

A Quality of Service Routing and Distributed Clustering Framework in Mobile Ad Hoc Networks

Mohit Garg

Guide: R. K. Shyamasundar

Abstract—In this document, we present a completely distributed algorithm for partitioning a given set of mobile nodes into clusters. The proposed algorithm tries to reduce the amount of computational and information overhead while maintaining a stable cluster formation. The algorithm is a distributed version of the *Basic Leader-Follower Algorithm* used in pattern recognition. Our algorithm does not require maintaining a cluster head. We also propose a modified Quality of Service routing algorithm for a partitioned (clustered) set of mobile nodes in an ad-hoc network. The proposed algorithm works on the basis of an *acquaintance* based approach in which temporary contacts enable fast inter-cluster routing.

Index Terms—Quality of Service, Mobile Ad-Hoc Networks, Distributed Clustering, Distributed Basic Leader-Follower Algorithm, Acquaintance-based approach.

I. INTRODUCTION

MOBILE Ad-Hoc Networks (MANets) were primarily researched for their use in military applications. But with the burgeoning need for more and more services for the users, MANets have attracted a lot of industry interest recently. Ad-Hoc networks are wireless networks which do not require any fixed infrastructure. They can be quickly set up without the need for any base station or access points as in the case of traditional wireless networks. The temporary nature of Ad-Hoc networks makes them ideal for applications like military communication, disaster relief work, conferences etc.

An ad-hoc network is usually modeled by an undirected graph with each vertex representing a node in the network. An edge between two vertexes implies that the two corresponding nodes can communicate with each other (*i.e.* each link is bi-directional). Such nodes are called *neighbours*. Since the transmission range of each node is limited, this puts a limit on the maximum length of each edge in the graph (if the graph is drawn to scale). Thus a node **A** may be able to communicate directly (neighbour) with another node **B**, but not so with a node **C**. So, in order for **A** to communicate with **C**, it will need to route the packets via another node or set of nodes. This is termed as *multi-hopping*.

A. Routing in MANets

Finding and maintaining stable routes in an ad-hoc network is not an easy task. The major stumbling block is the mobility of the nodes. Almost all the approaches in literature can be broadly classified into the following two categories :-

- *Pro-Active routing* involves keeping routes to all possible destinations (which is usually all nodes in the system) and keeping a track of the link parameters so as to find QoS enabled paths quickly as and when needed by an application. These techniques require an overhead for maintaining and exchanging information about the state of the network. [10], [11]
- *Reactive routing* on the contrary tries to find paths on demand. *i.e.* as and when needed by an application, the source node will initiate a route discovery. These algorithms involve relatively lesser overhead for exchanging information about network state but are characterized by larger delays in finding paths. Flooding algorithms can also be clubbed with reactive routing algorithms. [12], [13]

Both pro-active and reactive approaches are not scalable. In a large network, while the former would need a very large overhead to exchange network state information (which will compete with the data packets for the already scarce bandwidth), the latter will lead to large delays especially when the source and destination nodes are separated by a large no. of hops. Clearly, some sort of a compromise needs to be achieved between the two strategies. *Clustering* strategies tend to do just that. Clustering divides the given network into discrete partitions. Most of the cluster-based routing algorithms tend to use pro-active approaches within the cluster and reactive routing for inter-cluster routing [1], [5]. This enables one to reach a sort of compromise between pro-active and reactive routing strategies. Our approach also uses the same routing strategy with a slight addition (see Section IV). Clustering also provides one with some sort of a veil w.r.t. the mobility of a node *i.e.* a node in cluster **X** does not bother about the motion of a node in

cluster **Y**. Only nodes in the respective clusters update their own links and routes when a node (which is in their cluster) moves. We intend to discuss clustering in detail in section III.

The remaining part of this document is organized as follows: In Section II we discuss the significant works done in the field of cluster-based routing in Mobile Ad Hoc Networks. In Section III we present our clustering algorithm and discuss the related issues. Section IV discusses our *acquaintance-based* approach to route packets in our cluster-based system. The results of the clustering technique are in Section V. Finally, we present the conclusions of our work in Section VI.

II. RELATED WORK

Cluster formation in Mobile Ad-Hoc Networks have attracted considerable attention in recent times [5]–[8]. Most of the current approaches are centered around the idea of a clusterhead which is entrusted the task of controlling the cluster nodes and maintaining the cluster. This leads to unnecessary burden on the clusterhead which itself has limited resources. Also, since most of the routing decisions are taken by the clusterheads and they also serve as routers for the packets [17], these nodes may become bottlenecks in the system. Our scheme does not involve clusterheads and hence does not suffer from this drawback.

Some literature is also available for clusterhead-less strategies. Krishna *et al* [9] propose a clusterhead-less scheme with overlapping clusters. A drawback in this algorithm is that every new node leads to formation of new clusters and rearrangement of old ones which may lead to too many changes in a dynamic network. Also, the algorithm requires a three step procedure to update the clusters each time a change in the local topology occurs. Our proposed scheme does not require any iteration and converges fairly rapidly.

Lin and Gerla [2] present a simple and elegant distributed algorithm for partitioning the network into clusters (without the need of a clusterhead) with each cluster being at most 2-hops wide (*i.e.* each node in a cluster is at most 2 hops away from any other node in the same cluster). This algorithm assumes that each node knows *all* of its immediate neighbours at the time of cluster formation. Our algorithm makes no such assumption.

McDonald and Znati [1] propose a novel prediction based clustering strategy in which they use previous information to predict the motion of each node in a cluster and quantify the stability of a cluster based on it. Their algorithm does not need any clusterhead and they term it as (α, t) cluster. Their algorithm is computationally expensive and may be a drain on the

scarce resources of the mobile nodes. Our algorithm in contrast, does not try any prediction based approach in order to keep things simple and easy to implement. Perkins and Bhagwat note in [11] that “*Simplicity is one of the primary attributes which makes any routing protocol preferred over others for implementation in operational networks.*” Thus, our main goal is to simplify the clustering process and still provide a stable backbone for the network routing protocols. Next we present our clustering algorithm.

III. CLUSTERING

In Mobile Ad-Hoc Networks, mobility causes a lot of problems and link breakage may lead to invalidation of existing paths. This leads to problems in coping up with Quality of Service assurances. One could try and work around the mobility problem in two ways :-

- By using the mobility to our advantage *i.e.* increased throughputs for mobile networks have been reported in [23] but they are only confined to non-real time applications till now and hence, are not applicable in QoS frameworks.
- By trying to achieve some sort of stability and reducing the cascading effect of node mobility *i.e.* only large scale mobility of nodes be visible to the nodes far away in the network.

The latter motivates one to look for clustering techniques. Since an Ad-Hoc network is a completely distributed network with no central authority, it makes sense only to look for algorithms which work in a completely distributed manner. We propose an algorithm which in essence is the distributed version of the *Basic Leader-Follower Algorithm* used in pattern recognition [3]. We call it the *distributed-BLF (d-BLF) Algorithm*. The phrase “Leader-Follower” is misleading here in the sense that our clustering strategy does not require one to maintain a clusterhead or leader. We divide the given network into a set of non-overlapping clusters called *families*. Members of the same cluster (family) are related to each other and hence are termed *relatives*. Also, we can identify border-nodes *i.e.* nodes which can hear messages of more than one clusters. These are termed as *friends*, since they will help in finding paths across clusters. *Acquaintances* are nodes in other families to which a particular node has communicated in the not so distant past. The use of these terminologies is to provide an intuitive feel to the routing algorithm and it will become clear when we discuss routing in Section IV.

We emphasize that the clustering mechanism is completely independent of the routing algorithm employed.

TABLE I
BASIC LEADER FOLLOWER ALGORITHM

1) begin initialize θ
2) $\mathbf{c}_1 \leftarrow \mathbf{x}$
3) do accept new \mathbf{x}
4) $j \leftarrow \arg(\min_i \ \mathbf{x} - \mathbf{c}_i\)$
5) if $\ \mathbf{x} - \mathbf{c}_i\ < \theta$
6) then add \mathbf{x} to c_j and update c_j
7) else add new $\mathbf{c} \leftarrow \mathbf{x}$
8) until no more clusters
9) end

Any routing algorithm which can work on a partitioned network should in principle be able to work with any clustering algorithm. Nevertheless, we present a clustering algorithm and our *acquaintance-based* routing framework which should be expected to work efficiently in a clustered environment.

A. Cluster Formation

The basic leader-follower algorithm used for clustering in pattern recognition systems, is ideally suited for Mobile Ad-Hoc networks since it is an online algorithm (*i.e.* it dynamically forms new clusters as and when data points come up) and is a part of the large class of Unsupervised Learning algorithms [3] *i.e.* no central control is necessary in order to implement it.

Table I lists the steps involved in clustering the data points into clusters. Note that this algorithm is single pass and does not require iteration and hence converges much faster than many other clustering algorithms. In order to adapt it to our case, we need to :-

- 1) First define the metric on the basis of which we will be adding nodes to clusters and the threshold involved. Note that the standard BLF algorithm uses the Euclidean distance as the measure of closeness since it is only concerned with the static case. We will need to do better than this. (Stability Metric)
- 2) Second, we will need to adapt the algorithm so that clustering can be performed in a distributed manner and each node on the basis of local information can join the cluster that leads to a “globally best” partition. (Distributed-BLF Algorithm)

1) *Stability Metric*: Optimality of the algorithm can be derived from the optimality of the BLF algorithm but that will be valid only for the static case, *i.e.* the network topology being fixed and nodes not moving. The notion of “optimal clustering” for a dynamic topology is difficult to define and hence, the cluster formation and stability will be determined by the metric that we choose in the BLF algorithm. It should be clear that clusters in

which nodes are mobile are less likely to remain stable for longer times than those in which they are stationary. Hence, each cluster should have some measure of node mobility in it. One could attempt a prediction based metric (as in [1], [5]) but all such algorithms, apart from being too complex, tend to be computationally prohibitive and hence are not likely to be used in actual implementations.

A compromise can be reached if each node monitors the number of new nodes in the cluster *i.e.* the time for which they have been up – new nodes can be classified as transient (liable to move away soon) and those which are old can be classified as stable nodes. Thus each node is initially kept in a transient state (for $T = T_{transient}$). The motivation behind this is that newer nodes are more likely to move away in the near future (since the node may be mobile). *The fraction of the nodes in the cluster which are stable determine the stability of the cluster and is taken as the stability metric in our approach.*

Thus, this metric is known to all nodes in the cluster and an incoming node can inquire about this from them and make a decision about its joining that particular cluster or not. We wish to emphasize that this metric is only used for an incoming node that wishes to join the clusters. Cluster disintegration does not take place if the above mentioned stability metric goes below a threshold. The exact conditions and mechanism of cluster disintegration is discussed later.

2) *Distributed BLF Algorithm*: We now present the d-BLF algorithm. We assume that there exists a MAC protocol which takes care of collisions and that a channel access policy is in place. We also limit the size (number of nodes) of each cluster to be less than *MAX_CLUS_SIZE* and larger than *MIN_CLUS_SIZE*. This keeps a check on the number of nodes in a cluster. The lower limit is not a strict limit *i.e.* there may exist clusters smaller than that size but they will tend to disintegrate.

Each node on Switch-ON is currently not a member of any cluster. It periodically broadcasts a *CLUS_FIND* message which if heard by a node which is currently a member of a cluster it sends back a *CLUS_INFO* reply containing the cluster name, the number of members in the cluster and its stability metric. The new node, keeps accumulating all such messages for a given amount of time, chooses the cluster which it wants to join and sends back an *CLUS_JOIN_REQUEST* message. The cluster node which had earlier sent the *CLUS_INFO* message, on receiving the *CLUS_JOIN_REQUEST* message adds the node to the cluster, sends *CLUS_JOIN_ACCEPT* message back to the requesting node and broadcasts the

updated routing table within the cluster. The node now becomes a member of the cluster and is included in all future routing table updates. All the above message exchanges will be accompanied by timeouts so as to prevent a node from waiting ad infinitum for a particular reply.

The new node will select the cluster based on the stability metric and the size of the clusters. The primary criteria will be the largest stability metric (greater than the threshold: *STABILITY_THRESH*) but in case the cluster already has a size equal to *MAX_CLUS_SIZE*, it will proceed with other options. An exception to the above rule will be when the node finds a cluster with $\text{size} = \text{MIN_CLUS_SIZE} - 1$. In this case addition of this node to that cluster will make it a stable cluster and hence it will join this cluster if its stability metric is above the threshold irrespective of the stability metrics of other larger sized clusters. A tie in the exception can be broken by the larger stability metric criteria.

In the above scenario, it is possible that the node – a) does not find a cluster around it, b) does not find one satisfying both the stability and size criteria. In both these cases the node forms its own cluster. In this case, its size is less than *MIN_CLUS_SIZE* (if $\text{MIN_CLUS_SIZE} > 1$) and it keeps working according to cluster disintegration mechanism as explained in subsection III-C.

At every point in time, every cluster has an ID which is same as the ID of the lowest ID member of the cluster. In case, the cluster ID changes, each border node broadcasts a *CLUS_ID_CHANGE* message so as to inform the neighbouring clusters about the change. This message is propagated upto the 2-cluster-hop clusters.¹

B. Cluster Maintenance

There are four topology changes that will require cluster updation:-

- 1) *A Node Switches ON*: This will be handled in exactly the same way as cluster formation. This case includes the case when a new node comes up which is not currently a member of any cluster.
- 2) *A Node Switched OFF (or fails)*: This may lead to one or more links inside a cluster being broken. The nodes neighbouring to the failed node will detect the link loss and will send the routing table updates inside the cluster. Every node will update their respective tables and will find that

¹We only let this message to go upto the 2-cluster-hop clusters since each border node of a cluster keeps a track of only its adjacent clusters, and clusters adjacent to the adjacent cluster. This will become clearer when we discuss the routing protocol.

no intra-cluster path exists to this node. Hence, it will be deleted from the cluster and the cluster parameters (*i.e.* cluster ID, stability metric, size) will be updated. In case the failed node was a border node, the nodes in the adjacent cluster will also note a loss of link to it and update their respective routing tables.

- 3) *A new link (with a node already bound to a cluster) is detected*: This means that an existing node (say, Node **A**) has moved into the hearing range of another node (Node **B**). Both nodes **A** and **B** will find out the cluster affiliations of each other, update their own routing tables and broadcast this information in their respective next routing table updates. Note that both these nodes may now become border nodes (if both of them are in different clusters).
- 4) *A link loss is detected*: This scenario is characterized by a node not receiving the routing table update from its neighbour for a particular interval of time. It will then, wait for another larger interval of time and declare the link as lost if it does not hear from the node again. The next routing update will reflect this new information. Note that a link loss may also occur between border nodes which can again be detected by a border node not receiving any adjacent cluster routing table update² from its neighbour in an adjacent cluster.

In all the above cases, each node in the cluster on receiving information about the change, will update the cluster parameters (*i.e.* cluster ID, stability metric, size etc.) independently of each other.

One can see that the above algorithm inherently assumes that the topology changes are “slow enough” so as to allow the updates due to an earlier topology change to settle down. Such an assumption is inherent in any pro-active algorithm and in case the rate of topological changes in the network are too high, flooding and reactive algorithms may be the only option left.

C. Cluster Disintegration

In order to push the network towards forming larger and hence lesser number of clusters, we keep a minimum limit on the cluster sizes (*MIN_CLUS_SIZE*). Thus clusters with sizes less than the minimum size are unstable and will tend to disintegrate. Note that this is the **ONLY** condition under which cluster members are allowed to defect from the cluster in case they

²Each border node keeps a routing table for the adjacent cluster also since it can already hear the routing updates of its neighbours in adjacent clusters. This will be discussed in Section IV

find better and larger clusters to join. Clusters do not disintegrate even if their stability metrics go below the *STABILITY_THRESHOLD*. This threshold is only provided for the incoming nodes to select the cluster they want to join (if they want to join or else form their own cluster).

Hence, if a cluster has a size lower than the minimum allowed, each member in the cluster is allowed to look for “greener pastures” independently and join one of them if it finds a better cluster(s). *i.e.* each node periodically broadcasts a *CLUS_FIND* similar to the one sent by a new node in the cluster formation process. A similar exchange of messages occur and the node joins the new cluster if it is better than its current cluster. In this case apart from the cluster stability metric (which is the primary filtering criteria), the node will also look only for clusters with a size greater than or equal to the size of its current cluster. In case the node finds only clusters around it which have the same size as its own cluster, it will compare the cluster IDs and will decide to join the new cluster (whose ID is the smallest of the new cluster IDs), only if the new cluster ID is smaller than its current cluster ID. If this case does not happen, then the selection (after using both the stability and size metrics) of the new cluster will happen based on the cluster formation rules as discussed in Section III-A.

IV. ROUTING FRAMEWORK

We now present a routing mechanism to work with our cluster based approach. Cluster based approaches, tend to reach a compromise between Pro-Active and Reactive routing protocols as discussed in the introduction. Almost all the clustering schemes that we looked at maintained pro-active routes within the cluster and reactively discovered routes for inter-cluster traffic [1]. We also adopt a similar approach but with a slight addition. Each node in the network is required to keep a routing table (similar to the one in DSDV [11] or Ticket-Based scheme [10]) which contains routing information to *relative nodes* (those belonging to the same cluster). The tables for *friend nodes* (border nodes) will also contain information about routing in adjacent clusters (which they border). This will not need any more transmission overhead since the border nodes will in any case be able to hear the routing table broadcasts of their neighbours in adjacent clusters.

This table will contain information about the paths to each destination (relatives for interior nodes and relatives plus each node in the neighbouring cluster for border nodes). Within a cluster, each node keeps the following data for each border node:-

- 1) The clusters the border node is adjacent to.
- 2) The clusters which the adjacent clusters border.

So, each node in a cluster keeps the IDs of the adjacent clusters and the IDs of the clusters which the adjacent clusters border. Similarly, a border node, keeps a full view of the adjacent cluster too, *i.e.* it keeps information about the the IDs of the friend nodes in the adjacent clusters and the clusters which they link to. This will help in routing the inter-cluster packets.

Thus for each destination, the routing table will contain, all possible options for next hop along with the QoS parameters (bandwidth and delay) for the paths. The routing tables will be updated as in the DSDV [11] protocol with the sequence numbers to prevent loops. These routing tables will be exactly the same as the ones used in the ticket-based routing scheme [10]. Apart from this table, each node will also contain an “acquaintance list” for inter-cluster routing. We explain this in the Inter-Cluster routing section.

Given that the routing tables are maintained and updated whenever required (apart from periodic updates), we need to look at the actual route discovery mechanism. A node on receiving a connection request from an application (source node) or another node (intermediate node), will first search for the node among the list of its relatives (which it pro-actively maintains in its routing table). If it finds it, then an Intra-Cluster routing will be performed, else an Inter-Cluster route discovery will be initiated.

A. Intra-Cluster Routing

If the destination node is a relative node, then an intra-cluster route discovery is initiated. Note that here we need to find a path which satisfies the required Quality of Service parameters. The intra-cluster path discovery algorithm is exactly the same as the one used in the Ticket-based scheme [10]. The advantages of this strategy is that it discovers routes satisfying the criteria and gives us complete control over the overhead involved in exchanging information and reserving the required resources. Since the number of nodes within a cluster are limited by an upper bound, the amount of overhead can be effectively controlled.

Alternate and backup paths provide the stability in cases of link failure or when a particular node cannot handle the required QoS assurances. Route maintenance mechanism is very nicely explained in the ticket-based scheme and can be directly applied here. One can also have multi-path routing so as to prevent over-loading of some paths in the network and also to provide graceful degradation of service in case of link failure.

Thus, route discovery and maintenance can be performed via any protocol as long as it supports the formation of links which guarantee the required QoS levels.

B. Inter-Cluster Routing

For inter-cluster routing, we propose a slight modification to the existing reactive protocols by adding another class of devices called *acquaintances*. We explain the idea behind acquaintances next.

Acquaintances: An acquaintance can be thought of in quite similar terms to acquaintances in real life. They are in some sense attached to us “temporarily”. Similarly, in the context of MANets, two nodes are acquaintances if they are not relatives and had been in contact in the not so distant past. In other words, Node **A** is an acquaintance of node **B** and vice versa, if they had exchanged some packets recently in time. The amount of time characterizing the recency in their communication can be given by a fixed number of seconds and denoted by $T_{acquaintance}$. Acquaintances are maintained and updated dynamically. Thus if node **A** and **B** are acquaintances and one of them changes its cluster binding, it sends out a *CHANGE_CLUS* message to the other one and the other one responds back. This communication resets the $T_{acquaintance}$ timer again (at both the nodes) and the two nodes will remain acquaintances for another interval of time. So, two nodes renew their contacts with each other by sending messages about their cluster bindings to each other and hence extend their “acquaintanceship”. In case a node does not hear from an acquaintance for a period greater than $T_{acquaintance}$, it will remove it from its list of acquaintances and the two nodes will now become unknown to each other and hence will no longer track each other’s movements.

A point to be noted here is that a node sends a *CHANGE_CLUS* message to all its acquaintances only when it changes its cluster. No mutual exchange of messages takes place if both of them remain in their respective clusters. This is done to reduce the messaging overhead and allow message exchange to be limited only for a particular amount of time and not go on infinitely.³ To limit the exchange overhead, we impose the condition that a *CHANGE_CLUS* message is sent to an acquaintance only if it is in an adjacent cluster or in a cluster adjacent to an adjacent cluster (2-cluster hop

³This protocol for renewal of acquaintance contact information (cluster binding) is not unique and can be tried out with different combinations and rules for message exchanges. The best strategy can be selected via simulation for different types of network environments.

cluster. Since each node keeps this information, this can be implemented easily). This mechanism of acquaintance update will lead to more message transfer in a more dynamic environment and lesser overhead for the static case. This is reasonable since one would like to keep some idea about the network topology if it is changing too fast so as to be able to take routing decisions quickly without resorting to flooding the network.

Now, coming back to the routing problem. If an node does not find the destination in the list of its relatives, it looks for it in its acquaintance list. If found, it immediately knows the cluster binding of the destination. This simplifies the routing process a lot since the only task now remaining is to route this packet to the destination cluster which is known. If the destination cluster is within 2-cluster hops (this information is available at each node), the route can be discovered by sending the packet to the corresponding friend in its own family which will then route it (through the adjacent cluster which borders it, if need be) to the destination cluster and hence the route discovery can be performed.

In the worst case, the destination node’s cluster binding is not available at the node, and the route discovery should be performed through a reactive protocol (e.g. [12], [13]) in which the border nodes will play a crucial role in determining whether the route request goes through a particular cluster or not.⁴ Since each border node knows about the clusters it borders and the clusters which the bordering clusters border, it can take quick decisions about the path of the packet and hence enable finding the path to the destination by avoiding complete flooding.

Route Maintenance can be performed by the actual protocol which is used over our framework. We are avoiding the details of the message transfers between nodes since any reactive protocol with the modifications we have proposed should in principle be able to work.

V. CLUSTERING SIMULATIONS

Simulations were performed to measure the performance of the clustering strategy. We took an area of 100×100 units, 75 nodes and a stability threshold of 0.3.

Random number of nodes were switched ON at random times in the initial part of the simulations. Clustering was performed dynamically and movement was imparted to a random number (with mean half of

⁴Acquaintances can come in handy in such a situation also, since it is likely that an intermediate node may have the desired destination node in its list of acquaintances and hence the path can be found out faster.

the nodes ON at that time) of nodes with exponential velocities (mean=10 units/iteration) and destinations at each instant of time (each iteration). Thus, almost all the nodes were mobile at each instant of time and hence, a highly mobile environment was simulated. We used $T_{transient} = 2$ iterations. Each node had a transmission range of 15 units. Each reading was averaged over 10 random graphs each run for 100 iterations.

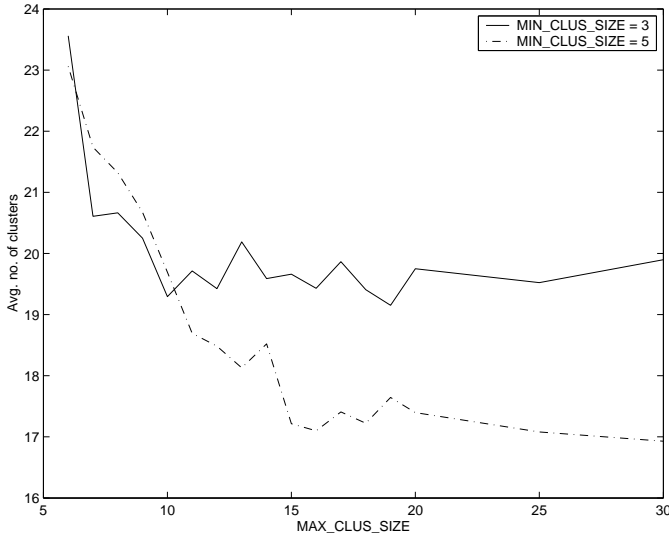


Fig. 1. Number of Clusters vs. MAX_CLUS_SIZE

In Fig. 1 we can see that the number of clusters can be efficiently controlled by suitably choosing the MAX_CLUS_SIZE and MIN_CLUS_SIZE parameters. One can see that (as expected) the number of clusters decreases as larger clusters are allowed. It is important to note that since we chose random graphs, in many cases there were isolated nodes in the network which did not have any neighbours with which they could form larger clusters. Hence there were unavoidable singleton clusters in some cases. Thus both the parameters can be chosen and varied to get the desired clustering behaviour.

In Fig. 2 one can see the variation of number of nodes with transmission range. Here we chose $MAX_CLUS_SIZE = 12$ and $MIN_CLUS_SIZE = 3$. The graphs are as expected intuitively – larger range means more clusters can be formed. We performed this simulation for two different stability metrics and one can see that a higher stability metric leads to the formation of more number of clusters since many clusters may not satisfy the stability criteria.

Fig. 3 shows the variation of number of clusters breaking in the system with the stability threshold. Again, we see a downward trend in the cluster breakings as the stability threshold is increased. Note that a higher

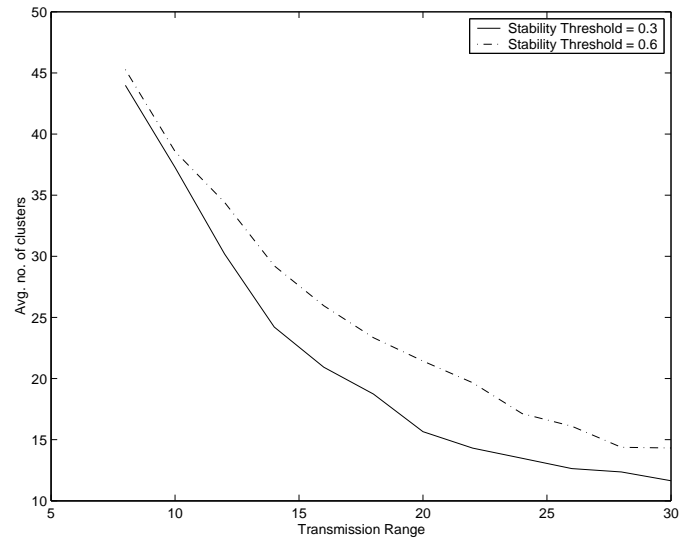


Fig. 2. Number of Clusters vs. Transmission Range

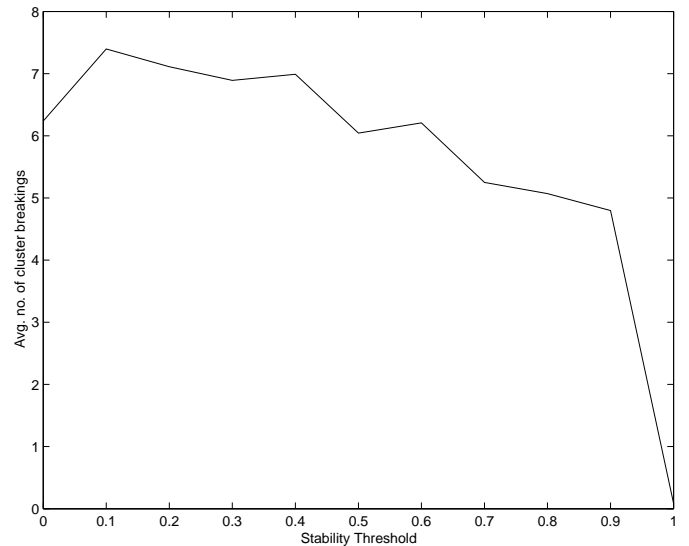


Fig. 3. Number of cluster breakings vs. Stability Threshold

stability metric leads to a larger number of clusters which may not be desirable. Thus, we can choose the suitable stability threshold and optimise the performance of the clustering technique.

VI. CONCLUSION

In this work, we have proposed a new, completely distributed, adaptive clustering algorithm for Mobile Ad Hoc Networks. The algorithm is a distributed version of the Basic Leader-Follower Algorithm and we term it as the d-BLF algorithm. The size of the clusters is upper bounded and the number of clusters and their dynamic behaviour can be controlled by using the $STABILITY_THRESHOLD$, MAX_CLUS_SIZE and MIN_CLUS_SIZE . We

also propose a modification to the existing cluster based routing protocols. Our *acquaintance* based approach helps reduce the latencies in inter-cluster routing algorithms and yet tries to keep the overhead to low levels.

The proposed approach tries to maintain a stable cluster topology by considering a “node age” based stability metric and needs little resources in terms of computational effort.

ACKNOWLEDGEMENT

I sincerely thank Prof. R. K. Shyamasundar for his guidance and unending encouragement. I would also like to thank Prof. V. S. Borkar for his tips and suggestions which really helped me.

REFERENCES

- [1] A. B. McDonald and T. F. Znati, *A Mobility-Based Framework for Adaptive Clustering in Wireless Ad Hoc Networks*, IEEE Journal on Selected Areas in Communications, Vol. 17, No. 8, pp. 1466-1487, Aug. 1999.
- [2] C. R. Lin and M. Gerla, *Adaptive Clustering for Mobile Wireless Networks*, IEEE Journal on Selected Areas in Communication, Vol. 15, No. 7, pp. 1265-1275, Sept. 1997.
- [3] R. O. Duda, P. E. Hart and D. G. Stork, *Pattern Classification*, 2nd ed., New York: John Wiley & Sons, 2001.
- [4] A. Jain, B. Rajan, and R. K. Shyamasundar, *Failure Resilient Communication and Membership Management for Clusters of Grid Environments*, manuscript submitted.
- [5] S. Sivavakeesar and G. Pavlou, *A Prediction-Based Clustering Algorithm to Achieve Quality of Service in Multihop Ad Hoc Networks*, London Communication Symposium, 2002, <http://www.ee.ucl.ac.uk/lcs/papers2002/>.
- [6] S. Basagni, *Distributed Clustering for Ad Hoc Networks*, Proceedings of the 1999 International Symposium on Parallel Architectures, Algorithms and Networks (I-SPAN'99), IEEE Computer Society, pp.310-315, Australia, Jun 23-25,1999.
- [7] B. Das, R. Sivakumar and V. Bhargavan, *Routing in Ad Hoc Networks Using a Spine*, Sixth International Conference on Computer Communications and Networks (ICCCN '97), Las Vegas, NV, Sept 22 - 25, 1997.
- [8] S. Basagni, I. Chlamtac, A. Faragó, *A Generalized Clustering Algorithm for Peer-to-Peer Networks*, Workshop on Algorithmic Aspects of Communication, satellite workshop of ICALP'97, invited paper, Bologna, Italy, Jul 11-12, 1997.
- [9] N. H. Vaidya, P. Krishna, M. Chatterjee and D. K. Pradhan, *A Cluster-based Approach for Routing in Dynamic Networks*, ACM SIGCOMM Computer Communication Review, Vol. 27, No. 2, Apr. 1997.
- [10] S. Chen and K. Nahrstedt, *Distributed Quality of Service Routing in Ad-Hoc Networks*, IEEE Journal on Selected Areas in Communication, Vol. 17, No. 8, Aug. 1999.
- [11] C. R. Perkins and P. Bhagwat, *Highly Dynamic Destination Sequenced Distance Vector Routing (DSDV) for mobile computers*, Proceedings of ACM SIGCOMM, pp. 234-244, Oct. 1994.
- [12] C. E. Perkins and E. M. Royer, *Ad-hoc On-demand Distance Vector Routing*, Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications 1999, pp. 90-100, 25-26 Feb. 1999.
- [13] D. B. Johnson, D. A. Maltz, *Dynamic Source Routing in Ad Hoc Wireless Networks*, Mobile Computing, edited by Tomasz Imielinski and Hank Korth, Chapter 5, pp. 153-181, Kluwer Academic Publishers, 1996.
- [14] D. Wu and R. Negi, *Effective Capacity: A Wireless Link Model for Support of Quality of Service*, to appear in IEEE Transactions on Wireless Communications, Vol. 2, No. 4, Jul. 2003.
- [15] C. R. Lin and M. Gerla, *MACAPR: An Asynchronous Multimedia Multihop Wireless Network*, Proceedings of IEEE Infocom '97, 1997.
- [16] http://www.zytrax.com/tech/wireless/802_mac.htm.
- [17] P. Sinha, R. Sivakumar and V. Bhargavan, *CEDAR: A Core Extraction Distributed Ad Hoc Routing Algorithm*, IEEE Infocom 99, New York, NY, Mar. 1999.
- [18] R. Braden, D. Clark and S. Shenker, *Integrated Services in the Internet Architecture - an Overview*, IETF RFC1663, Jun. 1994.
- [19] S. Blake, *An Architecture for Differentiated Services*, IETF RFC2475, Dec. 1998.
- [20] K. Wu and J. Harms, *QoS Support in Mobile Ad-Hoc Networks*, Crossing Boundaries- the GSA Journal of University of Alberta, Vol. 1, No. 1, Nov. 2001, pp.92- 106, <http://www.cs.ualberta.ca/~wkui/research/QoSReview.ps>.
- [21] <http://www.mpls.com>.
- [22] C. H. Yeh, *Ad hoc MPLS for virtual-connection-oriented mobile ad hoc networks*, IEEE 55th Vehicular Technology Conference, 2002. VTC Spring 2002., pp. 1101-1105, Vol. 3, 6-9 May 2002.
- [23] M. Grossglauser and D. Tse, *Mobility Increases the Capacity of Ad-Hoc Wireless Networks*, IEEE Transactions on Information Theory, Vol. 48(6), Jun. 2002.