

# Design and Simulation of a Distributed Dynamic Clustering Algorithm for Multimode Routing in Wireless Ad Hoc Networks

**A. Bruce McDonald**

Northeastern University  
Department of Electrical and Computer Engineering  
Boston, MA 02115  
*mcdonald@ece.neu.edu*

**Taieb F. Znati**

University of Pittsburgh  
Department of Computer Science  
Pittsburgh, PA 15260

This article presents analysis of the design and simulation of a new class of environmentally aware clustering algorithms for wireless ad hoc networks. The clustering algorithm establishes an adaptive self-organizing arrangement that enables multimode routing strategies. Multimode operation is needed to enhance scalability and robustness. In the present case, clustering is based on a criterion that enforces an upper bound on the probability of path failure within a cluster over time. The result is an adaptive hybrid routing strategy that dynamically balances routing overhead against routing optimality. The clustering algorithm unifies the routing strategies according to localized and time-varying mobility characteristics. A simulation model was developed to validate the effectiveness of the clustering algorithm and demonstrate the multimode routing behavior. Results show that the algorithm adapts effectively to node mobility and achieves relatively consistent overhead regardless of network size and mobility.

**Keywords:** Simulation, ad hoc networks, clustering, routing, mobility

## 1. Introduction

A wireless ad hoc network is a self-organizing collection of *user* nodes that must cooperate to provide basic networking functionality. In general, the nodes of an ad hoc network are mobile and rely entirely on wireless transmission without fixed infrastructure or dedicated communications devices. Consequently, packet-switched routing is required to manage limited device power, manage unpredictable variation in channel quality, and reduce media access contention. Hence, an ad hoc network effectively consists of a set of mobile wireless routers. As such, the user nodes must participate in an adaptive routing algorithm that is responsive enough to meet application requirements without overusing limited resources. Furthermore, because each user may be an active router, the routing algorithm must scale to the total number of *users*.

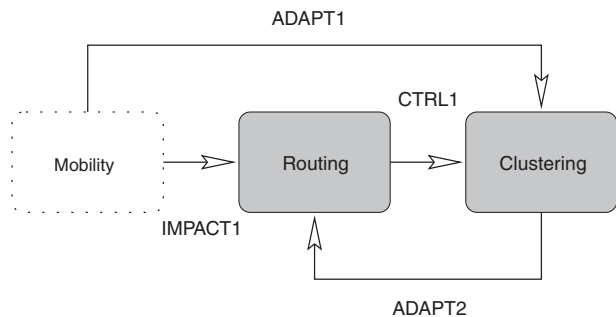
**Submission Date:** xxx

**Accepted Date:** xxx

*SIMULATION*, Vol. 78, Issue 7, July 2002 xxx-xxx  
©2002 The Society for Modeling and Simulation International

The purpose of this article is to present an analysis of the design and performance characteristics of an innovative new class of clustering algorithms first proposed in McDonald and Znati [1]. The objective is to improve routing scalability by supporting a multimode strategy that divides routing into two components: intracluster and intercluster. The clustering algorithm adapts to node mobility and the time-varying characteristics of wireless channels using criteria that establish an upper bound on the probability of path failure between a cluster head and the nodes of the cluster over time. Highly mobile nodes form smaller clusters, but nodes that are less mobile with respect to each other associate with larger clusters over time. Hence, the relative contribution of each routing component is dynamically adjusted to balance routing overhead and maintain adequate route quality and responsiveness.

In contrast to other clustering strategies, there are no inherent constraints on the maximum number of hops or the positioning of nodes within a cluster. Nodes need not retain their relative position to prevent reclustering or reconstruction of existing intercluster routes. Clusters are dynamically created, expanded, reorganized internally,



**Figure 1.** Organization of framework elements

contracted, partitioned, and terminated based on the relative positions of the nodes, velocity, and a mobility model used to predict the probability of link and path survival over time. The objective is to support a unified multimode routing strategy that is scalable and robust with respect to mobility. In this article, the criteria for shifting the routing mode are based on node mobility; however, given valid models, it is readily generalized to include other critical factors (e.g., available power or spare processing capacity).

Figure 1 depicts the core elements of the clustering framework and how they are related. Mobility drives the dynamic clustering as modulated by the routing algorithm—mobility also forces the routing algorithm response. This is achieved using prediction of the future state of the network links to provide a quantitative bound on the availability of paths to cluster destinations [2, 3]. The increasing routing overhead due to mobility is then balanced by the clustering algorithm, which provides a logical topology for routing. Complete development of a mobility-based clustering metric, including model validation and performance analysis with discrete event simulation, appears in McDonald and Znati [4]. The remainder of this article presumes the use of this metric.

The remainder of this paper is organized as follows: related work and the core elements of the clustering framework are characterized in Section 2. The clustering algorithm is presented in Section 3, which describes the set of states and events that drive algorithm actions. Simulation is used to validate the hypotheses regarding the adaptive clustering algorithm and the multimode routing behavior with its impact on routing overhead. The simulation model and analysis are presented in Section 4. Discussion includes selection of experimental factors and parameter values, as well as experimental results used to analyze the performance of the clustering algorithm. Results show that the algorithm adapts effectively to node mobility and achieves relatively consistent overhead regardless of network size and mobility. Finally, conclusions are presented in Section 5.

## 2. Dynamic Clustering and Multimode Routing

The clustering and hybrid routing strategy analyzed in this paper adopts a dynamic clustering algorithm similar to the one described in McDonald and Znati [5] for dynamically maintaining the cluster topology. The distributed dynamic cluster algorithm (DDCA) builds on concepts first developed in McDonald and Znati [1]. The idea is to dynamically partition the network into nonoverlapping clusters of nodes consisting of one *parent* with zero or more *children*. The affiliation of a node with a cluster depends on meeting a lower bound on the probability of path survival between that node and the parent node of the cluster [2, 3, 6].

Cluster formation in DDCA uses mobility-based criteria. To join a cluster, a node must be able to reach the parent node of the cluster along a cluster-internal path (Definition 2.3) that is expected to survive for a period of time  $t$  with a probability of at least  $\alpha$ . Such a cluster will be referred to as an  $(\alpha, t)$ -cluster. The scheme proposed in Basagni [7] also uses a mobility-based criterion, but it builds clusters that consist of a set of nodes that are adjacent to a cluster head. An  $(\alpha, t)$ -cluster is a multiple-hop subnet in which the maximum number of hops between any pair of nodes in the same cluster varies dynamically depending on the mobility characteristics of the nodes. Hence, the size and structure of each cluster are adapted continuously to the environment. The next subsection presents characterization of the  $(\alpha, t)$ -cluster as adapted from McDonald and Znati [1, 5].

### 2.1 $(\alpha, t)$ -Cluster Characterization

An  $(\alpha, t)$ -cluster is a dynamically organized set of nodes that are connected over paths that are internal to the members of the cluster. Each cluster contains one leader or parent node and zero or more children. The basic idea is that a node without a cluster affiliation can join a cluster subject to the requirement that it can establish a path to the parent of the cluster that meets a lower bound on the probability of availability over a specified interval of time. Nodes within the same cluster proactively maintain paths to all the nodes in the cluster and to the external nodes that border nodes in the cluster—that is, the nodes not affiliated with the cluster that are adjacent to one or more nodes in the cluster. Each node must belong to one and only one cluster. Once joining a cluster, the node remains in the cluster so long as it maintains a cluster-internal path to the parent node. The following definitions are required to formally characterize the  $(\alpha, t)$ -cluster and specify the clustering algorithm:

**DEFINITION 2.1:** *Path availability.* Let  $\mathcal{P}_{i,j}^k(t)$  indicate the state of path  $k$  between node  $i$  to node  $j$  at time  $t$ .  $\mathcal{P}_{i,j}^k(t) = 1$  if all the links in the path are active at time  $t$ , and  $\mathcal{P}_{i,j}^k(t) = 0$  if at least one link in the path is inactive at time  $t$ . The *availability* of path  $k$  at time  $t$ ,  $\Pi_{i,j}^k(t)$  is defined as follows:

$$\Pi_{i,j}^k(t) \equiv Pr(\mathcal{P}_{i,j}^k(t) = 1). \quad (1)$$

DEFINITION 2.2:  $(\alpha, t)$ -path. Let  $\mathcal{P}_{i,j}^k(\tau)$  indicate the state of path  $k$  between node  $i$  and node  $j$  at time  $\tau$ , and let  $\Pi_{i,j}^k(\tau+t)$  be its availability at time  $\tau+t$ . Path  $k$  is defined as an  $(\alpha, t)$ -path iff the following two conditions hold:

$$\mathcal{P}_{i,j}^k(\tau) = 1, \quad (2)$$

$$\Pi_{i,j}^k(\tau+t) \geq \alpha. \quad (3)$$

DEFINITION 2.3: *Internal path*. Let  $k$  be a path from node  $i$  to node  $j$ , and let  $\mathcal{N}^k$  be the set of nodes that lie along that path. Path  $k$  is defined as an *internal path* with respect to a set of nodes,  $\mathcal{S}$ , if  $\mathcal{N}^k \subseteq \mathcal{S}$ .

DEFINITION 2.4:  $(\alpha, t)$ -availability. Two nodes  $i$  and  $j$  are defined as being  $(\alpha, t)$ -available at time  $\tau'$  if there exists an  $(\alpha, t)$ -path,  $k$ , between them at some time  $\tau \leq \tau'$  and if there is at least one active path between them at all times during the interval  $(\tau, \tau')$ :

DEFINITION 2.5:  $(\alpha, t)$ -cluster. Let  $p^1$  be a node and  $\mathcal{S}$  be a set of nodes: an  $(\alpha, t)$ -cluster is defined as the set of nodes  $\mathcal{C} = \mathcal{S} \cup p$ , such that for each  $n \in \mathcal{S}$ ,  $n$  and  $p$  are  $(\alpha, t)$ -available along an internal path with respect to  $\mathcal{C}$ , and  $\mathcal{C}$  adheres to Properties 2.1 to 2.3.

The basic idea of the  $(\alpha, t)$ -cluster algorithm is to dynamically partition an ad hoc network into clusters that meet the stability criteria as given in Definition 2.5. Namely, it is desirable for a node to remain affiliated with its cluster sufficiently long enough to establish and maintain intercluster communications—either for itself or as an intermediate node participating in routing. Frequent cluster changes are undesirable because they may incur excessive cluster algorithm processing and communications overhead and may disturb intercluster routing. However, cluster stability must be balanced against the overhead incurred by the proactive intracluster routing algorithm. Hence, a probabilistic stability criterion is used rather than a fixed hop count or fixed cluster size, as used by previous dynamic clustering algorithms. The  $(\alpha, t)$ -criteria balance the overhead because proactive routing overhead is directly related to the number of nodes in the rate of topological change among those nodes. Thus, the number of nodes and the diameter of [WORD MISSING?] on  $(\alpha, t)$ -cluster will vary dynamically according to the mobility of the nodes. In addition to the definitions provided above, the intercluster routing methodology described in the next section requires the following properties to be satisfied:

PROPERTY 2.1: *Node covering*. Let  $N$  be the set of all nodes in the network and  $Q$  be the set of all the clusters in the network. The union of all the clusters  $C_i \in Q$  must equal  $N$ —the set of clusters covers the network.

PROPERTY 2.2: *Cluster exclusivity*. Let  $Q$  be the set of all

1. Node  $p$  is referred to as the parent node of cluster  $\mathcal{C}$ .

the clusters in the network. Then the intersection of any pair of clusters  $C_i, C_j, i, j$  in  $Q$  must be empty:  $C_i \cap C_j = \emptyset$ .

PROPERTY 2.3: *Identifier uniqueness*. All nodes in a given cluster share a common cluster identifier (CID). The CID must be unique among all the clusters in the network.

To improve clustering performance, the characterization presented in this section differs from McDonald and Znati [1] in two ways: first, clustering no longer requires mutual path availability between every pair of nodes in the cluster. A cluster is defined with respect to the availability of the paths between the parent and each child node, thus reflecting the probability that each node will remain associated with its parent. This is sufficient to ensure cluster stability. Furthermore, so long as each node remains connected to its parent over a cluster-internal path, it is guaranteed to be connected to every other node in the cluster over a cluster-internal path.

The second difference is that nodes are no longer required to leave a cluster when their path availability to other nodes in the cluster falls below  $\alpha$ . Hence, once a node has joined a cluster, it remains part of that cluster until it can no longer reach its parent along a cluster-internal path. This change minimizes unnecessary shifting of nodes between adjacent cluster and eliminated unnecessary cluster creation. Together, these changes improve the cluster residence time without increasing the overhead associated with intracluster routing.

## 2.2 $(\alpha, t)$ -Cluster Routing Methodology

This subsection introduces a hybrid mobility-based routing strategy supported by the  $(\alpha, t)$ -cluster organization. The cluster-based routing strategy is designed to leverage the adaptive characteristics of the  $(\alpha, t)$ -cluster and dynamically divides routing into two components: (1) intracluster routing for maintaining routes between destinations that reside within the same cluster and (2) intercluster routing for establishing routes between destinations that reside in different clusters. According to the  $(\alpha, t)$ -cluster-framework, the routing strategy consists of a proactive intracluster routing algorithm and a reactive intercluster routing algorithm. The contribution of the two components is balanced dynamically according to node mobility by the clustering algorithm.

Intracluster routes are maintained proactively using a table-driven routing algorithm. The framework is flexible and independent of the specific intracluster routing algorithm. Any routing algorithm designed for proactive operation in an ad hoc network can be used. Examples include DSDV, STARA, and WRP [8-10]. The  $(\alpha, t)$ -cluster ensures that the domain of operation for each *instance* of the algorithm is controlled. The only special requirement is that the algorithm can be adapted to compute paths according to a mobility-based metric [2, 3] or similar adaptive metric.

The idea of the intercluster routing strategy is to use the dynamic cluster topology and the intracluster routing tables to efficiently manage the routing process. The intercluster routing protocol (ICRP) is a fully reactive cluster base routing protocol that establishes and maintains routes on a demand basis only. In ICRP, the parent nodes of each cluster cooperate to control the route query process without flooding. By relaying the queries through each cluster, a *virtual route* (VR) is established that consists of a sequence of relay nodes (RN)—specifically, one node per cluster between the source and destination clusters. Although the parent nodes manage the query process, the relay nodes in an ICRP route can be any node—ICRP routes need not involve a parent.

There are two phases of the intercluster routing process. In the first phase, the cluster topology facilitates a route search process that eliminates the need for *flooding* yet does not require the complex query control mechanisms needed by other hybrid schemes such as ZRP [11]. The route construction and maintenance protocol (RCMP) handles this process. In the second phase, packet forwarding is achieved on a *cluster-by-cluster* basis. Each intercluster route is effectively hierarchical. Thus, unlike ZRP node-level topology, changes are handled by the proactive intracluster algorithm. Thus, reactive route maintenance can be significantly reduced. The packet forwarding protocol (PFP) handles the second phase. PFP routes packets between RNs that lie along an ICRP-VR. The intermediate nodes along an RN-RN path are determined by the intracluster routes. Unlike most cluster-based schemes, PFP does not require cluster head participation. Thus, it is more robust and efficient since traffic is not concentrated on a limited set of nodes.

Figure 2 depicts the logical organization of the network-layer entities that comprise the  $(\alpha, t)$ -cluster framework and their interfaces to service users and providers. At the highest layer is the Internet protocol (IP) that packetizes data and enforces an addressing structure on the network.<sup>2</sup> IP requires routing services to determine the next hop to reach each destination. These services are provided by the routing layer, which is depicted by two components: ICRP and DSDV. The routing layer is coupled with the clustering layer, which is represented as the DDCA for the  $(\alpha, t)$ -cluster. The clustering layer provides a logical network over which the routing algorithms build the routing tables that provide the service interface to IP. Finally, at the lowest layers, Figure 2 depicts a generic media access and control protocol (MAC) interfaced with a physical layer. The MAC provides a common set of services to all the network-layer elements, controls access to the shared wireless medium, and transfers data between directly adjacent nodes. Mobility information is required by the intracluster

2. Issues including dynamic naming and addressing services, as well as the binding of names and addresses, are beyond the scope of this research. However, we recognize that these become crucial issues with respect to system implementation and internetworking.

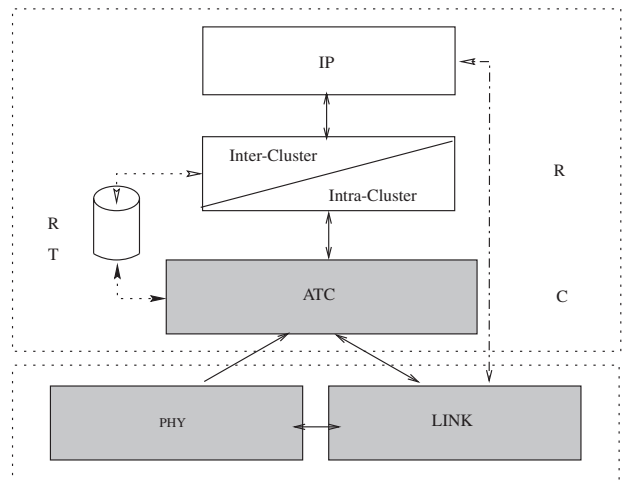


Figure 2. Logical organization of network-layer entities

routing algorithm and the DDCA for computation of path availability. This is required for intracluster routing and for evaluating the  $(\alpha, t)$ -cluster-criteria. The figure depicts an interface between the physical layer and the DDCA that is used for this purpose. It is assumed that a physical-layer entity, such as a location tracking system or sensor, periodically computes mobility parameters and reports them to the DDCA entity.

### 2.3 System Parameters

Two system parameters are central to the adaptive control and performance of the cluster-based routing framework. Specifically, the parameter  $\alpha$  places a theoretical upper bound on the probability of path failure between a node and the parent node of its cluster due to node movement, and the parameter  $t$  defines the time interval over which the bound holds. These parameters represent critical factors with respect to system performance and behavior. The determination of *optimal* values for  $\alpha$  and  $t$  is a difficult problem. This subsection discusses issues related to the problem of determining the system parameters and a framework for characterizing optimal values that is adaptive with respect to network traffic and size.

Within each cluster, all the nodes participate in a proactive intracluster routing protocol. Maximum availability paths are maintained between each pair of nodes; hence, link availability is used as the metric for computing optimal paths. According to the  $(\alpha, t)$ -criteria, a node is permitted to join a cluster *iff* it can establish an internal  $(\alpha, t)$ -path between itself and the parent node of the cluster. Hence, path availability information must be exchanged frequently enough to ensure that all the computed availabilities reflect an interval of time that is *at least* of length  $t$ . That is, new paths must be computed using up-to-date routing

information before the time  $t$  elapses. Therefore, it becomes clear that the value of  $t$  must be tied to the value of the proactive routing update interval that is fixed by the intracluster routing protocol.

For a fixed value of the parameter  $t$ , the parameter  $\alpha$  controls cluster adaptation. Consequently, it controls the performance of the scheme. If the value of  $\alpha$  is at or near to zero, the scheme will become fully proactive as little or no path availability bound is required. Hence, the cluster will grow to include all the nodes of the network regardless of the characteristics of node mobility. If the value of  $\alpha$  approaches unity, it will become nearly impossible to maintain clusters of any significant size, unless the nodes become nearly stationary or if the nodes are moving in lock step and the behavior can be captured by the link availability model in use. Consequently, if  $\alpha$  is large, routing will generally become fully reactive as each node maintains its own cluster.

It is likely that the impact of system parameter selection will depend on the level of node mobility and network size. This raises the question as to how optimal values should be determined. There are many alternatives for characterizing optimal system parameters—a methodology for finding the optimal ZRP zone radius based on minimization of ZRP traffic is presented in Pearlman and Haas [12]. A similar approach can be adopted for optimization of the system parameters in the  $(\alpha, t)$ -cluster. Assuming that the value of  $t$  can be expressed as a function of the proactive routing update interval, the optimal proactive routing update interval should maximize intracluster throughput. This is itself a difficult problem for *any* proactive routing algorithm; however, if the maximum cluster size is bounded, an approximate solution may be determined experimentally via simulation based on assumptions regarding traffic distribution, transmission bandwidth, and the range of node mobility rates. Given a fixed value for  $t$ , a dynamic approach for optimizing the value of  $\alpha$  can be used that is adaptive with respect to both traffic characteristics and the size of the network. Specifically,  $\alpha$  can be determined locally by applying the same approach as the one used in Pearlman and Haas [12] for finding the optimal zone radius. As such, the value of  $\alpha$  will adapt to the size of the network, as reflected by the overhead resulting from existing traffic conditions. Thus,  $\alpha$  ties mobility together with the dynamically changing traffic patterns and numbers of nodes in the network.

### 3. The Dynamic Clustering Algorithm

The objective of DDCA is to partition the network into  $(\alpha, t)$ -clusters, and DDCA maintains  $(\alpha, t)$ -clusters asynchronously in a distributed fashion. As such, the algorithm runs continuously and asynchronously on each active node in the ad hoc network. There is no need for centralized control or periodic *reclustering*. This differs from a number of earlier clustering schemes that require periodic

topology reorganization [13, 14]. For a cluster to be a *feasible cluster* for a given node, the  $(\alpha, t)$ -criteria must be satisfied. Namely, before a node can join a cluster, it must find an  $(\alpha, t)$ -path from itself to the node that is the parent of the cluster.

Each node must be in one of the five DDCA node states, as specified in Table 1—namely, inactive, unclustered, orphan, parent, and child. These states provide the means for distributed control over the clustering process. The main idea of DDCA is to use path availability information maintained by the underlying intracluster routing algorithm to determine if a given cluster is feasible. The *cluster strength* is used to determine cluster feasibility. Cluster strength for a given unclustered node,  $n$ , as evaluated by a given node  $m$  that is in the cluster, is a measure of the path availability from  $n$  to the parent node of the cluster, along a path whose initial hop is node  $m$ . An unclustered node can join the cluster *iff* there exists a node  $m$  through which the cluster strength is at least  $\alpha$ . This constraint is referred to as the  $(\alpha, t)$ -criteria.

In DDCA, each unclustered node seeks a feasible cluster by broadcasting a *join-request* message. If it receives no responses, it creates a new cluster in which it is alone—an orphan. To prevent adjacent unclustered nodes from each creating new clusters, simultaneous requests are handled by forcing nodes with higher identifiers to back off and try again. A node that receives at least one join-response message joins the maximum-strength cluster from which a response was received. A node joins a cluster by changing its state, setting its cluster identifier (CID), and initiating an intracluster routing exchange with its neighbors. As a child, each node must process and respond to join-request messages and detect if it has become disconnected from the cluster or if a cluster partition has occurred. The parent of every cluster is initially an orphan. Each orphan node periodically attempts to join an adjacent cluster until it detects that at least one child has joined its cluster. This can be detected by the reception of routing information and the subsequent increase in size of the intracluster routing table. Each parent node must process and respond to join-request messages and detect if it has become disconnected from its children. The following subsections present detailed specification for each DDCA state, including discussion of the relevant events and actions associated with the corresponding state.

#### 3.1 Inactive State Specification

An inactive node does not participate in the ad hoc routing. The only event that evokes DDCA action is *node activation*. An activating node rapidly seeks to join the best cluster it can find by querying its neighbors as follows: the node broadcasts a join-request message to its neighbors and initiates a *join timer*, which specifies the amount of time the node will wait to gather responses. Finally, the node transitions into the *unclustered* state, where it awaits responses but cannot participate in network routing.

**Table 1.** Description of distributed dynamic cluster algorithm (DDCA) states

State	Description
Inactive	The node communications process is either nonoperational or is active but not currently participating in ad hoc routing.
Unclustered	The node is active but not currently affiliated with a cluster. In this state, the node is not able to communicate with other nodes.
Orphan	The node has created a new cluster in which it is currently the only member. The cluster identifier (CID) of the cluster is assigned the orphan-node node identifier (NID). The node participates fully in intercluster communications.
Parent	The node is affiliated with a cluster that includes at least one additional node. The CID of the cluster is assigned the parent-node NID. The node participates fully in both intracluster and intercluster communications.
Child	The node is affiliated with a cluster in which it is not the parent node or an orphan node. The CID is assigned to the NID of the cluster's parent-node NID. The node participates fully in both intracluster and intercluster communications.

### 3.2 Unclustered State Specification

Three events evoke a DDCA action at an unclustered node. Predicates determine the precise actions or state transition. During this state, the node is waiting for responses to its broadcast join-request message. Each *join-response* message received prior to the expiration of the join timer is processed by the unclustered node. The message reporting the *maximum* cluster strength is accepted. Namely, after expiration of the join timer, the node will join the cluster that offers the greatest reported  $(\alpha, t)$ -availability. After processing each join-response message, the node continues to wait for additional responses and remains in the unclustered state.

As described previously, there is no centralized control over the clustering process. This raises the possibility of multiple simultaneous activations or, more generally, broadcast join-request messages. If the nodes are adjacent, this will result in each unclustered node receiving a join-request message from its neighbor. This can become a problem if some or all of the unclustered nodes fail to receive any join-response messages. In this case, one or more new clusters need to be created. If all the nodes are permitted to create new clusters, there will be a proliferation of clusters—this could lead to excessive adopt request traffic in the future and potentially longer intercluster routes than necessary. A better approach would be to limit the number of new cluster formations and permit them to expand rapidly by adding other unclustered nodes.

A deferral algorithm is used based on the node node identifier (NID). In each clique involving unclustered nodes receiving join-request messages, only the node with the minimum NID is permitted to create a new cluster. The other nodes must wait until the expiration of their join timers before they are permitted to retry. If the mobility conditions are sufficient to satisfy the  $(\alpha, t)$ -criteria, many of the nodes will be able to join the newly created cluster of their lower ID neighbors in a subsequent round. A node that receives join-request messages either prior to or after receiving a join-response message must be unaffected by the join-request messages.

The third event that is processed by unclustered nodes is the *join-timer expiration*. The action taken by DDCA depends on the events that preceded it. If the node has received at least one join-response message, the *join flag* will be in the *set* state, and the node will join the cluster from which it received the maximum-strength response. Joining a cluster entails setting a node's CID to the selected CID and broadcasting a routing update to the neighbors that effectively requests an immediate complete routing update.

If no join-response message was received during the timeout interval, the action will depend on whether any simultaneous join-request messages with lower NID were received. If at least one lower NID request was received that was not being blocked from the previous round, the node will rebroadcast its join-request message and restart the join timer. However, if no simultaneous join-request messages were processed from lower NID nodes, then the node will create a new cluster in which it will be the only node. The node sets its CID to its NID, initiates its *adopt timer*, and DDCA transitions to the *orphan* state.

### 3.3 Child State Specification

A child node is a node that is affiliated with one or more other nodes in a cluster, and it is not the parent node of that cluster. The child node participates actively in the proactive intracluster routing algorithm, processing any and all routing updates received from its neighbors. It also participates as necessary in the reactive intercluster routing algorithm. It may be required to process or forward route query and response messages or to forward data across the cluster.

Nodes become children in a cluster when they are unclustered and have accepted a join response or when they are orphan nodes and have accepted an adopt response. As a child, each node has the responsibility of processing join-request and adopt-request messages. Each child also must detect the events that remove it from the cluster: *cluster disconnection* and *cluster partition*. A cluster disconnection occurs when the node no longer has an internal path to any other node in its cluster. It is detected when the size of

the intracluster routing table goes to 1—the self-entry. A cluster-disconnected node broadcasts a join-request message, initiates its join timer, and changes to the unclustered state.

A cluster partition occurs when one or more multiple-node subsets of the cluster become disconnected from the remaining nodes in the cluster. Cluster partition is detected when a node, which is not cluster disconnected, notices that the parent node is no longer reachable.

Detection of a cluster partition is simple, but partition repair can become difficult to achieve. The objective is to rapidly unify multiple partitions sharing a common CID. The basic algorithm, referred to as simple partition detection (SPD), requires that each node detecting a partition effectively assumes that it has become cluster disconnected and attempts to join a new cluster. Stability is ensured by forcing the node to defer its join request for one jointimer interval and preventing it from rejoining its previous cluster.

Based on the SPD strategy, two algorithms—namely, the least overhead partition repair (LOPR) algorithm and the optimal partition repair (OPR) algorithm—can be used. LOPR attempts to reduce the overhead associated with the simultaneous reclustered of all the nodes in a partition. The idea is to dynamically reassign as many of the nodes as possible to a cluster that is induced on the existing intracluster routing tables. In LOPR, each node detecting the partition selects the node in its cluster partition with the lowest NID as the implicit parent of its next cluster. If the node determines that it has the lowest NID, then it creates a new cluster in which it assigns itself as the parent. Otherwise, it evaluates its  $(\alpha, t)$ -criteria with respect to the lowest NID node. If the clustering criteria can be satisfied with respect to that node, then it joins the new cluster as a child.

The OPR strategy is designed to select the *best* node in the partition as the parent of the new cluster. It is based on a distributed election process in which each node advertises its own computed *strength* as the parent of the partition. The strength could be based on any desired metric. For example, it could be the mean path availability from the bidding node to all destinations in the partition. The join timer provides a settling time to improve partition detection and consistency of the routing tables. Each node generates and floods its bid within the partition. However, the flooding is terminated early by any node that has previously seen a better bid. At the end of the repair time, each node selects as its new parent the node from which it had received the best bid.

In addition to maintenance of its own state and cluster affiliation, each child node must cooperate with other nodes in the maintenance of their state and cluster affiliation. Whenever a join-request or adopt-request message is received, the child node uses knowledge of the link metric between itself and the source of the request to compute the *strength* of the cluster as seen by requesting node. Specifically, it computes the availability of the path from

the requesting node to the parent using itself as the initial hop. If the strength meets the  $(\alpha, t)$ -criteria and the cluster size does not exceed a predefined maximum value, a join-response message is returned.

### 3.4 Orphan State Specification

Clusters are created whenever an unclustered node cannot find a feasible cluster to join. All clusters begin with a single node, referred to as an orphan. If the cluster later expands to include child nodes, the orphan becomes the parent of the cluster. As an orphan, a node is able to fully participate in both intracluster and intercluster routing. Intracluster routing involves the maintenance of routes to each *external border node* and the detection of cluster expansion. An orphan node must perform all intercluster route processing for the cluster.

Orphan nodes actively seek to expand their own cluster by adopting other orphans or accepting unclustered neighbors into their cluster. They also periodically attempt to join other clusters by broadcasting an adopt-request message. Because orphan nodes will use this dual strategy to affiliate with a larger cluster, the orphan state is effectively the most complex of the DDCA states.

The adopt timer regulates the periodic broadcast of adopt requests. At each expiration of the adopt timer, an adopt request is broadcast and the join timer starts. An *adopt cycle* consists of one adopt-timer interval followed by a join-timer interval. While the adopt timer is running, the node cannot join another cluster, but it can expand its own cluster. While the join-timer is running, the node can be adopted. However, any attempt to be adopted will be preempted by a cluster expansion. If at least one nonpreempted adopt response is received, the node will become a child in the cluster of the node with the maximum-strength response.

Cluster expansion is preferable to being adopted because it will not disturb intercluster routes or route searches that traverse the orphan node. Hence, join-request and adopt-request messages effectively preempt any attempt by the orphan node to seek adoption into another cluster. Hence, a response is returned so long as the  $(\alpha, t)$ -criteria hold. Once a response to either a join request and an adopt request has been sent, any responses to the orphan node's own adopt request previously or subsequently received during the current adopt cycle are ignored. Furthermore, if the adopt timer is running when a response message is sent, the adopt request is not sent at the next expiration of the timer. This reduces the probability of an unclustered node joining the orphan node cluster *after* it has been adopted. This measure further favors expansion of the existing cluster and reduces potential instability that can be caused if nodes continuously join clusters that no longer exist. A concurrency problem arises when two adjacent orphan nodes both seek to be adopted by the other. To overcome this problem, an efficient lowest NID first algorithm is used to

resolve multiple simultaneous adopt requests. Namely, a response is returned only by the node with the lowest NID.

Finally, a cluster expands when an orphan node sends a join response or an adopt response that is accepted by its recipient as a maximum-strength response. Cluster expansion is detected when the orphan node receives an intracluster routing update from the new cluster member, which results in an increase in the size of the intracluster routing table. The state changes from orphan to parent when the size of the cluster increases beyond one node.

### 3.5 Parent State

The *parent* state is the simplest of the DDCA states. There are no timers to manage, and there are only three possible events that can occur. Namely, the node may receive a join-request message, it may receive a adopt-request message, or it may detect that it has become disconnected from the remaining nodes in the cluster. Request messages are handled in a manner identical to the handling by a child node. Cluster disconnection is also detected as in the case of a child node by loss of reachability of all nodes in the cluster. The parent node detecting its disconnection will broadcast a join-request message, initiate a join timer, and enter the unclustered state. The disconnection of a parent node from its cluster will be detected as a *cluster partition* by the child nodes in the cluster, regardless of the connectivity that exists among them.

## 4. Simulation

Several factors need to be considered in evaluating the performance of the  $(\alpha, t)$ -cluster protocol. Namely, DDCA should (1) adapt to node mobility by dynamically changing the cluster size and membership according to the  $(\alpha, t)$ -criteria and changes in link status, (2) provide a relatively stable infrastructure to support effective routing both within and between clusters, (3) achieve cluster maintenance with minimal communications overhead, and (4) achieve scalability by limiting routing overhead and the far-reaching effects of topological changes. Based on these observations, a discrete event simulation was developed to evaluate the dynamic properties of the  $(\alpha, t)$ -cluster protocol with respect to cluster stability and protocol efficiency. Specifically, simulation was used to measure clustering algorithm effectiveness in terms of key performance metrics. These metrics include the following: mean cluster size and intracluster routing table size, mean node residence time within a given cluster, the probability that a node is in a cluster (excluding orphan clusters), cluster algorithm control message-processing rate, and the normalized load due to proactive routing update processing. The remainder of this section discusses the simulation model and presents analysis of the results.

### 4.1 Cluster Algorithm Simulation Model

A complete state transition specification of DDCA appears in McDonald [15]. For the current analysis, DDCA was

implemented using *CSIM*, with DSDV as the underlying proactive intracluster routing algorithm. Networks ranging from 50 to 400 nodes were simulated under conditions ranging from nearly fixed to highly mobile. Simulation runtimes for large or highly mobile ad hoc networks become enormous when one is attempting to generate statistically significant results. Hence, it becomes infeasible to simulate networks much larger than 400 nodes unless the models are very simple. Fortunately, the results either demonstrate insensitivity to the number of nodes, or the effects of network size are readily apparent. Consequently, the results can be generalized to networks significantly larger than 400 nodes. This is also equivalent to the largest ad hoc network simulation to appear in the literature to date.

The hypothesis underlying this research is based on the assumption that the node mobility model is valid—that is, we are interested in evaluating the performance of the clustering and routing algorithms, given that the model used to evaluate the clustering criteria is accurate. Thus, in this section, we are evaluating the intrinsic performance of the clustering strategy rather than the validity of the mobility model (see Fig. 2).

Node mobility was modeled according to a random-independent model used to evaluate link availability. Mean velocity was varied between 1.0 and 10.0 meters per second (3.6–36.0 kph). To model a moderately sparse network in which multiple-hop routing performance would be a crucial concern, the mean number of neighboring nodes was kept at approximately 3. This also limits the effects of MAC-level contention, which was modeled using a distributed queue to prevent collisions and resolve the hidden terminal problem. We assumed an ideal physical layer in which transmissions were received error free if the sending node was within a fixed-distance threshold from the receiver at the time it acquired the channel. For all experiments, transmission range was assumed to be 250 meters.<sup>3</sup>

Figure 3 shows the measured network characteristics as a function of mean node velocity for each of the network sizes that was simulated. All data points plotted in this section reflect the sample mean of the 99.44% confidence interval. Figure 3a shows the resulting aggregate mean rate of link status change measured as the sum of all link-ups and link-downs observed per second. As expected, this is an increasing function of both node mobility and the number of nodes. Figure 3b shows the mean node density, which remains relatively stable for all mobility rates and network sizes. The apparent spread is due mainly to random variation, which becomes more pronounced at higher mobility. It is also compounded by the granularity for assessing link status, which becomes more important at higher mobility. Hence, the differences are not statistically significant.

Two system parameters—namely,  $\alpha$  and  $t$ —are central to the adaptive control and performance of the cluster-based routing framework. Specifically, the parameter  $\alpha$

3. This value is fairly representative of commercially available IEEE 802.11 NICs in free space.

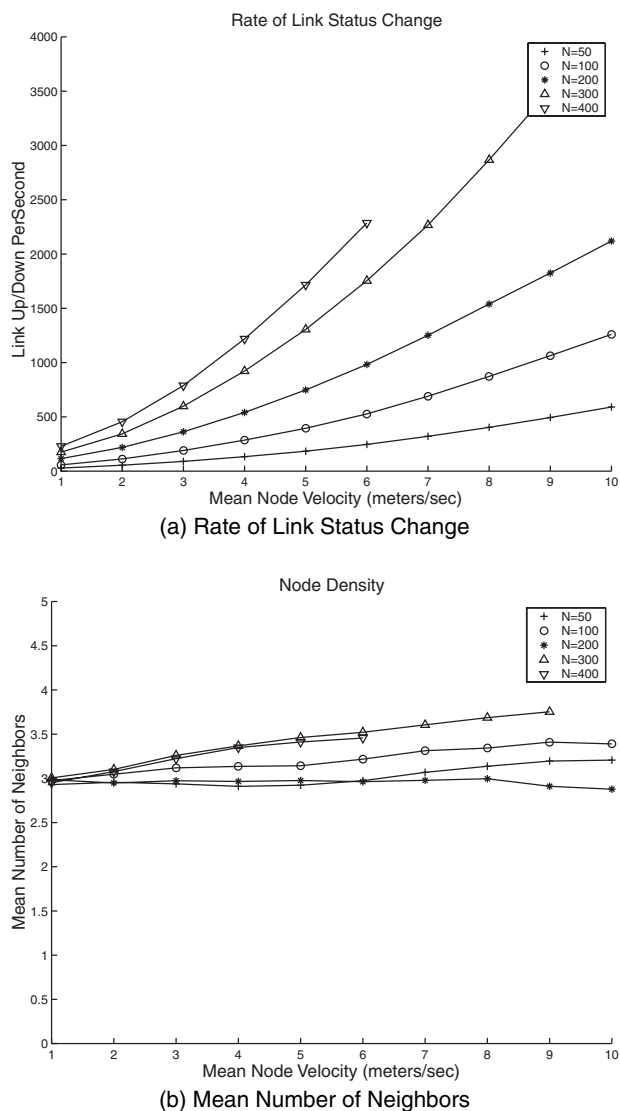


Figure 3. Network characteristics as a function of mobility

places an upper bound on the probability of path failure due to node mobility within a cluster, and the parameter  $t$  defines the time interval over which the bound holds. These parameters represent critical factors that are expected to affect the performance on the  $(\alpha, t)$ -cluster framework. The unique feature of the  $(\alpha, t)$ -cluster scheme is that for fixed values of system parameters, the scheme is intrinsically adaptive—this is in contrast to schemes such as ABR and ZRP, which rely on fixed-system parameters that entirely define the behavior of the system. It is not feasible to exhaustively test all possible combinations of system parameters. Instead, we adopt an approach based on the 2k factorial experimental design in which we choose two values for each parameter—a high and low value. These values

were selected as follows:  $\alpha_{low} = 0.25$ ,  $\alpha_{high} = 0.50$ , and  $T_{low} = 0.0$ ,  $T_{high} = 30.0$  in all experiments. The values for  $\alpha$  were chosen to account for the fact that path availability is multiplicative; hence,  $\alpha_{high} = 0.50$  represents a maximum two-hop path with a probability of survival of about 0.7. Since link availability drops off quite rapidly once moderate velocities are attained, this represents a reasonable upper bound that still allows the possibility of multiple-hop paths from the cluster parent node. The low value of  $\alpha_{high} = 0.25$  implies a two-hop path availability of only 0.06, which may not be a very strong path for routing. Hence, anything less would not be acceptable. The value of  $T = 0.0$  was chosen to reflect the instantaneous conditions, whereas the value of  $T = 30.0$  reflects the intracluster routing update interval and thus bounds the time until the next link availability update. For each of the metrics, plots are only shown for selected combinations of the factors: (0.25, 0.0), (0.25, 30.0), (0.50, 0.0), and (0.50, 30.0). Similar trends were observed in the combinations omitted in this paper.

The steady state of the system had to be verified prior to collecting data. To this end, startup transients presented a problem due to bunching of link events at the start of each run. The Welch method [16] was used to find a basis value for the warm-up period under low mobility, which was the slowest to reach steady state. To avoid wasting simulation time, the method of batch means was used to compute statistics for each metric. Each experiment ran until either a desired precision was obtained for each metric or a maximum of 100 batches were obtained.

#### 4.2 Simulation Output Analysis

Cluster affiliation is crucial because a node cannot participate in routing unless it belongs to a cluster. Furthermore, to ensure that the union of all clusters covers the network, every node must belong to a cluster by itself—an orphan cluster. However, such a strategy reduces to a pure reactive scheme. In fact, it is the intent of the  $(\alpha, t)$ -cluster to provide pure reactive routing when the operational domain warrants it. However, it tends to be more desirable to cluster with multiple nodes so long as the proactive routing overhead is not excessive. Thus, if mobility is low or the routing overhead is not excessive, nodes should be affiliated with multiple node clusters. Figure 4 shows this probability. Specifically, it shows the probability that an arbitrary node will be in a cluster with at least one other node at an instant in time. Thus, it does not include orphan nodes that can participate in routing, nor does it include unclustered nodes that cannot participate in routing.

The data show that there is very little difference in the clustering probability based on the size of the network. This result is expected as clustering is a localized action that depends on the mobility and topology of nearby nodes. Figure 4a shows the most robust case in terms of multinode clustering in which the low values are used for both factors. Even at the highest mobility, between 70% and 80% of the

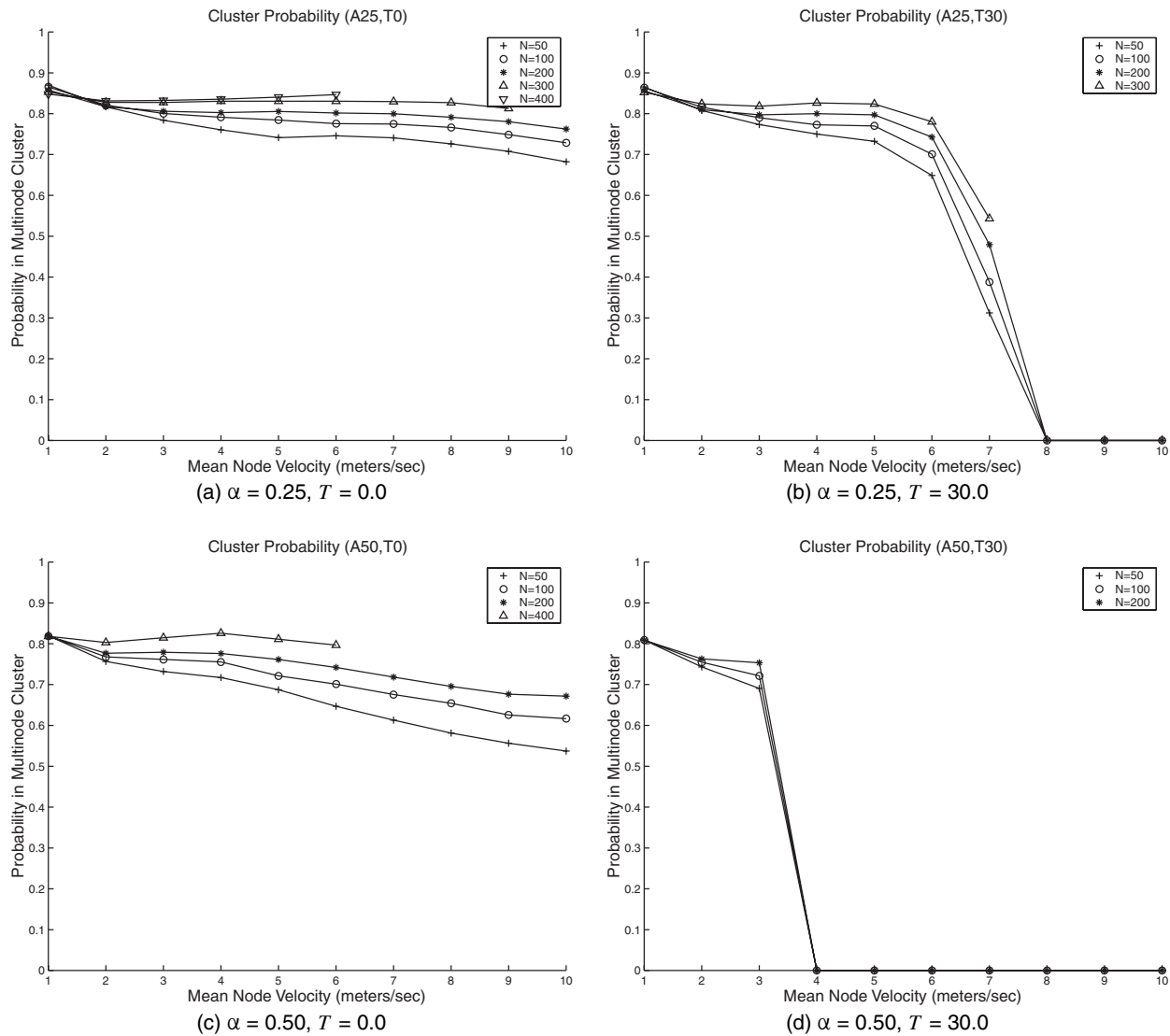


Figure 4. Node cluster probability as a function of mobility

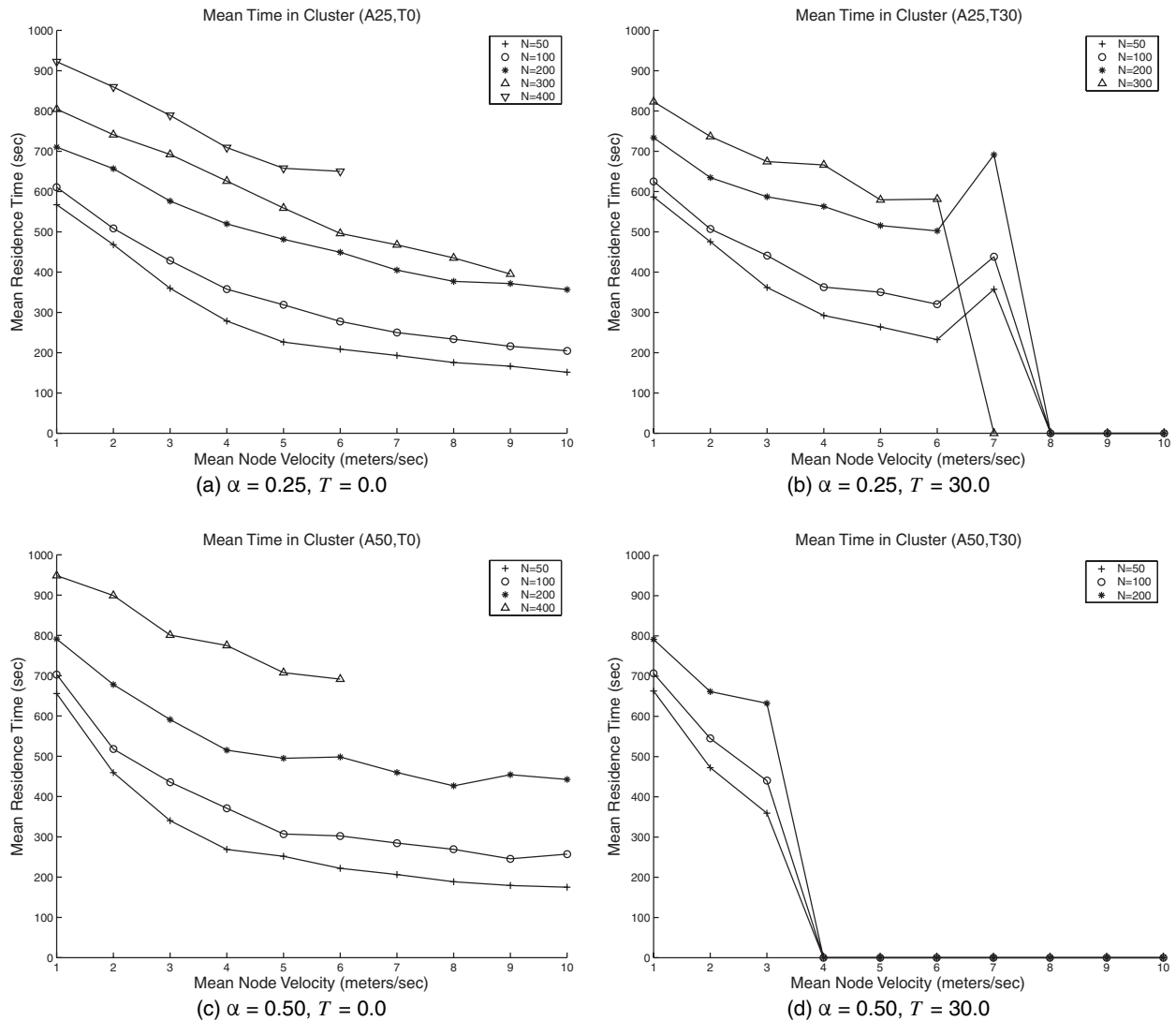
nodes remain clustered at all times. However, one would expect the clusters to be very small or for nodes to remain in their clusters for shorter durations at the highest mobility. These expectations are validated by the respective data for each metric.

The most striking thing about the plots in Figure 4 is the relative effects of the parameters. Specifically, increasing  $\alpha$  to its high value introduces a *knee* into the curve. The precise point of the knee depends on the value of  $T$ , which affects the rate at which cluster probability drops off with increasing mobility. The question is whether this behavior is a good predictor of the operational domain and hence the routing mode that should be “activated.” The answer requires examination of additional metrics in parallel with

routing performance. As a general observation, it appears that for  $\alpha = 0.25$ , clustering probability is relatively insensitive to node mobility. How insensitive it is depends on the value of  $T$ . At the other extreme setting,  $\alpha = 0.50$  provides very high probabilities of clustering up to a clear breakpoint that is determined by the value of  $T$ . It appears that with  $T = 0.0$ , this breakpoint provides clustering at moderate and high rates of mobility and a smoother transition, whereas with both parameters set to their high values, the drop is precipitous.

In Figure 5, cluster residence time is plotted versus node mobility. Residence time is a good measure of cluster stability. It is analogous to cell residence time in cellular systems, which is a determinant of the distribution and rate of

## SIMULATION OF DISTRIBUTED DYNAMIC CLUSTERING ALGORITHM



**Figure 5.** Cluster residence time as a function of mobility

handovers. Residence time measures the mean time that a clustered node remains in the same cluster. The higher this value, the more stable the cluster topology will be. This has an impact on the routing overhead within a cluster, as well as the stability of intercluster routes, which rely on continuity at the cluster level.

Comparing the plots in Figures 4 and 5 reveals similar effects due to the values of  $\alpha$  and  $t$ . However, cluster residence time is more sensitive to the number of nodes. For example, in Figure 5a at the 95.44% confidence level, there is a significant difference between the response for  $n = 400$  versus both  $n = 50$  and  $n = 100$ . However, the difference cannot be verified between  $n = 50$  and  $n = 100$ . The trend, however, seems quite clear. Large networks in-

duce improved cluster residence times. The reason for this is not absolutely evident. However, it is possible that it is related to the greater number of paths that are possible in a large network that may enable nodes with a cluster to remain connected over a longer period of time.

Irrespective of the sensitivity to network size, cluster residence times seem adequate or excellent in all cases for  $\alpha = 0.25$ . However, the high-mobility cases reflect the small size of the clusters themselves and may not result in a great benefit. The most interesting case again appears to be the where  $\alpha = 0.25$  and  $T = 0.0$ . The metric reflects mean residence times that range from 5 to almost 9 minutes at speeds of up to 20 kph. These are excellent results. Cluster residence times begin to drop sharply after this point,

reflecting a natural breakdown in the cluster topology that is necessary to manage the intracluster proactive routing overhead, which is reflected in Figure 6.

Proactive routing overhead is a direct measure of the processing rate of intracluster routing updates. It is the number of routing updates processed per node, per second. Clearly, small values are desirable so long as overall routing performance is adequate. This cannot be assessed directly through cluster metrics. What is interesting to see is that in the cases that appeared *most* robust relative to cluster probability and cluster residence time, the routing overhead increases more rapidly with node mobility. In contrast, the rate of proactive routing overhead increase appears to be reduced by the *declustering* effects of the  $\alpha = 0.50$  system. In fact, the proactive routing overhead seen at the high-mobility ends reflects updates that are processed primarily by orphan nodes that must maintain routes to all neighboring clustered nodes. Due to extreme variation in these numbers, very little statistical significance can be verified based on the number of nodes, with the exception of the highest mobility cases.

It is interesting to compare the proactive routing update overhead induced by the  $(\alpha, t)$ -cluster to other schemes. Figure 7 compares the proactive routing overhead of the  $(\alpha, t)$ -cluster for various system parameter combinations to three *versions* of ZRP. High values of zone radius tend to be ZRP proactive, whereas small values are reactive. Values in the middle represent a fixed hybrid strategy. Based on these results, it is evident that the proactive routing generated by the  $(\alpha, t)$ -cluster is significantly less than ZRP and proactive schemes. Furthermore, the scaling demonstrates that it is relatively insensitive to changes in node mobility. This is strong evidence in support of scalability.

One of the arguments that is often raised against dynamic clustering is the presumed substantial overhead required to maintain the cluster topology. In Figure 8, this argument is defeated with respect to the  $(\alpha, t)$ -cluster. Underscoring the robustness of the  $(\alpha, t)$ -cluster supported by the previous metrics, Figure 8 demonstrates the profound communications and processing efficiency of the clustering algorithm. The figure shows the aggregate rate of DDCA control messages processed in the network. Specifically, it is the sum of *all* DDCA messages—requests and responses processed per second in the network. Due to the statistically identical characteristics with the other cases, only the two  $T = 30.0$  cases are shown. These numbers verify that the communications overhead of the DDCA protocol is insignificant. Note that some of the overhead associated with clustering activities is included in the proactive routing overhead. Namely, full-routing table dump requests and full-dump responses are routing protocol updates that are triggered by clustering actions. These are reflected in the previous measures of proactive routing overhead. Hence, when accounting for the full communications overhead associated with DDCA, both the proactive routing and the DDCA control message overhead must be

considered together.

The final metrics of interest are the cluster size and intracluster routing table size. For space reasons, the plots are not shown here, but they may be found in McDonald [15]. The routing table size represents the effective cluster or proactive routing region. It determines the efficiency of both the route search process and the intracluster routing algorithm. Comparisons are made of the routing table sizes for each combination of system parameters. The results are independent of the size of the network.

As expected, the shape of the plots of the intracluster routing table size reflect results shown previously for cluster probability and residence time. Mean cluster size (not shown) tends to approach the average number of neighbors as mobility increases. Hence, in the case of  $\alpha = 0.25$ , the scheme effectively reduces to a *one-hop* cluster architecture under high mobility, whereas when  $\alpha = 0.50$ , the scheme becomes effectively fully reactive, with all the nodes becoming orphans under high mobility. The routing table size reflects the sum of cluster entries and *border nodes*, those nodes that are not in the same cluster but are adjacent to at least one node in the same cluster. The proactive routing overhead reflects the full complement of nodes in the routing table.

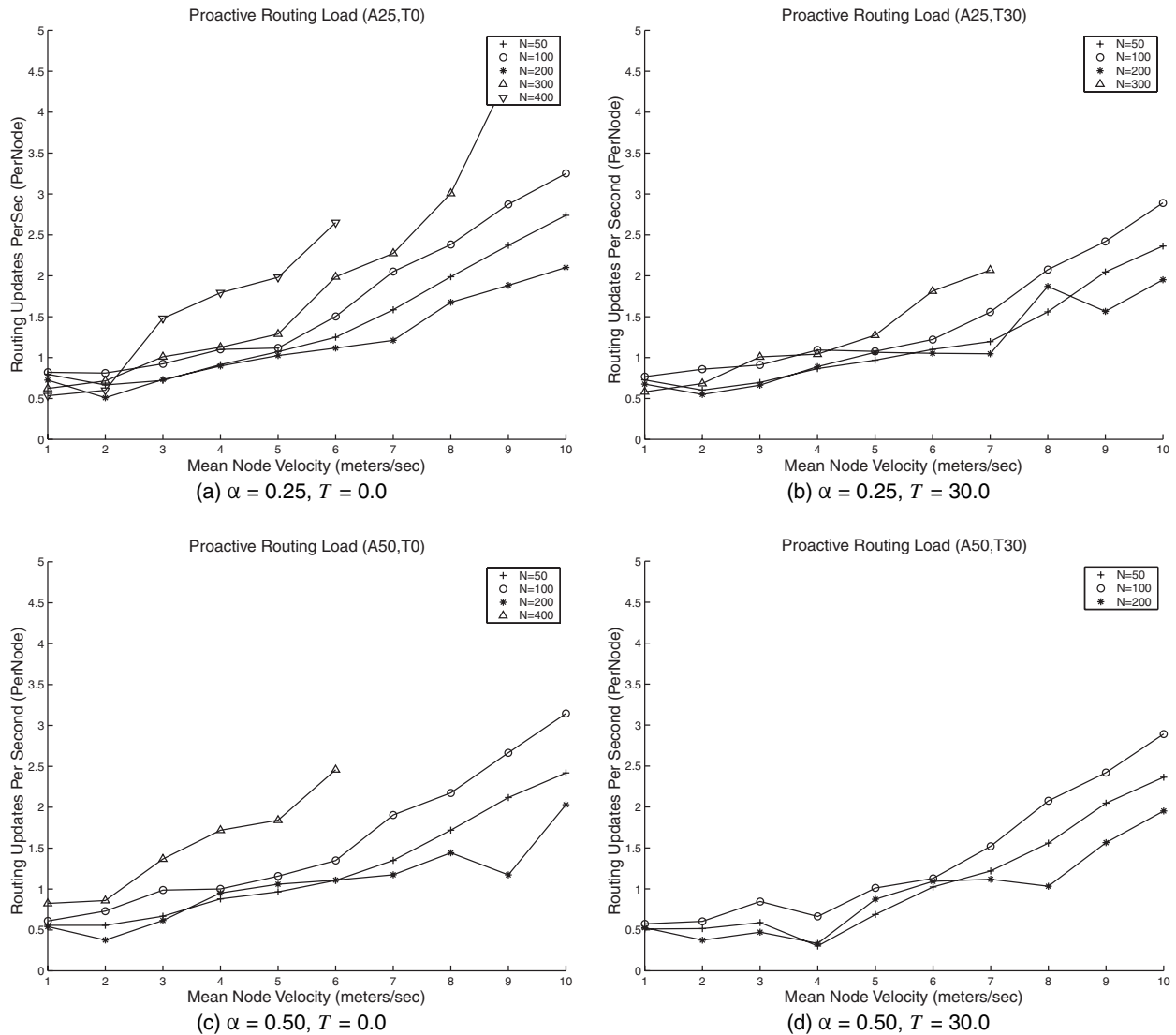
Additional results not shown in the figures demonstrate that the  $(\alpha, t)$ -cluster achieves longer intercluster path survival times than ZRP and pure reactive schemes due to the hierarchical nature of the paths. This results in fewer path setup and/or repair operations. Furthermore, it was also shown that total reactive query processing for the  $(\alpha, t)$ -cluster varies from only slightly reactive to fully reactive as mobility is increased, clearly demonstrating the desired adaptive multimode behavior.

## 5. Conclusions

This article presented the design and simulation analysis of a new class of clustering algorithms for wireless ad hoc networks. DDCA is an instance of the new class of *environmentally aware* adaptive dynamic clustering algorithms first introduced in McDonald and Znati [1]. DDCA uses a probabilistic mobility model to effectively sense the state of the network. Based on this technique, the class of algorithm is able to adapt to changing dynamics, thus supporting multimode routing. The purpose of multimode routing is to enable the routing algorithm to adjust itself based on spatial and temporal dynamics. This allows factors that affect performance, including but not limited to routing overhead, path quality, and cluster stability, to be dynamically balanced. This capability can be leveraged to enhance system scalability.

The DDCA was defined in terms of a finite set of states, events, and actions. Each node maintains its current state and information regarding its cluster affiliation. Based on the properties defined in Section 2, every node must belong to one—and only one—cluster. Thus, nodes actively seek

## SIMULATION OF DISTRIBUTED DYNAMIC CLUSTERING ALGORITHM



**Figure 6.** Proactive routing load as a function of mobility

a new cluster whenever they first activate or become disconnected from their previous cluster. A feasible cluster is described as a cluster that meets the stability requirement, referred to as the  $(\alpha, t)$ -criteria. An efficient broadcast request-response protocol is used to seek a feasible cluster. A node that is unable to find a feasible cluster within a timeout window creates a new cluster. As such, clusters are created, expanded, contracted, and eventually terminated.

The unique features of the  $(\alpha, t)$ -cluster are that it is an adaptive multihop subnet whose size and membership depend on a cost function, referred to as the cluster strength. Furthermore, the dimensions of a cluster may differ across a network and are likely to change over time. This adap-

tive functionality is an important innovation that represents a paradigm shift in dynamic clustering. In this paper, it is assumed that node mobility represents the critical factor affecting path availability and hence forms the sole basis for assigning cluster strength. However, the concept of clustering based on these adaptive criteria can easily be generalized to include other factors, including but not limited to power, channel characteristics, bandwidth, and security. Previous clustering strategies have been built around fixed criteria or cluster structure. The most common type of clustering is based on a central control entity, referred to as the cluster head (CH) [14]. In CH-based schemes, a cluster is defined as a set of nodes that are directly adjacent to one CH. All clustering decisions are made by the CH, and the

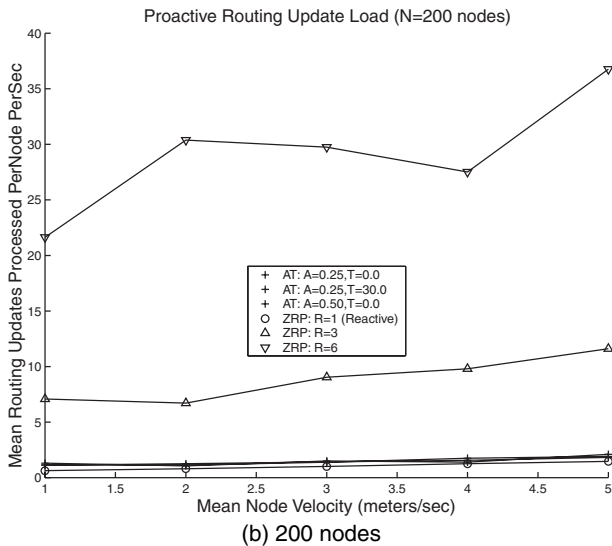
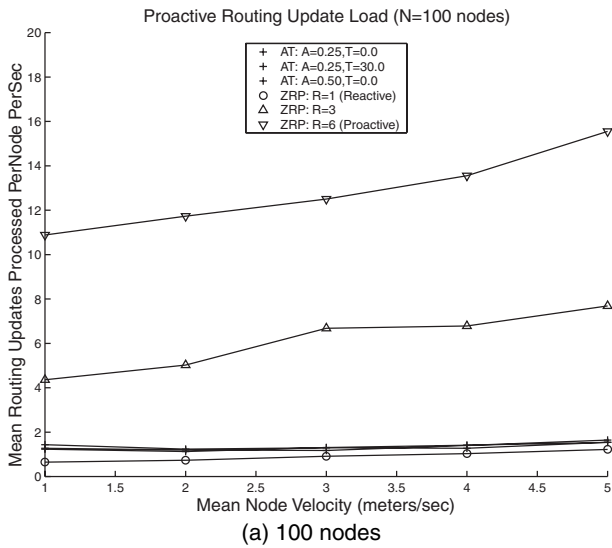


Figure 7. Comparison of proactive routing update processing rates

maximum path length is two hops. One scheme proposed by Basagni [7] uses an unspecified mobility-based cost function as the basis for cluster selection, but the cluster structure remains fixed. Recent clustering strategies have also been defined that are based on fixed hop counts [17, 18], fixed maximum cluster sizes [19], or geographical location [20]. The  $(\alpha, t)$ -cluster is the only fully adaptive clustering strategy and the only true clustering strategy that does not require the centralized control of a cluster head for clustering decisions. ZRP and geographical-based zone routing do not require cluster heads because they do not build true clusters—hence, they are not hierarchical and cannot achieve the benefits of hierarchical routing.

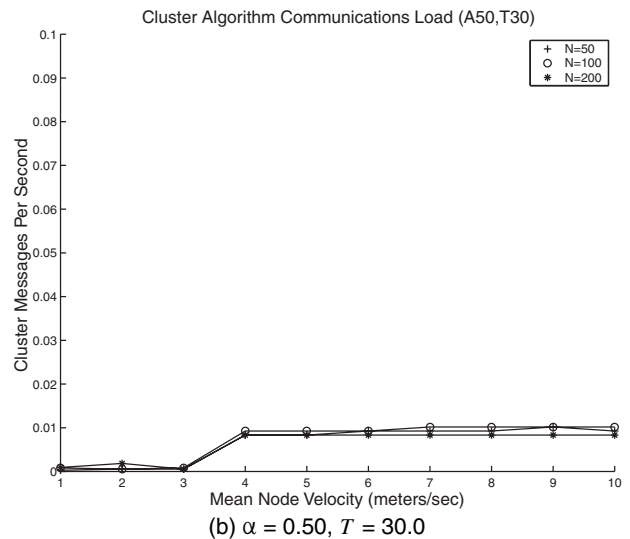
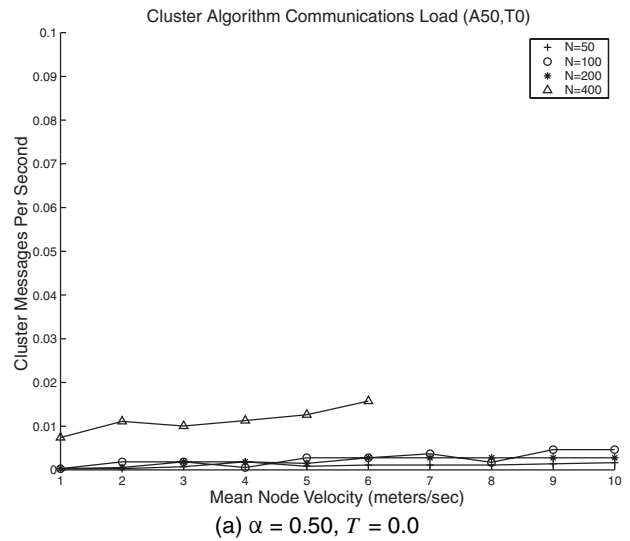


Figure 8. Cluster control message rate as a function of mobility

Complex systems such as ad hoc networks can benefit from analytical theory; however, researchers must apply simulation to study these systems using more realistic scenarios and adequate levels of detail. The development of a simulation model played a crucial role in understanding the clustering and routing algorithms, as well as comparing them to alternative approaches. The model was developed to evaluate the inherent stability and efficiency of the  $(\alpha, t)$ -cluster algorithm. Specifically, the simulation was used to measure the strategy's effectiveness in terms of mean performance metrics, including the probability of a node being clustered, cluster residence time, the routing message-processing load due to proactive routing updates,

and the cluster control message rate. Simulation results demonstrate the efficiency of the clustering algorithm and its ability to adapt to node mobility in effective ways that have a positive effect on network performance.

## 6. Acknowledgments

This work was supported in part by the National Science Foundation (NSF) award No. 0073972, funded by CISE/ANIR.

## 7. References

- [1] McDonald, A. Bruce, and Taieb Znati. 1999. A mobility based framework for adaptive clustering in wireless ad-hoc networks. *IEEE Journal on Selected Areas in Communications* 17 (8): **000-000**.
- [2] McDonald, A. Bruce, and Taieb Znati. 1999. A path availability model for wireless ad-hoc networks. *Proceedings of the IEEE Wireless Communications and Networking Conference 1999 (WCNC'99)*, September, New Orleans, LA.
- [3] McDonald, A. Bruce, and Taieb Znati. 2000. Predicting node proximity in ad-hoc networks: A least overhead adaptive model for electing stable routes. *Proceedings of the First IEEE/ACM Workshop for Mobile Ad Hoc Networking and Computing (MobiHOC)*, August.
- [4] McDonald, A. Bruce, and Taieb Znati. **YEAR?** An analytical framework for computing the probability of link availability for routing in ad hoc networks. *Wireless Communications and Mobile Computing* [Online]. Available: <http://www.ece.neu.edu/faculty/mcdonald/wireless-draft.ps>
- [5] McDonald, A. Bruce, and Taieb Znati. 2000. A dual-hybrid adaptive routing strategy for wireless ad-hoc networks. *Proceedings of the IEEE Wireless Communications and Networking Conference 2000 (WCNC'00)*, September, Chicago.
- [6] Basagni, S., I. Chlamtac, A. Faragó, V. R. Syrotiuk, and R. Talebi. 1999. Route selection in mobile multimedia ad hoc networks. *Proceedings of the Sixth IEEE International Workshop on Mobile Multimedia Communications, MOMUC'99*, November, San Diego.
- [7] Basagni, S. 1999. Distributed clustering for ad hoc networks. *Proceedings of the 1999 International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN'99)*, edited by A. Y. Zomaya, D. F. Hsu, O. Ibarra, S. Origuchi, D. Nassimi, and M. Palis, 310-15. Perth/Fremantle, Australia: IEEE Computer Society.
- [8] Perkins, C. R., and P. Bhagwat. 1994. Highly dynamic destination sequenced distance vector routing (DSDV) for mobile computers. In *ACM SIGCOMM*, pp. 234-44, October.
- [9] Garcia-Luna-Aceves, J. J., and M. Spohn. 1999. Source-tree routing in wireless networks. *Proceedings of the IEEE ICNP 99: 7th International Conference on Network Protocols*, November, Toronto, Canada.
- [10] Murthy, S., and J. J. Garcia-Lunes-Aceves. 1996. An efficient routing protocol for wireless networks. *ACM Balzer Mobile Networks and Applications Journal* **VOLUME:000-000**.
- [11] Haas, Z., and M. Pearlman. 1998. The performance of query control schemes for the zone routing protocol. *Proceedings of ACM Sigcomm'98*, October.
- [12] Pearlman, M., and Z. Haas. 1999. Determining the optimal configuration for the zone routing protocol. *IEEE Journal on Selected Areas in Communications* 17 (8): 1395-1414.
- [13] Vaidya, N. H., P. Krishna, M. Chatterjee, and D. K. Pradhan. 1997. A cluster-based approach for routing in dynamic networks. *ACM Computer Communications Review* 27 (2): **000-000**.
- [14] Lin, C. R., and M. Gerla. 1997. Adaptive clustering for mobile wireless networks. *IEEE Journal on Selected Areas in Communications* 15 (7): **000-000**.
- [15] McDonald, A. Bruce. 2000. A mobility-based framework for adaptive dynamic cluster-based hybrid routing in wireless ad-hoc networks. Ph.D. diss., Department of Information Science, Telecommunications Program, University of Pittsburgh.
- [16] Law, A., and W. D. Kelton. 1991. *Simulation modeling and analysis*. New York: McGraw-Hill.
- [17] Haas, Z. 1997. A new routing protocol for reconfigurable wireless networks. *Proceedings of the IEEE International Conference on Universal Personal Communications (ICUPC)*.
- [18] Krishna, P., et al. **[PLS. PROVIDE NAMES OF ALL AUTHORS]** 1997. A cluster-based approach for routing in dynamic networks. *ACM Computer Communications Review* 27 (2): **000-000**.
- [19] Ramanathan, R., and M. Steenstrup. 1998. Hierarchically-organized, multihop mobile wireless networks for quality-of-service support. *Mobile Networks and Applications* 3:**000-000**.
- [20] Joa-Ng, M., and I. Lu. 1999. A peer-to-peer zone-based two-level link state routing for mobile ad-hoc networks. *IEEE Journal on Selected Areas in Communications* 17 (8): 1415-25.

**A. Bruce McDonald** is **TITLE?** in the Department of Electrical and Computer Engineering at Northeastern University, Boston.

**Taieb F. Znati** is **TITLE?** in the Department of Computer Science at the University of Pittsburgh.