Modeling and Performance Analysis of Address Allocation Schemes for Mobile *Ad Hoc* Networks

Sehoon Kim, Jinkyu Lee, *Student Member, IEEE*, and Ikjun Yeom, *Member, IEEE*

***Abstract*—Address allocation is an essential part in maintaining a mobile *ad hoc* network (MANET) effectively, and several address allocation schemes have been proposed. In this paper, we present a set of analytical models to evaluate the efficiency of address allocation schemes. The derived models quantitatively characterize the efficiency of four popular address allocation schemes in terms of latency and communication overhead. Through the analysis, we achieve numerical results that show the impact of network parameters on the efficiency of these schemes. We also conduct simulations and compare with analytical results to validate our models. The analytical model developed in this paper is able to more accurately predict the performance of address allocation schemes over a various range of loss rates and would be useful in providing more insights for the study of an efficient address allocation scheme in MANETs. To our understanding, this is the first attempt in mathematically investigating the performance of addressing schemes in *ad hoc* networks.**

***Index Terms*—Dynamic addressing scheme, mobile *ad hoc* network (MANET), performance analysis.**

I. Introduction

A MOBILE *ad hoc* network (MANET) is a collection of mobile nods forming a temporary network without any established infrastructure or centralized administration. In a MANET, it is important to establish a multihop routing path for mobile nodes to communicate over several intermediate nodes. There have been several protocols that were proposed for multihop routing, such as destination-sequenced distance-vector routing [1], *Ad hoc* On-demand Distance-Vector routing (AODV) [2], and dynamic source routing [3]. These protocols attempt to find an effective path through dynamically moving nodes based on the assumption that each node has a unique address. However, unlike fixed wired networks, mobile nodes are frequently in and out of a MANET, and thus, it is not easy to assign a unique address to each node. To maintain a MANET effectively, it is essential to allocate an address to each mobile node.

There have been several schemes that were proposed for address allocation in a MANET. The key technique of address allocation is how to guarantee the uniqueness of the assigned addresses, and we may categorize the proposed schemes into four different families based on their techniques for unique address allocation. In decentralized schemes [4]–[6], a global agreement is required for each address allocation. In centralized schemes (leader-based schemes) [7]–[10], a single node (leader) is in charge of address allocation to avoid inconsistency. In neighbor-based schemes [11]–[14], to provide uniqueness with only local communication, a disjoint address space is maintained by each node, or a specially designed allocation function is needed. In passive schemes [15], [16], address conflict is passively detected via routing information.

Since these schemes take quite different approaches with different assumptions on the network condition, it is hard to compare their performance directly. There have been several researches on performance comparison of address allocation schemes based on simulation or asymptotic analysis [14], [17], [18]. However, simulation-based comparison is limited by several specific topologies, and asymptotic analysis does not provide enough details of their performance. The immediate motivation of this paper arises from the lack of abstract models for performance of various address allocation schemes.

To evaluate the efficiency of address allocation schemes, there are two widely used metrics: 1) address allocation latency and 2) communication overhead. Address allocation latency is defined as the time taken from when a node requests an address to when it is assigned an address. This includes all possible delays that are caused by message exchanges and timeouts. Communication overhead is the number of control packets that are transmitted during the address allocation process. Here, each hopwise packet transmission is counted as one transmission.

In this paper, we focus on the efficiency of IP address autoconfiguration (IPAA) [4], MANETconf [5], a token-based scheme [10], and neighbor-based address allocation schemes. IPAA and MANETconf are selected to represent decentralized schemes, and the token-based scheme is selected to represent centralized schemes. For neighbor-based schemes, we do not select a specific one since they operate similarly from a performance perspective.

The objective of this paper is to derive a set of analytical models to evaluate the efficiency of address allocation schemes. The derived models characterize these schemes as functions of the number of nodes, packet loss rate, and the number of available addresses. Through analytical evaluation, we comprehensively study the impact of network parameters on the efficiency of these schemes and compare them. The models are validated through simulations using $ns-2$ [19]. The main contribution of this paper is to present analytical models to evaluate the efficiency of address allocation schemes in a MANET. To our knowledge, this is the first quantitative

Manuscript received February 17, 2006; revised September 8, 2006, January 15, 2007, and April 19, 2007. This work was supported by the University IT Research Center Project. The review of this paper was coordinated by Prof. Y.-C. Tang.

The authors are with the Department of Computer Science, Korea Advanced Institute of Science and Technology, Daejeon 305-701, Korea (e-mail: kimsh@cnlab.kaist.ac.kr; jklee@cnlab.kaist.ac.kr; yeom@cs.kaist.a.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2007.901859

study on address allocation in a MANET. We believe that this paper plays an important role for understanding and further study on address allocation process in a MANET.

The rest of this paper is organized as follows: We categorize and summarize known address allocation schemes in Section II. In Section III, we provide an overview of our analysis, followed by a detailed description of analytical models for address allocation schemes. In Section IV, we analyze the efficiency of address allocation schemes and validate our models with $ns-2$ simulation. We conclude this paper in Section V.

II. Background and Related Work

A. Decentralized Schemes

In decentralized schemes, each node independently configures its address, and then, the uniqueness of the address is validated through global agreement from all the other nodes in the network. Therefore, the performance of these schemes usually relies on network-wide flooding.

IPAA [4] adopts a trial-and-error policy to find an available Internet Protocol (IP) address for a new node. A new node (requester) randomly selects two IP addresses each from the IP address block, which is divided into two categories: 1) a temporary address for duplicate address detection (DAD) and 2) the actual address to use for communication. The former is used only once as the source address of the new node during the DAD phase. The node floods a request with the selected address to check if there already exists a node using the same address. If no reply is returned for the selected address, the node considers that the address is free and takes that address for its own. Otherwise, the node randomly selects another address and repeats the aforementioned procedure.

MANETconf [5] is based on a distributed mutual exclusion algorithm to check the uniqueness of the address. Assigning an address to a new node requires an agreement from all the known nodes in the network. In this scheme, each node has a global address allocation table that maintains currently in-use addresses. When a new node comes up, it selects one of its neighbors as its agent (called an initiator in MANETconf), which performs address allocation on behalf of the requester. The agent picks a currently unused address from its table and floods a message to obtain an agreement from all the other configured nodes in the network. Upon receiving a message from the agent, each node in the network sends a reply back to the agent. If the agent receives a reply from all the other nodes, it assigns the address to the requester by sending a message.

B. Centralized Schemes (Leader-Based Schemes)

In centralized schemes, there is a single server (called leader) in a network to assign addresses to new nodes without conflict. As long as there exists only a single server, the uniqueness of allocated addresses is naturally guaranteed. The technical challenge of centralized schemes is how to maintain a single server in a MANET, in which mobile nodes are constantly in and out.

In the token-based scheme [10], the node holding a token, which is called the allocator, takes charge of address allocation to a new node. The address allocation procedure is similar to that of MANETconf, except that it does not require a global agreement since a single node has the right to assign addresses. The agent unicasts a message only to the allocator. Upon receiving a message, the allocator sends a reply back to the agent. The agent forwards the new address to the requester. In this scheme, all configured nodes in a network maintain an address of the allocator.

C. Neighbor-Based Schemes

In neighbor-based schemes, address allocation is performed only via local communication with neighbor nodes. Hence, they show better performance in terms of communication overhead and latency than other schemes. The key technique in neighbor-based schemes is to guarantee uniqueness without global agreement, as in decentralized schemes, or centralized control, as in centralized schemes.

In [11] and [12], the concept of binary splitting is employed for the uniqueness of allocated addresses. The first node has the entire pool of addresses initially. When a new node joins the network, one of its neighbors hands over half of its pool of addresses. By recursively splitting the address pool, each node can maintain a disjoint address pool, and new nodes can obtain their addresses through only local communication without address conflict.

In [13], a prime numbering address allocation algorithm called Prime DHCP has been proposed. It has originated from the canonical factorization theorem of positive integers, i.e., every positive integer can be written as a product of prime numbers in a unique way. The first node in a MANET has an address of 1 and assigns prime numbers, in ascending order, to new nodes that are attached to it. Another node can assign the address equal to its own address multiplied by the unused prime number, starting from the largest prime factor of its own address. Each node needs to maintain its allocation status to record the last assigned address.

The prophet scheme in [14] uses a stateful address generation function $f(n)$ for each node to generate a series of random numbers to be assigned to new coming nodes. Function $f(n)$ is carefully designed, so that the possibility of duplicates is kept low. The initial state of $f(n)$ is called the seed. Different seeds lead to different sequences, and the state of $f(n)$ is updated. The address allocation process is similar to a Prime DHCP.

III. Modeling and Performance Analysis

A. Overview of Analysis

In the following sections, we model the behaviors of address allocation schemes in the presence of packet loss. We assume that mobile nodes are uniformly distributed in a network topology with a common transmission range. We assume that flooded packets are eventually delivered to all the nodes, even in the presence of packet loss, since there are sufficient duplicated packets. This is a reasonable assumption, as presented in [20].

To facilitate the subsequent analysis, we define several constants in Table I. We also define several random variables. Let

TABLE I
CONSTANT DEFINITIONS

T_{ne}	MANETconf Token-based Neighbor-based	Timeout interval set by a requester when it sends a message to neighbors
T_{ad}	IPAA Neighbor-based	Timeout interval set by a requester when it sends a request to allocate an address for a requester
	MANETconf Token-based	Timeout interval set by an agent when it sends a request to allocate an address for a requester
T_{re}	MANETconf Token-based	Timeout interval set by a requester when it sends a request to the agent for address allocation
K_{ne}	MANETconf Token-based Neighbor-based	The maximum number of times a requester tries to find neighbors.
K_{ad}	IPAA Neighbor-based	The maximum number of times a requester tries to allocate an address for a requester
	MANETconf Token-based	The maximum number of times an agent tries to allocate an address for a requester
K_{re}	IPAA	The maximum number of addresses a requester tries to check the address duplication
	MANETconf Token-based	The maximum number of times a requester requests an address to the agent

N be the number of mobile nodes in a network. Let $loss_{\text{br}}$ and $loss_{\text{un}}$ denote the average loss rates of broadcast and unicast packets in link level, respectively. t_{br} and t_{un} are the average delays of broadcast and unicast packets between two adjacent nodes, respectively. Let h_{av} and h_{mx} be the average and maximum numbers of hops on a path between any two end nodes, respectively. Finally, let R denote the average proportion of the transmission range of a node over a given network topology. When the network topology is relatively much larger than the transmission range, it can be simply defined as $\pi \cdot r^2/A$, where r is the transmission range of a node and A is the area of the network topology. Otherwise, R is determined by the shape of the network topology and the location of an individual node. While deriving our models, we assume that the topology is large enough for simplicity. A similar assumption has been made in [21]. In Section IV, we describe how to calculate R more precisely.

Here, note that we use different variables for unicast and broadcast packets since their characteristics in link level are quite different. A similar observation has been presented in [22].

B. IPAA

Fig. 1 depicts the address allocation process of IPAA. A new node (requester) selects a random address and floods *ADDR_REQ* (*Msg*. 1 in Fig. 1). Then, it waits a reply for T_{ad}. If there exists a node with the same address, the node announces the collision to the requester via *ADDR_REP* (*Msg*. 2 in Fig. 1). If there is no response during K_{ad} times *ADDR_REQ*, address allocation is successfully finished. In the case of address allocation success, latency L_{su} and the number of packets sent C_{su} are given by

$$L_{\text{su}} = K_{\text{ad}} \cdot T_{\text{ad}} \tag{1}$$
$$C_{\text{su}} = K_{\text{ad}} \cdot N. \tag{2}$$

If there is *ADDR_REP*, which means that address duplication is detected, the requester selects another address

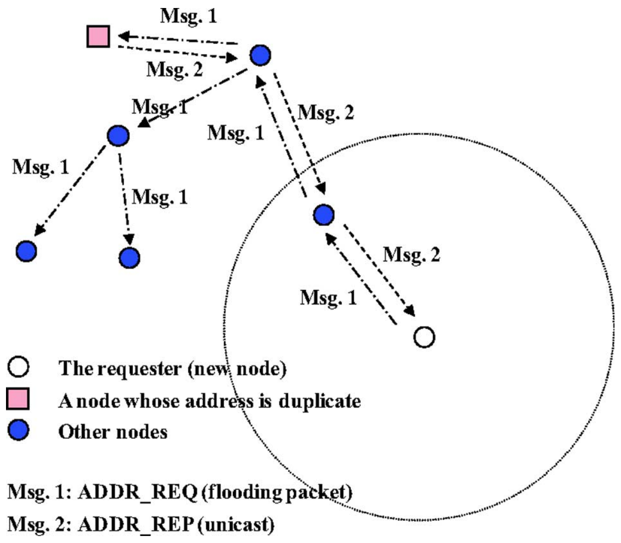

Msg. 1
Msg. 2
Msg. 1
Msg. 1
Msg. 2
Msg. 1
Msg. 1
Msg. 2
Msg. 1
The requester (new node)
A node whose address is duplicate
Other nodes
Msg. 1: ADDR_REQ (flooding packet)
Msg. 2: ADDR_REP (unicast)

Fig. 1. Address allocation process of IPAA.

and tries the aforementioned process again. The number of *ADDR_REP* messages sent can be expressed as a geometric distribution. If i number of *ADDR_REP* messages are sent, the latency is the sum of $i-1$ timeouts and the time interval from sending *ADDR_REQ* via broadcast and to receiving *ADDR_REP* via unicast. As we mentioned, flooded packets are assumed to be delivered eventually. Then, the expected latency L_{du} in the presence of address duplication is given by

$$L_{\text{du}} = \sum_{i=1}^{K_{\text{ad}}} \left\{ \frac{loss_{\text{mul}}^{i-1}(h_{\text{av}}) \cdot (1 - loss_{\text{mul}}(h_{\text{av}}))}{1 - loss_{\text{mul}}^{K_{\text{ad}}}(h_{\text{av}})} \cdot ((i-1) \cdot T_{\text{ad}} + (t_{\text{br}} + t_{\text{un}}) \cdot h_{\text{av}}) \right\} \tag{3}$$

where $loss_{\text{mul}}(h)$ is the probability that there is a loss when a unicast packet passes through h hops. Since the probability of nonexistence of a loss through h hops is $(1 - loss_{\rm un})^h$, $loss_{\rm mul}(h)$ is given by

$$loss_{\rm mul}(h) = 1 - (1 - loss_{\rm un})^h. \tag{4}$$

The expected number of packets sent in the presence of address duplication $C_{\rm du}$ is similar to $L_{\rm du}$. If i number of *ADDR_REP* messages are sent, the number of packets sent is the sum of i flooding packets, $i - 1$ *ADDR_REP* messages that are lost, and one *ADDR_REP* message that is delivered to the requester. So, we get

$$C_{\rm du} = \sum_{i=1}^{K_{\rm ad}} \left\{ \frac{loss_{\rm mul}^{i-1}(h_{\rm av}) \cdot (1 - loss_{\rm mul}(h_{\rm av}))}{1 - loss_{\rm mul}^{K_{\rm ad}}(h_{\rm av})} \cdot (i \cdot N + (i-1) \cdot h_{\rm lo}(h_{\rm av}) + h_{\rm av}) \right\} \tag{5}$$

where $h_{\rm lo}(h)$ is the expected number of actually delivered hops of a packet, which is sent on a path with h hops but dropped. The loss rate of each hop is $loss_{\rm un}$, and the probability of the number of delivered hops is calculated by

$$h_{\rm lo}(h) = \sum_{i=1}^{h} \left\{ \frac{(1 - loss_{\rm un})^{i-1} \cdot loss_{\rm un}}{1 - (1 - loss_{\rm un})^h} \cdot i \right\}. \tag{6}$$

The overall latency of address allocation is the sum of the latency of, at most, $K_{\rm re} - 1$ times address duplication and address allocation success for successful allocation and the latency of $K_{\rm re}$ times address duplication. So, the overall latency of address allocation is given by

$$L_{\rm total} = \sum_{i=1}^{K_{\rm re}} \left\{ P_{\rm du}^{i-1} \cdot (1 - P_{\rm du}) \cdot ((i-1) \cdot L_{\rm du} + L_{\rm su}) \right\} + P_{\rm du}^{K_{\rm re}} \cdot K_{\rm re} \cdot L_{\rm du} \tag{7}$$

where $P_{\rm du}$ is the probability of address duplication when a requester selects a random address and is given by

$$P_{\rm du} = \frac{N-1}{A} \tag{8}$$

where A is the number of available addresses, and there are $N - 1$ nodes in the network, except for the requester.

The total number of packets sent for address allocation $C_{\rm total}$ is similarly derived as in $L_{\rm total}$ and is given by

$$C_{\rm total} = \sum_{i=1}^{K_{\rm re}} \left\{ P_{\rm du}^{i-1} \cdot (1 - P_{\rm du}) \cdot ((i-1) \cdot C_{\rm du} + C_{\rm su}) \right\} + P_{\rm du}^{K_{\rm re}} \cdot K_{\rm re} \cdot C_{\rm du}. \tag{9}$$

C. MANETconf

The address allocation process in MANETconf starts when a new node (requester) requests an address and ends when it acquires one from another node or it configures itself. We derive models for two cases: When a node joins a MANET, the node is either the first one of the network or not.[1]

1) Case 1 (No Neighbor): If a requester is the first one of the network, the requester sends *NEIGH_REQ* (*Msg*. 1 in Fig. 2) and waits for *NEIGH_REP* (*Msg*. 2 in Fig. 2). Whenever the requester sends *NEIGH_REQ*, it waits to receive replies from neighbors until the timer expires. Since there is no neighbor, the requester will configure itself after repeating this process $K_{\rm ne}$ times. L_1 is the latency to create an address when there are no neighbors and is given by

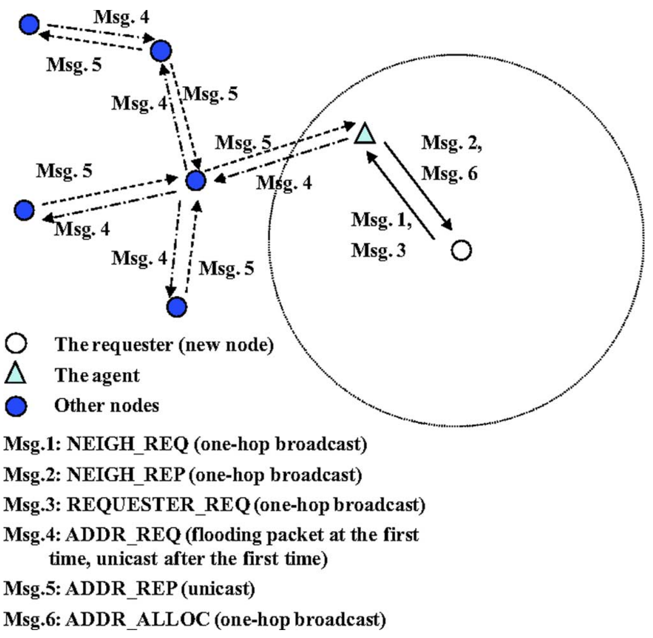

Msg. 4
Msg. 5
Msg. 4
Msg. 5
Msg. 5
Msg. 4
Msg. 5
Msg. 4
Msg. 4
Msg. 5
Msg. 2,
Msg. 6
Msg. 1,
Msg. 3
The requester (new node)
The agent
Other nodes
Msg.1: NEIGH_REQ (one-hop broadcast)
Msg.2: NEIGH_REP (one-hop broadcast)
Msg.3: REQUESTER_REQ (one-hop broadcast)
Msg.4: ADDR_REQ (flooding packet at the first time, unicast after the first time)
Msg.5: ADDR_REP (unicast)
Msg.6: ADDR_ALLOC (one-hop broadcast)

Fig. 2. Address allocation process of MANETconf.

$$L_1 = K_{\rm ne} \cdot T_{\rm ne}. \tag{10}$$

For this process, the requester sends $K_{\rm ne}$ packets, so the number of packets sent when there are no neighbors is given by

$$C_1 = K_{\rm ne}. \tag{11}$$

Since the probability that a node is not a neighbor of the requester is $1 - R$ and there are $N - 1$ nodes in the network, except for the requester, the probability that there is no neighbor P_1 is calculated by

$$P_1 = (1 - R)^{N-1}. \tag{12}$$

2) Case 2 (At Least One Neighbor): Let $P_{\rm ne}$ be the probability that a requester does not receive any *NEIGH_REP*. This happens due to either *NEIGH_REQ* loss, whose probability is $loss_{\rm br}$, or *NEIGH_REP* loss after receiving *NEIGH_REQ*, whose probability is $(1 - loss_{\rm br}) \cdot loss_{\rm br}$. Thus, we have

$$P_{\rm ne} = (loss_{\rm br} + (1 - loss_{\rm br}) \cdot loss_{\rm br})^n \tag{13}$$

[1] To simplify the model, we neglect the case when the new node misunderstands that it is the first one due to repeated message losses.

where n is the expected number of neighbors within a transmission range of the requester and is given by

$$n = (N-1)\cdot R. \tag{14}$$

We consider the expected latency $L_{\rm ne}$ and the expected number of packet sent $C_{\rm ne}$ to find neighbors. The requester sends *NEIGH_REQ* (*Msg*. 1 in Fig. 2) and waits for *NEIGH_REP* (*Msg*. 2 in Fig. 2) from its neighbors. If there is no reply until timeout $T_{\rm ne}$, the requester repeats sending *NEIGH_REQ*, at most, $K_{\rm ne}$ times. The number of repeated *NEIGH_REQ* is determined by a geometric distribution. If *NEIGH_REQ* is repeated i times, the latency is $i\cdot T_{\rm ne}$, and the expected number of packets sent is i multiplied by the sum of one (which is the number of *NEIGH_REQ*) and $(1-loss_{\rm br})\cdot n$ (which is the expected number of *NEIGH_REP*). Then, we have

$$L_{\rm ne} = \sum_{i=1}^{K_{\rm ne}} \left\{ \frac{P_{\rm ne}^{i-1}\cdot(1-P_{\rm ne})}{1-{P_{\rm ne}}^{K_{\rm ne}}}\cdot i\cdot T_{\rm ne}\right\} \tag{15}$$

$$C_{\rm ne} = \sum_{i=1}^{K_{\rm ne}} \left\{ \frac{P_{\rm ne}^{i-1}\cdot(1-P_{\rm ne})}{1-P_{\rm ne}^{K_{\rm ne}}}\cdot i\cdot (1+(1-loss_{\rm br})\cdot n)\right\}. \tag{16}$$

Now, we consider the communication between an agent and the allocator, where the agent is one of the neighbors who sends *NEIGH_REQ* first. $L_{\rm ad}$ and $C_{\rm ad}$ are the expected latency and the expected number of packets from sending *ADDR_REQ* (*Msg*. 4 in Fig. 2) to receiving *ADDR_REP* (*Msg*. 5 in Fig. 2), respectively. $L_{\rm ad}$ is the sum of the expected latency in case where the number of repeated *ADDR_REQ* changes from 1 to $K_{\rm ad}$. We define the number of repeated *ADDR_REQ* as -1 when the agent does not receive *ADDR_REP* for $K_{\rm ad}$ trials. $C_{\rm ad}$ is similarly calculated as in $L_{\rm ad}$. We have

$$L_{\rm ad} = \sum_{i=1}^{K_{\rm ad}} \{P_{\rm ad}(i)\cdot l_{\rm ad}(i)\} + P_{\rm ad}(-1)\cdot l_{\rm ad}(-1) \tag{17}$$

$$C_{\rm ad} = \sum_{i=1}^{K_{\rm ad}} \{P_{\rm ad}(i)\cdot c_{\rm ad}(i)\} + P_{\rm ad}(-1)\cdot c_{\rm ad}(-1) \tag{18}$$

where $P_{\rm ad}(i)$ is the probability of receiving the reply of the ith *ADDR_REQ*. $l_{\rm ad}(i)$ and $c_{\rm ad}(i)$ are the latency and the expected number of packets sent at that time, respectively.

The first *ADDR_REQ* is flooded, and then, subsequent *ADDR_REQ*s are sent via unicast. All *ADDR_REP*s are sent via unicast. The address allocation process is finished only when all the nodes reply *ADDR_REP*. Since the probability that the agent receives a particular *ADDR_REP* is $1-loss_{\rm mul}(h_{\rm av})$ and that there are $N-2$ nodes in the network, except for the requester and the agent, the probability to succeed in address allocation with only one *ADDR_REQ* is $(1-loss_{\rm mul}(h_{\rm av}))^{N-2}$. The probability to succeed in address allocation within more than one *ADDR_REQ* is similar to that with only one *ADDR_REQ*, but the expected number of nodes that send *ADDR_REP* is different. The probability that the agent fails to receive *ADDR_REP* is the complementary event of the sum of other probabilities, so $P_{\rm ad}(i)$ is calculated by (19), shown at the bottom of the page.

$l_{\rm ad}(i)$ is the expected latency to succeed in address allocation with the ith *ADDR_REQ*. If address allocation succeeds with only one *ADDR_REQ*, the latency is the time interval from sending *ADDR_REQ* via broadcast to receiving *ADDR_REP* via unicast. If address allocation succeeds with i trials of *ADDR_REQ*s, it takes $i-1$ times of timeout and latency to communicate *ADDR_REQ* via broadcast and *ADDR_REP* via unicast, respectively. If address allocation fails after the $K_{\rm ad}$th trial, the latency is simply $T_{\rm ad}$ multiplied by $K_{\rm ad}$. Then, $l_{\rm ad}(i)$ is given by (20), shown at the bottom of the page.

$c_{\rm ad}(i)$ is the expected number of packets sent to succeed in address allocation with the ith *ADDR_REQ*. The way to calculate $c_{\rm ad}(i)$ is similar to the way to calculate $P_{\rm ad}(i)$. If the agent receives all *ADDR_REP*s at once, the number of packets sent is the sum of N (flooding cost) and $(N-2)\cdot h_{\rm av}$ (the reply cost of all nodes, except the requester and the agent). If the agent receives all *ADDR_REP*s in more than one try, the number of packets sent is calculated by the expected number of nodes that send *ADDR_REQ* and *ADDR_REP* and the expected number of delivered hops. $c_{\rm ad}(i)$ is given by (21), shown at the bottom of the next page.

When the requester sends *REQUESTER_REQ* (*Msg*. 3 in Fig. 2), it may not receive *ADDR_ALLOC* (*Msg*. 6 in Fig. 2) due to the following three types of losses. First, *REQUESTER_REQ* may be lost. Since *REQUESTER_REQ* is a broadcast message to neighbors, the probability of this case $P_{\rm re1}$ is simply $loss_{\rm br}$. Second, *ADDR_REQ* may be repeatedly lost $K_{\rm ad}$ times. The probability of this case $P_{\rm re2}$ is calculated by (22), shown at the bottom of the next page. Finally, the agent sends *ADDR_ALLOC*, but the requester cannot receive

$$P_{\rm ad}(i) = \begin{cases} \prod_{j=1}^{i-1}\left\{1-(1-loss_{\rm mul}(h_{\rm av}))^{(N-2)\cdot loss_{\rm mul}^{j-1}(h_{\rm av})}\right\}(1-loss_{\rm mul}\,(h_{\rm av}))^{(N-2)\cdot {loss_{\rm mul}}^{i-1}} & \text{for } 1\le i\le K_{\rm ad} \\ 1-\sum_{j=1}^{K_{\rm ad}} P_{\rm ad}(j) & \text{for } i=-1\text{(failure)} \end{cases} \tag{19}$$

$$l_{\rm ad}(i) = \begin{cases} (i-1)\cdot T_{\rm ad} + (t_{\rm un}+t_{\rm br})\cdot h_{\rm mx} & \text{for } 1\le i\le K_{\rm ad} \\ K_{\rm ad}\cdot T_{\rm ad} & \text{for } i=-1\text{(failure)} \end{cases} \tag{20}$$

ADDR_ALLOC due to loss. The probability of this case P_{re3} is calculated by (23), shown at the bottom of the page.

The probability that the requester may not receive *ADDR_ALLOC* is the sum of three cases, i.e.,

$$P_{\mathrm{re}} = P_{\mathrm{re1}} + P_{\mathrm{re2}} + P_{\mathrm{re3}}. \tag{24}$$

If address allocation fails, the requester tries address allocation, at most, K_{re} times. If address allocation succeeds when the requester sends the ith *REQUESTER_REQ*, the latency is the sum of $i-1$ timeouts; i times L_{ne}; two t_{br}'s, which are for *REQUESTER_REQ* and *ADDR_ALLOC*; and L_{ad}. If address allocation fails in spite of K_{re} retransmissions, the latency is the sum of K_{re} timeouts and K_{re} times L_{ne}. The latency to get an address from the allocator through an agent L_2 is given by

$$\begin{aligned} L_2 = \sum_{i=1}^{K_{\mathrm{re}}} \Big\{ & P_{\mathrm{re}}^{i-1} \cdot (1 - P_{\mathrm{re}}) \\ & \cdot ((i-1) \cdot T_{\mathrm{re}} + i \cdot L_{\mathrm{ne}} + 2 \cdot t_{\mathrm{br}} + L_{\mathrm{ad}}) \Big\} \\ & + P_{\mathrm{re}}^{K_{\mathrm{re}}} \cdot K_{\mathrm{re}} \cdot (T_{\mathrm{re}} + L_{\mathrm{ne}}). \end{aligned} \tag{25}$$

The expected number of packets sent to get an address from the allocator through an agent C_2 is similarly calculated as in L_2 and is given by

$$\begin{aligned} C_2 = \sum_{i=1}^{K_{\mathrm{re}}} \big\{ & P_{\mathrm{re}}^{i-1} \cdot (1 - P_{\mathrm{re}}) \cdot ((i-1) \cdot C_{\mathrm{fa}} + i \cdot C_{\mathrm{ne}} + 2 + C_{\mathrm{ad}}) \big\} \\ & + P_{\mathrm{re}}^{K_{\mathrm{re}}} \cdot K_{\mathrm{re}} \cdot (C_{\mathrm{fa}} + C_{\mathrm{ne}}) \end{aligned} \tag{26}$$

where C_{fa} is the expected number of packets sent when the requester requests an address but fails. So, we have

$$C_{\mathrm{fa}} = \frac{P_{\mathrm{re1}}}{P_{\mathrm{re}}} \cdot 1 + \frac{P_{\mathrm{re2}}}{P_{\mathrm{re}}} \cdot (1 + C_{\mathrm{ad}}) + \frac{P_{\mathrm{re3}}}{P_{\mathrm{re}}} \cdot (2 + C_{\mathrm{ad}}). \tag{27}$$

The probability of process P_2 to get an address when one of its neighbors is chosen as an agent is the probability of existence of at least one neighbor, so we have

$$P_2 = 1 - P_1. \tag{28}$$

3) Combining the Results: The overall latency for address allocation is calculated by

$$L_{\mathrm{total}} = P_1 \cdot L_1 + P_2 \cdot L_2. \tag{29}$$

The total number of packets sent from nodes for address allocation is calculated by

$$C_{\mathrm{total}} = P_1 \cdot C_1 + P_2 \cdot (C_2 + N) \tag{30}$$

where the last term N is for updating the address table.

D. Token-Based Scheme

The address allocation process of the token-based scheme is similar to that of MANETconf, except that the agent communicates with only the allocator, instead of receiving agreements from all the nodes. In this section, we derive L_{ad}, C_{ad}, P_{re1}, P_{re2}, and P_{re3} for the token-based scheme. Note that the other parts are the same as those in MANETconf.

As described in Fig. 3, the agent sends *ADDR_REQ* (*Msg*. 4) and receives *ADDR_REP* (*Msg*. 5) via unicast. If the agent receives *ADDR_REP* with i tries, the latency is the sum of $i-1$ timeouts and the round-trip time (RTT) from the agent to the allocator, and the number of packets sent consists of the expected number of hops that a packet has passed through and eventually lost, as well as the number of hops from the agent to the allocator. The probability of the number of repeated *ADDR_REQ* is calculated by a geometric distribution. L_{ad} and

$$c_{\mathrm{ad}}(i) = \begin{cases} i \cdot N + (N-2) \cdot h_{\mathrm{av}} + \sum_{j=1}^{i-1} \left\{ loss_{\mathrm{mul}}^{j}(h_{\mathrm{av}}) \cdot (N-2) \cdot h_{\mathrm{lo}}(h_{\mathrm{av}}) \right\} & \text{for } 1 \le i \le K_{\mathrm{ad}} \\ K_{\mathrm{ad}} \cdot N + (N-2) \cdot h_{\mathrm{av}} + \sum_{j=1}^{K_{\mathrm{ad}}} \left\{ loss_{\mathrm{mul}}^{j}(h_{\mathrm{av}}) \cdot (N-2) \cdot h_{\mathrm{lo}}(h_{\mathrm{av}}) \right\} & \\ \quad - loss_{\mathrm{mul}}^{K_{\mathrm{ad}}-1}(h_{\mathrm{av}}) \cdot (N-2) \cdot h_{\mathrm{av}} & \text{for } i = -1 \text{(failure)} \end{cases} \tag{21}$$

$$P_{\mathrm{re2}} = (1 - loss_{\mathrm{br}}) \cdot \prod_{j=1}^{K_{\mathrm{ad}}} \left\{ 1 - (1 - loss_{\mathrm{mul}}(h_{\mathrm{av}}))^{(N-2) \cdot {loss_{\mathrm{mul}}}^{j-1}(h_{\mathrm{av}})} \right\} \tag{22}$$

$$P_{\mathrm{re3}} = (1 - loss_{\mathrm{br}}) \cdot \left(1 - \prod_{j=1}^{K_{\mathrm{ad}}} \left\{ 1 - (1 - loss_{\mathrm{mul}}(h_{\mathrm{av}}))^{(N-2) \cdot loss_{\mathrm{mul}}^{j-1}(h_{\mathrm{av}})} \right\} \right) \cdot P_{\mathrm{br}} \tag{23}$$

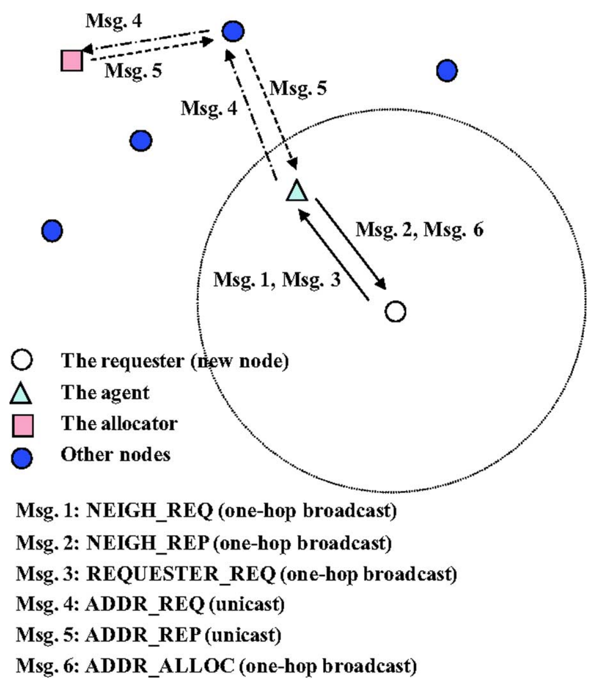

Msg. 4
Msg. 5
Msg. 4
Msg. 5
Msg. 2, Msg. 6
Msg. 1, Msg. 3
The requester (new node)
The agent
The allocator
Other nodes
Msg. 1: NEIGH_REQ (one-hop broadcast)
Msg. 2: NEIGH_REP (one-hop broadcast)
Msg. 3: REQUESTER_REQ (one-hop broadcast)
Msg. 4: ADDR_REQ (unicast)
Msg. 5: ADDR_REP (unicast)
Msg. 6: ADDR_ALLOC (one-hop broadcast)

Fig. 3. Address allocation process of a token-based scheme.

C_{ad} are calculated with the assumption of success of address allocation and are given by

$$L_{\text{ad}} = \sum_{i=1}^{K_{\text{ad}}} \left\{ \frac{loss_{\text{mul}}^{i-1}(2 \cdot h_{\text{av}}) \cdot (1 - loss_{\text{mul}}(2 \cdot h_{\text{av}}))}{1 - loss_{\text{mul}}^{K_{\text{ad}}}(2 \cdot h_{\text{av}})} \cdot ((i-1) \cdot T_{\text{ad}} + 2 \cdot h_{\text{av}} \cdot t_{\text{un}}) \right\} \quad (31)$$

$$C_{\text{ad}} = \sum_{i=1}^{K_{\text{ad}}} \left\{ \frac{loss_{\text{mul}}^{i-1}(2 \cdot h_{\text{av}}) \cdot (1 - loss_{\text{mul}}(2 \cdot h_{\text{av}}))}{1 - loss_{\text{mul}}^{K_{\text{ad}}}(2 \cdot h_{\text{av}})} \cdot ((i-1) \cdot h_{\text{lo}}(2 \cdot h_{\text{av}}) + 2 \cdot h_{\text{av}}) \right\}. \quad (32)$$

Similar to MANETconf, when a requester sends *REQUESTER_REQ*, it may not receive *ADDR_ALLOC* due to three types of losses: 1) *REQUESTER_REQ* loss; 2) either *ADDR_REQ* loss or *ADDR_REP* loss; and 3) *ADDR_ALLOC* loss. The probability of *REQUESTER_REQ* loss P_{re1} is simply $loss_{\text{br}}$. P_{re2} is the probability of either *ADDR_REQ* loss or *ADDR_REP* loss. The agent sends *ADDR_REQ* K_{ad} times in case it does not eventually receive *ADDR_REP*; therefore, we get

$$P_{\text{re2}} = (1 - loss_{\text{br}}) \cdot loss_{\text{mul}}^{K_{\text{ad}}}(2 \cdot h_{\text{av}}). \quad (33)$$

The probability of *ADDR_ALLOC* loss P_{re3} is calculated by

$$P_{\text{re3}} = (1 - loss_{\text{br}}) \cdot \left(1 - loss_{\text{mul}}^{K_{\text{ad}}}(2 \cdot h_{\text{av}})\right) \cdot loss_{\text{br}}. \quad (34)$$

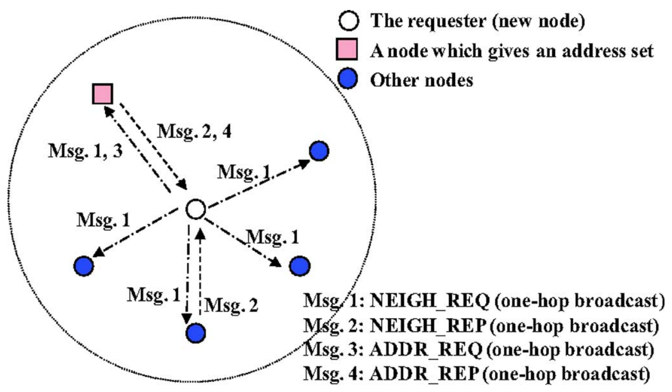

The requester (new node)
A node which gives an address set
Other nodes
Msg. 1, 3
Msg. 2, 4
Msg. 1
Msg. 1
Msg. 1
Msg. 1
Msg. 2
Msg. 1: NEIGH_REQ (one-hop broadcast)
Msg. 2: NEIGH_REP (one-hop broadcast)
Msg. 3: ADDR_REQ (one-hop broadcast)
Msg. 4: ADDR_REP (one-hop broadcast)

Fig. 4. Address allocation process of a neighbor-based scheme.

E. Neighbor-Based Schemes

From a performance perspective, neighbor-based schemes are much simpler than others since the address allocation is accomplished only based on local communication with neighbor nodes. In this section, we present a simple analysis for the address allocation latency and communication overhead of general neighbor-based schemes. In Fig. 4, we depict the address allocation process in neighbor-based schemes.

In the case that a requester is the first one in a network (self allocation), L_1, C_1, and P_1 are the same as those of MANETconf [refer to (10)–(12)]. When there is at least one neighbor, the requester sends *NEIGH_REQ* (*Msg.* 1 in Fig. 4), and neighbor nodes reply with *NEIGH_REP*. Then, the requester sends *ADDR_REQ* to the randomly selected node among its neighbor nodes. Upon receiving *ADDR_REQ*, the neighbor node generates (or selects) a new address using an address generation function and sends it with *ADDR_REP*. Since the latency and communication overhead for exchanging *NEIGH_REQ* and *NEIGH_REP* are the same as those of MANETconf, P_{ne}, n, L_{ne}, and C_{ne} are given by (13)–(16), respectively. The probability that a new node cannot receive *ADDR_REP* P_{ad} is calculated by

$$P_{\text{ad}} = loss_{\text{br}} + (1 - loss_{\text{br}}) \cdot loss_{\text{br}}. \quad (35)$$

Then, latency L_{ad} and communication overhead C_{ad} for exchanging *ADDR_REQ* and *ADDR_REP* can be expressed with a geometric distribution as follows:

$$L_{\text{ad}} = \sum_{i=1}^{K_{\text{ad}}} \left\{ \frac{P_{\text{ad}}^{i-1} \cdot (1 - P_{\text{ad}})}{1 - P_{\text{ad}}^{K_{\text{ad}}}} \cdot ((i-1) \cdot T_{\text{ad}} + 2 \cdot t_{\text{br}}) \right\} \quad (36)$$

$$C_{\text{ad}} = \sum_{i=1}^{K_{\text{ad}}} \left\{ \frac{P_{\text{ad}}^{i-1} \cdot (1 - P_{\text{ad}})}{1 - P_{\text{ad}}^{K_{\text{ad}}}} \cdot i \cdot 2) \right\}. \quad (37)$$

Finally, the overall latency and the total number of packets sent for address allocation in a neighbor-based scheme are calculated by

$$L_{\text{total}} = P_1 \cdot L_1 + (1 - P_1) \cdot (L_{\text{ne}} + L_{\text{ad}}) \quad (38)$$

$$C_{\text{total}} = P_1 \cdot C_1 + (1 - P_1) \cdot (C_{\text{ne}} + C_{\text{ad}}). \quad (39)$$

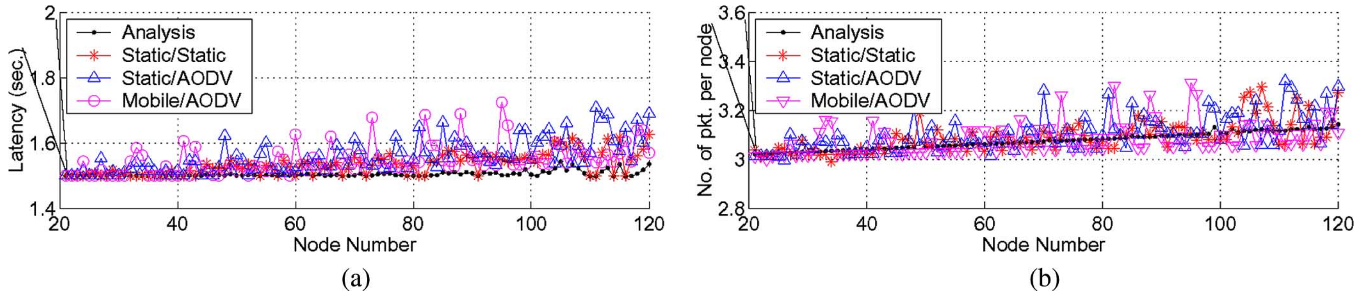


Fig. 5. Comparison of analytical and simulation results for IPAA. Here, $T_{\mathrm{ad}} = 0.5$ s, and $A = 1024$. (a) Latency. (b) Communication overhead per node.

F. Network Partition and Merging

A MANET can be repeatedly partitioned and merged due to the frequent in and out of mobile nodes. Most address allocation schemes deal with network partition and merging with a similar method. To detect network partition, each network has a unique network identifier, and it is periodically advertised. Usually, the first node in a network generates a network identifier and floods advertisement messages periodically. If some nodes fail to receive the advertisement message several times, they consider that the network has been partitioned. Network merging can also be detected by the network identifier advertisement. When two networks are merged, two network identifiers become advertised. The first node that receives two different network identifiers in the network floods a message to inform network merging and distribute a new network identifier.

The period of the advertisement messages determines the latency and communication overhead for dealing with network partition and merging. As we send the message more frequently, the latency for detecting partition or merging decreases with increased communication overhead. Basically, the latency for detection can be several times of the period, and the communication overhead is the same as the overhead for flooding. In [17], a worst-case analysis for communication overhead of network partition and merging has been presented.

IV. Model Validation and Performance Analysis

In this section, we analyze the efficiency of address allocation schemes. First, we validate our models with $ns-2$ [19] simulations. We also present numerical results for latency and communication overhead from analytical models.

A. Model Validation

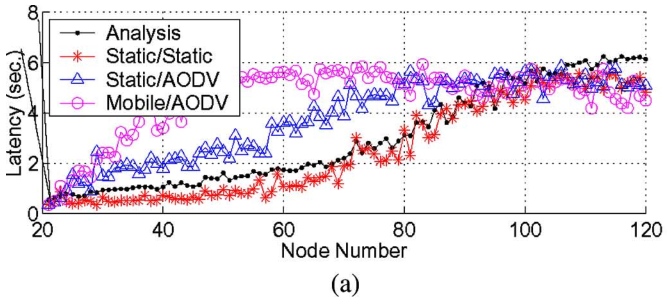
We provide $ns-2$ simulation results for address allocation schemes and compare them with the analytical results that were predicted by the modeling framework developed in this paper. In simulation, the distributed coordination function of the IEEE 802.11 for wireless local area networks [23] is used as the medium access control layer protocol. We use the standard values for Lucent's WaveLAN as the radio model, whose bandwidth is 2 Mb/s, and a nominal radio transmission range for each node is about 250 m.

To validate the proposed models, we perform simulations in three different routing and mobility environments: 1) static nodes with static routing; 2) static nodes with dynamic routing (AODV); and 3) mobile nodes with dynamic routing (AODV). For the neighbor-based scheme, we perform simulations in just two different mobility environments (static/mobile topology) since only local communication is involved in the address allocation process, and thus, it does not need multihop packet routing. For each environment, we conduct simulations using 30 scenarios with different seed numbers and average the results. In each run of simulation, nodes join a MANET sequentially, and for each joining of a new node, we measure the latency to get an address and the number of packets sent for that. We run simulations until 120 nodes have joined a network. To avoid network partitioning, 20 nodes are preconfigured in a 1000×1000 m square region.

To get analytical results, we measure parameters including h_{av}, h_{mx}, t_{br}, t_{un}, $loss_{\mathrm{br}}$, and $loss_{\mathrm{un}}$ from simulation with static nodes/static routing. t_{un} is calculated by $RTT/(2 \cdot h)$, where RTT is the average RTT between two end nodes. Address allocation latency and communication overhead are correlated with the values of the timer interval and the retry number. Selecting these values is difficult in a MANET since packet delay cannot be bounded. If we set higher values for timeout interval and retry number, the probability of address allocation failure decreases, but the latency and the communication overhead increase. We set the values of these constants with consideration of the average latency and the number of hops observed in simulations. For IPAA, MANETconf, and the token-based scheme, T_{ne}, T_{ad}, and T_{re} are set to 0.1, 0.5, and 2 s, respectively. However, for the neighbor-based scheme, T_{ad} is set to 0.1 s since address allocation is performed only via local communication with neighbor nodes. The maximum retry numbers are commonly set to three.

In Figs. 5–8, we present comparisons of analytical models and simulation results in terms of the expected latency and the communication overhead. In Fig. 5, the analytical results that were calculated with the derived model are quite accurate in estimating the performance of IPAA. We can observe that mobility and routing do not significantly impact the performance of IPAA since addresses are obtained within a short period.

In Fig. 6, it is observed that the performance of MANETconf is highly impacted by node mobility and dynamic routing since it requires replies from all the nodes in a network to allocate a new address. In Figs. 7 and 8, we can observe that the proposed models for the token-based and neighbor-based schemes are accurate, while mobility and dynamic routing slightly increase latency and communication overhead.



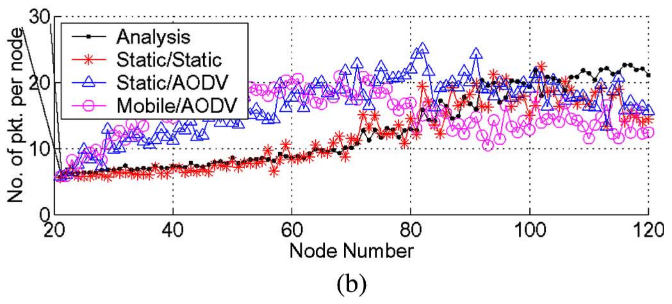
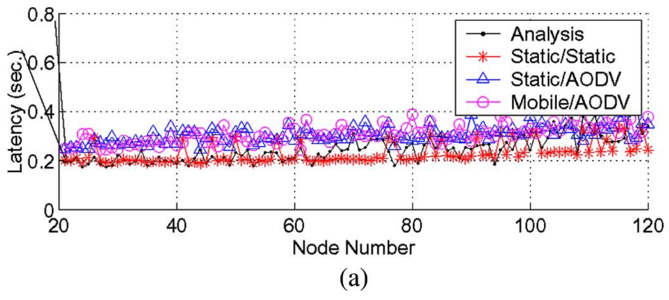


Fig. 6. Comparison of analytical and simulation results for MANETconf. Here, $T_{\mathrm{ne}} = 0.1$ s, $T_{\mathrm{ad}} = 0.5$ s, and $T_{\mathrm{re}} = 2$ s. (a) Latency. (b) Communication overhead per node.



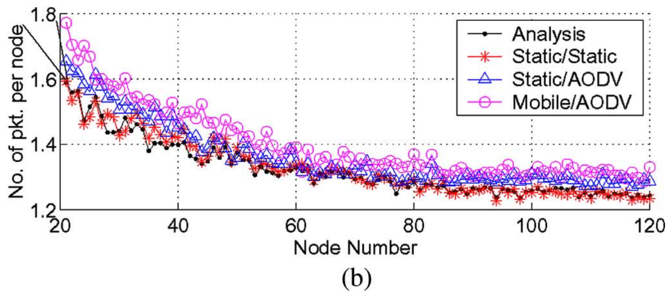
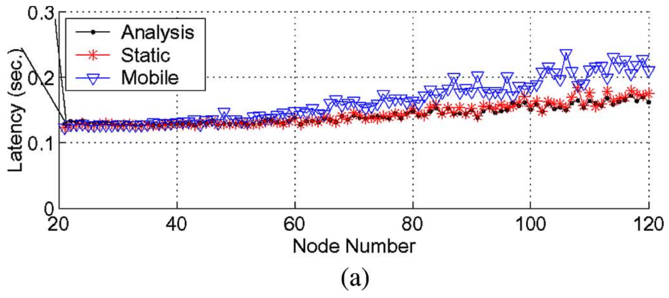


Fig. 7. Comparison of analytical and simulation results for the token-based scheme. Here, $T_{\mathrm{ne}} = 0.1$ s, $T_{\mathrm{ad}} = 0.5$ s, and $T_{\mathrm{re}} = 2$ s. (a) Latency. (b) Communication overhead per node.



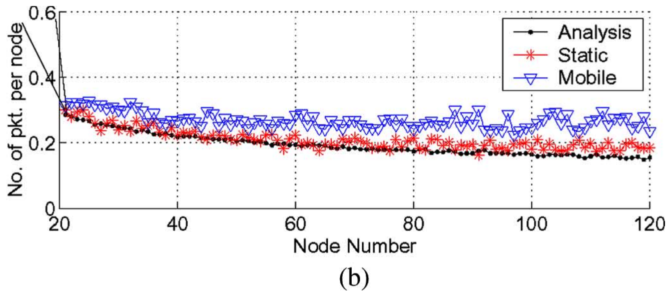


Fig. 8. Comparison of analytical and simulation results for a neighbor-based scheme. Here, $T_{\mathrm{ne}} = 0.1$ s, and $T_{\mathrm{ad}} = 0.1$ s. (a) Latency. (b) Communication overhead per node.

B. Numerical Results

In this section, we analyze the efficiency of four address allocation schemes with various parameter settings. In the rest of this section, we use the following parameter settings: K_{ne}, K_{ad}, and K_{re} are set to three. T_{ne}, T_{ad}, and T_{re} are set to 0.1, 0.5, and 2 s, respectively (T_{ad} is set to 0.1 s for the neighbor-based scheme). $loss_{\mathrm{br}}$ and $loss_{\mathrm{un}}$ are set to 0.01, unless otherwise stated. We set $t_{\mathrm{br}} = 30$ ms and $t_{\mathrm{un}} = 50$ ms. To set h_{av} and h_{mx}, we measured them in simulation, as presented in Fig. 9. It is observed that h_{av} is maintained to be around four, whereas h_{mx} increases as more nodes join the network. Since we employ shortest-path routing, h_{av} is mainly determined by the transmission range and topology size, rather than the number of nodes. h_{mx}, however, can be impacted by the number of nodes since a packet may not be delivered through the shortest path. Similar observations are presented in [24]. Based on this observation, we set h_{av} as four and vary h_{mx} from 6 to 14 as the number of nodes increases.

For setting R, we assume that the topology is a square. Then, when the topology size is much larger than the transmission range, R can be calculated approximately by $\pi \cdot r^2/d^2$, where r is the radius of the transmission range, and d is the side length of the square. The detailed derivation of R in case of the square region is presented in [10]. With $r = 250$ and $d = 1000$, we have $R = 0.156$ from [10].

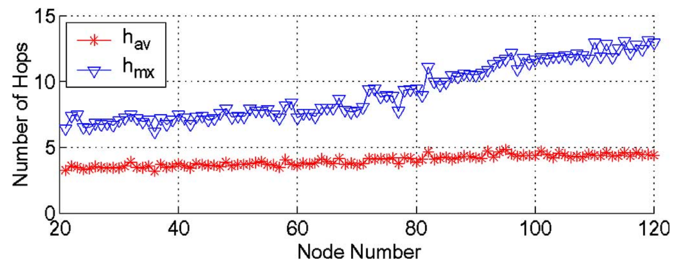


Fig. 9. Number of hops measured in simulations.

In Figs. 10 and 11, we show the expected latency and the communication overhead with various loss rates of unicast packets. Here, note that the results of the neighbor-based scheme are not presented here since it does not use unicast communication. In Figs. 10(a) and 11(a), the results of IPAA with a 512-address space are presented. It is shown that both latency and communication overhead moderately increase as

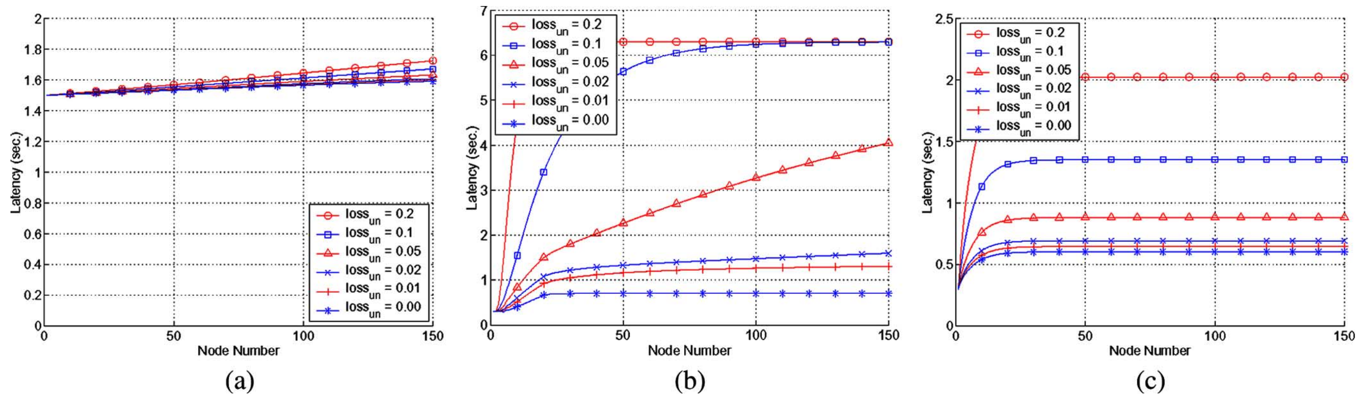


Fig. 10. Expected latency with various loss rates of unicast packet. (a) IPAA. (b) MANETconf. (c) Token-based scheme.

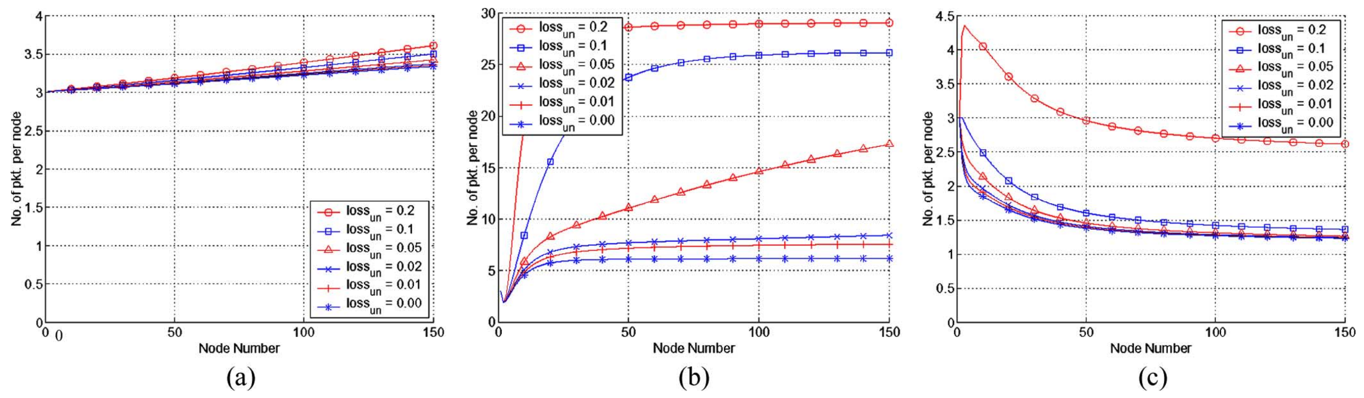


Fig. 11. Expected communication overhead with various loss rates of unicast packet. (a) IPAA. (b) MANETconf. (c) Token-based scheme.

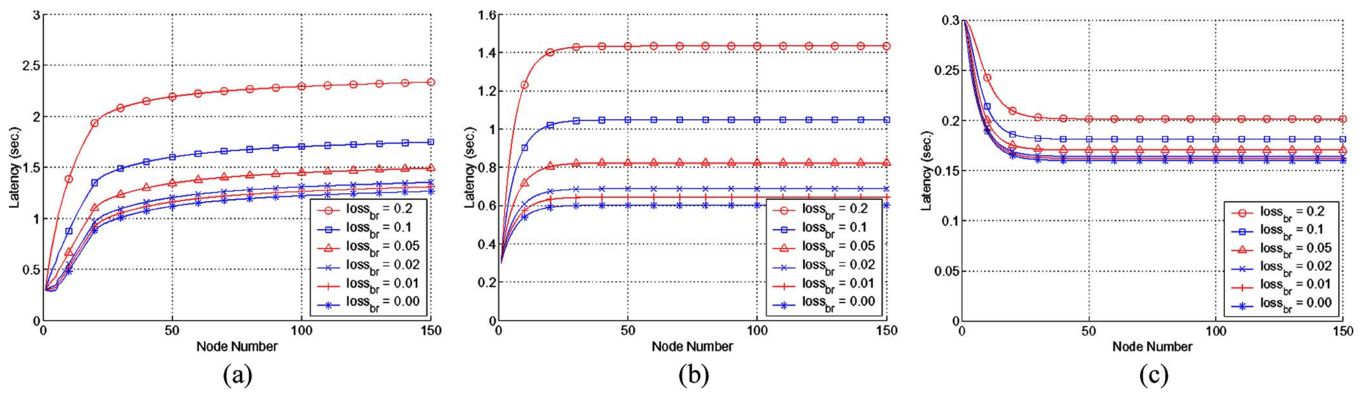


Fig. 12. Expected latency with various loss rates of broadcast packet. (a) MANETconf. (b) Token-based scheme. (c) Neighbor-based scheme.

the number of nodes in the network increases since, in the limited address space, address conflict is more likely to happen as more nodes join the network. It is also shown that the increase of loss rate slightly enlarges both latency and communication overhead.

In Figs. 10(b) and 11(b), we can observe that the efficiency of MANETconf depends more on loss rate rather than the number of nodes in the network. In MANETconf, to provide uniqueness of an assigned address, the global agreement is required, and packet loss is fatal for achieving the agreement from all the nodes in the network.

In Fig. 10(c), similar to MANETconf, the latency of the token-based scheme mainly depends on loss rate since packet loss increases the delay to reach the allocation (token holder). It is also observed that latency is maintained constantly with more that 25 nodes in the network since the token-based scheme works only between a new node and the allocator, regardless of the number of nodes existing in the network. The total number of packets sent in the token-based scheme is constant, regardless of the number of nodes in the network due to the same reason mentioned before, and thus, communication overhead per node is inversely proportional to the number of nodes, as shown in Fig. 11(c).

In Figs. 12 and 13, we present the efficiency with various loss rates of broadcast packets. Here, note that the results of IPAA is not present since, in IPAA, broadcast is only used for flooding, and some loss in flooding does not have much impact on the efficiency. In Figs. 12 and 13, the impact of broadcast packet losses looks similar to the impact of unicast packet losses, as shown in Figs. 10 and 11. However, the actual impact of

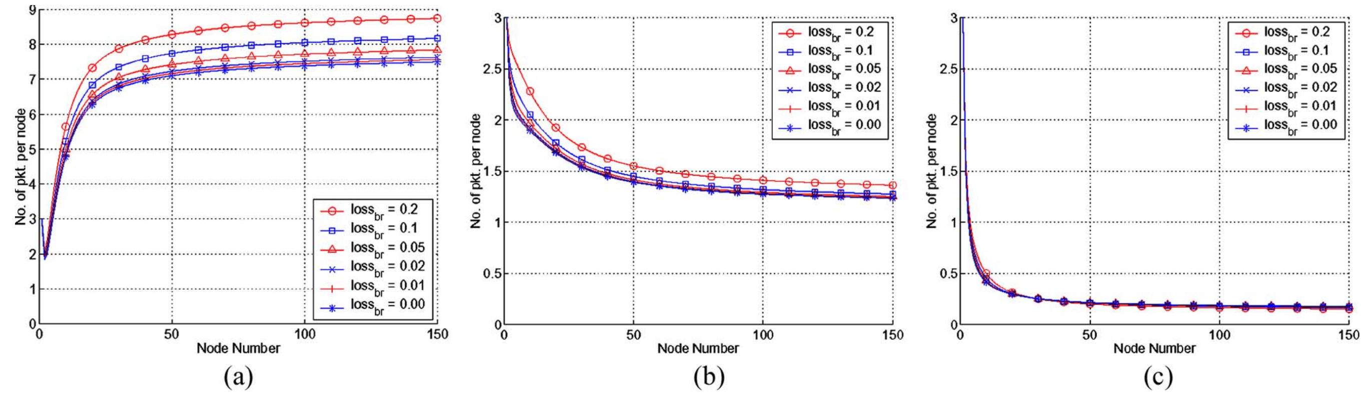


Fig. 13. Expected communication overhead with various loss rates of broadcast packet. (a) MANETconf. (b) Token-based scheme. (c) Neighbor-based scheme.

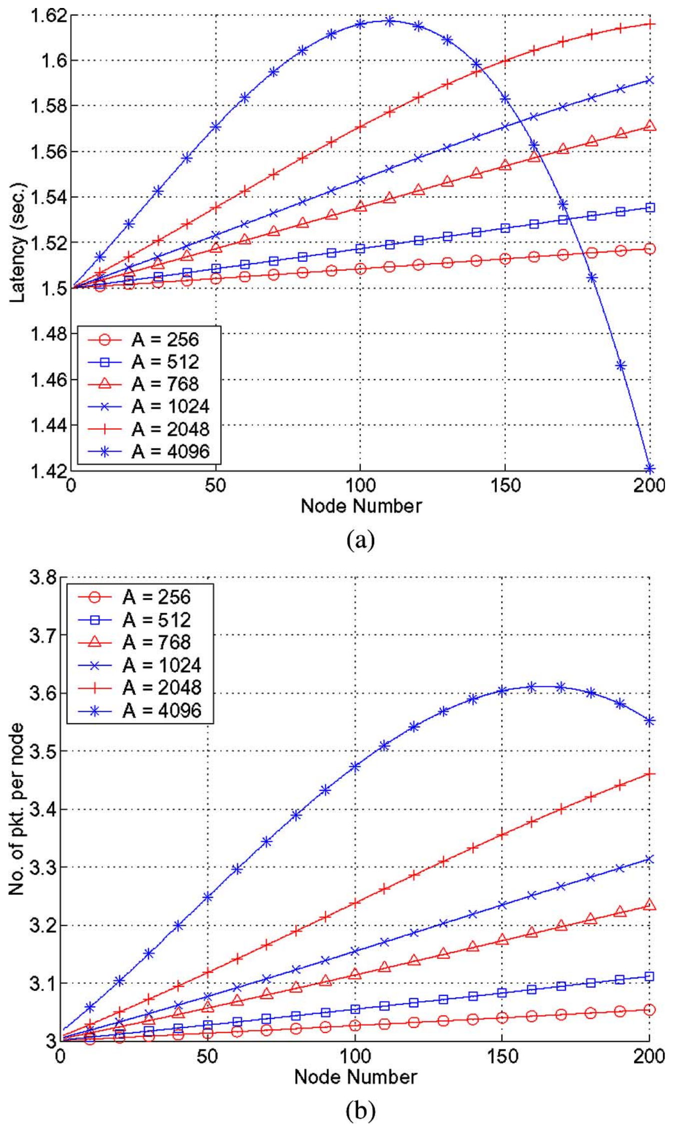


Fig. 14. Expected latency and communication overhead with various address space sizes in IPAA. (a) Latency. (b) Communication overhead per node.

broadcast packet losses is less than that of the loss of unicast packet losses since the number of broadcast packets generated for the address allocation is less than that of unicast packets. In the token-based scheme, particularly, it is observed that the loss rate of broadcast packets does not significantly impact communication overhead. This is because broadcasting is used only for communication between a new node and its agent, which can be anyone among its neighbors. In Figs. 12(c) and 13(c), we can observe that both the latency and communication overhead of the neighbor-based scheme are not impacted by variation of loss rate, and it shows that neighbor-based schemes might be suitable for MANETs observing high loss rate.

Finally, in Fig. 14, we observe the impact of address space size on the efficiency of IPAA. As more nodes join the network, latency and communication overhead increase due to address conflict. With a small size of address space ($A = 256$ in the figure), however, it is also observed that latency and overhead decrease as more than a certain number of nodes join the network. This is because all the attempts for an address quickly fail by address conflict before timeout when there are not enough free addresses remaining. Here, note that the impact of address space on other schemes is not presented since they are not very sensitive to the address space, as long as the number of addresses is enough to resolve the current nodes.

V. Conclusion

In this paper, we have derived a set of analytical models that quantitatively characterize the efficiency of four popular address allocation schemes in terms of latency and communication overhead. Through the analysis, we have achieved numerical results that show the impact of network parameters on the efficiency of these schemes. These results can be useful in knowing precise data to aid in selecting a protocol. The models have been validated through $ns-2$ simulations. To our understanding, this is the first attempt to perform mathematical analysis of addressing schemes in *ad hoc* networks. This study is important since it is a basis for the performance evaluation of new address allocation schemes that may appear in the future, without requiring resorting to simulations.

References

[1] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *Proc. SIGCOM Conf. Commun. Architecture, Protocols Appl.*, London, U.K., Aug. 1994, pp. 234–244.

[2] C. Perkins and E. Belding-Royer, "Ad-hoc on-demand distance vector routing," in *Proc. 2nd IEEE WMCSA*, New Orleans, LA, Feb. 1999, pp. 90–100.

[3] D. Johnson and D. Maltz, “Dynamic source routing in *ad hoc* wireless networks,” in *Mobile Computing*, T. Imielinski and H. F. Korth, Eds. Norwell, MA: Kluwer, 1996, ch. 5, pp. 153–181.

[4] C. Perkins, J. Malinen, R. Wakikawa, E. Royer, and Y. Sun, *IP Address Autoconfiguration for Ad Hoc Networks*, Nov. 2001. IETF Internet draft, draft-ietf-manet-autoconf-01.txt.

[5] S. Nesargi and R. Prakash, “MANETconf: Configuration of hosts in a mobile *ad hoc* network,” in *Proc. 21st Annu. Joint Conf. INFOCOM*, New York, Jun. 2002, pp. 206–216.

[6] J. Boleng, “Efficient network layer addressing for mobile *ad hoc* networks,” in *Proc. ICWN*, Las Vegas, NV, Jun. 2002, pp. 271–277.

[7] P. Patchipulusu, “Dynamic address allocation protocols for mobile *ad hoc* Networks,” M.S. thesis, Comput. Sci., Texas A&M Univ., College Station, TX, Aug. 2001.

[8] Y. Sun and E. Belding-Royer, “Dynamic address configuration in mobile *ad hoc* networks,” Comput. Sci., Univ. California, Santa Barbara, Santa Barbara, CA, Tech. Rep. 2003-11, Jun. 2003.

[9] S. Toner and D. O’Mahony, *Self-Organising Node Address Management in Ad-Hoc Networks*, vol. 2775. Englewood Cliffs, NJ: Springer-Verlag, 2003, pp. 476–483.

[10] S. Kim, J. Lee, and I. Yeom, “A token-based dynamic address allocation protocol for mobile *ad hoc* networks,” Dept. Comput. Sci., KAIST, Daejeon, Korea, 2005. Tech. Rep.

[11] A. Misra, “Autoconfiguration, registration, and mobility management for pervasive computing,” *IEEE Pers. Commun.*, vol. 8, no. 4, pp. 24–31, Aug. 2001.

[12] M. Mohsin and R. Prakash, “IP address assignment in a mobile *ad hoc* network,” in *Proc. IEEE MILCOM*, Anaheim, CA, Oct. 2002, pp. 856–861.

[13] Y.-Y. Hsu and C.-C. Tseng, “Prime DHCP: A prime numbering address allocation mechanism for MANETs,” *IEEE Commun. Lett.*, vol. 9, no. 8, pp. 712–714, Aug. 2005.

[14] H. Zhou, L. Ni, and M. Mutka, “Prophet address allocation for large scale MANETs,” in *Proc. 22nd Annu. Joint Conf. INFOCOM*, San Francisco, CA, Apr. 2003, pp. 1304–1311.

[15] N. Vaidya, “Weak duplicate address detection in mobile *ad hoc* networks,” in *Proc. ACM Int. Symp. MOBIHOC*, Lausanne, Switzerland, Jun. 2002, pp. 1059–1068.

[16] K. Weniger, “Passive duplicate address detection in mobile *ad hoc* networks,” in *Proc. IEEE WCNC*, Florence, Italy, Feb. 2003, pp. 271–277.

[17] Y. Sun and E. Belding-Royer, “A study of dynamic addressing techniques in mobile *ad hoc* networks,” *Wireless Commun. Mobile Comput.*, vol. 4, no. 3, pp. 315–329, Mar. 2004.

[18] C. Bernardos and M. Calderon, *Survey of IP Address Autoconfiguration Mechanisms for MANETs*, Jul. 2005. IETF Internet draft, draft-bernardos-manet-autoconf-survey-00.txt.

[19] *The Network Simulator—ns-2*. [Online]. Available: http://www.isi.edu/nsnam/ns/

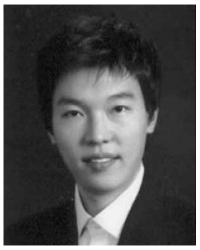
[20] B. Williams and T. Camp, “Comparison of broadcasting techniques for mobile *ad hoc* networks,” in *Proc. ACM Int. Symp. MOBIHOC*, Lausanne, Switzerland, Jun. 2002, pp. 194–205.

[21] Y. Zhao, B. Li, Q. Zhang, and W. Z. Yan Chen, “Efficient hop ID based routing for sparse *ad hoc* networks,” in *Proc. 13th ICNP*, Boston, MA, Nov. 2005, pp. 179–190.

[22] S. Das, C. Perkins, and E. Belding-Royer, “Performance comparison of two on-demand routing protocols for *ad hoc* networks,” in *Proc. 19th Annu. Joint Conf. INFOCOM*, Tel Aviv, Israel, Mar. 2000, pp. 3–12.

[23] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification,* IEEE Std. 802.11, 1997.

[24] K. Nahm, A. Helmy, and C.-C. J. Kuo, “Improving stability and performance of multihop 802.11 networks,” in *Proc. ACM Int. Symp. MOBIHOC (poster)*, Urbana-Champaign, IL, May 2005.



Sehoon Kim received the B.S. degree in computer science from Handong University, Pohang, Korea, in 2002 and the M.S. degree in computer science from Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 2004 where he is currently working toward the Ph.D. degree in computer science within the Department of Computer Science.

His research interests include mobile *ad hoc* networks and wireless networks.

Jinkyu Lee (S’05) received the B.S. and M.S. degrees in computer science in 2004 and 2006, respectively, from Korea Advanced Institute of Science and Technology, Daejeon, Korea, where he is currently working toward the Ph.D. degree in computer science within the Department of Computer Science.

His research interests include throughput enhancement for mobile *ad hoc* networks and vehicular *ad hoc* networks.

Ikjun Yeom (M’02) received the B.S. degree in electronic engineering from Yonsei University, Seoul, Korea, in 1995 and the M.S. and Ph.D. degrees in computer engineering from Texas A&M University, College Station, in 1998 and 2001, respectively.

Since January 2002, he has been an Assistant Professor with the Department of Computer Science, Korea Advanced Institute of Science and Technology, Daejeon, Korea. His research interests include home networks, wireless networks, and Internet QoS.