Scalable Democratic Routing in Wireless Mesh Networks

Ronit Nossenson
Faculty of Computer Science
Jerusalem College of Technology (JCT)
Jerusalem, Israel
nossenso@jct.ac.il

***Abstract* — We propose a new method for scalable routing in large wireless mesh network: the democratic routing scheme. In this new schema, the nodes are divided into components according to their connectivity classes. As oppose to hierarchical routing, here, all nodes in the component are equal. The routing decisions are performed according to nodes connectivity structure together with a proper routing performance metric. Every node holds a view including its neighbor set and a dynamic connectivity model of the network. The node uses the view to understand which of its topology changes should be announced and to identify the set of nodes that should get this specific update. In this way, the routing overhead is significantly reduced, and, yet, the necessary routing information is available.**

***Index Terms* — Wireless mesh network; scalable routing algorithms; connectivity models.**

I. INTRODUCTION

Various wireless networks have evolved into the next generation to provide better services. A key technology, Wireless Mesh Networks (WMN), has emerged recently [1]. A WMN is dynamically self-organized and self-configured, with the (possibly mobile) nodes in the network automatically establishing and maintaining mesh connectivity among themselves. In such systems, the implied routing protocol directly affects scalability, efficiency, and reliability.

When the network is large, hierarchical routing protocols tend to achieve better performance [2-10]. Hierarchical routing protocols build a hierarchy of nodes, typically through clustering techniques. These protocols divide the nodes in the network into backbone nodes and regular nodes arranged in clusters. Every cluster uses a *cluster head* node that is a part of the backbone. The cluster head node acts as a local coordinator of transmissions within the cluster and is responsible for keeping and updating routing information beyond the cluster. However, explicit clustering schemes have several drawbacks [3]:

- Clustering usually requires that the cluster heads will be more powerful (battery life, transmission range and capacity) than the regular nodes in the clusters. Unless being intentionally designed so, the cluster head may become a bottleneck.
- The complexity of maintaining the hierarchy compromises the performance of the routing protocol. There is a significant overhead in the maintenance of the cluster (e.g., electing the cluster head and maintaining the cluster's members);
- The centralization and load of the cluster-heads routes;
- The routing path may be longer than a direct path;
- Clustered protocols are sensitive to failures of the cluster heads;
- The fragmentation of the network nodes into clusters may result in a large number of clusters.
- Change of cluster heads results in routing changes and hence generates routing overhead.

In this research, we propose a completely different method for scalable routing in large WMN, the *democratic routing scheme*. In this new schema every node holds a *view*. The view includes the node neighbor set and a *dynamic connectivity model* of the network. Using its view, the node can keep updated information on the network and can perform educated routing decisions. The dynamic connectivity model represents the network topology in a compact structure (O(n), where n is the number of nodes in the network) and can be updated in an efficient manner to capture the dynamic changes in the network topology, as nodes move from one place to another while establishing and releasing links. On one hand, this model should be small to avoid unnecessary updates (meaning, high overhead); on the other hand it should be detailed enough to allow good routing decisions. The dynamic connectivity model is updated whenever an *essential* change in the network accrues. The node uses the view to understand which of its topology changes (for example, link establishment or link release) should be announced and what is the exact set of nodes that should get the specific update. In this way the routing algorithm overhead is significantly reduced and, yet, the necessary routing information is available.

We choose to describe our new concept using an extension of the cactus-tree model of Dinitz et al. [11] and the two-level cactus-tree model [12] to enjoy their elegancy and simplicity. However, these models support incremental maintenance only and they do not support node-deletion and edge-deletion (link release). Thus, we extend the models to support node-deletion and edge-deletion for low connectivity levels.

This paper is organized as follows. In the next section, related works are listed. In section III, basic definitions are described. Section IV presents the connectivity model dynamics. Section V describes the new democratic routing scheme. Finally, conclusions and further research topics are given in Section VI.

 ISBN: 978-1-61208-203-5

II. Related Work

Typically, when wireless network size increase flat routing schemes become infeasible because of link and processing overhead. One way to solve this problem and to produce scalable solutions is hierarchical routing. The common way of building hierarchy is to group nodes geographically close to each other into explicit clusters. In explicit clustering schemes, each cluster has a leading node (cluster head) to communicate to other nodes on behalf of the cluster. An alternate way is to have implicit hierarchy. In this way, each node has a local scope. Different routing strategies are used inside and outside the scope.

Cluster head-Gateway Switch Routing (CGSR) [7] is typical of explicit cluster-based hierarchical routing. A stable clustering algorithm, Least Cluster head Change (LCC), is used to partition the whole network into clusters, and a cluster head is elected in each cluster. A mobile node that belongs to two or more clusters is a gateway connecting the clusters. Packets are routed through paths having a format of "Cluster head–Gateway Cluster head–Gateway…" between any source and destination pairs.

Additional well known explicit clustering routing protocol is the Hierarchical State Routing (HSR) [5]. It is a multilevel clustering-based Link State routing protocol. It maintains a logical hierarchical topology by using the clustering scheme recursively. Nodes at the same logical level are grouped into clusters. The elected cluster heads at the lower level become members of the next higher level. These new members in turn organize themselves in clusters, and so on. The cluster head summarizes link state information within its cluster and propagates it to the neighbor cluster heads (via the gateways).

The basic idea in the Zone Routing Protocol (ZRP) [6] is that each node has a predefined zone centered at itself in terms of number of hops (implicit cluster). For nodes within the zone, it uses proactive routing protocols to maintain routing information. For those nodes outside of its zone, it does not maintain routing information in a permanent base. Instead, on-demand routing strategy is adopted when inter-zone connections are required.

A recent study [3] considers a routing technique which can implicitly cause nodes that are in the "center" of dense areas to act as cluster heads ("natural clustering"). Using the Metrical Routing Algorithm (MRA) [4], it maintains a dynamic set of coordinates to every node. Thus, if the coordinates of the destination are known, the MRA sends a message to this destination through the shortest path based on the estimated metrical distances.

We propose a new democratic routing scheme which can be classified as an implicit clustering. In this scheme, the nodes are logically divided into their connectivity classes (components). The components do not have component heads and the traffic is handled in a complete democratic manner. That is, in our scheme all nodes are equal. The routing decisions are performed according to nodes connectivity structure together with a proper routing performance metric (for example, minimum hops metric). Unlike other implicit clustering schemes, in our solution, nodes do not have geographic location information on the other nodes (coordination). In addition, we use the same routing protocols for routing inside and outside the connectivity components.

III. Definitions

Let G=(V,E) be a weighted undirected connected multi-graph without loops induced from a specific network topology, where every vertex represents a node in the network and every edge between two vertices represents a wireless link between a pair of corresponding nodes that are in communication range.

A minimal edge-cut C of G is an edge set whose removal disconnects G and removal of any proper part of C does not disconnect G. If |C|= k then C is called a k-cut. If C={e} (that is |C|=1) then the edge e is called a *bridge*. Two vertices {u,v} are called k-edge-connected if no k'-cut, k' < k, separates u from v. It is well known that the property ``there exist k edge-disjoint paths between u and v in G" defines the same relation as k-edge-connectivity. The equivalence classes of this relation are called the k-edge-connected classes (*k-classes* for short). The partition of V into the (k+1)-classes is a refinement of the partition of V into k-classes. Thus, the connectivity classes have a hierarchical structure.

For a k-connected graph, its connectivity model represents both its (k+1)-classes and its k-cuts. For example, the well known bridge-tree model of a 1-connected graph represents its 1-cuts (the so-called *bridges*) and its 2-classes [16]. Similar, the cycle-tree connectivity model of a 2-connected graph represents its 2-cuts and its 3-classes [13] [14]. These connectivity models are, in fact, special cases of a more general connectivity model called the cactus-tree model [11]. The cactus-tree model [11] of a k-connected graph represents both its (k+1)-classes and its k-cuts.

For the simplicity of this short presentation, we assume that the network is not highly connected and use the bridge-tree model of [16] together with the cycle-tree model of [14] [15] for each 2-class in the graph. This joined two-level connectivity model is, in fact, a special case of the two-level cactus-tree model of [12]. Using the general model of [12], our result can be easily adjusted to highly connected networks.

By definition, the size of this connectivity model is O(n), where n is the number of nodes in the network. If n is very large, it is possible to divide the network according to geographic location and to define proper gateways nodes to connect the networks parts.

Let S be a sub-set of V. The induced graph G(S) consists of the vertices S and all edges in E connecting vertices in S. For a 3-class S, the associated 3-componnet graph is the induced graph G(S) together with virtual edges. The virtual edges represents cycles in the cycle-tree model that are attached to distinct vertices of S. The 3-commponent mimics the connectivity structure of S in a localized fashion [13].

 ISBN: 978-1-61208-203-5

IV. CONNECTIVITY MODEL DYNAMICS

Each node in the network holds a *view*. The view of node v includes:

- The global connectivity model of the network (the bridge-tree and the cycle-tree of each 2-class),
- Its local connectivity model: the cactus-tree model of v's 3-componnent graph.
- The set of v's neighbors

In this section, we describe the dynamics of the connectivity model. The *vertex insert* and *edge insert* procedures are given in [12]. Here, we provide an intuitive explanation, and present two examples for the model completeness. For formal description of these procedures, see [12] [14]. We add new procedures to support *vertex delete* and *edge delete*. Note that the new transformations described here are designed for the simple case of low connectivity only.

Regarding a new vertex insertion, the vertex is inserted as a singleton. Meaning, until edge insertion it act as an isolated node. The corresponding connectivity models are trivial.

Regarding a new edge e=(u,v) insertion (new link establishment), there are four cases according to the relation between the vertices u and v. The four cases are: u and v belong to distinct 1-classes, u and v belong to the same 1-class but to distinct 2-classes, u and v belong to the same 2-class but to distinct 3-classes, u and v belong to the same 3-class.

The change in the connectivity models due to new link establishment between two nodes belonging to separated networks (1-classes) is a simple connection of the two representing models by adding a new bridge associated with the new link e. To connect nodes with higher connectivity a simple *squeeze* operation is performed on the path-of-edges-and-cycles, in which the set of all nodes along the path are replaced by a single node [12] [14].

The following new procedure defines the vertex deletion operation.

```
Void VertexDelete(v)
Begin
1: for every edge e attached to v do {
2:   remove edge (e); }
3: release vertex v;
End
```

This procedure functionality includes releasing of all the edges attached to this vertex and then releasing the vertex resources (memory etc.).

Theorem 1

The procedure VertexDelete(v) correctly updates the connectivity models.

Regarding an edge deletion operation (see EdgeDelete procedure below), we have three cases. Assume that the edge e=(u,v) needs to be deleted. First, the vertices u and v can belong to the same 1-class but to distinct 2-classes. Second, the vertices u and v can belong to the same 2-class but to distinct 3-classes. In the third case, u and v belong to the same 3-class.

Consider the first case (u and v belong to the same 1-class but to distinct 2-classes).

Lemma 2

Assume that e=(u,v), u and v belong to the same 1-class but to distinct 2-classes. Then, e is a bride in the graph and has direct representation in the bridge-tree model.

Proof

By bridge definition.

In this case (steps 1-3 in the EdgeDelete procedure below), the global connectivity model is changed by removing the edge which represents e in the model. The local connectivity models of u and v are not affected since u and v do not belong to the same 3-class (component).

Consider the second case (u and v belong to the same 2-class, but to distinct 3-classes).

Lemma 3

Assume that e=(u,v), u and v belong to the same 2-class but to distinct 3-classes. Then, e has direct representation in the cycle-tree model of the 2-class.

Proof

Since u and v belong to the same 2-class but to distinct 3-class there are exactly two edge-paths between u and v. One path consists of {e} and let us denotes the second one by P. Removing e together with any edge from P result in a minimal cut that separates u and v and must be represented in the cycle-tree. That is, the cycle L= {e}U{P} is in the cycle-tree.

In this case (steps 4-6 in the EdgeDelete procedure below), the global connectivity model is changed by transforming the cycle which include the edge which represents e into a path of bridges by removing this edge. The local connectivity models of u and v are not affected since u and v do not belong to the same 3-class and to the same 3-componnent. In addition, by definition, the cycle L is not represented by a virtual edge in the 3-componnets of u and v.

Consider the third case (u and v belong to the same 3-class, denoted by S). Removing an edge in this case might result a fragmentation of the 3-class S. Formally, now we have two cases: u and v can belong to the same 4-class or they can belong to distinct 4-classes. If u and v belong to the same 4-class (steps 18-23 in the EdgeDelete procedure below) then the global connectivity model will not change (the class S is still a 3-class) as a result of removing e. Only the local cactus-tree might change as follows. The node that represents S is replaced with a cactus-tree (*implanting* the model instead of the node). This is done via 3-componnent discovery operation in addition to virtual edges that result from the cycle-tree.

In the second case (steps 7-17 in the EdgeDelete procedure below), the node that represents S in the global

connectivity model, should be replaced with a cycle-tree that will represent S as a 2-class (*implant* the model instead of the node). That is, if u and v belong to distinct 4-classes then the removal of e will change the 3-class S and turn it to a 2-class. The cycle-tree that represents S is generated from the cactus-tree model of the 3-componnent associated with S in the following manner. Let T be a path in the cactus-tree between the node that represent the 4-class of u and the node that represents the 4-class of v. Every 3-cut that is represented by an edge on this path is now a 2-cut. Thus, the cycle-tree of S is a sequence of cycles (each of size 2). All nodes in the cactus-tree before the path T stay in one 3-class, and all nodes after this path stay in one 3-class. Each node in the path T is now a 3-class. In addition, each node in these new 3-classes should construct a new local connectivity model: the cactus-tree of its new 3-class. This is done via 3-componnent discovery operation in addition to virtual edges that result from the cycle-tree.

```
Void EdgeDelete(u,v)
Begin
1: if (u and v belong to the same 1-class
     but to distinct 2-classes) do {
2:   Remove e from the bridge-tree;
3:   Update u and v global models;}
4: else if (u and v belong to the same
     2-class but to distinct 3-classes) do
{
5:   Remove e from the cycle-tree;
6:   Update u and v global models;       }
7: else if (u and v belong to the same
     3-class but to distinct 4-classes) do
{
8:   find the path between u's 4-class and
     v's 4-class in the cactus-tree;
9:   create the proper cycle-tree from the
     cactus-tree;
10:  implant the new cycle-tree in the
     global model;
11:  Update the global models of all nodes
     in the 3-class;
12:  for every new 3-class created do{
13:     discover the 3-componnet;
14:     add virtual edges according to the
          cycle-tree;
15:     calculate the cactus-tree;
16:     for every node in this 3-class do{
17:        Update the local model; } } }
18: else if (u and v belong to the same
     4-class) do {
19:  discover the 3-componnet;
20:  add virtual edges according to the
          cycle-tree;
21:  calculate the cactus-tree;
22:  for every node in this class do {
23:  Update the local model; } }
End
```

The example plotted in Figure 1 describes a delete edge operation, where e=(u,v) and the vertices u and v belong to the same 3-class (denoted by N_2 in the figure). The induced graph is presented in (a); the global connectivity-model of this network is presented in (b), it consists of all 1 and 2 – minimal cuts of the graph. In (c), the 3-componnent corresponding to the 3-class N_2 is presented together with the local model of the vertices in the 3-class N_2. In addition, we can see the transformation of this 3-class due to the removal of the edge e, with the proper transformation of its cactus-tree model (representing the minimal 3-cuts of 3-class N_2) into a cycle-tree (representing the minimal 2-cuts of 3-class N_2). Finally, (d) shows the updated global connectivity model.

Theorem 4

The procedure EdgeDelete(v) correctly updates the connectivity models.

V. The Democratic Routing Scheme

In this section, we describe the democratic routing scheme. As mentioned before, each node has its view. The node uses the view to decide which of its topology changes should be announced and the exact set of nodes that should get the specific update.

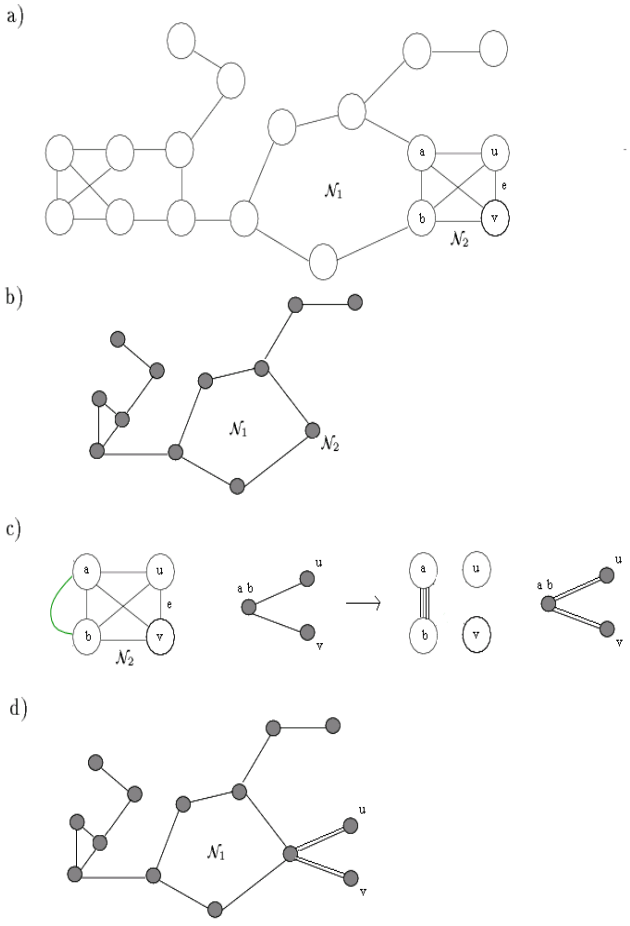

Figure 1. Deleting an edge between vertices belonging to the same 3-class but to distinct 4-classes

When a new node joins the network it performs an Insert vertex operation. As a result, its view is initiated. Next, the node discovers its neighbors. For each link establishment between the node and an adjusted node, the node received the neighbor's global and local connectivity model. Once an update is required on the global connectivity model, an 'update global model' message is broadcast to all nodes. This message includes the origin node identity, a sequence number generated in the origin node, and the update description. An important observation is that a node which receives two distinct 'update global model' messages can join these messages into one message including both updates and by that to reduce the routing overhead.

Once an update is required on the local connectivity model, an 'update local model' message is send to the nodes in the 3-componnent only. If there are no updates to the connectivity models, no updates messages are sent. In component discovery message, each node which belong to the specific 3-class responses with its neighbors set.

As a node move from one place to another its links changes. Some new links are added, some old links are been released. The node continues to update the necessary routing information and the truly important network topology changes are continually monitored.

Since the connectivity models can be decomposed according to the network topology, it is possible to divide the network to sub-networks and to define the models for each part. Using special nodes as gateways can solve the problem of connecting the sub-networks.

VI. CONCLUSION AND FUTURE WORK

In this on-going research, we proposed a new democratic routing scheme. In this scheme all nodes are equals and routing decisions are based on the nodes view of the network. To limit the routing algorithm overhead, the nodes use connectivity model to decide which node should be informed of a topology change if any.

Future work includes implementation of this scheme and evaluation of the algorithm performance comparing to others scheme for scalable routing in large wireless mesh networks.

REFERENCES

[1] Akyildiz, I., F., Wang, X., and Wang W., "Wireless mesh networks: a survey", Computer Networks 47 (2005) pp. 445–487, Jan 2005.

[2] Belding-Royer, E., M., "Multi-level hierarchies for scalable ad hoc routing", ACM/Kluwer Wireless Networks vol. 9 issue 5 pp. 461–478, 2003.

[3] Ben-Asher, Y., Feldman, S., Feldman M., and Gurfil, P., "Scalability Issues in Ad-Hoc Networks: Metrical Routing Versus Table-Driven Routing", Wireless Pers Commun (2010) 52:423–447.

[4] Ben-Asher, Y., Feldman, S., and Feldman, M. "Ad-hoc routing using virtual coordinates based on rooted trees", In IEEE SUTC, Taiwan, 2006.

[5] Chiang, C., and Gerla, M., "Routing and Multicast in Multihop, Mobile Wireless Networks," Proc. IEEE ICUPC '97, San Diego, CA, Oct. 1997.

[6] Haas, Z. J. and Pearlman, M. R., "The Performance of Query Control Schemes for the Zone Routing Protocol," ACM/IEEE Trans. Net., vol. 9, no. 4, Aug. 2001, pp. 427–38.

[7] Pei, G. et al., "A Wireless Hierarchical Routing Protocol with Group Mobility," Proc. IEEE WCNC '99, New Orleans, LA, Sept. 1999.

[8] Hagouel, J., "Issues in Routing for Large and Dynamic Networks", PhD thesis, Columbia Univ., May 1983

[9] Saha, A., K., Johnson, D., B., "Self-organizing hierarchical routing for scalable ad hoc networking", Technical Report, TR04-433, Department of Computer Science, Rice University

[10] Tsuchiya, P., F., "The landmark hierarchy: A new hierarchy for routing in very large networks", ACM SIGCOMM Comput. Commun. Review 18, 4, pp. 35–42, August 1988.

[11] Dinic, E., A., Karzanov A., V., and Lomonosov, M. V., "On the structure of the system of minimum edge cuts in a graph", Studies in Discrete Optimization, A. A. Fridman (Ed.), Nauka, Moscow, 1976, pp. 290-306 (in Russian).

[12] Dinitz, Ye., and Nutov, Z., "A 2-level cactus tree model for the minimum and minimum+1 edge cuts in a graph and its incremental maintenance", Proc. the 27th Symposium on Theory of Computing, 1995, pp. 509-518.

[13] Dinitz, Ye., "The 3-edge components and the structural description of all 3-edge cuts in a graph", Proc. 18th International Workshop on Graph-Theoretic Concepts in Computer Science (WG92), Lecture Notes in Computer Science, v.657, Springer-Verlag, 1993, pp. 145-157.

[14] Dinitz Ye., and Westbrook, J., "Maintaining the Classes of 4-Edge-Connectivity in a Graph On-Line", Algorithmica, Volume 20, Number 3, 1998, pp. 242-276.

[15] Galil, Z., and Italiano, G., F., "Maintaining the 3-edge-connected components of a graph on line", SIAM J. Computing 22(1), 1993, pp. 11-28.

[16] Westbrook, J., and Tarjan, R., E., "Maintaining bridge-connected and biconnected components on line", Algorithmica, 7, 1992, pp. 433-464.

 ISBN: 978-1-61208-203-5