

PAPER

Scalable and Efficient Ant-based Routing Algorithm for Ad-Hoc Networks

Yoshitaka OHTAKI^{†a)}, *Student Member*, Naoki WAKAMIYA[†], Masayuki MURATA[†],
and Makoto IMASE[†], *Members*

SUMMARY Ants-based routing algorithms have attracted the attention of researchers because they are more robust, reliable, and scalable than other conventional routing algorithms. Since they do not involve extra message exchanges to maintain paths when network topology changes, they are suitable for mobile ad-hoc networks where nodes move dynamically and topology changes frequently. As the number of nodes increases, however, the number of ants (*i.e.*, mobile agents or control messages) also increases, which means that existing algorithms have poor scalability. In this paper, we propose a scalable ant-based routing algorithm that keeps the overhead low while keeping paths short. Our algorithm uses a multistep TTL (Time To Live) scheme, an effective message migration scheme, and an efficient scheme for updating the probability of packet forwarding. Simulation experiments have confirmed that our proposed algorithm can establish shorter paths than the conventional ant-based algorithm with the same signaling overhead.

key words: *ant algorithms, MANET, routing, scalability*

1. Introduction

Driven by the emerging need for impromptu communication between mobile and wireless equipment, many researchers are developing proactive and reactive routing algorithms for mobile ad-hoc networks (MANETs). DSR [1] and AODV [2] are reactive algorithms, which investigate and establish paths only when they are needed. When a node has some data to send to another node, it searches for a path by flooding the network with control messages. The node does not need to keep exchanging control messages in order to maintain paths, but their dissemination introduces some delay before data packets can be sent and reactive routing algorithms are inefficient when there is much continuous but intermittent traffic in the network. OLSR [3] and TBRPF [4], on the other hand, are typical proactive routing algorithms. They prepare paths to all destination nodes beforehand and maintain them by exchanging control messages periodically. Although proactive routing algorithms immediately route a data packet toward a destination node through a pre-established and well-maintained path, they require each node to perform complicated controls for the discovery, maintenance, and updating of appropriate paths while keeping all paths consistent through the whole network. They also require a network to carry a lot of control traffic into a network. Proactive routing algorithms are therefore not applicable to networks composed of nodes that

have low-performance CPUs with small amounts of memory and that operate on battery power. This means they are not applicable to networks made up of PDAs, mobile phones, and wireless sensor nodes.

Another type of routing algorithms—one inspired by the behavior of social insects, more specifically, the foraging behavior of ants—has recently attracted the attention of researchers [5–8]. Ants establish the shortest path between food and their nest in a fully-distributed and autonomous fashion. An ant first wanders to find food. When it finds some, it returns to the nest while leaving a trail of chemical substances called pheromones on its way back. The pheromones attract other ants and guide them to the food. Although pheromones evaporate and decay, the following ants also leave additional pheromones and thus reinforce the path. Since the number of ants that complete their journeys to food in a given time is larger on a shorter path than on a longer path, a shorter path can accumulate more pheromone and attract more ants. Finally, the shortest of alternative paths comes chosen preferably and the most of ants take the shortest way to the food. Longer paths, however, are also maintained because some ants are attracted to them by remnants of pheromones. This makes the routing scheme robust and adaptable. When the shortest path is accidentally broken, a longer alternative path eventually becomes the new shortest path. Some papers [8, 9] verified that ant-based routing algorithms provided more reliable delivery of data packets on shorter paths than other conventional routing algorithms.

Such ant-based routing algorithms are simple but also robust as well as adaptive to topology changes because each ant carries only a fraction of the routing information and has a little influence on routing control. This has disadvantages as well as advantages. For example, control messages are necessary for constructing, maintaining, and updating routing tables, but data packets are delivered to their destinations even if some of the control messages are lost or carry incorrect information. An ant-based routing algorithm, however, cannot establish the shortest or appropriate path before enough control messages have been emitted. This means that as the number of nodes increases, the control messages needed to establish and maintain the shortest paths to all nodes increase the load on the network considerably.

One way to solve this load problem and attain scalability is by using hierarchical routing. Adaptive-SDR [9], for example, groups sensor nodes into clusters and directs

[†]The author is with the Graduate School of Information Science and Technology, Osaka University, 1-5 Yamadaoka, Suita, Osaka, 565-0871 Japan

a) E-mail: y-ohtaki@ist.osaka-u.ac.jp

data packets from a source node to a destination by using intra- and inter-cluster routing. And MABR [10] uses geographical information to divide a network into zones and uses intra- and inter-zone routing. These are more scalable ant-based routing algorithms, but they are complex and require additional information such as geographical location.

In this paper, we propose a simple and scalable ant-based routing algorithm that can keep the overhead low while keeping paths short. It is based on the uniform ant algorithm [11], which is a simple and basic ant-based algorithm in which each node has a probabilistic routing table. In the uniform ant algorithm, each entry of a table gives the probability of a neighboring node being chosen to be sent a data packet bound for a destination node. To update its corresponding entry in the tables of other nodes, each node periodically emits a control message without specifying a destination. When a node receives a message from a neighboring node, it increases the probability of forwarding a data packet bound for the source of the message to that neighboring node because it must be on a path leading to the source node of the message. In [11], it was shown that the uniform ant algorithm accomplished as low delivery delay as distance vector and link state algorithms, but it was more resilient to router state corruptions than those algorithms. However, to maintain routing tables up-to-date, nodes frequently emit control messages. Consequently, as the number of nodes increases, the load on a network increases considerably.

To make the uniform ant algorithm scalable to the number of nodes, we first control the range of message propagation by our multiple TTL scheme. It is based on an idea of the Hazy Sighted Link State (HSLS) algorithm [12] to limit TTL values. The HSLS algorithm was proposed in order to make a link-state algorithm scalable without using a complicated method like clustering or layering. HSLS uses a flooding scheme to effectively disseminate control messages over the whole network, whereas ant-based routing algorithms have control messages moving around the network in a node-by-node fashion. We therefore proposed the efficient message migration scheme and the forwarding probability updating scheme that let control messages travel to distant nodes and update routing tables effectively. We conducted simulation experiments to evaluate basic characteristics of our algorithm in stable ad-hoc networks. Then, we verified that our algorithm can establish drastically shorter paths than the uniform ant algorithm with the same signaling overhead can.

The rest of this paper is organized as follows. Section 2 describes the uniform ant algorithm on which our algorithm is based and Section 3 introduces the HSLS algorithm we adapted to our algorithm. Section 4 describes our algorithm, which consists of schemes for multistep TTL, effective message migration, and efficient updating of forwarding probabilities. Section 5 evaluates our algorithm with regard to the *shortestness* of established paths. Section 6 summarizes this paper and shows a future direction of our research.

Table 1 Example of a routing table

		Next Hop		
		node 1	node 2	node 3
Destination	node 4	0.65	0.2	0.15
	node 5	0.1	0.2	0.7

2. Uniform Ant Algorithm

In this section we explain the uniform ant algorithm [11], the basic ant-based algorithm on which our algorithm is based. In the uniform ant algorithm, packet forwarding is performed in a probabilistic way at each node. A node periodically emits an ant - that is, a control message - in order to maintain its entry in the routing tables of other nodes.

2.1 Routing Table

In the uniform ant algorithm, each node has a probabilistic routing table like Table 1 to perform multipath routing. Each row in the table corresponds to a destination and each column corresponds to a neighboring node. Neighboring nodes are nodes the node can communicate with directly. In this example, the node has three neighbors - node 1, node 2 and node 3 - and it knows of distant nodes 4 and 5. The value of an entry (i, j) in a table defines the forwarding probability of a data packet bound for node i to neighboring node j . For example, when the node generates or receives a data packet for node 5, it sends the packet to node 3 with a probability of 0.7, to node 2 with a probability of 0.2, and to node 1 with a probability of 0.1 according.

The probability of successful data delivery in a dynamically changing network is higher when such a probabilistic routing table is used than it is when a deterministic routing algorithm is used. Even if a neighboring node has a low forwarding probability, some of data packets are forwarded to it. This contributes to the robustness and adaptability of the algorithm. When the shortest path becomes unavailable because of a link failure or node disappearance, the other data packets on the alternative paths can arrive at the destination.

2.2 Route Discovery, Maintenance, and Updating

Each node periodically emits control messages to maintain and update, in routing tables of other nodes, a row in which it is the destination node. A control message is of the form of (h_s, c, TTL) , where h_s is an identifier of the source node of the control message, c is the sum of the costs of links that the message has traversed, and TTL (Time To Live) represents the number of hops that it can move further. A control message does not have any specific destination, and it wanders around a network until its TTL becomes zero. When a control message arrives a node, the cost of the link from which it came is first added to the sum c , a routing table is updated, and a next hop node is then chosen at random from all n neighboring nodes. The probability q_j that node j is chosen as a next-hop of a control message among all n

neighboring node except for the previous node i is given as

$$q_j = \frac{1}{n-1}. \quad (1)$$

Finally, the TTL is decreased by 1, and if it is still greater than 0 the control message is sent to the chosen neighbor.

A node receiving a control message updates its routing table. A key to the construction of the shortest path by randomly wandering control messages is that the direction from which a control message came indicates the direction of the source node of the message. Thus, a better (*i.e.*, shorter) path is expected to be established by increasing the probability that the neighboring node, from which the node receives a control message, is chosen as the next-hop node of data packets whose destination is the source of the control message. In some cases, a control message takes a long walk and arrives from a wrong neighboring node, and the probability is updated inappropriately. The number of control messages received from a correct direction, however, grows faster than the number of control messages from the other directions, so the probability of forwarding a data packet to an appropriate neighboring node is reinforced as time passes and the number of control messages increases. In addition, in updating the forwarding probability, the uniform ant algorithm takes into account the goodness of the path that a message traversed.

Consider that a node which has n neighbors (N_1, N_2, \dots, N_n) received a control message from neighboring node N_i ($1 \leq i \leq n$). The probability that N_j will be chosen as the next-hop node for the source of the message is represented as p_j . The node updates forwarding probabilities as follows:

$$p_j = \begin{cases} \frac{p_j + \Delta p}{1 + \Delta p}, & \text{if } j = i \quad (1 \leq j \leq n) \\ \frac{p_j}{1 + \Delta p}, & \text{if } j \neq i \quad (1 \leq j \leq n), \end{cases} \quad (2)$$

where Δp is given by

$$\Delta p = \frac{k}{f(c)} \quad (k > 0). \quad (3)$$

Here $f(c)$ is a non-decreasing function of the total cost c , and the constant k is called the learning rate of the algorithm. The learning rate defines the weight of one control message, and it is generally less than 0.1.

3. HSLS Algorithm

HSLS (Hazy Sighted Link State) [12] is an algorithm that improves the scalability of a link-state routing algorithm for ad-hoc networks. In generalized link-state routing, at regular intervals the nodes advertise link status to the other nodes (extremely, to the whole network) by means of message flooding whenever there was any change in the network topology (*e.g.*, link failure or addition of a new node) during the preceding interval. Consequently, when the topology changes frequently, as it does in mobile ad-hoc networks, the network is flooded with messages. HSLS avoids

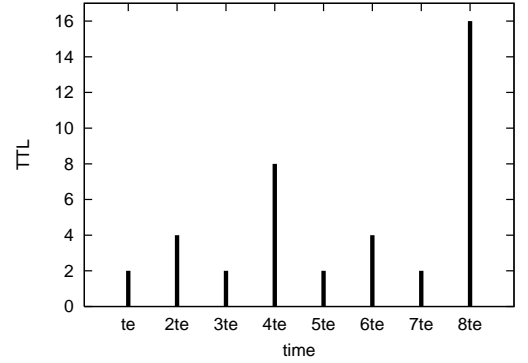


Fig. 1 Example of TTL changes in HSLS

this flooding by advertising topology changes more gradually and moderately by limiting the range of dissemination. Since topology changes influence paths traversing distant nodes far less than they do paths around points of changes, they need not be promptly reported to distant nodes.

When a node joins a network, it first advertises its appearance by emitting a global LSU (link state update), which is a LSU that has an infinite TTL and spreads over the whole network. We should note here that an infinite TTL does not imply that the message is immortal and remains in a network forever, because nodes discard control messages that have the same identifier that an already received message had. After the global advertisement, the newly joined node wakes up every t_e seconds and sends a LSU with TTL 2 if there had been any topology changes in the last t_e seconds. It also wakes up every $2 \times t_e$ seconds and sends a LSU with a TTL equal to 4 if there had been any changes in the last $2 \times t_e$ seconds. In short, a node wakes up every $2^{i-1} \times t_e$ ($i = 1, 2, 3, \dots$) seconds and transmits a LSU with TTL 2^i if there had been any changes in the last $2^{i-1} \times t_e$ seconds. If the TTL of a LSU becomes larger than the distance from the node to the most distant node in a network, that LSU becomes a global LSU and all counters and timers are initialized. Even when there are no topology changes, a node sends a global LSU at least every t_b seconds. Figure 1 illustrates how TTL changes in HSLS. If a change only takes place during 0 and t_e , a node sends messages at t_e , $2t_e$, $4t_e$, and $8t_e$ in the figure, where the range of message propagation is increased gradually.

4. Scalable Ant-based Routing Algorithm

In the uniform ant algorithm, as the number of nodes increases, the number of control messages increases and the network becomes heavily loaded. To reduce the load on a network and achieve higher scalability while providing low-delay delivery of data packets, in this section we propose the multistep TTL scheme, the message migration scheme, and the probability updating scheme.

4.1 Multistep TTL Scheme

In our proposal, TTL of the k -th control message is given as

$$T_k = 2^{x_k+1}, \quad (4)$$

where

$$x_k = \min(x_{max}, \max(x|k \equiv 0 \pmod{2^x})). \quad (5)$$

x_{max} is the maximum value used for preventing TTL from growing infinitely. We need to introduce a maximum bound on the TTL, since the way that a control message propagates a network differs between HSLs and ant-based algorithms. HSLs distributes control messages by using a flooding scheme. An original message is replicated and forwarded in a network, and the nodes within the range of the TTL are fully covered by copies of the message. On the other hand, an original control message itself walks around in a node-by-node fashion and does not clone in an ant-based scheme. In addition, no node discards a control message until its TTL becomes zero. Therefore, if we give an infinite TTL, a control message lives forever in our proposed scheme. For example, simulation results indicate that x_{max} should be set to the half of the number of nodes in a network.

4.2 Message Migration Scheme

In the uniform ant algorithm, a control message chooses a next-hop node at random when it moves. Such random walking leads to inefficient updates of routing tables because a control message often moves toward the source and the forwarding probability to a neighboring node in the opposite direction is reinforced. In addition, the routing tables of distant nodes cannot be updated efficiently because a control message walking randomly tends to stay around the source node. Thus, a control message with a TTL of x does not necessarily mean that it always reach a node x hops away from the source node. Furthermore, the number of control messages with a large TTL becomes particularly small when we use a multistep TTL scheme to attain higher scalability. Consequently, the probability that a control message reaches distant nodes decreases considerably.

We can consider that the higher the forwarding probability is, the closer the neighboring node to the destination of a data packet in a routing table. In other words, a neighboring node whose probability of being chosen as the next-hop node is small is in the direction opposite the destination node. Since the destination node in a routing table is the source node of a control message, a control message can move toward distant nodes by choosing as the next-hop node a neighboring node with lower forwarding probability. Therefore, in our algorithm, we use a probabilistic message migration scheme instead of random walking. In our message migration scheme, Eq. (1) is written as

$$q_j = \frac{\frac{1}{p_j}}{\sum_{k=1}^n \frac{1}{p_k} - \frac{1}{p_i}}. \quad (6)$$

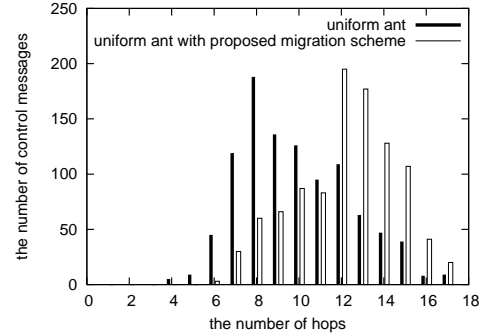


Fig. 2 Frequency distribution of maximum distance reached

To verify the effectiveness of our proposed migration scheme, we conducted a simulation experiment. The frequency distribution of the maximum distance that control messages reached is shown in Figure 2. The simulated network consisted of 100 nodes randomly distributed in a region 4550 m by 4550 m, and links were established between nodes whose separation was less than 250 m. The average degree was 9.3 and the diameter of the network was 33 hops. A node sent 1000 control messages with a fixed TTL value of 64. The x axis corresponds to the distance of nodes from the source of the control message, and the y axis shows the number of control messages. We show the results of the uniform ant algorithm and the uniform ant algorithm with our migration scheme. From the figure, it is clear that with our proposed migration scheme the control messages moved toward distant nodes effectively. One reason that not all messages moved effectively is that there was still a high probability of ineffective movement. Since the average degree of nodes was rather high and there were several neighboring nodes which were at the same or closer distance from the source node, the probability to choose the most preferable node became relatively small.

4.3 Forwarding Probability Updating Scheme

The scheme described in 4.2 enables control messages with a large TTL can to reach distant nodes effectively. In the uniform ant algorithm, however, the amount by which the forwarding probability in a routing table is increased or decreased inversely proportional to the cost of a path as given by Eq. (3). Therefore, even if a control message traversed the shortest path, the routing table on a distant node will not be updated often enough. We therefore modified the $f(c)$ term in Eq. (3) to $f(c - c_{min})$, where c_{min} corresponds to the minimum cost among all preceding control messages from the same source node. Consequently, a control message which successfully reaches a distant node can put a sufficient amount of pheromone on the path it traversed. To maintain the minimum cost c_{min} up-to-date, a node has to check the cost of every new control messages and compare the value with c_{min} . However, since the original uniform ant algorithm always evaluate the cost c and a node only needs to keep one minimum value for each of source nodes, the

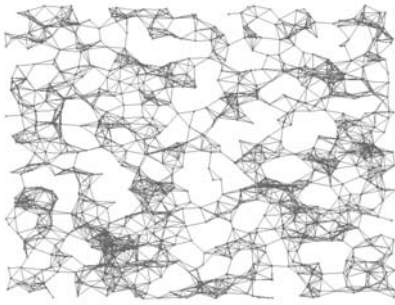


Fig. 3 Network topology

overhead is not high. When we consider mobility of nodes and changes of quality of wireless links, we have to introduce a timer mechanism or a smoothing function to avoid the influence of out-of-date conditions. However, this remains a future research work.

5. Simulation and Evaluation

We conducted simulation experiments to evaluate the performance of our algorithm.

5.1 Simulation Model

We generated a network by randomly placing 1024 nodes in a 4550x4550-m region and, assuming the range of the radio signals to be 250 m, establishing links between nodes less than 250 m apart. Nodes were stable and did not move. The average degree of a node was 9.3, and the maximum distance between two arbitrary nodes was 33 hops. An example of the simulated network topology is illustrated in Fig. 3. The cost between all links was identical and was set to 1. Parameters were set as $f(c) = 0.1 \times c$ and $k = 0.1$. We conducted ten simulation experiments and used averaged values to draw each line and point in the following figures.

5.2 Basic Characteristics

We first evaluated the basic characteristics of our proposed algorithm. For comparison, we conducted simulation experiments with the uniform ant algorithm, our proposed algorithm, and three different combinations of our proposed schemes. Method A used $f(c - c_{min})$ instead of $f(c)$ in Eq. (3), but the other terms were the same as in the uniform ant algorithm. Method B additionally used our message migration scheme. And method C used the multistep TTL scheme and $f(c - c_{min})$. The algorithms and methods we evaluated are summarized in Table 2. First, one designated node was randomly chosen in a network. It became a source of control messages and tried to establish paths from the other nodes to itself by emitting control messages. Then, at the timing appropriate for evaluation, all other nodes sent a data packet to the chosen node. To compare the algorithms and methods from the viewpoint of routing efficiency, we defined a

Table 2 Evaluated algorithms and methods

	prob. update	migration	multiTTL
uniform ant			
method A	O		
method B	O	O	
method C	O		O
proposed alg.	O	O	O

measurement of the average hop ratio. Before the experiments, we calculated the optimal shortest hop count from each of the other nodes to the chosen node. The optimal hop count corresponds to the distance to a node. By dividing the observed number of hops by the optimal hop count, we obtained the hop ratio for each of the nodes. Finally, by taking an average of hop ratios of nodes on the same distance, i.e., optimal hop count, from the chosen node, the average hop ratio was calculated for each distance.

To compare methods fairly, we should consider the load on the network. Of the five evaluated algorithms and methods, our proposed algorithm and method C use the multistep TTL scheme, whereas the other three emit control messages with the fixed TTL $2^{x_{max}+1}$. When we assume that, independently of algorithms and methods, a node emits control messages at regular and identical intervals and x_{max} is set to 8, (i.e., the maximum fixed TTL is set to 512), the total number of control messages in the network per unit time when the algorithm and method using multistep TTL are used is about 1/50 of what it is when the algorithm and methods using a constant TTL are used. Thus, to carry out comparisons fairly from the viewpoint of the load on a network, we consider the case that the total number of hops that all control messages took (i.e., the amount of network resource consumed) are the same among the algorithms and methods. For example, when the total hop count is 10000, the number of control messages sent in the uniform ant, method A, and method B is about 20, whereas in method C and the proposed algorithm it is about 1024.

We define the signaling overhead as $\frac{2sTTL}{id}$, where s is the size of the control message, TTL is TTL value of control messages, i is the interval at which control messages are emitted, and d is the average degree. It is derived by dividing the amount of control messages in a network $\frac{sNTTL}{i}$ by the number of links $\frac{dN}{2}$, where N corresponds to the number of nodes. The signaling overhead corresponds to the amount of control traffic per link. When we assume that the size of a control message is 100 bytes, each node emits 8 messages per sec, the average degree is 10, and maximum TTL is set to 512, the overhead is 655 kbps in the uniform ant algorithm and is only 12 kbps in the proposed algorithm.

Results are shown in Figs. 4(a) through 4(d). Each figure shows how the average hop counts between nodes depends on the distance from the chosen node. On x-axis, the distance of the source node of data packets to the chosen node is shown in terms of the number of hops. On y-axis, the average hop count that data packet experienced is shown. Figures 4(a), 4(b), 4(c), and 4(d) respectively correspond to

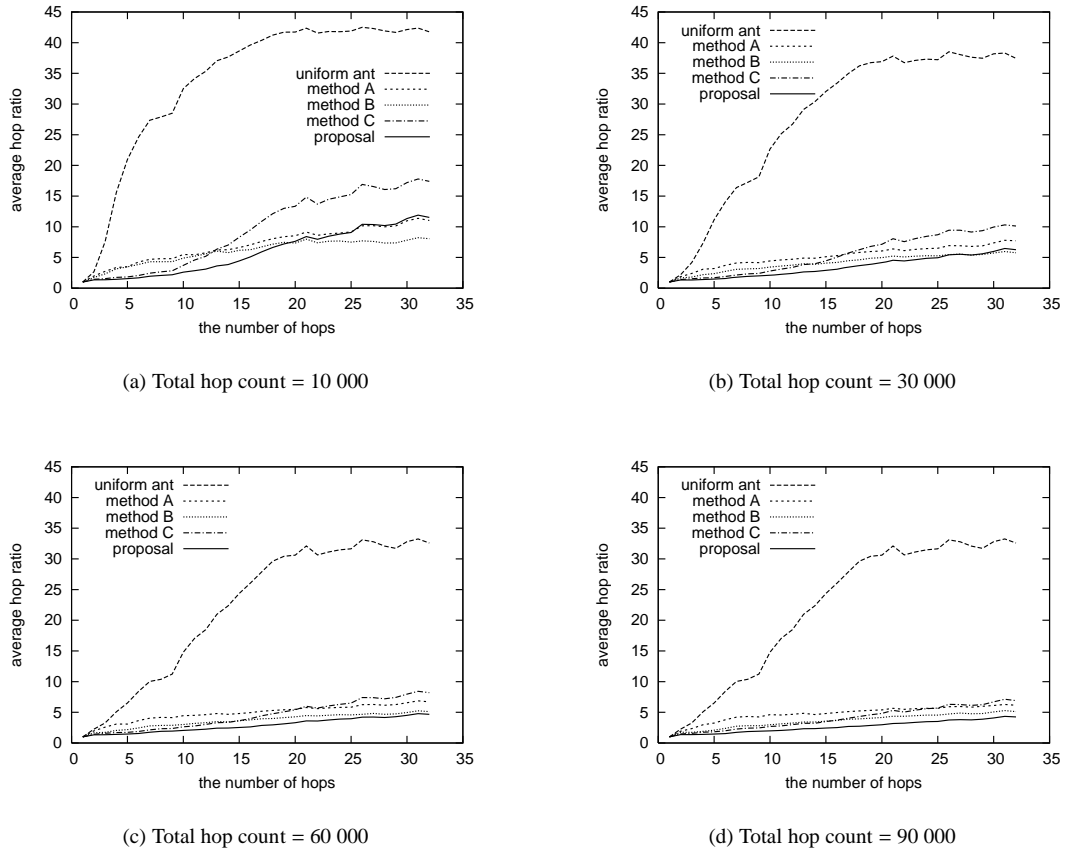


Fig. 4 Average hop ratio per distance

total hop counts of 10 000, 30 000, 60 000, and 90 000.

First, by comparing the uniform ant algorithm and method A, we can see that the average hop count is apparently reduced by using our scheme for updating the forwarding probability. In the uniform ant algorithm, a control packet that reaches a distant node has little influence on a routing table. Therefore, the quality of paths from distant nodes stays low even if some control messages reach them. Although when the uniform ant algorithm is used, paths gradually improved as time passes and the total number of control messages increases from Fig. 4(a) to Fig. 4(d), the average hop count is much larger than that it is when the other algorithm or methods are used.

Second, as shown in Fig. 4(a), paths from distant nodes are better when method B is used than when method A is used. When our migration scheme is used, control messages move toward distant nodes effectively and the routing tables of those nodes are updated often. Furthermore, as time passes from Fig. 4(a) to Fig. 4(d), the average hop ratio at closer nodes also becomes smaller with method B than with method A. This is because randomly walking control messages in method A wrongly and wastefully update routing tables around the source nodes of control messages.

Third, by comparing methods A and C in Fig. 4(a), we can see that the multistep TTL scheme improves paths

around the source of control messages at the cost of paths from distant nodes, but the average hop ratio at distant nodes is still far smaller than it is when the uniform ant algorithm is used. In addition, Fig. 4(d) shows that paths gradually improve as time passes when method C is used. When we consider an ad-hoc network, not necessarily a mobile one, where the topology changes and is not stable, it is inefficient to maintain the exact and optimal routing information at distant nodes by consuming much bandwidth with control messages [12, 13]. Even if routing information is abstract at distant nodes, data packets can be appropriately guided and delivered as they approach the destination node as long as the information around the destination is well-maintained and up-to-date. Thus, we consider that in mobile ad-hoc networks the multistep TTL scheme works better than the uniform ant algorithm does.

Finally, Fig. 4(a) shows that our proposal outperforms the others in sending data packets from nodes at distance of less than 20 hops. The quality of paths to more distant nodes is still small, though, and is similar to that when method A is used. This is because the number of control messages that reach distant nodes is small in the proposal due to the multistep TTL scheme. However, the quality of paths was improved by our migration scheme in comparison with method C. As time passed more control messages reached distant

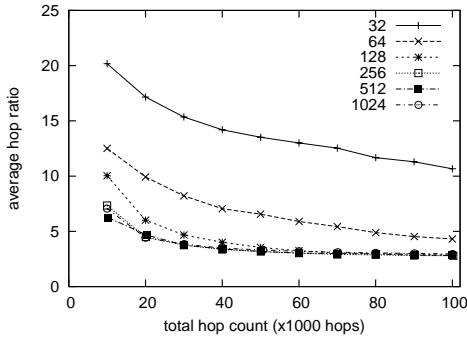


Fig. 5 Influence of Maximum TTL

nodes, then the proposed scheme further improved all paths and yielded the shortest paths.

5.3 Setting of Maximum TTL Value

In this subsection, we consider the influence of the maximum TTL $2^{x_{max}+1}$. Figure 5 illustrates the transition of the average hop ratio for cases of maximum TTL = 32, 64, 128, 256, 512 and 1024. The x axis corresponds to the total hop count (*i.e.*, time) and the y axis shows the hop ratio averaged over all nodes. As the maximum TTL increases, more control messages can reach distant nodes and update their routing tables. Thus the average hop ratio is expected to decrease. However, the maximum TTL of 1024 leads to slightly larger average hop ratios when the total hop count is low. This is because the total number of hops of 10 000 means that 987 control messages have been sent when the TTL is 1024 whereas 1024 control messages have been sent when the TTL is 512. For routing tables to be well updated, the sufficient number of control messages should be sent. Thus, a scheme with a TTL of 1024 needs more time to establish appropriately short paths than the others do. The difference is not large, however, and we think that our proposed algorithm is insensitive to the maximum TTL at about half of the number of nodes in a network.

We do not define the method for setting the maximum TTL of control messages in our proposed algorithm. We cannot assume that all nodes know the exact number of nodes in a network. One possible way for a node to estimate the number of nodes independently is to derive the maximum number of nodes from its own degree d , (*i.e.*, the number of neighboring nodes) and the maximum distance T_{max} of nodes which it knows as $d^{T_{max}}$. Although this is a rough and over-estimation, we consider that it is effective enough for the insensitivity of the proposal to the maximum TTL shown in Fig. 5. The verification of this idea and considerations on a better estimation remain as future research works.

5.4 Setting of Multistep TTL

In our proposed algorithm, we empirically use the same sequence of multistep TTL as in HSLs. Although this way

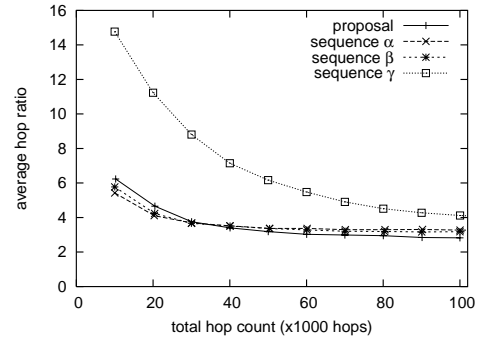


Fig. 6 Influence of TTL sequence

was shown to be optimal for HSLs [12], we should consider the difference in the way of message dissemination as mentioned in 2.1. In this subsection, we consider three other sequences given as follows:

$$T_k = \begin{cases} 2^{2x_k+1} & (\text{sequence } \alpha) \\ 3^{x_k+1} & (\text{sequence } \beta) \\ x_k^2 & (\text{sequence } \gamma) \end{cases} \quad (7)$$

where x_k is the same as in Eq. (5). We evaluated those alternatives in simulation experiments. In our proposed algorithm, the frequency that a control message with a given TTL appears is defined by a function of $\log_2 TTL$. Sequence α generates control messages of larger TTL more often than our Proposed algorithm does. Sequence β corresponds to a function of $\log_3 TTL$, and sequence γ generates control messages as a function of \sqrt{TTL} . To have similar maximum TTL values for all the sequences, we respectively set x_{max} to 8, 4, 5, and 22 for the proposed algorithm, sequence α , sequence β , and sequence γ . They lead to maximum TTL values of 512, 512, 729, and 529.

The results shown in Fig. 6 indicate that the sequence does not influence much except for sequence γ . Since the number of control messages with a large TTL is too small in sequence γ , the quality of paths from distant nodes is low and reduces the average hop ratio. The proposed algorithm provides paths slightly worse than sequences α and β do when the total number of hops, which corresponds to the number of control messages, is small. As time passes, the quality of paths improves, however, and the proposed algorithm outperforms the others. However, the difference is small in our simulation environments.

6. Conclusions and Future Works

We obtained a scalable ant-based routing algorithm by using a multistep TTL scheme proposed in HSLs, and we also proposed several schemes for effective message diffusion and efficient table updating. Through simulation experiments, we showed that the overhead of our proposed algorithm is about $12/655=1.8\%$ that of the uniform ant algorithm and that our proposed algorithm constructs better paths than the uniform ant algorithm does.

Although we have shown the basic characteristics of

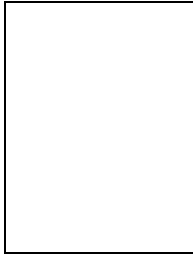
our proposed algorithm, we need to evaluate its effectiveness in some other environments where the network topology is different and changes often where nodes are mobile. Conjectured from inherent features of ant-based algorithms, e.g., robustness, adaptability, and reliability, and known results of performance evaluations of proactive routing protocols with a multistep TTL scheme such as FSR (Fisheye State Routing) and FZRP (Fisheye Zone Routing Protocol), we consider that our proposal attains good performance in mobile environments. We also plan to conduct comparisons with other routing algorithms such as the above two. In addition, we based our algorithm on the uniform ant routing algorithm, but we think our approach would be viable if coupled with other ant-based algorithms. We therefore plan to consider other combinations.

Acknowledgements

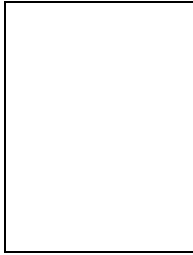
This work was partly supported by “The 21st Century Center of Excellence Program” and a Grant-in-Aid for Scientific Research (A)(2) 16200003 from The Japanese Ministry of Education, Culture, Sports, Science and Technology.

References

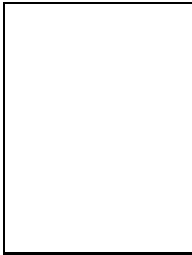
- [1] D.B. Johnson, D.A. Maltz, Y.C. Hu, and J.G. Jetcheva, “The dynamic source routing protocol for mobile ad hoc networks,” IETF Internet Draft `draft-ietf-manet-dsr-09.txt`, April 2003.
- [2] C.E. Perkins and E.M. Royer, “Ad hoc on-demand distance vector routing,” Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '99), New Orleans, LA, pp.90–100, Feb. 1999.
- [3] T. Clausen and P.J. et al., “Optimized link state routing protocol,” Request for Comments (RFC) 3626, July 2003.
- [4] R. Ogier, M. Lewis, and F. Templin, “Topology dissemination based on reverse-path forwarding (TBRPF),” IETF Internet Draft `draft-ietf-manet-tbrpf-10.txt`, July 2003.
- [5] M. Güneş and O. Spaniol, “Routing algorithms for mobile multi-hop ad-hoc networks,” Proceedings of International Workshop on Next Generation Network Technologies, Oct. 2002.
- [6] J.S. Baras and H. Mehta, “A probabilistic emergent routing algorithm for mobile ad hoc networks,” Proceedings of Workshop on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, March 2003.
- [7] S. Marwaha, C.K. Tham, and D. Srinivasan, “A novel routing protocol using mobile agents and reactive route discovery for ad hoc wireless networks,” Proceedings of IEEE International Conference On Networks (ICON 2002), pp.311–316, Aug. 2002.
- [8] G.D. Caro, F. Ducatelle, and L.M. Gambardella, “AntHocNet: an ant-based hybrid routing algorithm for mobile ad hoc networks,” Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII), pp.461–470, Sept. 2004.
- [9] I. Kassabalidis, M. El-Sharkawi, R. II, P. Arabshahi, and A. Gray, “Adaptive-SDR: Adaptive swarm-based distributed routing,” Proceedings IEEE World Congress on Computational Intelligence (WCCI 2002), Hawaii, May 2002.
- [10] M. Heissenbüttel and T. Braun, “Ants-based routing in large scale mobile ad-hoc networks,” Proceedings of Kommunikation in verteilten Systemen (KiVS '03), Leipzig, Germany, pp.91–99, Feb. 2003.
- [11] D. Subramanian, P. Druschel, and J. Chen, “Ants and reinforcement learning: A case study in routing in dynamic networks,” Tech. Rep. TR96-259, Rice University, July 1998.
- [12] C.A. Santiváñez, R. Ramanathan, and I. Stavrakakis, “Making link-state routing scale for ad hoc networks,” Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing, pp.22–32, Oct. 2001.
- [13] G. Pei, M. Gerla, and T.W. Chen, “Fisheye state routing: A routing scheme for ad hoc wireless networks,” Proceedings of IEEE International Conference on Communications, pp.70–74, June 2000.



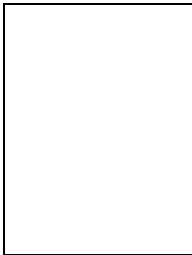
Yoshitaka Ohtaki received the M. E. degree in the Information and Computer Sciences from Osaka University, Osaka, Japan, in 2005. He studies information networking and his research work is in the area of ad hoc networks. He is a member of IEICE.



Naoki Wakamiya received the M.E. and Ph.D. degrees from Osaka University, Japan, in 1994 and 1996, respectively. He has been with Osaka University since April 1996, and he is now an Associate Professor of Graduate School of Information Science and Technology since April 2002. His research interests include overlay networks, mobile ad-hoc networks, and wireless sensor networks. He is a member of IEICE, IPSJ, ACM, and IEEE.



Masayuki MURATA received the M.E. and D.E. degrees from Osaka University in 1984 and 1988, respectively. In April 1984, he joined Tokyo Research Laboratory, IBM Japan, as a Researcher. In September 1987, he moved to Osaka University, Japan, as an Assistant Professor. He is now a Professor in the Graduate School of Information Science and Technology since April 2004. He has more than three hundred papers of international and domestic journals and conferences. His research interests include computer communication networks, performance modeling and evaluation. He is an IEICE Fellow. He is a member of IEEE, ACM, The Internet Society, IEICE and IPSJ.



Makoto Imase received his B.E. and M.E. degrees in information engineering from Osaka University in 1975 and 1977, respectively. He received his D.E. degree from Osaka University in 1986. From 1977 to 2001, He was engaged Nippon Telegraph and Telephone Corporation (NTT). He has been a Professor of Graduate School of Information Science and Technology at Osaka University since 2002. His research interests are in the area of information networks, distributed systems and graph theory. He is a member of IEICE, IPSJ, and JSIAM