Routing with Dijkstra in Mobile Ad-Hoc Networks

Khudaydad Mahmoodi, Muhammet Balcılar, M. Fatih Amasyalı, Sırma Yavuz, Yücel Uzun, and Feruz Davletov

Yıldız Technical University, Computer Science Department, Istanbul
{khudaydad.mahmoodi,fdavletov}@gmail.com,
{muhammet,mfatih,sirma}@ce.yildiz.edu.tr,
yuceluzun@windowslive.com

Abstract. It is important that robot teams have an effective communication infrastructure, especially for robots making rescue operations in debris areas. The robots making rescue operation in a large area of disaster are not always directly connected with central operator. In such large areas robots can move around without losing communication with each other only by passing messages from one to another up to the central operator. Routing methods determine from which node to which node the messages are conveyed. In this work blind flooding and table-based routing methods are tested for three different scenarios to measure their effectiveness using the simulation environment USARSIM and its wireless simulation server WSS. Message delay times and maximum data packet streaming rates are considered for measuring the effectiveness. Although it has some deficiencies, it was observed that table-based approach is more advantageous than blind flooding.

Keywords: Mobile ad-hoc networks, Routing protocols, USARSim, WSS.

1 Introduction

In recent years, the importance of the teleoperation of mobile robots and teams of mobile robots increased. Recently, more and more mobile robots are developed which are capable of operating in impassable or hazardous environments with little or no communication infrastructure [2]. Along with technological advances robots became much more intelligent and much more capable. It means that they must be developed to possess the capability of constructing a network and performing cooperative works [1].

A key driving force in the development of cooperative mobile robotic systems are their potential for reducing the need for human presence in dangerous applications. Such applications as the disposal of toxic waste, nuclear power processing, firefighting, civilian search and rescue missions, planetary exploration, security, surveillance and reconnaissance tasks have elements of danger. In these cases, wireless communication provides the low-cost for mobile robot networks to cooperate efficiently [1].

There is increasing demand for connectivity in places where there is no base station or infrastructure available. This is where ad-hoc networks came into existence.

S. Behnke et al. (Eds.): RoboCup 2013, LNAI 8371, pp. 316–325, 2014.

Wireless networks can be classified into infrastructure networks and infrastructure-less networks or mobile ad-hoc networks (MANETs) [3].

MANETs are autonomously self-organized and self-configuring networks without infrastructure support. To create a temporary network there is no need for any centralized administration or infrastructure. In such networks, due to the absence of dedicated routers, each member node is also responsible for routing messages to other nodes. If mobility is very high, then the network may experience frequent and unpredictable topology changes [3], [5], [8].

Recently, mobile ad-hoc networks became a hot research topic among researchers due to their flexibility and independence of network infrastructures such as base stations. The infrastructure-less and the dynamic nature of these networks demand a new set of networking strategies to be implemented in order to provide efficient end-to-end communication. MANETs can be deployed quickly at a very low-cost and can be easily managed [3].

There have been a number of ad-hoc routing protocols developed for MANETs, each with benefits relating to specific usage scenarios. The majority of routing protocols for MANET try to reduce bandwidth usage, minimum energy consumption, throughput, packet delay time, etc. Different routing protocols use different measures to determine the optimal route between the sender and receiver. Each protocol has its own advantages and disadvantages. In this application, we try to reduce packet delay time by minimizing hop count for better and faster communication between robots.

In this study, we implement Dijkstra's algorithm with minimum hop in order to minimize packet transmission time and tested our method on USARSim simulation software. This paper proceeds as follows. Section 2 shows the classification of mobile ad-hoc routing protocols. USARSim and WSS are briefly introduced in section 3. Section 4 explains implemented the Dijkstra's algorithm. Experimental results of algorithms are analyzed in section 5. Finally, Section 6 concludes the paper.

2 Routing Protocols

Routing protocols in MANETs can be classified into two categories based on routing strategies and network structure [5]:

1. Proactive Protocols (Table-Driven)
2. Reactive Protocols (On-Demand)

2.1 Proactive Routing Protocols

Proactive or table-driven routing protocols maintain the routing information even before it is needed. Each and every node in the network maintains routing information to every other node in the network. Routes information is generally kept in the routing tables and is periodically updated as the network topology changes. The proactive protocols are not suitable for larger networks, as they need to maintain node entries for each and every node in the routing table of every node. This causes more overhead in the routing table leading to consumption of more bandwidth [6], [7].

2.2 Reactive Routing Protocols

Reactive or on-demand routing protocols create routes only when required by a node. If a node requires a path to a destination, it starts a route discovery process in the network. It can be either source initiated or destination-initiated. Once a route has been established, the route discovery process ends and the route will be valid until it breaks down or is no longer desired [4]. Packets are sent through this route. If there is no communication between two nodes then it is not necessary to maintain routing information at these nodes [3]. Such protocols often perform better than proactive protocols when implemented in a large network due to a smaller overhead. However a large amount of network traffic can cause the performance to deteriorate sharply as most such protocols flood the network while looking for a route, and this can lead to clogging of links. Another major disadvantage is the delay required to find a route which in some applications might be unacceptable [5], [7].

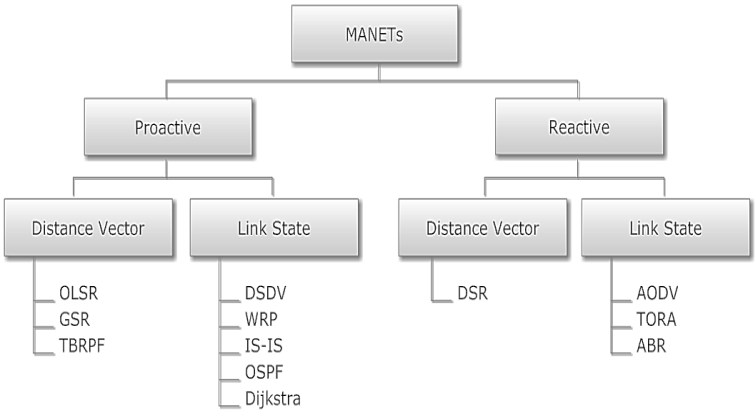

Fig. 1. Classification of routing protocols in MANETs

3 USARSim and WSS

The Urban Search and Rescue Simulator (USARSim) is built on top of the Unreal Engine, and uses the Unreal Engine to simulate the environment and the robots [5]. It is designed as a simulation companion to the National Institute of Standards (NIST) reference test facility for autonomous mobile robots for urban search and rescue [9].

Creating robots and preparing test environments are very costly and difficult tasks in real world. USARSim helped us to create robots easily in desired positions and prepare test environment with minimal cost. Maps in RoboCup competition contain specific disaster environment, victims, routes with different types of obstacles, etc. Robots have some certain missions due to competition rules. The main goal of robots are accomplishing mission as a desired manner in a given disaster environment.

The Wireless Simulation Server (WSS) is developed by Jacobs University which is used to simulate wireless communications between all robots and the operator that are

created in USARSim simulation program. According to RoboCup virtual robot competitions rules, all communications between robots should be through WSS software. WSS allows us to get the signal strength of any pairs of robots whenever it is required. In order to know status of link between two nodes we have to compare the signal strength value with predefined threshold value. If the signal strength value is equal or bigger than threshold value (-93 dBm) then connection available, otherwise connection between nodes breaks down [10].

Registering is required to a specific port of WSS before starting communication. Once the connection has been opened, the WSS allows sending of messages and closing the connection. Each time a message is sent, the path loss between the end points is checked. If it is better than the threshold then the message is forwarded, otherwise message will be discarded and connection will be closed [5].

4 Routing with Blind Flooding Algorithm

The blind flooding approach is most widely employed strategy to perform and distribute messages to all robots in the networks [11]. Implementation of this algorithm is very simple. Because of exploring every possible nodes in the network, it increases the possibility of delivering packet to destination. On the other hand, flooding algorithm has some drawbacks like implosion, overlap and resource blindness. For instance, many unneeded packets on the network leads to use more bandwidth. Flooding exhibits a desirable behavior when adopted in wired networks.

Algorithm: Routing with Blind Flooding Algorithm

```
Inputs: r_dest, r_cur, m, S_i,j, th
when packet m is received do
    if r_dest = r_cur
        Take packet and process it
    else
        if S_rcur,rdest ≥ th
            Send the packet m directly to r_dest
        else
            Send the packet to all neighbors of r_cur except
            itself and the robot which packet came from.
        endif
    endif
endDo
```

5 Routing with Dijkstra Algorithm

Dijkstra's algorithm was created in 1959 by Dutch computer scientist Edsger Dijkstra. Dijkstra's Algorithm is a graph search algorithm that solves the single-source shortest path problem for a graph with non-negative edge path costs, producing a shortest path tree. This algorithm is often used in routing and other network related protocols [12].

Dijkstra's algorithm finds the shortest path from a given starting node to all of the other nodes in the graph. It requires that the weights of all edges are non-negative. It operates by maintaining a set of visited nodes and continually updating the tentative distance to all of the unvisited nodes. At each iteration, the closest unvisited node is added to the visited set and the distances to its unvisited neighbors are updated.

Algorithm: Finding optimum route from comstation to other robots

Inputs: $N = \{1,2,..\#Robots\}$, $S_{i,j}$ $i = 1..\#Robots$, $j = 1..\#Robots$
Output: $Path(n)$ $n = \{2..\#Robots\}$
$R = \{1\}$ // Currently only robot1 (comstation) can be reach from robot1
Calculate $Cost(i,j)$ using with equation (1)
for each robot n **in** $N - \{1\}$ **do** //initialization step
    $TCost(n) = Cost(1,n)$
    $Path(n) = \{n\}$
endfor
while $R \neq N$ **do** // actual algorithm steps
    Find the robot m in $\{N - R\}$ that has minimum value for $TCost(m)$
    $R = R \cup \{m\}$
    for each robot n **in** $\{N - R\}$ **do**
        $TCost(1,n) = min(TCost(n), TCost(m) + Cost(m,n))$
        if $TCost(m) + Cost(m,n) < TCost(n)$
        $Path(n) = Path(m) \cup n$
        endif
    endfor
endwhile

Dijkstra's algorithm works for graphs with non-negative edges, but in our application robot signal strengths take zero or negative values. According to these values, if the signal strength is between 0 and -93, then the robot is in the coverage area. If the signal strength is below -93, then the robot is out of coverage area. In order to apply Dijkstra's algorithm in our application we normalized the robot signal strengths as in equation (1).

$$Cost(i,j) = \begin{cases} -93 \leq S_{i,j} \leq 0 & 1 - \left(S_{i,j}/1000\right) \\ S_{i,j} < -93 & \infty \end{cases} \quad (1)$$

In the above mathematical equation, $S_{i,j}$ is used to represent the signal strength value between *i*th robot and *j*th robot. $Cost(i,j)$ is used to represent link cost value between *i*th robot and *j*th robot which is used by the algorithm.

Let's consider $\#Robots$ is number of total robots, R is set of reachable robots, $Path(n)$ is routing path between comstation and *n*th robot. $TCost(n)$ is total cost of routing path between comstation and *n*th robot. According to these definitions, the steps in the algorithm are as follows [13].

In our application optimum route calculation is performed once by comstation in every 5 second. Then comstation sends updated path information to all robots periodically. Robots are using these paths for sending messages. Table-based algorithm which is used by robots in order to send messages in the network is as follow.

Algorithm: Routing with Table-Based Approach

Inputs: $r_{dest}, r_{cur}, m, Path$
when packet m is received **do**
    if $r_{dest} = r_{cur}$
        Take packet and process it
    else
        Get $Path$ from packet m
        Get next robot node from $Path$
        Forward the packet m to the next robot node
    endif
endDo

6 Experimental Results

In this section, implemented routing algorithms are tested on two different propagation models of WSS. For all tests USARSIM and WSS software run on a PC that used as a server, test code ran on another PC that was used as a client. Client connected to the server through a router which has maximum 100Mps link speed.

6.1 Noop Propagation Model

For an increasing number of robots, the average message delays are tested for the situation that all the robots are directly connected with comstation, before testing the rates the routing algorithms transmitted messages to destination. By doing this the average message delays could be seen for different number of robots in the most ideal situation, independent of routing approaches. For test scenarios, except from comstation respectively 1 robot and 2, 4, 8, 15 robots are created. All the robots are keep in touch with comstation. Each robot continuously sends a 2048 byte message to

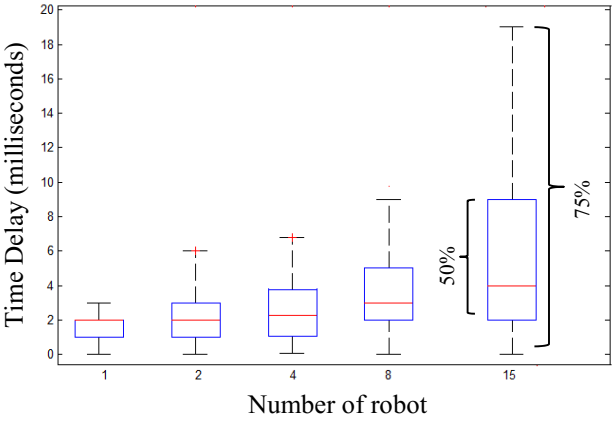

Fig. 2. Boxplot of message delay versus robots number in Noop Propagation Model

the comstation and once it sends a message, robot logs its id along with sending time. In addition, comstation logs the id of the message and its receiving time. Test continues till each robot sends 20000 messages to the comstation.

Fig.2 shows the boxplot diagram indicating the densities of the distribution of message delay times for varying the number of robots between 50% and 75% confidence interval, according to test results. When the test results are analyzed it can be said that in the situation that no routing method is needed, mainly when the Noop Propagation model is used, message delays are increased linearly with the number of robots. However, even with 15 robots no bad condition occurs while controlling the robots because the message delays have 11.04 average and 22.4 ms standard deviation at this situation.

6.2 Distance and Obstacle Propagation Model

In the Distance and Obstacle Propagation Model, robots are not in communication if their signal strength value is smaller than the threshold value. In this case, the robots' sensor information and comstation's controls commands are sent via other robots. Route selection process is very important for sending message packets to destination. In Blind Flooding method, each node sends the message packet received from other nodes to its neighbor. This method is known as baseline method in the literature. It is compared with table-based method which finds shortest path between robots dynamically and sends it to other robots periodically. At runtime, robots can be anywhere on the map and robots communication graph changes dynamically. It's obvious that whenever the graph structure changes, test results changes as well. In order to do a fair test 4, 8 and 15 robots are created except comstation and positioned on map as Fig.3. Graphs in Fig.3 are bidirectional and colored nodes represent the comstation. This test is done by calculating the number of packets sent by robots to the comstation per unit of time and packet delay. During the test comstation does not send any message to robots in blind flooding method. On the other hand, the table-based method, the comstation sends dynamically calculated route information to robots at specified intervals.

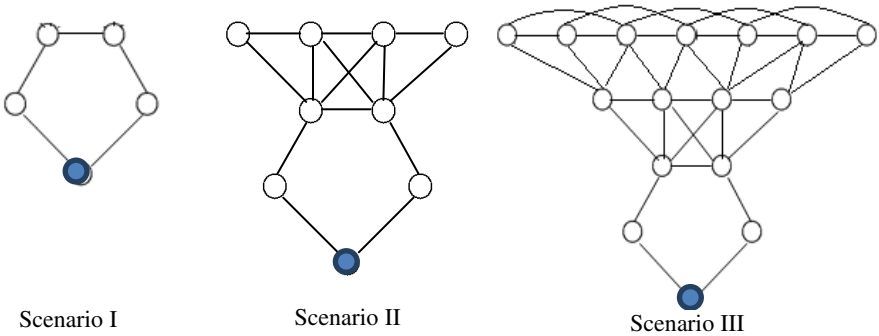

Fig. 3. Test scenarios

In a disaster area, researcher robot teams must send sensor information to the central operator as fast as possible. In this test optimum data transmission (packet number) is researched to prevent systematic network delay and packet losses caused by buffer overflows for both routing algorithm. Different delay time is set between sequential message packets to examine buffers in the network. For each of the three scenarios, packet success rates versus data transmission rates obtained with blind flooding algorithm are shown in Fig.4.

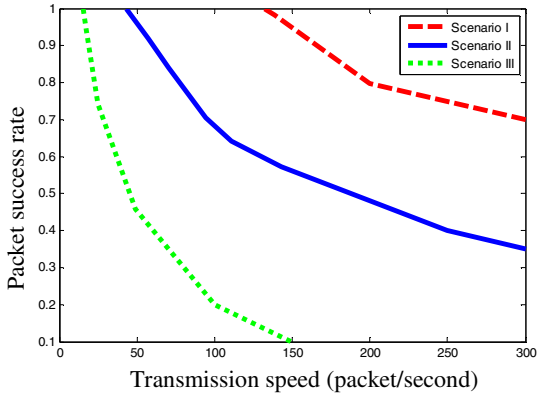

Fig. 4. Packet success rate versus packet/second rate for blind flooding algorithm

For all three scenarios even with maximum packet rates the system allows, any systematic delay or packet loss does not occur in the table-based routing method. Each tests carried on until each robot sends 5000 messages to the comstation. Total packet rate and packet rate per robot is shown as below.

Table 1. Results of maximum packet transmission speed

Scenario	Method	Total Packet Speed (pck/sec)	Packet Speed for each robot (pck/sec)
I	BF	132.45	33.11
I	TB	398.32*	99.58
II	BF	43.52	5.44
II	TB	312.76*	39.09
III	BF	15.16	1.01
III	TB	229.11*	15.27

In the table above, (*) represents the maximum packet rate allowed by the system. When table-based method is used neither packet losses nor systematic delay is observed. Therefore, rates with (*) signs are not actual rates; these rates may increase if faster data generated. The high number of unnecessary internal messaging in the network causes unnecessary bandwidth usage in Blind Flooding method. When Blind Flooding is used for routing, to avoid network buffer overflow each robot must

generate no more than 1.01 packets/sec. This number is approximately 15 times greater in Table-Based routing.

In order to measure the delays of packets sent from robots to the comstation for each scenario, it's ensured that robots generate packets at rates obtained from previous analysis. Each test is carried on until each robot sends 5000 messages to the comstation and the delay of each message packet is calculated.

Table 2. Results of packet delay

Scenario	Method	Worst Robot mean delay (ms)	Best Robot mean Delay(ms)	Mean Delay (ms)	Standard Deviation of delay (ms)
I	BF	29.75	10.03	18.46	16.80
I	TB	10.10	5.27	7.67	6.58
II	BF	91.39	11.07	52.41	42.45
II	TB	20.56	7.27	15.32	10.46
III	BF	361.87	15.25	218.74	155.77
III	TB	42.10	13.16	30.40	17.36

Test results are shown in Table 2. According to the table it's seen that blind flooding (BF) gives similar results to table-based routing (TB) for lower number of nodes. But for bigger network structures (the more connection between nodes) the delay of BF method is increased exponentially. However for TB method this increase occurs linearly. The histogram of packet delays occurred with blind flooding and table-based method for Scenario III is as in Fig.5.

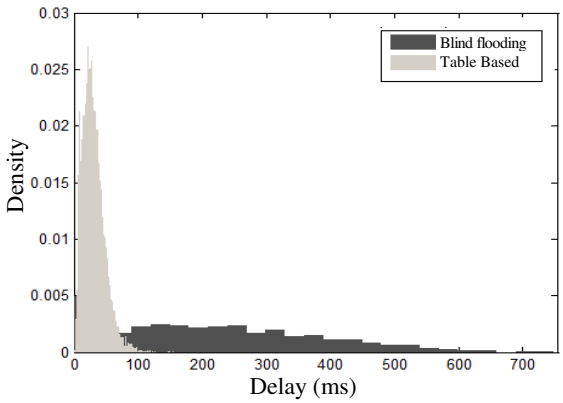

Fig. 5. Histogram of packet delay in scenario III using with Blind flooding and Table-based method

7 Conclusion

Ad hoc mobile robot communications are a promising networking technology for multi-robot communication in debris areas. Yet ad hoc robot communication repre-

sents a relatively underdeveloped application field. Blind Flooding approach can be preferred at situations with relatively few nodes, due to the simplicity of its application, its ensuring packets to reach target in any architecture (if there is no overflow in buffer). One of the advantage of this approach is does not need extra configuration messages. However, it must be cautioned that message delay increases with the square of number of connections and the slowdown in packet transmission speed. Message delay increase linearly with the number of connections between nodes and high packet transmission speeds are important advantages of table-based routing. However, as a requirement of its method, the messages including dynamic routing tables having to be sent to other nodes periodically consume extra bandwidth. When the frequency of these messages are decreased, the packet loss is increase because the robots find out routing table late or even does not. A good ad-hoc network routing algorithm can be developed with a table based approach by balancing this value in real or simulation debris areas.

References

1. Wang, Z., Zhou, M., Ansari, N.: Ad-hoc robot wireless communication. In: IEEE International Conference on Systems, Man and Cybernetics 2003, vol. 4. IEEE (2003)
2. Zeiger, F., Kraemer, N., Schilling, K.: Commanding mobile robots via wireless ad-hoc networks—A comparison of four ad-hoc routing protocol implementations. In: IEEE International Conference on Robotics and Automation, ICRA 2008. IEEE (2008)
3. Wahi, C., Sonbhadra, S.K.: Mobile Ad Hoc Network Routing Protocols: A Comparative Study. International Journal of Ad Hoc, Sensor & Ubiquitous Computing 3(2) (2012)
4. Wang, Z., Liu, L., Zhou, M.: Protocols and applications of ad-hoc robot wireless communication networks: An overview. Future 10, 20 (2005)
5. Nevatia, Y.: Ad-Hoc Routing for USARSim (2007)
6. Gorantala, K.: Routing protocols in mobile ad-hoc Networks. Master's Thesis in Computing Science (June 15, 2006)
7. de Morais Cordeiro, C., Agrawal, D.P.: Mobile ad hoc networking. Center for Distributed and Mobile Computing, ECECS. University of Cincinnati (2002)
8. Shrivastava, A., et al.: Overview of Routing Protocols in MANET's and Enhancements in Reactive Protocols (2005)
9. Carpin, S., et al.: USARSim: a robot simulator for research and education. In: 2007 IEEE International Conference on Robotics and Automation. IEEE (2007)
10. Pfingsthorn, M.: RoboCup Rescue Virtual Robots: Wireless Simulation Server Documentation, pp. 1–8 (October 2008)
11. Giudici, F.: Broadcasting in Opportunistic Networks, Universita Degli Studi di Milano, thesis (2008)
12. Dijkstra, E.: Dijkstra's algorithm. Dutch scientist Dr. Edsger Dijkstra network algorithm (1959), http://en.wikipedia.org/wiki/Dijkstra's_algorithm
13. Dijkstra's Algorithm (2013), http://www.mathcs.emory.edu/~cheung/Courses/455/Syllabus/5a-routing/dijkstra.html (accessed May 21, 2013)