A Delay-aware Clustering Protocol for Wireless Ad Hoc Networks

Jin Sun Kan Liu
Department of Computer Science
University of California
Riverside, CA 92507
Email: {jsun, kanliu}@cs.ucr.edu

April 2005

Abstract

The proliferation of small wireless devices today has made the development of scalable ad hoc wireless protocols critical to their success. This paper presents design and evaluation of a distributed, delay-aware clustering protocol, which forms clusters based on node's *average media access delay*. We also develop intra/inter-cluster routing to utilize the cluster structure, in which cluster heads do not perform any special functionality in data forwarding for its members but take responsibility in control and routing information dissemination. The simulation result shows our protocol outperforms AODV in terms of routing control overhead and network throughput.

1 Introduction

In the past, the design of ad hoc protocols was mostly focused on flat topology without hierarchical structures, like DSR [1], OLSR [2], AODV [3] and etc. A lot of refinements have been proposed based on these protocols. However, most of these protocols can work with only hundreds of nodes and scalability has become more and more a concern for such networks to take off. Besides, the emerging sensor networks [4, 5] which can be envisioned as a spinoff from ad hoc networks also call for scalability.

Many routing protocols have been proposed for ad hoc networks, among which DSR and AODV are most popular ones. Most of the routing protocols are designed for flat architecture. When network size increase, these flat routing protocols become infeasible because the route's hop count will become bigger and thus link breakage will happen frequently. Even worse, if nodes are highly mobile, link failures are much more common. Excessive routing message overhead caused by the increase of network population and mobility will overwhelm the network traffic.

To answer these concerns, a series of scalable routing protocols have been proposed and a survey has been done by Hong et al. [6] in 2002. Most notably, hierarchical structures seem to be the major theme in these protocols where nodes organize themselves recursively based on their proximity to one another.

A number of approaches to scalable ad hoc networks have been proposed in [6–9]. In hierarchical routing, self-organization scheme is employed to group network nodes into clusters or zones. Nodes geographically close to each other are grouped into logical clusters, and different nodes may be assigned different functionalities inside and outside a group. One node inside a cluster may carry out special task in data packet forwarding or providing location and direction service to this cluster. This node will then be called the *cluster head*. Each cluster has one or more cluster heads. Nodes in a cluster can be one or several hops away from the cluster head. Some nodes can communicate with more than one cluster and work as gateways to ensure connectivity between clusters.

Routing within a cluster and routing among clusters may use different mechanisms. With appropriate modifications, some of the "flat" routing protocols can be used in hierarchical scenarios. For exam-

ple, inter-cluster routing can be a proactive protocol, while intra-cluster routing can be on demand [9].

In high-density networks, hierarchical routing protocols tend to achieve much better performance. Because the routing table storage, update packet size and processing overhead can be reduced drastically as a node may only need to know the exact route to destination inside it own cluster or zone instead of the total information about the whole network. Instead of recording route hop-by-hop, hierarchical routing records a route cluster-by-cluster. Apart from making network seem smaller, clustering also makes dynamic topology appear less dynamic due to topology aggregation.

However, the complexity of maintaining the hierarchy may degrade the performance of these routing protocols. The overhead and complexity of hierarchical routing protocols come from the selection and maintenance of the cluster heads as well as address management. Frequent cluster changes can incur prohibitively high overhead, thus compromise its performance. So the clustering algorithm should be able to maintain its cluster structure as stable as possible while topology changes. Unless being intentionally designed so, a node selected as a cluster head may not necessarily have higher processing capability and channel capacity than the other nodes and they tend to become a bottleneck. In short, though hierarchical routing provides a possible approach for scalability, it also faces some possible implementation difficulties.

The rest for this report is organized as follows. Section 2 gives a brief literature review of the existing work on clustering protocls. We try to assimilate those useful ideas while identifying their limitations and this helps our work on advancing dynamic clustering approach. Section 3 presents our delay-aware clustering algorithms. In section 4, we evaluate this algorithms through simulations and analysis. Section 5 concludes this paper.

2 Related works

A number of clustering algorithms have been proposed based on different objectives such as low maintenance, energy efficient, load balanced, and so on. Least Cluster Change (LCC) Algorithm [10] was developed to minimize changes of cluster head. Wu [11] proposed a Connected Dominating Set (CDS) algorithm to reduce the number of nodes involved in routing-related tasks. Adaptive Multihop Clustering [12] algorithm maintains a multihop cluster structure where the number of members in each cluster is not too large, nor too small. MOBIC [13], a mobility-aware clustering algorithm, take a node's speed relative to each of its neighbors into consideration in cluster head election. On-Demand Weighted Clustering Algorithm (WCA) [14] considers multiple metrics for each node in cluster head election procedure.

Previous routing protocol research was focused on layer-3 functionality only, and it seems that the traditional methodology of layered protocol design does not necessarily lead to an optimal solution for wireless networks. Interaction between MAC and routing is so close that merely exchanging parameters between protocol layers is not adequate. Therefore, protocols for different layers need to not only work together but also work collaboratively among themselves. Such interactions require designing protocols in a cross-layer fashion. Merging certain functions of MAC and routing is a promising way. For example, we can adopt multiple performance metrics from layer-2 into routing protocols.

Here we define a new term, *Average Media Access Time* (AMAT), which can be used as a metric during the formation of clusters. On receiving a data packet from its neighbor, a node first checks if it is the desired recipient of this packet, if not, it will put this packet into the transmission queue, and forward this packet if required by the routing protocol. The packet will be kept in this queue until a successful transmission acknowledgement is received after the node sends the packet out. By calculating the average time in the transmission queue for every data packet, the node can get its average media access delay. Average media access delay is the dominant factor that affects network transmission delay and throughput. Intuitively, the leader in a cluster should be the least loaded node, that is, node with the least average media access time.

As mentioned in previous section, different criteria are used for the organization of the clusters and implementation of the distributed clustering algo-

rithms. However, none of the proposed schemes has taken the underlying layer information into account. To be specific, none uses average media access time of nodes as a criterion for cluster organization. In some other cluster head selection algorithm, lowest ID [15] is used to elect the leader for a cluster. Obviously it is not an optimal method. In the clustering process, the Lowest-ID algorithm and its LCC variant [10] do not consider the network throughput. If a node with a lowest ID happens to be heavy-loaded node, it will cause severe delay and contention problem.

3 Delay-aware clustering protocol

The basic assumption we use for mobility pattern is that all the nodes in the network should not be fast-moving ones, otherwise the cluster re-election overhead will be prohibitively high.

A successful dynamic clustering algorithm should achieve a stable cluster topology and maximum transmission throughput. In order to use our metric presented for clustering, we propose a two step distributed clustering algorithm with average media access metric as the basis of level-1 cluster formation instead of node IDs. In the remainder of this section, we describe how the new metric is used in the formation of clusters which are at most 2 hops in diameter. The basic idea is that the clustering process should take into account the load status of individual nodes with respect to its neighboring nodes.

Our algorithm is executed in the following steps.

- First, we group individual nodes into level-1 clusters based on the AMAT metric mentioned above. But a single-level scheme does not scale well when the network grows larger and larger to accomodate thousands of nodes. To dynamically adapt to the topology of the network, a level-1 clustering system should be extended to a multi-levels hierarchy to adapt to the changing conditions of the network, thus further increasing the scalability of the protocols.

- Second, for the upper levels, the hierarchy is formed as a recursive organization of level-1 clusters into level-2 clusters, level-$k-1$ clusters in to level-k clusters, and so on. Based on this definition, we call level-1 clusters as *fundamental clusters*, as at this level, the cluster is composed only of individual nodes. Each kth level cluster is also a node at level-k, and any kth level cluster has associated a level-k leader with it. As the network grows, more hierarchy levels need to be added. Conversely, when nodes leave the network, the hierarchy levels should be reduced.

We use different cluster formation method for level-1 cluster and upper level clusters. The first level cluster formation and cluster head election is based on average media access delay of a node to achieve maximum network throughput. The upper level cluster formation is established above level-1 clusters but the goal is to minimize the effect of one cluster change to other clusters in the network, which is called *rippling effect*. We will describe different cluster formation and election method for level-1 and upper levels respectively as follows.

3.1 Cluster formation and head election

3.1.1 Node status and functionality

The main rules of cluster formation include:

- Each node belongs to exactly one cluster
- Each node can be a head for exactly one cluster
- Two cluster heads can not be neighbors with each other
- Each member node is one-hop away from its cluster head
- Head election is based on the node and its neighbor's *AMAT* value

In our protocol, nodes have three status: cluster head(CH), ordinary member(OR) and Non-clustered node(NON). The Hello message exchanged periodically which includes cluster information, such as AMAT value, node status and cluster ID. Base on these informaiton, each node will update its neighbor table and decide their cluster status distributedly.

There three phases in our protocols: initialization phase, maintanence phase and merging phase, as described in detail in following subsections.

3.1.2 Initialization Phase

During the initialization phase, nodes periodically broadcast hello messages. A hello message includes node's average media access time. It also indicates the presence of the node and its current level status. In the very beginning, each node sets its status as cluster head for itself and AMAT field in hello message is initialized to zero.

After some predefined transmission time T_{init}, during which data packets have been transmitted for a while, each node will begin to update its AMAT according to the accumulative information from its MAC layer's transmission queue. Its AMAT will be included in the subsequent Hello message it broadcasts to all its neighbors.

Through gathering AMAT values from its neighbors, each node knows the local work-load of all its neighbors and compares its own AMAT value with neighbors'. After negotiation with all its neighbors, each node will decide its status independently based on the neighbor information advertised in the Hello message. The node with lowest AMAT value will declare itself as a new leader (cluster head) and other nodes will become ordinary nodes. In case of several nodes are candidates for cluster heads, the node with lowest ID is chosen. Nodes cannot find any neighbors to form a cluster will declare its status as Non-clustered.

3.1.3 Maintenance Phase

Once the clusters have been formed, node will in a distributed way decide whether to join another cluster, based on its own status and the information in the reveived hello message. Each node will maintain a neighbor table, which should contain information as follows:

- Neighbor's ID
- Neighbor's status
- Neighbor's *AMAT* value
- Neighbor's cluster ID

To ensure the cluster head is the least congested node within the cluster, re-election will be performed once every T_d. However, there is one tradeoff here, the reclustering procedure can not happen too frequently: if T_d is set too small, reclusering will happen more frequently and cause more overhead on the network. Similarly, if T_d is set to a large value, the existing cluster head may not be always the optimal choice because of unpredictable traffic patterns. In our simulation, we set $T_d = 6 * HelloInterval$, a heuristic value. After T_d expires, nodes will begin to compare the AMAT value with all the neighbors again to find a better candidate and reset their status. During T_d, reclustering will not happen even though the AMAT value of head is not the least among all its neighbor, unless the AMAT value of the cluster head is at least twice the minimum AMAT value of its neighbors, as show in the equation below

$$AMAT_{head} >= 2 * min\{AMAT_{neighbor}\} \quad (1)$$

By doing so, we try to reduce the control overhead while still ensuring that the overall transmission delay will not become too large.

3.1.4 Cluster Merging Phase

Because of the node's mobility, the network topology will change over time. A node may join or leave an existing cluster at any time due to its movement. There are two situations for cluster merging to take place. First, for the joining node, if it is an ordinary or non-clustered node that request to join a cluster i, no cluster leader change happens in i. If two cluster leaders becomes neighbors, the reclustering pprocedure will be triggered immediately. These two clusters will merge into one and the head with lower AMAT value will remain the cluster head status while the other one will become an ordinary node. If a cluster member finds out that all its other cluster members have left and doesn't have neighbor to form a cluster either, it will claim itself as a non-clustered node.

Second, when a node leaves the cluster which happens to be the cluster header, other members in this cluster will lose the leader and cannot get the advertisements from the cluster header. After a predefined timeout period, they will exchange hello messages and initialize the discovery process again as described previously and a new cluster head will be elected accordingly.

3.2 Intra-cluster routing and Inter-cluster Routing

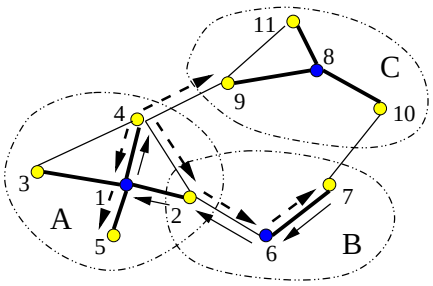
We also develop intra/inter-cluster routing to utilize the cluster structure. Our modification is based on AODV. In order to avoid heavy load in cluster heads, we propose a routing protocol in which cluster heads only take responsibility in control and routing information dissemination for its members and do not perform any special functionality in data forwarding. Also, information in neighbor tables is used to reduce unnecessary routing information transmission. Our routing protocol uses different strategies for intra-cluster routing and inter-cluster routing.

For intra-cluster routing, the cluster heads announce all the member informations in hello messages and thus each node knows the existence of all other members within the same cluster. When a source node S wants to deliver data to an unknown (no match in routing table) node D, S will first check its neighbor table, if there is a match, it simple adds this route into routing table and directly send data to D. Otherwise, S will check it S and D belont to the same cluster or not. If both S and D are members of some cluster, S would broadcast a Route Request (RREQ) within this cluster.

For inter-cluster routing, e.g., the unknown destination D is outside the cluster which S belongs as an ordinary member, the source node S will unicast RREQ to the cluster head CH.

Once CH receives the unicast RREQ, it would check its routing table for an entry to D. If such an entry is found, CH will modify this route and unicast a reply to S, with the next hop field in RREP being the pre-hop node to CHin this route, instead of CH itself. Otherwise, if there is no existing entry for D, CH will broadcast RREQ for D and add the RREQ to its request table. We maintain a request table in cluster head to keep track of each unicast RREQ for its members, with four fields for every entry: RREQ source, Source Cluster ID, RREQ destination and Destination Cluster ID.

On any received RREP, CH will check its request table to see if there is any match for this reply. If no match exists, this means the RREP is for CH itself, CH only need to update its routing table. Otherwise, if any matches found, CH would modify the RREP



If node 4 wants to reach node 7, the inter−cluster routing procedure works as follows:
Step1: node 4 unicast RREQ to node 1
Step2: node 1 broadcast RREQ for node 7
Step3: node1 receives RREP (7, 6, 2, 1)
Step4: node 1 sends RREP (7, 6, 2) to node 4
Step5: node 4 update the route to node 7 (4, 2, 6, 7)

If node 4 wants to reach node 9, the route (4, 9) can be directly determined since they are neighbors.

If node 4 wants to reach node 5, the intra−cluster routing procedure works as follows:
Step 1: node 4 broadcast RREQ for node 5
Step 2: node 4 update the route to node 5 (4, 1, 5)

Figure 1: Normalized Overhead packets per node with different network density

with the information about the next hop, which wil be set to the prehop in this route, instead of CH itself. This modified RREP is unicasted to the source node. Also, CH updates its routing table for D.

On receiving RREP from cluster head CH, S would update its routing table. If the next hop N in the RREP, belongs to the same cluster and is not CH, N will be used as a replying node for data delivery. Thus, we need to concatenate the route to from S to N with the newly received route from N to D as a complete route to the destination D. If no entry for N can be found from the routing table and neighbor table of S, S will broadcast an intra-cluster RREQ for N.

Figure 1 is an example to show how our routing protocol works. Suppose node 4 wants to reach node 5, because both of them are members of cluster A and they are not neighbors, so node 4 would broadcast the intra-cluster RREQ for node 5. If node 4 wants to reach node 9 which belongs to cluster C, node 4 can add an entry for node 9 to its routing table and directly delivers data to node 9 since they are neighbors. Another case is that node 4 wants to reach node 7 which belongs to cluster B, node 4 will first

Table 1: Summary of Simulation Parameters

Parameters	Value in Our Simulation
Number of nodes	50, 75, 100, 125, 150, 175
Rate(pkt/sec)	20
Network size	1500 * 1500 m^2
MaxSpeed	10 m/sec
Simulation time	500 sec
Transmission range	125 m

unicast a RREQ to node 1 (the cluster head of A) and then node 1 will broadcast RREQ for node 7. Once node 1 receives the RREP (1, 2, 6, 7), it would modify the RREP with node 2 being the next hop, and then send this reply (2, 6, 7) to node 4. After getting this RREP, node 4 will use route (4, 2, 6, 7) to node 7 and add it to its routing table.

4 Simulation results and analysis

4.1 Simulation setup

In order to validate our protocol, we implemented our protocol in NS2 simulator with CMU wireless extension, and compare its performance with AODV in terms of network throughput, overhead and average end-to-end transmission delay for each packet.

We evaluate the scalability of our protocol with increasing network size as well as the performance of the architecture under varying node mobility and offered traffic load.

The scenarios were generated with input parameters as listed in the Table. We did our simulations in various network topologies with size ranging from 25 nodes to 150 nodes randomly distributed in a two dimensional topology ($1500 * 1500 m^2$). Each simulation runs for 500 sec. Default parameters in the ns-2 for the wireless MAC and physical layers settings are used in our simulations, that is, 802.11 based 2 Mbps data rate and nominal transmission range of 250m.

- *Mobility Pattern*: We used the random waypoint model to simulate the mobility of the network nodes. Mobile nodes have a maximum speed of 10 m/sec, and an average speed of 5 m/sec, and zero pause time.
- *traffic Pattern*: We used constant bit rate (CBR) flows to generate the offered traffic load on the network. Each flow consists of a randomly chosen pair of source and destination nodes. Each flow lasts 90 seconds and generates 64 bytes packets at a constant rate of 4 packets per second.

The most important goal of our project is to improve the throughput and reduce average transmission delay for data packets for large-scale Ad Hoc networks. In the following sections, we evaluate our proposed algorithms and protocols through simulation and analysis compared with AODV in terms of:

- *Control packet overhead*: The total number of control (non-data) packets transmitted by the protocol. For AODV, we count the sum of RREQ and RREP packets. In our protocol, because the cluster formation is maintained by periodical hello message, which belongs to control packets, we count the total number of RREQ, RREP and Hello message as overhead.
- *Packet delivery ratio*: The ratio of the number of data packets received by the destination and the number of data packets generated by the source nodes.
- *Average data transmission delay*: It is measured as the aggregation of all possible delays, such as the queuing time at the interface queue, retransmission delay at the MAC layer and buffering delay during route discovery etc.

Figure 2 shows the comparative performance of AODV with our scheme in terms of packet overhead versus node density. we notice that our protocol demonstrates significantly lower routing overhead than AODV, usually by a factor of 3-4. This is one of the important foal of clustering protocol when scalability is a major concern. In our scheme, the total number of RREQ and RREP packets was significantly reduced. The control overhead is dominated by the hello messages overhead which are transmitted by nodes periodically, as shown in figure 3. Consider the periodic broadcast of the Hello message, the

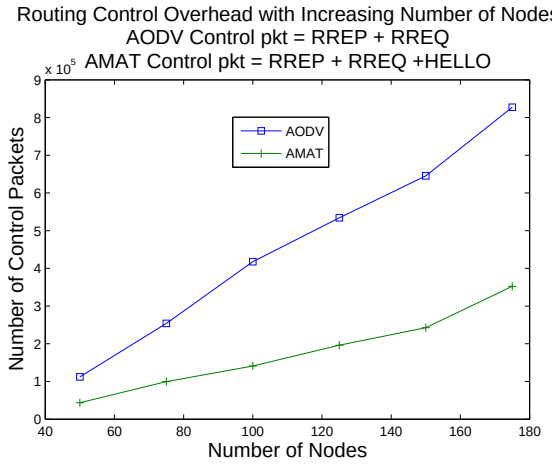


Figure 2: Routign Control Overhead with different number of nodes

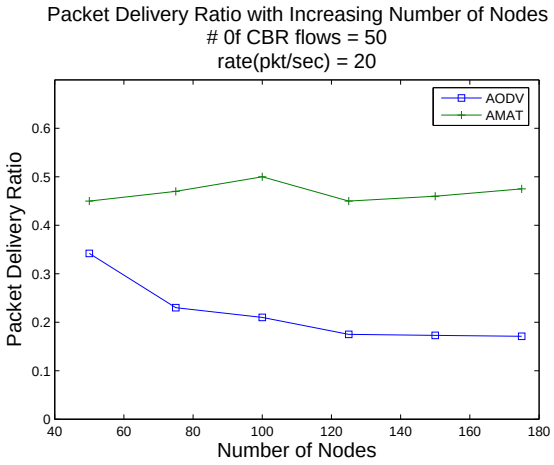


Figure 4: Packet Delivery Ratio with different number of nodes

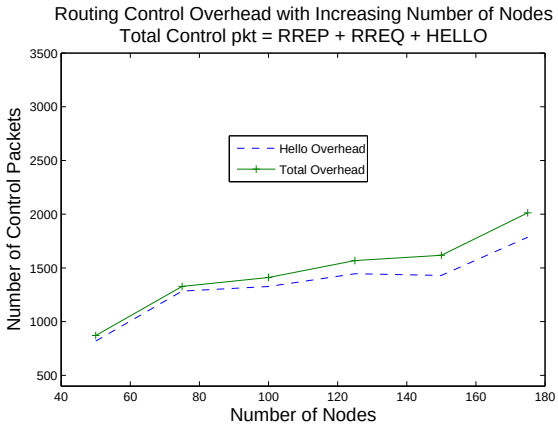


Figure 3: Normalized Overhead packets per node with different network density

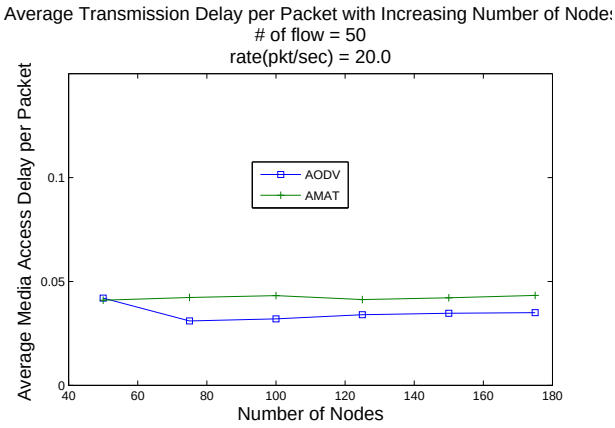


Figure 5: Average End-to-end transmission delay per data packet with different number of nodes

total number of hello transmission per Hello interval is bounded to $\Theta(1)$.

Figure 4 shows the packet delivery ratio (PDR) of two protocols. As the number of nodes increase, the PDR of AODV decreases from 0.34 to 0.17. On the other hand, our algorithm performs better, the PDR stays above 0.45. This shows that our schemes can scale up to larger networks.

Figure 5 shows the comparative performance of the popular ad hoc routing protocol, AODV, with our protocol. We can see that as the total number of nodes increases, the average transmission delay decreases, because more contentions and congestions appear in higher density network. As we can see from fig 5, our protocol has slightly higher the average end-to-end transmission delay by 1-3 ms. because the clustering procedure will introduce some delay and overhead. But we can expect that, if compared with other clustering protocols, such as lowest-ID scheme, our protocol will achieve less delay.

5 Conclusion and future work

In this project we study and analyze some existing hierarchical routing protocols, followed by ideas of our own: a delay-aware level-1 clustering algorithm for wireless Ad Hoc networks, in which node's average media access time is used as metric during cluster

formation. We also propose intra/inter-cluster routing to utilize the cluster structure. We show through detailed analysis that such an ad hoc routing protocol can scale to a large number of nodes and function well in simulations.

Due to the limited time, we only implemented level-1 clustering algorithm. As future work, we are also interested in designing a multi-level hierarchical routing protocol for ad hoc networks, and also take consideration of node load status in level-1 clustering to achieve better throuput and different cluster size.

References

[1] D. B. Johnson and D. A. Maltz, *Dynamic Source Routing in Ad Hoc Wireless Networks*, ch. 5, pp. 153–181. Kluwer Academic Publishers, 1996.

[2] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)." RFC 3626, Oct. 2003.

[3] C. Perkins, E. Belding-Royer, and S. Das, "Ad Hoc On-Demand Distance Vector (AODV) Routing." RFC 3561, Sept. 2003.

[4] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Communications Magazine*, pp. 102–114, Aug. 2002.

[5] C.-Y. Chong and S. Kumar, "Sensor Networks: Evolution, Opportunities, and Challenges," *Proceedings of the IEEE*, vol. 91, pp. 1247–1256, Aug. 2003.

[6] X. Hong, K. Xu, and M. Gerla, "Scalable Routing Protocols for Mobile Ad Hoc Networks," *IEEE Network*, vol. 16, no. 4, pp. 11–21, 2002.

[7] E. M. Belding-Royer, "Multi-Level Hierarchies for Scalable Ad Hoc Routing," *Wireless Networks*, vol. 9, pp. 461–478, Sept. 2003.

[8] I. S. Suli Zhao and D. Raychaudhuri, "Performance and Scalability of Self-Organizing Hierarchical Ad Hoc Wireless Networks." Rutgers WINLAB publications, 2004.

[9] "Self-Organizing Hierarchical Routing for Scalable Ad Hoc Networking." Rice Computer Science Technical Reoprt TR04-433, 2004.

[10] W. L. C.-C. Chiang, H.-K. Wu and M. Gerla, "Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel," in *Proc. IEEE SICON97*, pp. 197–211, 1997.

[11] J. Wu and H. L. Li, "On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks," in *Discrete Algorithms and Methods for Mobile Comp. and Commun*, pp. 7–14, 1999.

[12] S. I. T. Ohta and Y. Kakuda, "An Adaptive Multihop Clustering Scheme for Highly Mobile Ad Hoc Networks," in *Proc. 6th ISADS03*, 2003.

[13] N. K. P. Basu and T. D. C. Little, "A Mobility Based Metric for Clustering in Mobile Ad Hoc Networks," in *Proc. IEEE ICDCSW01*, pp. 413–418, 2001.

[14] S. K. D. M. Chatterjee and D. Turgut, "An On-Demand Weighted Clustering Algorithm (WCA) for Ad hoc Networks," in *Proc. IEEE Globecom00*, pp. 1697–1701, 2000.

[15] J. W. A. Ephremides and D. Baker, "A design concept for reliable mobile radio network with frequency hopping signaling," in *IEEE 75*, 1987.