A Simulation Engine Model Analysis for Reliable Multicast Protocol in Ad Hoc Network

T. Alahdal, S.Shamala, M.Othman, Z. Zukarnain

Department of Communication Technology and Networks, Faculty of Computer Science and Information Technology, University Putra Malaysia (UPM), 43400 Serdang, Selangor, Malaysia

Summary
Traditionally, reliable multicast protocols are deterministic in nature. It is precisely this determinism which tends to become their limiting factor when aiming at reliability and scalability, particularly in highly dynamic networks, e.g., ad hoc networks. In multicast communication, many reliable multicast schemes were studied in order to overcome packet losses in the network. This paper describes our effort to build a detailed simulation model for the reliable multicast transport protocol based on measurements taken from a variety of mapping sources and tools. We identify key attributes of a network design to develop the simulation engine model. The attributes of the model, are discussed in details to ensure the features of the protocol those are captured by the simulator. Finally, the results acquired have proven the ability of the simulator to provide a good analysis tool.

Key words:
Ad Hoc network, Reliable Multicast, Simulator Engine.

1. Introduction

There are three techniques that may be used as methodologies to model and evaluate the performance of the computer networks; *formal analysis*, *real life measurements* and *simulation* [16][17]. As ad hoc networks are comprised of multiple entities (nodes) interacting in a complex and non deterministic manner, it is not sufficient to only model a single entity nor feasible to model all the interactions that occur. The dynamic nature and fundamental complexity of the MANETs makes formal analysis and real life measurements extremely complicated and certainly costly [10]. From the other side, there are no real-world implementations in "best-practice" for conducting experiments of MANETs, which could play the role of reference models and limit the range of the possible alternatives [3][12]. Thus, the use of simulation in ad hoc network research has proven indispensable and necessary in gathering an understanding of the interactions and performance of proposed mechanisms in ad hoc network research.

The current state of MANET simulation studies published the Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc) from 2000-2005 consider simulation as an often used tool to analyze MANETs; 114 out of the 151 MobiHoc papers published (75.5%) used simulation as the basis for the study to test their research [11].

Simulation tools allow researchers to gather an understanding of the complex interactions and resulting performance achieved by proposed mechanisms in an environment that allows for repeatability of experiments and easy prototyping of proposed mechanisms. There are many discrete-event network simulators tools available for the MANET community that allow testing of proposed ad hoc network research [18]. The most popular commercial simulation tools are QualNet [19] and OPNET [15], while the most popular non-commercial simulation tools are Network Simulator-2 (NS-2) [14] and a non-commercial version of QualNet called GlomoSim [26].

Depend on the survey results on [11] shows NS-2 is the most used simulator in MANET research; 35 of the 80 simulation papers 43.8% used NS-2, 27.3% Self-developed or custom simulators, 10% GloMoSim, 6.3% OPNET, 6.3% QualNet, 3.8% MATLAB and 2.5 CSIM simulators tools.

Developing a network simulator requires much time and efforts [21]. Due to limited development resources, network simulators version 2 have the following limitations:

- NS-2 suffers for its lack of modularity and its inherent complexity. Indeed, adding components/protocols or modifying existing ones is not as straightforward as it should be.
- Another well-known weakness of NS-2 is its high consumption of computational resources. A harmful consequence is that NS-2 lacks scalability, which impedes the simulation of large networks [10]. Also, there is a lack of tools to describe simulation scenarios

Manuscript received May 5, 2008
Manuscript revised May 20, 2008

and analyze or visualize simulation trace files. These tools are often written with scripting languages. The lack of generalized analysis tools may lead to different people measuring different values for the same metric names [4].

- In NS-2, it is self-documented that ''there is no dynamic receiver's advertised window for reliable transmission.''

Simulating an ad hoc network using any software tool is a complex task and feces several key challenges because of the dynamic nature of wireless ad hoc nodes. The network topology can change randomly, at unpredictable times [8]. Wireless links generally have limited bandwidth [6]. The majority of nodes rely on short-living batteries [8][9]. The objective of the simulator is to implement the Reliable Multicast Protocol for Wireless Mobile Multihop Ad Hoc Networks (ReMHoc) [20] which allows a single sender to deliver packets in ordered manner to a group of multicast receivers. Simulation of ReMHoc protocol also is subject to ad hoc constraints. Therefore, the design of an efficient simulator is critical. Basically every simulation model is a specification of a physical system (or at least some of its components) in terms of a set of states and events. Performing a simulation thus means mimicing the occurrence of events as they evolve in time and recognizing their effects as represented by state future event occurrences induced by states have to be planned (i.e. scheduled).

This paper presents a proposed discrete event simulator model for ReMHoc protocol using C++ code which incorporates efficient characteristics as proven by the result. The simulator encompasses the analysis of feedback suppression in order to avoid negative acknowledgement (NACK) implosion and retransmission exposure.

The remainder of the paper is organized as follows. Section 2 describes the details of the multicast discrete event simulator used and its components. Section 3 describes the simulator structure, simulator engine operations, data structures and events that implemented in the simulator. The simulator framework and its network topology model and mobility model used in the simulator are presented in Section 4. The performance metrics and the results obtains are presented in Section 6. Section 7 gives a conclusion of the paper.

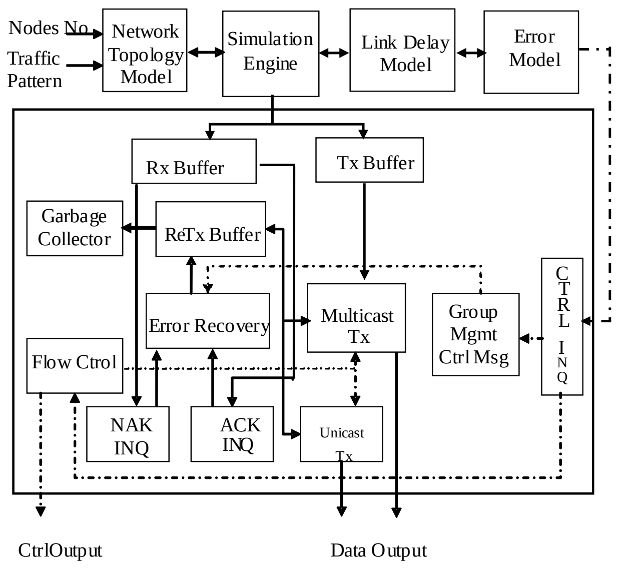

Fig. 1 Block diagram of MDES

2. Multicast Discrete Event Simulator (MDES)

Fig.1 depicts the block diagram of the developed Multicast Discrete Event Simulator (MDES). The figure shows the data flow and control flow inside a component of the simulator. The component could be a sender or receivers. The data flow is depicted in regular dark line while the control flow is depicted in dashed line. There are two separate buffers for transmission and retransmission. Similarly there are separate transmission mechanisms for multicast and unicast. There are separate entities for managing the control messages for group management and flow control. Also, separate buffers are provided for ACK and NAK. There are two components in the simulator design: Sender and Receivers. The functions of each of them are detailed as follows.

- Sender Component

The sender component is responsible for the transmission of the new data packets; retransmit lost packets or send messages to advertise itself as an ACK processor to the whole group. In addition, it is also responsible for number of other functions like error recovery and congestion control.

- Receiver Component

The receiver component delivers the received data packets to the receiver application. It also sends the ACK and NAK to the sender in accordance with the reception of data packets.

The block diagram in Fig. 1 illustrates briefly how the simulator works. Further details of the simulation procedure are explained below:

1. Simulation variables input is required.
2. From the number of nodes provided, a mesh network topology is created from a set of receiver nodes that are randomly distributed.
3. Built connectivity for each node. When any pair of nodes is located within a distance that is based on the transmission range provided, they would be connected.
4. Implement the traffic mobility model.
5. The sender multicasts the data packets globally to the entire multicast mesh.
6. Each receiver sends their ACKs / NAKs to the sender.
7. The sender node receives the ACKs / NAKs from its children attached below them.
8. The sender performs local re-transmissions for its children that send NAK packets.
9. After the application of each algorithm, the simulation variables output would be provided. The output includes the performance metrics of the algorithms.

The following sections describe the structure of the MDES and its engine that is used to drive the simulator developed in this research.

3 MDES Structure

The MDES structure depends on messaging system between each component. To understand the details of the messaging system used in the MDES, some understanding of the simulator engine, data structures, message types and events handling are necessary. The following sections describe the development of a discrete event engine that is used to manage the events used for the simulator developed in this paper.

3.1 MDES Engine

The simulator engine is responsible for providing the framework which allows MDES operations to be simulated and data traffic conditions to be generated in the simulator. The following *List Events*, *Event List* and *Timer Engine* are used to implement such an engine.

List Events are responsible for implementing the event driven aspect of the simulator engine and represent events that occur in the simulator. Each *List Event* contains data associated with the event it represents as well as a reference to the event handler which processes the event. When an event occurs, a corresponding *List Event* is created and dispatched by activating its handler. So in effect, *List Events* act as triggers for handlers that carry out operations in the simulator.

In contrast, the *Event List* and *Timer Engine* implement the serial and discrete aspect of the simulator engine. Both components form an event scheduler that schedules List Events for future execution.

The event scheduler has being designed to operate as follows. Whenever an event is scheduled, a corresponding *List Event* is created and passed to the *Event List* that stores it for future execution. The actual execution of List Events is triggered by the *Timer Engine* which keeps track of the virtual time in the simulator (in simulator time units) via a counter that is periodically incremented every time cycle. During each cycle, the *Event List* is called to retrieve and sequentially dispatch all *List Events* scheduled for the current time. On completion, the *Timer Engine* increments its counter and proceeds to the next cycle. It is this continual scheduling of *List Events* on the *Event List* for their execution by the *Timer Engine* that drives the MDES simulator forward.

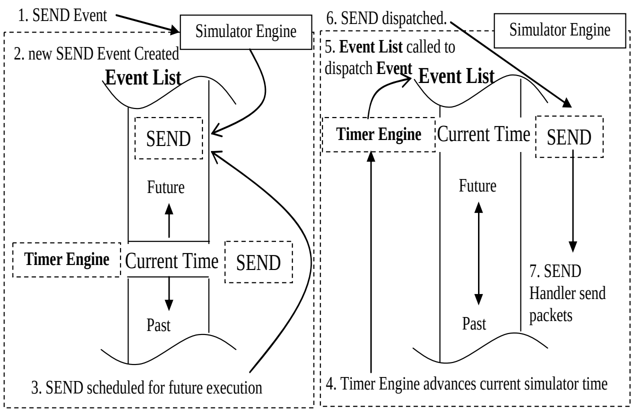
1. SEND Event
Simulator Engine
2. new SEND Event Created
Event List
SEND
Future
Timer Engine
Current Time
SEND
Past
3. SEND scheduled for future execution
6. SEND dispatched.
Simulator Engine
5. Event List called to dispatch Event
Event List
Timer Engine
Current Time
SEND
Future
7. SEND Handler send packets
Past
4. Timer Engine advances current simulator time

Fig. 2 Simulator engine operations

Fig. 2 illustrates how the simulator engine operates. The sender initiates a SEND primitive, causing the engine to schedule a new SEND timer for other outgoing packets. This is accomplished by creating a SEND *List Event* that represents a SEND timer event and scheduling it via the *Event List* so that it executes at a future time corresponding to the packets SEND timeout**.** The *Event List* then stores this *List Event* by appending it onto the *List Events* for execution during that simulator time cycle.

With the simulator engine defined, the remainder of the simulator must be structured around this simulator engine. This is described in the next section.

3.2 Data Structures

These data structures are maintained by the sender and receivers mobile nodes in the simulator. All the nodes running in the simulator are required to maintain the following data structures [13]:

1. **Member Table**
   Member table needs to be maintained by multicast receivers. For each multicast group they are

participating, they should store multicast source *id* and the timestamp value to detect the expired source.

2. **Routing Table**
   A routing table is created on demand and is maintained by each node. The routing table includes the next hop address for each destination maintained in the routing table.
3. **Forwarding Node**
   When a node is not a receiver node and located between two receiver nodes in the multicast group, this node is uses as a forwarding node of the multicast group, it use the variable *forwarding_node*. The value of this variable is true if the node is a member of this group, otherwise it is false.
4. **Message Cache**
   The Message Cache is maintained by each node to detect duplicates of the packets. When a node receives a new *Route_Update* or *Data_Msg,* it stores the source *id* and the sequence number of the packet.

When the data structures defined, the message types of the simulator must be defined. This is described in the next section.

3.3 Messages Types

The following four message types are used in the implementation:

Route_Update: This message is used for constructing the routing mesh rooted at the sender on demand. While a multicast sender has packets to send, it periodically broadcasts this message to all members as long as the sender has packets to send. This periodic transmission refreshes the membership information and updates the route in the face of node movements. Every node on receiving a *Route_Update* message updates the *Message Cache* by including the sender *id* of the message, the hop count value and the sequence number of the packet to detect duplicates.

Route_Table: This message used for establishing or updating the group memberships and routes. When the *Route_Update* message reaches a multicast receiver, the receiving node creates or updates the sender entry in its Member Table. A multicast receiver from its *Member Table*, it builds a *Route_Table* and periodic sends it to its neighbors

Data_Msg: Messages of this type is used by the nodes to send, receive and forward data. On receiving this message the node adds it to the data buffer.

ACK_Msg: Messages of this type is used by the receiver nodes to inform the sender of a packet received. On receiving this message the sender marks the packet as stables.

After the simulator engine, data structures, message types are defined, the simulator must be show how the events handling by the simulator. This is described in the next section.

3.4 Events Handling

The simulator needs to provide event handlers for the following events:

Sender sends a Data_Msg: When a sender wants to send a Data_Msg *m* to a multicast group, it executes the following command:

SEND MULTICAST(s,seq,m) the sender *s* transfer of this message by broadcasting it to all its one-hop neighbors. Indicates that the sender *s* wants to multicast a message *m* to a multicast group; *seq* is the sequence number.

Node *i* receives Data_Msg: When a node receives a Data_Msg (j,seq,m) message, it store the *seq* numbers of the packets that received and the sender *id* in the *Message Cache.* This *seq* number uses to filter out duplicate receipt of a Data_Msg.

Node *i* sends unicast ACK_Msg: The receiver node sends ACK_Msg to its sender node indicating the status of the received packets. The ACK_Msg used to record the existence of correctly received packets stored in its buffer. The sender node upon receiving ACK_Msg from all the receiver nodes marks message *m* as stable.

Receiving a Route_Update (i,k,HighestPkt) message: the Route_Update message works together with the JOIN_REQUEST_ACK message to found a path from the multicast mesh to the node requesting to JOIN the multicast. The sending node *i* is requesting to join the multicast group; *k* is the *id* of the node which initiated this Route_Update message; Highestpkt is the maximum value of the packet belonging to the node which have been received.

The Route_Update message initiated by a multicast node (i.e. node *u*) is processed by each node in the path from *u* to a current node in the multicast mesh. The path is progressively determined hop-by-hop: the current node (i.e. node *i*) queries the underlying protocol for the next hop node in the path to the sender node from itself. As the Route_Update message progress towards the sender node, a mesh path in the reverse direction is established by receiver of the Route_Update message (i.e. node *i*) adding the node from which it received the Route_Update message as on of its *Member Table*.

A node sends a Route_Update message when it moves to a new location in the network or when it gets (re)connected to the network. Due to mobility or disconnection, a node may have not received some of the data messages for the multicast group it belongs to. Hence, Route_Update message is used not only for the initiation of setting up a path for future multicast data messages but also for sending (retransmission) any missing multicast data messages to the joining nodes along the new path being setup. A node *i* receiving a Route_Update message can be of one of the following three types: i) a regular node, ii) a multicast node connected to the multicast mesh, and iii) a forwarding node.

Receiving Route_Table (i,j,stable) message: the message indicates that a node *i* will forward any message which needs to be multicasted to the receiver of this message; node *j* is the intended receiver of this Route_Table message; stable is the value of the *HighestPKT* received correctly just before sending this Route_Table message. A node *i* receiving a Route_Update message sends out a Route_Table message to the next node in the path to the initiator of the Route_Update message, unless the node *i* is a regular node for which *HighestPkt i* ≤ *HighestPkt u*. Similar to Route_Update message, the Route_Table message also establishes path from the initiator node and the final recipient of the Route_Table message. The rationale for having both Route_Update and Route_Table message establish path to multicast mesh is as follows. Suppose a node *k* sends out a Route_Update message. In case the topology does not change because of the mobility, the Route_Table will follow the reverse path of the Route_Update message. In this case the Route_Table message just helps to propagate the message stabilization information to the nodes in the path. However, in the presence of node mobility the Route_Update message helps to expedite the setting up of paths to mobile nodes. For example, suppose node *k* moves after sending out the Route_Update message. In this case the Route_Table message will be routed to the new location of node *k*. The portion of the path setup by the Route_Update message will be torn down by the QUIT message sent out by a forwarding node which becomes a leaf node after node *k* moves. If a node *i* which is already a receiver nodes, receives a Route_Table message for node *k* it does two things: i) sends QUIT message up to the sender node and ii) forwards Route_Table message towards node *k*. This ensures that every node has a single parent node, i.e., paths established by Route_Table messages in fact construct a mesh.

Receiving a QUIT (i) message: the message indicates that node *i* does not need multicast message. A QUIT message is initiated by a forwarding node in the mesh which becomes a leaf node due to node movements. The QUIT message travels hop-by-hop up the multicast mesh branch tearing down the multicast path up to the sender node. A sender node receiving a QUIT message deletes the node from which it receives the QUIT message from its *Member Table*.

4. Simulator Framework

In this research, a discrete-event simulation program using visual C++ is developed. It is assumed that all nodes and links work properly and none of them fail during the simulation time. The simulation program is run 10 times for the same topology and input variables. The results are taken as the average among these iterations in order to have stable results. The following is the details description of the simulator program constructions.

4.1 Network Topology Model

Network topology is randomly generated for a fixed number of nodes. The location of each node is assumed to be on the two-dimensional coordinate system (*x*, *y*) and each location is generated randomly in a uniform distribution. The variables input that needed to create random topology is:
1. The fixed number of nodes in the network
2. The initial network area to begin with
3. Every node has a unique address or ID.

Table 1 summarizes the variables used as input for a simulation of running the reliable multicast routing algorithms. The transmission range used is 250m since it is typical for mobile devices. The scenario area of the simulation is limited to 700mx700m to reduce the likelihood of no connectivity at all when the nodes are allowed to go further than that. This consideration is based on the number of nodes that is 30.

Radio irregularity factors are 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*.

Table 1: Simulation Parameters

Variable	Value
PACKET_SIZE	512
NO_NODE	30
LINK_BW	2Mbps
TRANSMISSION_RANGE	250 m
AREA_SIZE	700*700
MESSAGE_SIZE	2000 pkt
LOSS_RATIO	0.1
CACH_SIZE	2000 pkt
MAX_QUEUING_DELAY	80 ms
MIN_QUEUING_DELAY	20 ms
PROPAGATION_DELAY	10 ms

4.2 Neighboring Membership

In order to run a multicast algorithm in the network topology, first specifying the measure to link two adjacent nodes. It is important to determine which nodes could be categorized as neighboring nodes of one node since multicast involves broadcast to initiate the routing to find the multicast membership. The measure to connect one node to its neighboring nodes is the maximum distance that can be reached by the node. Represent this measure by the Eq. 1:

$i_{(x,y)}$ is neighbor to $j_{(x,y)}$ if, $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \leq R$ (1)

where (i,j) represents two adjacent nodes i and j such that the distance between those two nodes (represented by the coordinate (x, y) of each node) is less than or equal to the transmission range (R).

4.3 Transmission Range

The transmission range is the maximum distance at which the radio signal from a node can be received. Several articles (e.g., [23]) in the literature have discussed the problems associated with specifying the transmission range of a node as a uniform circular representation of the transmission range [7]. Transmission range is assumed to be equal for all nodes in the network. Since it is the characteristic that is represented by a circle centered at each node, the transmission range is therefore the R mentioned in the previous subsection. The wireless transmission range commonly used is between 100m – 250m. [22] stated that the optimum transmission range in ad hoc wireless networks cannot be represented by a fixed number because there are many factors influence the transmission.

4.4 Error Model

There are two causes for packet dropping. The first is due to the probability of error inherent to each link. The packets dropping probability due to this kind of errors is small. It is assumed that the packet loss probability due to this reason equal to 10^{-3} at each wireless link.

The second reason of errors is a result of buffer overflow in routers and this represents the dominant source for packet error. The value of packet loss probability due to the second reason is generated according to a uniform distribution U [0, 1]. Thus, the packet loss probability in the link can be represented by the Eq. 2:

$$\textit{Probability of error} = \textit{link error rate} + \textit{buffer overflow} \quad (2)$$

In the simulation program, the first component of the Eq. 2 is included in the second component in order to determine an average value for the lost ratio for the wireless link. This lost ratio taken as 0.1 as shown in Table 1. Thus the packet is considered to be lost if the uniform function return a value less than 0.1.

4.5 Link Delay Model

There are three reasons related to delay in a link. The first is due to the propagation delay and represent small value and is the same for the same kind of links. The second reason depends on the link bandwidth and packet size. The packet will encounter a smaller delay when it passes a link with higher bandwidth. Also a packet of a smaller size will encounter a smaller delay when passes the same link as compared to a packet of a larger size. Thus, delay in the link is decreased by increasing the link bandwidth and decreasing packet size. The packet size is taken as 512 bytes and the bandwidth is taken as 70 kbps for this research. The third reason of delay in the link is due to congestion in the link and this causes the major reason of delay. The queuing delay is obtained randomly from the uniform distribution U [20,80] ms. Thus, the link delay can be represented by the Eq. 3:

$$\textit{Link Delay} = \textit{prop. delay} + s / bw + \textit{queuing delay} \quad (3)$$

Where s is the packet size and bw is the link bandwidth. The first and second parts are constant for each link and the third part is obtained from a uniform distribution.

4.6 Random Waypoint Mobility Model

The Random Waypoint Mobility (RWM) model includes pause times between changes in direction and/or speed [25][27]. First the node stays in one location for a certain period of time (i.e., a pause time). Once this time expires, the node chooses a random destination in the simulation area and a speed that is uniformly distributed between [min_{speed}, max_{speed}]. The node then travels toward the newly chosen destination with the selected speed. Upon arrival, the node pauses for a specified time period before starting the process again. RWM model is also a widely used mobility model (e.g., [2][5][1][24]).

5. Performance Metrics

The performance of our reliable multicasting protocol in terms of delivery guarantee and bandwidth consumption (control overhead) using the Visual C++ language is experimented extensively. For reasons of comparison, a reliable multicast protocol using a NAK-based buffer management scheme is implemented [20]. Each receiver node multicast a NAK to entire group whenever it detect a packet loss. Before sending a NAK or a retransmission, a node waits for a random period and suppresses its transmission if it hears a transmission from another node. The nodes that do not cache packets are sent a NAK directly back to the sender. This protocol is denoted as ReMHoc and used as baseline for comparison. The impact of a node mobility and session size on the performance of reliable multicast protocol in a variety of MANET scenarios is analyzed.

The metrics are chosen to evaluate the efficiency in addition to the effectiveness of the protocols. Duplicate request and retransmitted messages are taken into account in these measurements. Then, mean values are calculated for each simulation. The following metrics in performance comparison are used:

- **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 original data packets transmitted by sender.

- **Average retransmission packets,** average retransmission packets which is the ratio between the number of retransmission packets transmitted by the sender and group members and the total number of original data packets transmitted by 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 it is received by the receiver nodes, and then averaged over the total number of receiver nodes.

5.1 Simulator Results

Figures 3, 4, 5 provide the results of the MDES with the effect of mobility and session size. The Figures show a single source multicasts 2000 packets each packet size is 512-byte data packets to a multicast group. The packet transmission rate is 500 ms between each two transmission window of packets. A receiver's cache size (CACH_SIZE) of 2000 packets and the cache replacement strategy is First In First Out (FIFO). Session size (i.e., number of multicast group members) varies from 10 to 30 members with steps of 5 nodes. There are more frequent disconnection in the mesh topology caused from the underline on demand multicast protocol [13]; hence, the number of missing packets increases and accordingly the percentage of Requests increases (Fig. 3). It should be noticed that the change in the percentage of Requests is insignificant for higher session sizes. This indicates that more receiver nodes implemented to forwards data packets to other nodes when the speed equal to 20 m/s.

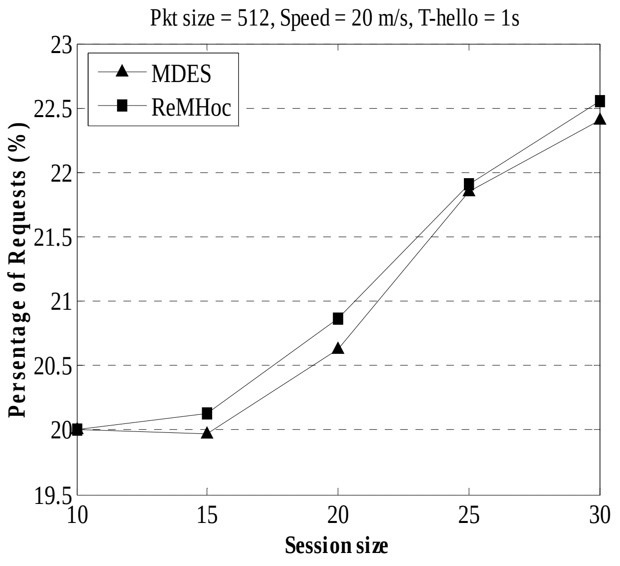

Fig. 3: The percentage of requests with the session size

As is seen, the end-to-end delay from the MDES results closely matches the ReMHoc protocol simulation results. At a higher mobility speed of 20 m/s and 10 receiver nodes of 30 nodes, the difference between end-to-end delay from MDES and ReMHoc protocol differ only by $\pm$ 0.1528 ms (Fig. 4) while for a 30 nodes this difference goes down to around $\pm$0.0104 ms (Fig. 4). In general the mean value of this difference is within $\pm$0.04106 ms. This is true for retransmission probabilities as well. The difference between retransmission from MDES and ReMHoc protocol differ only by ±0.0001 (Fig. 5) while for a 30 nodes this difference goes up to around ±0.308 (Fig. 5). In general the mean value of this difference is within ±0.0115. From the results obtained, it can be seen that the MDES can recapture the results of previous work.

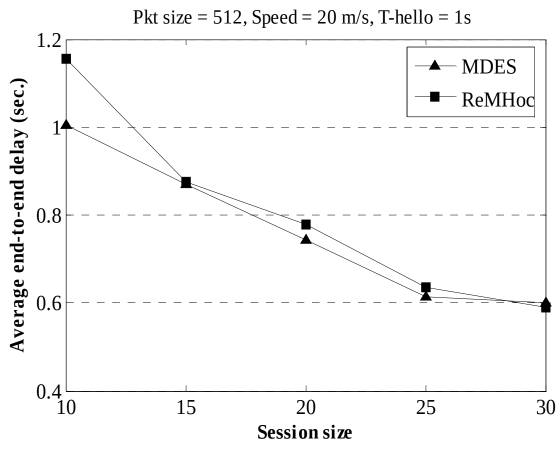

Fig. 4. The average end-to-end delay with the session size.

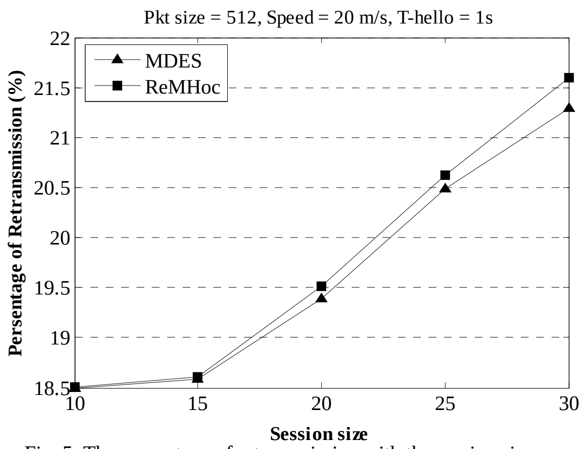

Fig. 5. The percentage of retransmission with the session size

6. Conclusion

This paper detailed the proposed simulator engine model for reliable multicast protocol in wireless ad hoc network. This simulator builds on messaging system. It could be an important aspect for the planning and deployment of new reliable multicast protocol. In this paper the development environment based on messaging system that implement in the simulator. Then discussed the design of simulator with basic features including data structures and events used, the network topology uses the random waypoint model for node mobility. In future research to enhancing the ReMHoc protocol requires that the set of receivers to be organized in a structure as a tree. In such an organization, a source node is the root of the tree, and leaf nodes are nodes with no children. The intermediate nodes are responsible for requesting lost packets to their parents and retransmitting those packets to their children under request.

References

[1] Garcia, J.J. and Spohn, M.(1999). Source-Tree Routing in Wireless Networks. In Proceedings of the 7th International Conference on Network Protocols (ICNP).

[2] Broch, J.; Maltz, D.; Johnson, D.; Hu, Y. and Jetcheva, J. (1998). Multi-Hop Wireless Ad Hoc Network Routing Protocols. In Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM), pp. 85–97.

[3] Caro G.A. (2003). Analysis of simulation environments for mobile ad hoc networks. Technical Report IDSIA-24-03, IDSIA, Lugano, Switzerland.

[4] Cavin, D.; Sasson, Y. and Schiper, A. (2002). On the Accuracy of MANET Simulators. In Proceedings of the 2nd ACM International Workshop on Principles of mobile networks, pp. 38-43.

[5] Chiang, C.C. and Gerla, M. (1998). On-Demand Multicast in Mobile Wireless Networks. In Proceedings of the IEEE International Conference on Network Protocols (ICNP).

[6] Chlamtac, I.; Conti, M. and Liu, J. (2003). Mobile Ad Hoc Networking: Imperatives and Challenges. Ad Hoc Networks, Vol. 1, No. 1, pp. 13-64.

[7] Clark, B.N.; Colbourn, C.J.; and Johnson, D.S. (1990). Unit disk graphs. Discrete Mathematics, vol. 86, pp. 165-177.

[8] Corson, M.S. and Macker, J. (1999). Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations. Mobile Ad-hoc Networks Working Group, http://www.ietf.org/rfc/rfc2501.text.

[9] Cavilla, A.L.; Baron, G.; Hart, T.E.; Litty, L.; and de Lara, E. (2004). Simplified simulation models for indoor MANET evaluation are not robust. In Proc. of the IEEE SECON.

[10] Hogie, L.; Bouvry, P. and Guinand, F. (2006). An Overview of MANETs Simulation. Electronic Notes in Theoretical Computer Science, vol. 150, no. 1, pp. 81-101.

[11] Kurkowski, S.H. (2006). CREDIBLE MOBILE AD HOC NETWORK SIMULATION-BASED STUDIES. Ph.D. dissertation, Board of Trustees of the Colorado School.

[12] Kiess, W. and Mauve, M. (2007). A Survey on Real-World Implementations of Mobile Ad-Hoc Networks. Elsevier's Ad Hoc Networks, Vol. 5, No. 3, pp. 324-339.

[13] Lee, S.J.; Su, W. and Gerla, M. (2002). On-demand multicast routing protocol in multihop wireless mobile networks. Mobile Networking Application, vol. 7, no. 6, pp. 441-453.
[14] NS-2. The Network Simulator. http://www.isi.edu/nsnam/ns/.
[15] OPNET. OPNET Network Simulator. http://opnet.com/.
[16] Sadiku, M.N.O. and Ilyas, M. (1995). "Simulation of Local Area Networks", CRC Press.
[17] Sarela, M. (2004). Measuring the Effects of Mobility on Reactive Ad Hoc Routing Protocols. Helsinki University of Technology Laboratory for Theoretical Computer Science, Research Reports 91.
[18] SEACORN Project. SEACORN simulation tools. URL: http://seacorn. ptinovacao.pt.
[19] SNT. Scalable Network Technologies: QualNet WiFi Network Simulator. http://www.scalable-networks.com/.
[20] Sobeih, A.; Baraka, H. and Fahmy, A. (2004). ReMHoc: A Reliable Multicast Protocol for Wireless Mobile Multihop Ad Hoc Networks. Proceedings of IEEE Consumer Communications and Networking Conference (IEEE CCNC 2004), Las Vegas, NV, pp. 146-151.
[21] Wang, S.Y.; Chou, C.L. and Lin, C.C. (2007). The design and implementation of the NCTUns network simulation engine. Simulation Modeling Practice and Theory. Vol. 15, pp. 57–81.
[22] Sanchez, M.; Manzoni, P. and Haas, Z.J. (1999). Determination of critical transmission range in ad-hoc networks. Proc. of Multiaccess Mobility and Teletraffic for Wireless Communications'99. Venice, Italy.
[23] Rappaport, T. (1996). "Wireless Communications - Principle and Practice", Prentice-Hall.
[24] Johansson, P.; Larsson, T.; Hedman, N.; Mielczarek, B. and Degermark, M.(1999). Routing Protocols for Mobile Ad-Hoc Networks - A Comparative Performance Analysis. In Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM), pp. 195–206.
[25] Johnson D. and Maltz D. (1996). Dynamic Source Routing In ad hoc Wireless Networks. In Mobile Computing, edited by Tomasz Imielinski and Hank Korth, chapter 5, Kluwer Academic Publishers, pp. 153- 181.
[26] Bajaj, L.; Takai, M.; Ahuja, R.; Bagrodia, R. and Gerla, M. (1999). GloMoSim: A Scalable Network Simulation Environment. Technical Report 990027.
[27] Bettstetter, Christian; Resta, Giovanni and Santi, Paolo (2003). The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks. IEEE Trans. Mob. Comput. Vol. 2, no. 3, pp. 257-269.

Tariq A. Alahdal received the B.S. degree in Computer Science from Baghdad University, Iraq, in 1998, The M.S. degree from University Putra Malaysia, Malaysia in 2003. Currently he is a PhD Student at faculty of Computer Science and Information Technology, UPM. His research area focuses on reliable multicasting protocol. Especially, he is interested in congestion control and reliable protocol for multicast protocol and buffer management. He is also doing research on reliable multicast routing protocol for ad hoc networks.

Subramaniam K. Shamala received the B.S. degree in Computer Science from University Putra Malaysia (UPM), in 1996, M.S. (UPM), in 1999, PhD (UPM) in 2002. Her research interests are Computer Networks, Simulation and Modeling, Scheduling and Real Time System. She already published several journal papers.

Mohamed Othman is an Associate Professor at the Department of Communication Technology and Network, University Putra Malaysia. He received his PhD from the National University of Malaysia with distinction (Best PhD Thesis in 2000 awarded by Sime Darby Malaysia and Malaysian Mathematical Science Society). In 2002, 2003, 2004, 2005 and 2006, he received gold medal awards for Research and Development Exhibitions. His main research interests are in the fields of parallel and distributed algorithms, grid computing, high-speed computer network, network management (security, wireless and traffic monitoring) and scientific computing. He is a member of IEEE Computer Society, IEICE Communication and Engineering Science Societies, Malaysian National Computer Confederation and Malaysian Mathematical Society. He already published more than eighty National and International journal papers. He is also an associate researcher and coordinator of High Speed Machine at the Laboratory of Computational Science and Informatics, Institute of Mathematical Science (INSPEM), University Putra Malaysia.

Zuriati A. Zukarnain is head of Department of Communication Technology and Networks, University Putra Malaysia (UPM). She received the B.S.Ed degree in (Hons) Major in Physics from UPM, in 1997, M.S. in Information Technology (UPM), 2000, PhD degree In Quantum Computation and Information from University Of Bradford, United Kingdom, 2006. Her research interests are Quantum Computation, Quantum Information & Communication (Entanglement, Teleportation), Computer Networks and Distributed Computing. She already published several journal papers.