work on adaptive vertical handoff in heterogeneous wireless systems. He is a member of Tau Beta Pi, IEEE, and ACM and serves on the organization and technical program committees of a number of major conferences each year.

Zygmunt J. Haas received his B.Sc. in EE in 1979 and M.Sc. in EE in 1985. In 1988, after earning his Ph.D. from Stanford University, he joined the AT&T Bell Laboratories in the Network Research Department. There he pursued research on wireless communications, mobility management, fast protocols, optical networks, and optical switching. From September 1994 till July 1995, Dr. Haas worked for the AT&T Wireless Center of Excellence, where he investigated various aspects of wireless and mobile network technologies. In August 1995, he joined the faculty of the School of Electrical and Computer Engineering at Cornell University, where he is now a Professor and the Associate Director for Academic Affairs.

Dr. Haas is an author of numerous technical conference and journal papers and holds fifteen patents in the areas of high-speed networking, wireless networks, and optical switching. He has organized several workshops, delivered numerous tutorials at major IEEE and ACM conferences, and serves as editor of several journals and magazines, including the IEEE Transactions on Networking, the IEEE Transactions on Wireless Communications, the IEEE Communications Magazine, and the ACM/Kluwer Wireless Networks journal. He has been a guest editor of IEEE JSAC issues on "Gigabit Networks," "Mobile Computing Networks," and "Ad-Hoc Networks." Dr. Haas served as a Chair of the IEEE Technical Committee on Personal Communications and is currently serving as the Chair of the Steering Committee of the IEEE Pervasive Computing magazine. His interests include: mobile and wireless communication and networks, performance evaluation of large and complex systems, and biologically inspired networks. His e-mail is: haas@ece.cornell.edu and his URL is: http://wnl.ece.cornell.edu .