Intelligent Automation & Soft Computing
DOI:10.32604/iasc.2022.021127
Article

Tech Science Press

A Dominating Set Routing Scheme for Adaptive Caching in Ad Hoc Network

Raed Alsaqour[1], Ammar Al-hamadani[2], Maha Abdelhaq[3,*] and Joud Almeheimidy[3]

[1]Department of Information Technology, College of Computing and Informatics, Saudi Electronic University, 93499, Riyadh, Saudi Arabia
[2]Optical Cable Maintenance Center, Ministry of Communications, Baghdad, Iraq
[3]Department of Information Technology, College of Computer and Information Sciences, Princess Nourah Bint Abdulrahman University, 84428, Riyadh, Saudi Arabia
*Corresponding Author: Maha Abdelhaq. Email: maha.ukm@gmail.com
Received: 24 June 2021; Accepted: 02 September 2021

Abstract: Current efforts for providing an efficient dynamic source routing protocol (DSR) for use in multi-hop ad-hoc wireless are promising. This is since DSR has a unique characteristic in that it uses source routing, instead of relying on the routing table at each intermediate device. This study addresses the current challenges facing DSR protocol in terms of the dynamic changes of the route and how to update such changes into the route cache of the DSR. The challenges typically persist when a sudden route break occurs resulting in a delay in updating the new node location into the cache of the DSR protocol. For that, this study proposes a novel algorithm to improve the cache updating of DSR protocol in the ad-hoc network using dominating set-based routing (DBR). In DBR, the dominating nodes establish node update cache in accordance with the characteristics of the new route. Network Simulator version 2 (NS2) was used to implement and evaluate the proposed algorithm. A comparison of certain performance metrics was carried among the proposed DSR-DBR, DSR-route-cache, and DSR-original in the transmission control protocol and user datagram protocol. The DSR-DBR performance result showed a significant improvement in the average throughput, average end-to-end delay, average discovery time and routing overhead.

Keywords: Dynamic source routing; ad-hoc network; routing; cache; dominating set

1 Introduction

Due of its relevance in essential applications such as battlefield communication and disaster assistance without requiring a fixed or static setup, the rising interest in Mobile Ad-hoc Networks (MANETs) has had an influence on academic research [1–3]. MANETs may be defined as a wireless network with a quickly changing topology [4,5]. MANET can also be used independently. It has the potential to be linked to a larger Internet network. Due to the rising popularity and relevance of laptops and Wi-Fi, MANETs became well-known among academics in the 1990s [6].

DSR is a simple, efficient wireless protocol for the multi-loop setting of MANET's nodes [7,8]. Additionally, DSR protocol features low overhead, and the caches of the route reduce the cost of route discovery. Its scope of application is in multi-path routing and non-symmetrical transmit pattern [9]. DSR

as a protocol is categorized as an on-demand protocol. It prioritizes finding a route to a destination when it is presented with a packet looking to reach that destination. It features route maintenance procedures until that destination becomes unreachable from every possible path in the network.

Cache staleness is regarded one of the difficulties in DSR protocol development [10,11]. Incomplete error notification and inadequate cache size may lead to further issues with DSR. The ad hoc network has nodes interacting with each other by means of a series of intermediary nodes that serve as routers. However, because nodes are transmitting randomly, there will be changes in the network topology over time. The last contributing factor was the changes in network topology, which caused inadequate routing protocol performance. Concretely, the design of protocols like DSR is used as a medium for transmitting node till its discovered route expires or is broken. The method therefore creates several communication issues owing to the movement of network hosts and/or low signal strength. Furthermore, when the next hop learns of a connection break, it will report a route error back to the originating node in order to start route discovery. Because of this, route discovery processes in a major network flow may be costly. Wasting wireless bandwidth as well as frequent route-finding packets are additional consequences of having low bandwidth.

One source that causes cache staleness is linked breakage for DSR protocol [12,13]. Thus, Efficient algorithms are required to refresh network information in the cache. Once broken, a connection affects the entire network. A prediction algorithm is usually used to kick-off a full maintenance process ahead of time [14]. Therefore, this study is attempting to overcome the dynamic changes of the route and how to update such changes into the route cache of the DSR. This was investigated from the sudden route breaking and how the update of the new node location in the cache of the DSR protocol.

The remainder of this work is structured as follows: Section 2 presents the background and related work. In Section 3, we present the algorithm for altering and modifying the route cache. In Section 4, we present the implementation of dominating set-based routing in DSR protocol. Section 5 presents the simulation and experiment Settings. Section 6 presents the research results and discussion, and Section 7 presents the conclusion and future work.

2 Background and Related Work

The Internet Engineering Task Force (IETF) created a routing architecture for IP-based protocols on MANETs, which is divided into two categories: table-driven protocols and on-demand protocols [15]. Each node in a MANET may serve two purposes: it can operate as a router for other nodes and it can be routed to another node. In MNET, different routing protocols have been developed, including dynamic source routing protocol (DSR), destination-sequenced distance-vector (DSDV) routing [16], and ad hoc on-demand distance vector (AODV) [2,9].

Route discovery and route maintenance are the two major functions of the DSR protocol. When a network source wants to send a packet to a certain destination, it broadcasts a request to find a route to that destination. The source and destination node addresses, a unique sequence number, and an empty route record are all included in the request packet. The intermediary nodes will then do a cache check. If the cache doesn't hold the ultimate destination address, the intermediary node will broadcast the request. However, as the size of the network expanded, so did the route cache in the DSR protocol, slowing down the route-finding process [17]. Furthermore, by recording a route in the cache that may break in the future, the broken route obstructs the search process (route changes).

Several studies have looked into the issue of abrupt route changes. For example, in [18], the authors have concentrated on specific drawbacks of DSR in terms of routing cache. These drawbacks are consisting of three sub-problems stated as: incomplete notifying error, no expiry, and quick pollution. For this purpose, the authors have proposed a Wider Error Notification approach which rely on a rapid and wide propagation of Route Error (RERR) packet to speed up the transmission and expand the size the of (RERR) packet. This can be represented by an update task conducted by the node for its route cache once it receives a RERR packet includes the link failure information.

In [15], the authors have presented an adaptive link cache (ALC) scheme, which is a combination of link cache and adaptive timeout policy. ALC aims to overcome the problem of stale routes in DSR cache routing. This can be represented by an elimination process for the stale routes based on heuristics search that aim to predict the lifetime of a link. Nonetheless, these heuristics search do not provide robust estimation for the timeout of the link, this is due to the unpredictable changes that would be occurred in the topology of the network.

In [19], the authors have presented an algorithm called distributed adaptive cache update (DACU), which aims to accommodate an updating process for the DSR routing cache. DACU algorithm utilizes a proactive cache update rather than the adaptive timeout approach in order to eliminate the stale routes. In addition, DACU algorithm aims to gather information in terms of distributing the routing information through the network. Since DACU algorithm is relying on a path cache thus, the routing information about the state of the network for each node would not be effectively exploited. Furthermore, the cache timeout is not being used.

In [20], a caching method was devised that allows nodes to quickly react to changes in the cache network. It entails adding a new packet to the DSR cache and having the new packet visit each node in the network twice. The topological data is gathered on the initial visit, and the neighbor finding method is activated. The new packet resumes its visit to another node after storing this information into a compressed matrix. The node updates and verifies the nodes' link caches during the second visit of this packet.

The researchers in [21], suggested a novel method that uses the DSR routing cache to enhance the routing between mobile nodes in order to decrease the result of mobility in link transmission, which might help to overcome the link broken crisis. This technique updates the DSR's route cache by using proactive cache replace rather than an adaptive timeout mechanism in a link cache structure to delay a stale route in a session.

In [22], the authors have presented an ant colony approach in order to enhance the routing of DSR. Basically, when a node sends a packet to another node, it firstly identifies the cache in terms of existing routes. If there are no routes, the node will locally transmit a route request control packets in order to figure out the routes. Like the biological ant behaviour where the ant that delivering the food aims to drop a sign for other ants in order to determine the route, the request will be transmitted through the network with all the information including length of route, size of packets and the crowd within the route. Once the destination receives the request, it replies with the same information. In the case of multiple destinations responses obtained, the ant colony approach will identify the best route and select it by the sender node. However, if the number of nodes is increased, the ant colony approach will have difficulty in identifying the best route regarding to the complex computations required.

3 DSR Route Caching Algorithm

The issue with cache route using the DSR protocol is the dependence on the cache route, when it has to route the network's packet [23]. This presents the greatest challenge of the size of cache route; usually, as the route cache size will be already enlarged with plenty cache records holding the whole of the route. Therefore, a source will first check the route cache when a node needs to send a packet to a destination one. This is for finding the best route usable for sending packet to destination. The broken route is experienced because of nodes speed, empty battery, and route order problem as well as out of transmission range. When the new route attached to the route cache is used in indicating where a new route will append to the end of the cache, the node selects the old route to send a possibly broken packet.

Our solution to this problem is the proposal of a new algorithm for altering and modifying the route cache. This is the proposed solution aimed at solving broken off routes, route search times and cache size problems. In addition, there will be a division into two sub-caches, the cache of DSR protocol. The first cache part is referred to as "Master Route Cache", and it is for saving all relevant source node

information, the numbers of hops, route status as well as the destination node. The second cache part is referred to as "Route index", and it is for saving all route indexes, not exempting route from source to destination node. Algorithm 1 lists the pseudo code for DSR cache updating.

Algorithm 1: Pseudo code for update cache

```
broke link = false
While ( broke link = false ) {
  cache status = check node cache ( )
  If ( cache status = true ) {
    Take route for second part ( )
    }
  Else {
    Receives RREQ = false
    While ( receives RREQ = false ) {
      Send RREQ to neighbor node ( )
      Receives RREQ = intermediate node receives ( )
       If ( receives RREQ = true ) {
         Reverse route( )
         Send RREP to source ( )
         While ( route already = true ) {
             Take route for second part ( )
         }
       else
         Add new route to cache ( )
         If ( has route to destination )
            Add it's to route ( )
         else
            Put its retransmit RREQ ( )
         }
    Faction Take route for second part ( ) {
      Take route ( );
      if ( broken link = false )
        End
      else {
        Send RREP ( )
        Source deactivate route (0)
        }
    }
}
```

As shown in Algorithm 1, the algorithm begins by initializing a stimulating environment, cache node structure. If there is a packet requires sending by a node, then the algorithm will check for router status. Such checking aims to determine the route activeness of node cache. thus, if there is an available one, then this route will used for the transmission. Otherwise, a request will send to the neighbouring nodes. In this manner, an intermediate node will receive the request and check its destination in terms of availability whether it is available (then store such routes in the available routes) or it is not possible (then sends the request to another neighbouring node). Once a neighbouring node is figuring out a possible route, a reply will be sent to the source. Hence, the source will analyse the cache node of the replay if such route has been already updated to a status (1) then the route of second part of the cache will be used for the transmission. Otherwise, such route will add to the new route cache. A final procedure by the algorithm will be represented as identifying a broken link, if there is no broken link then the algorithm will be ended. Otherwise, a reply message will be sent to the source that will deactivate the route to the status (0) and then the algorithm will be ended.

4 Implementation of DBR in DSR Protocol

In graph theory, for a graph G = (V, E), a dominant set is a subset D of V where every vertex not in D is adjacent to at least one member of D. The domination number γ(G) is the number of vertices in a smallest dominating set for G. For example, in the Petersen graph illustrated in Fig. 1, the set S = {1, 2, 9} is a dominating set.

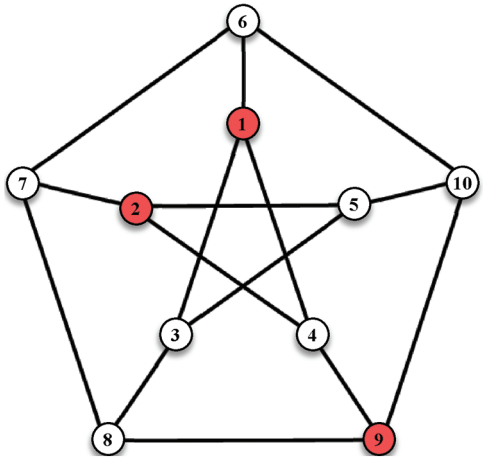

Figure 1: Dominating set concept

[24]. The current issues associated with identifying the dominant of nodes was resolved by using the method of “HELLO” message.

The proposed DSR-DBR algorithm starts with initiating Node (A). Algorithm 2 lists the pseudo code for DSR-DBR. As shown in Algorithm 2, The dominating set is found as the first step in the algorithm. The route will then be determined depending on the destination and only through the dominating nodes. The associated nodes in the dominating set will then be utilized to link all of the nodes in the network, making it simple to reach the destination in a shorter amount of time. However, if a route failure happens, the associated dominating node recognizes the issue and resolves it locally. If practicable, this can be accomplished by allowing the node to reach the destination through other nodes. If not, it will send a router failure message to the other dominant nodes. The adjacency matrix is formed in this step, with each node

determining its neighbor node by delivering the HELLO packet. The neighbouring list is transmitted to the neighboring nodes when the neighbors have been determined, and each node produces the adjacency matrix. It is expected that by using this matrix, identifying the dominant nodes and dominating set will take less time.

Algorithm 2: pseudo code for DSR-DBR.

```
If (B is a neighbor) {
   add to neighbor list
   send this list to its neighbors
}
If (A is a neighbor of B)
    adjacency [A][B] = 1
else
    adjacency [A][B] = 0
For each row in adjacency matrix {
   Compute the row sum in adjacency matrix
   Find the maximum connected node
   Append this to "dominating set"
   If any node is not connected to the nodes in the dominating set then
       add this node to the dominating set
}
```

The suggested algorithm's scenario is shown in Fig. 2. It begins by establishing the dominant set as <N3, N6, N10>, with N1 as the source and N10 as the destination nodes. N10 is accessible from N1 via N3 and N6. Each node seeks to identify the dominating node from the source node during the first route discovery procedure. Even if the shortest path to the target exists, a node's journey to the destination is always through the dominant nodes. As a result, it offers a customizable option for locating an alternate path that is considerably faster than the traditional method. If the N6-N8 connection fails, the node N6 can simply establish the N6-N7 connection to the destination [25].

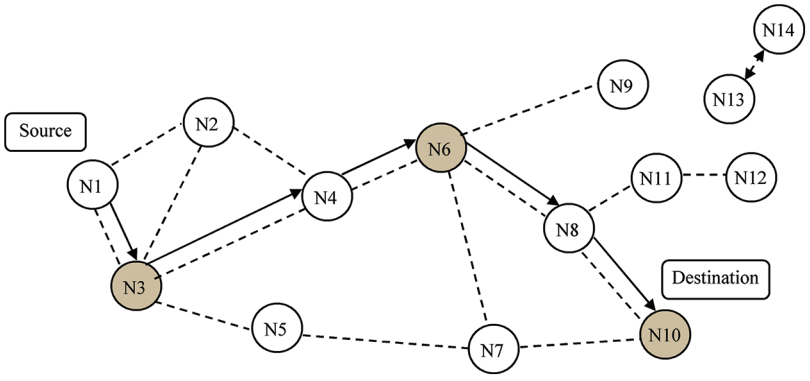

Figure 2: DSR-DBR mechanism

Based on this, it can be concluded that the proposed algorithm handles two main situations; building a link verifier and a new cache that contains an index to the links, as well as to the original master cache. The original master cache will save the information on the source and destination nodes. The second cache will save the indices of the links and routes. A detailed flowchart of the algorithm is shown below in Fig. 3.

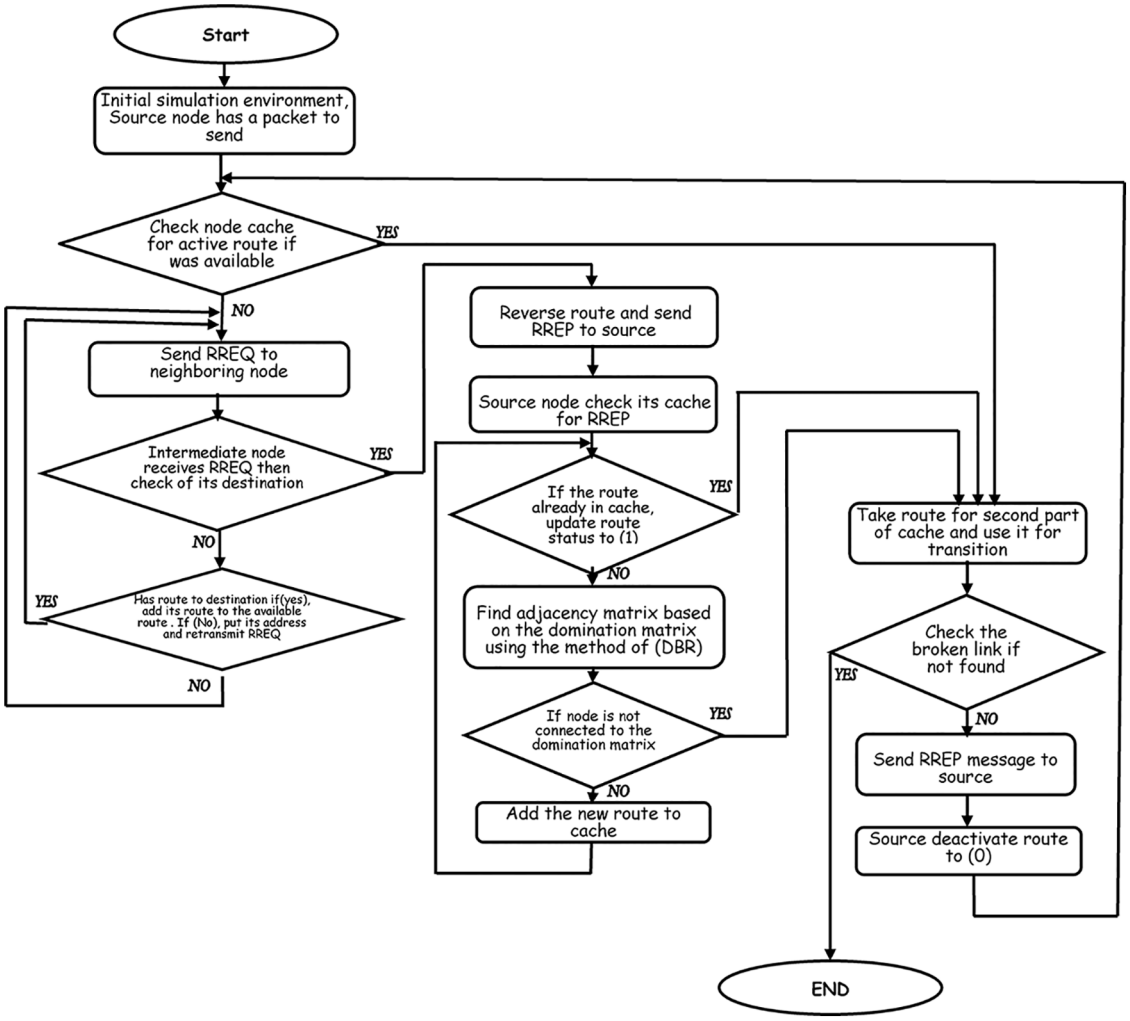

Figure 3: DSR-DBR algorithm

5 Simulation and Experiment Settings

5.1 Simulation Environment

To perform the simulation, the Network Simulator (version 2) has been used. The aim of this simulation is analysis the DSR protocol for its efficiency in terms of (average throughput, average E2E delay, average discovery time and routing overhead), this has been made by measuring these metrics in a different (node speed and packet size and taking into caused evasion the remaining number of node and traffic connection. The simulation parameters that have been used in this study are shown in Tab. 1.

Table 1: Simulation parameters

Description	Value	Unit
No. of node	30	node
Network area size	800 × 800	m^2
Node speed	0, 2, 4, 6, 8	m/s
Node transmission range	30	meter
Data packet size	512,1024, 1536, 2048, 2560	bytes
Network layer protocol	DSR protocol	-
Mobility model	Random waypoint	-
Transport layer protocol	TCP/UDP	-

5.2 Performance Metrics

5.2.1 Routing Overhead

Routing overhead refers to the ratio of the amount of routing-related control packet transmissions to that of data transmissions. The number of control packets transmitted in the (*ith*) application traffic is represented by (*cpki*), while the number of data packets transmitted in the (*ith*) application traffic is represented by (*pkti*). The average routing overhead of application traffic (*n*), indicated by (*RO*), is calculated as follows:

$$\mathrm{RO}\ (\%) = \frac{1}{\mathrm{n}} \sum_{i=1}^{\mathrm{n}} \frac{\mathrm{cpki}}{\mathrm{pkti}} \times 100\% \tag{1}$$

5.2.2 Average End-to-End Delay

This metric refers to the average time the source-to-the destination of a packet requires. Using (*di*) to represent destination node has accumulated total delay of packets, $(pktdi)$ represents the total number of packets that the destination node has received. The average application traffic end-to-end delay (*n*), denoted as (*E2E*), is calculated as follows:

$$\mathrm{E2E}\ (\mathrm{ms}) = \frac{1}{\mathrm{n}} \sum_{i=1}^{\mathrm{n}} \frac{\mathrm{di}}{\mathrm{pktdi}} \tag{2}$$

5.2.3 Throughput

This metric refers to the total data amount (*bi*) received by the destination divided by the time (*ti*) before the destination gets the final packet. The resulting output denotes the number of bits transferred each second. The throughput of the application traffic (n), is obtained as:

$$\mathrm{Throughput}\ (\mathrm{bytes/s}) = \frac{1}{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{n}} \frac{\mathrm{bi}}{\mathrm{ti}} \tag{3}$$

5.2.4 Route Discovery Time

This metric is the time duration from the source node to the destination when a Route Request (RREQ) packet is assembled until the Route Reply (RREP) packet is received. The *RDT* is obtained as:

$$\text{RDT (ms)} = \frac{1}{n}\sum_{i=1}^{n}\frac{\text{RREQti} - \text{RREPti}}{\text{No.of Pkt}} \quad (1)$$

6 Results and Discussions

This section explains the results from running the NS2 on metrics such as throughput, end-to-end latency, routing overhead, and route discovery time. Results were evaluated using both the TCP and UDP protocols. For DSR-DBR, two prior studies were compared: DSR-ROUTE-CACHE [21] and the original DSR. Using the three approaches, we ran the comparison with relation to the speed of the nodes. This is to offer a wider look at the suggested approach to deal with transferred data in varying nodes' speed.

Fig. 4a, showed the average throughput result for the three techniques in TCP with regards to the node speed. From the comparison result, it can be summated that the proposed solution offered a relatively better average throughput in TCP. These results were found to be better than the throughput results of DSR-ROUTE-CACHE and DSR-ORIGINAL respectively. Fig. 4b, showed the average throughput value for these techniques in UDP. The result revealed that the proposed DSR-DBR outperformed other techniques which were found to consume 46419.34 bytes when the speed is 0 ms and 46416.51 bytes when speed is 8 ms. However, DSR-ORIGINAL was found to be the lowest in providing a reliable throughput with bytes consumption ranged between 43644.4–43697.25 followed by DSR-ROUTE-CACHE, which offered relatively better result with throughput ranged between 45126.09–45019.1.

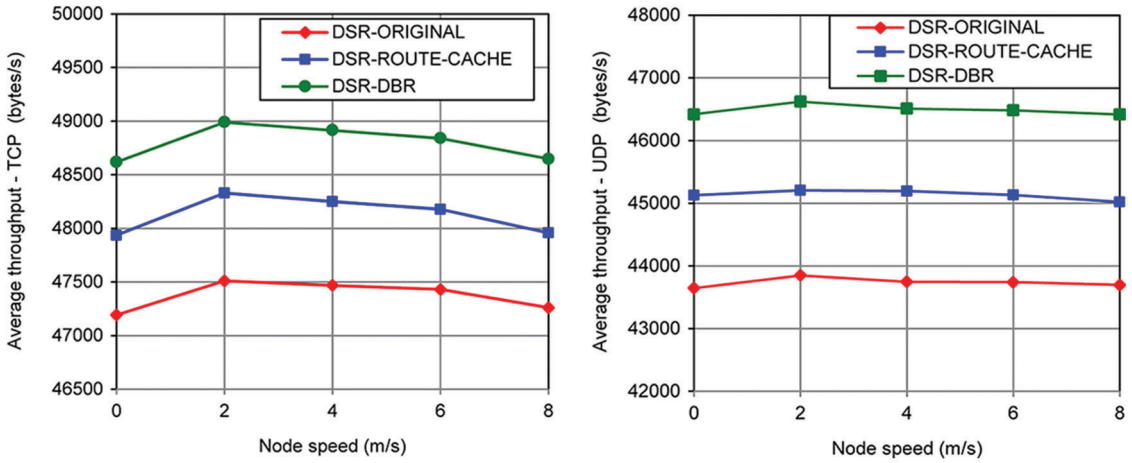

Figure 4: Average throughput versus node speed, (a) TCP, (b) UDP

It can be noted that protocol stability was steady in the proposed DSR-DBR in UDP more than TCP. This can be reasoned to that TCP usually utilizes window-based congestion control which it requires an additive increase in the network flow. It is also assumed that the proposed scheme managed the speed of each node by distributing it in an interval way. In addition, the average throughput *vs.* node speed in the proposed scheme is better when there is less aggregation of nodes. This is because the balance of nodes here plays a key role in routing protocol performance as compared to other schemes for both TCP and UDP in terms of node speed for delivering packets per second.

The E2E delay for the three techniques was also compared in both TCP and UDP with regards to the gradual increase in network speed. Fig. 5a, shows the average E2E delay in TCP which revealed a

relatively less delay when using the proposed DSR-DBR with a period ranged between 8.30051 ms - 8.24124 ms. This result is somehow better than the delay of DSR-ROUTE-CACHE which resulted in a delay ranged between 8.39081–8.32975 followed by DSR-ORIGINAL (8.46643–8.40919).

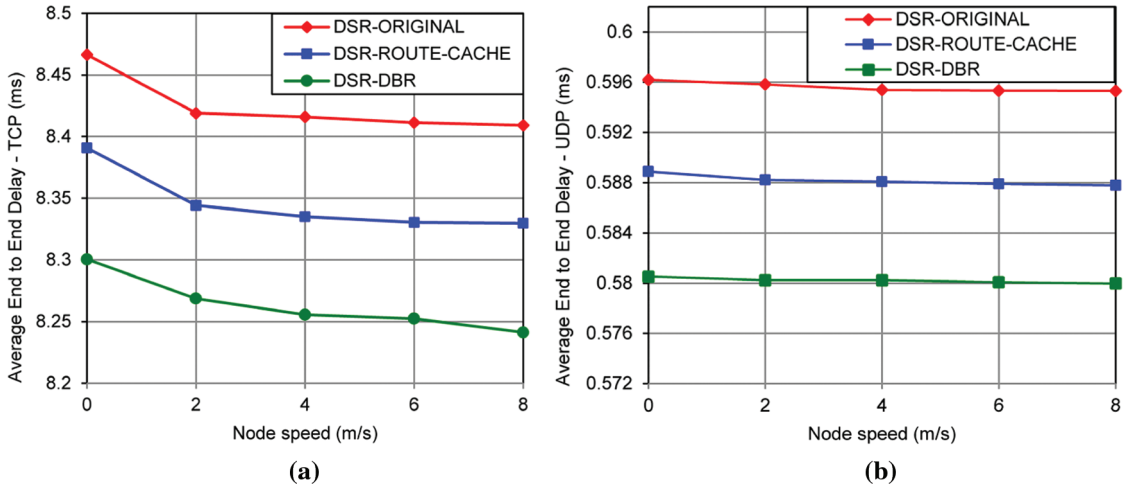

Figure 5: Average E2E delay versus node speed, (a) TCP, (b) UDP

The same is true in UDP in which the result of the average E2E delay was in its lowest value when using DSR-DBR with delay value ranged between 0.580528 ms - 0.579981 ms see Fig. 5b. This was followed by DSR-ROUTE-CACHE (0.588892–0.587797). From these results, it can be concluded that DSR-DBR was able to manage data transfer with minimal delay as compared to other techniques. This can be reasoned to the state of these techniques in route discovery which may takes longer in DSR-ORIGINAL and DSR-ROUTE-CACHE, which results in more delay for TCP and UDP packets. It can be also said that the proposed scheme's average packet delay decreases when the number of nodes increase which may be caused by buffering resulted during the process of route discovery latency, queuing at the interface queue in both TCP and UDP. Therefore, the proposed scheme provides less delay for a packet to reach the destination by balancing the delays, in the source and each intermediate host, caused by other events.

The routing overhears also tested on the three techniques in TCP and UDP with regards to the gradual increase in network speed. The result is shown in Fig. 6a, revealed that the routing overhead in TCP was in its lowest state when using the proposed DSR-DBR, which was found to consume 60.0016% when the speed is set to 0 ms and 60.00235% when the speed is set to 8 ms. The routing overhead, however, was higher in DSR-ROUTE-CACHE with consumption ranged between 60.0018–60.0024 followed by DSR-ORIGINAL (60.0021–60.00245).

Furthermore, the routing overhead in UDP with regards to the network speed was also examined see Fig. 6b. The result showed a steady routing overhead consumption for the three techniques when network speed is increased gradually. However, the result was at its best when using the proposed DSR-DBR with routing overhead consumption of 60.0023%-60.00406%, followed by DSR-ROUTE-CACHE (60.0028–60.00415) and DSR-ORIGINAL (60.0031–60.00425). It is also believed in this study that the suggested method efficiently controls node transfers by automatically lowering the routing packet overhead of DSR, allowing it to precisely track the routes currently in use. However, other schemes seem to experience a progressive frequency of route breaking in which the routing overhead to discover new routes also increase.

The RDT results for both protocols with regards to the network speed were examined using the three techniques. Fig. 7a, showed the RDT result for TCP in seconds in which it can be clearly observed that

the proposed DSR-DBR provided stable transfer rate followed by DSR-ROUTE-CACHE and DSR-ORIGINAL, respectively. This was further observed from the result of average RDT in ms shown in Fig. 7b. From the figure, it can be concluded that DSR-DBR provided the lowest RDT with a value range from 5.6839066 ms when the speed is 0 ms and 7.7629749 ms when the speed is 8 ms. However, DSR-ORIGINAL was found to consume higher RDT with value ranged between 7.8971926 ms -9.3455145 ms while DSR-ROUTE-CACHE was found to offer relatively better RDT (7.3009532–8.579142). On the other hand, the proposed scheme is found to provide a comparative RDT average as compared to other schemes where route replies from less strong links. This, as a result, increases the chance of having some packets that may not have any routes to be maintained. In TCP, the proposed scheme provides the required control of velocity when movement increases. This, as a result, led the routes to experience shorter time and more route discoveries as compared to the other two schemes.

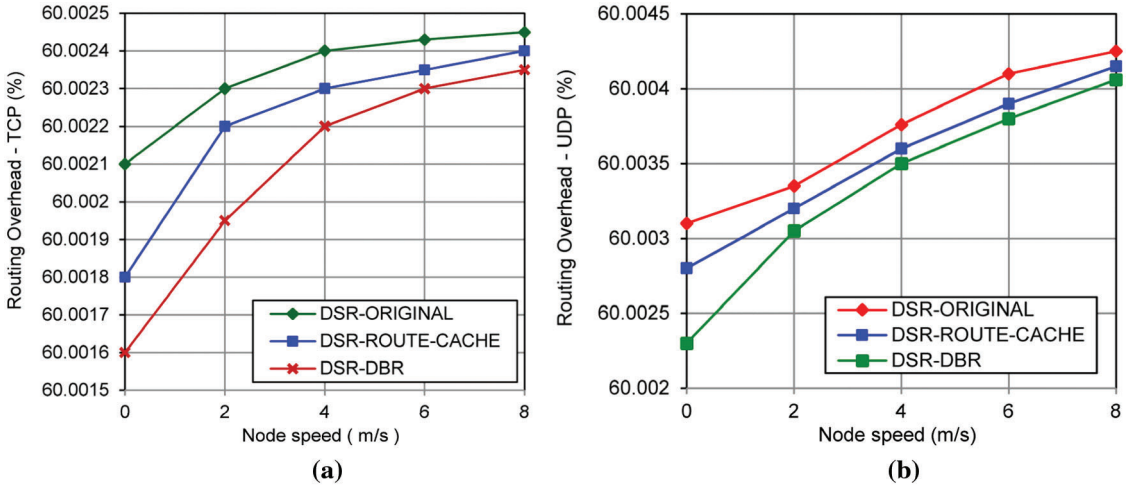

Figure 6: Routing overhead versus node speed, (a) TCP, (b) UDP

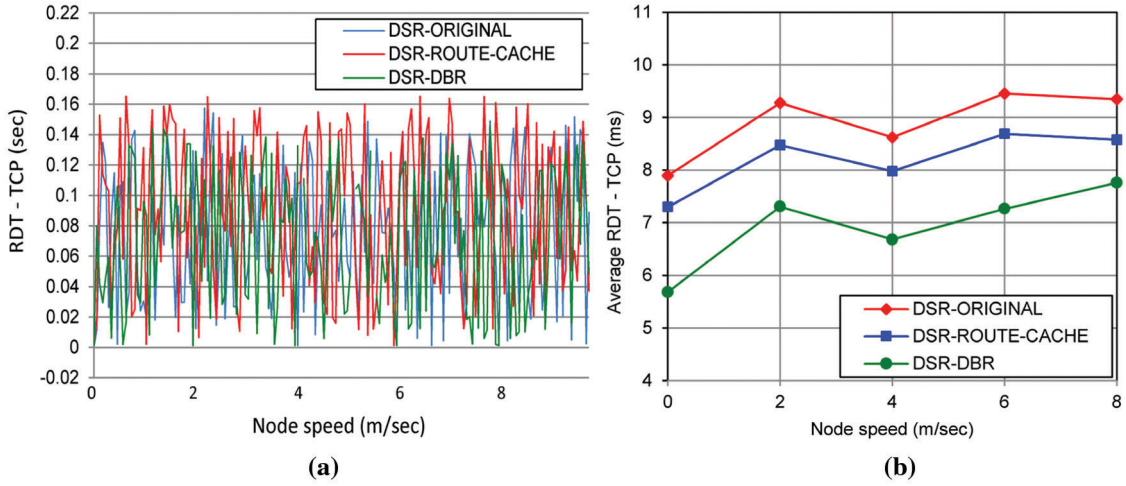

Figure 7: RDT versus node speed, (a) RDT-TCP, (b) average RDT-TCP

Fig. 8a, showed the RDT in UDP, which revealed a consistent result when using the proposed DSR-DBR followed by DSR-ROUTE-CACHE and DSR-ORIGINAL. On the other hand, the average RDT in ms was also calculated for the three techniques. The result Fig. 8b, showed that the RDT for DSR-DBR (0.5671702 ms-0.5442459 ms) outperformed the RDT for DSR-ROUTE-CACHE (0.593485–0.5773177) and DSR-ORIGINAL (0.610298–0.6029591) respectively. In UDP, is can be said that the proposed scheme provided a sufficient way of handling the incoming and outgoing packets once it has been first discovered. Meanwhile, the proposed scheme is believed to offer the suitable basis for processing each node in the network especially when the route error message reaches the initiator. In contrast to other schemes, the initiator deletes all routes identified as broken from its route cache, making the proposed scheme more compatible than others based on the results presented below.

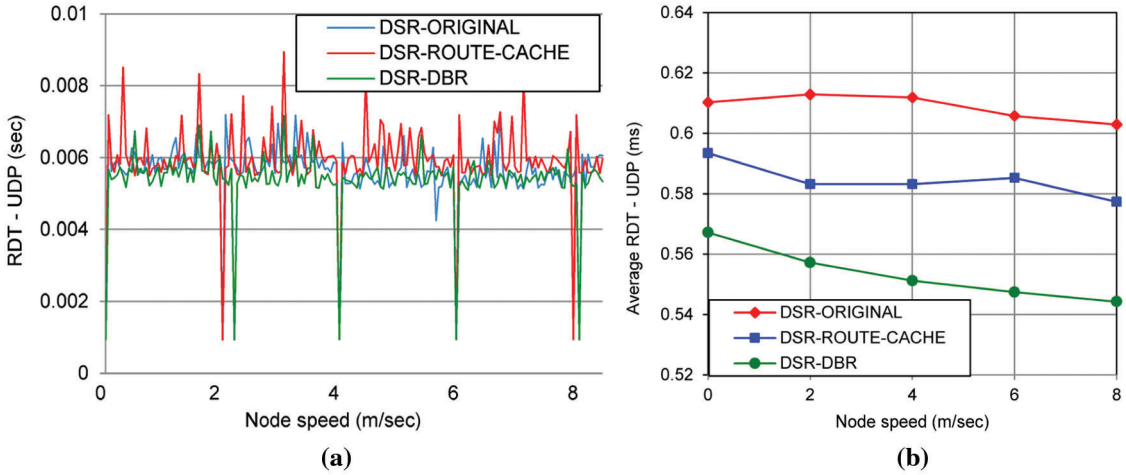

Figure 8: RDT versus node speed, (a) RDT-UDP, (b) average RDT-UDP

Meanwhile, the network metrics in terms of the average throughput, E2E delay, routing overhead, and RDT verses the packet speed in TCP and UDP were examined. Fig. 9a, shows the average throughput value for the three techniques with regards to the packet speed in TCP. The obtained result clearly shows how the proposed technique DSR-DBR performed better than other techniques. For example, DSR-DBR resulted in a throughput value of 49496.03 bytes when the packet speed was 512 sec and 101723.96 bytes when the packet speed was 2560 sec. based on this, it can be concluded that the proposed method outperformed DSR-ROUTING-CACHE (48115.86–99260.03) and DSR-ORIGINAL (43665.75–96676.02) respectively.

As for the UDP, the result shown in Fig. 9b, revealed that DSR-DBR provided a stable average throughput value, which was ranged from 46428.61 bytes - 46554.3 bytes as compared to DSR-ROUTING-CACHE (45068.62–45119.32) and DSR-ORIGINAL (43665.72–43872.35). From this, it can be summated that the proposed technique offered a better throughput result when gradually increases the packet speed in UDP. The result proved the compatibility of the proposed scheme against others by increasing the chances of discovering broken links, which result in significant throughput improvement in both TCP and UDP. In addition, the proposed scheme is believed to experience fewer rates of link changes and the throughput for all nodes.

Fig. 10a, shows the average end-to-end delay results of the three techniques in TCP with regards to the packet speed. The obtained result clearly indicates that the proposed DSR-DBR provided the minimal delay which started at 8.03743 ms and ended at 23.0015 ms. This result as compared to DSR-ROUTING-CACHE (8.26899–24.1569) and DSR-ORIGINAL (8.43966–24.8058) is somehow promising. In addition, Fig. 10b,

showed the average end-to-end delay result in UDP in which the proposed technique provided much constant and steady delay when increasing the packet size from 512 sec (0.562737) to 2560 sec (1.80873) as compared to the DSR-DBR (0.583483–1.84563) and DSR-ORIGINAL (0.599576–1.8669). On the other hand, the proposed scheme offered significantly less delays as compared to other schemes. This is because it was able to manage buffering during route discovery and queuing in both TCP and UDP, whereas the processed packets were evaluated based on their delivery ratio caused by data packet buffering during route discovery and queuing. Therefore, it becomes evident that the proposed scheme reduced the delay for processing packets that can be explained by the low velocities and the adverse effect on the routing load and packet delivery ratio where route discovery mechanism can allow multiple paths to be returned when needed.

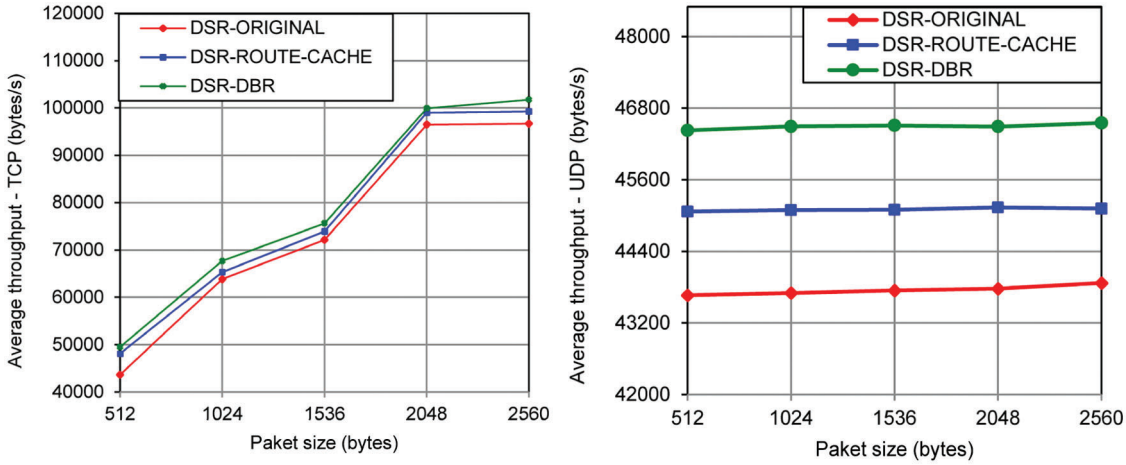

Figure 9: Average throughput versus packet size, (a) TCP, (b) UDP

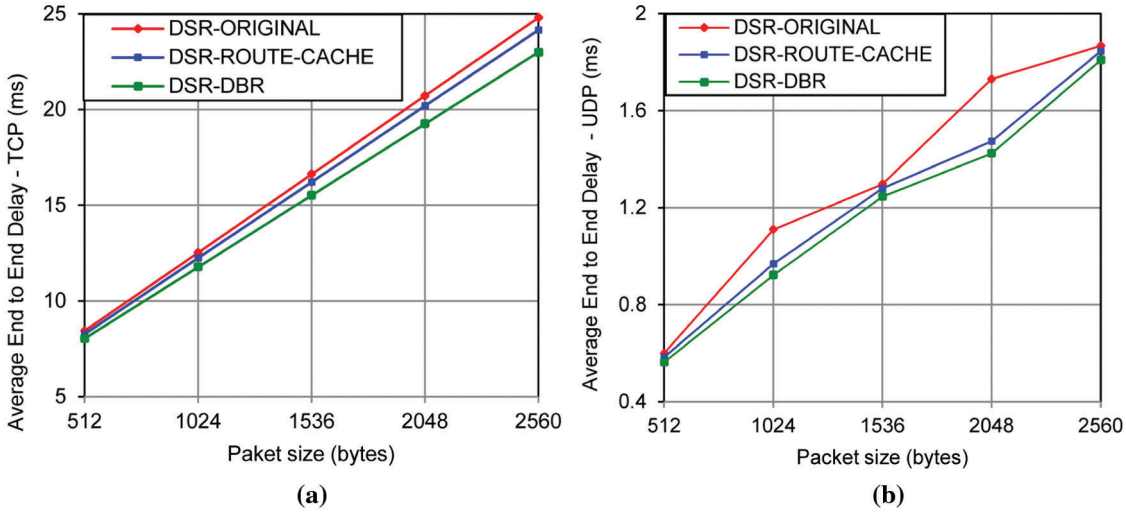

Figure 10: Average E2E delay versus packet size, (a) TCP, (b) UDP

Fig. 11a, shows the routing overhear in TCP for the three techniques with regards to the packet speed. The result showed significant differences among the three techniques in which the routing overhear were in its minimal value when using the proposed DSR-DBR with overhead consumption starting with 60.0017% and ending with 60.0048% as compared to DSR-ROUTING-CACHE (60.0021–60.0061) and DSR-ORIGINAL (60.0026–60.0068). This indicates that DSR-ORIGINAL consumed higher routing overhead when gradually increasing the packet speed, followed by the DSR-ROUTING-CACHE and DSR-DBR respectively.

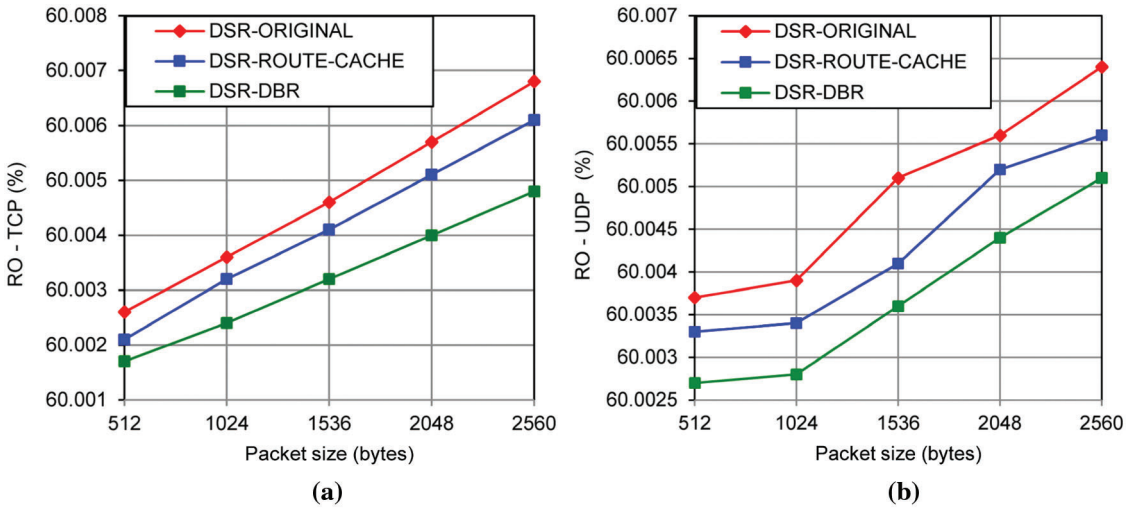

Figure 11: RO versus packet size, (a) TCP, (b) UDP

As for UDP, Fig. 11b, showed the routing overhead result for the three techniques with regards to the packet speed. The result revealed remarkable differences in routing overhead when using the proposed DSR-DBR (60.0027–60.0051) which offered minimal overhead consumption as compared to DSR-ROUTING-CACHE (60.0033–60.0056) and DSR-ORIGINAL (60.0037–60.0064) respectively. From these results, it can be concluded that the proposed DSR-DBR offered a better way of managing the consumption of routing overhead in both TCP and UDP especially when the packet speed increases gradually. It is also assumed that the proposed scheme allowed packets to locate the associated links by constructing or mapping routes to destinations, as they are required. This was achieved by sending packets to the neighbor closest to the packet's ultimate destination in both TCP and UDP. Such process is assumed to significantly enhance the routing overhead with regard to the speed of packet. Hence, the proposed scheme allows packets to learn more routes by adding it to the constant knowledge of the network topology. This is also believed to result in a more reliable routes and less routing overhead due to better maintain routes when send packet increase.

Finally, the RDT results shown in Fig. 12a, for the three techniques revealed substantial outcome for the proposed DSR-DBR against DSR-ROUTING-CACHE and DSR-ORIGINAL. This was deeply looked at by calculating the average RDT in TCP with regards to the packet speed. The result in Fig. 12b, showed that outstanding result for the proposed DSR-DBR with average RDT starting at 5.3770275 ms when packet speed is 512 msec and 12.3842825 when packet speed is 12.3842825. This is significantly better than DSR-ROUTING-CACHE (6.283864–16.9438303) and DSR-ORIGINAL (8.7415884–20.7670072). Furthermore, the proposed scheme facilitates the process of balancing node's necessary information by simplifying the forwarding data packets for other nodes. This is believed to improve the lifetime of routes if unused beyond its limit as it is usually done in other schemes. Meanwhile, the proposed scheme is also

assumed to handle all possible error messaged from buffering route discovery latency and queuing in TCP. In addition, the proposed scheme simplified the detection of nodes that are less effective in the source route.

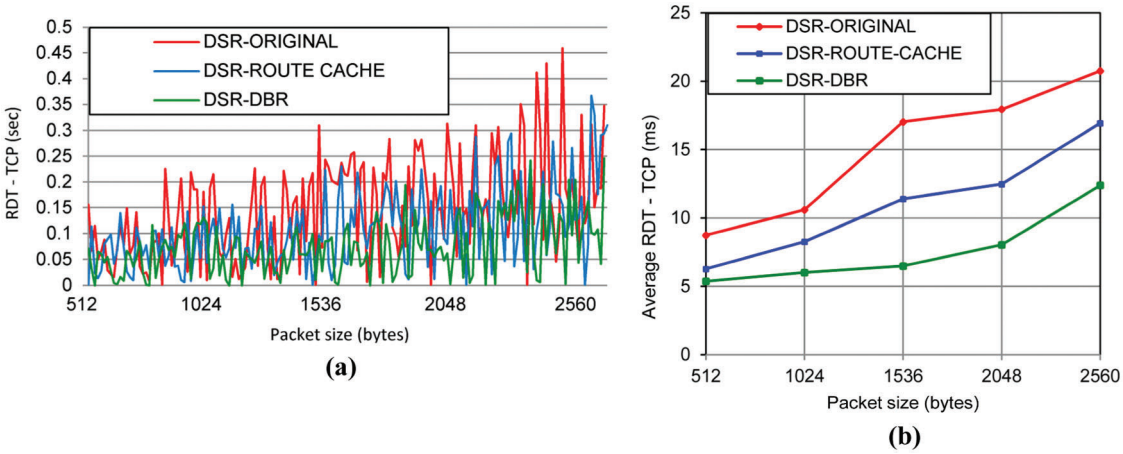

Figure 12: RDT versus packet size, (a) RDT-TCP, (b) average RDT-TCP

As for the UDP, it can be observed from Fig. 13a, that the proposed DSR-DBR offered minimal RDT required for different packet speeds. The result in Fig. 13b, clarified this further in which it showed that the average RDT for DSR-DBR (0.5482818–0.8752383) has a minimal value when comparing it to DSR-ROUTING-CACHE (0.5849467–0.9407948) and DSR-ORIGINAL (0.6369976–0.9919272) respectively. The proposed scheme is believed to provide the essential elements for the node to acquire a new route to the destination which as a result reduced the delay in time especially when waiting for new routes to be determined. In addition, it is assumed that the proposed scheme makes the route discovery time shorter that is an overhead of discovering new routes.

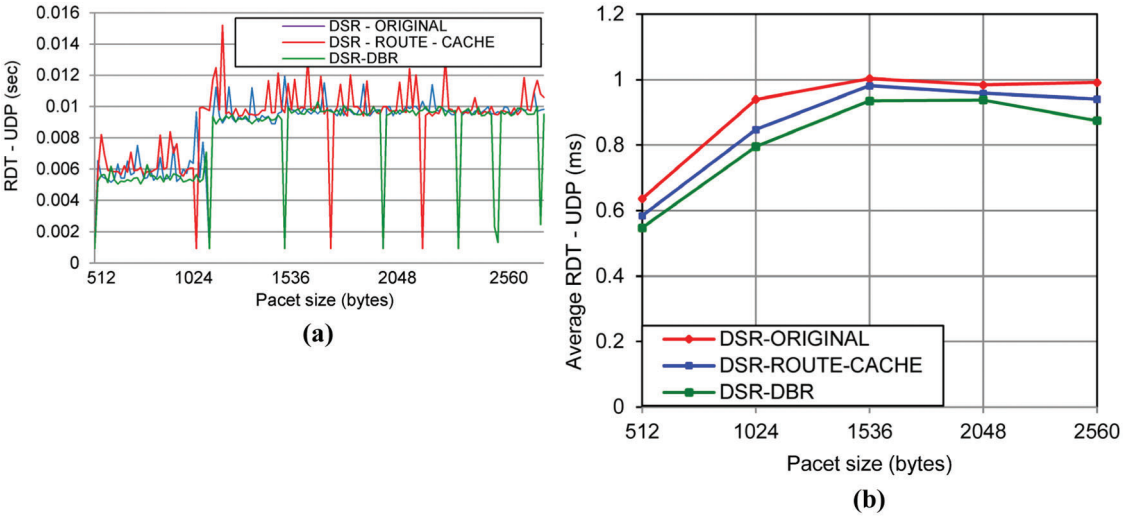

Figure 13: RDT versus packet size, (a) RDT-UDP, (b) average RDT-UDP

7 Conclusion and Future Research

This study investigated the potential of DSR-DBR in providing a better DSR performance to be utilized in multi-hop wireless ad-hoc. The key issued related to the MANET were addressed from the perspective of network performance with regards to the node speed and packet size. NS2 was used to develop and test the proposed DSR-DBR against other techniques like DSR-ROUTING-CACHE and DSR-ORIGINAL. Four performance metrics were used, these are average throughput, an E2E delay, routing overhead, and route discovery time. The comparison result was obtained with regards to node speed and packet size in TCP and UDP. The obtained results revealed that the proposed DSR-DBR to offer higher average throughput result, less E2E delay, less routing overhead, and less route discovery time, compared to both DSR-ROUTING-CACHE and DSR-ORIGINAL respectively.

Despite the effectiveness of the proposed DSR-DBR solution, some limitations need to be mentioned here. The first limitation includes examining certain network metrics without considering the packets dropped by each protocol in which it is because the researcher did not measure data and control packets in order to estimate the fall of packets. In addition, this study was limited to node speed and packet size that can be further adjusted to estimate the performance properly.

Based on the mentioned limitation, future works can be further extended to provide an in-depth understanding of the DSR-DBR potential by examining its performance using different metrics compared to other techniques. In addition, future study can also validate the proposed solution in other protocols like video on demand (VOD) and interactive TV (iTV). Meanwhile, future works can also consider the use of the heterogeneous network in which the sources can be distributed based on the characteristics of the protocols.

Acknowledgement: This research was funded by the Deanship of Scientific Research at Princess Nourah bint Abdulrahman University through the Fast-track Research Funding Program.

Funding Statement: This research was funded by the Deanship of Scientific Research at Princess Nourah bint Abdulrahman University through the Fast-track Research Funding Program.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] N. A. Husieen, O. B. Ghazali, S. Hassan and M. M. Kadhum, "Route cache update mechanisms in DSR protocol: a survey," in *Int. Conf. on Information and Network Technology*, Singapore, pp. 136–41, 2011.

[2] E. M. Royer and C. K. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," *IEEE Personal Communications*, vol. 6, no. 2, pp. 46–55, 1999.

[3] D. -E.-Naya, M. H. Zafar and M. Basheri, "Adaptive expanding ring search based per hop behavior rendition of routing in MANETs," *Computers, Materials & Continua*, vol. 67, no. 1, pp. 1137–1152, 2021.

[4] C. Kumar, G. Kumar and P. Rani, "Efficient-dynamic source routing (E-dSR)," in *Communications and Information Technologies (ISCIT), 2012 Int. Symposium on, Gold Coast*, QLD, Australia, pp. 319–324, 2012.

[5] J. D. Brown, M. Salmanian and T. J. Willink, "Analysis and performance of topology inference in mobile ad hoc networks," in *Int. Conf. on Ad Hoc Networks*, Paris, France, pp. 70–86, 2020.

[6] M. Sivaram, D. Yuvaraj, A. S. Mohammed, V. Manikandan, V. Porkodi *et al.*, "Improved enhanced dbtma with contention-aware admission control to improve the network performance in manets," *Computers, Materials & Continua*, vol. 60, no. 2, pp. 435–454, 2019.

[7] S. Menaka and M. Jayanthi, "Effective stale routes management using preemptive routing in DSR," *World Applied Sciences Journal*, vol. 22, no. 11, pp. 1554–1560, 2013.

[8] S. A. Almazok and B. Bilgehan, "A novel dynamic source routing (DSR) protocol based on minimum execution time scheduling and moth flame optimization (MET-mFO)," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, no. 1, pp. 1–26, 2020.

[9] J. Chen, Y. Tang, D. Fu and H. Chang, "On the improving strategies upon the route cache of DSR in MANETs," in *Int. Conf. on Ubiquitous Intelligence and Computing*, Xi'an, China, pp. 445–458, 2010.

[10] S. Ali, A. Ahmed and M. Raza, "Towards better routing protocols for IoT," in *2019 2nd Int. Conf. on Computing, Mathematics and Engineering Technologies (iCoMET)*, Sindh, Pakistan, pp. 1–5, 2019.

[11] N. Ahmad, S. Sethi and M. Ahmed, "Cache-aware query-broadcast to improve QoS of routing protocols in MANETs," *Wireless Personal Communications*, vol. 113, no. 1, pp. 481–498, 2020.

[12] Q. Liang, T. Lin, F. Wu, F. Zhang and W. Xiong, "A dynamic source routing protocol based on path reliability and link monitoring repair," *Plos One*, vol. 16, no. 5, pp. e0251548, 2021.

[13] H. Alani, M. Abdelhaq and R. Alsaqour, "Dynamic routing discovery scheme for high mobility in mobile ad hoc wireless networks," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 4, pp. 3702–3714, 2020.

[14] S. R. Malwe, N. Taneja and G. Biswas, "Enhancement of DSR and AODV protocols using link availability prediction," *Wireless Personal Communications*, vol. 97, no. 3, pp. 4451–4466, 2017.

[15] W. Lou and Y. Fang, "Predictive caching strategy for on-demand routing protocols in wireless ad hoc networks," *Wireless Networks*, vol. 8, no. 6, pp. 671–679, 2002.

[16] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, T. Imielinski and H. F. Korth (Eds.), Boston, MA: Springer, US, pp. 153–181, 1996.

[17] V. Mandhare, V. Thool and R. Manthalkar, "A novel approach to improve quality of service in MANET using cache update scheme for on-demand protocol," *International Journal of Communication Networks and Distributed Systems*, vol. 18, no. 3–4, pp. 353–370, 2017.

[18] M. K. Marina and S. R. Das, "Performance of route caching strategies in dynamic source routing," in *Proc. 21st Int. Conf. on Distributed Computing Systems Workshops*, Arizona, USA, pp. 425–432, 2001.

[19] X. Yu and Z. Kedem, "A distributed adaptive cache update algorithm for the dynamic source routing protocol," in *Proc. IEEE 24th Annual Joint Conf. of the IEEE Computer and Communications Societies*, Florida, USA, pp. 730–739, 2005.

[20] D. Marandin, "Improvement of link cache performance in dynamic source routing (DSR) protocol by using active packets," in *Int. Conf. on Next Generation Wired/Wireless Networking*, St. Petersburg, Russia, pp. 367–378, 2007.

[21] A. A. Ayoob, N. Sulaiman, M. N. Mohammed and G. M. Abdulsahib, "Reduction the effect of mobility in link transmitting using efficient DSR route cache for MANETs," *Journal of Advances in Computer Networks*, vol. 2, no. 4, pp. 254–260, 2014.

[22] S. Chatterjee and S. Das, "Ant colony optimization based enhanced dynamic source routing algorithm for mobile Ad-hoc network," *Information Sciences*, vol. 295, pp. 67–90, 2015.

[23] C. Pu, S. Lim, J. Chae and B. Jung, "Active detection in mitigating routing misbehavior for MANETs," *Wireless Networks*, vol. 25, no. 4, pp. 1669–1683, 2019.

[24] T. Pino, S. Choudhury and F. Al-Turjman, "Dominating set algorithms for wireless sensor networks survivability," *IEEE Access*, vol. 6, pp. 17527–17532, 2018.

[25] K. Preetha and A. Unnikrishnan, "Improving the routing performance of mobile ad hoc networks using domination set," *Procedia Computer Science*, vol. 46, pp. 1209–1215, 2015.