On The Implementation and Experiences of A MIL-STD-188-220C Ad Hoc Routing Protocol

Chih-Min Yu[1*], Wen-Cheng Hsiao[2], Hsueh-Chin Cheng[3], Chuan-Yi, Hsieh[4]
Information and Communication Research Division,
Chung-Shan Institute of Science and Technology,
Taoyuan, Taiwan 325, Republic of China
Tel: +886 3 4712201ext 353369[1,2,3,4]
Fax: +886 3 4712591 [1,2,3,4]

E-mail: hankycm@ms47.hinet.net[1], shiao@exodus.cs.ccu.edu.tw[2], s0321506@ncnu.edu.tw[3]

Abstract

MIL-STD-188-220C defines a routing protocol to achieve the unicast and multicast capabilities for a multihop ad hoc network. In order to achieve these capabilities, a topology update and a source directly relay modules are defined for the routing protocol.

In this paper, we implement this routing protocol over the 802.11b testbed and evaluate its system performance. During the implementation, several technical issues are discovered and the corresponding solutions are proposed to realize this routing protocol. In the topology update module, a topology update algorithm is implemented. This algorithm solves the assumption issue of this protocol in the standard and can compute the topology table with two shortest paths correctly. In the source directly module, a common path selection algorithm is proposed and implemented. This algorithm is used to select the common shortest paths from a source node to multiple destinations and saves wireless bandwidth to deliver packets. Furthermore, the system throughput performance is measured to demonstrate that both modules and their algorithms can operate correctly. Finally, the performance results also show that our routing protocol over the testbed is feasible for both unicast and multicast applications of multihop as well as mobile ad hoc networks.

Index Terms—Ad hoc network, routing protocol, multicast application, spanning tree

1. INTRODUCTION

Currently, routing protocols used in ad hoc networks can be divided into three categories: proactive, reactive, and hybrid routing protocols. In the proactive category [1], each node maintains a routing table. The advantage is little delay involved in determining a route to achieve the QoS requirement but causes the overhead in routing information exchanges. In the reactive category [2], a flooding method is usually used to search for the optimal path from a source node to a destination node and this will incur a certain amount of delay. However, this reactive approach provides better network scalability. In the hybrid category [3], a proactive scheme is used locally and a reactive scheme is used globally to discover a route. Optimally, this protocol achieves the advantages of both proactive and reactive categories.

MIL-STD-188-220C routing protocol [4] is designed for the mobile ad hoc network that attempts to achieve the unicast and multicast applications in military operational environment. This routing protocol belongs to the proactive category that each node maintains the global network topology and exchanges the routing information with its nearest (one-hop) neighbors periodically. As a result, this protocol allows each node to conduct the packet transmission for unicast and multicast applications [5] as needed. In order to achieve the routing capability, a topology update module is defined. To support the unicast and multicast applications, a source directly relay module is defined.

In this paper, we implement both the topology update and source directly relay modules over the 802.11b platform to build a multihop ad hoc testbed [6][7]. During the implementation, several issues are discovered and the corresponding solutions are proposed. In the topology update module, an assumption issue is how to be aware of the nearest neighboring information for each node. The other issue is how to realize the topology update algorithm that is used to compute the network topology table. In the source directly relay module, the implementation issue is how to choose the final selected paths from the potential paths for multicast applications since the standard does not defined these issues clearly.

In order to deal with the assumption issue, a hello-like message is combined into the topology update message to broadcast its own information for all nodes to be aware of its nearest neighbors. In the topology update algorithm, an identical redundant path condition rule is added in this algorithm to compute and update the topology routing information correctly. In the source directly module, a common path selection algorithm is proposed to select the common shortest paths for multicast applications. Finally, a testbed is built to measure and evaluate the

system throughput performance of this routing protocol.

The rest of this paper is organized as follows. Section 2 briefly describes the topology update and source directly relay module of MIL-STD-188-220C routing protocol and discusses the potential implementation issues. Section 3 describes the corresponding solutions for those implementation issues and lists the pseudo code for both algorithms. Section 4 evaluates the system performance of this implemented routing protocol through the measuring results of testbed and Section 5 concludes the paper.

2. MIL-STD-188-220C ROUTING PROTOCOL OVERVIEW

2.1 Topology Update Algorithm

In this section, an example is used to illustrate the topology update algorithm. Figure 1 shows a link diagram of a sample network. In this network, all nodes are communicated within a single wireless transmission channel. Each node labeled 1 through 8 is a radio communication processor.

Assuming each node knows about its nearest neighboring information initially, the topology update message is used to exchange the topology information in order to build up a complete network topology at every node.

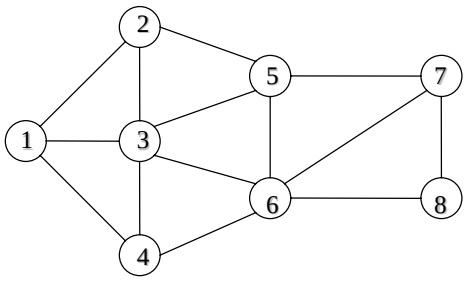

Figure 1. Link diagram of a sample network

Then each node will gain more topology information by exchanging its own topology routing tree with the nearest neighbors periodically. Finally, The resulting topology routing tree information will be stored in each node's topology table.

In order to compute the topology routing tree in a topology table, three topology update rules are defined in the standard for the topology update algorithm. First, before the topology routing tree is saved in a topology table, each node will prune all identical redundant links. Second, only the shortest paths from the source node to all destination nodes are retained. Third, at most two shortest paths can be retained for redundant paths to other destinations.

After several iterations of topology update message exchanges between nodes, each node will update the topology routing tree with these three rules. The final topology routing tree is computed and updated to the topology table. Table 1 lists the final converged topology table for node 1. The column of node predecessor represents the upstream node of destinations. The column of hops is used to specify the hop distance from source node to destination nodes.

After we investigated the behavior of the routing algorithm, two implementing issues are found. One is the assumption issue of how to be aware of the nearest neighboring information for every node. The other issue is how to implement this topology update algorithm correctly.

Destination	Node Predecessor	Hops
2	1	1
3	1	1
4	1	1
5	2	2
6	4	2
5	3	2
6	3	2
7	5	3
7	6	3
8	6	3

Table 1. Final topology table of node 1

2.2 Source Directly Relay Algorithm

For packet transmission, each node can send data to either one or many destinations by the source directly relay module together with the topology update module. This module can compute the shortest transmission paths from one source node to multiple destination nodes from the final topology table as Table 1. Then this module assigns the whole paths in packet header as source routing method to deliver packets. In packet reception, this module also checks the received packet header and decides whether to receive or forward packets.

Here we describes the algorithm in this standard for computing the common shortest path to all selected destinations. First, each node needs to select the whole potential paths from source to all destinations. Then it compares all potential paths with common nodes to decide the most one potential path for each destination. Finally, this node merges all these paths together and generates the common shortest path and puts in the packet header to deliver packets to multiple destinations at one time to save network bandwidth.

Figure 2 shows the topology routing tree of Table 1. This is a spanning tree structure rooted from source node to all destinations with up to two shortest paths. For example, when node 1 sends packets to destinations 5 and 6, the potential paths are 1-2-5, 1-3-5, 1-3-6, and 1-4-6. Of these potential

paths, the most one potential path for destination 5 an 6 are 1-3-5 and 1-3-6 respectively.

After we investigated this example of this algorithm, an implementation issue is found that the standard does not define clearly of how to implement this algorithm.

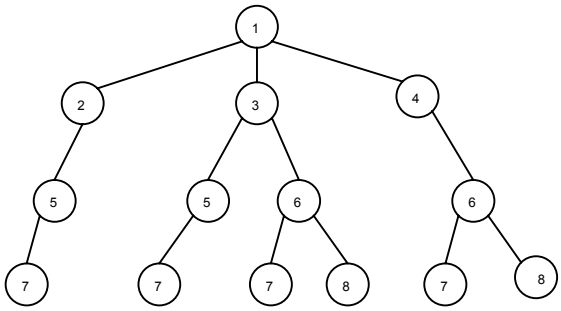

Figure 2. The final topology routing tree of node 1

3. ALGORITHM IMPLEMENTATION

In this section, we deal with the implementation issues and the corresponding solutions for this routing protocol. First we deal with the assumption issue that each node is aware of its nearest neighboring information initially. Then we describe our implemented topology update algorithm. Finally, a common path selection algorithm is developed for source directly relay module to figure out the resulting paths from the potential paths for all designated destinations.

3.1 Implementation of Topology Update Algorithm

In order to be aware of the nearest neighbors for each node, a hello-like message is designed as Figure 3. This routing entry is added into the topology table that is shown in Table 1 and then sent out by the topology update message. This routing entry represents the initial topology table format for each node that contains only the information of node itself.

After exchanges this routing information by topology update messages with its nearest neighbors, each node obtains the routing information of the nearest neighbors.

Node Address	Node Predecessor	Hops
1	0	0

Figure 3. The initial routing table at each node

In the topology update algorithm, each node updates its topology table from the received topology update message by the three topology update rules described in section 2.1. Whenever each node receives new topology update message, the topology table is updated as the topology changes. Since each node may retain up to two shortest paths to all destinations, a phenomenon called the identical redundant path problem is discovered during implementation.

Here we use an example to illustrate this problem. In Figure 4, node 1 contains the nearest neighboring information after receiving the initial topology of its nearest neighbors. First node 1 exchanges routing information with node 2 and updates its topology table in accordance with the three topology update rules. Then node 1 receives the topology table from node 3 and updates its topology table. As a result, the topology table contains two identical redundant paths 1-4-5 with 3 hops after computation. This condition will make the other useful shortest path cannot be included and updated in the topology table since each node can only retain up to two shortest paths. Therefore, an additional rule needs to be added and checked for this problem during topology table updates.

The detailed topology update algorithm is described by the pseudo code listed in Figure 5. This algorithm considers all three criteria including the non-identical path condition to check the identical redundant path problem. With this new criterion, we can make the final resulting topology table fulfill the standard requirement and generate the correct topology table as Table 1 and the topology routing tree in Figure 2.

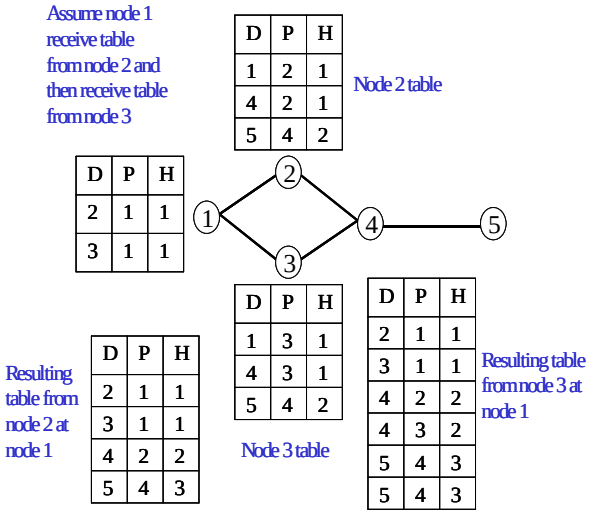

Figure 4. An identical redundant path problem

```
Topology_update_proc() {
do{
   for (node i and j, i≠j){
      if (node i connects with node j){
         for (every j's routing entry m){
            for (every i's routing entry n){
               if (identical link)
                No update node i's routing entry n;
                else
                 if (shorter path exists)
                    update node i's routing entry n;
                else
                if (same shortest path length exists){
                   if (shortest path num < 2 and
                      non-  identical path)
                   insert m into node i's routing entry;
                }
            }
            if (no condition matches)
               insert m into node i's routing entry;
         }
      }
   }
}while (routing table of any node i changes);
}
```

Figure 5. Topology update algorithm

3.2 Implementation of A Common Path Selection Algorithm

In order to determine the final selected paths from the potential paths for multicast applications, a common path selection algorithm is developed.

From the topology routing tree structure in Figure 2, we observe an interesting phenomenon that the identical nodes in the topology routing tree have the same descendant nodes. In order to generate this multicast routing table, we search the identical node in the column of destination from the smallest hop distance to the highest in Table 1.

When the identical nodes are discovered, we locate their ancestor nodes and then compare the total number of immediate descendant nodes of these ancestors. The immediate descendant nodes of the ancestor are defined as the downstream nodes that directly connect with the ancestor node.

If more than two ancestor nodes have the same largest immediate descendant nodes, one ancestor node is selected randomly. As a result, only one ancestor with the largest immediate descendant nodes will be kept. However, the other identical nodes and their descendant nodes along the downstream branch of identical nodes will be pruned.

Then we deal with all other identical nodes using the same criterion and keep only one identical node. Finally, the selected paths for all destinations will be stored in the multicast routing table.

For example, for node 1 in Table 1, there is no identical node in the column of one hop. Then we search the column for two hops, there exist two identical nodes for node 5 and its ancestors are node 2 and node 3. Node 3 owns two immediate descendant nodes and node 2 owns only one immediate descendant node. Then the path from node 3 to node 5 is kept and the path from node 5 to node 7 including node 5 is pruned. We process other identical nodes in accordance with this rule continuously. The resulting selected paths for node 1 to all other destinations are shown in Figure 6. The detailed algorithm to generate the multicast routing table is described in Figure 7.

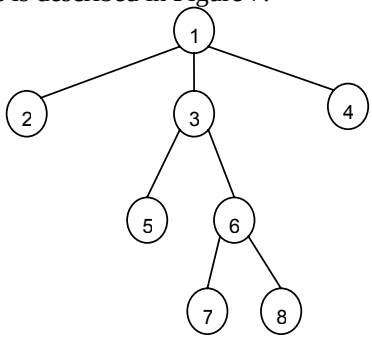

Figure 6. The selected paths to all destinations

```
Common_path_selection_proc()
{
   int ancestor[an_size];
   int max_no;
   while (search hop <= largest hop distance)
   {
      for (search all nodes which have the same hops)
      {
            if (identical nodes exists)
            {
           find all identical nodes' ancestors and
           store them in ancestor[an_size]
            }
      }
      for (j=1;j<=an_size;j++)
      {
      compute the numbers of 1-hop children for
      all the ancestors which in the ancestor array
      and choose the ancestor which has the largest
      1-hop children and store it as max_no
      if (more than two ancestors have same 1-hop nodes)
      {
      randomly select one ancestor and store it as max_no
      }
      }
      while (search the whole topology  routing table
               entries)
      {
      prune all identical nodes and their descendants
      except which ancestor is max_no
      }
   }
}
```

Figure 7. The common path selection algorithm

4. TESTBED PERFORMANCE

In this section, both algorithms are implemented in the Linux operation system kernel 2.4 over the 802.11b platform and the system performance is evaluated.

4.1 Emulation Scenario

In order to simulate the multihop scenario and measure the routing protocol performance, a string topology is used that is shown in Figure 8. We also edit the IP table in each platform to make nodes only deal with the topology update messages from its nearest neighbors for indoor performance measurement.

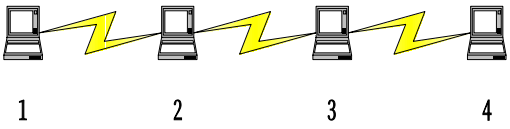

Figure 8. The multihop ad hoc testbed scenario

4.2 Average Throughput Performance

A network performance and measurement tool Iperf [8] is used to measure the throughput performance for the following two network configurations. This source node delivers UDP packets to destinations and the destinations measure the throughput results. These two configurations include an indoor configuration and a mobility configuration.

4.2.1 Indoor Configuration

The indoor connection configuration is same as Figure 8. Node 1 sends out the multicast packets and node 2, 3, and 4 measure the desired throughput performance. The performance result is shown in Figure 9. The resulting throughput performance is decreasing when the hop distance is increasing. This performance results reflect that the hop distance has significant influence to the throughput performance for this testbed. In addition, we demonstrate that the source directly relay algorithm can also operate correctly.

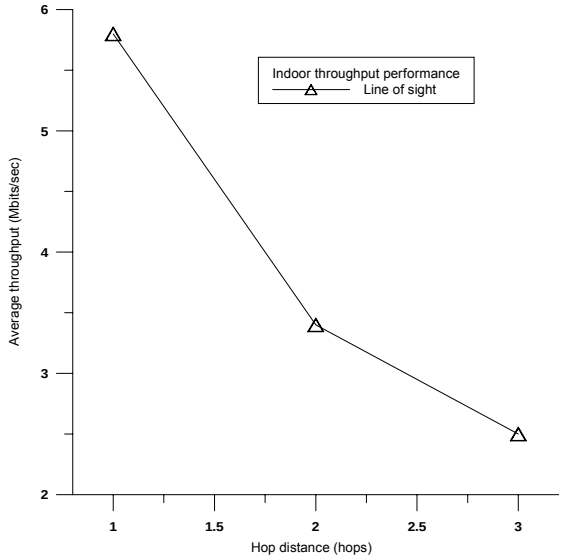

Figure 9. Throughput of indoor configuration

4.2.2 Mobility Configuration

In this mobility configuration, only two nodes are put into vehicles to measure the throughput performance in outdoor environment. Each node is mounted with a 5 dBi omni-directional vehicle antenna to achieve 300 meter communication distance with an average 6 Mbits/sec transmission rate in an outdoor static environment.

Two mobility cases are used and measured for throughput performance. One is the group movement pattern in which two nodes go around a building together with a 40 km/hr speed. The other one is the relative movement pattern that two nodes head to each other with 40 km/hr speed in the line of sight environment. Figure 10 shows the throughput performance results for these two movement cases.

With the result of group movement case, there is a deep fade in throughput performance. From our observation, this performance is seriously influenced by the shadow effect when two nodes cannot see each other directly by building blocking. Otherwise, the average throughput performance of group movement pattern is almost same as the static case when two nodes can see each other directly.

With the result of relative movement case, the average throughput is about 4.5 Mbits/sec in this high relative speed environment. This throughput performance is smaller than the group movement case in the line of sight condition. This result reflects that the mobility issue has significant influence on this throughput performance.

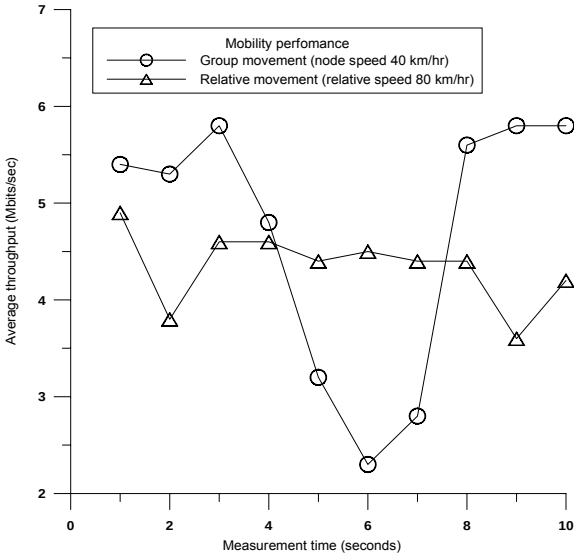

Figure 10. Throughput of mobility configuration

5. CONCLUSIONS

In this paper, we build a multihop ad hoc testbed by implementing the MIL-STD-188-220C routing protocol over the 802.11b platform. During the implementation, some technical issues were found and we developed our solutions for implementing this routing protocol.

In the topology update module, a hello-like routing entry is designed to deal with the assumption issue. In addition, the topology update algorithm is implemented in accordance with the three topology update rules. Besides, the identical redundant path problem is solved in this algorithm and the topology table is updated correctly. In the source directly relay module, a common path selection algorithm is developed and implemented to select the common shortest paths for multicast applications.

Finally, the performance results also show that our MIL-STD-188-220C routing protocol over the 802.11b testbed is feasible for both unicast and multicast applications of multihop as well as mobile ad hoc networks.

REFERENCES

[1] T. Clausen et al., Optimized Link State Routing Protocol, *IETF MANET Working Group Internet Draft, draft-ietf-manet-olsr-06.txt*, Sep. 2001.

[2] D. B. Johnson, D. A. Maltz, and Y. C. Hu, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks, *IETF MANET working group, draft-ietf-manet-dsr-07.txt*, Feb. 2002.

[3] M. R. Pearlman and Z. J. Haas, "Determining the Optimal Configuration for the Zone Routing Protocol," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, Aug. 1999.

[4] DoD Interface Standard-Digital Message Device Subsystems (MIL-STD-188-220C), May. 2002.

[5] B. Wang and C. J. Hou, "A Survey on Multicast Routing and its QoS Extension: Problem, Algorithm, and Protocol", *IEEE Network Magazine*, Jan./Feb. 2000, pp. 22-36.

[6] E.M. Royer and C.E. Perkins, "An Implementation Study of the AODV routing protocol", *IEEE Wireless Communication and Networking Conference*, 2000, pp. 1003-1008.

[7] D. A. Maltz , J. Broch, and D. B. Johnson, "Lessons from a full-scale multihop wirelsss ad hoc network testbed", *IEEE Personal Communications*, Feb. 2001, pp. 8-15.

[8] http://dast.nlanr.net/projects/iperf. "Network performance and measurement tools Iperf", Version 1.7.0, March 2003.