

RAS: A Reliable Routing Protocol for Wireless Ad Hoc and Sensor Networks ^{*}

Imad Jawhar, Zouheir Trabelsi, and Jameela Al-Jaroodi

College of Information Technology
United Arab Emirates University
P.O. Box 17551, Al Ain, UAE
E-mail: {ijawhar, trabelsi.j.aljaroodi}@uaeu.ac.ae

Abstract. Wireless sensor and ad hoc networks are gaining a lot of attention in research lately due to their importance in enabling mobile wireless nodes to communicate without any predetermined infrastructure. Routing protocol in wireless sensor and ad hoc networks discover a multi-hop route between source and destination nodes. This paper presents RAS: a Reliable routing protocol for wireless Ad hoc and Sensor networks. In the RAS protocol, increased reliability is achieved by the maintenance of a reliability factor by the nodes. The value of this factor is increased when nodes participate successfully in data transmissions. This is determined through the use of positive and passive acknowledgements. During the path discovery process, an intermediate node only extends the request message to nodes that have a minimal reliability factor which is specified by the source. Additional optimizations are included in order to increase the efficiency and performance of the network.

Keywords: Mobile ad hoc networks (MANETs), wireless sensor networks (WSNs), reliability, quality of service (QoS), routing.

1 Introduction

The lack of a fixed topology and central control in mobile ad hoc and sensor networks poses a great challenge to the routing process in this environment. Particularly, when the issue of trust is in question. Routing protocols designed for ad hoc networks such as the Dynamic Source Routing protocol (DSR) [16], Ad hoc On-Demand Distance Vector (AODV) protocol [17], Temporally Ordered Routing Algorithm (TORA) [14], and many others [2][4][6][7][8][9][13][19][21][22] work very well under certain conditions where nodes are trusted, they all behave correctly and there are no intruders or malicious attacks on the network. However, we run into problems when we consider the reality that not all nodes will be cooperative and there are no guarantees that any of the nodes will be malicious. Routing protocols were investigated and modified and new protocols were introduced to enhance the routing process in MANETs. Many of these protocols provide some solution to parts of the problem.

To start, in [20] the authors argue that TCP is not suitable for ad hoc networks and propose a new transport layer routing protocol ATP. This enforces our approach to

^{*} This work was supported in part by UAEU Research grant 08-03-9-11/07.

providing routing at a higher level and allowing the applications to take control of the process. Furthermore, the utilization of middleware to provide this type of functionality is another viable approach. The study in [5] provides an overview of possible middleware schemes and how some provide feasible solution to the ad hoc routing issues. One way to increase reliability is by using regenerating nodes [11]. This approach to ad hoc routing increases the reliability of packet delivery by allowing intermediate nodes on the routing path to reconstruct packets and send them to the destination in case of a problem. This reduces the traffic between the source and destination and speeds up the recovery of lost packets. However, this scheme assumes regenerating nodes are always available and willing to do the job and does not account for selfish nodes. Yet in many cases those nodes may not be able to do that because of their mobility, low resources, high traffic or plainly because they are malicious. Another approach is using a distributed multi-path DSR protocol for MANETs to improve QoS support for end-to-end reliability [10]. The protocol forwards outgoing packets along multiple paths based on specified end-to-end reliability requirements. As a result the packet will have better chances of arriving at the destination; however, this introduces high traffic volumes on the links due to the duplicate packets. An enhancement to DSR called DSR with Connection-Aware Link-state Exchange aNd DiffERentiation is introduced [3] to effectively collect and disseminate neighbor link states to nodes which may potentially use them. Neighbors exchange link-state information as soon as a connection is established using piggy-backed messages. This information speeds-up route discovery in case of link failures. In addition fidelity is used to assess the cost of a link, so when a new/alternate route has to be computed, the level of fidelity is figured into the link cost. In another approach, a reputation-based system built on the use of state model [1] is introduced to detect selfish nodes and encourage them to be cooperative by providing benefits. The techniques also handle suspicious nodes that may behave selfishly and encourage them not to. While the authors in [12] combine the reputation-based technique and the virtual currency to enforce cooperation between nodes. The combined approach takes advantage of the fairness of the virtual currency mechanism, while maintaining high cooperation levels based on the reputation approach. Another issue being investigated is trust. The authors in [18] introduce a mechanism to establish trust between nodes based on their reputation. During the network lifetime, nodes gain reputation value as they behave well and that allows other nodes to assess the reputation of their neighbors and the nodes to be selected for the routes needed. This approach is mainly useful to detect nodes that maliciously drop packets and allow nodes to avoid these malicious nodes. Furthermore, the authors in [15] provide mechanisms to detect attacks on the ad hoc network from selfish nodes and introduce a low-cost scheme to inform others about the accusations. In addition, the approach provides an inference scheme to back the accusations and allow nodes to make informed decisions on how to deal with selfish nodes. Many of the approaches discussed here and in other research articles target one main goal which is enhancing the routing process and maintaining reliable communication over multi-hop ad hoc networks. Our approach provides reliable routing based on a reliability factor established and maintained by the nodes in the network. The reliability factor is maintained by the nodes and is updated based on the successful participation of nodes in previous data transmissions. Our protocol is based on the

DSR routing protocol, and is on-demand. In addition, it is a distributed protocol. Unlike other protocols such as the link-state-based ones, each node only has to maintain topology and reliability information about its immediate neighbors and not the entire network. These characteristics enhance the scalability and performance of our proposed protocol.

The remainder of the paper is organized as follows. Section 2 presents the RAS routing protocol, along with the associated data structures, messages, and algorithms. Simplified and detailed examples are also provided. Section 3 discusses additional security issues and related future work in ad hoc and sensor networks. Finally, the last section concludes the paper.

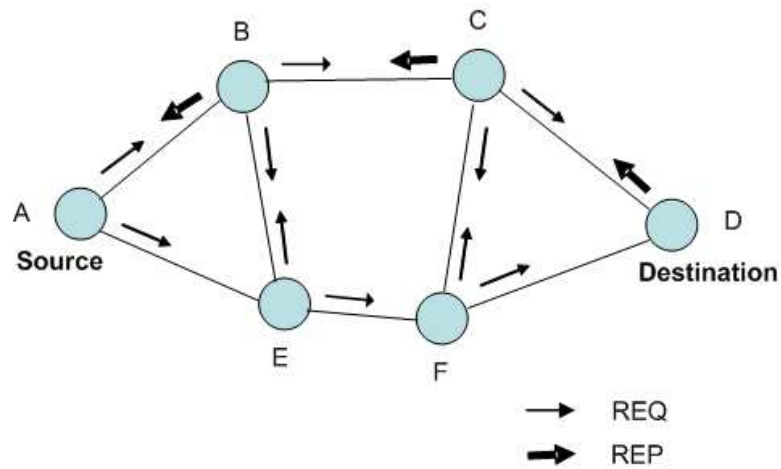


Fig. 1. A simplified example of the route discovery process.

2 The RAS Routing Protocol

This section describes the RAS routing protocol. It starts with a brief description of the DSR-based routing process along with a simplified example. Next, the RAS routing protocol data structures, messages, parameters, and algorithms are described along with a detailed example that illustrates the routing process. The maintenance of the reliability factor and data transmission process are described later.

2.1 A simplified overview of the DSR-based routing process

The protocol that is presented in this paper is based on the Dynamic Source Routing (DSR) protocol. It is on-demand which makes it more scalable. Nodes do not need to keep information about the entire topology and routes are only discovered as the need arises. When a source node s wants to send data to another destination node d which is

not within its transmission range. It will try to discover a multi-hop path to it. In order to do that, node s broadcasts a request (REQ) message to all of its neighbors. Each of the neighbors adds its ID to the accumulating path in the message and in turn forwards it to all of its neighbors. This process continues until the REQ message reaches the destination, which then unicasts a reply (REP) message back to the source. Upon receiving the REP message the source updates its routing table and starts the data transmission process.

A simplified example of the route discovery process is shown in Figure 1. In the figure, node A is a source node that needs to send data to a destination node D. Node A checks its routing table and realizes that it does not have a path to D. Consequently, node A starts a route discovery process by broadcasting a REQ message to its neighbors B and E. Nodes B and E, not having processed a REQ with the (s, d, ID) tuple, check the information in the REQ message and realize they are not the destination and that they do not have a path to D. Each of them forwards the REQ message to its neighbors except for the node from which they received the request (i.e. A). This process continues until the REQ message reaches the destination D. Node D then unicasts an REP message with the discovered path $A - B - C - D$ to A. Upon receiving the REP message, node A updates its routing table and starts the data transmission process along the discovered path.

2.2 The request message and associated data structures

In our protocol, the source sends the request message REQ ($s, d, id, x, r_{min}, r_{max}, r_{cum}, PATH, NH,$) which contains the following fields:

1. s : ID of the source node.
2. d : ID of the destination node.
3. ID : Message ID. The (s, d, ID) triple is therefore unique for every REQ message and is used to prevent looping.
4. x : The node ID of the host that is forwarding this REQ message.
5. r_{min} : The minimum value for the reliability factor required in the path from s to d .
6. r_{cum} : The cumulative reliability factor the the path that is being discovered.
7. $PATH$: It contains the accumulated list of hosts, that the REQ message has passed through.
8. NH : It contains the next hop information. If node x is forwarding this REQ message, then NH contains a list of the next hop host candidates that satisfy the reliability requirements for the path that is being discovered.

2.3 Application specific parameters

In addition to the parameters specified above, the application layer is able to specify certain parameters that are used in the route discovery and communication process. These parameters depend on quality of service (QoS) requirements of the application, and are the following:

- *MAX_NH*: Maximum number of nodes in the NH list. This parameter is intended to control the flooding of the REQ message during route discovery. If this number is infinity, then the process reverts to normal flooding. Otherwise, if this number is set to k , then only a maximum of k neighbors that satisfy the reliability requirements can be selected as next hop for the REQ message.
- *BACKUP_PATHS*: This is the maximum number of backup paths that can be included in the routing table of the source node. If this number is 0, then only the most reliable path is selected by the source and stored in its routing table. Otherwise, if this number is more than 0, then the primary path along with up to *BACKUP_PATHS* discovered paths are stored in the routing table. The additional paths can be used as backup in case of failure of the primary path. The backup paths are stored in sorted order based on their path reliability factor.

2.4 An overview of RAS routing process

When an intermediate node y receives the REQ message from a node x , it checks if it already processed this message which is uniquely specified by the (s, d, ID) tuple. If it already processed this message, it drops it. This prevents looping. Otherwise, it checks if it is the destination indicated in the message. If it is not, it checks its routing table for a path to the destination with the required minimum reliability factor. If such a path exists, it appends it to the PATH list and unicasts a reply REP(s, d, ID, x, PATH) message back to the destination. If a path to the destination does not exist in its routing table, it appends its ID to the PATH list in the REQ message and broadcasts the message to all of its neighbors excluding x . The PATH list in the message is an accumulated list of nodes that the REQ message has propagated through. This process continues until the REQ message arrives at the destination node d . At that time, d unicasts a REP message back to the source along the discovered path saved in the PATH list. When the source receives the REP message, it updates its routing table with this information and starts the data transmission process.

2.5 The algorithm at an intermediate node

Algorithm 1 presents the algorithm at an intermediate node in the RAS protocol. When an intermediate node y receives a REQ message from a node x , it sorts all of its 1-hop neighbors in descending order using their reliability factors $rf[z]$, where $z = 1..n_y$ are the 1-hop neighbors of y , and n_y is their total number. Then, the node builds the next hop list (NH) which includes a maximum of *NH_MAX* 1-hop neighbors of y with reliability factors higher than or equal to the minimum acceptable reliability factor, r_{min} . Afterwards, node y 's reliability factor, r_y is added to the cumulative path reliability factor r_{cum} which is included in the REQ message. Finally the REQ message is broadcast to all of y 's 1-hop neighbors.

Upon the reception of the REQ message, each of the neighbors will in turn check if its ID is in the NH list that is included in the received REQ message. If its ID is not included then it drops the REQ message. Otherwise, it processes the REQ message and continues its propagation until the message reaches the destination node which will

Algorithm 1 The main algorithm at an intermediate node

```

When a node  $y$  receives a REQ message
  Let  $r_y$  be the reliability of  $y$ 
  let  $r_{min}$  be the minimum acceptable reliability factor of the path
  let  $n_y$  be the number of 1-hop neighbors of  $y$ .
  let  $rf[z]$  be the reliability factor of node  $z$ .
  let  $MAX\_NH$  be the maximum number of 1-hop neighbors that can be included in the NH
  list
   $NH\_temp = \phi$ 
  Sort all 1-hop neighbors of  $y$  in descending order using  $rf[z]$  as a key
  for ( $z = 1; NH\_temp < MAX\_NH$  and  $z \leq n_y; z = z + 1$ ) do
    if ( $rf[z] \geq r_{min}$ ) then
      Add  $z$ 's ID to the  $NH\_temp$  list
    end if
  end for
   $r_{cum} = r_{cum} + r_y$ 
  if  $NH\_temp \neq \phi$  then
    let  $PATH\_temp = PATH \mid y$ 
    broadcast  $REQ(S, D, ID, r_{cum}, r_{min}, r_{max}, y$ 
       $PATH\_temp, NH\_temp)$  message
  end if

```

then unicast a replay (REP) message back to the source along the nodes collected in the PATH list to finish the path discovery process.

When and if the destination receives multiple REQ(s,d, ID) messages for the same path uniquely specified by the (s, d, ID) tuple, it can take one of the following two actions depending on the value of the application specific parameter MAX_PATHS that is defined earlier. If *BACKUP_PATHS* is equal to 0 then, the destination selects the path with the highest path reliability factor $PRF=r_{cum}/n$, where n is the number of intermediate nodes in the path (not including the source and destination nodes), and unicast a REP message back to the source. Otherwise, if *BACKUP_PATHS* > 0, then the destination sends up to (*BACKUP_PATHS* + 1) REP messages (if that many have been discovered) back to the source which can use the path with the highest *PRF* as the primary path and the other paths as secondary paths which can be used as backup paths when the primary path breaks.

2.6 A detailed example

Figure 2 shows a detailed example that illustrates the route discovery process using the RAS routing protocol between the source node A and the destination node G. In this case, the required path to transmit the data has a minimum reliability factor $r_{min} = 5$, and $MAX_NH = 2$. The source node, A, starts by sorting its 1-hop neighbors, nodes P, B, and S according to their respective reliability factors 10, 7, and 6. This information is contained in its reliability table which keeps updated information about the reliability factors of the node's 1-hop neighbors. All of A's neighbors meet the minimum requirement of 5. However, only nodes P and B are included in the NH list which can only

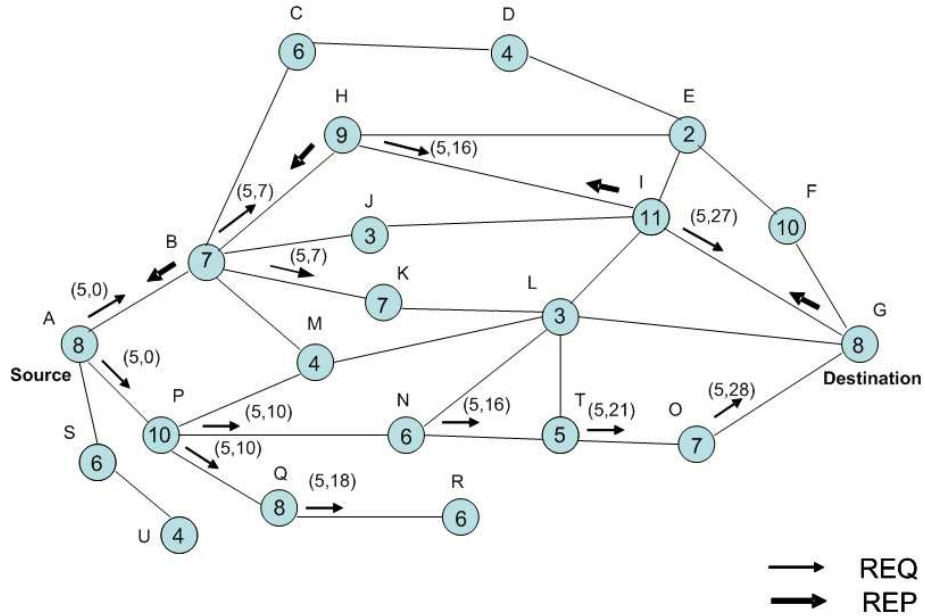


Fig. 2. A detailed example of the route discovery process.

contain a maximum of two 1-hop neighbors (for this path, $MAX_NH = 2$). The NH list is included in the REQ message along with a cumulative path reliability factor r_{cum} of 0. Note that the source and destination nodes' reliability factors are not included in r_{cum} because the cumulative reliability factor is meant to measure the reliability of the intermediate nodes that will be used for data transfer along the discovered path. The source and destination nodes need to send and receive the data and must be a part of the final path by default regardless of their own reliability factors. Node A includes its ID in the PATH list contained in the REQ message and broadcasts it to its neighbors.

In turn, when node B receives the REQ message, it sorts its 1-hop neighbors nodes H, K, C, M, and J according to their respective reliability factors of 9, 7, 6, 4, and 3. Only nodes H, K, and C meet the minimum requirement of 5, and only nodes H, and K are included in the NH list, since $NH_MAX = 2$. Subsequently, node B adds its ID to the PATH list included in the REQ message along with the constructed NH list and a cumulative reliability factor of $r_{cum} = 0 + 7 = 7$ (0 is the current cumulative reliability factor in the REQ message and 7 is the reliability of node B which will be broadcasting the REQ message). When the REQ message is received by the 1-hop neighbors, H, K, C, M, and J, each node will examine the NH list in the REQ message and check if its ID is included. Only the nodes whose id is included in the NH list will process the REQ message and try to propagate it further. The other nodes will simply drop the message. Using the same process, node H will further propagate the REQ message to node I with a cumulative factor $r_{cum} = 7 + 9 = 16$ (node E with the reliability factor $2 < 5$ will not be included in node H's NH list). Node I will send the REQ message to node G with a

cumulative reliability factor of $r_{cum} = 16 + 11 = 27$. Finally, the REQ message, arrives at the destination node G with the discovered path $A - B - H - I - G$ and a final cumulative reliability factor of 27.

Similarly to what node B did, node P, having received the REQ message from node A, will only include nodes N, and Q in its NH list along with a cumulative reliability factor of 10. The REQ message propagates from Q to R but does not go further since no further links exist out of R. The message also propagates from node N along nodes T and O to the destination node G with a final cumulative reliability factor of 28, and a discovered path which includes nodes $A - P - N - T - O - G$.

When the destination node G receives both REQ messages for the two discovered paths, it divides each path's final cumulative reliability factor with the number of nodes in the path to calculate the *normalized path reliability* factor $PRF = r_{cum}/n$. In this case, the destination node G determines that the discovered paths $A - B - H - I - G$, and $A - P - N - T - O - G$ have normalized reliability factors of $27/3 = 9$ and $32/4 = 8$ respectively. Therefore it chooses the more reliable path $A - B - H - I - G$ with the higher normalized path reliability factor of 9. It then unicasts a REP message back to the source node A along the discovered intermediate nodes in the PATH list to inform it of the discovered path. When node A receives the REP message, it updates its routing table with the discovered path and starts the data transmission process to node G. In our example, it is assumed that $BACKUP_PATHS = 0$. However, if $BACKUP_PATHS > 0$ then REP messages for both discovered paths would be sent back to the source. The latter will then use the path with the higher reliability factor as a primary path and saves the other discovered path in its routing table as backup path that would be used if and when the primary path fails later.

2.7 Maintenance of the reliability factor

The reliability factor of a particular node z is a measure of its past performance in being a part of successful data transmissions and it is maintained in the following fashion. At network initialization, all reliability factors are assigned an initial value $INIT_REL_FACTOR$. During normal network operation, once a path from the source to the destination is discovered, the source updates its routing table and starts data transmission along the discovered path through the intermediate nodes that belong to this path. There are several different algorithms that can be used to maintain the reliability factor:

Self maintenance of the reliability factor by the node: Each time a node successfully transmits a number of packets equal to $REL_FACTOR_RESOLUTION_NUMBER$ to another node along the path it increases its reliability factor by 1. The $REL_FACTOR_RESOLUTION_NUMBER$ is a constant set by the system administrator to make sure the actual reliability factor does not grow too large and is only incremented once for each predetermined number of successfully transmitted packets. This policy of updating the reliability factor by the node itself has minimal overhead and assumes no malicious nodes exist in the network. It is a policy designed to increase the reliability of the future discovered paths and not to combat malicious intentions by the nodes. Periodically, the reliability factor is automatically decremented by each node

in order to keep this variable from overflowing while keeping the relative values of the reliability factors in the nodes consistent.

Maintenance of the reliability factor by the acknowledgements: Using this policy, the destination sends positive acknowledgments to the source. These acknowledgments are cumulative and can be piggy backed with data transmissions in the case of a two-way communication between two nodes in order to minimize overhead. the reliability factor is incremented when an intermediate node that transmitted participated in data transmission receives positive acknowledgements from the destination.

Maintenance of the reliability factor by 1-hop neighbors: Another strategy for maintenance of the reliability factor is that each node y maintains a reliability factor for each of its n neighbors $z_1 - z_n$. This set of reliability factors for each of its neighbors constitutes its a measure of the reliability of each of them in its view based on their past performance in successful data transmissions. The reliability factor $rf[z]$ in node y is increased by 1 for each number of packets equal to $REL_FACTOR_RESOLUTION_NUMBER$ forwarded by this neighbor. During the data transmission phase, when y forwards a data packet to a neighbor z which is not the destination and is a part of the discovered path for that session, it listens to the subsequent transmission by z to its successor node in the path. In this case, y is applying what is called a *passive acknowledgement* process.

Maintenance of the reliability factor using link-based-acknowledgement: If z did follow up and forward the packet to its successor node, then it is rewarded by increasing its reliability factor in y . Otherwise, if z did not forward the data as it should, then it is penalized by not increasing its reliability factor. Consequently, it is less likely to be selected by y for forwarding REQ packets during future path discoveries. Therefore, it is also less likely to be a part of future data transmission paths.

2.8 The data transmission process

Once the source receives the REP message from the destination, it updates its routing table with information of the newly discovered path and starts data transmission. During the data transmission process, positive acknowledgments are sent from the destination to confirm successful reception of packets. As indicated earlier, in order to save bandwidth and minimize overhead, acknowledgments are cumulative and can be piggy-backed with data messages that could be sent in the opposite direction from the destination to the source during a two-way communication session. The acknowledgements are used in the process of maintenance of the reliability factor at all of the nodes involved in the data transmission path. In addition to the acknowledgement process, the source node uses a time out timer named ACK_REC_TOT (acknowledgement reception time out timer), which is initialize at the beginning of the data transmission phase to the value $INIT_ACK_REC_TOT$. The timer is refreshed each time an acknowledgment is received by the source. This indicates that the path to the destination is still valid. If the ACK_REC_TOT expires, then the source retransmits the unacknowledged packets. If the retransmitted packets are also not acknowledged, another retransmission attempt is made, until a maximum number of attempts is reached $MAX_RETRANS_ATTEMPTS$. At that time, the source assumes that the path is broken and starts another path discovery process.

3 Additional Security Issues and Future Work

This paper presented a reliable routing protocol for wireless ad hoc and sensor networks. However, malicious nodes in the network may affect the operation and efficiency of the proposed protocol. We identified four types of attacks that malicious nodes can generate. (1) The first type is related to the sniffing nodes. To do its attack, the malicious node redirects the selected traffic so that it can perform traffic sniffing. The second type is concerned with malicious nodes that attempt to attack the path discovery process. The malicious node does not extend the request message to nodes that have the minimal reliability factor specified by the source. Rather, the malicious node sends the request message to nodes with less reliable factor. So that, the source node will later use a non optimal path, hence damaging the efficiency and performance of the network. (3) The third type is concerned with malicious node that attempt to block any traffic that go through. Obviously, its reliability factor will decrease considerably, in the case where the factor is maintained by other nodes, and consequently no path will include that node. However, for a period of time, this attack may disturb the normal activities of the network. (4) The fourth type is concerned with a malicious node that, at the beginning, behaves as a very reliable node, in order to be included in the paths, and then the node can carry an attack. In such a situation, the network performance may be considerably affected since most traffic will go through the malicious nodes and consequently a denial of service attack can take place.

4 Conclusions

In this paper a reliable routing protocol for wireless mobile ad hoc and sensor networks was presented. The protocol provides increased reliability of communication by having intermediate nodes extend the REQ message only to the more reliable neighbor nodes using a reliability factor. The reliability factor is increased when nodes participate successfully during the data transmission process by using an acknowledgement mechanism. Positive and passive acknowledgement mechanisms are used in the maintenance of the reliability factor. Different parameters are used to optimize and enhance the routing process like reducing flooding during the path discovery phase, and the use of backup routes for more timing critical applications with increased priority. In the future, we intend to improve this protocol further by applying more techniques in optimizing the selection of the next-hop neighbors. In addition, we intend to further study, and analyze the performance of the protocol through simulation.

References

1. T. Anantvalee and J. Wu. Reputation-based system for encouraging the cooperation of nodes in mobile ad hoc networks. *Proc. of IEEE ICC*, 2007.
2. M. Barry and S. McGrath. QoS techniques in ad hoc networks. *Proc. of 1st International ANWIRE Workshop, Glasgow*, April 2003.
3. W.-P. Chen and J. C. Hou. Dynamic, ad-hoc source routing with connection-aware link-state exchange and differentiation. *Proc. IEEE Globecom*, November 2002.

4. S. De, S.K. Das, H. Wu, and C. Qiao. A resource efficient RT-QoS routing protocol for mobile ad hoc networks. *Wireless Personal Multimedia Communications, 2002. The 5th International Symposium on*, 1:257–261, 2002.
5. S. Hadim, J. Al-Jaroodi, and N. Mohamed. Trends in middleware for mobile ad hoc networks. *The Journal of Communications*, 1(4):11–21, July 2006.
6. Y. Hwang and P. Varshney. An adaptive QoS routing protocol with dispersity for ad-hoc networks. *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, pages 302–311, January 2003.
7. I. Jawhar and J. Wu. Qos support in tdma-based mobile ad hoc networks. *The Journal of Computer Science and Technology (JCST)*, 20(6):797–910, November 2005.
8. I. Jawhar and J. Wu. Quality of service routing in mobile ad hoc networks. *Resource Management in Wireless Networking*, M. Cardei, I. Cardei, and D. -Z. Du (eds.), Springer, *Network Theory and Applications*, 16:365–400, 2005.
9. I. Jawhar and J. Wu. Race-free resource allocation for QoS support in wireless networks. *Ad Hoc and Sensor Wireless Networks: An International Journal*, 1(3):179–206, May 2005.
10. R. Leung, J. Liu, E. Poon, Ah-Lot. Chan, and B. Li. Mp-dsr: A qos-aware multi-path dynamic source routing protocol for wireless ad-hoc networks. in *Proc. 26th IEEE Conference on Local Computer Networks (LCN 2001)*, pages 132–141, 2001.
11. R. Ma and J. Ilow. Reliable multipath routing with fixed delays in manet using regenerating nodes. in *proc. 28th IEEE International Conference on Local Computer Networks (LNC'03), Bonn, Germany*, pages 719–725, October 2003.
12. Jamal N., Al-Karaki, and Ahmed E. Kamal. Enforcing cooperation in mobile ad hoc networks. *Springer Wireless Personal Communications*, 2006.
13. S. Nelakuditi, Z.-L. Zhang, R. P. Tsang, and D.H.C. Du. Adaptive proportional routing: a localized QoS routing approach. *Networking, IEEE/ACM Transactions on*, 10(6):790–804, December 2002.
14. V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. *INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 3:1405–1413, April 1997.
15. K. Paul and D. Westhoff. Context aware detection of selfish nodes in dsr based ad-hoc networks. *Proc. IEEE Semiannual Vehicular Technology Conference (VTC-2002)*, September 2002.
16. C. E. Perkins. *Ad Hoc Networking*. Addison-Wesley, Upper Saddle River, NJ, USA, 2001.
17. C. E. Perkins and E. M. Royer. Ad hoc on demand distance vector (AODV) routing. *Internet Draft*, August 1998.
18. Y. Rebahi, V. E. Mujica-V, and D. Sisalem. A reputation-based trust mechanism for ad hoc networks. *Proc. 10th IEEE Symposium on Computers and Communications (ISCC 2005), Spain*, pages 37–42, June 2005.
19. J. L. Sobrinho and A. S. Krishnakumar. Quality-of-service in ad hoc carrier sense multiple access wireless networks. *Selected Areas in Communications, IEEE Journal on*, 17(8):1353–1368, August 1999.
20. K. Sundaresan, V. Anantharaman, H-Y, Hsieh, and R. Sivakumar. A reliable transport protocol for ad hoc networks. *IEEE Transactions on Mobile Computing*, 4(6):588–603, Nov/Dec 2005.
21. H. Xiao, K. G. Lo, and K. C. Chua. A flexible quality of service model for mobile ad-hoc networks. *Proceedings of IEEE VTC2000-Spring, Tokyo*, May 2000.
22. Z. Ye, S. V. Krishnamurthy, and S. K. Tripathi. A framework for reliable routing in mobile ad hoc networks. *IEEE INFOCOM 2003*, 2003.