Simulation and Comparison of the RASC and the BSMA Wireless Medium Access Protocols

Mohammed Omari, Warda Hadj Fateh, and Zohra Kacemi

***Abstract*— In wireless communication, there are different medium access techniques that allow users to share a common channel. Each protocol is entitled to do its best to avoid collision. In this paper, we have selected two random access protocols for simulation and comparison purposes. The first protocol is the Random Access Scheduling (RASC) protocol that supports unicast communication. The second protocol is the Broadcast Support Multiple Access (BSMA) protocol that was implemented in Ad hoc network environment. We built our own real-time simulator with wireless environment parameters such as zone size and node velocity. In order to gain simulation confidence, we run extensive number of experiments using different probabilistic distributions. Our conducted simulations of the two protocols showed a clear advantage of BSMA compared to RASC vis-à-vis the throughput and the association performance.**

***Index Terms*— Random medium access protocols, ad hoc networks, quality of service.**

I. INTRODUCTION

Random Medium Access Control (RMAC) is a class of techniques used when stations are sharing a transmission medium [4]. These techniques are characterized by the randomness in accessing a medium which leads to collision of simultaneously sent packets. In RMAC techniques, each node has the right to access the common channel without any control, depending on the state of medium (idle or busy) [4][7].

Wireless networks are set of devices connected without using cables. Wireless networks offer more flexibility and adapt easy to change in the configuration of the network. But when we have multicast connection the total throughput is affected [3]. The communication network consists of *m* users transmitting data to a central controller through a common wireless network. The three basic components of wireless network model are the user, the channel, and the central controller. The user uses the network model for communication (transmission and receiving data) in the form of equal-sized packets. Each user is associated with a single buffer. The channel is slotted so that the probability of accessing a slot depends only on the number of transmitted packets. This general model for multi-packet reception (MPR) channels applies to, as special examples, the conventional collision channel and channels with capture. The central controller controls the access to the common wireless channel [12].

There are different approaches of sharing a transmission medium [4]. For instance, in static channelization approach, the partitioned channels are dedicated to individual users, so no collision occurs. This technique is good for steady traffic and achieves efficient usage of channels. However, in dynamic medium access control, the incidence of collision is minimized to achieve reasonable usage of medium. This technique is good for bursty traffic, and it was implemented with scheduling or random access. In scheduling way, stations are scheduled in an orderly access of the medium. Some RMAC protocols focus on scheduling policies despite that the scheduling computation time increases with the network size [15]; nevertheless, scheduling performed well with heavier traffic. In random access way, stations try to perform transmission by accessing the medium hoping that there will be no collision. But in case of a collision, a random period is used to back off before trying to transmit again [7][12].

The MAC sub layer of the data link layer acts as an interface between the LLC sub layer and the network's physical layer [4]. The MAC layer utilises a full-duplex logical communication channel in a multi-point network. This channel may provide a unicast, multicast or broadcast communication service. It provides addressing and channel access control mechanisms that make it possible for several terminals or network nodes to communicate within a multi-point network, typically a local area network [4]. The performance of random access highly relies on the correlation property of random access code, which is affected by channel characteristics [14].

The rest of this paper is organized as follows. The RMAC protocols are discussed briefly in the second section. In Section 3, we present some of the state-of-the-art protocols such as BSMA and RASC and we layout their different characteristics and environment specifications. Section 4 summarizes our contribution in simulating the above-mentioned protocols, along with the analysis of the obtained results. Section 5 is the conclusion.

M. Omari, associate professor, is with LDDI laboratory, at the University of Adrar, Adrar 01000, Algeria (phone: 213-49-967571; fax: 213-49-967572; e-mail: omari@univ-adrar.org).

W. Hadj Fateh, BS in computer science, was with the University of Adrar, Adrar 01000, Algeria (e-mail: hadj_fateh_warda@yahoo.com).

Z. Kacemi, BS in computer science, was with the University of Adrar, Adrar 01000, Algeria (e-mail: kacemi_zohra@yahoo.com).

II. Random Medium Access Control

When a user wants to connect with another user using point-to- point connection, no collision occurs in the pathway. But when we have multipoint connection (common link between many devices) a collision might happen. In order to reduce collision effect, multiple access protocols are needed [6].

There are three types of multiple access protocols: the random access protocols, the controlled access protocols, and the channelization protocols [4]. Each type includes many protocols. In this paper, we focus on the first type in wireless domains.

A. Random Medium Access (RMA)

RMA contention methods are characterized by the absence of a controller: no station is entitled to permit, or being permit by another, before starting the transmission. At each instance, when a station has data to send, it should use a procedure defined by the protocol which gives decision for sending data. This decision depends on the condition of the medium, either idle or busy. Based on RMA techniques, every station can send data whenever it is ready; a simple reason to inherit the collision problem. Many protocols have been proposed to alleviate this problem [4].

B. Famous Random Medium Access Protocols: (RMAP)

The RMA protocols have evolved from a very interesting protocol known as ALOHA, which used a very simple procedure called multiple access (MA) [4][11]. ALOHA was implemented in different ways such as N-user slotted random access system and stabilized slotted Aloha [16].

The method was improved with the addition of a procedure that forces the station to sense the medium before transmitting. This was called carrier sense MA. This method later evolved into two parallel methods: carrier senses multiple access with collision detection (CSMA/CD) and carrier sense multiple accesses with collision avoidance (CSMA/CA). Some CSMA variations include slotted CSMA with mini-slots and unslotted CSMA [16]. CSMA/CD tells the station what to do when a collision is detected. The CSMA/CA tries to avoid the collision instead of detecting it.

In CSMA, each station senses the carrier before sending data and a station can start transmitting only when the carrier is idle. CSMA attempts to avoid collision by testing the signal. For instance, when we have three stations A, B and C, and A wants to connect with B, and C wants to transmit data to B, A and C sense carrier and find it idle. But when the data is received by B, a collision error happens. So, CSMA reduces the collision but it cannot eliminate it [4][11].

In wireless domain, the CSMA/CA is widely used. This protocol came to reduce the probability of collision. When the sender wants to send frame, he waits some time called distributed interframe space (DIFS) and then sends a frame request to send (RTS) to the receiver. Then, the receiver replies with a frame clear to send (CTS) after a short time called short interframe space (SIFS) [4]. In order to avoid collision while receiving data, the source/destination stations include the duration of time that they need to access the channel in RTS and CTS. Then other stations can avoid collision with network allocation vector (NAV) time which counts down time that other stations should stay idle before sensing the channel.

C. Hybrid MAC Protocols (HMAC)

HMAC protocols aim to combine the best of channel access schemes. They include scheduled and unscheduled periods in a slotted frame. The scheduled portion of the frame allows nodes to communicate without collision, distribute state information quickly and reliably, and guarantees a certain data rate available to each node in the network. On the other hand, the unscheduled portion allows nodes to adapt changing traffic conditions [6] [8].

All nodes can hear each other and listen to the channel at all times [11]. HMAC protocol schemes operate between the two extremes of scheduled and random access. The main distinction is that each scheduled slot is assigned to one node for a very long time, while the unscheduled slots will be used by various nodes in different frames. In order to avoid collision, the propagation delay is added to the time slot (long enough to transmit a maximum length packet) and each node will have a complete received packet before another node begins transmitting [8].

III. Related Work: BSMA and RASC Protocols

In this section, we present some protocols which try to solve some problems of RMAC such as quality of service (QoS), multi-packet reception (MPR), spread spectrum (SS), and the inherent dilemma of collision [9]. The key issue of these protocols is to coordinate transmission of all users.

A. Quality of Service (QOS)

QoS is a group of technologies for managing network traffic in a cost effective manner used to enhance user experiences for home and enterprise environments. QoS technology allows us to measure bandwidth, detect changing network conditions (such as collision or accessibility of bandwidth), and prioritize or throttle traffic [9]. For example, QoS technologies can be applied to prioritize traffic for latency-sensitive applications (such as voice or video) and to control the impact of latency-insensitive traffic (such as data transfers).

B. Multi Packet Reception (MPR)

Users in a wireless network share a common medium, and their transmissions may interfere with one to another. MPR node of the network is capable of correctly receiving signals from multiple transmitters. The MPR matrix is the tool used to describe the capability of the receiver to detect multi-packet simultaneously [2]. The key to maximizing throughput is to grant an appropriate subset of user's access to the MPR channel. For the conventional collision channel, this can be accomplished by splitting users in the event of collision. A more flexible approach is necessary for MPR channels because the protocol should allow the optimal number of users to transmit. This implies that the set of users accessing the channel should be enlarged if there were not enough users holding packets in the previous slot and shrunk if too many users attempted to transmit. Ideally, the approach should allow many users to perform transmission in order to achieve the maximum throughput. Unfortunately, this is not always possible because the number of users holding packets is a random variable not known to the receiver [10].

C. *Unicasting and Broadcasting in Ad hoc network*

Ad hoc random access MAC protocols deals with unicast and broadcast packets differently [2][5]. In fact, unicast packets are preceded by control frames of the MAC sub-layer. For instance, IEEE802.11 protocols uses collision avoidance along with RTS/CTS/ACK control frames to transmit unicast packets in order to combat hidden and exposed terminals. Broadcast packets are sent blindly without any control frames that can assure the availability of the destination, and without consideration of hidden and exposed terminals and channel noise.

D. *Broadcast Support Multiple Access protocol (BSMA)*

In Ad Hoc Networks, there are many random access protocols proposed in the literature. Other protocols have been proposed in the literature, such as Floor Acquisition Multiple Access (FQMA), and IEEE 802.11 protocols. Yet, none of these protocols were initially designed to support the reliable brodcasting of data. Broadcast packets are meant to be received by all neighbors of the source node randomly [2][11].

The BSMA Protocol:

First, the source sends `Request_To_Send` (`RTS`) packet to all neighbors and sets a timer `WAIT_FOR_CTS` (clear to send). Then, the source neighbors, upon receiving `RTS`, send `Clear_To_Send` (`CTS`) packet if not in `YIELD` state and set timer to `WAIT_FOR_DATA`.

If the source receives `CTS`, it sends `DATA` and sets a timer `WAIT_FOR_NAK` (negative acknowledgment). Otherwise, if no `CTS` is received, and the `WAIT_FOR_CTS` timer expires, the source starts all over at the first step. In addition, the nodes that are not involved in the broadcast exchange, upon receiving `CTS`, set their state to `YIELD` and set their timer long enough to allow for the broadcast exchange to terminate. On the other hand, the neighbors send `NAK` if `WAIT_FOR_DATA` timer expires and `DATA` has not been received.

When the source receives a `NAK` before `WAIT_FOR_NAK` timer expires, it starts all over at the first step. Otherwise, if no `NAK` was received and the `WAIT_FOR_NAK` timer expires, the broadcast is considered complete. In this case, the source starts all over again and get ready to transmit new `DATA` [11].

E. *Spread Spectrum (SS)*

Spread-spectrum techniques are methods by which a signal generated in a particular bandwidth are intentionally spread in the frequency domain [2]. These techniques are used for a variety of reasons, including the establishment of secure communications.

Most spread-spectrum signals use a digital scheme called frequency hopping, where the transmitter frequency is stable between all hops. The length of time that the transmitter leftovers on a given frequency between "hops" is known as reside time. A few spread-spectrum circuits employ continuous frequency variation, which is an analog scheme.

F. *Random Access Scheduling protocol (RASC)*

Random Access scheduling (RASC) is a protocol that uses seed exchange method. This seeds are used for pseudo-random generation. The RASC decomposes the network into independent clusters (groups) containing a single receiver. After that, the network resembles to the up-link of cellular networks whose medium access techniques are well developed [1][5].

In spread-spectrum networks, RASC works to avoids collisions and provides Quality-of-Service at the MAC layer. The throughput performance of the protocol is usually analyzed in fully connected networks using simulations.

The RASC Protocol:

The protocol organizes the hosts based on who wants to communicate with whom. Therefore, some of them are transmitters, others are receivers. First, a path is distinguished between every transmitter and its neighbors (receivers). Then, data is transmitted between a sender and one of his neighbours. In addition, the protocol states that a receiver is allowed to receive data only from one transmitter [1].

There are many different implementations of protocol, but only the perfect local scheduling/polling is considered in this paper. In perfect local scheduling, every receiver controls its associated neighbors and chooses those that are holding packets. A receiver can learn about the transmitters holding packets by employing `RTS/CTS` communication or by polling. Such implementations provide higher local throughput when the time spent for polling is neglected compared to that of data transmission [1].

IV. Simulation, Results, and Analysis

In our work, we used a simulation framework developed initially by Soliman and Omari [13] to evaluate security mechanisms such as WEP (Wired Equivalence Privacy) and SDES (Synchronous Dynamic Encryption System). We applied some updates to implement the two selected protocols (BSMA and RASC). The conducted simulation was built based on the Java programming language, which provides interesting features such as threads and TCP/IP connection libraries.

The initial simulator uses a number of base stations that receive data from their neighbor nodes, i.e., all nodes transmit data only to the nearest base station. Initially, each node associates with the nearest base station, then registers and starts sending data message. Then, the nodes may change position randomly, and may associates with a different base station in case it migrates from one cell to another. The simulator keeps track of the total data messages as well as the control messages of the implemented protocol.

A. The BSMA protocol simulation

1) Source nodes

In BSMA simulation, the source node broadcasts message to all nodes which are in range. Initially, the source node sends an `RTS` message and waits for a `CTS` message before start sending data. A `NAK` can also be received in case the neighbor node is not ready to receive data.

In the simulator, each source node is simulated by a thread as follows:

```
// start source node threads
for (int i= 0; i < simulationParameters.
  getNumberOfSourceNode(); i++) {

  SourceNodeThreads[i].start();
}
```

We added a new method to transmit `RTS` message from source node to its neighbours, and wait for some probabilistic time (thread becomes asleep) before receiving a `CTS`. The code of the `RTS` method is shown next:

```
// a method to send RTS and wait for CTS
void RTSmessage() throws Exception{

  // generate RTS message
  for (int i = Constant.PD_RANGEMIN;
    i < Constant.PD_RANGEMAX; i++)
    dataToSend[i] = (byte) (Math.random() * 256);

  // wait for some time
  Thread.sleep
      (Constant.DISTRIBUTION_DEFAULT_MEAN);

  // send the packet
  dataToReceive =
        sendAndReceiveData(dataToSend, port);
}
```

This method calls the `sendAndReceiveData` method in order to send data and wait for a reply. So, we altered this method in order to allow the source node to broadcast messages to all neighbour nodes within its range:

```
// a method to send data to all neighbor nodes
public byte[] sendAndReceiveData(byte[] dataToSend,
  int port) {

  // calculate distance
  distance =((int) Math. sqrt((Math. pow((Integer.
    parseInt (nodeInfo[i][Constant.NODE_X].
    toString())) -(Integer.parseInt
    (nodeInfo[Constant.SOURCENODE_X].toString())),
    2))+ (Math.pow((
    Integer.parseInt(nodeInfo[i][Constant.NODE_Y].
    toString()))- (Integer.
    parseInt(nodeInfo[Constant.SOURCENODE_Y].
    toString())), 2))));

  if (distance <= Constant.PD_RANGEMAX) {
    // create sendPacket
    sendPacket = new
    datagramPacket(dataToSend,dataToSend.length,
    InetAddress.getLocalHost(), Constant.PORT+i);

    // send packet
    socket.send(sendPacket);
  }
}
```

If the distance is less than the range perimeter of the source node, an `RTS` message is generated and sent. For statistic purposes, counters for different types of messages (`RTS`, data, total data messages) are created and incremented after each transmission.

In our simulation, nodes are allowed to move randomly in a specific area of 100 x 100. For the source node, there is no need to change the position (nodes are stationary). Next, we show the `run()` method that summarizes the protocol's mechanism:

```
  //start running the node's thread
  public void run() {
    // loop forever
    while (true) {

      // send RTS
      try {
          RTSmessage();
      } catch(Exception e) {
        e.printStackTrace();
      }

      if(NbOfReceiverCTS != 0){ // At least one CTS
        try {

          // generate messages
          generateMessages();
        } catch(Exception e) {
          e.printStackTrace();
        }
      }
      // change position
      changePosition();
    }
  }
```

2) Receiving nodes

A receiving node is a node where one of its neighbor nodes is a source node. In our simulator, we create these nodes as follows:

```
// start node threads
for (int i= 0; i < simulationParameters.
    getNumberOfNodes(); i++) {

  nodeThreads[i].start();
}
```

The receiving nodes should wait for packets to arrive, then start processing them using the `processPacket()` method. This method verifies the type of the received packet. So, if the received packet is `RTS`, the node replies with a `CTS` message (ready to receive) or a `NAK`. The receiving nodes change their position randomly by method called `changePosition()`.

Next, we show the `run()` method that summarizes the general task of a receiving node:

```
//start the node Thread
public void run() {
  // loop forever
  while (true) {
    // wait for packets
    receivedData = waitForPackets();

    // process packet
        try {
```

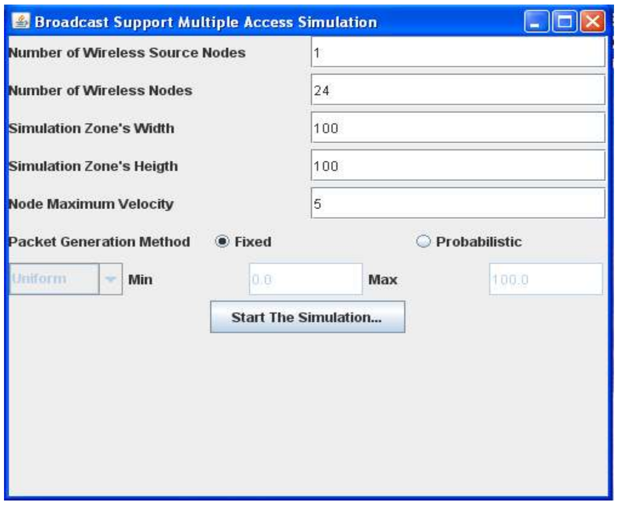
```
        dataToSend = processPacket(receivedData);
  } catch (Exception e) {
  }

  // send packet back
  if (receivedData[Constant.PH_TYPE] !=
    Constant.NO_REPLY)

    replyPacket(dataToSend);
  }
}
```

3) *BSMA simulator interfaces*

Fig. 1 represents the first interface of the BSMA protocol simulator:

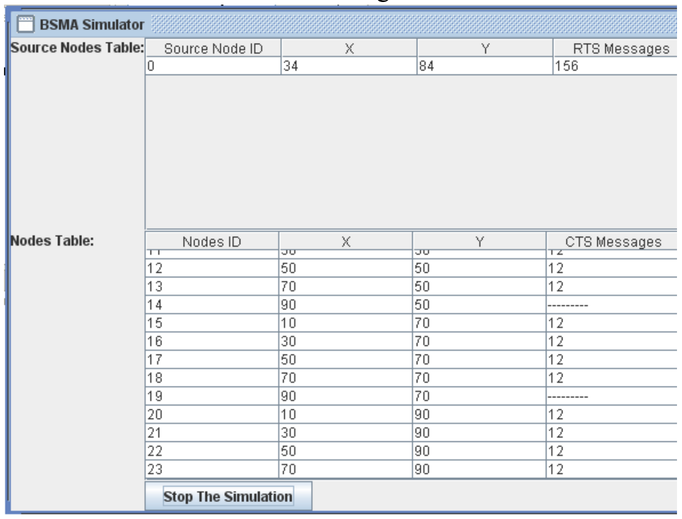


Broadcast Support Multiple Access Simulation

Number of Wireless Source Nodes	1
Number of Wireless Nodes	24
Simulation Zone's Width	100
Simulation Zone's Heigth	100
Node Maximum Velocity	5
Packet Generation Method	[✓ Fixed] Probabilistic
Uniform	Min 0.0 Max 100.0

Fig. 1: BSMA Simulator interface.

When the simulation parameters are specified, the simulation is started as shown in Fig. 2:



BSMA Simulator

Source Nodes Table:

Source Node ID	X	Y	RTS Messages
0	34	84	156

Nodes Table:

Nodes ID	X	Y	CTS Messages
12	50	50	12
13	70	50	12
14	90	50	---------
15	10	70	12
16	30	70	12
17	50	70	12
18	70	70	12
19	90	70	---------
20	10	90	12
21	30	90	12
22	50	90	12
23	70	90	12

Fig. 2: BSMA Simulation.

The simulation experiments show, in real time, the position of each node (source or receiver nodes), the number of `RTS` messages at the source node, the number of `CTS` messages at the receiving source node, and the volume of sent and received data.

B. *The RASC simulation*

1) *Simulation of transmitter nodes*

In RASC protocol, the transmitter nodes are responsible for generating initiating the communication. Therefore, each source node is simulated by a thread as follows:

```
  // start transmitter nodes threads
  for(int i = 0; i < simulationParameters.
    getNumberOfTransmitterNodes(); i++) {

    transmitterNodeThreads[i].start ();
  }
```

The association of a transmitter node with a unique receiver node is implemented through a new method `associateWithNearestBaseStation()`. In order to avoid collision, a transmitter node rejects any further request for association with other receivers.

In order to determine the requested type service, the transmitter sends an `RTS` to the associated receiver, and then the latter replies back with a `CTS` message in order to start transmitting data. The transmitter nodes change their position randomly by method called `changePosition()`.

The main function `run()` that implements the transmitter's tasks is shown next:

```
//start running the node's thread
public void run() {
  while (true) {

    // associate
    associateWithNearestBaseStation();

    try {
      //send RTS message
      RTSmessage();
    } catch(Exception e) {
    }

    if(NbOfReceiverCTS!=0){

      try {
        // generate messages
        generateMessages();
      } catch(Exception e) {
      }
    }

    // change position
    changePosition();
  }
}
```

2) *Simulation of Receiver Nodes Thread*

In our simulation, we implemented the RASC protocol so that each node receives data from only one associated transmitter. The receiver nodes change their position randomly with the maximum velocity that is introduced as a simulation parameter.

```
//start the node Thread
public void run() {
  // loop forever
  while (true) {
    // wait for packets
    receivedData = waitForPackets();

    // process packet
```

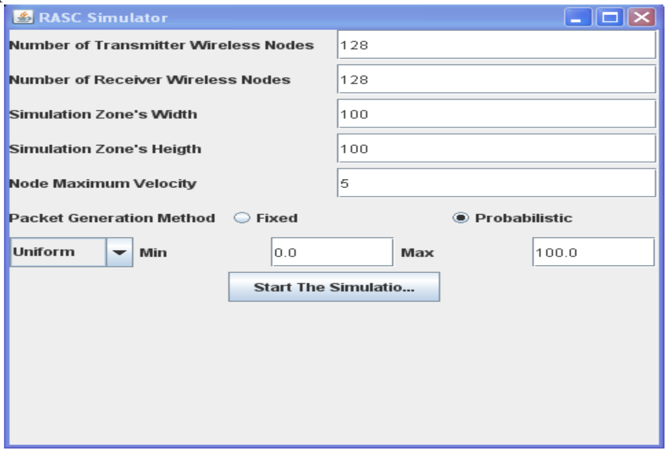
```
        try {
          dataToSend = processPacket(receivedData);
      } catch (Exception e) {
      }

      // send packet back
      if (receivedData[Constant.PH_TYPE] !=
        Constant.NO_REPLY)

        replyPacket(dataToSend);
      }
      // change position
      changePosition();
}
```

3) *RASC simulator interface*

Fig. 3 shows the RASC simulator interface of the BSMA protocol:



RASC Simulator

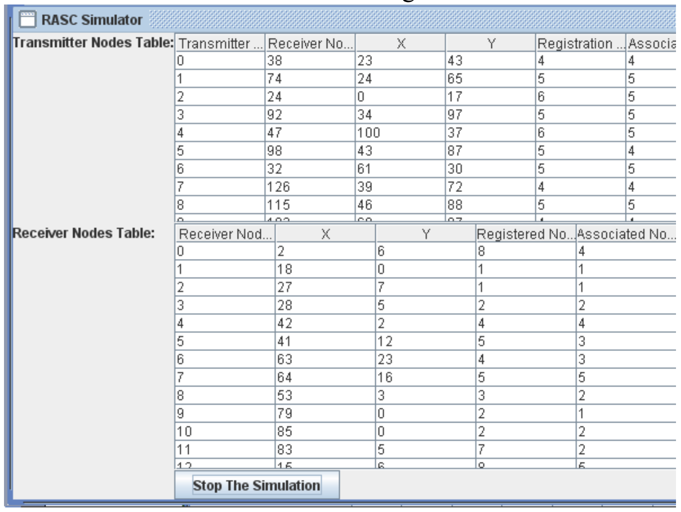
Number of Transmitter Wireless Nodes	128
Number of Receiver Wireless Nodes	128
Simulation Zone's Width	100
Simulation Zone's Heigth	100
Node Maximum Velocity	5

Packet Generation Method ○ Fixed [● Probabilistic]

Uniform Min 0.0 Max 100.0

Fig. 3: RASC Simulator interface.

When the "Start the Simulation" button is hit, the simulation is launched as shown in Fig. 4:



RASC Simulator

Transmitter Nodes Table:

Transmitter ...	Receiver No...	X	Y	Registration ...	Associa
0	38	23	43	4	4
1	74	24	65	5	5
2	24	0	17	6	5
3	92	34	97	5	5
4	47	100	37	6	5
5	98	43	87	5	4
6	32	61	30	5	5
7	126	39	72	4	4
8	115	46	88	5	5

Receiver Nodes Table:

Receiver Nod...	X	Y	Registered No...	Associated No...
0	2	6	8	4
1	18	0	1	1
2	27	7	1	1
3	28	5	2	2
4	42	2	4	4
5	41	12	5	3
6	63	23	4	3
7	64	16	5	5
8	53	3	3	2
9	79	0	2	1
10	85	0	2	2
11	83	5	7	2

Fig. 4: RASC Simulation.

C. *Hardware specification*

Our simulation of the RASC and the BSMA protocols were conducted using the hardware specification as shown in Table 1:

TABLE I
HARDWARE SPECIFICATIONS OF THE SIMULATION MACHINE

Hardware component	Characteristic
Memory RAM	1 GB
Microprocessor	Pentium 4, 3.00 GHz

D. *The experiments*

In this section, we will lay out the steps to compare the two selected protocols (BSMA and RASC). For each simulation instance we calculate the amount of received data of all nodes, and then divide it by the number of nodes in order to get the proportional data per each node. These experiments were run several times to gain high simulation confidence. The experiments are also run with deferent probabilistic distributions of packet generation (uniform, exponential, Gamma, and pareto) in order to get more solid and accurate results. The simulation time was set to several minutes for all our conducted experiments. Table 2 shows the different parameters used in the simulation along with their range values:

TABLE II
INPUT PARAMETERS FOR CONDUCTED SIMULATIONS OF BSMA AND RASC

Parameter	Value	Observation
Number of Source Nodes	1	(BSMA)
Number of Nodes	Variable	(BSMA) 5, 10, 20, …
Number of Transmitter	Variable	(RASC) 5, 10, 20, …
Number of Receiver	Variable	(RASC) 5, 10, 20, …
Zone Height	100	(RASC and BSMA)
Zone Width	100	(RASC and BSMA)
Velocity	5	(RASC and BSMA)
Distribution	Variable	(RASC and BSMA) Uniform, Gamma, Exponential, pareto, and logarithmic

E. *Results and Analysis:*

In the first phase we compared the data rate and the performance when varying the number of nodes. For better analysis, deferent distributions of packet generation are used.

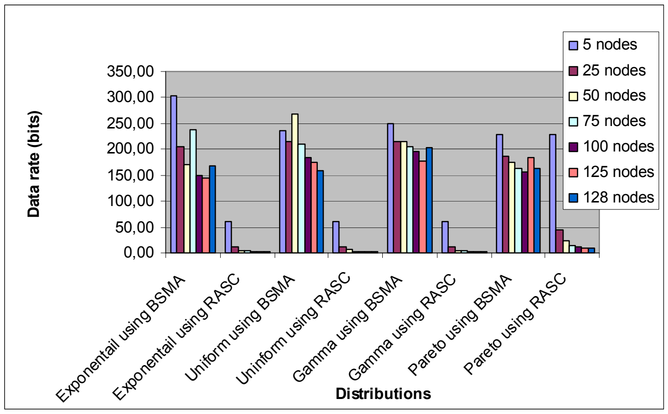


Fig. 5: Data rate comparison between BSMA and RASC.

Fig. 5 shows in general that the data rate decreases when the number of nodes increases. Yet, BSMA protocol simulation showed a higher data rate compared to RASC, even with different number of nodes, at different distributions. Results also show that BSMA data rate decreases slowly when the number of nodes increases, while in RASC, the data rate decreased dramatically.

In order to explain these results, we should recall that in BSMA protocol the source node broadcasts data to its neighbour nodes; so whenever the number of nodes increases, source node waits for more `CTS` messages from its neighbors, which clearly affects the time to send data. In case of RASC, the loss is greater since increasing the number of nodes affects directly the search time to associate with the nearest receiver; thus, the data rate is decreased dramatically.

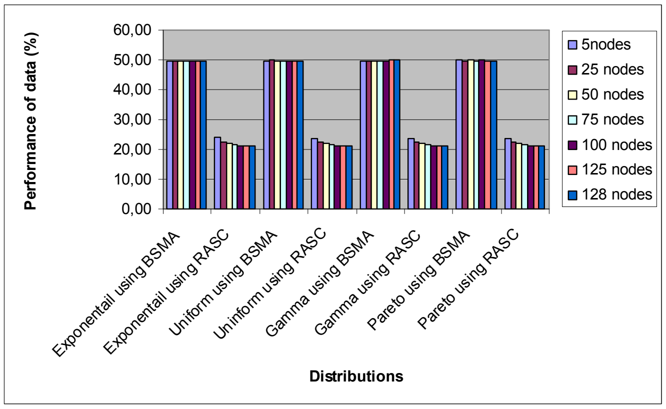


Fig. 6: Performance comparison between BSMA and RASC.

Fig. 6 shows that the performance of the data rate is stable in both protocols, even with different probability distribution of packet generation. Yet, experiments showed a higher performance of BSMA protocol compared to RASC protocol.

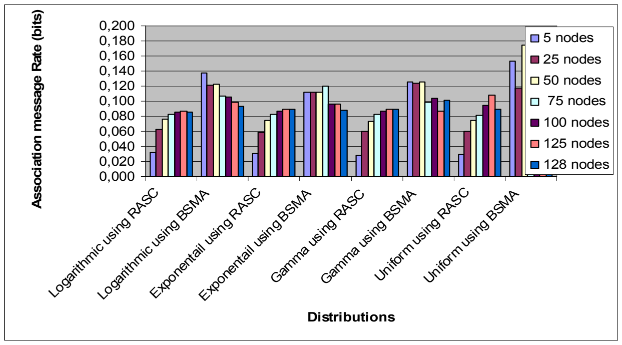


Fig. 7: Association message rate comparison between BSMA and RASC.

In the second phase, we compared the association message generation rate in both protocols when varying the number of nodes. Fig. 7 shows that BSMA has a higher association rate compared to RASC. This is because, in BSMA, a source node associates with many neighbors (sending `RTS` message), whereas in RASC, a transmitter associate with one receiver only. Moreover, in RASC, a transmitter node has to de-associate with the current receiver node before associating with a new node, which results in more wasted time.

V. Conclusion

Random medium access protocols are widely used to alleviate the problem of multiple channel access. In Ad hoc networks, this problem becomes more disrupting since wireless channels are accessed randomly and dynamically by mobile stations. Many studies have been conducted to enhance the performance of the mobile station's throughput and reducing the power of packet transmission.

In this paper we selected two state-of-the-art random-access protocols (BSMA and RASC) for implementation and comparison purposes. The BSMA protocol is characterized by broadcasting packets from one source node to all its neighbors in the range. On the contrary, The RASC protocol is unicast, where each node transmitter is associated with the nearest receiving node only.

Our experimental results showed a better data rate and throughput performance of BSMA compared to that of RASC. Moreover, we found that BSMA is more scalable that RASC when the number of nodes increased. In terms of control packets, association messages were reduced in BSMA due to the unique association between neighbor nodes. In fact, increasing the number of nodes leads to decreasing the association which clearly affects the overall performance. As a future work, we propose implementing BSMA and RASC in a multicasting and broadcasting environment, where one-to-many associations becomes mandatory between nodes.

References

[1] G. Mergen and L. Tong, "Random Scheduling Medium Access for Wireless Ad Hoc Networks," *in the Proceedings of IEEE MILCOM 2002 Military Communications Conference*, Anaheim, CA, October 2002.

[2] Q. Zhao and L. Tong, "A Multiqueue Service Room MAC Protocol for Wireless Networks with Multipacket Reception," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, February 2003.

[3] K. Gorantala, "*Routing Protocols in Mobile Ad Hoc Network*," Master's Thesis in Computing Science, June 15, 2006.

[4] B. A. Forouzan, "Data communications and Networking," Fourth Edition, *Mc Graw-Hill*, 2006.

[5] L. Tong, V. Naware, and P. Venkitasubramaniam, "Signal processing in random access," *IEEE Signal Processing Magazine*, vol. 21, no. 5, pp. 29–39,. Sep. 2004

[6] C. H. Foh and M. Zukerman, "A New Technique for Performance Evaluation of Random Access Protocols," *in the Proceedings of IEEE ICC 2002,* vol. 4, April 2002, pp. 2284-2288

[7] A. T. Koc, M. Torlak, "Simulation based Analysis of Random Access CDMA Networks," *Journal of Naval Science and Engineering*, vol. 2, no. 2, pp 49-76, 2004.

[8] K. B. Kredo II and P. Mohapatra, "A Hybrid Medium Access Control Protocol for Underwater Wireless Networks," *in proceedings of the ACM International Conference on Underwater Networks*, pp 33-40, September 2008.

[9] R. Majoor, "Quality of Service in the Internet Age," *in the preceedings of the SATNAC conference*, 2003.

[10] L. Tong, Q. Zhao, and G. Mergen, "Multipacket Reception in Random Access Wireless Networks: From Signal Processing to Optimal Medium Access Control", *IEEE Communications Magazine*, pp 108-112, November 2001.

[11] K. Tang, M. Gerla, "Random Access MAC for Efficient Broadcast Support in Ad Hoc Networks," *in the preceedings of IEEE WCNC*, 2000.

[12] R. Lin and A. P. Petropulu, Senior Member, IEEE, "A New Wireless Network Medium Access Protocol Based on Cooperation," *in IEEE*

Transactions on Signal Processing, vol. 53, no. 12, pp 4675-4684, December 2005.

[13] H. S. Soliman and M. Omari. "An Efficient Application of a Dynamic Crypto System in Mobile Wireless Security," *in the proceedings of IEEE Wireless Communications and Networking Conference*, Atlanta, Georgia, March 2004.

[14] D. H. Lee and H. Morikawa. "Analysis on random access process of single carrier FDMA system," *in the proceedings of the 3rd International Conference on Wireless Internet (WICON '07)*, October 2007.

[15] C. Joo and N. B. Shroff. "Performance of Random Access Scheduling Schemes in Multi-Hop Wireless Networks," *IEEE/ACM Transactions on Networking (TON)*, Vol. 17, issue. 5, pp 1481-1493, October 2009.

[16] J. Paek and M. J. Neely. "Mathematical Analysis of Throughput Bounds in Random Access with ZIGZAG Decoding," *Mobile Networks and Applications Journal*, vol. 16, issue 2, pp 255-266, April 2011.

Mohammed Omari received his B.E degree from the University of Es-Senia Oran (Algeria) in 1995 and the M.S. degree from New Mexico Tech (New Mexico, USA) in 2002, as well as the Ph.D. degree in 2005. His major is in computer science. He is currently an Associate Professor at the computer science department, and a unit president at the LDDI laboratory (University of Adrar, Algeria). His research interests include network security, cryptography, sensors and ad hoc protocols, image processing, neural networks, and genetic algorithms.

Warda Hadj Fateh received her B.S degree in computer science from the University of Adrar, Algeria, in 2011. Her research interests are in ad hoc networks and their performance analysis.

Zohra Kacemi received her B.S degree in computer science from the University of Adrar, Algeria, in 2011. Her research interests are in ad hoc networks and their performance analysis.