SIMULATION OF AN EVENT-DRIVEN WIRELESS SENSOR NETWORK PROTOCOL FOR ENVIRONMENTAL MONITORING

Maher Al Rantisi[1], Ali Maqousi[2], Glenford Mapp[3a], Orhan Gemikonakli[3b]

Specialized Technical Services (STS), Amman, Jordan[1]
Faculty of Information Technology, Petra University, Amman, Jordan[2]
School of Science and Technology Middlesex University, London, UK[3]
mr818@mdx.ac.uk[1], amaqousi@uop.edu.jo[2], g.mapp@mdx.ac.uk[3a], o.gemikonakli@mdx.ac.uk[3b]

Abstract

Sensor nodes in Wireless Sensor Networks (WSNs) are connected via wireless links. The function of these nodes is to sense particular events when placed in the environment and to send the sensed data to a central processing station usually via a base-station. Many routing, power management and data dissemination protocols have been developed for WSNs. However, such work poses some challenges. Because the data must be delivered in real time and in all circumstances, a reactive and robust routing protocol is needed to deal with changing outdoor enviromments. For example, with limited water resources in Middle East, moisture soil measurements must be used to manage irrigation and agricultural projects. The main factors for changes in the soil moisture are the seasonal rains. So an event-based routing protocol to determine the correct path used for sending the data is needed.This paper outlines the challenges in supporting such an environment and demonstrates a solution approach based on the modification of the well known reactive routing protocol, AOMDV, as well as different routing algorithms that are able to adapt to changes in part or all of the sensor network. In this paper, a scenario consisting of 7 nodes is used. Packets are sent from Node 0 to Node 3. AOMDV has been changed to be an event-based reactive protocol so when a node (in this case: Node 5) has signalled it has detected rain, the routing protocol is changed to route data away from the rain-affected node.

Keywords - Wireless Sensor Networks, Reactivity, Reliability, Soil Moisture

1. INTRODUCTION

The rapid evolution of wireless technologies and the significant growth of wireless network services have made wireless communications a ubiquitous means for transporting information across many different domains. Within the framework of Wireless Sensor Networks (WSNs) [1], there are many potential situations where a WSN can be deployed to support numerous applications. However, current real-life applications of WSNs are very limited. The main reason for the delay in the development of new services is the lack of a complete set of standard mechanisms which can be used to build different application environments [2].

Environmental monitoring is a natural candidate for applying sensor networks [3], since the variables to be monitored (e.g., temperature and soil moisture) are usually distributed over a large spatial region. For example, when the sun is present for at least 12 hours per day (as it happens in the Middle East) and since the environment is quite dry, soil moisture needs to be regularly and accurately measured to allow targeted irrigation techniques to be implemented. This means that wireless sensor networks could be widely deployed to detect when and where it is raining and hence when certain measures could be applied [4].

In order to achieve this using Wireless Sensor Networks, certain challenges need to be addressed. Firstly, we need the system to react correctly to specific events: in this case the presence of rain. However, this can be over all or the part of the region being monitored. Secondly, once the event is detected, the affected nodes need to sample the environment at a much faster rate (i.e. from sampling once every ten

minutes, say, when it is not raining to sampling every few seconds when it is raining).This means that much more bandwidth should be made available to these rain-affected nodes. Hence resources and routing mechanisms must react in this situation to ensure that the required information is quickly and reliably sent to the monitoring centre or base-station. Thirdly, depending on the resources, it might be necessary to trade off these two requirements. So the routing protocol needs to be reactive and robust but in the face of given events, the routing protocol may choose to trade between these two requirements.

The rest of the paper is outlined as follows: Section 2 further explores the problem definition. Section 3 details a solution approach. Section 4 looks at proposed changes to support reactivity by enhancing the Ad-hoc on-Demand Multi Path Distance Vector, (AOMDV) protocol. Section 5 presents the simulation tool, setup, example and result. The paper concludes in Section 6.

2. PROBLEM ANALYSIS

Figure 1 shows the wireless sensor network considered covering a large geographical area. Each node is able to measure the amount of soil moisture in the ground as well as detect the presence of rain. The latter is necessary as the amount of soil moisture may not be enough to detect the presence of rain. In addition, the system must quickly react in order to discover which part of the sensor network is experiencing the event; in this case rain. When rain is detected at any part of the area covered, the data (soil moisture) must be collected from the relevant nodes and sent to a central administrative server or base-station. This means that it will be necessary to change the routing so that information on the change in the soil content in the affected area could be sent to the server as quickly as possible. This data should be routed at a higher priority than other information being sent. The key thing to note is that these changes must be dynamically setup as there will be little foreknowledge involved and so changes in the normal routing protocol are necessary to facilitate this, hence the need for reactivity at the routing level.

Wireless systems are prone to errors associated with the environment in which these systems are deployed. Harsh environments such as very dry conditions can affect their performance and these systems must be carefully tuned to overcome environmental hindrances. However, this also means that a severe change in the environment such as changing from dry to rainy, and sometimes raining very heavily, will likely affect how these wireless nodes operate and may introduce more errors. It will therefore be necessary to boost the reliability of the system to ensure that the required information reaches the server. Furthermore, depending on the amount of rain being experienced, the amount of data moving towards the central server may be substantial and could overwhelm certain links.

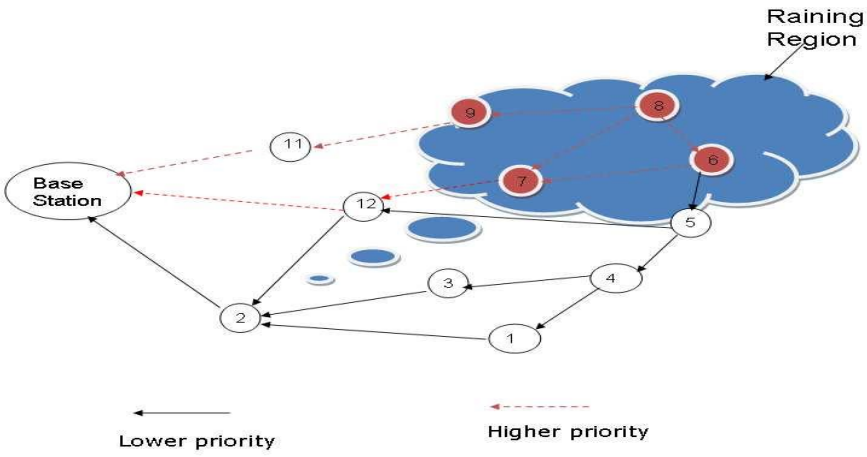

Fig. 1: Sensor Network Measuring Moisture

3. SOLUTION APPROACH

The first requirement is the need to look at developing a reactive routing protocol. This can be achieved by tailoring or enhancing a well-known one. The Ad-hoc on-Demand Multipath Distance Vector (AOMDV) Protocol [5] is a routing protocol designed for use in ad-hoc mobile networks. The AOMDV protocol [5] is an extension to the AODV protocol for computing multiple loop-free and link disjoint paths [6]. The routing entries for each destination contain a list of the next-hops along with the corresponding hop counts. All the next hops have the same sequence number. This helps identify and keep track of a given route. For each destination, a node maintains the advertised hop count, which is defined as the maximum hop count for all the paths, which is used for sending route advertisements of the destination. Each duplicate route advertisement received by a node defines an alternate path to the destination. Loop freedom is assured for a node by accepting alternate paths to destination if it has a lower hop count than the advertised hop count for that destination. Because the maximum hop count is used, the advertised hop count therefore does not change for the same sequence number [6]. When a route advertisement is received for a destination with a greater sequence number, the next-hop list and the advertised hop count are reinitialized.

AOMDV can be used to find node-disjoint or link-disjoint routes. To find node-disjoint routes, each node does not immediately reject duplicate RREQs. Each RREQ arriving via a different neighbour of the source defines a node-disjoint path. This is because nodes cannot broadcast duplicate RREQs, so any two RREQs arriving at an intermediate node via a different neighbour of the source could not have traversed the same node. In an attempt to get multiple link-disjoint routes, the destination replies to duplicate RREQs. The destination only replies to RREQs arriving via unique neighbours. After the first hop, the RREPs follow the reverse paths, which are node disjoint and thus link-disjoint. The trajectories of each RREP may intersect at an intermediate node, but each takes a different reverse path to the source to ensure that the links are disjoint [6]. The advantage of using AOMDV is that it allows intermediate nodes to reply to RREQs, while still selecting disjoint paths. But there is a larger overhead associated with AOMDV. This is because the protocol has more message overheads during route discovery due to increased flooding and since it is a multipath routing protocol in which the destination replies to the multiple RREQs; there is a larger overhead involved in using the protocol.

4. PROPOSED CHANGES TO AODV

The plan is therefore to enhance AOMDV to move towards supporting a dynamic event-driven system as detailed in this paper. The first issue is that the routes may change due to the change in status of the individual nodes, it is necessary to allow HELLO messages to not just monitor links but to fully indicate the status of the node at the other side of the link. So in this case, HELLO messages would also indicate whether or not it is raining on the other end, the network load as well as the power left in the system. A key piece of data is whether or not a node is able to route data on behalf of other nodes. As explained before, if a node is at the centre of a downpour, its data is probably more important than the data of its neighbouring nodes so it should not route data on behalf of these nodes. In such a situation, the node should also not respond to routing request (RREQ) messages. In addition, though HELLO messages will be sent periodically, it is necessary to have another message type which can be sent immediately in response to sudden environmental, link, or node changes. These messages are called STATUS CHANGE messages and are broadcasted to the entire system. For example, when a node first detects rain, it will broadcast a STATUS CHANGE message which is picked up by other nodes. By storing this information from various nodes, it would be possible to detect where the rain is falling in the sensor network.

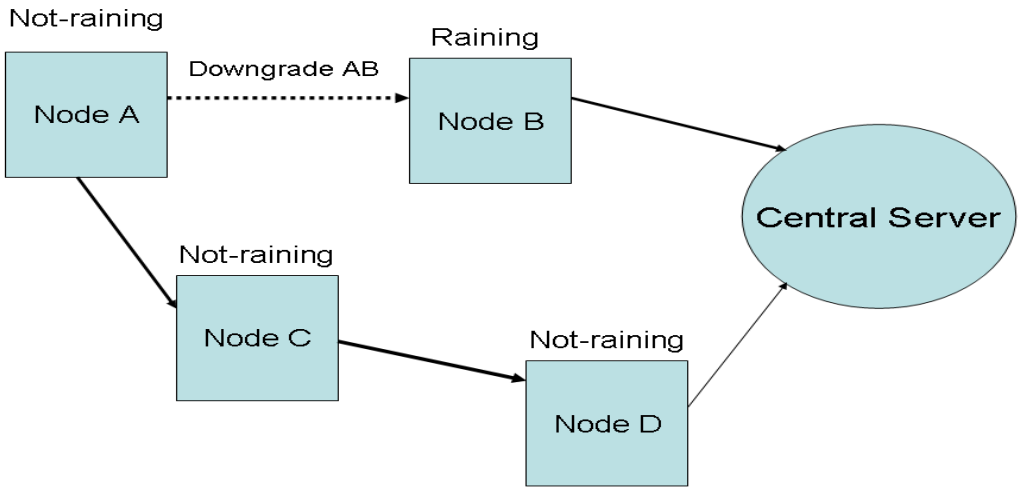

Fig. 2: Routing decisions made based on external events

STATUS CHANGE messages will cause routes to the central server to be re-evaluated. Let's consider the case depicted in Figure 2. If node A that is not affected by rain was using a route to the central server via node B, now node A receives a STATUS CHANGE message from node B saying that it has detected rain. Then node A, which is not rain-affected, should no longer send data through node B if possible, since the data being generated by node B is more important. Node A should look for another route back to the server using other non-affected nodes. This would suggest that routes may have priorities based on the importance of the data being routed relative to the data being generated. In this case, node A will downgrade its route through node B, resulting in other routes through non-affected nodes being favoured. This is illustrated in Figure 2.

If rain is detected by both nodes, A and B, both nodes will send each other their absolute measurements as well as relative changes in the soil moisture content. Nodes with less relative soil moisture content changes which indicate less rain will downgrade routes through regions with high relative soil moisture content changes. So data will be routed away from the most rain-affected areas to the least rain-affected areas.

5. SIMULATIONS

5.1 Network Simulator

Network Simulator (Version 2), widely known as NS2, is an event driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. It consists of two simulation tools. The network simulator (ns) contains all commonly used IP protocols. The network animator (NAM) is use to visualize the simulations. NS2 [7] fully simulates a layered network from the physical radio transmission channel to high-level applications. The simulator was originally developed by the University of California at Berkeley and VINT project. The simulator was recently extended to provide simulation support for ad hoc networks by Carnegie Mellon University (CMU Monarch Project homepage, 1999).

NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl) while the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a frontend). The C++ and the OTcl are linked together using TCL after simulation; NS2 outputs either text-based or animation-based simulation results. To interpret these results graphically and interactively, tools such as NAM (Network AniMator) and XGraph are used.

5.2 AOMDV Modification

As discussed above, any node sensing the rain will get the highest priority to send data, so no node will send data through it. That means this node will be removed from the routing table of its neighbouring nodes. The source code of AOMDV was modified to be an event-driven protocol as follows:

```
node-status is an integer; 1 indicates rain; 0 indicates not-rain;
If node-status = 1 and node not != dest
  {
remove path (node);
print routing path;
 }
else
print routing path;
```

In this code, we see that if the rain-affected node is part of a route to another destination, then that path is removed from the routing table. Note that paths to the rain affected node itself are maintained but the rain-affected node is no longer being used as an intermediate node to send packets to other nodes.

5.3 Simulation Setup

This project consists of seven nodes named 0, 1, 2, 3, 4, 5 and 6 where Node 3 has elected as the destination node. So, the simulation scenario is displaying the routing table of the packets sending from Node 0 to Node 3. Figure 3 shows the logical diagram of the nodes.

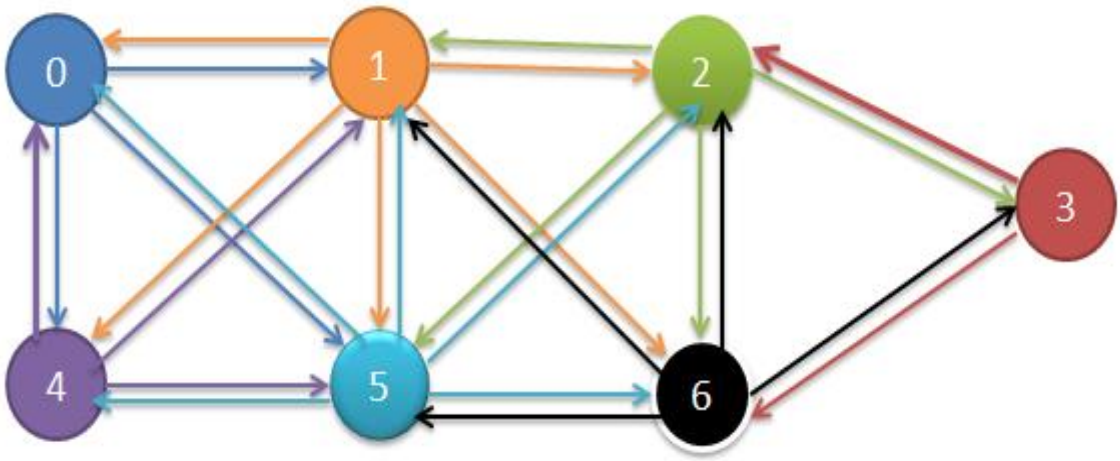

Fig 3: Node Layouts

5.4 *Simulation example*

The source code is written in TCL language and runs the simulation example on ns-2.34 Linux Ubuntu version11 platform.

5.5 Simulation Experiment

As previously mentioned, this scenario uses seven nodes, 0-6, where Node 3 is the destination, so packets will be send from Node 0 through these nodes to the destination, Node 3. This experiment consists of two stages: In the first stage of this experiment, the routing table will be displayed without any changes to the existing AOMDV routing Protocol. So no changes have been done on the routing protocol. In the second stage of this experiment, the routing table will displayed after AOMDV routing protocol has been changed to remove any affected nodes by raining from the routing table. In this example, Node 5 is affected by rain.

5.6 Simulation Result

Below is the result of the simulation without any change to the routing protocol AOMDV

```
num_nodes is set 7
channel.cc: sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5, distCST_ = 550.0
Node 0: Route table Before Dynamic Change:

Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     4       2   16383
                              4       1       0

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     1       2   16383
                              1       1       0

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     5       2   16383
                              5       1       0

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     3       2   16383
                              1       3       6

Node 1: Route table Before Dynamic Change:
  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     4       2   16383
                              4       1       1

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     6       2   16383
                              6       1       1

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     2       2   16383
                              2       1       1

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     5       2   16383
                              5       1       1
```

```
  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     3       2       2
                             6       2       6
                             2       2       2

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     0       4       1
                             0       1       1

Node 2: Route table Before Dynamic Change:
  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     1       2   16383
                             1       1       2

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     6       2   16383
                             6       1       2

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     5       2   16383
                             5       1       2

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     3       2       1
                             3       1       2

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     0       4       2
                             1       2       1
                             5       2       5

Node 3: Route table Before Dynamic Change:
  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     6       2   16383
                             6       1       3

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     2       2   16383
                             2       1       3
  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     0       4   16383
                             6       3       1

Node 4: Route table Before Dynamic Change:
  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     1       2   16383
                             1       1       4

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     5       2   16383
                             5       1       4

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     0       4       1
                             0       1       4
```

```
Node 5: Route table Before Dynamic Change:
  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     4       2   16383
                              4       1       5

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     1       2   16383
                              1       1       5

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     6       2   16383
                              6       1       5

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     2       2   16383
                              2       1       5

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     0       4        1
                              0       1       5

Node 6: Route table Before Dynamic Change:
  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     1       2   16383
                              1       1       6

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     2       2   16383
                              2       1       6

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     5       2   16383
                              5       1       6
  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     3       2        1
                              3       1       6
  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     0       4        2
                              1       2       1
                              5       2       5
```

Assume that the Node 5 is now affected by rain, and then other nodes will send data through Node 1 only as seen with Node 2 going to destination Node 0 and Node 6 going to destination Node 0. The complete new routing table is given below:

```
Node 0: Route table After Dynamic Change:
  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     4       2   16383
                              4       1       0

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     1       2   16383
                              1       1       0

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     5       2   16383
                              5       1       0
```

```
  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     3       2   16383
                              1       3       6

Node 1: Route table After Dynamic Change:
  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     4       2   16383
                              4       1       1

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     6       2   16383
                              6       1       1

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     2       2   16383
                              2       1       1

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     5       2   16383
                              5       1       1

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     3       2        2
                              6       2       6
                              2       2       2

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     0       4        1
                              0       1       1

Node 2: Route table After Dynamic Change:
  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     1       2   16383
                              1       1       2
  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     6       2   16383
                              6       1       2
  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     5       2   16383
                              5       1       2

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     3       2        1
                              3       1       2

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     0       4        2
                              1       2       1

Node 3: Route table After Dynamic Change:
  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     6       2   16383
                              6       1       3

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     2       2   16383       2       1       3
```

```
  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     0       4   16383
                              6       3       1

Node 4: Route table After Dynamic Change:
  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     1       2   16383
                              1       1       4

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     5       2   16383
                              5       1       4

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     0       4        1
                              0       1       4

Node 5: Route table After Dynamic Change:
  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     4       2   16383
                              4       1       5

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     1       2   16383
                              1       1       5

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     6       2   16383
                              6       1       5

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     2       2   16383
                              2       1       5

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     0       4        1
                              0       1       5

Node 6: Route table After Dynamic Change:
  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     1       2   16383
                              1       1       6

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     2       2   16383
                              2       1       6

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     5       2   16383
                              5       1       6

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     3       2        1
                              3       1       6

  Dest    Seq#  Advhop  Nxthop  Hopcnt  Lsthop
     0       4        2       1       2       1
```

6. CONCLUSIONS

In this paper, the design and development of a dynamic event-based routing algorithm is presented. The outlined design consists of developing support for reactivity using an enhanced version of AOMDV. The NS2 simulator, and C++, TCL languages were used to develop the new enhanced protocol. The routing table of the protocol was displayed before and after the change was applied. The next steps of this project will be to show how the reactivity will be implemented using RREQ and RREP messages. Also changes will be introduced to AOMDV to improve reliability.

7. REFERENCES

[1] Culler, D., Estrin, D., Srivasta, M. (2004) Overview of Sensor Networks, IN *Computer*, vol. 37, pp. 41–49, August 2004.

[2] Steere, A., Baptista, A., McNamee, D., Pu, C., Walpole, J., (2004) Research challenges in environmental observation and forecasting systems, IN *Proceedings of the 6th International Conference on Mobile Computing and Networking (MOBICOM)*, 2004, pp. 292–299.

[3] Martinez, K., Hart, J., Ong, R., (2004) Environmental Sensor Networks IN *Computer*, vol. 37, pp. 50–56, August 2004.

[4] Cardell-Oliver, R., Smetten, K., Kranz, M., Mayer, K., (2004) Field testing a wireless sensor network for reactive environmental moisture measurement IN *Proceedings of the Conference on Intelligent Sensors, Sensor Networks and Information Processing*, Dec 2004, pp. 7–12.

[5] Trung, H.D., Benjapolakul, W., Duc, P.M., (2007) Performance evaluation and comparison of different ad hoc routing protocols‖, Department of Electrical Engineering, Chulalongkorn University, Bangkok, Thailand, May 2007

[6] Haas, Z.J., Pearlman, M.R., (2001) The Performance of Query Control Schemes for the Zone Routing Protocol, ‖ IN *ACM/IEEE Trans. Net. 9* (August 2001)

[7] Srikanth, D., Krishnamurthy, V., (NA) Ad hoc networks technologies and protocol by Prasantmohapatra University of California, University of California, Riverside

[9] Wischik, D., Handley, M., Braun, M.B., (2008) The resource pooling principle IN *ACM Computer Communications Review*, Vol. 38, no. 5, pp. 47–52, 2008

[10] Report, O (NA) TinyOS: a Component-Based OS for the Networked Sensor Regime," Available Online at: http://web.cs.berkely.edu/tos Accessed on 03/04/13

[11] Levis, P., Culler, D., (2002) Mate: a Tiny Virtual Machine for Sensor Networks IN *Proceedings of the 6th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOSX)*, October 2002.