A Novel Approach for Routing in Ad-Hoc Wireless Local Area Networks*

P.Krishna M. Chatterjee N. H. Vaidya D. K. Pradhan
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112
Contact E-mail: pkrishna@cs.tamu.edu

Technical Report #95-01

Abstract

An *ad-hoc* wireless network is a cooperative set of mobile hosts which can communicate with each other over the wireless links (direct or indirect) without any static network interaction. This report presents a "cluster-based" approach to routing in *ad-hoc* networks. A *k-cluster* is defined by a subset of nodes which are 'reachable' to each other by a path of length at most k for some fixed k. This report deals with clusters of $k = 1$, i.e., *1-clusters* (or simply clusters). Our approach is motivated by our study of existence of clusters (size greater than 2) in random graphs. The basic idea behind the protocol is to divide the graph into number of overlapping clusters. A change in the network topology corresponds to a change in the cluster membership. Performance of the proposed routing protocol (reconvergence time, and update overhead) will hence be determined by the average cluster size in the network graph. The effectiveness of this approach lies in the fact that existing routing protocols can be directly applied to the network – replacing the nodes by clusters. When the average cluster size is less than 2, the proposed approach does not perform any worse than the existing routing protocols.

***Note:** This is a slightly revised version of the paper originally presented at USENIX '95 [4]. Research reported is supported in part by AFOSR under grant F49620-94-1-0276, and Texas Advanced Technology Program under grant 999903-029.

1 Introduction

Mobile wireless networks gives users communicating capability and information accessing capability regardless of the location of the user. With the availability of wireless interface cards, mobile hosts are no longer required to remain confined within the static network premises to get network access. In order to communicate with any particular host, it is first necessary to locate the host in the network. This is due to the fact that the hosts are mobile and could be anywhere. In addition to mobility, the host can also be in a disconnected mode (power-saving). This dynamic feature in mobile wireless networks leads to the problem of keeping track of the topology connectivity. This problem becomes noticeable when the the rate of change is high, and the network sizes are large.

An important issue in mobile wireless networks is the design and analysis of topology management schemes. *This report investigates the consequence of mobility and disconnections of mobile hosts on the routing overhead in a "mobile" network.* We define a *mobile* network as a cooperative set of mobile hosts which can communicate with each other over the wireless links (direct or indirect) without any static network interaction[1]. Example of such networks are *ad-hoc* networks [1, 3, 21], and packet radio networks [2, 16, 17]. The term *ad-hoc* network is in conformance with current usage within the IEEE 802.11 subcommittee [21]. Ad-hoc networking in the wireless world refers to the ability to create a peer oriented network between several clients all of which are wireless and all of which are "virtually LAN'd" together. Ad-hoc also implies that the wireless network can be created dynamically or in an "ad-hoc" fashion. Once this type of network has been created by two clients then other users may freely gain media access (provided the specific security and configuration parameters of the physical link are valid). The wireless LANs and their standards address only the MAC and PHY layers and thus a wireless network which features this function relies on the upper level protocol stacks (i.e., IPX, IP, netBIOS, etc.) to allow for either peer-to-peer or client-server operation from a session/application point of view. The focus of this report is to introduce a new routing methodology more suited for such *mobile* networks.

Example applications of such *mobile* networks range from conference rooms to battlefields. To communicate with each other, each mobile user needs to connect to a static network (wide area network, satellite network). However, there might be situations where connecting each mobile user to a static network may not be possible due to lack of facilities, or may be expensive. In such situations, it would be more preferable for the mobile users to set up communication links between themselves without any static network interaction [3].

In the current existing proposals for mobile wireless networks, routing information of each mobile host is maintained in some database (*HLR* and *VLR* in IS-41 [5, 6], *home agent* and *foreign agent* in mobile IP [8, 9]) which is located in the static network. However, there is no such database available for *ad-hoc* networks. Due to limited range of the wireless transreceivers, a mobile host can communicate with another host only within a limited geographical region around it. Thus, it may be necessary for a mobile host to require the aid of other mobile hosts in forwarding data packets to its destination. The routing information will thus be maintained at the mobile hosts to

[1] We assume that a mobile host has the capability to communicate directly with another mobile host. It is also assumed that the mobile hosts have the capability to forward (relay) packets.

assist in forwarding packets to other hosts. The problem in hand is the complexity of updating the routing information in such a dynamic (due to the mobility of the hosts, and limited power on the hosts, thus, host disconnections) network. Let us illustrate it with an example.

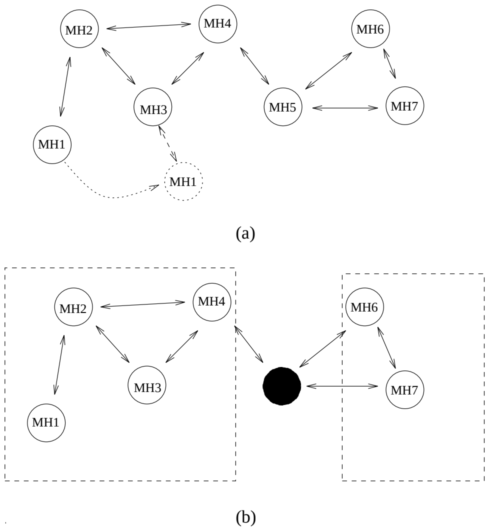

Figure 1: Ad-hoc Network – A Dynamic Network

Example 1.1: Figure 1(a) is an example of ad-hoc network. Routing information are maintained at each host. For example, routing table at $MH4$ will require packets destined to $MH1$ to be forwarded to $MH2$, which in turn will forward the packets to $MH1$. However, due to the movement of host $MH1$, the network topology changes. The communication link between $MH2$ and $MH1$ breaks, and, there is a new link between $MH3$ and $MH1$. Thus, the routing table at the hosts have to be updated to indicate this change in topology. For example, the routing table at $MH4$ has to be updated to indicate that the packets destined to $MH1$ have to be forwarded to $MH3$ and not $MH2$. The network topology also changes due to host disconnections. As illustrated in Figure 1(b), the network gets partitioned due to the disconnection of host $MH5$. Thus, the routing information at the hosts have to updated to indicate the change in topology. $\Box$

1.1 Previous Work

Numerous routing protocols have been proposed in the recent years. One of the most popular techniques for routing in communication networks is via distributed algorithms for finding shortest paths in weighted graphs [12, 13, 14, 18]. These distributed algorithms differ in the way the routing tables at each host are constructed, maintained and updated. The primary attributes for any routing protocol are :

- Simplicity : This is one of the most primary attributes for a routing protocol. Simple protocols are preferred for implementation in operational networks [1].
- Loop-free : At any moment, the paths implied from the routing tables of all hosts taken together should not have loops. Looping of data packets to be routed results in considerable

overhead.

- Convergence characteristics : The time required to converge to new routes after a topology change should not be high. Quick convergence is possible by requiring the nodes to frequently broadcast the updates in the routing tables.
- Storage overhead : The memory overhead incurred due to the storage of the routing information should be low.

The conventional routing protocols can be broadly classified as *distance vector* and *link state* protocols. The *distance vector* routing uses the classical distributed bellman-ford algorithm [11, 16, 18, 19]. Each host maintains for each destination a set of distances through each of its neighbors. In order to maintain up-to-date information, each host periodically broadcasts to each one of its neighbors, its current estimate of the shortest path to every other host in the network. For each destination, the host determines a neighbor to be the next hop for a message destined for the destination if the neighbor has the shortest path to the destination.

Link state routing requires each host to have knowledge of the entire network topology [20]. To maintain consistent information, each host monitors the cost of each communication link to each of its neighbors, and periodically broadcasts an update in this information to all other hosts in the network. Based on this information of the cost of each link in the network, each host computes the shortest path to each possible destination host. The processing overhead and the network bandwidth overhead of *link state* protocols are generally more than *distance vector* protocols.

The problems in using conventional routing protocols in an ad-hoc network have been discussed in great detail in [1, 3]. For completeness sake, we briefly list the problems in the following.

- The conventional routing protocols were not designed for networks where the topological connectivity is subject to frequent, unpredictable change as evident in ad-hoc networks. Most of them exhibit their least desirable behavior for highly dynamic interconnection topology.
- Existing protocols could place heavy computational burden on mobile computers, and the wireless networks, in terms of battery power and network bandwidth respectively.
- Convergence characteristic of these protocols is not good enough to suit the needs of ad-hoc networks.
- Wireless media has a limited and variable range, different from existing wired media.

The protocol described in [1] addresses some of the above stated problems by modifying the Bellman-Ford routing algorithm. They use sequence numbers to prevent routing table loops, and, settling-time data for damping out fluctuations in route table updates. The convergence on the average was rapid, however, the worst case convergence was non-optimal. Moreover, their protocol required frequent broadcasts of the routing table by the mobile hosts. The overhead of the frequent broadcasts goes up as the population of mobile hosts increases.

Johnson proposed a new routing method for ad-hoc networks based on separate route discovery and route maintenance protocols [3]. The concept of Address Resolution Protocol (ARP) was extended to discover routes. However, if proper measures are not taken, the network performance could degrade due to the propagation of redundant route discovery requests. Route maintenance was achieved by using hop-by-hop acknowledgement. However, due to such relaxed maintenance measures, the hosts could be using poor (long) routes when better (shorter) routes are available. This will degrade the network performance.

A distributed routing protocol for mobile packet radio networks was proposed by Corson et al. [2]. Similar to [10], routing optimality was of secondary importance. Rather, their goal was to maintain connectivity between the hosts in a fast changing topology. Moreover, instead of maintaining distances from all sources to a destination, the protocol guarantees route maintenance only for those sources that actually desire routes. This property helps in reducing the topology update overhead. The protocol is a compromise between two extremes : *flooding* [11] (suited for high rate of topology change), and *shortest-path* algorithms [15] (suited for low rate of topology change).

1.2 Proposed Approach

This report presents a new methodology for routing and topology information maintenance in mobile wireless network. Our approach is motivated by our study of existence of clusters (size greater than 2) in random graphs. The basic idea behind the protocol is to divide the graph into number of overlapping clusters. A change in the network topology corresponds to a change in the cluster membership. Performance of the proposed routing protocol (reconvergence time, and topology update overhead) will then be determined by the average cluster size. The effectiveness of this approach lies in the fact that existing routing protocols can be directly applied to the network – replacing the nodes by clusters. When the average cluster size is less than 2, the proposed approach does not perform any worse than the existing routing protocols. For future reference, let us formally define *clusters*.

Definition 1.1: A *k-cluster* is defined by a subset of nodes which are 'reachable' to each other by a path of length at most k for some fixed k. A *k-cluster* with $k = 1$ is a clique. This report deals with clusters of $k = 1$, i.e., *1-clusters*. (Hereafter, we refer *1-cluster* simply as cluster.) However, we can also generalize our protocols with values of k greater than one (subject of our ongoing research). Each cluster is identified by its members.□

Definition 1.2: The *size*, $S(C)$ of a cluster C is the number of nodes in C.□

Definition 1.3: A graph is *cluster-connected* if it satisfies the following two conditions :
1) The union of the clusters cover the whole graph.
2) There is a path from each node to every other node through the edges of the clusters in the graph. □

The main problem here is to develop protocols for cluster maintenance. The protocols should be simple and distributed, and, should incur low overhead. To this effect, we develop simple distributed protocols to detect, and, build irredundant clusters in a graph. We maintain a *minimal* number of clusters based on the connectivity criteria (Definition 1.3). Experiments are performed to determine the average cluster size in random graphs. Section 2 presents the problem of routing in mobile wireless networks. We present the protocols to divide the nodes into clusters in section 3. Section 4 presents the proposed routing protocol based on clusters. Section 5 presents experimental results and its discussions. Conclusions are presented in section 6.

2 Preliminaries

The problem addressed in this report can be defined as follows:

Given: A wireless mobile network configuration.
Problem: Find a 'good' loop-free routing between each mobile host in the network, where the topological connectivity is subject to frequent unpredictable change.

The problem requires a *loop-free distributed routing* protocol which determines an acyclic route between each host whenever a change in the topology is detected. The protocol is intended for use in networks where the rate of topological change is not so fast as to make "flooding"[2] the only viable routing method, but not so slow as to make any static topology routing applicable. A loop-free[3] routing is a routing where the path from one host to another does not traverse through the same node twice. A loop-free routing is desirable to minimize the consumption of resources during routing.

A 'good' route from one host to another is not necessarily the shortest path. In an environment of frequent topological change, a 'good' route connects a host to another host and the route length is comparable to the shortest one. Each host maintains a data-structure describing the network topology and some routing information pertaining to a common routing protocol. The routing protocol adapts asynchronously in a distributed fashion to arbitrary changes in topology in the absence of global topological knowledge. Let an undirected graph, $G = (V,E)$ represent the network of mobile hosts. Each node u, in the graph denotes a mobile host H_u. Due to the limited range of wireless transreceivers, a mobile host can communicate with another host only within a limited geographical region around it. This region is called the host coverage area – d being the radius. The geographical area covered by a host coverage area is a function of the medium used for wireless communication. A host H_u is in the vicinity of H_v if the distance between nodes u and v is less than or equal to d. An edge (u,v) connects node u and node v if the corresponding hosts are in the 'vicinity' and have a *direct connection* between each other. A host may sometime be isolated where it has no other mobile hosts in its vicinity. Such a host will be represented in the graph by a disconnected node. A host H_{v1} is connected to another host H_{v2} if there exists at least one path from node $v1$ to $v2$. The path length is given by the number of edges on the path. Routing from one node to another node should ideally use the path with the shortest length. The wireless mobile routing problem requires a distributed graph algorithm to determine a loop-free route from each node to every other node.

[2] Flooding can be described as an algorithm whereby a node broadcasts a message packet to its neighbors, who in turn broadcast the packet to all their neighbors, except the neighbor from which it was received. This process goes on till the message packet reaches the intended destination. This happens provided the destination is connected to the node which originated the flood [2].

[3] Loop-free routing requires prevention of loops in the routing tables. Here, existence of temporary loops are not of concern.

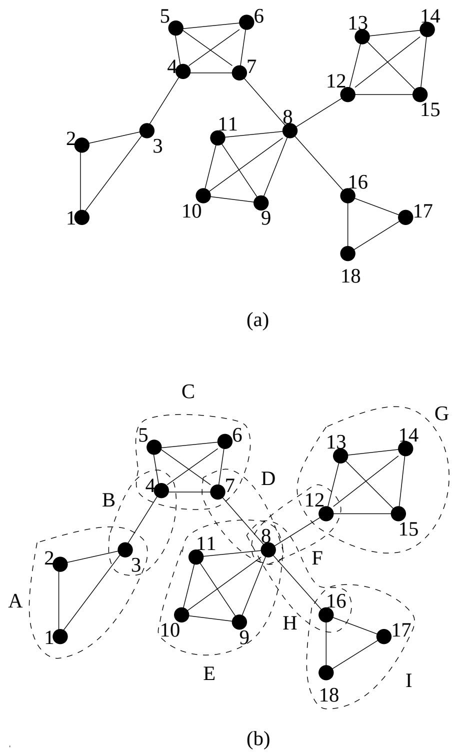

Figure 2: An Example

Example 2.1: The graph (in Figure 2(a)) is formed based on the geographical locations of the 18 mobile hosts. In this example, the graph is connected as each node is *reachable* to every other node. It can be observed that based on the positions, some nodes form clusters. The graph (in Figure 2(a)) can be divided into nine clusters (in Figure 2(b)). The clusters and their respective members are as follows : A (1,2,3), B (3,4), C (4,5,6,7), D (7,8), E (8,9,10,11), F (8,12), G (12,13,14,15), H (8,16) and I (16,17,18). Routing can be done from one node to another by only using clusters. Routing from node 1 to node 16 is done through the clusters A, B, C, D and F. The graph in Figure 2(b) is *cluster-connected* because, (i) the union of the clusters covers the whole graph, and (ii) there is a path from each node to every other node using the clusters. □

A topological change in the mobile host network corresponds to a change in the graph structure $G(V,E)$ to $G'(V',E')$. A change in the graph structure can be of the nature of a node (host) or an edge (connection between two hosts). We outline four types of events in the mobile network that can incur changes in the graph (in the following H_A and H_B are mobile hosts) :

A) <u>H_A switching ON:</u> A host H_A switching ON will include itself in the graph and make connection with all the hosts in its 'vicinity'. Hence, $V' = V \cup \{A\}$ and $E' = E \cup \{(u, A)$, s.t. H_u is connected to $H_A\}$.

B) <u>H_A switching OFF:</u> A host H_A switching OFF will exclude itself from the graph and delete all its edges. Hence, $V' = V - \{A\}$ and $E' = E - \{(u, A)$, s.t. $(u, A) \in E\}$.

C) H_A gets connected to H_B: Here, an edge between A and B will be added to the graph. Hence, $V' = V$ and $E' = E \cup \{(A, B)\}$

D) H_A gets disconnected from H_B: Here, the edge between A and B will be removed from the graph. Hence, $V' = V$ and $E' = E - \{(A, B)\}$

A routing protocol will change its routing information based on the afore-mentioned four types of changes in the graph. We add some definitions and properties which will assist in describing our proposed routing protocol.

Definition 2.1: Cluster set S_n of a node n is defined as the set of all clusters in which n is a member. □

Definition 2.2: If a cluster $C \in S_n$ can be removed and still all nodes $i \in C$, have paths to every other node j, where $j \neq i$ and $j \in C$, using other clusters in S_n, C is a *redundant* cluster. □

A cluster determined to be redundant for one node, may not be redundant for other nodes. A graph will have *irredundant* clusters if and only if each node n do not have *redundant* clusters in their cluster set S_n.

Definition 2.3: A node A is a *boundary* node if it is a member of more than one cluster. In Figure 2(b), node 3 is a boundary node as it belongs to two clusters, (1,2,3) and (3,4). However, node 1 is not a boundary node as it only belongs to (1,2,3).□

Property 2.1: Addition of each new node to the graph adds at least one new irredundant cluster[4]. □

Property 2.2: Given a graph with irredundant clusters, with the addition of new members, only the members of new clusters can identify the redundant clusters in the graph. □

3 Cluster Formation

Our proposed routing protocol is based on the formation of clusters. Hence, efficient cluster formation will be the crux of a routing protocol of this nature. Clusters should be formed in such a way that the resulting graph is *cluster-connected* (See Definition 1.3). Routing from one node to another will consist of routing inside a cluster and routing from cluster to cluster. A change in the mobile network may or may not result in a change in the cluster compositions. Here, we have assumed clusters with $k = 1$ (See Definition 1.1). As mentioned in Section 2, we have identified four different possible types of changes in the mobile network graph in the occurance of a single event.

We present an algorithm to divide the graph into clusters. The main contribution of this report is to present the effectiveness of the cluster-based approach for routing in mobile networks. We will now present the protocols for cluster updates with each type of topological change.

[4] However, one or more existing clusters can become redundant due to the addition of the new clusters.

3.1 Host H_A switches ON

The new graph structure $G'(V',E')$ is formed with the added node. The new node A will result in at least one new cluster so that with the cluster, node A can route to the rest of the graph. However, if A connects two disjoint subgraphs, it may result in more than one added cluster. These new clusters are denoted by *essential* clusters and can be detected by A itself. The addition of new clusters may result in zero or one or more clusters being redundant. The two tasks performed during the topological change are (i) addition of new clusters, and (ii) removal of redundant clusters. The goal is to have *minimal* number of clusters such that the graph remains *cluster-connected*. The protocol initiated by new node A is described as follows.

```
Procedure Switch ON(A);
Begin;
A sends messages to its neighbors about its new arrival;
For each of A's neighbors n do
    send list of its neighbors to A;
A gets information from all neighbors and creates all possible new clusters list;
A executes Find Essential(A,list);
A broadcasts Essential Clusters;
For each of A's neighbors n s.t. n∈Essential Clusters do
    S_n = S_n ∪ Essential Clusters ;
    n executes Find Redundant (n, S_n) ;
    n broadcasts Redundant Clusters;
Change in cluster structures are propagated to rest of the graph;
End;
```

The new node A broadcasts a message to its neighbors indicating its new arrival. Upon receipt of the new arrival message, the neighbors send a list of their neighbors to A. Based on the information received from its neighbors, A determines all possible clusters and stores them in *list*. The new node A then executes **Find Essential** function.

```
Function Find Essential(A, list);
Begin;
Sort the clusters in list in a non-descending order of their sizes;
For each cluster C∈list do
    Mark(C) :=essential ;
For each cluster (C∈list) ∧ (Mark(C) =essential) do
    For each node (n∈C) ∧ (n ≠ A) do
        For each cluster (C'∈list) ∧ (C' ≠ C) ∧ (Mark(C') =essential) do
            if (n∈C')
                Mark(C) :=non-essential;
                break;
    if (Mark(C) =essential)
        Essential Clusters := Essential Clusters ∪ C;
return;
End;
```

The **Find Essential** function sorts the clusters in *list* in a non-descending order of their sizes. Initially all the clusters are marked *essential.* It then goes through each *essential* cluster C to find if a node (other than the new node A) in C is a member of any other *essential* clusters. If so, it marks the cluster C as *non-essential.* This will ensure that a node (other than the new node A) is a member of no more than one *essential* cluster. Moreover, since the clusters are sorted in a non-descending order of their sizes, the **Find Essential** function returns the largest clusters possible. The *essential* clusters determined by **Find Essential** function are stored in *Essential Clusters.*

```
Function Find Redundant(n, S_n);
Begin;
For each cluster C∈S_n do
    Mark(C) :=redundant ;
    For each node (i∈C) ∧ (i ≠ n) do
        match := FALSE ;
        For each cluster (C'∈S_n)∧(C' ≠ C)do
            if (n∈C')
                match := TRUE ;
        if match = FALSE
            Mark(C) :=non-redundant;
    if (Mark(C) =redundant)
        S_n := S_n − C ;
        Redundant Clusters := Redundant Clusters ∪ C ;
return ;
End;
```

The new node A then broadcasts the *essential* cluster information its neighbors. Only the neighbors who are members of the *essential* clusters are involved in searching for redundant clusters. A neighbor will be a member of no more than one *essential* cluster. The neighbor then adds the *essential* cluster to its cluster set. Addition of the *essential* cluster might make one or more existing clusters in its cluster set *redundant.* The neighbor then executes **Find Redundant** function. This function determines *redundant* clusters based on *Definition 2.2.* The *redundant* clusters determined by **Find Redundant** function are stored in *Redundant Clusters.* The neighbor broadcasts the *redundant* cluster information to its cluster mates. The neighbor and its cluster mates then remove the *redundant* clusters.

Example 3.1: For an easier understanding, Figure 3 gives an example involving a graph with 4 nodes. Figure 3(a) has 4 nodes and two clusters, namely, (1,2,3) and (2,3,4). When node 5 is switched ON, it sends messages to nodes 1, 3, and 4 (Figure 3(b)). On receiving information back from the nodes 1, 3 and 4, node 5 forms clusters (1,3,5) and (3,4,5) as seen in Figure 3(c). It chooses (3,4,5) as the *essential cluster* and broadcast it to nodes 3 and 4. In the redundant removal phase, node 3 detects the cluster (2,3,4) to be redundant. The final clusters are (1,2,3) and (3,4,5) as in Figure 3(d).□

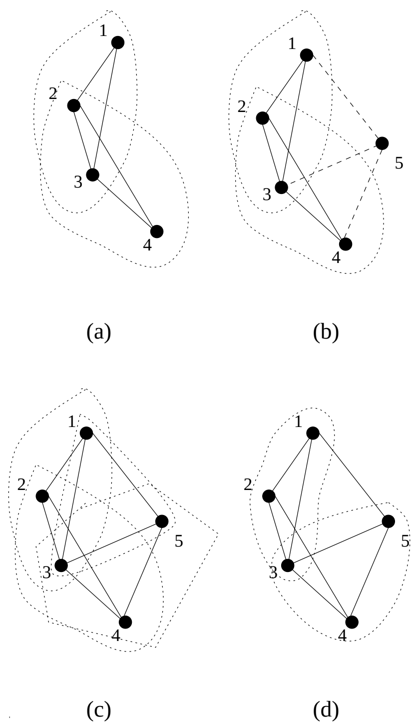

Figure 3: An Example of a Node Addition

3.2 Host H_A switches OFF

When host H_A turns OFF, its disappearance will only be detected by its neighbors. Hence, its neighbors will initiate a protocol to adapt to this topological change. We first illustrate the protocol with an example.

Example 3.2: Figure 4 shows the cluster formations when a node is turned OFF in a graph. Figure 4(a) has six nodes with three clusters, namely, (1,2,3), (2,3,4) and (4,5,6). When node 6 is turned OFF, the cluster (4,5,6) shrinks to (4,5) (Figure 4(b)). In the next step, nodes 4 and 5 try to expand their cluster and creates a cluster (3,4,5) (Figure 4(c)). In the redundant removal phase, node 3 detects the cluster (2,3,4) to be redundant. The final clusters are (1,2,3) and (3,4,5) as in Figure 4(d). □

There could be more than one node detecting the removal of a node. The number of nodes initiating the **Switch OFF** procedure should be the number of clusters A was a member. A node i detecting the removal of node A will initiate the procedure if and only if no other member of the cluster in which node i and A are members has already initiated the procedure.

Let B be one of the nodes detecting host H_A turning OFF. The procedure initiated by node B is described as follows.

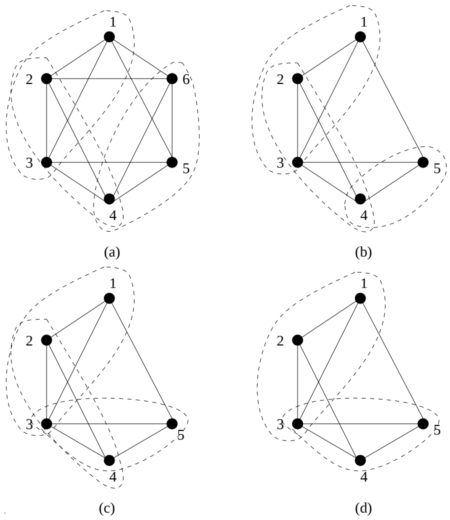

Figure 4: An Example of a Node Removal

Procedure Switch OFF(A,B);

Begin;
B requests the list of neighbors from the cluster mates of the shrinked cluster;
For each cluster mate n do
    send list of its neighbors to B;
B gets this information and tries to expand the shrunk cluster ;
B broadcasts the new cluster ;
For each cluster mate n do
    $S_n = S_n \cup$ new cluster;
    n executes **Find Redundant** (n, S_n) ;
    n broadcasts *Redundant Clusters*;
Change in cluster structures are propagated to rest of the graph;
End;

We outline two possible cases:

1. Node A was not a *boundary* node: A will be only contained in a single cluster. Hence the cluster shrinks after the removal of node A. The other members of the cluster try to expand the cluster size by including nodes from a different cluster. This might result in redundant clusters. The redundant clusters are detected and removed.

2. Node A was a boundary node: Here, the removal of node A will cause more than one cluster to shrink in size. This will give an opportunity to the shrinked clusters to combine and in that process create redundant clusters. All nodes n that were neighbors to A will look for clusters which can use itself, neighbors of A, and nodes from a different cluster. With the formation of new clusters, the redundant clusters are detected and removed.

The new cluster structures are then sent out to the rest of the graph.

3.3 Host H_A gets connected to Host H_B

The new connection between H_A and H_B is detected simultaneously by both the nodes. Both of them now try to create a cluster involving A, B and nodes from other clusters. Formation of these clusters will be consistent with A as well as B as the cluster includes both A and B. Once the cluster is formed, the other cluster-mates of A and B look for redundant clusters if any.

It should be noted here that there is a subtle difference in the cluster property observed between the case when either node A or B is a new node and the case when neither node A nor B is a new node. A new node can be defined as a node with no neighbors. In the case when at least one of the nodes is a new node, the following property is observed.

Property 3.1: The *essential* clusters determined by the new node are irredundant. □

Proof :Let the new node to be added be *new*. The *essential* clusters are determined by *new* using the **Find Essential** function. Steps 5-9 of the function ensure that a node n, $n \neq new$, is not a member of more than one *essential* cluster. Let S_n be the set of clusters in which n is a member. There will be only one *essential* cluster in S_n. Node n will not find the *essential* cluster to be redundant, because, the *essential* cluster contains the node *new* which is not a member of any other clusters in the cluster set S_n. Similarly, other cluster mates n' of *new* will determine independently that the *essential* cluster that n' is a member of, is irredundant. □

However, when neither node A nor B is a new node, the *essential* cluster determined by the nodes could be *redundant*. Let us illustrate it with an example.

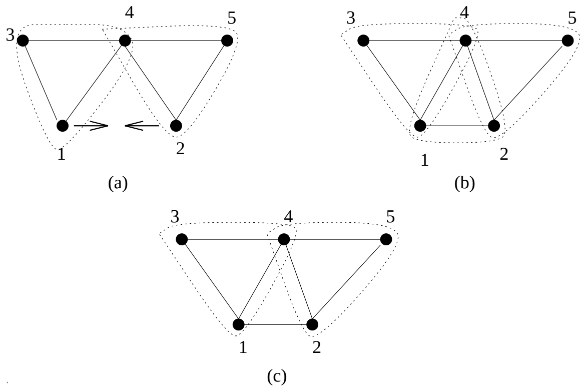

Figure 5: An Example of a New Connection

Example 3.3: Figure 5 shows the cluster formation when there is a new link between nodes 1 and 2. Figure 5(a) has five nodes with two clusters, namely, (1,3,4) and (2,4,5). When nodes 1 and 2 move towards each other, there is a new link created between nodes 1 and 2. Nodes 1 and 2 determine a new cluster namely (1,2,4) as the *essential* cluster (Figure 5(b)). The *essential* cluster information is sent to node 4. In the redundant cluster removal phase, node 4 detects the cluster (1,2,4) to be *redundant*. Thus, even though, cluster (1,2,4) was an *essential* cluster, it was determined to be *redundant* by node 4. The final clusters are (1,3,4) and (2,4,5) as in Figure 5(c).

3.4 Host H_A disconnects Host H_B

Here, we identify two cases as follows.

1. A was not in any cluster with B: The topological change will result in no change in any of the clusters.
2. A and B shared common clusters: Here, the topological change will result in the shrinking of the involved clusters. The hosts H_A and H_B initiate the **Switch OFF** protocol.

At the event of the change of a cluster structure, the neighbors will propagate the change to the rest of the graph.

4 Routing Protocol

Through an example, we will show in this section how to extend existing routing protocols to support cluster-based approach. We will extend a standard distance vector routing protocol [11, 16, 18, 19] to support clusters.

We will first discuss the necessary data structures to be maintained at each node for the routing protocol. We will then present an overview of the extensions to the standard protocol.

4.1 Data Structures

The following tables are maintained at each node :

- *ClusterTable* : This table provides the mapping between the nodes and their clusters. It might so happen that a node is a member of more than one cluster. Thus, for each node, the identifiers of all the clusters in which the node is currently a member, is maintained.
- *RouteTable* : For each destination cluster, the node maintains the identifier[5] of the next hop node, say n, and the number of hops it will take to reach a node in the destination cluster through n. This is the table which is referred to while routing a packet. This table maintains the shortest available path to every destination cluster.
- *AllRouteTable* : For each destination, this table maintains route information of all possible paths from the node. This table is used to determine the shortest available path to each destination node, which is maintained in *RouteTable*.

The *RouteTable* and *ClusterTable* for network in Figure 2 are shown in Tables 1 and 2. The *AllRouteTable*, for Figure 2, happens to be same as its *RouteTable* (Table 1), because, there is just one possible path between any two clusters.

[5] In *ad-hoc* networks, MAC address can be used to transmit packets directly to that node [3].

$ClusterId$	$NextHop$	$Hops$
A	4	2
B	4	1
C	-	0
D	7	1
E	7	2
F	7	2
G	7	3
H	7	2
I	7	3

Table 1: *RouteTable* at node 6

$ClusterId$	$Nodes$
A	1,2,3
B	3,4
C	4,5,6,7
D	7,8
E	8,9,10,11
F	8,12
G	12,13,14,15
H	8,16
I	16,17,18

Table 2: *ClusterTable* at Each Node

4.2 Protocol

A routing protocol can be divided into two phases, namely, *route construction* and *route maintenance*. During the *route construction* phase, routes are constructed between all pairs of nodes. The *route maintenance* phase takes care of maintaining loop-free routes in the face of unpredictable topological changes.

4.2.1 Route Construction Phase

The protocol to divide the network graph into clusters have been explained earlier. After clustering, each *boundary* node forwards the cluster information (i.e., cluster id and its members) to the other clusters it is part of. Along with the cluster information, a hop counter is included. The hop counter keeps track of the number of hops needed to reach any *boundary* node of that cluster. The *boundary* node of the new cluster increments the hop counter to 1 before forwarding the cluster information. If a *boundary* node gets information of a new cluster, it stores the cluster information in its *ClusterTable*, and the hop information in *AllRouteTable* and *RouteTable*. It increments the hop counter and then forwards the cluster information. A *boundary* node has to forward information of a new cluster only once.

Let us illustrate it with an example. In Figure 2, the *boundary* nodes are 3, 4, 7, 8, 12, and 16. Node 3 will send {A, (1, 2, 3)} (hop counter for A=1) to node 4 in cluster B, and, send {B, (4)} (hop counter for B=1) to nodes (1, 2) in cluster A. Since, 4 receives information of a new cluster (A), it increments the hop counter for A to 2 and forwards the cluster information of A. Thus, the *boundary* node 7 will get this information, increment the hop counter and forward the cluster information too. In this manner, the information of all clusters are distributed to all the nodes.

Upon receipt of information of all the clusters, the data structures *RouteTable*, *ClusterTable*, and *AllRouteTable* at each node will have the topology information of the whole network.

Each message packet will contain the identifier of the destination node in its header. When a node receives a message packet, it looks up the *ClusterTable* to determine the cluster in which the destination node is currently a member. Using the identifier of the destination node's cluster it looks up the *RouteTable* to determine the next hop node for the packet's destination. The node then forwards the message packet to the next hop node. This process of forwarding continues till the packet reaches its destination.

4.2.2 Route Maintenance Phase

This phase begins when there is a change in the network topology (host connection/disconnection, link failure/recovery). The route maintenance in our approach basically boils down to cluster maintenance. The protocols for cluster maintenance have been explained previously. Loop freedom can be achieved using techniques suggested in the existing literature [1, 2] e.g., sequence numbers, link status, etc.

The new cluster information will be propagated throughout the network. It should be noted that only the boundary nodes are responsible for broadcasting and re-broadcasting any new information. In addition, the boundary nodes need to broadcast the new information only once. Upon receipt of new information, nodes other than the boundary nodes just update their tables and do not re-broadcast. This helps in quick dissemination of information across the network. Thus, the reconvergence of the cluster-based protocols is very quick. Let us illustrate it with an example. Let node 2 in Figure 2 disconnect. This event will be detected by nodes 1 and 3. Since node 1 is not a boundary node, it will just update its tables to indicate the change. Node 3 being the boundary node broadcasts the new cluster information. Node 4, a boundary node, upon receipt of the new cluster information from node 3, re-broadcasts it. This broadcast will be received by nodes 3, 5, 6 and 7. Since node 3 has already broadcasted this cluster information, it neglects this information. Nodes 5 and 6 being non-boundary nodes just update their tables. However, node 7 being a boundary node, updates its tables and rebroadcasts the new cluster information. Similarly, other boundary nodes 8, 12 and 16 upon receipt of the new cluster information re-broadcast it so that every node in the network have the new cluster information. And, the non-boundary nodes just listen and do not re-broadcast.

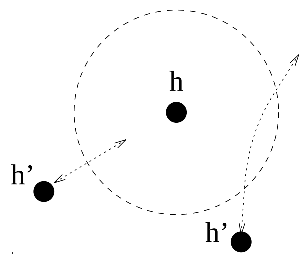

Figure 6: Movements That Cause Unnecessary Link Creations/Deletions

4.3 Implementation Details

- Detection of a new link : Each host periodically broadcasts a beacon, which, includes its identifier. If a host h receives a beacon from another host h' which is not in its current neighbor set, it means that there is a prospective new link to be created. However, the **Switch ON**(h') procedure is not immediately initiated. This is to avoid unnecessary oscillations due to the host h' moving in and out of host h's vicinity. Figure 6 shows the scenarios where the movement of h' could cause a sequence of unnecessary link creations/deletions.

- Detection of a link break : If a host h does not receive a periodic beacon from h' which is one of its cluster mates, it will assume that either h' has moved out of its vicinity (cluster) or that h' is disconnected. Host h will then follow the procedure for host disappearance as explained in Section 3.2.

5 Experimental Results

We performed experiments to determine the average size of clusters in random graphs. The clusters were determined using the **Switch ON** procedure described in Section 3.1. Later in the section we will compare the performance of the proposed cluster-based approach with couple of existing routing protocols.

5.1 Experimental Framework

Input to the simulations are (i) N (number of nodes), and (ii) ρ (ratio of the total area to the host coverage area (D/d)). We assume a square room to be the total area, where, D is the dimension of the square room. The value of d was chosen to be 10 units (The typical range of infra-red transreceivers which is commonly used for indoor wireless communication is of the order of 10 meters). The node positions ((x, y) coordinates) were chosen randomly using uniform distribution within the total area i.e., (0,0) to ($D-1$,$D-1$).

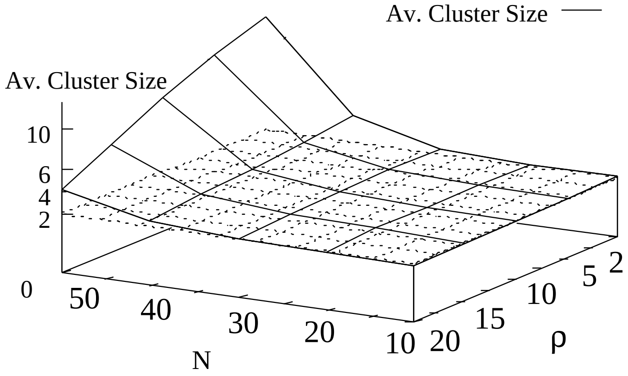

Figure 7: Average Cluster Size Vs. N and ρ

5.2 Results and Discussions

For each N (10, 20, 30, 40, 50), ρ (2, 5, 10, 15, 20), we ran the simulations for 10000 iterations. We believe that the results obtained are transient-free. It should be noted that *maximal* clusters are always sought. It was noticed that for a connected graph, the sequence of **Switch ON** procedures always produced a *cluster connected* graph. As shown in Figure 7, the average cluster size increases as N increases. It also increases when ρ decreases. For example, the average cluster size will be very high in a small crowded room. Any cluster size bigger than 2 will benefit the protocol. Figure 7 shows the region of values of N and ρ where the average cluster size is greater than 2. In these scenarios, clustering will benefit.

As ρ grows (as it would for public wide area services), N must be large to get an average cluster size of 2 or more. However, the effectiveness of the proposed approach is that the existing routing protocols can be directly applied to the network – replacing the nodes by clusters. Thus, when the average cluster size is less than 2, the proposed approach does not perform any worse than the existing routing protocols.

5.3 Performance Evaluation

5.3.1 Comparison with *DSDV* [1]

- Our approach does not require the frequent broadcasts of routing tables to the neighbors as long as there is no change in a cluster membership. The proposed approach will however incur some form of cluster maintenance overhead as explained in Section 5.3. However, the size of a periodic beacon from each host is much smaller than the size of a routing table.

- Quick reconvergence in *DSDV* is obtained by quick re-broadcast by each and every recipient of the broadcast, causing degradation of the availability of the wireless medium. However, in our approach, re-broadcast is done **only** by the *boundary* nodes. Nodes other than *boundary* nodes just listen.

- Memory overhead due to storage of data structures are considerably smaller for the cluster-based approach when the average cluster size is more than 2. This is due to the fact that the routing information in our protocol is cluster-based which is smaller than node-based in *DSDV*.

5.3.2 Comparison with Corson's Protocol [2]

- Routing optimality is of secondary importance in [2]; finding a route is what mattered. This reduces the topology update overhead, because, as long as there is a route to a destination available, any changes in link status to that destination will not cause new routes to be searched and created. If the goal of our approach were to be similar to [2], our approach will incur lower topology overhead because of the fact that broadcasts and re-broadcasts are done **only** by the *boundary* nodes.

- The novel property of the protocol in [2] is that the routing is "source-initiated". Instead of maintaining distances from all sources to a destination, the protocol guarantees route maintenance only for those sources that actually desire routes. This property helps in reducing the topology update overhead. Our approach does not restrict a routing protocol to maintain routes between all pairs of nodes. However, if we were to maintain routes to a destination for only those sources that actually needs them, the performance of our approach will be "at least" same as [2].

6 Conclusion

Proposed in this report is a new methodology for routing in mobile wireless networks. This report shows that routing protocols based on clusters could obtain performance improvements over previous approaches. Cluster-based protocols allow the network to enjoy the liberty of maintaining routes between all pairs of nodes at all times, without causing much network overhead. Thus, a compromise on routing optimality as suggested in [2] to avoid network congestion might not be required.

Similar to [2, 10] the cluster-based approach does not guarantee shortest path. This is due to the fact that *maximal* clusters are always sought in the proposed approach. We are currently involved in the analysis of the routing overhead. Routing overhead is ratio of the path length between the source and the destination as determined by the proposed algorithm and the actual shortest path length between them. We expect the path length determined by the cluster-based approach to be comparable to the shortest path length. The tradeoff of a routing algorithm in such high-rate topological change environment such as ad-hoc networks and packet radio networks, is between the overhead due to topology update messages, and the routing overhead. We are also currently analyzing the overhead due to cluster maintenance and the topology updates.

This report dealt with protocols for discrete events (host connection/disconnection, etc.). Future work will involve extensions of these protocols to support concurrent events. This report assumes that the transmitted packets are received correctly, i.e., reliable links. We are currently investigating schemes to tolerate corrupt wireless links.

Apart from providing connectivity in a dynamic topology, maintaining routing information (in mobile wireless networks) has other advantages such as, extending the base station area coverage (Figure 8); consequence of that is delayed and might be much smoother handoffs.

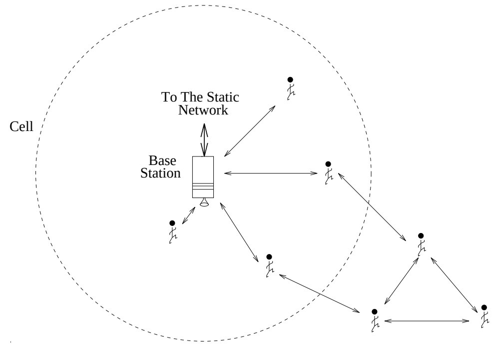

Figure 8: Extending base station coverage area

References

[1] C. Perkins, P. Bhagwat, "Highly Dynamic Destination Sequenced Distance Vector Routing (DSDV) for Mobile Computers," *Proc. ACM SIGCOMM Symposium on Communication, Architectures and Protocols,* pp. 234-244, 1994.

[2] M. Scott Corson, A. Ephremides, "A Distributed Routing Algorithm for Mobile Wireless Networks," to appear in *ACM Journal on Wireless Networks,* Vol.1, No.1, 1995.

[3] David B. Johnson, "Routing in Ad Hoc Networks of Mobile Hosts," *Proc. of Workshop on Mobile Computing and Applications,* Dec., 1994.

[4] P. Krishna, et. al., "A Cluster-based Approach for Routing in Ad-Hoc Networks," *Proc. of USENIX Symposium on Location Independent and Mobile Computing,* April, 1995.

[5] S. Mohan and R. Jain, "Two User Location Strategies for Personal Communication Services," *IEEE Personal Communications,* Vol. 1, No. 1, 1994.

[6] P. Krishna, N. H. Vaidya and D. K. Pradhan, "Efficient Location Management in Mobile Wireless Networks," Technical Report, Dept. of Computer Science, Texas A&M University, Feb., 1995.

[7] P. Krishna, N. H. Vaidya and D. K. Pradhan, "Location Management in Distributed Mobile Environments," *Proc. of Third Intl. Conf. on Parallel and Distributed Information Systems,* pp. 81-88, Sep., 1994.

[8] J. Ioannidis, et. al., "IP-based Protocols for Mobile Internetworking," *Proc. ACM SIGCOMM Symposium on Communication, Architectures and Protocols,* pp. 235-245, October 1991.

[9] C. Perkins, "IP Mobility Support," Internet Draft, IETF Mobile IP Working Group, Oct., 1994.

[10] E. Gafni, D. Bertsekas, "Distributed Algorithms for Generating Loop-free Routes in Networks with Frequently Changing Topology," *IEEE Trans. on Comm.,* January, 1981.

[11] D. Bertsekas, R. Gallager, *Data Networks,* Prentice-Hall, 1987.

[12] J. J. Garcia-Luna-Aceves, "A Unified Approach to Loop-free Routing Algorithm Using Distance Vectors or Link States," *ACM SIGCOMM Symposium on Communication, Architectures and Protocols,* Sep., 1989.

[13] J. M. Jaffe and F. M. Moss, "A Responsive Routing Algorithm for Computer Networks," *IEEE Trans. on Communications,* pp. 1758-1762, July, 1982.

[14] M. Schwartz and T. E. Stern, "Routing Techniques used in Communication Networks," *IEEE Trans. on Communications,* pp. 539-552, April, 1980.

[15] J. J. Garcia-Luna-Aceves, "Loop-Free Routing Using Diffusing Computations," *IEEE Trans. on Networking,* Vol. 1, No. 1, Feb., 1993.

[16] J. Jubin and J. D. Tornow, "The DARPA Packet Radio Network Protocols," *Proceedings of the IEEE,* Vol.75, No. 1, pp. 21-32, January, 1987.

[17] P. R. Karn, H. E. Price, and R. J. Diersing, "Packet Radio in the Amateur Service," *IEEE Journal on Selected Areas in Communications,* Vol. 3, No. 3, pp. 431-439, May, 1985.

[18] J. M. McQuillan and D. C. Walden, "The ARPA Network Design Decisions," *Computer Networks,* Vol.1, No.5, pp. 243-289, Aug., 1977.

[19] C. Hedrick, Routing Information Protocol, RFC 1058, June, 1988.

[20] J. M. McQuillan, I. Richer, and E. C. Rosen, "The New Routing Algorithm for ARPANET," *IEEE Trans. on Comm.,* Vol. 28, No. 5, pp. 711-719, May, 1980.

[21] W. Diepstraten, G. Ennis, and P. Bellanger, "DFWMAC - Distributed Foundation Wireless Medium Access Control," *IEEE Document P802.11-93/190,* Nov., 1993.