Prophet Address Allocation for Large Scale MANETs

Hongbo Zhou
Dept. of Computer Science & Engineering
Michigan State University
East Lansing, USA
zhouhon1@cse.msu.edu

Lionel M. Ni
Dept. of Computer Science
Hong Kong University of Science & Technology
Hong Kong, China
ni@cs.ust.hk

Matt W. Mutka
Dept. of Computer Science & Engineering
Michigan State University
East Lansing, USA
mutka@cse.msu.edu

Abstract—**A mobile device in a MANET must be assigned a free IP address before it may participate in unicast communication. This is a fundamental and difficult problem in the practical use of any MANET. Several solutions have been proposed. However, these approaches have different drawbacks. A new IP address allocation algorithm, namely prophet allocation, is proposed in the paper. The proposed scheme may be applied to large scale MANETs with low complexity, low communication overhead, even address distribution, and low latency. Both theoretical analysis and simulation experiments are conducted to demonstrate the superiority of the proposed algorithm over other known algorithms. Moreover, the proposed prophet allocation is able to solve the problem of network partition and merger efficiently.**

Keywords—MANET; autoconfiguration; address allocation

I. Introduction

Mobile ad-hoc networks (MANET) are growing in popularity due to the abundance of mobile devices, the speed and convenience of deployment, and the independence of network infrastructure. In such an IP-based network, IP address assignment to mobile devices is one of the most important network configuration parameters. A mobile device cannot participate in unicast communications until it is assigned a free IP address and the corresponding subnet mask.

If a MANET is connected to a hardwired network by a gateway, all the nodes in the MANET should have the same network address for simplicity of routing among them and the hardwired nodes. In other words, their addresses should be either private addresses in IPv4 or with the same special prefix in IPv6. Thus a mobile node may initiate communications with a hardwired node with the aid of NAT. As for communications initiated by the latter, mobile IP may be necessary, which is beyond the scope of this paper.

For small scale MANETs, it may be simple and efficient to allocate free IP addresses manually. However, the procedure becomes difficult and impractical for a large-scale open system where mobile nodes are free to join and leave. Much effort has been spent on routing protocols for MANET in recent years, such as OLSR [1], FSR [2], DSR [3], and AODV [4], while research on automatic configuration of IP addresses (autoconfiguration [5]) for MANET is relatively less. Although there is a Working Group in IETF called Zeroconf [6], it mainly focuses on the environments such as small or home office and embedded systems.

This research was supported in part by NSF Grants No. CCR-0098017, EIA-9911074, MSU IRGP Program and Microsoft Research Foundation.

Automatic address allocation is more difficult in a MANET environment than that in hardwired networks due to instability of mobile nodes, low bandwidth of wireless links, openness of MANET, and lack of central administration. Therefore, more overhead occurs to avoid address conflict compared to the protocols for hardwired networks, such as DHCP [7] and SAA [8]. However, since address allocation is the first step toward the practical application of the MANET, it is worth further research effort.

Before discussing address allocation issues, several scenarios are described to illustrate the difficulty of the problem. In the simplest scenario, a mobile node joins and then leaves a MANET once, such as nodes A and B illustrated in Fig. 1. An unused IP address is allocated on its arrival and becomes free on its departure.

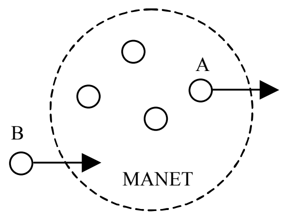

Figure 1. A node joins and leaves the MANET once

However, nodes are free to move arbitrarily during its session in the MANET. If one or more configured nodes go out of others' transmission range for a while, the network becomes partitioned as illustrated in Fig. 2 (a). When they approach each other, the partitions merge later. Because mobile nodes may not be aware of partitioning, they still use the previously allocated IP addresses. If a new node, say B, arrives at one partition and is assigned an IP address belonging to the other partition, say A's IP address, conflict happens when these two partitions merge as illustrated in Fig. 2 (b).

Another scenario is when two separately configured MANETs merge, which is illustrated in Fig. 3. Because address allocation in one MANET is independent of the other, there may be some duplicate addresses in both of them. For example, node A in MANET 1 has the same IP address as node B in MANET 2. As a result, some (or all) nodes in one MANET may need to change their addresses.

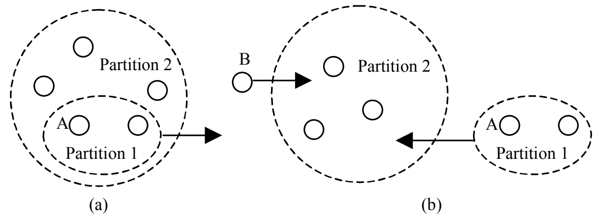

Figure 2. Network partitions and merges

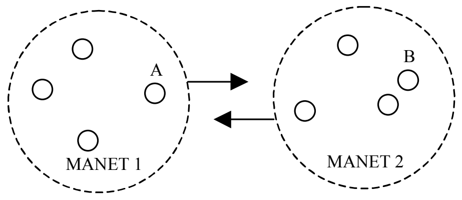

Figure 3. Merger of two indpendent MANETS

In another scenario, students are free to switch between a series of seminar rooms held at the same time. A mobile node leaves one MANET and then joins another MANET. This node could be regarded as the special case of the situation mentioned above because the single node could be viewed as a one-node partition.

The last scenario is fairly rare. Suppose there are two independent MANETs that are close to each other. A node in between decides to join a MANET nearby and functions as a relay node, which leads to connection of the two MANETs. This is the same as merger of two independent MANETs.

In summary, a feasible autoconfiguration algorithm should handle the following three general scenarios:

Scenario A: A mobile node simply joins a MANET and then leaves it forever;

Scenario B: A MANET partitions and then the partitions merge later;

Scenario C: Two separately configured MANETs merge.

The paper is structured as follows. Related research efforts are introduced in Section 2. A new IP address allocation algorithm, namely prophet allocation, is proposed in Section 3, which is based on sequence generation. With a little more effort, it is able to solve the problem of network partition and merger efficiently. Section 4 defines metrics for performance evaluation first and then applies them to all four address allocation schemes. According to these evaluation metrics, conflict-detection, conflict-free, and best-effort allocation have different drawbacks, while prophet allocation achieves low complexity, low communication overhead, even distribution, low latency, and high scalability, which is verified by the simulation results presented in Section 5. Section 6 concludes the paper.

II. Related work

Several solutions have been suggested and studied by other researchers, which can be divided into the following three categories.

A. *Conflict-detection allocation*

The conflict-detection allocation adopts a "trial and error" policy to find a free IP address for a new mobile node in the MANET. The new node chooses an IP address tentatively, and requests for approval from all the configured nodes in the MANET. If the conflict is found by veto from a node with the same IP address, the procedure is repeated until there is no duplicate address. At that time the node uses the latest chosen IP address as its "permanent" address. One of the conflict-detection allocation algorithms is the protocol proposed in [9]. Another is IPv6 autoconfiguration for MANET proposed in [10].

The procedure above is defined as strong DAD (Duplicate Address Detection) in [11], which is able to handle scenario *A* easily, without any solution for Scenarios *B* and *C*. The so-called weak DAD is proposed in [11], which aims to handle network merger. It favors proactive routing protocols and requires little modification to routing protocols.

B. *Conflict-free allocation*

The conflict-free allocation assigns an unused IP address to a new node, which could be achieved by the assumption that the nodes taking part in allocation have disjoint address pools. Thus they could be sure that the allocated addresses are different. Dynamic Configuration and Distribution Protocol (DCDP) [12] is a conflict-free allocation algorithm, which was originally proposed for autoconfiguaration in hardwired networks. Every time when a new mobile node joins, an address pool is divided into halves between it and a configured node.

One advantage of conflict-free allocation is that it still works in Scenario *B*. Even if the network becomes partitioned, the nodes in different partitions still have different address pools. Thus the addresses allocated are different as well. When the partitions become connected, no further work is necessary. As to Scenario *C*, it is very likely that there are conflicts if the configuration of two MANETs begins with the same reserved address range.

C. *Best-effort allocation*

In this approach, the nodes responsible for allocation try to assign an unused IP address to a new node as far as they know. At the same time the new node uses conflict detection to guarantee that it is a free IP address.

An example of best-effort allocation is Distributed Dynamic Host Configuration Protocol (DDHCP) proposed in [13]. DDHCP maintains a global allocation state, which means all mobile nodes are tracked, so it is known which IP addresses have been used and which addresses are still free. When a new node joins the MANET, one of its neighbors could choose a free address for it. The reason why it still bothers to detect conflict is that the same free IP address in the global address pool could be assigned to two or more new nodes arriving at almost the same time.

One advantage of DDHCP is that it works well with proactive routing protocols, since every mobile node broadcasts periodically. Another advantage is that it takes into account network partition and merger. A partition ID is generated by the node with the lowest IP address and broadcast throughout the partition periodically. Thus, the partition and merger may be detected by partition ID (with the aid of periodic exchange of HELLO messages). When partitions become connected, conflict detection and resolution is initiated.

III. Prophet allocation

IP address autoconfiguration is the same as assignment of different numbers from an integer range, say R, to different nodes. Conflict-detection allocation and best-effort allocation use random guesses and then make sure there is no duplicate by means of broadcast of conflict detection. Conflict-free allocation partitions R into several disjoint subsets R_1, R_2, …, R_m and chooses a random subset to divide between different nodes.

The idea included in these algorithms is that every mobile node obtains an unused IP address randomly on its own. Unless a node announces its IP address throughout the MANET, it cannot be known to others that this IP address is occupied. What if all the IP addresses that have been allocated and are going to be allocated are known to every participating node in advance? Broadcast could be avoided while conflict is still detectable.

A. Prophet allocation

Suppose we may obtain an integer sequence consisting of numbers in R by a function, say $f(n)$, which is stateful. The initial state of $f(n)$ is called the *seed*. Different seeds lead to different sequences with the state of $f(n)$ updated at the same time. The sequences of $f(n)$ satisfy the following two properties (if R is large enough):

(1) The interval between two occurrences of the same number in a sequence is extremely long;

(2) The probability of more than one occurrence of the same number in a limited number of different sequences initiated by different seeds during some interval is extremely low.

Thus we could derive an IP address autoconfiguration algorithm from the aforementioned sequence generation:

(1) The first node, say A, chooses a random number as its IP address and uses a random state value or a default state value as the seed for its $f(n)$;

(2) When another node, say B, approaches A and asks A for a free IP address, A uses $f(n)$ to obtain another integer, say n_2, and a state value, and provides them to B. Node A updates its state accordingly;

(3) Node B uses n_2 generated by A as its IP address and the state value obtained from node A as the seed for its $f(n)$;

(4) Now node A and node B are both able to assign IP addresses to other nodes.

The communication between node A and node B may be accomplished by means of one-hop broadcast since B does not have an IP address yet. However, it still saves much communication overhead compared with multi-hop broadcast needed in conflict detection.

The algorithm is illustrated as an example in Fig. 4. Suppose every node is represented by a 2-tuple: (*address*, *state of f(n)*). Here R is [1,8], $f(n)$ is ($address \times state \times 11$) mod 7 and the effective address range is [1,6]. In Fig. 4, A is the first node in the MANET and uses a random number of 3 as its IP address and seed. When node B joins, node A gets 1 (=(3×3×11) mod 7). Node A changes its state of *f(n)* to 1 and assigns 1 to B. When C approaches A and D approaches B, they receive 5 (=(3×1×11) mod 7) and 4 (=(1×1×11) mod 7) from A and B, respectively. In the third round of allocation, a conflict will happen. Note that 4 out of 6 addresses are allocated without conflict in the first 2 rounds of allocation, and the allocation later leads to a conflict. The reason of conflict is due to a small range of R.

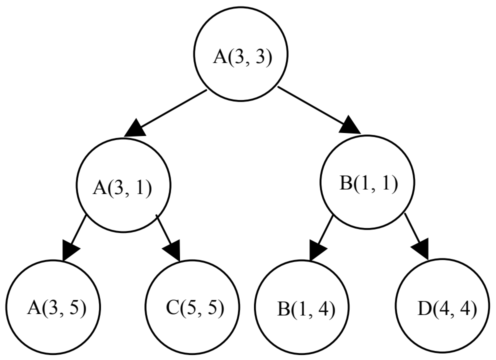

Figure 4. An example of prophet allocation

In the beginning of allocation, node A chooses the seed for the whole MANET and the sequences may be computed locally. Therefore, node A is a prophet in the MANET, which means it knows in advance which addresses are going to be allocated. Thus, we call this algorithm *prophet allocation*.

Because the potential conflict in the allocation may be known at node A in the beginning, it is able to launch local conflict detection before allocation. If there are many duplicate numbers in the sequences, node A could choose another seed to generate other sequences until there are fewer conflicts. Those duplicate numbers could be marked in the beginning of allocation.

Address reclamation is unnecessary for prophet allocation because the same number will reoccur in the sequence. Nevertheless, the minimal interval between two occurrences in the sequences is extremely long. When a node is assigned an old address, say *n*, the previous node with the same address of *n* has likely already left the MANET.

B. *Mechanism for network partition and merge*

Prophet allocation is able to solve the problem of network partition and merger of a MANET easily. As for Scenario *B*, because the sequences are different even if the MANET becomes partitioned, the newly allocated addresses are still different among the partitions. Therefore, there is no conflict if the partitions become merged later.

With regard to Scenario *C*, we borrow the idea of partition ID in DDHCP with a little modification. Here we designate the first node in the MANET to generate the network ID (NID) using a random number, which is propagated to new nodes during the course of allocation. Because NID is a random number, if the number of bits for NID is large enough, two MANETs will have different NIDs. Since some reactive routing protocols (e.g., AODV) require periodic exchange of HELLO messages between neighboring nodes, if NID is piggybacked in HELLO messages, the merger of two separate MANETs may be easily detected.

There are two methods to handle Scenario *C*. The first one is that the seed for the MANET is carried in the HELLO messages as well. The node that detects merger is able to find potential address conflicts between two MANETs locally by applying *f*(*n*) on the two seed values for MANETs and initiates conflict resolution if necessary. The possibly conflicting addresses are contained in the message that is broadcast to every node. If one's IP address is contained in the list, it changes its address accordingly. A new NID is then generated by the detecting node and then broadcast throughout the new larger MANET. If several nodes detect the merger at the same time, they could initiate conflict resolution independently or random delay is included to save repeated work. The largest NID generated among the detection nodes is chosen as the new NID.

However, the method above requires much computation and communication overhead. Another simpler method is that when mobile nodes detect the merger of two independent MANETs, the nodes in one MANET, say MANET 1 (for example, MANET 1 has a smaller NID), choose to discard their current IP addresses and acquire new addresses and NID from their neighbors in the other MANET (say MANET 2), which propagates from the intersection of the two MANETs to all the other nodes in MANET 1. Thus the overhead of local conflict detection and conflict resolution mentioned above is saved at the cost of breaking on-going connections in MANET 1. This is especially suitable for the situation of a merger of a MANET with a one-node partition, which will be aware that it has no neighbors with the same NID and will decide to change its IP address.

C. *Design of f(n)*

The stateful function *f(n)* should be carefully designed. In the example in Fig. 4, we used primes to scatter the numbers in the sequence. In a real design, *f(n)* is closely related to address range as well. For IPv4, class C private addresses of 192.168.0/24 are not large enough for dozens of mobile nodes in the MANET because of the high probability of collision. Class A private addresses of 10/8 and Class B private addresses of 172.16/12 will be suitable. As to IPv6, there is no need for such a concern because of its huge address range.

It is difficult to find such an *f*(*n*) that exactly satisfies the two properties mentioned before. However, an *f*(*n*) that approximately satisfies the properties is easy to design. One such *f*(n) we suggest is based on the fundamental theory in arithmetic: every positive integer may be expressed uniquely as a product of primes, apart from the rearrangement of terms. The canonical form of a positive number *n* is

$$n = \prod_{i=1}^{k} p_i^{e_i} \text{, where the primes } p_i \text{ satisfy } p_1 < p_2 < \ldots < p_k$$

and the exponents are non-negative integers. Apparently, if *k*-tuples (e_1, e_2, ..., e_k) have different e_i ($i = 1, \ldots, k$), there will be different *n*. Our idea is to generate different *k*-tuples.

Suppose $k = 4$. The first node obtains a random address of *a* and an initial state of (0, 0, 0, 0). Fig. 5 shows the procedure of generating new states and updating old states. A node is represented by (*address*, (e_1, e_2, e_3, e_4)), with $address = (a + 2^{e1}3^{e2}5^{e3}7^{e4}) \bmod range + 1$ (with the exception of the first node). The rules of state generation and update during the allocation are: (1) the underlined element in the 4-tuple of a configured node increases by 1; (2) the state of a new node is copied from the allocator, but the underline shifts right by 1.

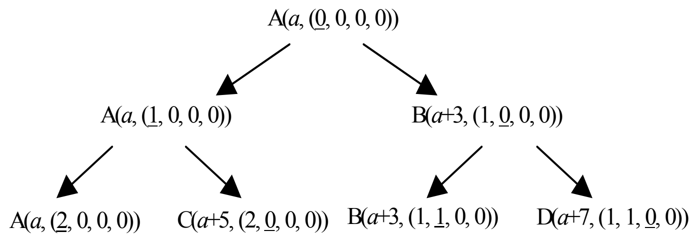
A(a, ($\underline{0}$, 0, 0, 0))
A(a, ($\underline{1}$, 0, 0, 0))
B(a+3, (1, $\underline{0}$, 0, 0))
A(a, ($\underline{2}$, 0, 0, 0))
C(a+5, (2, $\underline{0}$, 0, 0))
B(a+3, (1, $\underline{1}$, 0, 0))
D(a+7, (1, 1, $\underline{0}$, 0))

Figure 5. Generation and update of states in *f*(n)

k may be much larger in real applications. Thus, our algorithm requires an array of primes and exponents only and nothing else. However, the array of exponents need not be stored in nodes or carried in the messages between neighboring nodes during allocation. Our simulation also shows that optimization is achievable in the computation of addresses.

There will be infinite different numbers generated by *f*(*n*) in theory. However, given a small range of addresses, there might be duplicate numbers. The possibility of duplicate addresses is negligible for a small number of nodes or using class A private addresses.

D. Protocol

Fig. 6 depicts the state transitions of a mobile node during its session in the MANET.

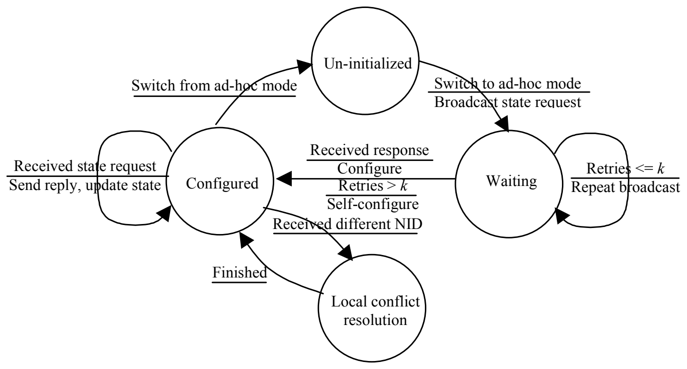

Figure 6. The finite state machine for prophet allocation

The protocol is as follows:

1) When a mobile node switches to ad-hoc mode, it begins periodic broadcast of state request packets, and changes from UN-INITIALIZED state to WAITING state. Note that only one-hop broadcast is necessary. Its MAC address may also be carried in the request packets, which is used by the responder to build a unicast reply;

2) The mobile node stays in the WAITING state and repeats state request for less than or equal to *k* times;

3) If the mobile node receives a reply during that time, it configures itself with the IP address, intial state value, and NID contained in the reply, and changes to CONFIGURED state;

4) Otherwise, it chooses itself an IP address and NID randomly and a default state value as its initial state value and changes to CONFIGURED state;

5) During CONFIGURED state, the mobile node repeats broadcasting HELLO messages, sends back replies on receipt of state request packets from other nodes, and updates its own state accordingly;

6) If the mobile node receives a HELLO message with a different NID, it switches to LOCAL CONFLICT RESOLUTION state, which is described in the second subsection. After completion of local conflict resolution, it returns to CONFIGURED state;

7) When the mobile node ends its session in the MANET, it switches out of ad-hoc mode and changes to UN-INITIALIZED state.

IV. Performance Analysis

In this section, evaluation metrics for allocation performance are first defined. Then theoretical analysis of all four kinds of solutions is presented.

A. Metrics for performance evaluation

1) Distributed operation: A specific node in a MANET cannot be trusted as a configuration server as the one in DHCP because of its mobility, limited transmission range, and power supply. Failure of any number of nodes should not prevent autoconfiguration from working. Therefore, the algorithm must be distributed.

2) Correctness: All three scenarios discussed in Section 1 need to be considered. No two or more nodes with the same address could coexist for a long time. Conflict resolution should be initiated as quickly as possible if necessary.

3) Complexity: Taking into account limited computation power and memory capacity of mobile nodes, the solution should be as simple as possible. The solution may consist of several modules: allocation, conflict detection, state maintenance, etc. The complexity of each module should be carefully considered.

4) Communication overhead: Does the solution require broadcast in a MANET? Or does the solution only incurs communication between neighboring nodes? Broadcast is extremely bandwidth-consuming, which should be avoided as much as possible. Periodic broadcast is surely unacceptable.

5) Evenness: If the allocated addresses of most mobile nodes are clustered in a subset of the whole address range, the address distribution is uneven, which also means the probability of conflict is high. Thus, conflict detection may be launched several times and will lead to high communication overhead. Otherwise, if the distribution is even, the probability of conflict is low, which results in low communication overhead.

6) Latency: The time between the point when a node initiates autoconfiguration and the one when it is assigned a free IP address is referred as latency. The shorter the latency, the better. Broadcast leads to longer latency, while local communication results in shorter latency.

7) Scalability: The bandwidth consumed by broadcast is positively related to the number of nodes in the MANET. The latency is proportional to the diameter of the network, which is also positively related to the number of nodes. Therefore, if multi-hop broadcast is required in autoconfiguration, it has poor scalability. If most of communications happen locally, it has excellent scalability.

All of these metrics are closely related. The more even the distribution and the lower communication overhead, the shorter the latency and the better scalability. In other words, evenness and communication overhead are more important than the other metrics.

B. Performance comparison

Table 1 presents a comparison of the aforementioned methods. The first four rows are a characteristics summary of the four allocation algorithms. The last five rows focus on the qualitative evaluation of their performance.

Conflict-detection allocation is the simplest method. No state is maintained. No address reclamation is needed. However, broadcast adopted in conflict detection leads to high communication overhead, high latency, and small scalability.

For example, suppose the number of mobile nodes is n, the number of links is l, the average transmission time between two adjacent nodes is t, the network diameter is d (in terms of nodes), and the retry time is k. If there is no address conflict, the number of packets needed in conflict detection is at least $(n+l)\times k$, and the time spent is $2\times t\times d\times k$. Otherwise, the communication overhead will be more and the latency will be longer. The distribution of addresses is even because it uses random guess. Therefore, the probability of conflict is rare with a large address range and small number of mobile nodes.

TABLE I. CHARACTERISTICS AND PERFORMANCE COMPARISON

	Conflict detection	**Conflict free**	**Best effort**	**Prophet**
Network organization	Flat / Hierachical	Flat	Flat / Hierachical	Flat
State maintanence	Stateless	Partially stateful	Stateful	Stateful
Address conflict	Yes	No	Yes	No
Address reclamation	Unneeded	Needed	Needed	Unneeded
Complexity	Low	High	High	Low
Communication overhead	$O((n+l)\times k)$	$O(2l/n)$	$O((n+l)\times k)$	$O(2l/n)$
Evenness of distribution	Even	Possibly uneven	Even	Even
Latency	$O(2\times t\times d\times k)$	$O(2t)$	$O(2\times t\times d\times k)$	$O(2t)$
Scalability	Small	Medium / Small	Small	High

Conflict-free allocation is simple in address assignment itself. However, a difficult problem arises in the management of the address pool. If a mobile node notifies others before it leaves or shuts down gracefully, it could release its IP address and address pool. However, if it leaves the MANET silently or shuts down abruptly, it will take away its IP address and address pool from the whole address range, which cannot be used by others. Thus, a mechanism for address reclamation is necessary, which is far more difficult and complicated than allocation. As other performance metrics, because most communication happens between neighboring nodes, it has low communication overhead, low latency, and medium scalability. For example, the packets needed are one-hop broadcast messages, which is proportional to the average number of degrees, i.e., $2l/n$. The latency is proportional to the round-trip time between two adjacent nodes, i.e., $2t$. However, the distribution of addresses depends on the allocation pattern, which is also important for determining its scalability. For example, if new nodes keep requesting the same configured node for address pools, the size of the address pool will decrease exponentially. Thus the scalability worsens. This could be remedied by balancing the address pools among the configured nodes, which makes the management of address pools more difficult.

The performance of best-effort allocation is expected to be almost the same as that of conflict-detection allocation: high communication overhead, even distribution, high latency, and low scalability. However, because global state is maintained, the complexity is higher due to overhead incurred by state management and synchronization.

From the analysis above, we can arrive at the conclusion that the allocation algorithm must satisfy the following properties to achieve low latency and high scalability:

(1) Local communication (which means low communication overhead);

(2) Random assignment (which leads to even distribution).

In prophet allocation, when a new node joins the MANET, it just asks for one of its configured neighbors for its IP address and initial state. Thus, the first property is satisfied. With a carefully designed $f(n)$, the numbers in sequences may be distributed evenly in the integer range, and hence the second property may be satisfied. Thus the performance in communication overhead and latency of prophet allocation is expected to be almost the same as that of conflict-free allocation, while the complexity of the former is much lower than that of the latter, and the distribution of the former is even. As a result, the prophet allocation is suitable for large scale MANETs.

V. SIMULATION

According to our analysis in the last section, the performance of best-effort allocation is similar to that of conflict-detection allocation. Simulation of the former has been done in [13]. Therefore, we chose to implement the conflict-detection allocation proposed in [9] together with prophet allocation to compare their performance.

The simulation was done on ns-2 (version 2.1b8a) with CMU extension for ad hoc networks [14]. Statistics about communication overhead and latency in Scenario *A* and Scenario *B* were collected to show that prophet allocation outperforms conflict-detection allocation and best-effort allocation for large scale MANETs.

A. *Simulation parameters*

The random waypoint mobility model was adopted in the simulation [15]. After a node pauses for several seconds, a random destination point is chosen. Then the node moves towards that point at a maximum speed of 5 m/s, which is repeated until the end of simulation. The pause time is 10 seconds for 50 and 80 nodes, and 20 seconds for 100, 120 and 150 nodes, respectively. Different area sizes are also introduced to demonstrate the effect of density of nodes on the performance. For example, scenario files of 800×800, 1000×1000, and 1200×1200 were simulated for 100, 120 and 150 nodes. The final results are the average of the results obtained with all the area sizes.

During the simulation, mobile nodes join the MANET every 30 seconds in the order of node ID. Because we aim to investigate the performance of large scale MANETs, no node departure is introduced in the simulation. Another reason is that the number of nodes has no effect on the correctness of the algorithms.

We used DSR as the ad hoc routing protocol during the simulation. Both conflict-detection allocation and prophet allocation have no assumptions on the underlying routing protocols, because multi-hop broadcast and one-hop broadcast were implemented without the aid of routing protocols.

B. Simulation verification

To verify correctness of the implementation of allocation simulation, we first ran the simulation for 3, 4 and 5 nodes separately. The area size was chosen to make all the nodes connected in the topology. The simulation results are equal to our analysis, which shows that multi-hop broadcast and one-hop broadcast were correctly implemented in conflict-detection allocation (CDA for short in the diagrams) and prophet allocation (PA for short in the diagrams), respectively. The number of received packets at each node for 3-node simulation is illustrated in Fig. 7.

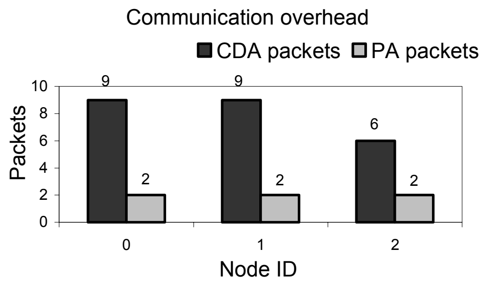

Figure 7. Received packets at each node for 3-node simulation

C. Communication overhead

Because every successfully received packet, either unicast packet or broadcast packet, must have consumed bandwidth (and power as well), we use it as the evaluation metric for communication overhead.

Fig. 8 shows the total number of packets received in 50-node simulation with different area sizes. The number of packets generated in conflict-detection allocation is 51.71 times of that in prophet allocation on average. As the density of nodes decreases, the communication overhead of conflict-detection allocation decreases because the link number decreases, and the network becomes partitioned during the simulation. The communication overhead of prophet allocation decreases because the neighboring nodes become fewer.

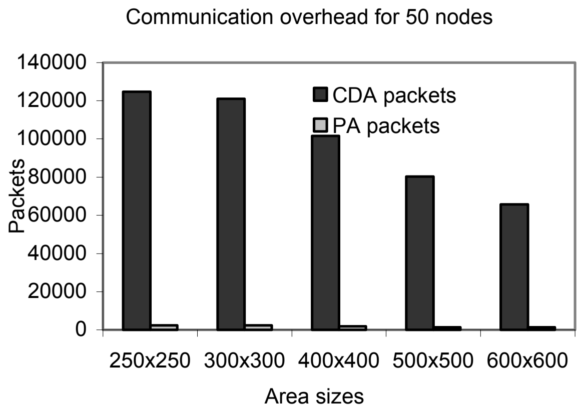

Figure 8. Communication overhead for 50 nodes

Fig. 9 shows the ratio of packets in conflict-detection allocation to those in prophet allocation for different number of nodes, together with a linear line. According to the diagram, the ratio of communication overhead in conflict-detection allocation to prophet allocation is approximately proportional to the number of nodes in the MANET, which means the more nodes, the more gain in communication overhead in prophet allocation.

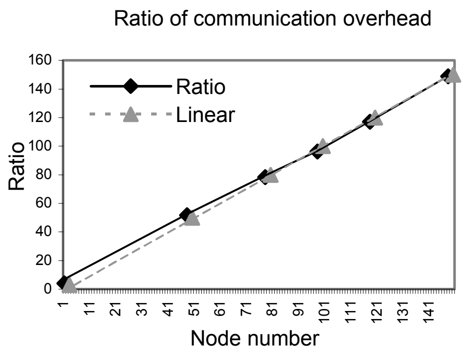

Figure 9. Ratio of communication overhead of CDA to PA

D. Latency

During the simulation, the nodes participating in the conflict-detection allocation tried a maximum of 3 times for broadcast of duplicate address detection packets. While in prophet allocation, except for the first node, every node tried infinitely to broadcast state request packets until it received a state reply from its configured neighbor. The intervals for both are set to be the same[1], so we need only to compare their retry times.

Fig. 10 shows the average retry times in a 50-node simulation within different sizes of areas. Most nodes receive their responses during the first round of state request. As the node density decreases, the retry time increases.

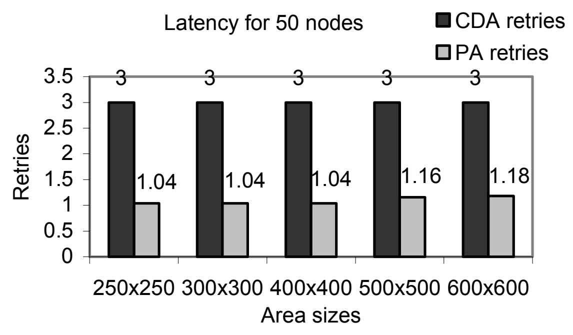

Figure 10. Latency for 50 nodes

Fig. 11 shows the relationship of retry times and the node number. According to the diagram, the average retry time for prophet allocation fluctuates around 1.5 regardless of how many mobile nodes are in the MANET. Taken into account that the round-trip time between neighboring nodes is independent of network size, the latency for large scale MANETs is nearly the same as small scale MANETs, while the latency in conflict-detection allocation increases for large scale MANETs.

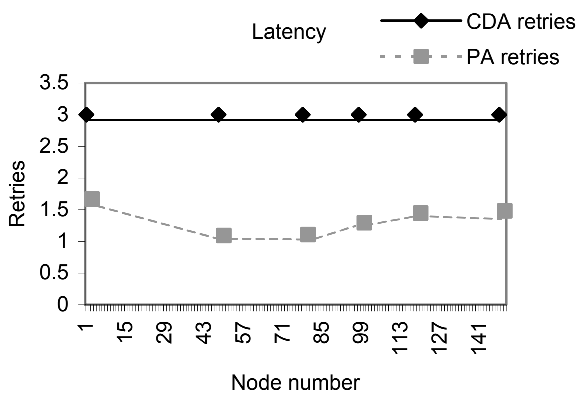

Figure 11. Latency for different node numbers

[1] Of course, the interval for multi-hop broadcast in CDA should be much longer than that for one-hop broadcast in PA; however, they are difficult to be computed in advance.

VI. CONCLUSION

Based on studies of scenarios in IP address allocation and several allocation algorithms proposed by other researchers, we proposed prophet allocation for large scale MANETs, which achieves low complexity, low communication, even distribution, and low latency. Both theoretical analysis and simulation results were conducted to demonstrate the superiority of prophet allocation over three other known methods.

With a little more effort, prophet allocation is able to handle all the three scenarios efficiently. However, the handling of Scenario *C* needs more research. For example, the procedure should be more specific, which will be our future work.

ACKNOWLEDGMENT

The authors wish to thank Jeff Boleng for providing the one-hop broadcast solution to ns-2, which is the basis for our simulation.

REFERENCES

[1] T. Clausen, P. Jacquet, A. Laouiti, et al., "Optimized Link State Routing Protocol", draft-ieft-manet-olsr-06.txt, September 2002 (work in progress)

[2] M. Gerla, X. Hong, and G. Pei, "Fisheye State Routing Protocol (FSR) for Ad Hod Networks", draft-ietf-manet-fsr-03.txt, June 2002 (work in progress)

[3] D. Johnson, D. Maltz, Y. Hu, and J. Jetcheva, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)", draft-ietf-manet-dsr-07.txt, February 2002, (work in progress)

[4] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", draft-ietf-manet-aodv-11.txt, June 2002 (work in progress)

[5] T. Narten, E. Nordmark, and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", Network Working Group RFC 2461, December 1998

[6] Zero Configuration Networking, http://www.ietf.org/html.charters/zeroconf-charter.html

[7] R. Droms, "Dynamic Host Configuration Protocol", Network Working Group RFC 2131, March 1997

[8] S. Thomson and T. Narten, "IPv6 Stateless Address Autoconfiguration", Network Working Group RFC 2462, December 1998

[9] C. Perkins, J. Malinen, R. Wakikawa, E. Belding-Royer, and Y. Sun, "IP Address Autoconfiguration for Ad Hoc Networks", draft-ietf-manet-autoconf-01.txt, November 2001 (work in progress)

[10] K. Weniger and M. Zitterbart, "IPv6 Autoconfiguration in Large Scale Mobile Ad-Hoc Networks", Proceedings of European Wireless 2002, Florence, Italy, Feb. 2002

[11] N. Vaidya, "Weak Duplicate Address Detection in Mobile Ad Hoc Networks", submitted for MobiHoc'02, June 2002

[12] Archan Misra, Subir Das, Anthony McAuley, and Sajal K. Das, "Autoconfiguration, Registration, and Mobility Management for Pervasive Computing", IEEE Personal Communication, August 2001, pp 24-31

[13] S. Nesargi and R. Prakash, "MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network", InfoCom 2002, June 2002

[14] K. Fall and K. Varadhan (editors), The *ns* Manual - the VINT Project, http://www.isi.edu/nsnam/ns/ns-documentation.html, April 2002

[15] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Routing Protocols", Proceedings of the Fourth Annual ACM/IEEE Inter-national Conference on Mobile Computing and Networking, pp. 85–97, October 1998.