ICST Transactions
on Mobile Communications and Applications

Research Article ICST.ORG

Analytical modeling of address allocation protocols in wireless *ad hoc* networks☆

Ahmad Radaideh, John N. Daigle*

University of Mississippi, University, MS 38677, USA

Abstract

Detailed descriptions of Internet Protocol Address Assignment (IPAA) and Mobile Ad Hoc Network Configuration (MANETconf) are presented and state diagrams for their behavior are constructed. Formulae for the expected latency and communication overhead of the IPAA protocol are derived, with the results being given as functions of the number of nodes in the network with message loss rate, contention window size, coverage ratio, and the counter threshold as parameters. Simulation is used to validate the analytical results and also to compare performance of the two protocols. The results show that the latency and communication overhead for MANETconf are significantly higher than the measures of the IPAA protocol. Results of extensive sensitivity analyses for the IPAA protocol are also presented.

Keywords: address assignment protocols, mobile *ad hoc* networks, performance evaluation

Received on 27 December 2010

doi: 10.4108/icst.trans.mca.2011.e3

1. Introduction

In this paper, detailed descriptions for two dynamic and distributed protocols proposed for address allocation in wireless *ad hoc* networks are presented, and analytical derivations for the expectations of performance measures—latency and communication overhead—for one of these protocols are carried out. Latency is the amount of time required for a newly joining node to obtain a network address, and communication overhead represents the number of messages sent during the allocation process. Expected values of the performance measures as a function of the number of network nodes at the time the new node joins the network are given. Analytical and simulation results are presented to show the effect of changes in message loss rate, coverage ratio, and the contention window size on the performance measures.

A Mobile Ad Hoc Network (MANET) has neither permanent infrastructure, nor centralized servers, nor connectivity to external networks. It consists of end systems, or nodes, that communicate with each other over a wireless medium. A node has a limited transmission range due to its limited power and it communicates directly with nodes within its transmission range. The MANET topology changes dynamically; nodes are free to move within the network, join, and leave at any time. Each node in the network runs a routing protocol so that a message from a source node can be transmitted to a destination node even if that node is outside the source's transmission range.

Address configuration can be performed using address mapping, static configuration, or dynamic configuration approaches. Address mapping uses a mapping function to derive a network address from a hardware interface identifier (MAC address) as in [1]. However, MANET nodes are not restricted to using 48-bit MAC addresses. Also the mapping is not guaranteed to be unique in IPv4 networks which use shorter 32-bit network addresses. The static configuration approach needs a user interaction and knowledge of the network's current configuration,

☆A preliminary version of this paper was presented at ICST ADOCNETS 2010, Victoria, BC.

*Corresponding author. Email: wcdaigle@olemiss.edu

which is not practical for a dynamic topology network. In the dynamic approach, a dynamic configuration protocol is used to assign a network address. Configuration protocols such as the Dynamic Host Configuration Protocol (DHCP) [2], which run on centralized machines, cannot be extended to MANETs because of their distributed and dynamic nature. Hence, a distributed dynamic configuration protocol is required for a MANET.

Some characteristics that need to be taken into account when designing address configuration protocols include multi-hop communication, dynamic topology, and network merging and partitioning [3]. Some proposed solutions for the address allocation problem in MANETs use the duplicate address detection (DAD) mechanism to verify the uniqueness of a network address throughout the network. Others use the binary-split idea [4] to distribute the address block among network nodes so that each node has a disjoint subset of network addresses and therefore DAD is avoided. Most of these mechanisms are classified in [5] based on the following factors: network scenario (stand-alone or connected to Internet), routing protocols' dependency, address uniqueness (DAD or non-DAD), distributed or centralized address allocation, and MANET characteristics support.

Perkins [6] configures by first choosing a random address then performing a DAD procedure within the MANET. To perform the uniqueness check, the node sends an address request (AREQ) message including the randomly selected address. This message is broadcasted to all nodes in the network. The source address of the AREQ is another temporary IP address used only for sending this message. It has a different non-overlapping prefix than the prefix of the address selected for allocation and it is selected randomly so that the duplicate probability is very low. An address reply (AREP) message is sent only if the address of the receiving node matches the address in the AREQ message. The new node concludes that the selected address is unique if there is no AREP message received after sending the AREQ a finite number of retries denoted as REQUEST_RETRIES. Since the protocol performs DAD only when assigning an IP address to a new node, the proposed protocol lacks support for partitioning and merging in MANET.

Jeong [7] proposes a protocol with two address detection mechanisms. A strong DAD, based on the protocol proposed in [6], is performed in the initial phase to verify the uniqueness of the randomly selected address and a weak DAD, based on [8], which is always executed in order to prevent address conflicts with existing nodes. The weak DAD uses the concept of a virtual address, which is a combination of an address and a key. The key, which is assumed to be unique in the network, is appended to the address in the routing messages as well as the routing table. The weak DAD identifies duplicate addresses by monitoring routing information and reports the address duplication by sending out an error (AERR) message to one of the conflicting nodes to change its address. This protocol monitors and changes the routing messages and therefore is considered routing protocol dependent.

A passive DAD approach [9] has been adopted in some proposed solutions for dynamic address configuration, such as in [10] and [11]. Passive DAD enables nodes to detect duplicate addresses in the network by analyzing received routing protocol messages. One way to detect address conflicts is based on the sequence number of a link-state routing message. The sequence number is always incremented and used to distinguish fresh from old routing information. Given that two messages with the same sequence number and source address are copies of the same message, a node may detect an address conflict if it receives a message with its own address as source address and a sequence number higher than its own counter. Other ways to detect address conflicts are based on locality and neighborhood and they are all based on analyzing routing information, which makes these solutions routing protocol dependent.

Zhou *et al.* [12] proposed a mechanism based on a stateful function, $f(n)$, to derive addresses with low probability of address duplication and therefore it avoids the use of DAD. The initial state of $f(n)$ is a seed that generates a sequence of unique numbers that can be used as network addresses. The function has to be designed carefully such that the interval between two occurrences of the same number in the generated sequence is extremely long and the probability of generating the same number in finite number of sequences initiated by different seeds is extremely low. A proposed solution based on the stateful function $f(n)$ works as follows: The first node in the network, say A, chooses a random number as its address and uses a random or default state value as a seed of its $f(n)$. When a new node, say B, joins the network and asks node A for an address, A generates an address using its state function and new state value and assigns them to B. Node A updates its state value accordingly. Node B uses the generated address as its address and the state value as a seed for its function so it can assign address to other nodes. The main concern about this protocol is designing an $f(n)$ that satisfies the properties mentioned above, which is considered a hard mathematical problem. Additional approaches based on genetic algorithms and a quadratic residue approach are presented in [13] and [14], respectively.

In MANETconf [15], an existing node is in charge of unique address allocation for a new joining node. Each configured node maintains state information of the currently assigned addresses so it can choose, based on its knowledge, an available address and verify its uniqueness throughout the network. When a new node joins the network, it asks one of its neighbors to perform the address allocation process on its behalf. The selected neighbor chooses an available address and performs the DAD procedure across the network to verify the uniqueness of the selected address. In case the new joining node is unable

to find a neighbor, it concludes it is the first node in the network and performs address configuration for itself. The proposed protocol handles network partitioning and merging by performing address recovery and duplicate detection procedures.

Mohsin and Parakash [16] propose the IPAA protocol which is based on a dynamic configuration of addresses using the concept of binary split. The protocol is classified as a proactive approach because each node can independently assign a unique address to a new node without consulting any other node in the MANET. Each node in the network has a disjoint subset of the address space. When a new node joins the network, it tries to find a neighbor node that can perform an address configuration on its behalf. If a neighbor is found, the new node asks that neighbor for an address allocation. The neighbor splits its available address space into two halves and sends one half to the new node. The new node then assigns itself the first IP address in the received address space and keeps the rest with itself to configure other nodes in the future. If the new node could not find a neighbor, it concludes it is the first node in the network. So, it assigns itself the first address in the address space and keeps the rest of the address space for itself. The protocol handles network partitioning and merging as well as address recovery due to node departures. A very similar approach to IPAA, which is based on the binary-split idea, is proposed in Tayal and Patnaik [17].

The main contributions in this work are building state diagrams that represent the behavior of the two address allocation protocols proposed in [15] and [16], deriving analytical formulations for the expectations of latency and communication overhead for the protocol in [16], and presenting analytical and simulation results for the performance measures of that protocol for different numbers of existing nodes in the network. The protocol in [15] represents the DAD-based allocation schemes with an advantage of selecting a network address based on state information a node maintains to enhance the protocol performance[1]. The second protocol, which is proposed in [16], performs address allocation through local communications with neighbor nodes and that is achieved through distributing the address space among network nodes based on the binary-split idea.

The paper is organized as follows. In Section 2, detailed descriptions of the proposed address allocation protocols in [15] and [16] are presented. For each protocol, two state diagrams are derived based on the protocol specifications. The state diagrams give a complete picture of the protocol behavior, messages, timers, and handshakes. One diagram shows the state of the new node during the address allocation process while the other one shows the state of an existing node that performs the address allocation for the new node. In Section 3, analytical formulas for latency and communication overhead for the protocol in [16] are derived. The analytical formulas represent the expected values of the performance measures as a function of the number of nodes in the network and with message loss rate, contention window size, coverage ratio, and counter threshold as parameters. Section 4 presents the analytical as well as the simulation results for different values of the message loss rate, contention window size, and the coverage ratio. Section 5 concludes the paper and suggests future research.

[1]The protocol proposed in [18] is compared to MANETconf via simulation with latency reported at about one half of that of MANETconf.

2. Detailed descriptions of two address allocation protocols

This section presents detailed descriptions of MANET conf [15] and IPAA [16] in separate subsections. Both protocols are designed with the following network operating characteristics in mind. Nodes are free to move in the network, join, and leave at any time. Address allocation and maintenance are the responsibility of existing nodes in the network and have to be performed when the topology changes to maintain uniqueness of allocated addresses. The MANET is configured as a private IPv4 network in which the participating nodes are configured in advance to use a specific private address block. At any given time a group of connected nodes forms a network partition that has a universal unique identifier (UUID). As time evolves, a partition could either split or merge with another partition; the UUID is the key to managing the partitions. The nodes in MANET communicate with each other using IP datagrams. Communication between distant nodes in a partition is carried over intermediate nodes running an *ad hoc* routing protocol.

2.1. MANETconf protocol

In MANETconf protocol, previously configured nodes in the network manage address allocation for newly joining nodes. A newly joining node, a *requester*, chooses one of its configured neighbors, an *initiator*, to perform the address allocation on its behalf. The initiator first selects a candidate address and then broadcasts a message to all nodes in its partition to verify the uniqueness of the selected address. If verification is successful, the initiator allocates the address to the requester and informs all other nodes of the address assignment. Otherwise, the initiator repeats this address selection-verification mechanism for a finite number of times before giving up.

The protocol performs maintenance operations that clean up addresses of departed nodes due to node crashes or network partitioning, resolves address conflicts after network merging, maintains correct allocation information in each node using a soft state mechanism (i.e. timers), and resolves concurrent allocation conflicts using a priority

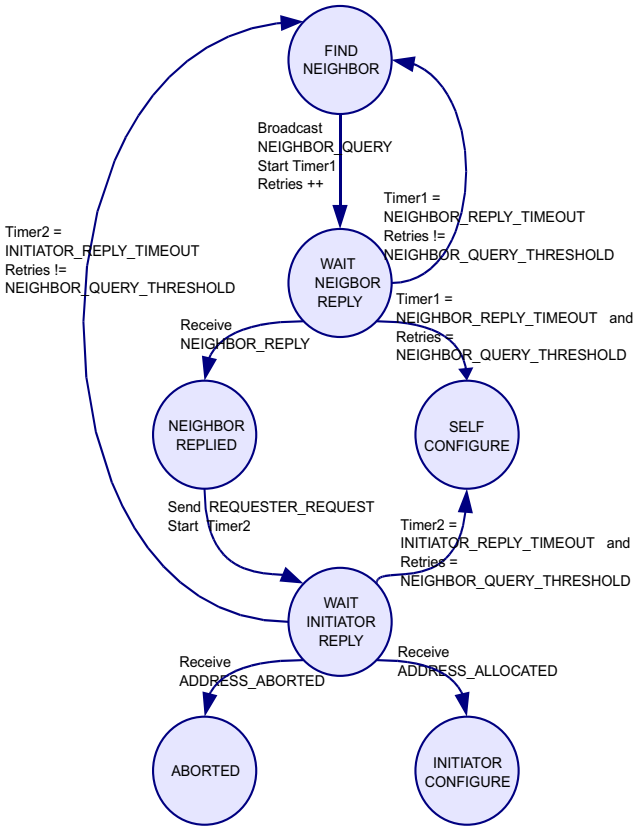

Figure 1. State diagram of a requester node during the address allocation process of MANETconf protocol.

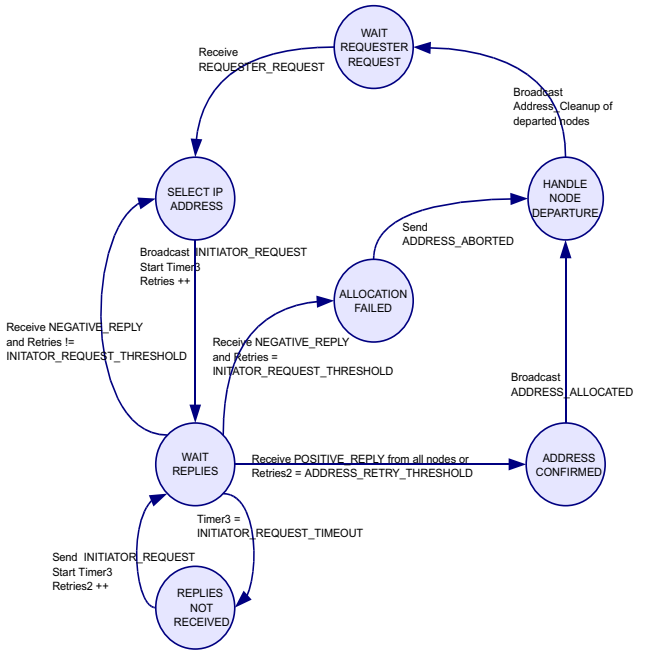

Figure 2. State diagram of an initiator node during the address allocation process of MANETconf protocol.

mechanism based on the initiator address. New node address allocation and network partitioning and merging are described in separate subsections below.

New node address allocation. Figures 1 and 2 show the state diagrams of the address allocation process for a requester and an initiator nodes, respectively. The requester broadcasts a NEIGHBOR_QUERY message. If at least one NEIGHBOR_REPLY message to this query is received within a timeout value of NEIGHBOR_REPLY_TIMEOUT, the requester selects one of the responders as an initiator; otherwise, the requester retries until NEIGHBOR_QUERY_THRESHOLD is reached and then allocates an address and forms its own partition.

When the REQUESTER_REQUEST message is received, the initiator starts the address allocation process. As stated above, address allocation is a two-phase process that starts by selecting an address and verifying its uniqueness over the network partition and then confirms the allocation of the unique address to the requester. In order to minimize the conflict probability, each configured node maintains state information of address allocation in two data sets. An *Allocated* set which lists all allocated addresses, and an *Allocate_Pending* set which lists the addresses that are being allocated to newly joining nodes. Each entry in Allocate_Pending shows the address being allocated and the initiator node performing the allocation process to that address. A soft state maintenance to Allocate_Pending set is considered where each entry is automatically deleted after it times out.

The initiator selects an address that is neither in its Allocated nor Allocate_Pending sets. It inserts the allocation information {selected address, initiator address} to its Allocate_Pending set and broadcasts an INITIATOR_REQUEST message to all nodes in its partition to verify the address uniqueness. Address uniqueness is successfully verified when all nodes that are listed in the initiator Allocated set send a POSITIVE_REPLY message back to the initiator. This message verifies that the address is unique based on the state information of the replied node. If positive replies from all nodes have been received, the initiator sends the selected address in an ADDRESS_ALLOCATED message to all nodes in the partition including the requester. Each node inserts the allocated address in its Allocated set and deletes the address from its Allocate_Pending set.

Due to message losses, node departure, or mobility some replies may not be received. If all received replies are positive, the initiator verifies the selected address again by sending INITIATOR_REQUEST to nodes that did not reply to the previous request for ADDRESS_RETRY_THRESHOLD number of times. The address is considered unique if no negative reply has been received. The nodes that have not replied to the initiator request are considered departed nodes and their addresses are cleaned up.

The uniqueness verification of the selected address fails when a node finds the address in its Allocated set. It also fails if the address found in a node Allocate_Pending set is

being allocated by a higher priority initiator. A node reports the failure by sending a NEGATIVE_REPLY message back to the initiator. If at least one negative reply has been received before INITIATOR_REQUEST_TIMEOUT, the initiator selects another address and repeats the allocation process again for INITIATOR_REQUEST_THRESHOLD times. If all trials have failed, the initiator sends an ADDRESS_ABORTED message to the requester indicating that all address allocation retries have failed. Receiving an ADDRESS_ABORTED message or no messages at all within the timeout period of INITIATOR_REPLY_TIMEOUT, the requester searches for another initiator for a maximum number of trails equal to NEIGHBOR_QUERY_THRESHOLD to perform the allocation process again.

Network partitioning and merging. The topology of a MANET changes dynamically due to node movement. The network may split into two or more partitions and partitions may also merge together to form a bigger partition. Address maintenance has to take place after these operations to solve address leak and duplicate problems.

In a MANET, each partition has a UUID, which is the lowest IP address of the partition. A newly joining node is provided with its partition UUID as well as its own IP address during its address allocation process, whereas a node that configured itself sets its partition's UUID to its own IP address.

When a group of nodes splits into two separate partitions due to nodes movement, nodes in both partitions will detect departure of other nodes during the address allocation process of a newly joining node. When a partition detects the partitioning event, address cleanup procedure is performed. The node that detects the partitioning event manages address cleanup of departed nodes, so it broadcasts an ADDRESS_CLEANUP message to all other nodes in its partition. A node that receives the address cleanup message deletes the addresses listed in the message from its Allocated set. Since the UUID is determined by the lowest address in the partition, one of the partitions has to be assigned a new UUID. In a partition that the lowest address has been deleted, each node selects the lowest remaining address allocated in that partition to be the new UUID.

Figure 3 shows the state diagram of the partitions merging process. When two nodes i and j from two different partitions get close to each other, they exchange their partition identifiers. If the received partitions identifier is different from a node partitions identifier then a node detects the merging of two partitions. Both i and j will detect merging of their partitions in that case. After detecting the merging event, both i and j exchange their Allocated sets. Each one of them broadcasts the Allocated set of the other to all nodes in its partition. Each node in both partitions takes the union of its Allocated set and the received Allocated set.

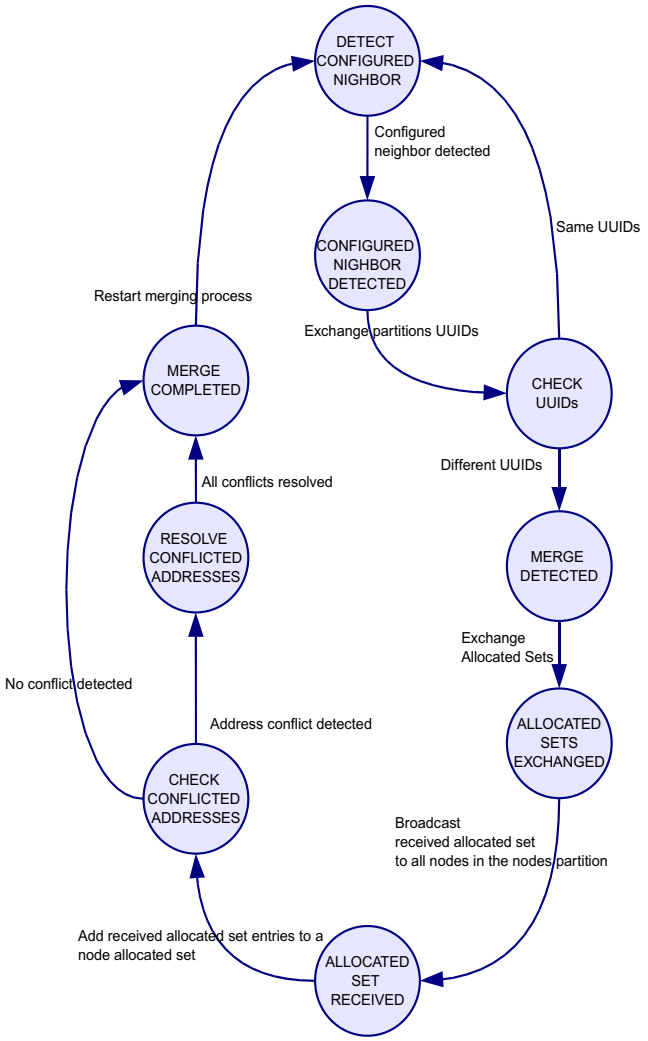

Figure 3. State diagram of the merging process of MANETconf protocol.

If the received Allocated set contains a node address, then there is a node in the other partition that has the same address. Those two nodes are called *conflicting* nodes. One of the conflicting nodes has to give up the conflicting address and ask existing neighbors for a unique address. The address allocation process has to be performed for each conflicting address and this time a global agreement on the selected address has to be granted from all nodes in both merged partitions. Nodes in one partition have to update their UUID to the lowest allocated address in both partitions. The merging of two partitions is completed when address conflict has been resolved.

2.2. IPAA protocol

The IPAA protocol presents a distributed dynamic address allocation protocol for a stand-alone MANET. The protocol avoids the DAD process by employing a proactive approach using the binary-split idea. The binary-split idea

is that each node has a disjoint subset of the address block and it can independently allocate a unique address and hand half of its address space to a newly joining node without getting an agreement from every other node in its partition.

When a new partition is formed, the only node in that partition reserves the complete address block and assigns itself the first address in that block. A newly joining node, a requester (or client), asks an existing neighbor node, an initiator (or server), for a network address. The server divides its address space into two halves and gives one half to the requester. The requester assigns itself the first address from the received address space and keeps the rest of it to serve other nodes in the future. If the initiator has no space left, it borrows an address space from an existing node and forwards it to the requester. This procedure avoids flooding the entire partition to verify the uniqueness of the selected address as DAD-based protocols do.

Maintaining the complete Address_Block is the key issue in this protocol. The protocol performs maintenance procedures to avoid address leak and conflict problems. The address leak problem happens when a node abruptly departs the network or moves out its partition without returning its address space. Without cleaning up departed nodes addresses, these addresses will be considered allocated to existing nodes and cannot be allocated to newly joining nodes. Nodes keep track of allocated address blocks to resolve the address leak problem. Graceful departure is provided where nodes that want to leave the network return their address blocks and confirm their departure. Address conflict problems could happen when two partitions merge together since each partition reserves the entire address block for itself. Nodes with the same address in both partitions are allocated the same address space when configured. The conflicting node that has a bigger address space should give up its allocated space and ask other nodes for new allocation.

New node address allocation. Figures 4 and 5 show the state diagrams of a client and a server nodes during address allocation process, respectively. When a client joins the network it broadcasts a one-hop REQUEST message. A neighbor replies with a REPLY message to the client. The client selects one of its existing neighbors that replied to its request message to be its server. The client sends an acknowledgement (ACK) message to the selected server asking for a unique address. When receiving the ACK message, the server starts the allocation process for the requested client. Neighbors are expected to reply to the client request within REPLY_TIMEOUT amount of time. If the timer expires without receiving any reply, the client repeats searching for neighbors for NEIGHBOR_ REQUEST_THRESHOLD number of times. If all trials have failed, the client reserves the entire address block for itself, allocates itself the first address in that block, and sets a UUID for that partition.

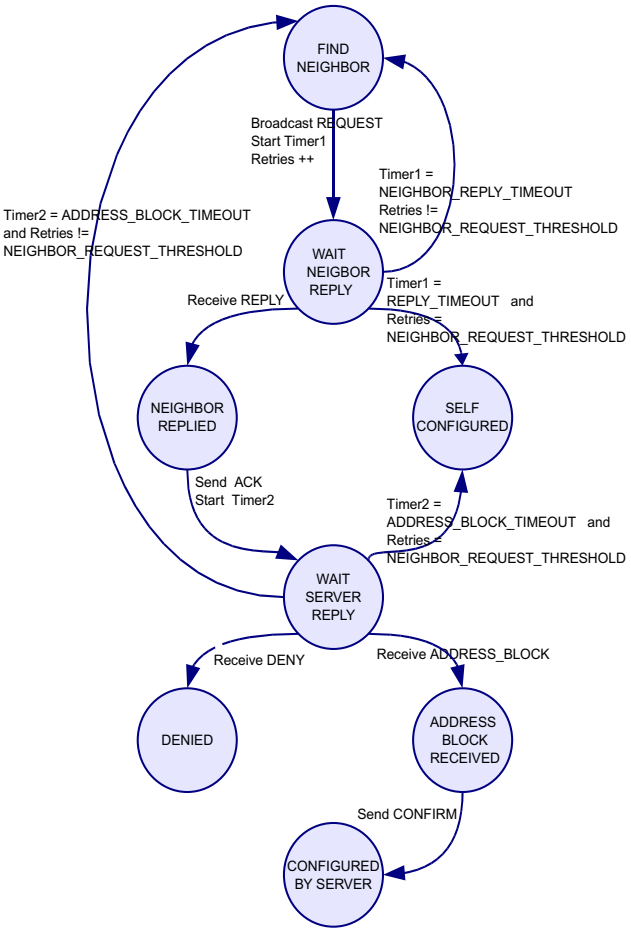

Figure 4. State diagram of a client node during the address allocation process of IPAA protocol.

If reply messages have been received, the client selects one of its neighbors to perform the address allocation process by sending an ACK message and starting a timer with a timeout value ADDRESS_BLOCK_TIMEOUT. When a node receives the ACK message, it divides its available address space into two disjoint subsets and sends one subset to the client in the ADDRESS_BLOCK message. The client receives the ADDRESS_BLOCK and the partition UUID, configures itself the first address in that block, and keeps the rest of its address space to configure newly joining nodes in the future. The client confirms a successful address allocation by sending CONFIRM message back to the server. If the client timer expires before receiving the ADDRESS_BLOCK message, the client considers that the server is no longer existing and searches for another server to perform the address allocation process again up to NEIGHBOR_REQUEST_THRESHOLD number of times. If the ADDRESS_BLOCK message has not been received in all trials, the client performs self-allocation to configure itself an address as described above.

In case the selected server has no available addresses to serve the client, the server searches for an existing node in

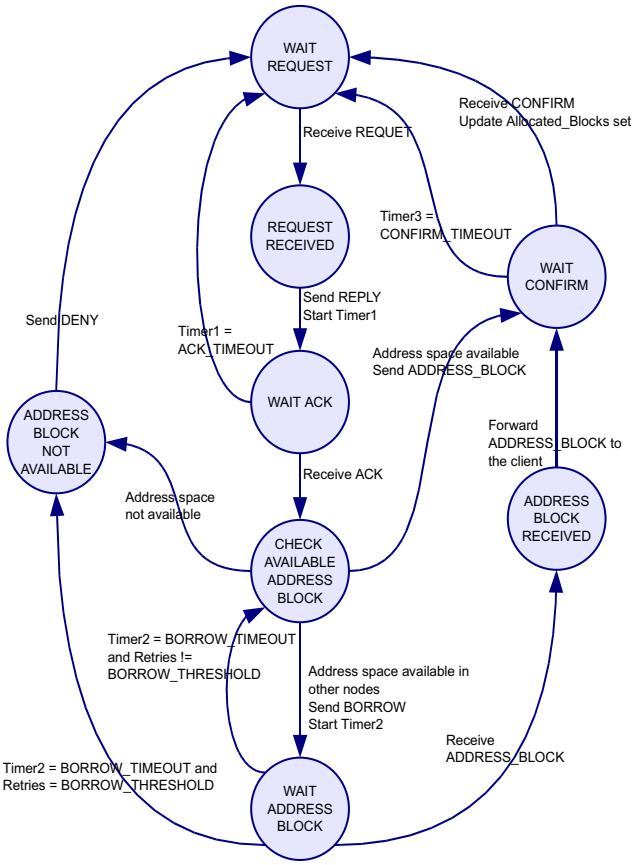

Figure 5. State diagram of a server node during the address allocation process of IPAA protocol.

the partition that has an available address. For this purpose, each node maintains state information about the allocated address blocks in Allocated_Blocks data set. The Allocated_Blocks set lists the configured nodes in the partition with their available Address_Blocks. The server selects the node with the largest available Address_ Block and sends it a BORROW message requesting half of its available space. Once the ADDRESS_BLOCK message is received from the requested node, the server forwards the message to the client. If all nodes in the server Allocated_Blocks set have no available address space, the server sends a deny (DENY) message to the client indicating that addresses are not available in this partition.

Nodes could depart the network or move out their partitions at any time. If a node does not respond to the BORROW message within a timeout period of BORROW_TIMEOUT, the server sends the borrow message to the node that has the second largest address space. The borrow process is repeated up to BORROW_THRESHOLD number of times. If all trials have failed or no address space is available in the rest of the partition nodes, the server sends DENY message to the client indicating that addresses are not available in this partition.

Network partitioning and merging. Nodes may depart the network or move out of their partitions at any time. Departure of a node leads to address leak problem where the nodes address block will not be used by other existing nodes. Each node is responsible for cleaning up the Address_Block of its missing buddy node. To achieve this, the Allocated_Blocks set in each node has to be updated regularly. Each node in the partition broadcasts its Allocated_Blocks set to every other node in the partition. A node updates its Allocated_Blocks set when receiving other nodes' sets to keep its information up to date.

Since the state information a node has is assumed up to date, each node looks up its Allocated_Blocks set from time to time to check the existence of its buddy node. If the buddy node is missing from the set, a node claims that its buddy has departed the network. Therefore, it merges its buddy node Address_Block with its block. Each partition has a UUID to be identified from other existing partitions. Partitioning is detected if the node with the lowest address is missing. When detecting the partitioning event, each node sets the partition UUID to the lowest address currently allocated to a node in their partition.

It is possible that a partition merges with another partition in the network. The merging process performed in this protocol is similar to the process described in Figure 3. When two configured nodes get close to each other, they exchange their partition UUIDs. If the nodes UUIDs are different, a merging event is detected. Those nodes that detect the merging event exchange their Allocated_Blocks sets. Each node broadcasts the other nodes' set to all nodes in its partition. When a node receives the Allocated_Blocks set, it searches the set to check if a node with the same address exists in the other partition. If an address conflict is detected, one of the two nodes with a larger address space gives up its address space and asks existing nodes for a new allocation. Merging of two partitions ends when all address conflicts are resolved. The partition UUID maintained by each node is updated to the lowest address allocated in the resulting partition.

3. Analytical modeling of address allocation protocols

In this section, we derive analytical expressions of the expected values of latency and communication overhead that are used to evaluate the performance of IPAA [16]. The main objective of such analytical derivations is to obtain mathematical formulations that can clarify the impact of network characteristics and the number of existing nodes in the network on the performance measures of the address allocation process under consideration. The network characteristics that have an impact on the performance measures are the network area, the node's coverage

area, collisions, and message loss rate. The derived formulas also show the impact of the protocol parameters which are the timeout values and the counter values on the performance measures of the selected protocol.

The derivations of the expected latency and communication overhead are first carried out for small number of existing nodes in the network and then generalized for an arbitrary number of nodes. Section 3.1 presents the network model under consideration. It defines the network boundary, the node's coverage area, the message loss as well as collision for the incoming traffic at the new node. In Section 3.2, derivations for the expected latency and communication overhead are conducted.

3.1. The network model

Let A represent the total area of the network. Each node in the network is located at a random position and nodes are assumed to be uniformly distributed over the network area. Nodes are assumed to have a coverage area of A_x. Let $\tilde{n}$ denote the number of existing nodes in the network at the time a new node wishes to join. Let $\tilde{x}_i$ indicate the presence of node i, $i = 1, 2, \ldots, n$, within the transmission range of the new node; that is, $\tilde{x}_i = 1$ is a neighbor of the newly entering node and $\tilde{x}_i = 0$ otherwise.

Since the existing nodes are uniformly distributed over the network area independent of the placement of all other nodes, $\tilde{x}_i$, $i = 1, 2, \ldots, n$, is a set of identical, independent, Bernoulli trials with success probability $\frac{A_x}{A}$. Thus, the number of existing nodes that are within the transmission range of the new node, denoted as $\tilde{x}$, is a binomial random variable with parameters $\left(n, \frac{A_x}{A}\right)$ and

$$E[\tilde{x}|\tilde{n} = n] = n\,P\{\tilde{x}_i = 1|\tilde{n} = n\} = n\,\frac{A_x}{A}. \qquad (1)$$

The new node communicates with a neighbor node over a wireless channel. A bad channel condition results in a low signal to interference plus noise ratio (SINR) for the received message and the message is considered lost if its SINR is below a given threshold. Assume that the wireless channel between the new node and a neighbor node has a message loss rate equal to ε. To avoid collision, each node in the network has a contention window of size W. A node that has data to send chooses a random slot number uniformly from $\{1, 2, \ldots, W\}$ and sends its data in that selected slot. Collision in a given slot could happen if two or more nodes transmit in the same slot despite the SINR value of their messages. A message is received successfully if it does not collide with other messages in the transmission slot and has a good SINR value.

3.2. Expectations of the performance measures

The amount of time for the new node to be configured with a network address is denoted as $\tilde{\ell}$ and the number of messages sent during the address allocation process is denoted as $\tilde{c}$. The derivations for the expected values of latency and communication overhead for a given number of nodes, $\tilde{\ell}|\tilde{n} = n$ and $\tilde{c}|\tilde{n} = n$, respectively, are first constructed for the case $\tilde{n} = 0$ and then generalized to an arbitrary number of nodes $\tilde{n} = n$.

The $\tilde{n} = 0$ case. In this case the new node is the only node in the network. The new node starts by broadcasting a REQUEST message in order to be allocated by an address through local communication with its neighbors. Since there are no nodes in the neighborhood, the timer for receiving the REPLY message will reach the timeout value T_{NR} without receiving any REPLY message. Since no REPLY messages have been received during the first timeout period, the new node sends another REQUEST message and again waits for timeout. The new node will repeat sending the REQUEST message and waiting for timeout for a maximum of K_{NR} trials. After the last timeout period, the new node considers itself the first node in the network and performs address configuration itself. It allocates the entire address space for itself and assigns itself the first address in that space.

The expected latency for a new node to be allocated a network address in this case, denoted as $E[\tilde{\ell}|\tilde{n} = 0]$, is the sum of the timeout value for all trials. The timeout value for each trial is T_{NR} and the maximum number of trials is K_{NR}. Therefore,

$$E[\tilde{\ell}\,|\,\tilde{n} = 0] = E[\tilde{\ell}\,|\,\tilde{x} = 0] = K_{\mathrm{NR}}\,T_{\mathrm{NR}}. \qquad (2)$$

The expected communication overhead in this case, which is denoted as $E[\tilde{c}|\tilde{n} = 0]$, is the number of REQUEST messages sent by the new node. The new node sends one REQUEST message in each trial for a maximum number of K_{NR} without getting any replies. Therefore,

$$E[\tilde{c}\,|\,\tilde{n} = 0] = E[\tilde{c}\,|\,\tilde{x} = 0] = K_{\mathrm{NR}}. \qquad (3)$$

General formulas for $\tilde{n} = n$. In this section we derive general formulas for the expectations of latency and communication overhead for the address allocation protocol presented in [12]. The latency, as defined earlier, is the amount of time required for a new node to be allocated a network address, and the communication overhead is the number of messages sent during the address allocation process. Here, we derive expectations for the latency and communication overhead given that the number of nodes in the network is $\tilde{n} = n$.

Since the selected protocol performs address allocation through local communications with the new node neighbors, the expectations for its latency and communication overhead are derived from their conditional values on the number of nodes that are within the transmission range of the new node. The general formulas for the expected latency and communication overhead are given by

$$E[\tilde{\ell} \mid \tilde{n} = n] = \sum_{x=0}^{n} E[\tilde{\ell} \mid \tilde{x} = x]\, P\{\tilde{x} = x \mid \tilde{n} = n\} \quad (4)$$

and

$$E[\tilde{c} \mid \tilde{n} = n] = \sum_{x=0}^{n} E[\tilde{c} \mid \tilde{x} = x]\, P\{\tilde{x} = x \mid \tilde{n} = n\}. \quad (5)$$

Recall that $\tilde{x}$ is a binomial random variable with parameters $\left(n, \frac{A_x}{A}\right)$. Thus,

$$P\{\tilde{x} = x | \tilde{n} = n\} = \binom{n}{x} \left(\frac{A_x}{A}\right)^{x} \left(1 - \frac{A_x}{A}\right)^{n-x}. \quad (6)$$

For the special case when there is no node within the transmission range of the new node, that is $(\tilde{x} = 0)$, the expected latency and communication overhead as derived in (2) and (3) are given by

$$E[\tilde{\ell} \mid \tilde{x} = 0] = K_{\mathrm{NR}}\, T_{\mathrm{NR}} \text{ and } E[\tilde{c} \mid \tilde{x} = 0] = K_{\mathrm{NR}}. \quad (7)$$

For all other cases, where $\tilde{x} > 0$, the expectations for the latency and communication overhead for a given number of neighbors are calculated from their conditional values on the number of trials, denoted by $\tilde{\kappa}$, required to successfully allocate a network address to the new node. Conditioning on the value of $\tilde{\kappa}$ yields

$$\begin{aligned} E[\tilde{\ell} \mid \tilde{x} = x] = &\sum_{\kappa=1}^{K_{\mathrm{NR}}} E[\tilde{\ell} \mid \tilde{x} = x, \tilde{\kappa} = \kappa] \\ &\times P\{\tilde{\kappa} = \kappa \mid \tilde{x} = x\} + E[\tilde{\ell} \mid \tilde{x} = x, \tilde{\kappa} > K_{\mathrm{NR}}] \\ &\times P\{\tilde{\kappa} > K_{\mathrm{NR}} \mid \tilde{x} = x\}, \end{aligned} \quad (8)$$

and

$$\begin{aligned} E[\tilde{c} \mid \tilde{x} = x] = &\sum_{\kappa=1}^{K_{\mathrm{NR}}} E[\tilde{c} \mid \tilde{x} = x, \tilde{\kappa} = \kappa] \\ &\times P\{\tilde{\kappa} = \kappa \mid \tilde{x} = x\} + E[\tilde{c} \mid \tilde{x} = x, \tilde{\kappa} > K_{\mathrm{NR}}] \\ &\times P\{\tilde{\kappa} > K_{\mathrm{NR}} \mid \tilde{x} = x\}. \end{aligned} \quad (9)$$

The kth trial is successful if an ADDRESS_BLOCK message is received from a neighbor node in response to the ACK message. The ADDRESS_BLOCK message contains a disjoint subset of the address space where the new node selects the first address from that space for itself and keeps the rest for serving other joining nodes in future. Assuming that successive trials are independent and identical, the probability that a successful address allocation occurs on the kth trial is given by

$$P\{\tilde{\kappa} = \kappa \mid \tilde{x} = x\} = \left(P\{\tilde{a} = 0 \mid \tilde{x} = x\}\right)^{\kappa - 1} P\{\tilde{a} = 1 \mid \tilde{x} = x\}, \quad (10)$$

where $\tilde{a}$ is the indicator random variable for the event of a successful allocation. As discussed earlier in previous sections, the trial fails if the ADDRESS_BLOCK message is not received when at least one REPLY message is received or when no REPLY message is received. Define $\tilde{s}$ as the indicator random variable for the event of the reception of at least one reply message from a neighbor node and that message is transmitted with no other messages in a slot and has a good SINR value. Then, the probability of a failed trial in the presence of x neighbors is given by

$$\begin{aligned} P\{\tilde{a} = 0 | \tilde{x} = x\} &= P\{\tilde{a} = 0 | \tilde{x} = x, \tilde{s} = 0\}\, P\{\tilde{s} = 0 | \tilde{x} = x\} \\ &\quad + P\{\tilde{a} = 0 | \tilde{x} = x, \tilde{s} = 1\}\, P\{\tilde{s} = 1 | \tilde{x} = x\} \\ &= P\{\tilde{s} = 0 | \tilde{x} = x\} + (\varepsilon + (1 - \varepsilon)\varepsilon) \\ &\quad \times P\{\tilde{s} = 1 | \tilde{x} = x\}. \end{aligned} \quad (11)$$

The failure to receive a REPLY message is due either to loss of the REQUEST message in transmission channels between the new node and all neighbor nodes or due to loss of the responders' REPLY messages due to either collisions or external noise. Each neighbor node that received a REQUEST message will send its reply in a randomly chosen slot from the range $(1, 2, \ldots, W)$, where W denotes the contention window size. A REPLY message is received successfully if it is transmitted with no other messages in the same slot and has a good SINR value. Define $\tilde{r}$ to be the number of responders that received a REQUEST message and $\tilde{r}_i$ to be the number of responders that responded in slot number i. Then the probability of receiving at least one REPLY message successfully given that there are x neighbors is given by

$$P\{\tilde{s} = 1 | \tilde{x} = x\} = \sum_{r=1}^{x} P\left\{\bigcup_{i=1}^{W} \{\tilde{r}_i = 1\} | \tilde{r} = r\right\} P\{\tilde{r} = r | \tilde{x} = x\}. \quad (12)$$

The first term of the equation represents the probability that at least one transmission slot has exactly one REPLY message and that message has a good SINR value. The second term represents the probability that $\tilde{r}$ neighbors received the REQUEST message out of the total number of neighbors. Since message transmissions are assumed to have identical failure probabilities of ε and messages are transmitted independently, $\tilde{r}$ is a binomial random variable with parameters $(x, (1 - \varepsilon))$. The probability that r nodes will receive the message out of the total number of x neighbors is given by

$$P\{\tilde{r} = r \mid \tilde{x} = x\} = \binom{x}{r} (1 - \varepsilon)^{r}\, \varepsilon^{x - r}. \quad (13)$$

The probability that at least one slot has exactly one reply message given that there are r responders is equal to[2]

[2]This is simply the probability of the union of arbitrary events as given in any probability book [19].

$$P\left\{\bigcup_{i=1}^{W}\{\tilde{r}_i=1\}\,|\,\tilde{r}=r\right\}$$
$$=\sum_{i}P\{\tilde{r}_i=1\,|\,\tilde{r}=r\}-\sum_{i<j}P\{\tilde{r}_i=1,\tilde{r}_j=1\,|\,\tilde{r}=r\}$$
$$+\sum_{i<j<k}P\{\tilde{r}_i=1,\tilde{r}_j=1,\tilde{r}_k=1\,|\,\tilde{r}=r\}-\ldots$$
$$+(-1)^{W+1}P\{\tilde{r}_i=1,\tilde{r}_j=1,\tilde{r}_k=1,\cdots,\tilde{r}_W=1\,|\,\tilde{r}=r\}. \quad (14)$$

The first term of (14) represents the probability that slot i has exactly one message, in other words, exactly one responder transmitted the message in slot number i and that message when received had a good SINR value. That is

$$P\{\tilde{r}_i=1|\tilde{r}=r\}=\binom{r}{1}\frac{1}{W}\left(1-\frac{1}{W}\right)^{r-1}(1-\varepsilon). \quad (15)$$

The other terms of (14) represent joint probabilities of having more than one slot with exactly one message. Note that the sum of the slots that have exactly one reply message, $\tilde{r}_i=1$, is equal to the total number of replies r. That is the probability in the second term of the equation evaluates to zero if the number of replies is less than two. In general, the joint probability of having exactly one reply in each of the slots i, j, …, z where the $\tilde{r}_i+\tilde{r}_j+\cdots+\tilde{r}_z=h$ is equal to zero for $h>r$ and for the $h\le r$ case it is calculated as

$$P\{\tilde{r}_{i_1}=1,\cdots,\tilde{r}_{z_h}=1|\tilde{r}=r\}$$
$$=\binom{r}{h}h!\,\frac{(W-h)^{r-h}}{(W)^r}(1-\varepsilon)^h. \quad (16)$$

Substituting this result in (14) and performing some algebra yield

$$P\left\{\bigcup_{i=1}^{W}\{\tilde{r}_i=1\}\,|\,\tilde{r}=r\right\}$$
$$=\sum_{h=1}^{\min\{r,W\}}(-1)^{h+1}\binom{W}{h}\binom{r}{h}h!\,\frac{(W-h)^{r-h}}{W^r}(1-\varepsilon)^h. \quad (17)$$

Now, we consider the conditional expectations of the latency and communication overhead on the number of trials required for the new node to be allocated a network address. Recall that $\tilde{\kappa}$ represents the number of trials that has been performed, the expected latency when the address allocation succeeded on the kth trial, where $\kappa\le K_{\mathrm{NR}}$ is the sum of the timeouts for the first $(k-1)$ failed trials plus the timeout of receiving a REPLY message and the timeout of receiving the ADDRESS_BLOCK message in the last successful trial. The latency for a failed trial is the sum of the timeout to receive the REPLY message plus the timeout to receive the ADDRESS_BLOCK message if at least one reply is successfully received. That is

$$E[\tilde{\ell}|\tilde{x}=x,\tilde{\kappa}=\kappa]=(\kappa-1)(T_{\mathrm{NR}}+P\{\tilde{s}=1|\tilde{x}=x\}\,T_{\mathrm{AR}})$$
$$+T_{\mathrm{AR}}+T_{\mathrm{AR}}=\kappa\,T_{\mathrm{NR}}$$
$$+T_{\mathrm{AR}}(1+(\kappa-1)\,\mathrm{P}\,\{\tilde{s}=1|\tilde{x}=x\}), \quad (18)$$

where $P\{\tilde{s}=1|\tilde{x}=x\}$ is presented in (12).

The expected communication overhead for a successful address allocation on the kth trial is the sum of messages sent during the first $(k-1)$ trials and they are one REQUEST message, the expected number of REPLY messages, and one ACK message that is sent if at least one reply is received correctly. In the last trial, the new node sent a REQUEST and received at least one REPLY so it sent out an ACK message to one of the responders and received an ADDRESS_BLOCK message that contains a portion of the address space. Therefore, the expected communication overhead is equal to

$$E[\tilde{c}|\tilde{x}=x,\tilde{\kappa}=\kappa]=(\kappa-1)\,(1+E[\tilde{r}|\tilde{x}=x]$$
$$+P\{\tilde{s}=1|\tilde{x}=x\})$$
$$+(1+E[\tilde{r}|\tilde{x}=x]+2) \quad (19)$$
$$=\kappa(1+E[\tilde{r}|\tilde{x}=x])$$
$$+(\kappa-1)P\{\tilde{s}=1|\tilde{x}=x\}+2.$$

In the case where all trials have failed, that is $\kappa>K_{\mathrm{NR}}$, the expected latency and communication overhead are given by

$$E[\tilde{\ell}|\tilde{x}=x,\tilde{\kappa}>K_{\mathrm{NR}}]$$
$$=K_{\mathrm{NR}}\,(T_{\mathrm{NR}}+P\{\tilde{s}=1|\tilde{x}=x\}\,T_{\mathrm{AR}}) \quad (20)$$

and

$$E[\tilde{c}|\tilde{x}=x,\tilde{\kappa}>K_{\mathrm{NR}}]=K_{\mathrm{NR}}\,(1+E[\tilde{r}|\tilde{x}$$
$$=x]+P\{\tilde{s}=1|\tilde{x}=x\}). \quad (21)$$

Substituting (10), (18), (19), (20), and (21) into (8) and (9) gives

$$E[\tilde{\ell}\,|\,\tilde{x}=x]=\sum_{\kappa=1}^{K_{\mathrm{NR}}}\kappa\,T_{\mathrm{NR}}+T_{\mathrm{AR}}\,(1+(\kappa-1)$$
$$\times P\{\tilde{s}=1|\tilde{x}=x\})\cdot(P\{\tilde{a}=0\,|\,\tilde{x}=x\})^{\kappa-1}$$
$$\times P\{\tilde{a}=1\,|\,\tilde{x}=x\}+K_{\mathrm{NR}}$$
$$\times(T_{\mathrm{NR}}+P\{\tilde{s}=1|\tilde{x}=x\}\,T_{\mathrm{AR}})$$
$$\times(P\{\tilde{a}=0\,|\,\tilde{x}=x\})^{K_{\mathrm{NR}}}, \quad (22)$$

$$E[\tilde{c}\,|\,\tilde{x}=x]=\sum_{\kappa=1}^{K_{\mathrm{NR}}}(\kappa\,(1+E[\tilde{r}|\tilde{x}=x])+(\kappa-1)\times P\{\tilde{s}=1|\tilde{x}=x\}+2)\cdot(P\{\tilde{a}=0\,|\,\tilde{x}=x\})^{\kappa-1}\times P\{\tilde{a}=1\,|\,\tilde{x}=x\}+K_{\mathrm{NR}}\,(1+E[\tilde{r}|\tilde{x}=x]\times+P\{\tilde{s}=1|\tilde{x}=x\})\,(P\{\tilde{a}=0\,|\,\tilde{x}=x\})^{K_{\mathrm{NR}}},\quad(23)$$

where $P\{\tilde{a}=0\,|\,\tilde{x}=x\}$ is from (11) and $P\{\tilde{s}=1|\tilde{x}=x\}$ is from (12).

4. Numerical results

In this section, we present numerical results for latency and communication overhead based on analysis and simulation, and in addition, we consider the sensitivity of these measures to message loss rate, contention window size, and coverage ratio. In Section 4.1, a description of the simulation carried out for the IPAA and MANETconf protocols is presented. Then, Section 4.2 presents a comparison of the IPAA and MANETconf protocols. Section 4.3 presents the analytical and simulation results, which show good agreement, for the IPAA protocol as functions of network size with different parameter values. A summary of results is deferred to Section 5.

4.1. Simulation description

The network dimensions are set to 100 m × 100 m. Each node is placed at a uniformly distributed location within the network area. Results are collected for node populations from 0 to 50. For a given number of nodes in the network, the number of nodes within the transmission range of the new node depends on its coverage area. Results are collected for coverage ratios of 10%, 15%, and 20%. Each message sent from one node to another is subject to loss with rate ε; loss rates examined were 0, 0.1, and 0.2. Contention window sizes, W, considered were 10, 20, and 30. For the IPAA protocol, the timeout value to find a neighbor T_{NR} was set to 0.2 s and the timeout to receive the Address_Block message T_{AR} was set to 0.02 s, and K_{NR} was varied from 3 to 5 to show the effect of the counter threshold on the performance measures of the protocol. For MANETconf protocol T_{NR} was set to 0.2 s and T_{AR} was set to 2 s, and K_{NR} was set to 3 and K_{AR} was set to 5.

The simulation results presented are the average values of 10000 simulation runs. For large number of simulation runs, the sample mean of a performance measure for any network size follows the normal distribution $N(\mu, \frac{s}{\sqrt{n}})$, where μ is the true population mean, s is the sample standard deviation, and n is the sample size. Throughout simulation, we have found that the maximum value of the sample standard deviation is 0.002 for latency samples and it is 0.1 for communication overhead samples for all values of network size. The 95% confidence interval, that is likely to include the true population mean of a performance measure, is equal to $(\bar{a} \pm 1.96\,\frac{s}{\sqrt{n}})$, where $\bar{a}$ represents the sample mean for the performance measure. Therefore, the 95% confidence interval for latency samples is $(\bar{x} - 3.92 \times 10^{-5}, \bar{x} + 3.92 \times 10^{-5})$ and for communication overhead it is $(\bar{y} - 0.002, \bar{y} + 0.002)$.

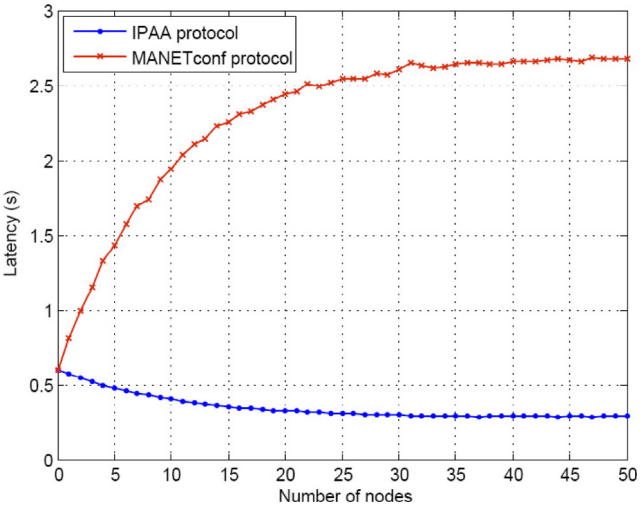

Figure 6. Latency of the IPAA and MANETconf protocols as a function of network size.

4.2. IPAA versus MANETconf protocol

Figure 6 compares the latency of IPAA protocol to the latency of the MANETconf protocol. In MANETconf, the initiator node selects an address and performs DAD process throughout the network. Therefore, the latency of the allocation process increases as the number of nodes in the network increases. In IPAA, the address allocation process is carried out by an existing neighbor or it may be carried out by the new node itself in case no neighbor exists. For large number of nodes, the probability of finding a node within the transmission range of the new node is high so that address allocation can be carried out by a neighbor node and therefore it decreases as the number of nodes increases.

Figure 7 presents the communication overhead for IPAA and MANETconf protocols. Since MANETconf performs the DAD process throughout the network, the number of messages sent increases as the number of nodes increases. The IPAA protocol performs address allocation through local communications with neighbor nodes only. Therefore, the number of messages sent is less than that for the MANETconf protocol.

4.3. Analytical and simulation results for the performance measures of the IPAA protocol

Figures 8 and 9 represent the latency and communication overhead at loss rates ε = {0, 0.1, 0.2}. The contention

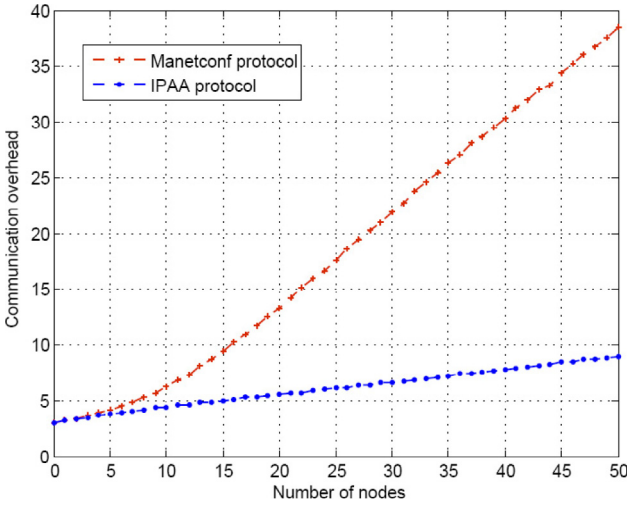

Figure 7. Communication overhead of the IPAA and MANETconf protocols as a function of network size.

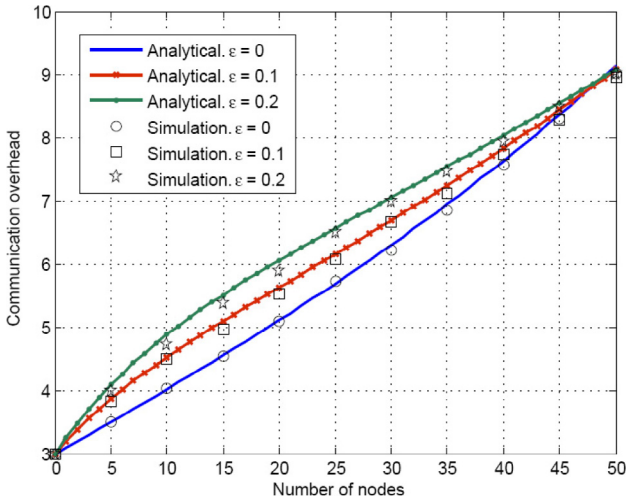

Figure 9. Communication overhead of the IPAA protocol as a function of network size with message loss rate as a parameter.

window size is set to $W = 30$, K_{NR} is set to 3, and the coverage ratio is set to $\frac{A_x}{A} = 10\%$. The results show that the latency increases as the loss rate increases since a loss of the protocol messages may result in a failed trial and force the new node to start the process again. For a relatively small number of nodes in the network, the communication overhead increases as the loss rate increases since there exist a small number of neighbor nodes within the transmission range of the new node so the probability of loss of the protocol message is high. For a large number of neighbors, the probability of loss for all messages is lower and the number of messages becomes closer to the case where no loss is assumed.

Figures 10 and 11 show the effect of the contention window size W on the latency and the communication

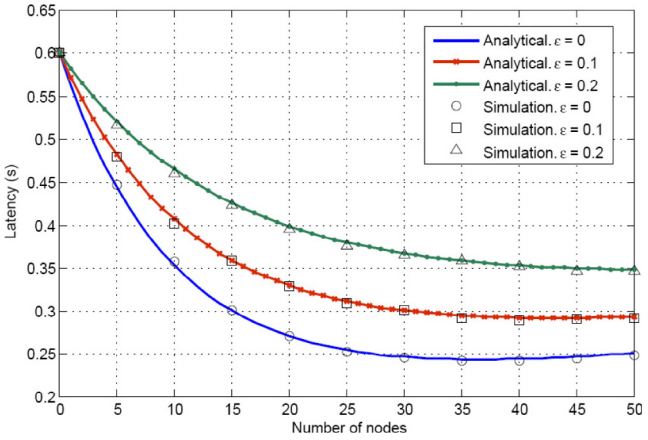

Figure 8. Latency of the IPAA protocol as a function of network size with message loss rate as a parameter.

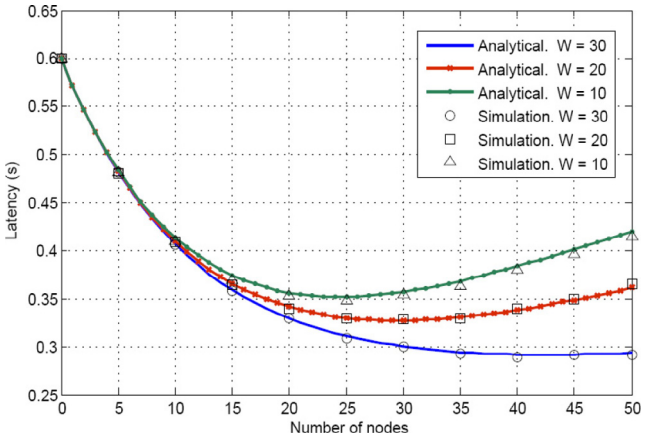

Figure 10. Latency of the IPAA protocol as a function of network size with contention window size as a parameter.

overhead, respectively, when $\varepsilon = 0.1$, $\frac{A_x}{A} = 10\%$, and $K_{NR} = 3$. The window size has a greater effect on latency and communication overhead for a large number of nodes since the collision probability increases as the number of nodes increases. For a given large number of nodes, decreasing the window size results in more collisions for the incoming reply messages at the new node which may result in a failed allocation trial. More failed trials result in more latency and more communication overhead as shown in the figures.

Numerous additional analytical and simulation results were carried out and are given in [20]. Also analyzed were the effect of the coverage ratio $\frac{A_x}{A}$ on the latency and communication overhead, effect of the protocol parameter K_{NR} on the latency and communication overhead, and effect of the loss ratio on the number of neighbor allocations. In all cases analytical and simulation results showed close agreement.

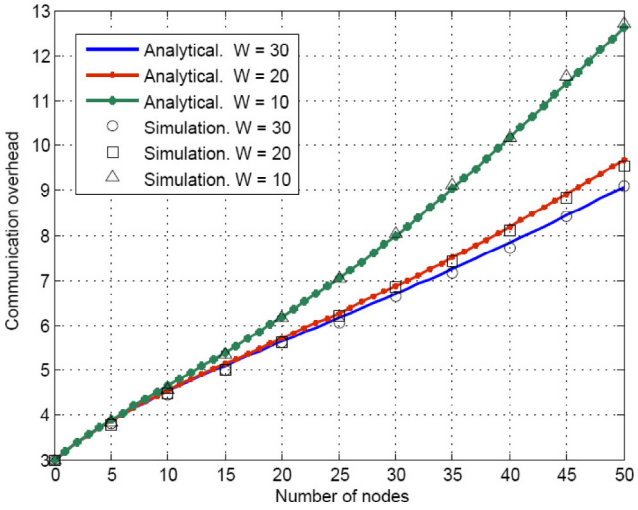

Figure 11. Communication overhead of the IPAA protocol as a function of network size with contention window size as a parameter.

5. Conclusions

Detailed descriptions for two address allocation protocols in wireless *ad hoc* network were presented. One of the protocols, the MANETconf protocol of [15], represents the DAD-based address allocation scheme and the other, the IPAA protocol of [16], represents the neighbor-based scheme. State machines that show the protocols' behavior are constructed for the two protocols mentioned above. The state machine gives a complete picture of the states, handshakes, timers, and types of messages for a protocol. For IPAA, analytical formulas for the expectation of latency and communication overhead are carried out as a function of the number of nodes in the network with message loss rate, contention window size, coverage ratio, and the counter threshold as parameters. Latency is the amount of time required for a new node to be allocated a network address, and communications overhead is the number of messages sent during the address allocation process.

Extensive numerical results were collected. The results show that the latency and communication overhead for MANETconf are higher than those measures for the IPAA protocol and that is due to performing the DAD process in MANETconf throughout the network. In IPAA, address allocation is performed through local communication between the new node and neighbor nodes and therefore its latency and communication overhead are low. It has been shown that the latency and communication overhead of the allocation process for the IPAA protocol increase as the message loss rate increases. Loss of REPLY or ADDRESS_BLOCK messages may result in a failed trial and requires the new node to repeat the allocation process again. The contention window size has an effect on the performance measures for relatively large number of nodes in the network. As the number of nodes in the network increases, the number of neighbor nodes increases and therefore the probability of collision increases which may result in a failed trial and forces the new node to perform the allocation process again. Additional numerical results given in [20] show that increasing the coverage ratio has basically the same effect as increasing the number of nodes within the transmission range of the new node which leads to high collision probability as mentioned above. Numerical results presented in [20] also show that for loss rate equal to 0.1, contention window size of 30, and coverage ratio of 10%, address allocation is carried out by neighbor nodes for almost 95% of the time when the number of nodes in the network is more than 30.

The address allocation problem for *ad hoc* networks is still unsolved. The work done in this paper provides understanding of the nature of the problem and the way to evaluate the performance of an allocation protocol. The objective of future research will be to contribute to the solution of the address allocation problem in *ad hoc* networks. What is needed is a protocol that handles all types of exceptions such as message losses, node departures, network partitioning, and merging. An address allocation protocol for an *ad hoc* network has to be a dynamic and distributed protocol that guarantees address uniqueness for each node in the network and scalable to large networks. Scalability means that the protocol should perform well as the network size increases. The protocol should perform the address allocation for a node with minimum communication overhead and in a timely fashion. Besides the verbal description of the protocol, state machines that show the protocol states, handshakes, types of messages, and timers should be constructed. For completeness, the performance of the protocol should be analyzed and compared to the performance of existing protocols.

References

[1] Thomson, S. and Narten, T. (1998) Ipv6 stateless address autoconfiguration, RFC 2462, IETF Zeroconf Working Group.

[2] Droms, R. (1997) Dynamic host configuration protocol, RFC 2131, IETF Network Working Group.

[3] Baccelli, E. (2007, September) Address autoconfiguration for MANET: terminology and problem statement, IETF draft-ietfautoconf-statement-02.

[4] Knowlton, K.C. (1965) A fast storage allocator. *Commun. ACM* **8**(10): 623–624.

[5] Bernardos, C., Calderon, C. and Moustafa, H. (2008, April) Survey of IP address autoconfiguration mechanisms for MANETS, IETF draft-bernardos-manet-autoconf-survey-03.

[6] Perkins, C. (2001, November) IP address autoconfiguration for ad hoc networks, IETF draft-perkins-manet-autoconf-01.

[7] Jeong, J. (2006, January) Ad hoc IP address autoconfiguration, IETF draft-jeong-adhoc-ip-addr-autoconf-06.

[8] Vaidya, N. (2002) Weak duplicate address detection in mobile ad hoc networks. In *Proceedings of ACMMOBIHOC* (Lausanne).

[9] Weniger, K. (2003, March) Passive duplicate address detection in mobile ad hoc networks. In *Proceedings of IEEE WCNC, IEEE* (New York: IEEE Press).

[10] Mase, K. and Adjih, C. (2006, April) No overhead autoconfiguration OLSR, IETF draft-mase-manet-autoconf-noaolsr-01.

[11] Weniger, K. (2005) PACMAN: passive autoconfiguration for mobile ad hoc networks. *IEEE J. Sel. Areas Commun.* **23**(3): 507–519.

[12] Zhou, H., Ni, L. and Mutka, M. (2003) Prophet address allocation for large scale MANETs. In *Proceedings of IEEE INFOCOM* (New York: IEEE INFOCOM).

[13] Yong, L., Ping, Z. and JiaXiong, L. (2009) Dynamic address allocation protocols for mobile ad hoc networks based on genetic algorithm. In *Proceedings of 5th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)* (Piscataway, NJ: IEEE Press).

[14] Chu, X., Sun, Y., Xu, K., Sakander, Z. and Liu, J. (2008) Quadratic residue based address allocation for mobile ad hoc networks. In *Proceedings of IEEE ICC* (Beijing, China), 2343–2347.

[15] Nesargi, S. and Parakash, R. (2002) MANETconf: configuration of hosts in a mobile ad hoc network. In *Proceedings of IEEE INFOCOM, IEEE* (New York: IEEE Press).

[16] Mohsin, M. and Parakash, R. (2002) IP address assignment in a mobile ad hoc network. In *Proceedings of IEEE MILCOM* (New York: IEEE Press).

[17] Tayal, A.P. and Patnaik, L.M. (2004) An address assignment for the automatic configuration of mobile ad hoc networks. *Pers. Ubiquitous Comput.* **8**(1): 47–54.

[18] Xu, T. and Wu, J. (2007) Quorum based IP address autoconfiguration in mobile ad hoc networks. In *ICDCSW '07: Proceedings of the 27th International Conference on Distributed Computing Systems Workshops* (Washington, DC: IEEE Computer Society), 1.

[19] Ross, S.M. (2008) Introduction to probability models (Academic Press), 9th ed.

[20] Radaideh, A.M. (2008, December) Analytical modeling of address allocation protocols in wireless ad hoc networks. Master's thesis, The University of Mississippi.