Design and Optimization of a Cluster Based Ad Hoc Wireless Network Routing Protocol

Daniël Kotze and Riaan Wolhuter
Department of Electronic Engineering
University of Stellenbosch
7602 Matieland, South Africa
Email: 14380552@sun.ac.za, wolhuter@sun.ac.za

***Abstract*—Cluster based routing protocols has been shown to reduce routing information overhead at high node densities. This paper documents the design process of a cluster based routing protocol. A simulation was created to implement the protocol. It is shown how the cost to maintain the cluster structure, can be reduced further by decreasing the number of nodes that transmit routing information. The formation of clusters is illustrated and the roles of different nodes are described, due to their place in the network topology. A mechanism for the fast detection of broken routes is presented. Basic congestion control is implemented with token scheduling. Performance is measured by a separate module for easier aggregate simulation time measurements.**

Index Terms—Ad hoc wireless network, cluster generation, congestion control, fast link break detection, simulation.

I. Introduction

Ad hoc wireless networks are communication networks without an inherent infrastructure. Nodes make use of neighbours to route packets to remote destinations that are multiple hops away. Using neighbours as routers extends the coverage of the network at a low cost. Ad hoc networks are ideal for deployment in rural areas, disaster recovery operations, environmental sensing and military operations [1]. This study focuses on ad hoc wireless networks where nodes have a high available bandwidth and high density, such as would be found operating in the K_a band. The Federal Communications Commission (FCC) has assigned the 57-64 GHz band for unlicensed use [**?**].

Cluster based hierarchical routing protocols are effective at high densities, because routing overhead can be reduced [2]. However maintaining the cluster structure comes at a cost. It will be shown how to reduce the cluster maintenance cost by using less overhead than previous implementations of the Cluster-head Gateway Switch Routing protocol (CGSR) [3], [4].

When network traffic increases, links tend to become saturated degrading the overall network performance. A simple mechanism is used to control congestion. Another drawback of current routing protocols is the slow detection of link breakages, because hello packets are sent periodically at the order of 2 seconds [5]. A link breakage can be detected sooner by sending smaller hello packets at a faster rate.

Background on ad hoc network routing protocols will be discussed first, in Section II. Next, the design steps and implementation environment will be documented in Section III. Section IV will present various tests and results.

II. Background

Routing protocols for wireless networks have different requirements, because nodes normally have limited bandwidth and mobile nodes frequently change the topology [1]. However, for this work, we studied an ad hoc network with high bandwidth and immobile hosts. Routing protocols follow two paradigms. Proactive routing protocols maintain routes to all destinations, while reactive protocols only set up a route when one is needed.

Destination Sequenced Distance Vector (DSDV) is a distance vector routing protocol based on the Bellman-Ford algorithm. With the use of sequence numbers loop-free paths to all destinations are provided [6]. Routing overhead of the DSDV protocol grows by $O(N^2)$ where N is the number of nodes [7]. Therefore, it becomes unsuitable for large networks. Optimised Link State Routing (OLSR) is a proactive link state protocol that provides an effective flooding mechanism with the use of Multi-Point Relays (MPRs) [7].

Dynamic Source Routing (DSR) and Ad hoc On-demand Distance Vector (AODV) are on demand routing protocols. Both rely on flooding Route Request packets for determining a route in the network. A Route Reply packet is generated by the node that provides a route to the destination [7], [8].

Hierarchical routing, groups sets of neighbouring nodes into a cell. A cluster-head is elected that coordinates the cluster. Hierarchical protocols allow for greater scalability. With a high node density, less control overhead is used and routes converge faster. [2], [9], [10]

Cluster-head Gateway Switch Routing (CGSR) is a hierarchical proactive routing protocol. Normal nodes only have to maintain the route to their cluster-head and only the cluster-head transmits routing information. One drawback of the protocol however, is that every node needs to transmit its cluster member table periodically, increasing the overhead. The cluster based protocol favours environments where nodes are not highly mobile. [3], [4], [7]

A. Clustering Algorithms

Mario Gerla et al. presented two methods of how clusters can be formed, namely the lowest-id and highest-connectivity clustering algorithms [4]. According to the lowest-id algorithm, the node in the neighbourhood with the lowest address becomes the cluster-head. With the highest connectivity algorithm the node that is the most highly connected in its neighbourhood, becomes the cluster-head.

Note that the protocol has a rule that states that two cluster-heads may not be neighbours. Nodes in a cluster are at most two hops away from one another, because every member of a cluster is a neighbour of the cluster-head.

B. Cluster Protocol MAC-Layer

First, Time Division Multiple Access (TDMA) was used in a cluster for inter node communication [4] and later token scheduling was implemented [3]. Different clusters used alternate spreading codes (Code Division Multiple Access) to prevent interference from neighbouring clusters. Token scheduling is a polling scheme that allocates transmission time among nodes in the cluster. First the cluster-head generates the token and transmits any data messages in its transmission queue. The token is then passed to a neighbouring node and the node transmits a message. The token is then returned to the cluster-head and the process repeats itself with the token passed to a different neighbour. Lost tokens are regenerated by the cluster-head after a time-out. More transmission chances can be offered to nodes with a higher priority.

C. Cluster Protocol Routing

The DSDV routing protocol is used as a basis for the CGSR protocol [3]. To improve DSDV, the unique property of the cluster protocol that a cluster member is one hop away from the cluster-head is used. This decreases the number of routes to the number of clusters and subsequently the routing update overhead is decreased. Each node however needs to periodically transmit its cluster member table containing the cluster-node associations. When a node has more than one cluster-head, the node is called a gateway. A gateway is used when data is routed from one cluster to another. A distributed gateway forms when nodes are neighbours, but in different clusters. Figure 1 shows a typical cluster network.

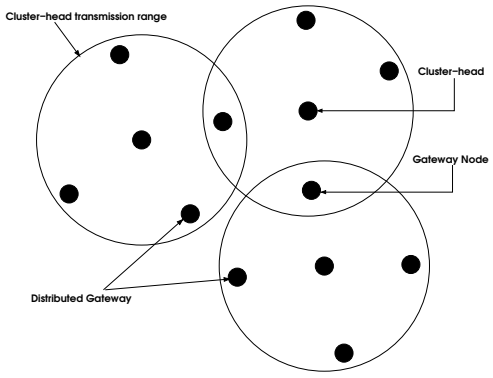


Fig. 1. Nodes grouped together in cluster topology.

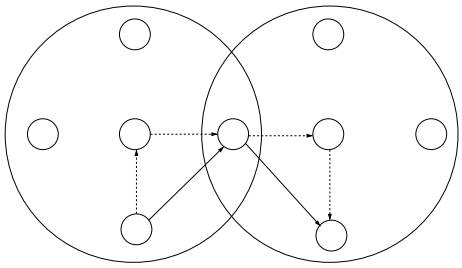
III. Design and Implementation

Due to the favourable advantage cluster based protocols give with regard to decreased routing overhead at high densities, it was decided to use CGSR routing as a base for designing the routing protocol. Note that the study required stationary nodes, so no mobility were allowed for in simulations.

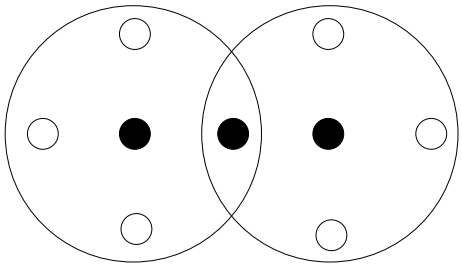
A. Drawbacks of CGSR and Proposed Improvements

The original CGSR has a few disadvantages. Routes are unnecessarily longer and the full cluster-node (cluster member table) does not have to be transmitted by every node. CGSR always routes messages first to its cluster-head and then to the appropriate gateway or destination. However, if the gateway or destination is in range of the original node it can be routed directly to them thus skipping the hop to the cluster-head. To shorten the route, the node handling the message must have knowledge of the next-hop-cluster and which neighbouring gateway nodes link to which cluster. A precursor to CGSR, Destination Sequenced Cluster Routing (DSCR) allowed nodes to make use of the shorter route, but every node had to transmit routing information.

Routing information can be reduced by only letting selected nodes transmit the full cluster-node association along with the route information. Other nodes will only transmit small hello messages containing basic information that share routing information on critical nodes that are at most 2 hops away. Figure 2 shows the proposed changes to the protocol.



(a) The original route chosen by CGSR is shown by the dashed line. The solid line route is the proposed improvement.



(b) Only selected nodes (filled nodes) generate full routing and node-cluster association information for reduced overhead. Other nodes only share information on critical nodes.

Fig. 2. Two possible protocol improvements are indicated.

B. Implementation Environment

The OMNeT++ discrete event network simulator was chosen to simulate the protocol [11]. A Mobility Framework is available for simulating wireless networks [12]. The simulator allows transmission parameters to be set. A wifi (802.11b) transceiver's transmission characteristics were chosen [13].

- Receiving sensitivity, $-85\,\mathrm{dBm}$
- Output power, $15\,\mathrm{dBm}$ ($31{,}6\,\mathrm{mW}$)
- Antenna gain, $2\,\mathrm{dBi}$

C. Routing Layer Overview

For the development of the protocol there are certain logical tasks that must be completed. First, the nodes must

organize themselves into a cluster by electing a cluster-head. After the clusters have been formed, routing information can start to be spread through the cluster. The DSDV protocol will be used as a basis, but only cluster-heads and other selected nodes will forward routing information. Lastly, when the routing information is known, traffic can be generated on the network and analysed.

D. Cluster Generation

The highest-connectivity clustering algorithm was chosen for creating clusters. If the situation arises where two cluster-heads are elected simultaneously within transmission range of one another, the cluster-head with the least connections will give up its cluster-head status. Otherwise, if the number of connections are equal, the cluster-head with the highest address will give up its cluster-head status. [3]

The cluster-head election process will now be described. When the network is started every node has the status of *unassigned*. In the unassigned state, the node continually broadcasts hello messages at a high rate. The hello message contains various data fields. Each hello message has a sequence number incremented with each transmission. Lost hello messages can be determined by inspecting the sequence number. The number of neighbours a node has, is included in the hello message. Each message contains an *election address* field. Every time a node receives a message and the election address is equal to its own address, it increments a counter. If the counter value exceeds a certain threshold, the node is elected as a cluster-head. The threshold can be calculated with the following equation:

$$election\ threshold = k_{threshold} \times connections \qquad (1)$$

The threshold prevents wrong nodes being elected as cluster-heads too quickly.

The node state field in a node's hello message is updated accordingly, when a node becomes a cluster-head. All nodes hearing the newly elected cluster-head change their state to *assigned*, to indicate that they are part of the cluster. Figure 3 shows how a cluster is created.

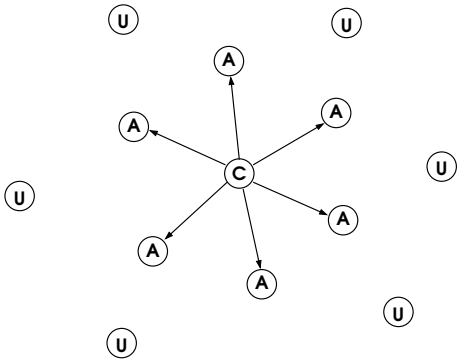


Fig. 3. The diagram illustrates cluster formation. "C" is the *cluster-head* node, "A's" are the *assigned* nodes and "U's" are the *unassigned* nodes.

Nodes that are part of a cluster now generate hello messages at a slower rate to maintain connections with their neighbours.

E. Node States

The protocol makes use of various states. In each state, a node has a different function or behaviour. When a node goes online it starts in the *unassigned* state. In the unassigned state, nodes try to elect a cluster-head. When a node can hear one cluster-head it goes into the *assigned* state. A node that can hear two cluster-heads becomes a *gateway*. Gateways provide a communication path from one cluster to another. A *distributed gateway* is formed between nodes that are in different clusters. *Satellite nodes* are nodes that do not have a cluster-head as a neighbour, but is a neighbour of nodes that can hear a cluster-head.

1) Motivation for Satellite Nodes: Rarely, instances occur where after the election process, a single uncovered node remains at the edge of the network. An uncovered node is a node not associated with a cluster. Only uncovered nodes can take part in cluster-head election and a single uncovered node cannot elect itself. A *satellite node* joins a cluster by relaying its information to the cluster-head through a common neighbour.

F. Routing Tables

The routing tables enable nodes to determine the next hop to a destination. The main routing tables are the neighbour table and the cluster table set. The cluster table set contains all nodes in the network's cluster-node associations and routing information to the specific cluster. Hello messages update the neighbour table and cluster table messages update the cluster table set.

Other tables assist the protocol by maintaining specialised information. The cluster-head table maintains information on cluster-heads within the two hop vicinity. The gateway table summarizes information on neighbour gateway or distributed gateway nodes and to which cluster they link. Routing information on satellite nodes within two hops is recorded in the satellite node table. Only cluster-heads maintain the neighbour cluster table that selects a designated gateway for each neighbouring cluster. The network node table contain all routable nodes and to which clusters they belong.

1) Cluster Table Set Updates: The cluster table set consists of cluster tables. Each cluster table represents a different cluster in the network. Similar to DSDV [6], when a cluster table message (routing update) is received, a cluster table inspects its sequence number to determine if it must be updated. A cluster table in the set with sequence number s_L will be updated if the sequence number in the cluster table in the update message s_R is greater. If however $s_L = s_R$, the cluster table in the set is only updated if the the cluster table in the update message has a smaller hop count.

G. Link Stability

Entries in the neighbour table and the cluster table set both have a TTL (Time-To-Live) field. When a node receives a hello message from a neighbour, a TTL field associated with the neighbour is set to its maximum 5. As long as the TTL is more than 0 the link state is *stable*. After a time nearly equal to the generation time of a hello message, the TTL field of every neighbour is decremented. The generation time of a hello message is $0{,}1\,\mathrm{s}$. Therefore, if no hello message is received from a neighbour, the TTL field will reach 0 after $0{,}5\,\mathrm{s}$. The link state is set to *unstable* if the TTL becomes 0.

Note that the hello message is kept small by only including information on critical nodes (maximum 4 of each type) within a two hop range. Critical node types include cluster-heads and satellite nodes.

H. Routing Information Distribution

Only selected nodes generate full routing information. Cluster-heads and forwarder nodes transmit cluster table messages. Forwarder nodes are selected by the cluster-head and are included in the header of the cluster table message. When a node is informed by a cluster table message that it is a forwarder node, it acquires a forwarding responsibility. Designated gateways indicated in the neighbour cluster table and the satellite service node (node between a cluster-head and satellite node) are selected as forwarder nodes. When a forwarder node receives a cluster table message and the header indicates that it is no longer a forwarder node, the node is relieved of its forwarding responsibility. Figure 4 shows how routing information is distributed.

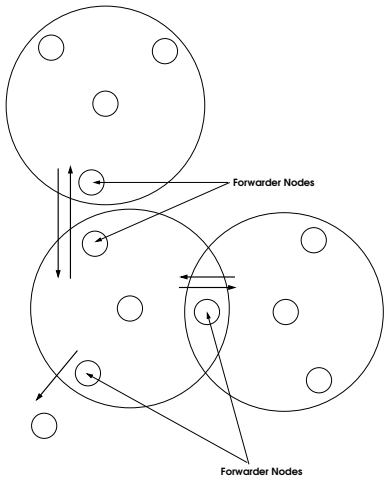


Fig. 4. The routing information is distributed to neighbouring clusters and *satellite nodes* through the forwarding nodes.

I. Routing Function

The routing function has two main parts. One part determines if the destination is available locally. The other part determines the next hop if the destination node is part of a remote cluster.

The neighbour table and the satellite node table are inspected first to find the destination's routing information. If the destination is found in the neighbour table, the destination is set as the next hop address. If the satellite node table contains the destination, an intermediate node is set as the next hop.

Now, if the local search has failed to deliver a result, the network node table is searched for the destination. If the node is found in the network node table, the cluster table set is utilised to determine the nearest cluster to which the destination belongs. The cluster table containing the destination node, contains a *reachable through cluster* field. The gateway table is searched for a gateway that links to the *reachable through cluster*. If such a gateway can be found, it is selected as the next hop, otherwise the cluster-head of the *current cluster* is selected as the next hop. Figure 5 explains the terms *reachable through cluster* and *current cluster*.

If the current node is a satellite node, the satellite service node is selected as the next hop.

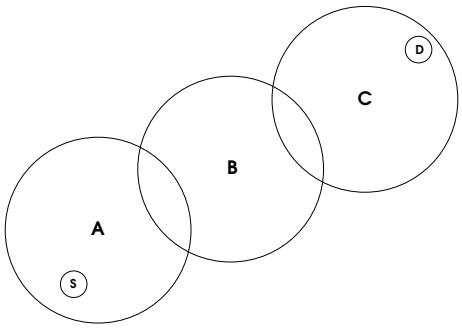


Fig. 5. Suppose a node "S" wants to route data to node "D". From the perspective of node "S", cluster "A" is the **current cluster**, cluster "B" is the **reachable through cluster** and cluster "C" is the **destination cluster**.

J. Token Scheduling

Token scheduling [3] was implemented as a measure of congestion control. Since cluster-heads transmit more routing information and nodes choose the cluster-head as a next hop if no shorter route can be found, cluster-heads are given more transmission chances. Note that different spreading codes for separate clusters were not used, as in the original CGSR, to simplify implementation of the protocol. The focus of token scheduling is to give a fair chance to each node to transmit data in the cluster. The token is generated by the cluster-head and passed back and forth between itself and its neighbours. Every time a token is received, the node can transmit one data message. The token can be regenerated after a time-out. The time-out is calculated by the following equation:

$$t_{timeout} = 2 \times t_{token\ transmission} + t_{data\ transmission} \quad (2)$$

If nodes have more queued traffic, they can be given more transmission chances in a transmission cycle. The number of transmission chances is equated with the following equation:

$$\textit{transmission chances} = \frac{m_i}{\max(1, \min(m_1, m_2, ..., m_n))} \quad (3)$$

Where m_i is the number of messages queued at a node and $m_1, m_2, ..., m_n$ are the messages queued at each cluster member.

Tokens are passed implicitly to satellite nodes.

K. Weighted Fair Queueing with Message Buffers

Weighted fair queueing [14] was implemented to prevent individual nodes from becoming congested. Two buffers are used to queue messages, namely the personal message buffer and general message buffer. The personal message buffer stores all messages originating from the node itself. All forwarded messages enter the general message buffer directly. When a token message is received, one data message from the general message buffer is sent. Messages proceed from the personal message buffer to the general buffer to maintain an acceptable ratio of forwarded messages to generated messages in the general message buffer. For each cluster a

node which is part of the cluster has a personal and general message buffer set. Different message buffer sets allows flows to different clusters to be separated. Routing packets receive preference (hello message and cluster table message) and are passed directly to the MAC-Layer for transmission.

L. Performance Monitoring

It was decided to create a separate module in the simulation for performance monitoring. This approach keeps the performance monitoring and simulation code separate for greater clarity. Simulation nodes have write access to the performance module to record certain values. At the start of the simulation, each node registers with the performance module by giving access to important data structures, for instance, the routing table. A *graphviz* (drawing tool for graphs) source file is generated by the performance module and compiled with the *neato* program [15] to generate a graphic of the network. The configuration is shown in Figure 6.

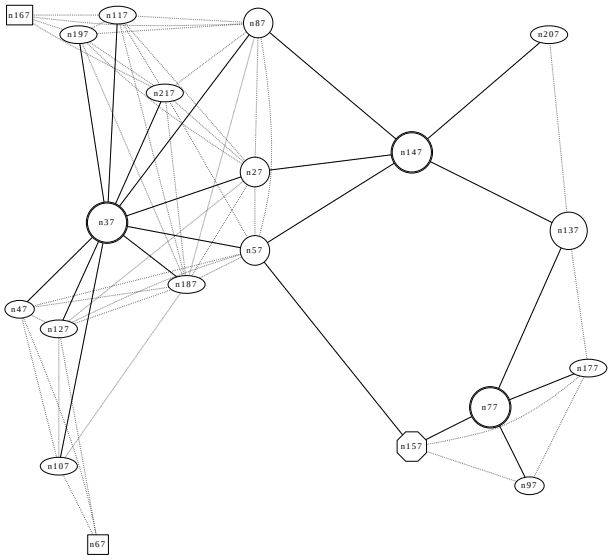


Fig. 6. Big circles represent cluster-heads, gateways are circles, ellipses indicate assigned nodes, distributed gateways are octagons and satellite nodes are rectangles.

IV. Tests and Results

Network setup tests were performed on various different randomised networks. An aggregate connectivity test was performed on the network shown in Figure 6. A test to show the scalability of the protocol at high node densities was conducted and the results are shown in Figures 8 and 9. Note that the hello message generation time is $0{,}1\,\mathrm{s}$ and the cluster table message generation time is $0{,}5\,\mathrm{s}$. Each simulation used 20 nodes with a transmission range of approximately $40\,\mathrm{m}$ bounded by an area of $90\,\mathrm{m}$ by $90\,\mathrm{m}$.

The network setup tests show that the cluster generation algorithm is successful. The results of 9 simulated random networks are given in Table I. For each network, the number of clusters, the average cluster size, number of forwarder nodes, the percentage of nodes that transmit full routing information and the last cluster-head setup time were determined. The number of clusters range from 2 to 4. By only letting cluster-heads and forwarder nodes transmit full routing information, the number of nodes generating full control overhead can be reduced at least to $40\,\%$. Cluster-heads are all elected within $1\,\mathrm{s}$.

TABLE I
Network Setup Test Results.

Simulation Number	Number of Clusters	Average Cluster Size	Forwarder Nodes	Percentage Full Routing	Last Cluster Election Time (s)
1	4	8	7	0,35	0,624
2	2	11	4	0,2	0,403
3	3	10	6	0,3	0,537
4	3	8,33	5	0,25	0,402
5	4	7	6	0,3	0,446
6	2	12,5	4	0,2	0,575
7	2	12	6	0,3	0,406
8	3	8	6	0,3	0,535
9	3	7,33	8	0,4	0,334

The aggregate network connectivity shows the percentage of nodes that are routable for each node in the network. Thus an aggregate network connectivity of 1 indicates that every node knows how to reach every other node in the network. Figure 7 provides the aggregate network connectivity at specific points in time. At approximately $1{,}5\,\mathrm{s}$ the network reaches an aggregate connectivity of 1.

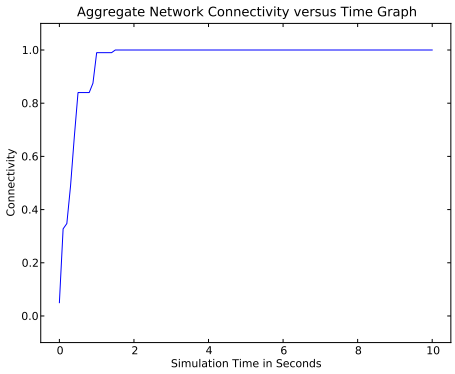


Fig. 7. Aggregate network connectivity versus simulation time graph.

To test network scalability, a test was conducted by increasing the node density in the $90\,\mathrm{m}$ by $90\,\mathrm{m}$ area. First we started with a 15 node simulation and gradually increased the number of nodes by 5 with each $30\,\mathrm{s}$ simulation run. The routing overhead of the cluster table messages and hello messages per node was measured in bytes and shown in Figure 8. The total control overhead and token control overhead is shown in Figure 9. The hello message control overhead per node remains fairly constant over the increasing node density, but it is noted that to send the hello messages frequently, has a high cost. The high cost is allowable, because of the high available bandwidth and it keeps the local routing information up to date. However the cluster table message overhead per node decreases slightly with increasing density, as expected. The token control messages make up the most of the control overhead and also decreases with node density. The less token control overhead per node simply means that every node in the network has less transmission chances as the node density increases.

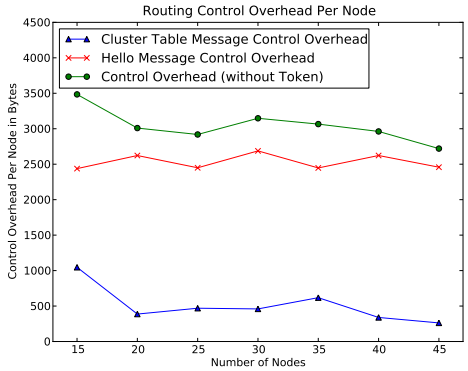


Fig. 8. Routing Control Overhead.

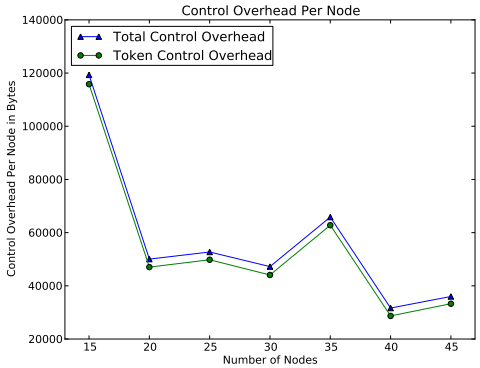


Fig. 9. Total Control Overhead.

V. Conclusion

A routing protocol for high density wireless ad hoc networks was investigated and background given on previous work which proved that cluster based routing protocols provide less control overhead for nodes at high densities. From this point of departure, the design process of an improved cluster based protocol was demonstrated. Two improvements to the CGSR protocol were stated and implemented in a simulated environment. Cluster generation, node states and routing tables of the protocol were discussed. A mechanism for the fast detection of link breakages by sending frequent small hello messages was introduced. The distribution of routing messages by selected nodes was shown. A routing function for the determination of a route to local and remote nodes was discussed.

Simple congestion control based on a token scheduling scheme was implemented, giving preference to nodes with more traffic. Congestion at individual nodes is prevented with weighted fair queueing by separating the forwarded traffic and the traffic generated by the node itself.

A separate performance module was implemented that allows aggregate simulation time measurements of parameters.

Lastly, tests and results were presented. The first test proved that clusters can be successfully set up and the possibility of reducing the number of nodes that transmit full routing information, ie. node-cluster associations and route information. A second test further importantly showed that an aggregate connectivity of 1 can be achieved and therefore, every node knows how to reach every other node.

Another test was conducted to show how the protocol scales in a high density environment. It showed that the cluster table message overhead per node, that contains the most routing information, decreases slightly with a higher density. The hello message overhead per node remains the same for increasing node density. The results show that the control overhead scales well with greater density and does not grow out of bounds. The only disadvantage is that each node gets less transmission chances with the higher node density. This work demonstrated the feasibility of the proposed improvements and desirability of further development and practical implementation.

References

[1] P. Mohapatra and S. V. Krishnamurthy, Eds., *Ad Hoc Networks, Technologies and Protocols*. Springer, 2004, ch. 1, p. 2.
[2] I. F. Akyildiz and X. Wang, “A survey on wireless mesh networks,” *IEEE Radio Communications*, Sep. 2005.
[3] C.-C. Chiang, H.-K. Wu, W. Liu, and M. Gerla, “Routing in clustered multihop, mobile wireless networks with fading channel,” 1997.
[4] M. Gerla and J. T.-C. Tsai, “Multicluster, mobile, multimedia radio network,” *Wireless Networks*, vol. 1, pp. 255–265, 1995.
[5] T. Clausen and P. Jacquet, *Optimized Link State Routing Protocol (OLSR) RFC 3626*, 2003.
[6] C. E. Perkins and P. Bhagwat, “Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers,” *SIGCOMM*, vol. 8, no. 94, 1994.
[7] M. Abolhasan, T. Wysocki, and E. Dutkiewicz, “A review of routing protocols for mobile ad hoc networks,” *Ad Hoc Networks*, vol. 1, no. 22, June 2004.
[8] D. B. Johnson and D. A. Maltz, “Dynamic source routing in ad hoc wireless networks,” 1996, will appear as chapter in the book Mobile Computing.
[9] A. Iwata, C.-C. Chiang, G. Pei, M. Gerla, and T.-W. Chen, “Scalable routing strategies for ad hoc wireless networks,” *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, vol. 17, no. 8, Aug.
[10] C. S. R. Murthy and B. S. Manoj, *Ad Hoc Wireless Networks, Architectures and Protocols*, 2004.
[11] J. Banks, I. John S. Carson, B. L. Nelson, and D. M. Nicol, *Discrete-Event System Simulation*, W. J. Fabrycky and J. H. Mize, Eds. Prentice Hall, 2001.
[12] M. Löbbers and D. Willkomm, *A Mobility Framework for OMNeT++ User Manual Version 1.0a4*.
[13] Wireless 802.11b transceiver. [Online]. Available: http://goods.us.marketgid.com/goods/19387/
[14] L. L. Peterson and B. S. Davie, *Computer Networks*, 3rd ed., R. Adams, Ed. Morgan Kaufmann, 2003.
[15] S. C. North, *Neato Users Manual*, April 2004.

Daniël Kotze completed his Baccalaureate in Electronical and Electrical Engineering in 2007 at the University of Stellenbosch. He is currently busy with his Masters degree at the same institution. His thesis is about ad hoc wireless networks with a high node density and high available bandwidth. Research focus: Ad hoc wireless networks, signal processing (echo hiding in sounds).

Riaan Wolhuter has a B.Sc. B.Eng, M.Eng and a Ph.D. from Stellenbosch University, and a B.Sc(Eng)(Hons) from the University of Pretoria, in Electronic Engineering. He is a senior researcher at the Faculty of Engineering at Stellenbosch University, South Africa.