Reliable Buffering Management Algorithm Support for Multicast Protocol in Mobile Ad-hoc Networks

Tariq Alahdal
Faculty of Computer Science and Information System, Thamar University, Thamar, Republic of Yemen
Email: dr.tariq@ieee.org

[1]Raed Alsaqour, [1]Maha Abdelhaq, [2]Rashid Saeed, and [3]Ola Alsaqour
[1]Faculty of Information Science and Technology, University Kebangsaan Malaysia, Bangi, 43600, Selangor, Malaysia
[2]Faculty of Engineering, Sudan University of Science and Technology (SUST), Khartoum, Sudan
[3]Faculty of Engineering and Technology, University of Jordan, 11942, Amman, Jordan
Email: {raed, maha}@ftsm.ukm.my, [2]eng_rashid@ieee.org, [3]alsukour@gmail.com

Abstract: **Multicasting is one of the relevant issues of communication in infrastructure or centralized administration networks. The reliable delivery of multicast data packets needs feedback from all multicast receivers to indicate whether a retransmission is necessary. A reliable multicast delivery in the wireless Ad-hoc network requires a multicast packet to be received by all multicast receiver nodes. Thus, one or all members need to buffer data packet for possible error recovery. Furthermore, different buffer strategies are essentially used in existing reliable multicast protocols towards support error recovery and reducing buffer overflow. This study proposed two algorithms to improve the performance of the source tree reliable multicast (STRM) protocol. The first algorithm was developed to avoid buffer overflow in the sender node as the forward server (FS) nodes of STRM. This reduction is achieved by managing the buffer of the FS nodes, i.e., selecting the FS nodes depending on its empty buffer size and reducing the feedback sent from the receiver nodes to their FS node. The second algorithm was developed to decrease duplicated packets in the multicast members of the local group, which may be achieved by sending the repair packets to the requesting member. The FS in the local group should create a dynamic and temporary subgroup whose members are only those that requested the repair packet retransmission. The algorithms were tested using detailed discrete event simulation models encompassing messaging systems including error, delay, and mobility models to characterize the performance benefits of the proposed algorithms compared with the existing wireless Ad-hoc network protocols. Several experiments were conducted, revealing numerous results that verify the superior performance of the proposed algorithms over the existing algorithms.**

Index Terms: **Buffer management, ad-hoc networks, forward servers, reliable multicast**

I. Introduction

Mobile ad-hoc network (MANET) is a collection of mobile nodes (MNs) that form dynamic and temporary networks. MNs work in collaborative nature to carry out a given task. They can receive and transmit data packets without using any existing network infrastructure or centralized administration. The reliable multicast delivery in the MANET requires a multicast packet to be received by all multicast receiver nodes. Thus, one or all members need to buffer data packet for possible error recovery. Buffer management algorithms in existing reliable multicast protocols essentially use different strategies to decide the members that should buffer packets for error recovery, and determine how long these packets should be buffered to reduce buffer overflow [1].

Earlier studies have demonstrated that several reliable multicast protocols adopt a distributed error recovery algorithm that allows one or all members to retransmit packets lost by other members. In the tree-based reliable multicast transport protocol (RMTP) [2, 3], a repair server buffers all data packets received in the current multicast session. The protocol was originally designed for multicast file transfer. In this protocol, a repair server buffers the entire file in a secondary storage. The approach is feasible only when the size of the data transmitted in the current session has a reasonable limit. For a long-lived session or setting where repair servers lack space, the amount of buffering can become impractically large. Several reliable multicast protocols use a stability detection algorithm to detect when a message is received by all members in the group, and determine whether the message can safely be discarded [4]. These protocols require group members to periodically exchange information on message history and the set of messages they received. In addition, a failure detection algorithm is needed to provide information to the current group membership, causing high message traffic overhead because the algorithm requires frequent message exchanges.

The source tree reliable multicast (STRM) [5] protocol provides a reliable multicast by constructing a logical tree at the transport layer for local error recovery. The STRM protocol group multicasts receivers into local groups and

doi: 10.12720/jcm.8.2.136-150

selects small sets of receivers, called FS, as error recovery representatives. The FS node should keep in its buffer the packets that were not correctly received from all multicast members in its local group. At the same time, the FS node retransmits the repair packet to all multicast members in its local group using multicast. However, the STRM protocol has two limitations. The first is the ability to efficiently manage the FS node buffer to avoid buffer overflow. When the sender sends new packets, the FS node is directed to drop new packets when its buffer is not empty, thereby preventing such packets from being forwarded. Meanwhile, instructing the FS node to drop old packets prevents these packets from being forwarded for a sufficient amount of time. The second limitation is when the FS node receives a number of requests for a certain packet from its multicast members in its local group. The algorithm will retransmit the requested packet to all group members that belong to its local region. This retransmission causes duplication for members that have correctly received the same packet. These observations were the motivation of the current study.

The two algorithms proposed in this study aims to provide an improved way to discard stable packets from the FS node buffers and decrease duplicated packets at the receiver nodes. In these algorithms, each FS node requires their receiver nodes to transmit both positive and negative acknowledgement to efficiently manage their buffers. Based on the feedback sent by their receiver nodes, the FS nodes forecast the packets that do not require retransmission and remove these particular packets from the buffer. At the same time, the FS node retransmits the requested packet to the receiver nodes that made the request. The results of various simulation experiments, have revealed the potential benefits of the proposed algorithms, including the following: (i) enhancing buffer management through an innovative proposed buffering algorithm that explicitly addresses the variances in delivery latency for multicast group caused by buffer overflow;(ii) decreasing buffer overflow by adaptively enhancing the FS node selection and decreasing the feedback control packets sent from the receiver nodes; (iii) reducing buffer requirements by adaptively enhancing FS node selection to buffer the data packets among members that have larger empty buffer spaces; and (iv) enhancing error recovery by eliminating the duplication of repair packet retransmission in the local groups.

The rest of the article is organized as follows: Section 2 introduces an overview of the mobile ad-hoc networks, buffer management, and relevant studies. Section 3 describes the proposed buffer mechanism. Section 4 presents the simulation scenarios, followed by the aspects to be studied in section 5. Performance evaluation and results are discussed in Section 6. The conclusion is presented in Section 7.

II. BACKGROUND AND RELEVANT STUDIES

A. Mobile Ad-hoc Networks

MANET is a wireless communication that allows nodes to communicate with or without existing infrastructures, as shown in Fig. 1. MANET operates in isolation or requires connection to a fixed network (such as the Internet) through a base station (gateway). MANET lacks the centralized administration or standard support services regularly available on conventional networks [6],[7].

Fig. 1 shows the MANET with six mobile nodes. If node N6 wants to communicate with node N4 in the ad-hoc network, N6 must use a routing protocol to communicate over multiple hops; for example (N6→ N5→N3→N4).

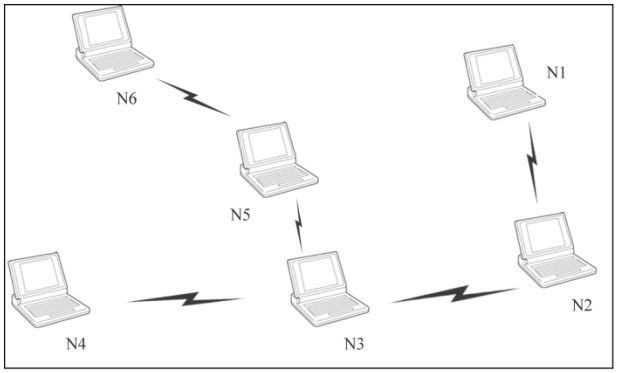

Figure 1. MANET

The nodes that form a MANET are capable of receiving and transmitting packets in an ad-hoc manner without a base station. More importantly, nodes act as an edge device and a router, and are thus able to route packets between the source and destination nodes not within their transmission range. Nodes can be constrained by battery power or processing capabilities. Nodes have varying mobility degrees; they can switch off or move into or out of the range of other nodes in the MANET, thereby changing the MANET topology. Meanwhile, wireless connectivity between nodes is limited by transmission range and signal attenuation due to terrain and interference. Thus, MANETs are characterized by low bandwidth, high error rates, intermittent connectivity, and dynamic topology [8], [6], [9], [10], [11], [12].

Ad-hoc networks employed in several scenarios are particularly useful in dynamic network environments, where network topology continuously changes. These networks are also useful in areas where networking infrastructure implementation is difficult. An increasing number of ad-hoc network applications require a sender to distribute the same data to a large group of receivers. These applications fall in the category of group communication, as opposed to the traditional one-to-one communication [13]. Accordingly, multi-point communication (i.e., multicast) offers the most efficient way to support this application by delivering a single message to multiple recipients [9], [14], [15].

B. Buffer management algorithms and relevant studies

For supporting multicast protocol reliability, the error recovery mechanism is achieved via efficient buffer management design. An efficient buffer management algorithm is an indispensable part of an error recovery mechanism. The existing buffer management algorithms are classified into reducing the buffer usage, flow control and providing packet stability.

1) Reducing the buffer usage

In [16], the researchers proposed an efficient buffering policy where only a small set of receivers buffer the packet to reduce the total buffer requirement amount. Upon receiving a packet, a receiver node determines whether it should buffer the packet using a hash function based on network address and the packet number. Receivers that lost packets use a hash function to select the set of members that have the packet in the buffer and request a packet retransmission. However, the uniform selection methods among different receivers do not consider new members in the system. Thus, scalability is constrained because the latency for error recovery increases with the number of participants.

The randomized reliable multicast protocol (RRMP) [17] is an improvement of the bimodal multicast protocol (BMP) [18]. BMP uses a simple buffer management policy, wherein each member buffers packets for a fixed amount of time after their initial reception before discarding them. By contrast, the buffer space in RRMP is divided into a two-phase buffering policy: a feedback-based short-term buffer and randomized long-term buffer. The members are grouped into local regions formed according to their distance from the sender. A receiver has the member information in its local and parent regions. Every member that receives a packet buffers it for a short period of time to facilitate the retransmission of lost packets in its local region. Only a small random subset of members in each region continues to buffer the packet. In RRMP, a member sends request messages to all members in its local region when it detects a missing packet. However, this process takes a long time for the receiver to search and find the correct repair nodes as the number of participants increase.

In the tree-based reliable multicast transport protocol (RMTP) [2, 19], a repair server buffers all data packets received in the current multicast session. The protocol was originally designed for multicast file transfers. In this protocol, a repair server buffers the entire file in a secondary storage. The approach is feasible only when the size of transmitted data in the current session has a reasonable limit. For a long-lived session or setting with a small buffering space in the repair servers, the buffering amount can become impractically large. A hierarchical tree-based approach is used in RMTP. Receivers are grouped into local regions and a special receiver, called the designated receiver, is assigned in each region. Each designated receiver has knowledge of the members in its local region and the sender. The designated receiver in each local region is responsible for periodically sending acknowledgments to the sender, processing acknowledgment from receivers in its domain, and retransmitting lost packets to the corresponding receivers. Unfortunately, the periodic feedback policy significantly delays error recovery. RMTP stores the whole multicast session data in the secondary memory of the repair nodes for retransmission, making it poorly suited for transfers of large data amounts. Some of these problems were addressed in RMTP-II by the addition of negative acknowledgment (NAKs) [20].

In the reliable multicast protocol for wireless Ad-hoc networks (ReMHoc) protocol [21], the error recovery load is distributed from the sender to all multicast receivers. In ReMHoc, when a receiver detects a lost packet, it waits for a random time determined by its distance from the original data source before sending a repair request. Repair requests are multicast to the whole group similar to regular data packets. This process allows the nearest receiver to retransmit the packet by multicasting. All receivers in the ReMHoc protocol must keep all the packets in their buffer for retransmission. In this process, the sender cannot detect or does not know the safe time to discard an already sent packet from the buffer. Similarly, this scheme is not well-scaled because the requests sent by each receiver for each incorrectly delivered packet can lead to a request implosion at the sender node.

A stepwise probabilistic buffering algorithm based on epidemic algorithms was proposed in [22] to provide scalability and reliability. The stepwise probabilistic buffering reduces the amount of buffering by distributing the buffering load to the entire system where every node does not have the complete view of the entire receiver group. In every receiver group, all peers only have partial knowledge of the participants. In this algorithm, only a small subset of the node keeps a data message in its long-term buffer. The long-term buffers are determined through a stepwise probability search algorithm inspired by the random forwarding encountered in epidemic algorithms [18], [23]. A node is susceptible when it has not yet received an update. The node implements the anti-entropy algorithm to recover missing update messages.

This algorithm provides probabilistic warranties that sent operations will be divided to all connected nodes. The anti-entropy algorithm randomly selects a neighbor in the local table of neighbors and sends a digest of received messages. In the anti-entropy process, non-faulty nodes are always either susceptible or infectious. In this algorithm, each node periodically picks f (fan-out) other nodes at random and exchanges its state information with the selected one. The algorithm uses a pull-based approach, where data dissemination is triggered by susceptible nodes when they are picked as gossip destinations by infectious nodes, to spread information.

Each node in the stepwise probabilistic buffering algorithm periodically selects f random nodes from its partial view and sends a digest including the recent message history. The node digest contains the state information for the last d messages the node has received so far and their bufferer identifiers. Upon receiving a digest, a node can determine the lacking messages and

can request them from the bufferers indicated in the digest for retransmission. If a bufferer crashed or cannot retransmit the message, the request can be forwarded to another bufferer. The source sends buffering request messages to randomly selected *b* nodes in its partial view to determine the bufferers of a data message. Parameter *b* is the number of bufferers per message. For a data message, the bufferer nodes are determined simultaneously if b > 1. Buffer fullness (BF) node ratio is the ratio of the number of messages stored in the node buffer to its long-term buffer capacity. Steps-to-live (STL) value attached to a buffering request message indicates the maximum number of times that request messages can be forwarded among nodes. When a node receives a buffering request message for a particular data, it accepts the request with probability (1 – BF). Otherwise, the node forwards the message to a randomly selected node from its partial view with a probability equal to BF. For example, if 90% of the long-term buffer is full, the node becomes the message bufferer with a probability of 0.1, and then sends the buffering request to one of its neighbors with a probability of 0.9.

A decision trade-off exists for the STL value of the bufferer request messages. Uniform selection of bufferers is easily achieved when the STL value chosen is large enough because the request message can visit more nodes and find a suitable buffer place. However, frequent exchange of history messages to determine the bufferers of a data message causes high traffic, resulting to higher delays because of the bufferer determination rounds.

2) Network flow control

Flow control is an adaptive mechanism that deals with varying resources such as CPU speed and receiver node bandwidth. Buffer optimization techniques in this category adjust the network rate to minimize buffer overflows at the receiver nodes.

In [24], the researchers investigated the effects of buffering rate and flow control in several acknowledgement (ACK)-based and NAK-based reliable multicast protocols. Most of the rate-based multicast protocols remain equally vague on that issue because the absence of NAK from a receiver for a given packet is not a definite indication that the receiver received the packet. In other major studies, [25] and [26] proposed a retransmission control scheme for NAK-based multicast protocols. Their schemes require the sender to reduce its transmission rate upon receiving NAKs for several of its packets. The sender keeps a log of its past transmission rate to prevent excessive transmission rate decrease. This scheme was efficient; however it minimizes buffer overflow occurrences rather than eliminating them, as that of a sliding window protocol.

A different idea explored in [27] requires every process to calculate the average buffer capacity among all communicated processes, and then transmit the information. When the rate is too high, with respect to the average, the process locally reduces the rate. Indirectly, the information sources get such feedback and reduce the information production rate. However, the information production rate is adjusted according to the process with the smallest buffer space.

3) Packet stability

A packet is stable when it is delivered to all group members. Buffer management approaches that explicitly take stability into account exist. In [4], the researchers proposed a stability detection algorithm for discarding safe packets from the buffers. The members are partitioned into groups, and every node is included in the error recovery. This process is achieved by letting the receivers periodically exchange history information about the received sets of packets. Eventually, one receiver in the group becomes aware that all the receivers in the group successfully received the packet and announces this to all the group members. Moreover, all members can safely discard the packet from the buffer. However, this algorithm causes high message traffic due to the frequent exchange of history messages.

The search party protocol [28] is another protocol where timer contribution helps to discard packets from the buffers. All the members discard packets after a fixed amount of time to achieve stability. However, the protocol remains vague on the problem of selecting the proper time interval for discarding packets. A heuristic buffer management method based on ACKs and NAKs is proposed in [29] to provide scalability and reliability. In every group of receivers, one or more members possess higher error rates than the other members. These nodes are those with the least reliable and slowest links. The idea of this method is that when a message is correctly received by the nodes, it has been probably received by all of the other nodes. Thus, repair nodes that buffer the message can discard it.

Network friendly epidemic multicast [30] combines a standard epidemic protocol with a novel buffering technique that combines different selection techniques for discarding messages in case of a buffer overflow. The used selection strategies are random purging, age-based purging, and semantic purging. Random purging refers to the random discarding of an item from the buffer. Age-based purging is simply discarding the oldest message, whereas semantic purging means discarding a message recognized as obsolete. Obsolescence relation is determined by the application.

The least recently used (LRU) buffer replacement scheme is considered in [31] for epidemic information dissemination. In the LRU scheme, a new message is placed on the first position and the message at the rear is discarded. However, when a request arrives for a message in the buffer, that message is placed on the first position by moving the front items one position down. The least used item stays at the rear of the stack and is possibly next to be discarded.

The aforementioned buffer management algorithms indicate that several existing approaches are not sufficient to guarantee an efficient buffer management. This result could be caused either by the fact that uniform selection method among different receivers does not consider new members in the system, or the receiver spends too much time searching and finding the correct repair nodes as the

number of participants increase, or the periodic feedback policy significantly delays error recovery. Furthermore, high message traffic due to frequent exchange of history messages can increase buffer overflow and network overhead. Moreover, the buffer messages are discarded by the repair nodes because if these messages were correctly received by these nodes, it has probably been received by all of the other nodes. By contrast, the proposed buffering mechanism appends the silent feature technique, which gains superior approach performance along with inferior buffer overflow. In addition, network overhead directs towards performance improvements to satisfy the Quality of Service (QoS) requirements for best-effort and real-time applications.

III. THE PROPOSED BUFFER MANAGEMENT ALGORITHM

This section provides a detailed description of the proposed ordered ACK (OACK) buffer management algorithm, along with consideration of its techniques, in enhancing the local error recovery using sub sub-casting and the improved of sub sub-casting algorithms.

A. Ordered ACK (OACK) buffer management

As described in [5], the STRM protocol distributes the responsibility of error recovery among the selected set of its one hop neighbor nodes. The sender ensures that all directly FS nodes connected, correctly received certain packet numbers by checking the ACK received from its FS nodes. In the next sending interval, the sender sends a number of new packets equal to those correctly received. Utilizing the method of computing, the available window can cause an overflow at the buffer of one or more FS nodes because the sender does not know the FS node situation. This problem occurs because each FS must keep all incorrectly received packets by the buffer of its receivers. As a result, the FS cannot retransmit all the retransmission requests from its receiver nodes. Consequently, the sender node workload increases, thereby causing more error recovery delays. An OACK buffer management algorithm adapted by STRM was proposed in this study to effectively reduce buffer requirement. The next section describes the OACK buffer management algorithm.

1) Description of the OACK algorithm

The OACK algorithm was designed to improve FS node selection. The algorithm reduces buffer usage because only a small subset of nodes was chosen as buffer for each message. Furthermore, this algorithm is applicable to large scale scenarios, provides reliable delivery, and is adaptable to dynamic addition and separation in the network. Determining the FS nodes is initiated by the sender. Upon determining the FS nodes, their IDs are piggybacked to the data packet and initially sent to the FS nodes.

The selection forward server process (SFSP) algorithm [5] is enhanced by allowing the nodes located in the first hop from the sender to be selected as FS nodes based on forwarding utility (U_f), as presented in Eq. (1). The forwarding utility (U_f) is an integrated function of a node's remaining buffer space utility (U_b), as presented in Eq. (2), and a neighbor utility (U_n), as presented in Eq. (3). Thus, a buffer awareness element is introduced into the selection. This element is important because the nodes selected as FS nodes must have sufficient buffer size to buffer the data packets for retransmitting requested packets.

$$U_f = U_b * U_n \tag{1}$$

$$U_b(i) = \frac{1}{1 + e^{-B_i + s}} \tag{2}$$

$$U_n(i) = \frac{UThN_i}{TThN_i} \tag{3}$$

The buffer utility (U_b) in Eq. (2) specifies the remaining node buffer space. A sigmoid function is used to determine the utility because it provides a good estimate of the required behavior, low utility, and slow change at low buffer; sharp change in utility at medium buffer; and high utility and slow change at high buffer. B_i is the remaining internal node buffer space and is mapped in the sigmoid function. s is defined as half the value of the full node buffer space to shift the sigmoid function to obtain the positive value. For (U_n) in Eq. 3, the $UThN_i$ is the unallocated two hop neighbors of node i, and $TThN_i$ is the total two hop neighbors of node i. The neighbor utility (U_n) for node i is equal to the number of unallocated nodes in the two hop pool neighboring node i divided by the total number of node i neighbors.

2) Example and explanation of the OACK algorithm

Fig. 2 shows an example of the OACK algorithm to select FS nodes that have a suitable buffer size with the following neighbor arrangement: N(I)={J,K,L}, N(J)={I,M,N}, N(K)={I,M,N,O,P}, N(L)={I,O,P,Q}. The sender node I calculates a pool of one hop neighbors H_1={J,K,L} and two hop neighbors H_2={M,N,O,P,Q}. Node L has a unique neighbor (node Q) not reachable from any other possible one hop nodes (nodes J or K), as shown in Fig. 2(a). Therefore, node L is added to the FS set, FSL_1={L}. The neighboring nodes of node L are removed from H_2, resulting in H_2={M,N} and H_1={J,K}. When no more nodes have unique neighbors, sender node I calculates the forwarding utilities for the remaining nodes in H_1, as shown in Fig. 2(b).

Node J's forwarding utility is calculated as follows:

$$B_J = 7$$

$$s = \frac{Buffer\ size}{2} = \frac{10}{2} = 5$$

$$U_B(J) = \frac{1}{1 + e^{-B_J + s}} = 0.87$$

$$U_n(J) = \frac{2}{2} = 1$$

$$U_f(J) = U_n(J) \times U_B(J) = 0.87$$

Node *K*'s forwarding utility is calculated as follows:

$$B_K = 2$$
$$s = \frac{Buffer\ size}{2} = \frac{10}{2} = 5$$
$$U_B(K) = \frac{1}{1 + e^{-B_K + s}} = 0.05$$
$$U_n(K) = \frac{2}{4} = 0.05$$
$$U_f(K) = U_n(K) \times U_B(K) = 0.025$$

Node *J* has higher forwarding utility than node *K*; thus, the remaining nodes in H_2 are allocated to node J, resulting in SFL_1={J,L}, as shown in Fig. 2(c).

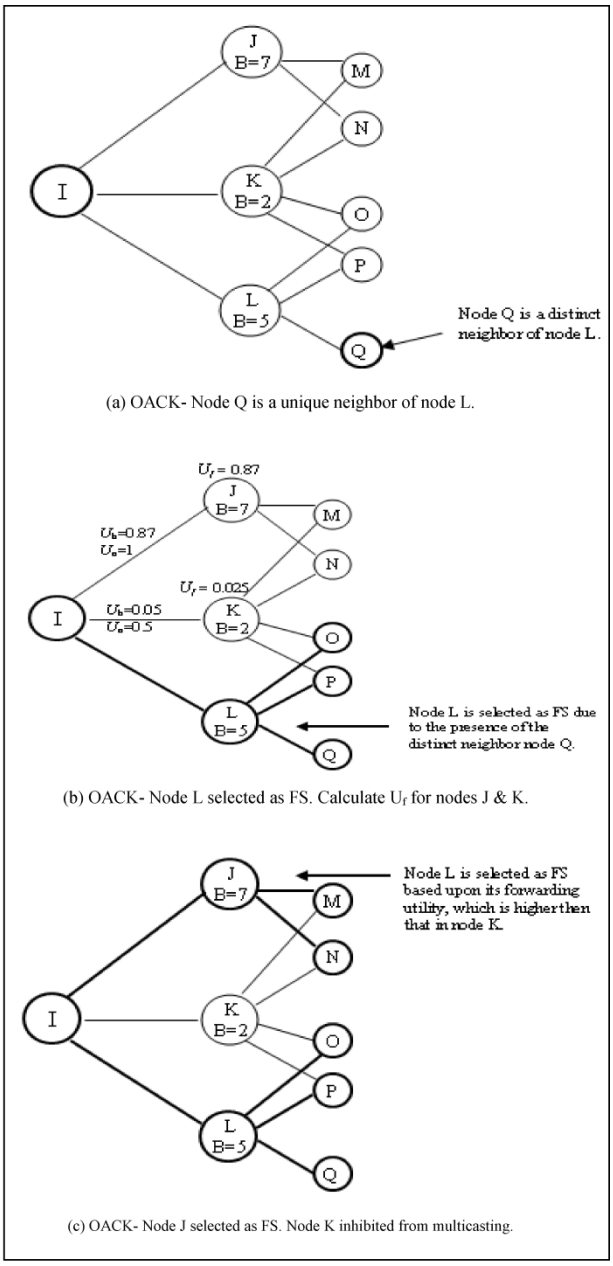

Figure 2. Example of the Ordered ACK to select FS nodes

3) The OACK flow control scheme

The OACK algorithm performs the flow control using the transmission and ACK window. The sender node maintains a fixed transmission window size. The sender node periodically multicasts the data packet window to the neighbor nodes located in its first hop. Similarly, the selected FS nodes multicast the data packet window to the receiver nodes located in its local group. In addition, the receiver node maintains an ACK window and periodically sends this window to its FS node. An ACK window carries two fields: <*N*,*L*>; where *N* is the less sequence number received and *L* is the last packet sequence number received in the transmission window. The *L* packet acts as an implicit ACK for the transmission received packet window. The FS node can now safely discard up to the packet number *L* from its buffer. The receiver nodes carry out this procedure for each of the received transmission windows.

When a receiver node *i* detects lost packets, it sends a NAK to its FS node. This NAK contains the sequence number of the last packet, similar to when all packets had been correctly received. For instance, assume that the sender's window size is 10 packets and the first ACK by node *i* is supposed to acknowledge packets 10, 20, 30, and so on. Therefore, if *i* does not receive packet 19, it sends a NAK informing that packet 19 was not received and that packet 18 is the last packet, similar to when all packets had been correctly received by node *i*. Node *i* continues the same sequence of ACKs regardless of the lost packet. Hence, the next ACK sent by node *i* only acknowledge the correct reception of packets 19 and 20. This process ensures that the FS node receives at least one ACK packet from the receiver node *i* in every sending window.

The mechanism to advance the transmission window is to have all FS nodes send an ACK to the sender node. This ACK contains the highest-numbered packet P_j that the FS node *j* can safely discard based on the ACK sent by its receiver nodes. When some packets are not ACK, but are still in the buffer of the FS nodes, different FS nodes would have different buffer sizes. The sender node can then initially send new packets equal to the available and sufficient space in the buffer of all FS nodes. Let P_j be the last packet acknowledged by *j*; hence, the highest-numbered packet P_H acknowledged by all FS nodes is given, where N^{FS} is the number of FS nodes. When all FS nodes have acknowledged P_H, the sender can now safely send the packets up to P_H. The summarized OACK flow control algorithm for each type of nodes, that is, sender, receiver, and FS, is given in Figs. 3, 4, and 5, respectively.

Algorithm Sender OACK()

1. $FS_i = 0$; for all $1 \le i \le N^{FS}$
2. $Pj = 0$; for all $1 \le j \le N^{PKT}$
3. **multicast** window of pkts
4. S = size of sent window
5. **while** (true)
6. **if** (ACK for pkt$P_j$from FS_i arrives) {
7. **select**P_H = min { $Pj \mid 1 \le j \le N_j^{FS}$ }
8. **multicast**pkts from S+1 to $S + P_H$
9. $S = S + P_H$}
10. **endif**
11. **if** (NAK for pkt$P_j$from FS_i arrives) {
12. **remulticast**pktP_j
13. **remove** up to pktP_{j-1} from buffer}
14. **endif**
15. **endwhile**

Figure 3. OACK algorithm of the sender

Algorithm Receiver_OACK(FS_i,P_j)

1. count = 0
2. **while** (true)
3. **if** (pktP_j from sender arrives) {
4. **if** (pktP_jNOT received before) {
5. **increment** count
6. **if** (count ==size of sent window) {
7. **unicast** ACK for count to FS_i node}
8. **endif**}
9. **endif**}
10. **endif**
11. **if** (missing pktP_j detect) {
12. **unicast** NAK pktP_j to FS_i}
13. **endif**
14. **endwhile**

Figure 4. OACK algorithm of the receiver

Algorithm FS_OACK(FS_i,P_j,R_k)

1. count = 0
2. **while** (true)
3. **if** (pktP_j from sender arrives) {
4. **if** (pktP_jNOT received before) {
5. **store**pktP_j in buffer
6. **increment** count
7. **endif**}
8. **endif**}
9. **if** (ACK pktP_j from receiver R_k arrives) {
10. P = min { $P_j \mid 1 \le j \le N_j^K$ }
11. **remove** up to pkt$P$ from its buffer
12. **if** (count ==size of sent window)
13. **unicast** ACK for P to sender node
14. **endif**}
15. **endif**
16. **if** (NAK pktP_j from receiver R_k arrives) {
17. **remulticast**pktP_j
18. P = min { $P_j \mid 1 \le j \le N_j^K$ }
19. **remove** up to pkt$P$ from its buffer
20. **if** (count == size of sent window) {
21. **unicast** ACK for P to sender node}
22. **endif**}
23. **endif**
24. **endwhile**

Figure 5. OACK algorithm of the FS

4) Stepwise Probabilistic Algorithm

OACK algorithm selects a subset of receiver nodes to serve as buffers by enhancing the SFSP algorithm, elaborated in [5] with the buffer size utility. In the stepwise probabilistic algorithm [22], the set of long-term buffers are chosen randomly from all receiver nodes. Every node sends buffering requests randomly to one of its neighbors with a probability equal to each node's buffer fullness ratio. The algorithm provides a fairly uniform distribution, where every node does not have the complete view of the entire group of receivers. The probabilistic algorithm works when the number of generated messages is lower than the total long-term buffer capacity of the system. When the long-term buffers of the nodes become full, and if a member receives a buffering request message, the node directly sends a buffering request to one of its neighbors, and the receiving neighbor performs the same process again. Therefore, the buffering request is forwarded peer-by-peer until the STL value expires. The nodes on which the STL value expires would buffer the corresponding message. The buffering load distribution uniformity of the probabilistic algorithm is observed only when the number of generated messages approaches the total long-term buffer capacity of the system.

The algorithm uses a probabilistic scheme that works on every node to determine the bufferers of a message. Every node sends the buffering requests randomly to one of its neighbors with a probability equal to the node's buffer fullness ratio. The algorithm provides a fairly uniform distribution, where every node does not have the complete view of the entire group of receivers.

The algorithm does not provide uniformity before the nodes reach their long-term buffer sizes, similar to the case when the number of generated messages exceeds the total long-term buffer capacity. In the algorithm, assuming that all buffers are initially empty, neighbors in the first hop of the sender accept the buffering requests with higher probability than the farther nodes. Thus, the buffer levels of the nodes close to the sender node are much higher, whereas those of the ones far from the sender are approximately zero. This algorithm is implemented in the STRM protocol.

The next section explains the enhancement to the STRM protocol to avoid duplication caused by a retransmission of lost packets in the FS node local group.

B. Enhancement of the Local Error Recovery

The second proposed enhancement for STRM, elaborated in the [5], is the local error recovery. STRM allocates FS nodes in each local region and makes them responsible for error recovery among all the other receivers in the same region. These FS nodes retransmit lost packets to all group members belonging to each node's local region. Therefore, this retransmission causes duplication for members that have already received the same packet correctly. This section proposes two algorithms for error recovery to avoid duplication for multicast data deliveries. The presented algorithms can be applied to tree-based multicast protocols [32]. Furthermore, dedicated support is provided for protocols

that organize their group members using local group multicast and whose leaders are connected with the sender node. The next section proposes the enhancements to the local error recovery for the STRM protocol using Sub Sub-Casting algorithm (SSC) that avoids duplication of retransmitted packets in local group members.

1) Sub Sub-Casting Algorithm

The SSC algorithm is used for lost packet recovery, where the multicast tree is partitioned into local groups. This algorithm is based on the NAK packets received from the local receiver nodes to the FS nodes. Each FS node acts as the local deliverer that provides a recovery service to the nodes of the local group. In this algorithm, each FS node is required to retransmit lost packets only to the effected receivers that have requested the packets.

The SSC is an algorithm where the FS nodes send data packets and receive the ACK/NAK packets from its local group nodes, which act as a sub-group of the original multicast group. In this local group, the FS node multicasts the requested lost packet only to the affected receiver nodes located in the FS node local group when the number of the affected receiver nodes requesting for a certain packet is greater than a specific threshold. This mechanism allows affected receiver nodes to recover from the same packet loss with only one retransmission from the FS node.

Multicasting the lost packet only to the receivers that request that packet decreases the number of duplicated packets, thereby, reducing the number of packets transmitted through the network. This reduction will in turn increase the performance of the system. Subsequently, this mechanism requires the FS node to create a temporary dynamic multicast group, whose members are only those that have requested the lost packet, as shown in Fig. 6.

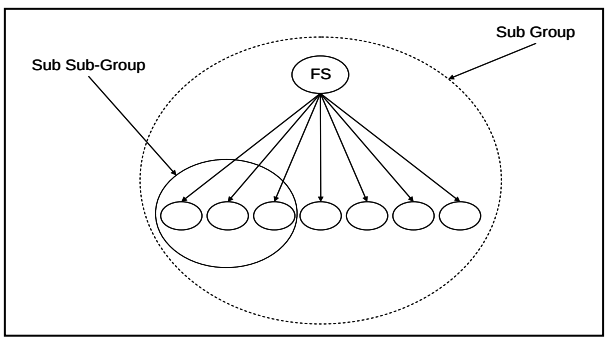

Figure 6. An example of sub sub-casting tree

The created temporary dynamic multicast group is a sub-group of the local group that is also a sub-group of the global multicast group. Thus, this group is a sub-group of another sub-group, denoted as sub sub-casting, as shown in Fig. 7.

Algorithm SubSub-Casting (THRESHOLD, P)

```
1.   Request_i = nodes that request the pktP
2.   if (Request_i ≥ THRESHOLD) {
3.     for all Request_i in FS_j; for all 1 ≤ j ≤ N^FS
4.       if (Request_i Status == 0) {// the pkt in the buffer
5.         multicast (Request_i(P))}
6.       endif}
7.   endif}
8.   else {
9.       for all Request_i in FS_j; for all 1 ≤ j ≤ N^FS
10.        if (Request_i Status == 0) {
11.          unicast (Request_i(P))}
12.        endif}}
```

Figure 7. Sub sub-casting algorithm

Any receiver that detects any lost packet is required to send a NAK to its FS node to build the sub sub-casting group. The FS node registers the sequence number of the lost packets and the ID of the receiver node that requests the lost packet in the retransmission queue. Upon retransmission, the FS node knows which receiver has requested for the lost packet. If the number of the receiver is greater than the multicast threshold, a temporary multicast group, consisting of the receiver nodes that have requested for the lost packet, is, therefore, created. Meanwhile, if the number of receiver nodes requesting for the packet is smaller than the multicast threshold, the packet is then unicast to each requesting receiver, as shown in the SSC algorithm in Fig. 7. The SSC algorithm cannot be implemented in protocols that use the NAK-based with NAK suppression because the idea of this algorithm is to reduce the number of receivers sending NAK packets to the sender node. Therefore, the SSC algorithm requires each receiver node to send a separate NAK packet to the FS nodes for the FS node to identify all the receiver nodes that need the lost packet.

The next section proposes improvements to the SSC algorithm to improve its performance in WLANs and reliability in MANETs. The improved algorithm, Improved Sub Sub-Casting Algorithm (SSC-I), is implemented to cause lower delay compared with the SSC algorithm.

2) Improved Sub Sub-Casting Algorithm

The SSC algorithm attempts to avoid duplication packets by creating a temporary dynamic multicast group. The FS nodes retransmit the lost packets only to the newly created sub-group. However, this solution is ineffective when the receiver node requesting a retransmission of the lost packet is distant from its FS node. Fig. 8 illustrates this case; the forward nodes located between the FS node and the requesting node receives the retransmitted packet again to forward the packet to the requesting node *R*.

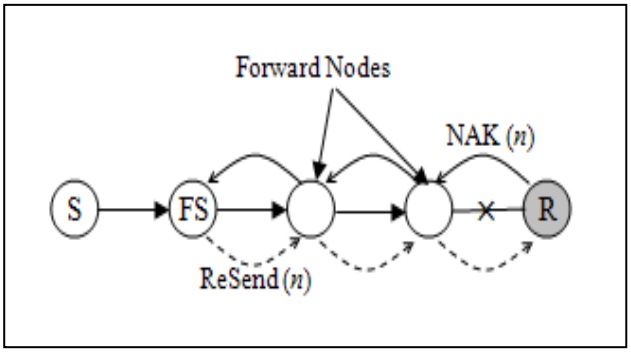

Figure 8. Request and repair in the SSC algorithm

The ACK packets in SSC-I are sent upstream to the FS nodes at regular intervals, but NAK packets are sent in an event-driven manner to reduce the end-to-end delay. Packet losses are identified through gaps in the sequence numbers of the received packets. In SSC-I, the affected node *R* is required to send a retransmission request for each of the missing packet to recover from such packet losses. If *R* does not receive the lost packet before the ACK time interval, *R* generates a retransmission request to its FS node directly. The FS node then unicasts the lost packet to *R*.

Fig. 9 shows that data paths are specific to each FS node. Efficiently recovering from packet losses necessitates that node *R* selects the retransmission request of the lost packet from the node that is an upstream node on a specific FS's data path. According to Fig. 9, node *R* requests retransmissions of packets sent by the FS node from node *B*. Node *B* is the forwarding node to node *R* and sends the lost packet to node *R* directly.

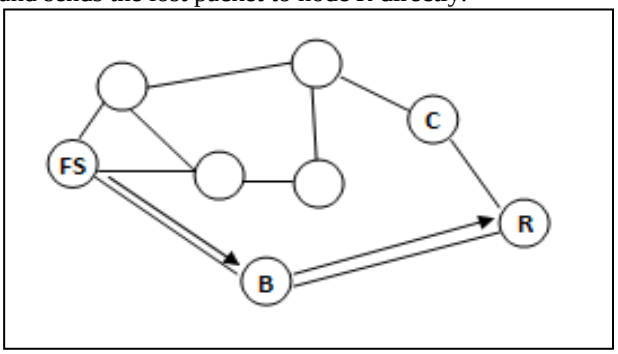

Figure 9. Specific forward data path in SSC-I

IV. Simulation Setup

A. Simulation Environment

The simulation tool was developed for an ad-hoc network environment to build a reliable multicast transport protocol that allows a single sender to deliver packets in an ordered manner to a group of multicast receivers. Furthermore, given the limited development resources in simulator tools, a discrete-event simulation tool for MANET environment was developed in Visual C++. This tool analyzes the performance of both the existing and proposed algorithms.

Fig. 10 depicts the block diagram of the developed simulation tool. The figure shows the data and control flows inside a component of the simulation tool. The component can represent a sender or receivers. The data flow is depicted in a regular dark line, whereas the control flow is depicted in a dashed line. Two separate buffers for transmission and retransmission are presented. Similarly, separate transmission mechanisms are presented for multicast and unicast. Separate entities are shown to manage the control messages for group management and flow control. In addition, separate buffers are provided for ACK and NAK. The block diagram contains the algorithms that improve the multicast, such as SFSP, buffer management, and local error recovery algorithms. Two main components are depicted in the simulation design: sender node and receiver nodes.

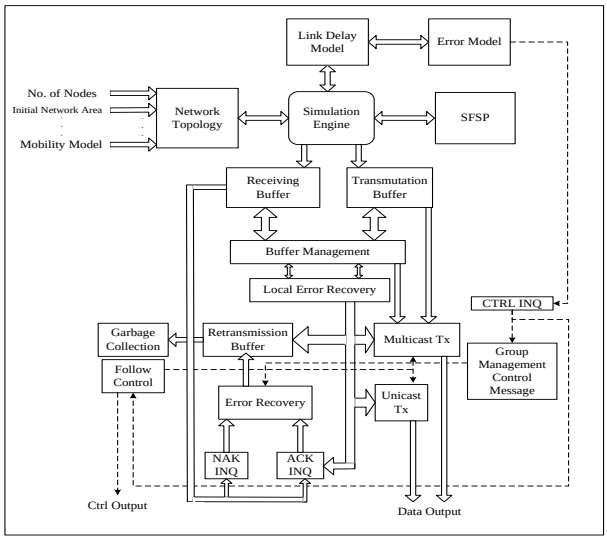

Figure 10. Block diagram of the simulation tool

In this study, all nodes and links were assumed to work properly and none of them fails during simulation. The simulation program was run ten times with the same input variables. The results were taken as the average of these iterations to obtain accurate results. Furthermore, the MANET topology model was randomly generated for a fixed number of nodes. The location of each node was assumed to be on the two-dimensional coordinate system (x,y), and each location is generated randomly in a uniform distribution. The variable inputs needed to create the random topology are as follows: first, the fixed number of nodes in the network; second, the initial network area to begin with; third, the mobility model and the node speeds; and finally, the unique address or ID of every node.

In addition, the transmission range was 250 m, typical for mobile devices. The scenario area of the simulation was limited to 700 m × 700 m to reduce the likelihood of no connectivity when the nodes are allowed to go beyond the area. This consideration was based on the session size. Radio irregularity factors were not considered in these simulations. Therefore, if node *i* can send a message to node *j*, node *j* can also send a message to node *i*. The parameter settings and values used in the simulation are shown in Table 1.

TABLE I.
SIMULATION PARAMETERS

Description	Value
PACKET_SIZE	512 bytes
NO_NODE	100
SESSION_SIZE	Variable
LINK_BW	2 mbps
TRANSMISSION_RANGE	250
AREA_SIZE	700x700 m^2
MESSAGE_SIZE	2000 packets
LOSS_RATIO	0.1
NODE_SPEED	0, 5, 10, 15, 20, 25 m/s
QUEUING_DELAY	20–80 ms
PROPAGATION_DELAY	10 ms
PAUSE_TIME	1, 2, 5, 10, 15, 20 s
HELLO_TIME	1 s
ROUTE_TABLE_TIMEOUT	960 ms
MEMBER_TABLE_REFRESH	400 ms
MEMBER_TABLE _TIMEOUT	960 ms
ROUTE_TABLE_REFRESH	160 ms
FG_TIMEOUT	480 ms

B. Performance Metrics

The following performance metrics were used to evaluate the efficiency and effectiveness of the algorithms. Duplicate request and retransmitted messages were taken into account in these measurements.

- **Buffering load:** Buffering load is the total number of packets buffered by each node when a specific sender node is chosen as the message sender.
- **Percentage of request packets:** Percentage of request packets is the ratio of the number of requests for lost packets transmitted by receiver nodes to the total number of the original data packets transmitted by the sender.
- **Average retransmission packets:** Average retransmission packets are the ratio between the number of retransmission packets transmitted by the sender and forwarding nodes and the total number of the original data packets transmitted by the sender.
- **Average end-to-end delay:** Average end-to-end delay is calculated as the average difference between the time each data packet is transmitted by the sender and the time the packet is received by the receiver nodes, and then averaged over the total number of receiver nodes.
- **Average recovery latency:** Average recovery latency is the average difference between the time at which a receiver node detects each missing packet and the time at which the missing packet is recovered at that receiver node, and then averaged over the total number of receiver nodes.
- **Percentage of duplicate packets:** Percentage of duplicate packets is the ratio between the total retransmitted packets that are duplicates received by each receiver node to the total number of original data packets transmitted by the sender. This ratio is then averaged over the total number of receiver nodes.

V. PERFORMANCE EVALUATION AND RESULT DISCUSSION

This section presents and discusses the performance of the OACK, SSC, and SSC-I algorithms. The constant simulation parameters shown in Table 1 are used throughout the simulation.

A MANET topology is randomly generated for 100 nodes. The location (*x*,*y*) of each node was generated randomly in a uniform distribution. The buffer capacity of each node is 20 packets, and 2000 packets are broadcasted from a single sender. The sender and receiver nodes were randomly selected. The data packet window was periodically sent by the sender to the receiver nodes. The STRM protocol groups receiver nodes into local regions that provide a high degree of reliability and avoid the Feedback Implosion Problem, but suffer from buffer overflow and duplication of received packets. The OACK, SSC, and SSC-I algorithms are enhancements to the STRM protocol.

First, the experimental results of the OACK buffering algorithm are presented. Its comparison with the Stepwise Probabilistic buffering and STRM protocol in terms of distributing the buffering load and data dissemination metrics are also exhibited. The Stepwise Probabilistic buffering utilizes the selected FS nodes in STRM as long-term buffers. The STL value is set as 10 hops, gossip interval is 200 ms, which is equal to the *Hello_Time* interval. The First In First Out policy was implemented for all algorithms when the node buffer as full. Furthermore, the experimental results of the SSC and SSC-I algorithms and its comparison with the STRM protocol in terms of data dissemination metrics are given. Hence, the following sections present the effects of various session sizes and different mobility speeds on all algorithms.

A. Buffering Load

Fig. 11 shows the total number of packets buffered by each node when a sender node, ID 0, is chosen to be the message sender. In the experiments, the number of receiver nodes is 100 and the maximum movement speed of the nodes is 5 m/s. The probabilistic algorithm utilizes the selected FS nodes in STRM as long-term buffer and provides these nodes higher probability to buffer the data packets for retransmitted requested packets compared with other nodes, which has the same low probabilities. This mechanism leads to the buffer of larger number of packets in the nodes close to the sender; the probability to buffer in these nodes is increased. The OACK selects the FS nodes that have a sufficient buffer size among the suitable nodes in the first hop from the sender. The OACK exhibits better usage of the buffer load than STRM and the probabilistic algorithm.

Fig. 11 shows that in all algorithms, some nodes buffer zero or fewer packets, which may be attributed to either a relatively higher number of nodes close to the sender in the first hop or the nodes located far from the sender during the entire simulation. STRM exhibits a significantly higher buffering load compared with OACK and the probabilistic algorithms.

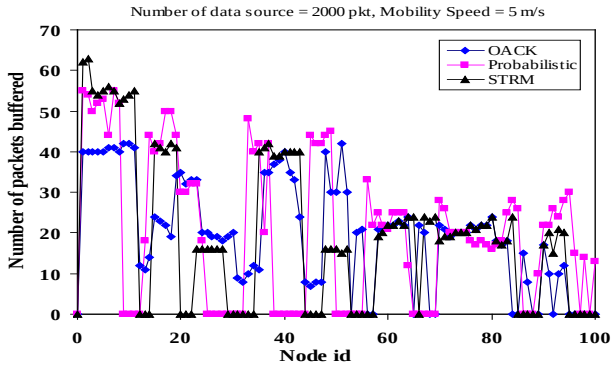

Figure 11. Comparison of buffering load, node 0 is the sender node

B. Percentage of Request Packets

Session Size Impact on the Percentage of Request Packets

Fig. 12 shows the percentage of request packets as a function of session size. As shown in Fig. 12, STRM and the probabilistic algorithm show comparable percentages of requests when the session size is less than 40 out of 100 receiver nodes. When the session size is more than 40 receiver nodes, the probabilistic algorithm exhibits an increase in the percentage of requests because the buffering nodes eventually receive the data packets during the data packet sending. The receiver nodes in the probabilistic algorithm do not send ACK packets to discard the data packets in the long-term buffers in the algorithms such as STRM and OACK. This behavior causes the buffer to overflow in some of its long-term nodes. From the figure, the OACK exhibits less percentage of requests than the others.

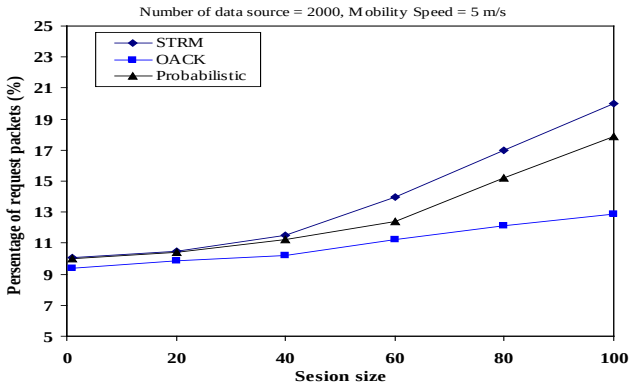

Figure 12. Percentage of request when the session size increases

Mobility Speed Impact on the Percentage of Request Packets

Fig. 13 shows the effect of mobility speed on the percentage of request packets. The mobility speed causes some FS nodes to move away from the sender node, thus the sender makes another selection for new FS nodes for retransmission requests. The figure shows that the probabilistic algorithm has a higher percentage of requests than the others when the mobility speed increases. This result is due to the fact that when the FS nodes receive a request from their receiver nodes, and the requested packet is not in the buffer of the FS node, then the FS node passes the request to the other FS nodes, thereby increasing the number of requests. In STRM and OACK, when a request for a certain packet is received, and this packet is not in the buffer, the FS nodes pass this request to the sender node. The sender then sends the request packet. From this figure, OACK has the least percentage of requests compared with the others.

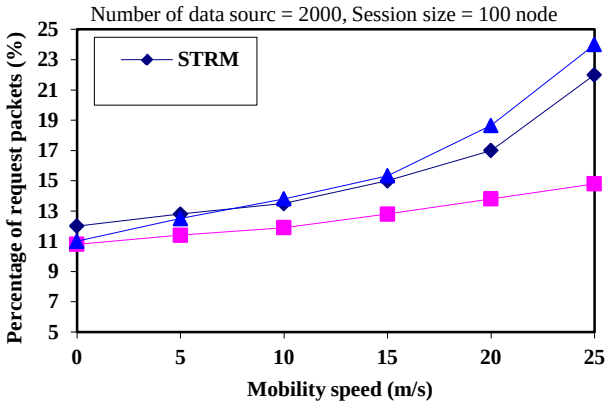

Figure 13. Percentage of request when the mobility speed increases

C. Average of Retransmitted Packets

Session Size Impact on the average of Retransmitted Packets

Fig. 14 shows the effect of different session sizes on the average of retransmitted packets. Comparing OACK and probabilistic algorithm, the performance of the latter with a low mobility of 5 m/s causes more retransmitted lost packets. This behavior is due to the fact that probabilistic algorithm retransmits lost packets to the entire receiver nodes; thus, the retransmission of lost packet overhead increases relative to the session size. Other receiver nodes that have not actually requested a retransmission also receive a retransmitted message and duplicate retransmitted messages.

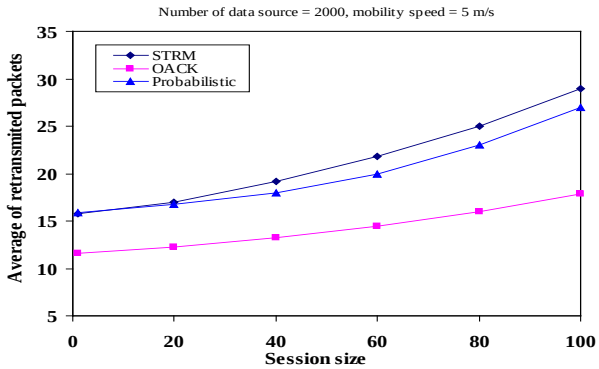

Figure 14. Average of retransmitted packet when the session size increases

Mobility Speed Impact on the Average of Retransmitted Packets

Fig. 15 shows the effect of mobility speed on the retransmission packet average. The probabilistic algorithm exhibits a higher average of retransmission than the others when the mobility speed is more than 10

m/s. Comparing the probabilistic algorithm and STRM, the average of retransmitted lost packets in the probabilistic algorithm increases relative to the mobility speed with high mobility. This result is due to the fact that probabilistic algorithm retransmits lost messages to the entire receiver nodes. From the figure, OACK causes less retransmitted request packets than the others when the speed increases. This behavior is due to the availability of the packets in the buffer of the FS nodes.

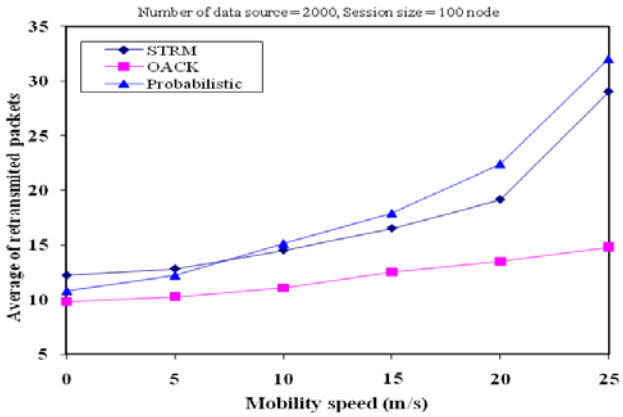

Figure 15. Average of retransmitted packet when the mobility speed increases

D. Average End-to-End Delay

Session Size Impact on the Average End-to-End Delay

Fig. 16 shows the average end-to-end delay of data packets with increasing session sizes. In STRM and probabilistic algorithm, the average end-to-end delay increases slightly with the group size. STRM has a larger delay than the probabilistic algorithm and OACK when the session size increases. The reason for this behavior is the high control overhead in requesting for lost packets, resulting in an overflow in the buffer. OACK exhibits a stable and less average delay than STRM and the probabilistic algorithm when the session size increases.

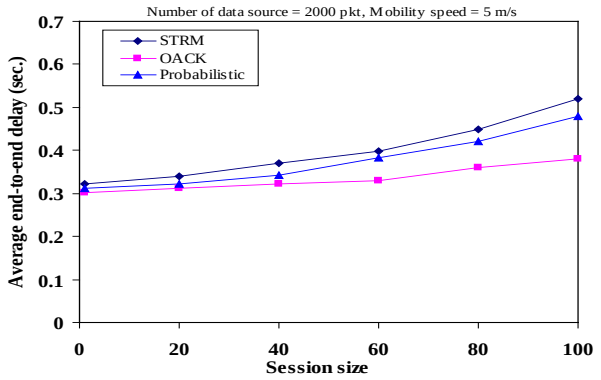

Figure 16. Average of end-to-end delay when the session size increases

Mobility Speed Impact on the Average End-to-End Delay

Fig. 17 shows the average end-to-end delay as a function of mobility speed. The probabilistic algorithm has a larger delay than STRM and OACK when the mobility speed increases. This difference becomes more evident as the mobility speed increases to more than 10 m/s, as shown in Fig. 17. The reason for this behavior is the overflow in the buffer caused by the high control overhead in requesting for lost packets. OACK exhibits a stable and less average delay than STRM and probabilistic algorithm when the mobility speed increases.

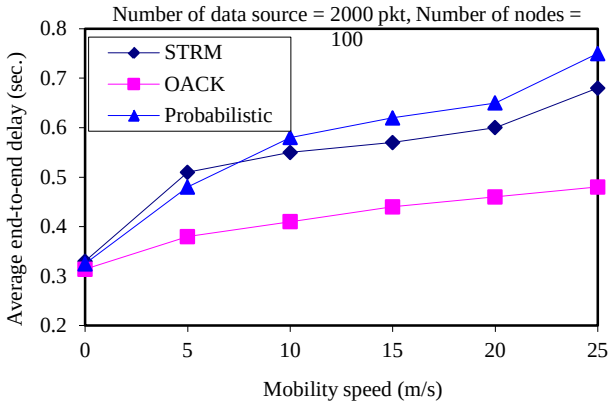

Figure 17. Average of end-to-end delay when the mobility speed increases

E. Average Recovery Latency

Session Size Impact on Average Recovery Latency

Fig. 18 shows the average recovery latency when the session size increases. Probabilistic algorithm causes a larger latency than STRM and OACK. OACK has less recovery latency than the others. The FS buffering nodes are selected immediately in OACK and STRM because the sender has a full knowledge of the receiver nodes in the network. Meanwhile, the probabilistic algorithm has a higher latency time than the STRM because the FS nodes determined by STRM are used as buffering nodes. Other buffering nodes are also determined and used when the probability to buffer on these nodes is high when a message is generated and is directly sent to the buffering nodes.

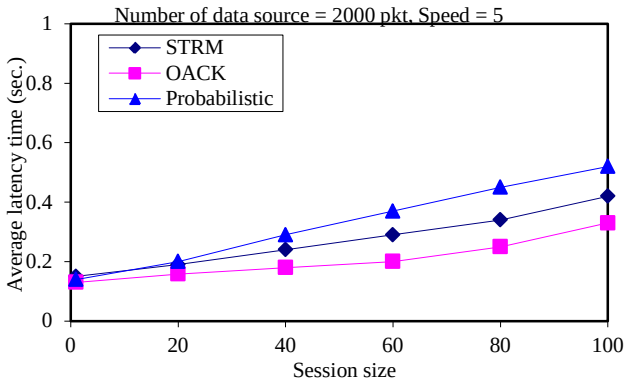

Figure 18. Average of latency time when the session size increases

Mobility Speed Impact on Average Recovery Latency

Fig. 19 shows the effect of mobility speed on the average recovery latency. Furthermore, the effect of mobility speed on the probabilistic algorithm causes a higher average latency compared with OACK and

STRM. Probabilistic algorithm causes a larger latency than STRM and OACK. OACK has less recovery latency than the others. Hence, the latency of OACK does not increase further with mobility speed, indicating that the algorithm scales well.

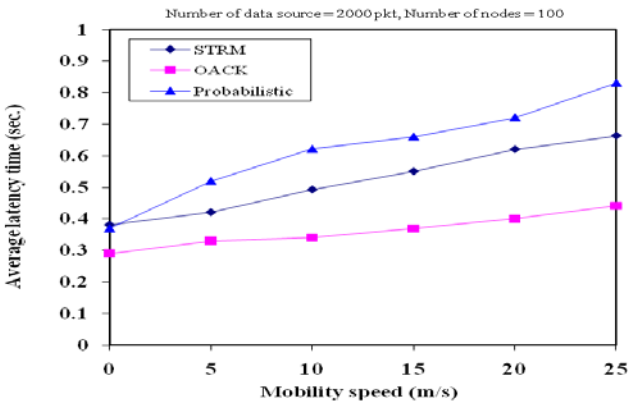

Figure 19. Average of latency time when the mobility speed increases

F. Percentage of Duplicate Packets Performance

Session Size Impact on the Percentage of Duplicate Packets

Fig. 20 shows the effect of session size on the percentage of duplicate packets. Each member calculates the percentage of retransmitted packets received. The result is averaged over all receivers in the group. Furthermore, SSC and SSC-I exhibit less percentage of duplicate packets compared with STRM and probabilistic algorithm when the session size increases. Probabilistic algorithm exhibits a higher percentage of duplicate packets than the other algorithms. The reason for this behavior is that when the probabilistic algorithm retransmits lost requested packets, it multicasts the packets to the entire receiver group, thereby causing duplicates in the receiver that already received the same packets during the transmission time.

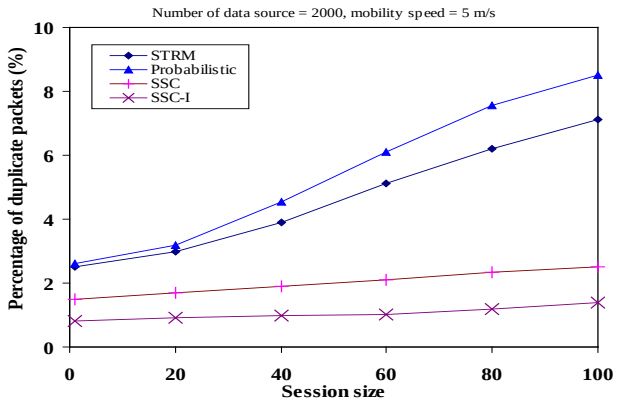

Figure 20. Percentage of duplicate packets when the session size increases

Mobility Speed Impact on the Percentage of Duplicate Packets

Fig. 21 shows the effect of mobility speed on the percentage of duplicate packets. SSC and SSC-I have less percentage of duplicate packets when the mobility speed increases. The probabilistic algorithm has a higher percentage of duplicate packets than the others. The reason for this result is that when the probabilistic algorithm retransmits lost requested packets, it multicasts these packets to the entire receiver group, causing duplicates in the receiver that already received the same packets during the transmission time.

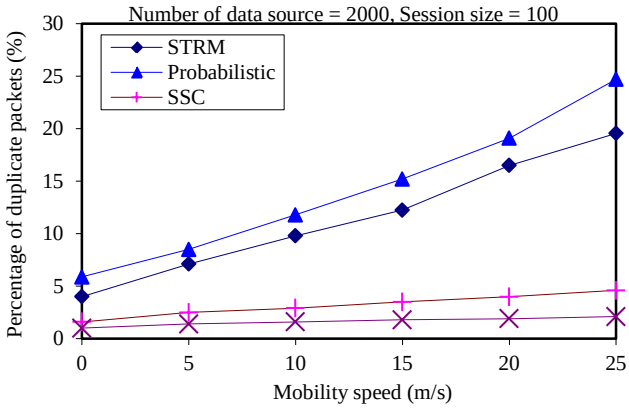

Figure 21. Percentage of duplicate packets when the mobility speed increases

VI. Conclusion

This article presented the OACK, SSC, and SSC-I algorithms. Buffer management was enhanced through an innovative OACK buffering algorithm that explicitly addresses the variances in delivery latency for a multicast group, caused by buffer overflow. The algorithm reduces buffer requirements by adaptively enhancing the selection of the FS nodes to buffer the data packets among members with larger empty buffer space. The key idea of the algorithm is to use the buffer utility approach in selecting the FS nodes, and the ordered ACK from receiver nodes to a sender node. This approach is a powerful technique to achieve high robustness and efficiency of a reliable multicast in MANETs. STRM optimizes local error recovery through innovative SSC and SSC-I algorithms.

The algorithms enhanced the error recovery by eliminating the duplication of retransmission repair packets in local groups. In these algorithms, the requested packet was multicast only to the receivers that request the packet. STRM implements the OACK algorithm to decrease the buffer overflow problem by adaptively enhancing the selection of the FS nodes and decreasing the feedback control packets sent from the receiver nodes. Thus, the selection of FS node was enhanced to buffer the data packets among suitable members with larger empty buffer space. However, the feedback control packets sent periodically from receiver nodes acknowledge only the last data packet of the transmission window received from the sender. STRM implements the SSC and SSC-I algorithms to enhance the error recovery in local groups by creating a temporary sub-group to overcome the duplication problem of the retransmission repair packets. Furthermore, the results proved that the SSC and SSC-I

algorithms are scalable and can be used for a large number of receiver nodes in tree-based protocols, where the increments of the average delay time are smaller.

For future research studies, we intend to evaluate the proposed algorithms in various wireless environments, along with more realistic and comprehensive mobility models and experiment scenarios.

ACKNOWLEDGEMENT

The authors gratefully acknowledge the support by the Centre for Research and Instrumentation Management (CRIM), University Kebangsaan Malaysia, Malaysia. Grant: UKM-GUP-2012-089.

REFERENCES

[1] L. Junhai, X. Liu, and Y. Danxia, "Research on multicast routing protocols for mobile ad-hoc networks," *Computer Networks*, vol. 52, pp. 988-997, 2008.

[2] J. C. Lin and S. Paul, "RMTP: A reliable multicast transport protocol," 1996, vol. 3, pp. 1414-1424.

[3] P. M. Jawandhiya, M. Ali, S. F. Husain, M. Parate, and J. Deshpande, "Reliable Multicast Transport Protocol: RMTP," *International Journal of Advanced Computer Science and Applications*, vol. 1, pp. 74-80, 2010.

[4] K. Guo and I. Rhee, "Message stability detection for reliable multicast," in *Proceedings of the 19th IEEE Conference on Computer Communications (INFOCOM 2000)*, New York, USA, 2000, vol. 2, pp. 814-823.

[5] T. Al-Ahdal, S. Subramaniam, M. Othman, and Z. Zukarnain, "A Source Tree Reliable Multicast Protocol for Ad-Hoc Networks," *The International Arab Journal of Information Technology*, vol. 5, pp. 273-280, 2008.

[6] S. Corson and J. Macker, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations," ed: Mobile Ad-hoc Networks Working Group, http://datatracker.ietf.org/wg/manet/charter/, 1999.

[7] J. Wu and I. Stojmenovic, "Ad hoc networks," *Computer Society*, vol. 37, pp. 29-31, 2004.

[8] J. Broch, D. A. Maltz, D. B. Johnson, Y. C. Hu, and J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocols," in *In Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM'98)*, Dallas, TX, 1998, pp. 85-97.

[9] I. Chlamtac, M. Conti, and J. J. N. Liu, "Mobile ad hoc networking: imperatives and challenges," *Ad Hoc Networks*, vol. 1, pp. 13-64, 2010.

[10] J. Lipman, P. Boustead, and J. Judge, "Neighbor Aware Adaptive Power Flooding (NAAP) in Mobile Ad Hoc Networks," *International Journal of Foundations of Computer Science*, vol. 14, pp. 237-252, 2003.

[11] C. S. R. Murthy and B. Manoj, *Ad hoc wireless networks: architectures and protocols*: Prentice Hall, 2004.

[12] S. A. H. Seno, R. Budiarto, and T. C. Wan, "Survey And New Approach In Service Discovery And Advertisement For Mobile Ad Hoc Networks," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 7, pp. 275-284, 2007.

[13] G. Benincasa, A. Rossi, N. Suri, M. Tortonesi, and C. Stefanelli, "An experimental evaluation of peer-to-peer reliable multicast protocols," in *Militart Communications Conference - MILCOM 2011*, 2011, pp. 1015-1022.

[14] K. Viswanath, K. Obraczka, and G. Tsudik, "Exploring mesh and tree-based multicast. Routing protocols for MANETs," *IEEE Transaction on Mobile Computing*, vol. 5, pp. 28-42, 2006.

[15] S. Sesay, Z. Yang, and J. He, "A survey on mobile ad hoc wireless network," *Information Technology Journal*, vol. 3, pp. 168-175, 2004.

[16] O. Ozkasap, R. Renesse, K. P. Birman, and Z. Xiao, "Efficient buffering in reliable multicast protocols," *Proceedings of the First International Workshop on Networked Group Communication (NGC99), Pisa, Italy*, pp. 188-203, 1999.

[17] Z. Xiao, K. P. Birman, and R. Van Renesse, "Optimizing buffer management for reliable multicast", Washington, DC, 2002, pp. 187-196.

[18] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky, "Bimodal multicast," *ACM Transactions on Computer Systems (TOCS)*, vol. 17, pp. 41-88, 1999.

[19] T. C. Chiang, C. Y. Hsu, and J. L. Chang, "Immediate Group ACK tree (IGA) for reliable multicast in mobile Ad Hoc networks," in *IEEE 3rd International Conference on Communication Software and Networks (ICCSN)*, 2011, pp. 483-487.

[20] B. Adamson, C. Bormann, M. Handley, and J. Macker, "NACK-Oriented Reliable Multicast Protocol," Internet draft, IETF, June 2009. Work in progress. http://www. ietf. org/internet-drafts/draft-ietf-rmt-pi-norm-revised-13. txt. See also NORM web page at http://cs. itd. nrl. navy. mil/work/norm2009.

[21] A. Sobeih, H. Baraka, and A. Fahmy, "ReMHoc: a reliable multicast protocol for wireless mobile multihop ad hoc networks," Las Vegas, NV, 2004.

[22] E. Ahi, M. Caglar, and O. Ozkasap, "Stepwise probabilistic buffering for epidemic information dissemination," in *International Conference on Bio-Inspired Models of Network, Information, and Computing Systems*, Cavalese, Italy, 2006, pp. 1-8.

[23] P. T. Eugster, R. Guerraoui, A. M. Kermarrec, and L. Massoulie, "From epidemics to distributed computing," *IEEE computer Society*, vol. 37, pp. 60-67, 2004.

[24] S. Mishra and L. Wu, "An evaluation of flow control in group communication," *IEEE/ACM Transactions on Networking (TON)*, vol. 6, pp. 571-587, 1998.

[25] K. Yamamoto, S. Yoshitsugu, M. Yamamoto, and H. Ikeda, "Performance evaluation of ACK-based and NAK-based flow control mechanisms for reliable multicast communications," *IEICE Transactions on Communications*, vol. 84, pp. 2313-2316, 2001.

[26] R. Yavatkar, J. Griffoen, and M. Sudan, "A reliable dissemination protocol for interactive collaborative applications," in *Proceedings of the 3rd ACM International Conference on Multimedia*, San Francisco, CA, 1995, pp. 333-344.

[27] L. Rodrigues, S. Handurukande, J. Pereira, R. Guerraoui, and A. M. Kermarrec, "Adaptive gossip-based broadcast," in *In IEEE International Conference on Dependable Systems and Networks (DSN'03)*, San Francisco, CA, USA, 2003, pp. 47-56.

[28] A. M. Costello and S. McCanne, "Search party: Using randomcast for reliable multicast with local recovery," in *Proceedings of the 18th IEEE Conference on Computer Communications (INFOCOM '99)*, New York, NY, 1999, pp. 1256-1264.

[29] J. Baek and J. F. Paris, "A heuristic buffer management and retransmission control scheme for tree-based reliable multicast," *International Journal of Electronics and Telecommunications Research Institute (ETRI) Journal*, vol. 27, pp. 1-12, 2005.

[30] J. Pereira, U. do Minho, L. Rodrigues, U. De Lisboa, M. Monteiro, R. Oliveira, and A. M. Kermarrec, "Neem: Network-friendly epidemic multicast " in *Proceedings on 22nd International Symposium on Reliable Distributed Systems (SRDS'03)*, Florence, Italy, 2003, pp. 15-24.

[31] C. Lindemann and O. P. Waldhorst, "Modeling epidemic information dissemination on mobile devices with finite buffers," *Pro. of the ACM. Int. Conf. on Measurement & Modeling of Computer Systems (SIGMETRICS'05)*, vol. 33, pp. 121-132, 2005.

[32] Y. H. Chen, G. H. Chen, and E. H. K. Wu, "Multiple Trees with Network Coding for Efficient and Reliable Multicast in MANETs," in *39th International Conference on Parallel Processing Workshops (ICPPW)*, 2010, pp. 581-585.

Tariq Abdullah is an assistance professor in Faculty of Computer Science & Information Systems, Thamar University, Thamar, Republic of Yemen. He received his B.Sc. degree in computer science from Baghdad University, Iraq, in 1998. M.Sc. degree in distributed computing from University Putra Malaysia, Malaysia, in 2004 and his PhD degree in computer networks and protocol design from University Putra Malaysia, Malaysia, in 2008. His research interests include wireless

networks, protocols design, and network performance evaluation.

Raed Alsaqour is an assistant professor in the Computer Science Department, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Malaysia. He received his B.Sc. degree in computer science from Mu'tah University, Jordan, in 1997. M.Sc. degree in distributed system from University Putra Malaysia, Malaysia, in 2003 and his PhD degree in wireless communication system from Universiti Kebangsaan Malaysia, Malaysia, in 2008. His research interests include wireless network, ad hoc network, vehicular network, routing protocols, simulation, and network performance evaluation. He also has a keen interest in computational intelligence algorithms (fuzzy logic and genetic) applications and security issues (intrusion detection and prevention) over network.

Maha Abdelhaq received her Bachelor and Master degrees in Computer Science in 2006 and 2008 from Jordan University, (Amman, Jordan). She expected to receive her PhD degree in wireless ad hoc network security in June 2013 from Universiti Kebangsaan Malaysia, Malaysia. Her research interests include ad hoc network, routing protocols, network security, artificial computational intelligence and network performance evaluation.

Rashid Saeed received his PhD majoring in Communications and Network Engineering, UPM, Malaysia. He is senior Assistant Professor since 2008 in SUST, Sudan. He was senior researcher in Telekom Malaysia™, Research and Development (TMRND) and MIMOS Berhad, in 2007, 2010 respectively. His areas of research interest include wireless broadband, WiMAX Femtocell. He successfully award 10 patents (two are U.S) in these areas. Dr. Rashid is an IEEE member since 2001 and Member IEM.

Ola Alsqour is a technical support and web development engineer in GIS & Remote Sensing Department, General Computer and Electronics Company, Amman, Jordan. She received her B.Sc. degree in computer engineering from Jordan University, Jordan, in 2010. She currently is acquiring her M.Sc. degree in computer engineering from Jordan University, Amman, Jordan. Her research interests include wireless network, ad hoc network, computer architecture, remote sensing and geographical information system.