Journal of Computer and Communications, **2013, 1, 33-40**
http://dx.doi.org/10.4236/jcc.2013.14006 Published Online October 2013 (http://www.scirp.org/journal/jcc)

Performance Analysis of an Autoconfiguration Addressing Protocol for Ad Hoc Networks

Hassan Al-Mahdi[1*], Hamed Nassar[2], SafaAbd El-Aziz[2]

[1]Department of Computers & Information, Faculty of Sciences and Arts, Al Jouf University, Al Quryyat, Saudi Arabia; [2]Department of Computer Science, Faculty of Computers and Informatics, Suez Canal University, Ismailia, Egypt.
Email: *hassanwesf@ju.edu.sa

Received September 28th, 2013; revised October 18th, 2013; accepted October 26th, 2013

ABSTRACT

Autoconfiguration protocols are important in maintaining mobile ad hoc networks (MANETs). In this paper, we present an autoconfiguration protocol called the one-step addressing (OSA) protocol characterized by its simplicity. Mathematical model is developed to evaluate the performance of the proposed protocol. The results of the model, which are validated by the simulation results, show that the OSA protocol outperforms the well-known token-based protocol in terms of both latency and communications overhead.

Keywords: IP Address; Ad Hoc; Performance Evaluation; Simulation; Markov Chain

1. Introduction

A mobile ad hoc network (MANET) is a group of mobile nodes which cooperatively and spontaneously form an IP-based network with no centralized administration [1]. A node communicates with other nodes either directly if they are within its transmission range or indirectly, using a multi-hop route through other nodes, if they are beyond that range. Various protocols have been developed to assign IPs to newly arriving nodes. The efficiency of such protocols is evaluated by two metrics: address allocation latency and communications overhead. Latency is defined as the time taken from instant when a node requests an address to that when it is assigned the address. Communications overhead is defined as the number of control packets transmitted during the address assignment process.

In this paper, we devise and analyze a new protocol-the one step address (OSA) protocol. We analyze the OSA protocol using a mathematical model whose results are validated by a discrete-event JAVA simulation program. We compare the efficiency of the proposed protocol with that of the token-based address allocation protocol and show that the OSA protocol is superior.

The rest of the paper is organized as follows. In Section 2 related research efforts are introduced. In Section 3, the description of the OSA protocol is provided. In Section 4, the mathematical model and measurements of the proposed protocol are presented. In Section 5, we validate the model by the simulation and compare its efficiency with the token-based protocol. In the last section, we draw conclusions.

2. Related Work

IP address autoconfiguration protocols in MANETs could be classified as stateful or stateless according to the management of the address space. All stateful protocols such as MANETconf [2], LHA [3], Token-based [4] and Prophet address allocation [5] maintain address allocation tables to monitor the assigned and free IPs, so existing nodes can easily assign unused IPs to requesting nodes. The challenge for stateful protocols is to synchronize the allocation tables. The advantage of stateful protocols is the duplication-free IP assignment. All stateless protocols such as IPAA [6], WDAD [7] and PDAD [8] are characterized by auto allocation of IPs, which means that each node randomly chooses its IP. Then the node should perform a mechanism for duplicate address detection to insure that its chosen IP is unique within the network. The challenge in stateless approaches is to detect in moderate latency and communications overhead, the potential address duplication. The advantage of stateless protocols is their relative simplicity compared to stateful protocols.

*Corresponding author.

In the IP Address Autoconfiguration (IPAA) protocol [6], a trial and error policy to assign an IP to a new node is adopted. Basically, the new node selects a random IP and floods a request with the selected IP to check if a node is using the same IP. If no reply is received, the node considers the IP free, and takes it as its own. Otherwise, the new node tries another IP and repeats the process. This protocol incurs high latency and communications overhead due to the random selection of the IP. Additionally, there is a possibility that a reply indicating that the IP is in use gets lost, resulting in a duplication of that IP.

In the MANETconf protocol [2], each node maintains two sets, the set of IP addresses already assigned and the set of IP addresses pending assignment. When a new node wishes to join the network, it floods a message to its neighbors to choose an agent. The agent selects an arbitrary IP that is neither assigned nor pending assignment, adds it to the set of IP addresses pending assignment, floods the IP to all other nodes, and starts aTimer. If a recipient node finds this IP in one of its sets, it sends a negative reply. Otherwise it adds it to its set of IP addresses pending assignment and sends an affirmative reply. If *all* nodes reply, and all replies are affirmative, the agent assigns this IP to the new node, adds the IP to its assigned set, and floods this information in the MANET so all nodes add this IP to their sets of assigned IPs. If at least one reply is negative, the agent selects another IP and the procedure is repeated. If at least one node does not reply, the agent sends the message again to them. The efficiency of MANETconf is higher than that of the IPAA, but is more sensitive to loss rate since it insists on replies from *all* nodes [4].

In the token-based protocol [4], the address allocation procedure is similar to that of MANETconf except that it doesn't require global agreement. Specifically, at any point in time there is a single node currently holding the token, called the allocator, which has the right to assign IPs. The efficiency of the token-based protocol is higher than that of the MANETconf protocol as shown in [4], but there is the problem of a single point of failure if the allocator is disabled for any reason.

In the Logical Hierarchical Addressing (LHA) protocol [3], each node can act as an agent. The agent uses a function that generates IPs within a subspace of the total MANET address space. When a new node wishes to join the MANET, it selects one of its neighbors as an agent. The agent generates an IP and assigns it to the new node. If the agent has already generated all the IPs in its subspace, it selects one of its neighbors to do the job. Clearly this protocol has some drawbacks. First, the IP generation function moves forward every time it is invoked, depleting its subspace at some point. Second, if a node that has been assigned an IP leaves the MANET, its IP remains unused.

In the Prophet [5], the authors tried to design a function that produces a sequence of IPs. The initial state of the function is called the seed. Different seeds lead to different sequences. This protocol works as follows: The first node in the MANET chooses a random number as its IP and uses a random state value or a default state value as the seed for its function. When a new node joins the MANET, it chooses an agent to get a free IP. The agent uses the function to generate an IP which is sent to the new node with the agent state. The new node uses the agent state as the seed for its function. The agent then updates its state accordingly. Prophet address allocation has obvious advantages such as: low latency time and low communications overhead. But it has also drawbacks such as: the function used cannot avoid address conflicts as each node produces a sequence in the same domain and two or more of the same numbers can be generated from the multiple sequences. Uniqueness is guaranteed only in one sequence, not among multiple sequences.

In the Weak Duplicate Address Detection (WDAD) [7], each node generates a key at the initialization phase, and distributes it with its IP address in all routing messages. This key will be used to detect duplicate IP addresses. Each node maintains keys along with IP addresses in its routing table. When a node receives a routing message with an IP address that exists in its table, it checks whether the keys are different or not. If they are different, a duplicate address is detected and the entry is marked as invalid and additional steps are taken to inform other nodes about this duplication. The main drawback of WDAD is its dependency on the routing protocol. WDAD detects address duplication based on local routing information, thus it is totally adapted to proactive routing where each node maintains a complete routing table. For reactive routing, it is not the case; the nodes cache partial routing information for only ongoing and relayed connections which reduces the possibility of detecting in moderate delays address duplication. For the overhead, WDAD requires no additional traffic for the auto-configuration mechanism, but the price is traffic overhead caused by the integration of the key value in routing packets.

Passive Duplicate Address Detection (PDAD) [8] is a duplicate detection mechanism designed for link state routing protocols. The idea behind PDAD is that instead of explicitly trying to detect and solve address duplication by sending control information, each node can investigate routing information and deduce address duplication from events that occur if there are address duplicates. With proactive routing, the nodes periodically flood the network to inform other nodes about their neighborhood. These control packets contain sequence numbers to distinguish between fresh and old packets.

Based on these information, PDAD analyzes incoming routing packets to detect address duplicate. Sequence numbers are increased with each packet, and reset occurs once in a long period of time. Normally, a node should not receive a message with its IP as the source address and a sequence number greater than its own counter value. Accordingly, if it receives such a packet an address conflict had been detected. The advantage of this protocol is that no additional overhead is generated; but it requires complex analysis of the routing information and is applicable only for proactive routing environment.

3. Protocol Description

The OSA protocol is stateful. Each node stores two records and a table as shown in **Figures 1-3**. First, **Figure 1** shows the Parameters Record with three fields:

- IP_NODE: IP address of the node.
- ID1_NODE, ID2_NODE: two integers that together uniquely identify the node.

Second, **Figure 2** shows the Address Table, with m rows, with four fields:

- IP: IP address generated by the node.
- ID1; ID2: two integers generated by the node and unique for each IP.
- U: flag to indicate whether the corresponding IP is used.

Third, **Figure 3** shows the Borrowed Address Record with three fields:

- IP_BOR: Borrowed IP address.
- ID1_BOR; ID2_BOR: two integers borrowed by the node and unique for each IP.

In the OSA protocol, the MANET starts with a single node initiating the configuration process, where other nodes can subsequently join and leave the MANET. Each node can generate up to *m* IPs to be granted to nodes requesting to join the MANET. When a new node wishes to join the MANET, it senses the medium for beacon messages from other nodes. When the timer expires without receiving any beacon, the node repeats the process up to T attempts and one of two scenarios can take place.

IP_NODE	ID1_NODE	ID2_NODE

Figure 1. Parameters Record.

IP	ID1	ID2	U
IP_1	$ID1_1$	$ID2_1$	U_1
IP_2	$ID1_2$	$ID2_2$	U_2
⋮	⋮	⋮	⋮
IP_m	$ID1_m$	$ID2_m$	U_m

Figure 2. Address Table.

IP_BOR	ID1_BOR	ID2_BOR

Figure 3. Borrowed Address Record.

Scenario 1: If all the attempts fail, the node concludes that there is no MANET, and that it has to start one itself. First, it fills out the Parameters Record as follows. The IP_NODE is set to the first IP in the space, e.g. 192:168:0:1, and the two integers ID1_NODE; ID2_NODE are set to 0. Second, the node employs these values, using Equations (1)-(3), to fill out the Address Table. Third, the node fills out the Borrowed Address Record by setting all its fields to 0.

Scenario 2: If the node finds a MANET in place, by receiving a beacon message, it broadcasts an Address Request (Add_Req) message up to $\mathcal{F}$ attempts in order to get a reply. Upon receiving this message, each neighbor node responds by sending an Address Reply (Add_Rep) message containing the number of available IPs in it. The node selects the responder with the largest number of available IPs as an agent and ignores all other responses. If all responders have the same number of available IPs, the node randomly selects one of the responders as an agent and ignores all other responses. The node then sends a unicast Address Selection (Add_Sel) message to the chosen agent. Upon receiving Add_Sel, the agent searches its Address Table for an unused IP (*i.e.* U = 0). If it finds such an IP, it copies its record in the table (less the U field) into an Address Confirmation (Add_Conf) message which it sends to the node, then sets the U field to 1. Else, if it does not find such an IP, it copies instead into Add_Confits Borrowed Address Record.

Upon receiving the Add_Conf message, the new node copies the record it contains into its Parameters Record. Then it uses the latter to fill out its Address Table, using Equations (1)-(3). Finally, the new node sets all the fields of its Borrowed Address Record to 0.

Now, the agent checks the number of unused IPs in its Address Table. If this number is greater than 0, meaning that the agent still has unused IPs, no action is taken. Else, if this number is 0, meaning that the agent has run out of IPs that can be granted to new nodes, it sends a unicast Address Borrow (Add_Borr) message to the node it has just granted the IP, seeking to borrow an IP from the latter. Upon receiving this message, the new node replies with a unicast Address Confirmation (Add_Conf) message containing the first record in its Address Table (less the U field) and then sets the U field to 1. Upon receiving the Add_Conf message, the agent copies the recordit contains into its Borrowed Address Record.

Suppose that a node has left the MANET. The IP of that node should be reclaimed by the agent that granted it. The OSA takes care of this by regularly updating the U field in the Address Table of each node using proactive routing protocols such as [9] or reactive routing protocols

such as [10,11]. Each node periodically checks the existence of the IPs in its Address Table by checking routing table if proactive routing protocols are used or by establishing paths to them if reactive routing protocols are used. If an agent finds that a particular IP no longer exists, the agent reclaims it by marking it in its Address Table as free.

Suppose that a reclaimed IP is of an agent that assigned a number of IPs to other nodes. Then if another node obtains this IP, after it has been reclaimed, that node will create the same IPs in its Address Table and may later assign them to other nodes—resulting in duplication. To overcome this problem, a new node that has just obtained a reclaimed IP from an agent and created its Address Table should check if the IPs of its Address Table are in use in the MANET (using the routing protocol). If the new node finds a path to an IP in its Address Table, the new node detects the existence of that IP and marks it in its Address Table as used. If the new node finds out that all IPs in the Address Table are already in use, then it will not be able to act as an agent, temporarily, until it finds IPs that have been freed due to nodes leaving the MANET.

Address Table: The entries of the Address Table of a node are generated from the Parameters Record of that node as follows.

$$\mathrm{ID1}_i = \mathrm{ID1\_NODE} + 1,\ \ i = 1,2,\cdots,m. \tag{1}$$

$$\mathrm{ID2}_i = \begin{cases} m \times \mathrm{ID1\_NODE} + 1, & i = 1,2,\cdots,m \\ \mathrm{ID2}_{i-1} + 1, & i = 1,2,\cdots,m \end{cases} \tag{2}$$

and

$$\mathrm{IP}_i = \sum_{j=0}^{\mathrm{ID1\_NODE}} m^j + m \times \mathrm{ID2\_NODE} + Z_i + 1 \tag{3}$$

where $i = 1,2,3,\cdots,m$ and $\left(Z_i = vZ_{i-1} + \kappa\right) \bmod m$ is Linear Congruential Generator (LCG). The integers v, Z_0 and κ are arbitrary integers but should each be less than m. The LCG is known [12] to generate a unique sequence Z_i. As for the U field, its value is 1 if the corresponding IP is in use, and 0 otherwise.

OSA Example

To illustrate the OSA protocol, we give an example in **Figure 4**, where the number m of IPs generated by each node is 2. In the Figure, we show the stages of how nodes are assigned their IPs. Initially, there is a node with the configuration shown in part **(a)** of the Figure. Assume now that this node has been selected as an agent by a new node. Then the new node will obtain its Parameters Record from the agent, and uses it to fill out its Address Table, and finally fill out its Borrowed Address Record. This configuration is shown in Part **(b)**. Now, the agent labels the U field of the IP it has granted the new node as used (set U = 1), as shown in Part **(c)**. In part **(d)**, we assume that a second new node has arrived and selected the same agent. Then the second new node will obtain its Parameters Record from the agent, and uses it to fill out its Address Table, and finally fill out its Borrowed Address Record. Again, the agent labels the U field of the IP it has granted the second new node as used (set U = 1), as shown in Part **(e)**. Since now the agent has no available IPs, it borrows an IP from the last node it has granted an IP (second new node). The resulting configuration is shown in Part **(f)**. Finally, the second new node labels the U field of the IP that has been borrowed by the agent as used (set U = 1) (not shown in the Figure).

4. Performance Analysis of OSA

In this section we develop a mathematical model to estimate the efficiency of OSA protocol. The model, which characterizes this protocol as functions of the number of nodes, packet loss rate, and the number of available addresses, is based on these assumptions.

1) The topology of the MANET is square with side d, assumed to be much larger than the transmission range of a node (as assumed in [4]).

2) The average proportion of the area that can be reached by a node to the total network area is β. Hence, the probability that a node is out of range of a given node is $(1-\beta)$ where β is derived as a function of a transmission range of a node r and d.

3) Number of addresses (hence, nodes) that the MANET can have is N.

4) The loss rates of broadcast and unicast messages between two nodes are s_b and s_u, respectively.

5) The average delays of one hop broadcast and unicast messages are t_b and t_u, respectively.

6) The arrival process of new nodes at the MANET is assumed to be Poisson with rate λ nodes per unit time.

7) The time a node stays in the MANET is exponentially distributed with expectation $\frac{1}{\mu}$.

It can be seen that the probability α that a new node does not successfully receive a reply message from a given neighbor in its transmission range is given by

$$\alpha = s_b + (1 - s_b)s_u \tag{4}$$

Let $P(t) = 0,1,2,\cdots,N$ be a RV denoting the number of nodes within the transmission range of a given node at instance t. The probability in steady state that i nodes are within the transmission range of a given node can be obtained as follows.

$$\begin{aligned} p_i &= \lim_{t\to\infty} \Pr\left[P(t) = i\right] \\ &= \binom{N}{i} \beta^i (1-\beta)^{N-i},\ \ i = 0,1,2,\cdots,N \end{aligned} \tag{5}$$

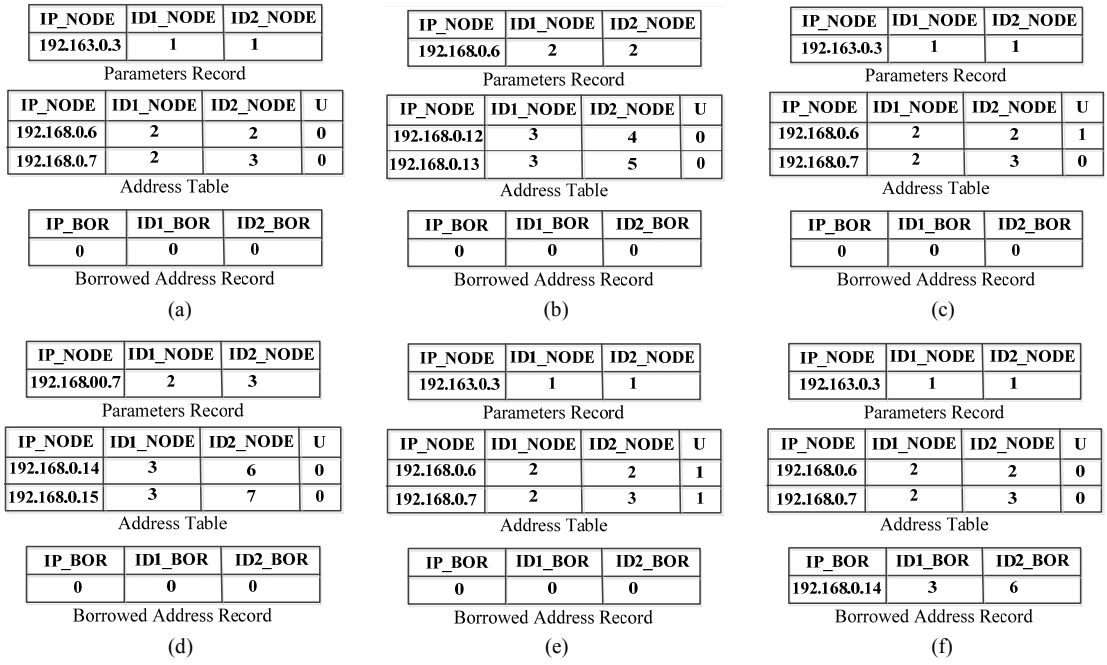

Figure 4. (a) Agent information initially; (b) First Requester information after obtaining its Parameters Recorded from agent; (c) Agent information after granting an IP to First Requester; (d) Second Requester information after obtaining its Parameters Record form agent; (e) Agent information after granting an IP to Second Requester; (f) Agent information after borrowing an IP from Second Requester.

When a new node arrives at the MANET, it broadcast Add_Req message to select an agent. All nodes within its transmission range will reply with Add_Rep messages. Let $D(t)$ be a RV denoting the number of Add_Repmessages arriving at the new node at time t with distribution $d_j(t), j = 0,1,2,\cdots,N$. In steady state, the RV D depends on the value of the RV P with the following conditional distribution

$$\begin{aligned} d_j &= \lim_{t\to\infty}\sum_{i=j}^{N}\Pr\left[D(t)=j\middle|P(t)=i\right]\Pr\left[P(t)=i\right] \\ &= \sum_{i=j}^{N}\Pr\left[D=j\middle|P=i\right]\Pr\left[P=i\right] \\ &= \sum_{i=j}^{N}\binom{N}{i}\binom{i}{j}\beta^{i}(1-\beta)^{N-i}(1-\alpha)^{j}\alpha^{i-j} \end{aligned} \tag{6}$$

4.1. Latency Time

As indicated above, upon the arrival of a new node, two possible scenarios are distinguished:

scenario 1: The node is the first in the MANET. Then the latency time is the period τ the node waits sensing beacon messages and failing to get one. In such a case, the latency time W_1 is given by

$$W_1 = \tau \tag{7}$$

scenario 2: The node arrives at an existing MANET. Then, the latency time is made up of two parts: the time till it finds an agent and the time till it obtains an IP from that agent. For the first part, the node will make up to $\mathcal{F}$ repeated attempts, as explained above, to get a reply. Let $A_1 = 0,1,2,\cdots,\mathcal{F}$ be a RV denoting the number of successful attempts. Then, A_1 has the distribution

$$\Pr\left[A_1 = i\right] = \frac{d_0^{i-1}(1-d_0)}{1-d_0^{\mathcal{F}}}$$

Then, the expected value of A_1 is given as

$$E\left[A_1\right] = \sum_{i=1}^{\mathcal{F}} i\frac{d_0^{i-1}(1-d_0)}{1-d_0^{\mathcal{F}}} \tag{8}$$

Multiplying this value by T gives the probabilistic part of the latency time, where T is time period of each attempt. To find the second part, recall that once a node finds an agent, it sends a unicast Add_Sel message to the agent and waits for a unicast Add_Conf message from it. Let $A_2 = 0,1,2,\cdots,\Psi$ be a RV denoting the number of successful attempts to get Add_Conf message.

Then, A_2 has the distribution

$$\Pr\left[A_2 = i\right] = \frac{\alpha^{i-1}(1-\alpha)}{1-\alpha^{\Psi}}$$

Then, the expected value of A_2 is

$$E\left[A_2\right] = \sum_{i=1}^{\Psi} i\frac{\alpha^{i-1}(1-\alpha)}{1-\alpha^{\Psi}} \tag{9}$$

As a result, the expected latency time W_2 until the

new node obtains its IP address is given as

$$W_2 = E[A_1]T + 2E[A_2]t_u + \tau \tag{10}$$

The expected latency time W required for obtaining an IP address is give as follows

$$W = W_1 p_0 + (1-p_0)W_2 \tag{11}$$

Substituting for p_0, W_1 and W_2 from (5), (7) and (10) respectively into (11), we get

$$W = \tau(1-\beta)^N + \left(1-(1-\beta)^N\right) \cdot \left(T\sum_{i=1}^{\mathcal{F}} i \frac{d_0^{i-1}(1-d_0)}{1-d_0^{\mathcal{F}}} + 2t_u \sum_{i=1}^{\Psi} i \frac{\alpha^{i-1}(1-\alpha)}{1-\alpha^{\Psi}} + \tau \right) \tag{12}$$

4.2. Equations Communications Overhead

In this section we turn attention to obtain an expression for the communications overhead. As mentioned before, upon the arrival of a new node at the MANET, it senses the existence of beacon messages for a period of time τ. If there is no beacon message for $\mathcal{F}$ trails, the new node will configure itself as the first node in the MANET and hence the number of messages sent in such a scenario is 0.

In the scenario that the new node finds a MANET the number of broadcast messages from the new node to its neighbors is upper bounded by the number of trials $\mathcal{F}$. Let *s* be the probability that a new node selects a particular node, called the *tagged node*, from the ones where the new node received replies (their number is the RV *D*). In the following, we will find *s*, assuming that the new node has received *j* other replies, besides the reply of the tagged node. To this end, if the tagged node has available IPs, $i, i = 1,2,3,\cdots,m$, then the following cases may occur:

1) If *i* is greater than the number of available IPs in any of the *j* replies, the tagged node will be selected, *i.e.* $s = 1$.

2) If *i* is less than the number of available IPs in any of the *j* replies, the tagged node will not be selected, *i.e.*, $s = 0$.

3) If *i* is equal to the number of available IPs in any of the *j* replies, the tagged node will be selected with probability, $s = \frac{1}{j-1}$.

In general, when there are *i* available IPs and *j* of *D* nodes have the maximum value i, then we can select *j* from the *D* nodes in $i\binom{n}{k}$ ways and the other nodes have $(i-1)^{D-j-1}$ combinations. Therefore the value of s can be written as

$$s = \sum_{k=1}^{N} \frac{d_k}{m^k} \sum_{i=1}^{m} \sum_{j=0}^{k-1} \binom{k}{j} \frac{(i-1)^{k-j-1}}{j+1} = \sum_{i=1}^{m} \sum_{i=1}^{N} \binom{N}{j} \frac{(i-1)^{N-j}}{jm^j} d_j \tag{13}$$

We note in passing that the factor $\frac{1}{m^k}$ in the above equation reflects the probability that the replies received by the new node carry an equal amount of available IPs. Substituting for d_j form (6) into (13), we get

$$= \sum_{i=1}^{m} \sum_{j=1}^{N} \sum_{k=j}^{N} \binom{N}{j} \binom{N}{k} \binom{k}{j} \frac{(i-1)^{N-j}}{jm^j} \cdot \beta^k (1-\beta)^{N-k} (1-\alpha)^j \alpha^{k-j} \tag{14}$$

Let $X(t)$ be a RV denoting the number of busy IPs at a given node within the MANET. And let $x_i(t), i = 0,1,2,3,\cdots,m$ be its distribution. It is clear from the assumptions that the $x_i(t)$ form a Markov chain [13-15]. For system stability, we insist that $\lambda s < m\mu$, in which case the system will reach steady state as $t \to \infty$, and $X(t)$ and $x_i(t)$ will converge to *X* and x_i respectively. The steady-state balance equations can be written as follows.

$$\begin{aligned} s\lambda x_0 &= \mu x_1 \\ (s\lambda + \mu)x_1 &= s\lambda x_0 + 2\mu x_2 \\ (s\lambda + 2\mu)x_2 &= s\lambda x_1 + 3\mu x_3 \\ &\vdots \\ (s\lambda + (m-1)\mu)x_{m-1} &= s\lambda x_{m-2} + m\mu x_m \end{aligned}$$

After straightforward algebra and applying a well-known solution of Markov birth-death process [15], the steady-state probabilities $x_i, i = 0,1,2,3,\cdots,m$ are given as follows.

$$x_i = \frac{1}{i!}\rho^i x_0, \quad i = 1,2,3,\cdots,m \tag{15}$$

where, $\rho = \frac{s\lambda}{m\mu}$. Using the normalization condition $\sum_{i=0}^{m} x_i = 1$, the value of x_0 is given as

$$x_0 = \left(1 + \sum_{i=1}^{m} \frac{1}{i!}\rho^i\right)^{-1} \tag{16}$$

The communications overhead per node until the tagged node obtains its IP within the maximum number of $\mathcal{F}$ trials can be found in view of the following two phases:

Phase 1: The tagged node solicits an agent. The broadcast message sent by the tagged node results in *k* replies from the *k* neighbors, *j* of them are received by the tagged node while $k - j$ are lost on their way, $j \le k$.

Therefore, this part of the communications overhead C_1 is given by

$$C_1 = \sum_{i=1}^{\mathcal{F}} s_b^{i-1}\overline{s_b}\left(1+\sum_{k=1}^{N}\sum_{j=1}^{k} k\binom{k}{j}\overline{s_b}^{j} s_u^{k-j} p_k\right) \quad (17)$$

Phase 2: The tagged node obtains an IP from the agent. There are one of two cases. Case 1: the agent has more than one IP to donate, in which case one will be given to the tagged node. Case 2: the agent has only one IP to donate, which occurs with probability x_m in which case it will be given to the *tagged node* and will have to borrow a replacement IP from the tagged node. In either case, each successful communication between the tagged node and the agent takes on average $E[A_2]$ attempts, with each attempt generating two messages. Therefore, this part of the communications overhead C is given by

$$C_2 = 2E[A_2](1+x_m) \quad (18)$$

Accordingly, and using (17), and (18), the total communications overhead per node is given by

$$C = \frac{C_1 + C_2}{N}$$

5. Simulation and Numerical Results

In this section, the efficiency of the OSA protocol is analyzed. We compared our mathematical results and simulation results with the token-based protocol. The reason for choosing the token-based protocol is that it generates small communications overhead and less latency to obtain a new address as shown in [4]. Moreover, token-based protocol and OSA protocol belong to statefulau to configuration protocols. So, it will be the most effective way to measure the performance of OSA protocol.

We developed a discrete-event JAVA simulation program to implement the OSA protocol. We used a radio transmission range *r* for each node with about 55 meters. The transmission area is assumed to be 243 meter by 243 meter square region. The simulations are conducted with the same assumption as in the mathematical model.

At each simulation run, we pre-configure the network with a random number of nodes and also setup a Parameter Record and an Address Table for each node. Each node joins the network at a random location uniformly distributed in the transmission area. Whenever a new node joins the network, we measure the latency and the communications overhead. The number of simulation runs per new node is 10^6 times. In **Figures 5** and **6**, we present comparisons of analytical and simulation results in terms of the latency and the communications overhead.

The operational parameter settings are set as follows: The average proportion of area where a node's transmission reaches β can be calculated using the same derivation in [4], yielding $\beta = 0.16$.

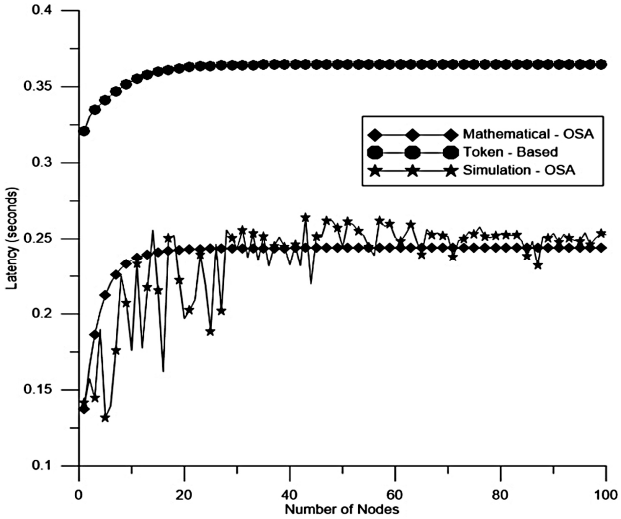

Figure 5. Latency time until obtaining an IP address.

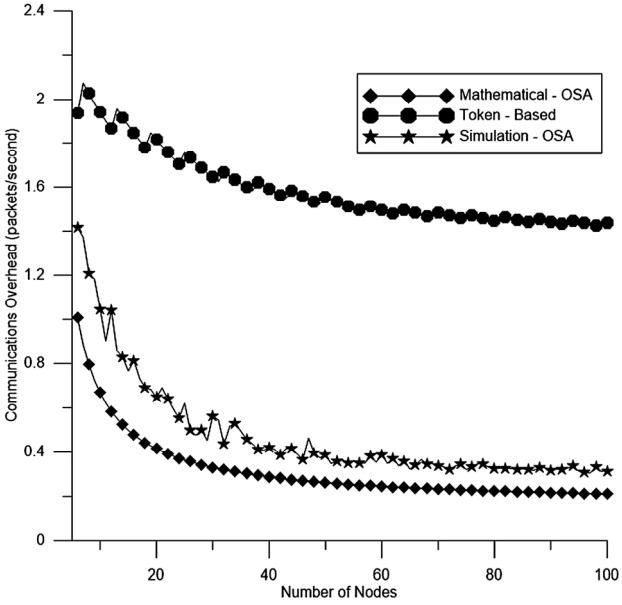

Figure 6. Number of packets sent until obtaining an IP address.

The number of available addresses $m = 3$. The number of trials $\mathcal{F}$ to find the set of neighboring nodes is set to 3 trials. The number Ψ of attempts to get Add_Conf message is set to 1. Both the time-out interval τ and the time for each trial *T* are set to 0.1 seconds. We set $t_b = 0.01$ seconds and $t_u = 0.02$ seconds. The average loss rates of broadcast and unicast packets in link-level s_b and s_u are set to 0.015 and 0.0002 respectively. The arrival rate λ and service rate μ are set to 0.1 and 0.15 respectively.

Figure 5 shows the latency versus the number of nodes *N* for the token-based and OSA protocols. This Figure indicates that the OSA protocol reduces the latency if it is compared with the token-based protocol.

Figure 6 shows the average number of packets sent per node to obtain an IP address versus the number of nodes N for the token-based and OSA protocols. This figure also indicates that OSA protocol reduces the communications overhead traffic if it is compared with token-based protocol.

6. Conclusion

We have presented a MANET address configuration protocol called OSA. The protocol enables nodes to join and leave the MANET with low latency and low communications overhead. The numerical results have been validated through discrete-event JAVA simulations and compared with the token-based protocol

REFERENCES

[1] C., Perkins, "Ad Hoc Networking," Addison-Wesley, Upper Saddle River, 2001.

[2] S. Nesargi and R. Prakash, "MANET Conference: Configuration of Hosts in a Mobile Ad Hoc Network," *Proceedings of the* 21*st Annual Joint Conference of IEEE Conference on Computer Communication* (*INFOCOM*' 02), New York, June 2002, pp. 206-216.

[3] A. Yousef, H. Al-Mahdi, M. Kalil, and A. Mitschele-Thiel, "LHA: Logical Hierarchical Addressing Protocol for Mobile Ad-Hoc Networks," 10*th ACM/IEEE International Symposium on Modeling*, *Analysis and Simulation of Wireless and Mobile Systems* (*MSWiM* 2007), Chania, 22-26 October 2007.

[4] S. Kim, J. Lee and I. Yeom, "Modeling and Performance Analysis of Address Allocation Schemes for Mobile Ad Hoc Networks," *IEEE Transactions on Vehicular Technology*, Vol. 57, No. 11, 2007, pp. 490-501.

[5] H. Zhou, L. Ni and M. Mutka, "Prophet Address Allocation for Large Scale MANETs," Infocom, 2003.

[6] C. Perkins, J. Malinen, R. Wakikawa, E. Royer and Y. Sun, "IP Address Auto-configuration for Ad Hoc Networks," IETF Internet Draft, 2001.

[7] N. H. Vaidya, "Weak duplicate address detection in mobile ad hoc networks," *ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Boston-Massachusetts, 9-11 June 2002, pp. 206-216. http://dx.doi.org/10.1145/513800.513826

[8] K. Weniger, "Passive Duplicate Address Detection in Mobile Ad Hoc Networks," *Proceedings of IEEE WCNC* 2003, New Orleans, March 2003.

[9] C. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for mobile Computers," *Proceedings of the SIGCOM*'94 *Conference on Communications Architecture*, *Protocols and Applications*, London, August 1994, pp. 234-244.

[10] C. Perkinsand and E. Royer, "Ad-hoc On-Demand Distance Vector Routing," *Proceedings of the* 2*nd IEEE Workshop on Mobile Computing Systems and Applications* (*WMCSA*'99), New Orleans, February 1999, pp. 90-100.

[11] D. Johnson and D. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," In: T. Imielinski and H. F. Korth, Eds., *Mobile Computing*, Chapter 5, Kluwer Academic Publishers, New York, 1996, pp. 153,181,

[12] A. M. Law and W. D. Kelton, "Simulation Modeling and Analysis," Third Edition, Mc. Graw Hill, New York, 2002.

[13] H. Nassar and H. Al Mahdi, "Queueing Analysis of an ATM Multimedia Multiplexer with Nonpreemptive Priority," *IEEE Proceedings on Communications*, Vol. 150, No. 3, 2003, pp. 189-196. http://dx.doi.org/10.1049/ip-com:20030218

[14] M. G. Giri, "Introduction to Probability and Statistics," 2nd Edition, Marcel Dekker, New York, 1993.

[15] G. Gross and C. Harris, "Fundamentals of Queueing Theory," John Wiley, New York, 1985.

