논문 17-42-01-10 The Journal of Korean Institute of Communications and Information Sciences '17-01 Vol.42 No.01
https://doi.org/10.7840/kics.2017.42.1.64

양방향 경로 설정 및 루프 방지를 통한 개선된 AntHocNet

라프만 샴스 우르◆, 남 재 충*, 아즈말 칸**, 조 유 제°

Improved AntHocNet with Bidirectional Path Setup and Loop Avoidance

Shams ur Rahman◆, Jae-Choong Nam*, Ajmal Khan**, You-Ze Cho°

요 약

MANET (Mobile Ad hoc Network)에서 라우팅은 네트워크 토폴로지의 동적인 변화에 큰 영향을 받는다. AntHocNet은 집단 개미가 최적 경로를 통해 먹이를 찾아가는 원리를 모방한 집단생태 특성 기반 MANET 라우팅 프로토콜이다. 하지만, AntHocNet은 다른 MANET 라우팅 프로토콜과 달리 단방향 경로만을 지원하여 양방향 통신이 요구되는 다양한 응용 환경에서 사용하기에 많은 제약이 따른다. 또한, AntHocNet은 다중 경로를 통한 확률적 라우팅으로 인해 루핑 문제 (looping problems)를 빈번히 발생시킨다. 본 논문에서는 AntHocNet에서 양방향 경로 수립을 위한 향상된 경로 수립 방안을 제안한다. 또한, 다양한 시나리오별 루핑 문제의 발생 원인을 분석하고 루프 방지를 위한 해결 방안을 제시한다. NS-2 시뮬레이션을 통해 기존 AntHocNet과의 성능을 비교하였으며, 제안 방안이 라우팅 오버헤드, 종단간 지연 시간, 패킷 전달률 측면에서 기존 방안에 비해 우수한 성능을 보임을 확인하였다.

Key Words : MANETs, routing, bio-inspired, ant colony optimization, AntHocNet, bidirectional path

ABSTRACT

Routing in mobile ad hoc networks (MANETs) is highly challenging because of the dynamic nature of network topology. AntHocNet is a bio-inspired routing protocol for MANETs that mimics the foraging behavior of ants. However, unlike many other MANET routing protocols, the paths constructed in AntHocNet are unidirectional, which requires a separate path setup if a route in the reverse direction is also required. Because most communication sessions are bidirectional, this unidirectional path setup approach is often inefficient. Moreover, AntHocNet suffers from looping problems because of its property of multiple paths and stochastic data routing. In this paper, we propose a modified path setup procedure that constructs bidirectional paths. We also propose solutions to some of the looping problems in AntHocNet. Simulation results show that performance is significantly enhanced in terms of overhead, end-to-end delay, and delivery ratio when loops are prevented. Performance is further improved, in terms of overhead, when bidirectional paths setup is employed.

※ 본 연구는 2016년도 한국연구재단[NRF-2015R1D1A1A01059623]의 지원을 받아 수행되었습니다.
◆ First Author : Kyungpook National University, shamsuet@gmail.com, 학생회원
° Corresponding Author : Kyungpook National University, yzcho@ee.knu.ac.kr, 종신회원
* Kyungpook National University, jch-nam@ee.knu.ac.kr, 학생회원
** COMSATS Institute of Information Technology Attock Campus, Pakistan, drajmal@ciit-attock.edu.pk
논문번호 : KICS2016-09-265, Received September 20, 2016; Revised December 20, 2016; Accepted December 20, 2016

Ⅰ. Introduction

Mobile ad hoc networks (MANETs) have attracted increasing research attention over the last few decades. A MANET is a network of mobile nodes that communicate over a wireless medium. Because of the network's ad hoc nature, there is no infrastructure, and therefore, the mobile nodes themselves have to perform the routing function. The infrastructure-less nature of MANETs brings with it many advantages, such as allowing rapid deployment. However, the dynamic nature of the topology of MANETs makes routing a challenging issue. Mobile nodes that are within transmission range of each other can communicate directly. However for communication among nodes that are not in the transmission range of each other, intermediate nodes have to route the packets towards the destination.

Several routing protocols for MANETs have been proposed using traditional approaches[1]. Because nature offers solutions to many complex problems, researchers have also looked into it for finding solutions to the routing problem and have proposed some bio-inspired routing protocols[2,3]. The laws that govern biological systems and their dynamics are based on very few simple and generic rules. However, without any centralized control, these systems exhibit patterns that are highly collaborative and effective in terms of synchronization, task allocation, etc. MANETs are inherently infrastructure-less and dynamic and have to operate autonomously. Bio-inspired routing approaches can provide efficient and scalable solution strategies to such types of networks[4,5].

Many routing protocols inspired by various biological species have been proposed[6-10]. Ant Colony Optimization (ACO)[11] is a problem-solving framework inspired by the foraging behavior of biological ants. An ant colony can to discover the shortest path to a food source. Ants indirectly share information with each other. They achieve this by laying a chemical substance, called pheromone, along the path they travel. This type of information sharing is called stigmergy. Other ants are attracted to the pheromone depending on its intensity. A shorter path is traversed more quickly and more frequently by ants, and therefore, receives more pheromone. This attracts even more ants to it. This way, suboptimal paths are eventually abandoned and all ants converge on the shortest path. ACO is extensively used in routing and load balancing[12]. Several routing protocols are based on the ACO framework, such as ARA[8], PERA[13], MACO[14], AntNet[15], and AntHocNet[9,10].

AntHocNet is a hybrid multipath algorithm for MANETs, designed along the principles of ACO routing[10]. It consists of both reactive and proactive components. It does not maintain routes to all possible destinations at all times (like the original ACO algorithms for wired networks), but only sets up paths when they required at the start of a data session. This is done in the reactive route setup phase, where ant agents called reactive forward ants (FANTs) are launched by the source in order to find multiple paths to the destination, and backward ants (BANTs) return to the source to set up the paths. According to the common practice in ACO algorithms, the paths are set up in the form of pheromone tables that indicate their respective quality. After the route setup, data packets are routed stochastically over the different paths following these pheromone tables. While the data session is running, the paths are monitored, maintained, and improved proactively using different agents, called 'proactive forward ants'. The algorithm reacts to link failures with either a local route repair or by warning preceding nodes on the paths.

AntHocNet overcomes some of the limitations found in other bio-inspired routing protocols. Unlike Termite[6], AntHocNet does not suffer from suboptimal paths problems. Unlike ARA[4] and Termite, AntHocNet's pheromone laying mechanism helps push data traffic away from congested paths thus, balancing load across the network. AntHocNet's path maintenance mechanism has the characteristic of maintaining existing paths and exploring new and better paths. Moreover, unlike BeeAdhoc[11], AntHocNet does not depend on the

availability of agents, such as foragers, to carry data packets.

However, AntHocNet suffers from two key shortcomings. First, unlike many other well-known MANET routing protocols, such as AODV[16] and DSR[17], AntHocNet's path setup procedure constructs unidirectional paths, which can be highly inefficient in the majority of cases. In this paper, we propose a bidirectional path setup scheme for AntHocNet. Second, the use of multiple paths and stochastic data routing in AntHocNet causes looping of data packets. which can severely degrade performance. But the looping problems in AntHocNet have not been mentioned in any other literatures. We elaborate the phenomenon of data packet looping with some scenarios and present some solutions to prevent such loops from occurring. Simulation results show that loop prevention significantly reduces overhead and end-to-end delay, and improves packet delivery ratio. Moreover, use of the proposed bidirectional path setup scheme further reduces overhead and end-to-end delay.

The rest of the paper is organized as follows. Section 2 presents an overview of AntHocNet. Section 3 describes the proposed scheme of bidirectional path construction. In Section 4, some loop formation scenarios in AntHocNet are identified and their solutions are presented. Section 5 discusses simulation results. Finally, Section 6 provides some conclusions.

Ⅱ. Overview of AntHocNet

In this section, we describe the procedures for path setup, path maintenance, and link failure handling in AntHocNet. We also describe the data routing algorithm of AntHocNet.

2.1 Reactive path setup

The source node broadcasts a reactive FANT. Every intermediate node that receives a reactive FANT will broadcast the ant further if it has no path to the given destination; otherwise, it unicasts the ant by choosing the next hop stochastically according to the following equation:

$$P_{rd} = \frac{(T_{rd}^i)^{\beta_1}}{\sum_{j \in N_d^i} (T_{jd}^i)^{\beta_1}}, \beta_1 \geq 1 \quad (1)$$

where T_{rd}^i is the amount of pheromone on the link from node n_i to n_r for destination d, whereas N_d^i is the set of neighbors of n_i over which a path to the destination d is known. Broadcasting results in quick ant proliferation. It is possible for a node to receive multiple replicas of the same ant. In such case, the length of the already travelled path and the time already taken by the replica (of ant) are compared to those of the previous replicas (of ant) and is only forwarded if they meet certain conditions. This helps remove ants that have come over bad paths while allowing for a mesh of sufficiently disjoint multiple paths. The reactive FANT maintains a list $\boldsymbol{P}$ of its visited nodes [n_1, n_2, ⋯, n_k]. When the ant reaches the destination, it is converted to into a BANT. BANT takes the same path as its corresponding FANT, but in the reverse direction. At each node, BANT computes $\widehat{T}_P$, the estimate of time required by a data packet to reach the destination from the current node. This estimate is computed incrementally by using the following relationship:

$$\widehat{T}_P = \sum_{i=1}^{k-1} \widehat{T_{i+1}^i} \quad (2)$$

where $\widehat{T_{i+1}^i}$ is a local estimate of time required by a data packet to travel from node n_i to n_{i+1}. $\widehat{T_{i+1}^i}$ is computed using the equation.

$$\widehat{T_{i+1}^i} = (Q_{mac}^i + 1)\,\widehat{T_{mac}^i} \quad (3)$$

where Q_{mac}^i is the number of packets in the queue at the Media Access Control (MAC) layer of node n_i and $\widehat{T_{mac}^i}$ is the running average of the time a packet waited at the MAC layer of node n_i before being successfully transmitted. $\widehat{T_{mac}^i}$ incorporates

channel access activities that include channel congestion. BANT sets up a path to the destination at every intermediate node. This is done by creating or updating an entry T_{rd}^{i} in the routing table. This entry indicates the goodness of going over the next hop n_r towards the destination d from the current node n_i. T_{rd}^{i} is computed using the following two equations:

$$\tau_d^i = \left(\frac{\widehat{T_d^i} + h\, T_{hop}}{2} \right)^{-1} \tag{4}$$

$$T_{rd}^{i} = \gamma T_{rd}^{i} + (1-\gamma)\tau_d^i, \gamma \in [0,1] \tag{5}$$

where h is the number of hops from the current node to the destination and T_{hop} is a constant that represents the time to take one hop in unloaded conditions.

2.2 Stochastic data routing

Multiple paths are constructed from the source to the destination during the path setup phase. Data are forwarded along these paths according to their goodness indicated as by the pheromone values in the routing table. At each node, the next hop is chosen stochastically using the following equation:

$$P_{rd} = \frac{(T_{rd}^{i})^{\beta_2}}{\sum_{j \in N_d^i} (T_{jd}^{i})^{\beta_2}}, \beta_2 \geq \beta_1 \tag{6}$$

This stochastic data routing results in automatic load balancing with better and less congested paths having a higher probability of being chosen.

2.3 Proactive path maintenance

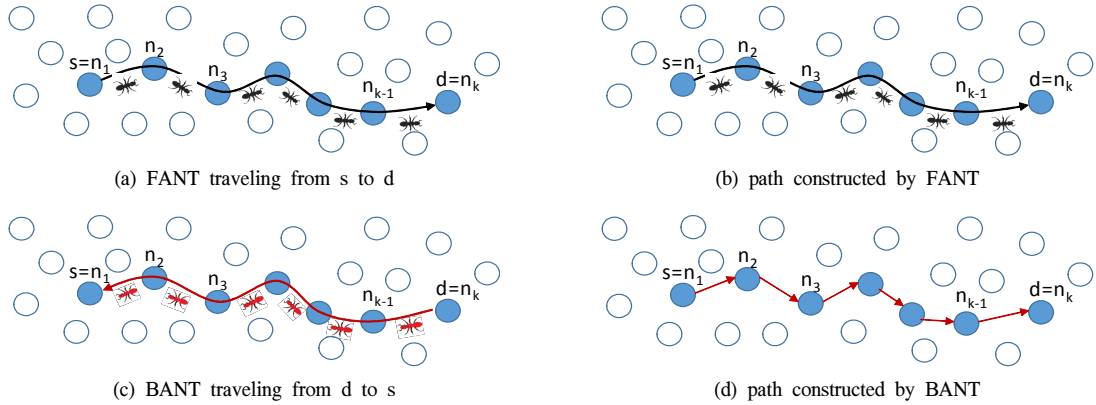
Another type of ants, called proactive FANTs, is launched regularly while the data session is running. For the most part, a proactive FANT is unicast by the intermediate nodes similarly to a reactive FANT. However, there is a small probability (0.1) that an intermediate node might decide to broadcast it. Although unicast results in sampling and refreshing existing paths, a broadcast at some intermediate node helps explore new paths. However, a proactive FANT cannot be broadcasted more than twice. This helps control flooding the network with proactive FANTs. Nodes regularly broadcast 'hello' messages that help each node know its neighbors. Because of this, when a FANT arrives at a neighbor node of the destination, it is forwarded directly to the destination. If a node misses two consecutive 'hello' messages from a neighbor, the node assumes it has lost that neighbor and removes its entry from the routing table.

2.4 Link failures

If a node does not receive a 'hello' message or any other signal from a neighbor for a certain amount of time (t_{hello} × *allowed-hello-loss*), the neighbor is assumed to have been lost. In addition, if the transmission of a unicast message to a neighbor fails, the neighbor is assumed to have been lost. The loss of neighbor means link failure, and thus the node notifies this to all its neighbors. If a link fails during an on-going data session, the node looks for alternate routes. If no alternate route is available, the node attempts to repair the route locally by broadcasting a route repair FANT towards the involved destination. The route repair ant is managed similarly to the reactive FANT by the intermediate nodes. However, the route repair FANT cannot be broadcasted more than twice. Meanwhile, packets destined for the involved destination are buffered. If the local link repair is successful, the buffered packets are forwarded along the repaired path; otherwise, the packets are dropped and a link failure notification is broadcasted.

Ⅲ. Bidirectional Path Setup

As described in Section 2.1, the path setup procedure in AntHocNet constructs only forward paths - paths that lead towards the destination node - and are therefore unidirectional paths. A BANT constructs a forward path from each intermediate node and source node as it visits them. However, communication sessions are usually bidirectional. This is particularly true when Transmission Control Protocol (TCP) or Stream Control Transmission



(a) FANT traveling from s to d

(b) path constructed by FANT

(c) BANT traveling from d to s

(d) path constructed by BANT

Fig. 1. Path setup procedure.

Protocol (SCTP)[18] is involved because received packets have to be acknowledged. In such cases, a bidirectional path is highly desirable because it can result in significant overhead and delay reduction for setting up backward paths.

The original AntHocNet utilizes BANTs to construct forward paths. Our proposed scheme constructs forward paths similarly to AntHocNet. However, to construct backward paths, our proposed scheme utilizes FANTs. This requires the FANT to carry two additional fields, hop count and time estimate, but no additional control packets are used. Source node s sends a FANT toward destination d, as shown in Fig. 1(a). After this FANT reaches destination node d, each of the intermediate nodes [n_2, n_3, ⋯, n_{k-1}] and destination d will have a path constructed to node s (FANT's source node) as shown in Fig. 1(b); that is, they will have an entry for s in their pheromone table τ^i. FANT, similarly to BANT, incrementally computes an estimate $\widehat{T}_P$ of the time required by a data packet to travel over the intermediate nodes toward the source node. This estimate is used to update the routing tables. $\widehat{T}_P$ is the sum of local estimates $\widehat{T_i^{i+1}}$ (note that this is the reverse of $\widehat{T_{i+1}^{i}}$, which is used in AntHocNet for path construction from source to destination). $\widehat{T_i^{i+1}}$ is the time estimate for going from node n_{i+1} to node n_i. The equation.

$$\widehat{T}_P = \sum_{i=1}^{k-1} \widehat{T_i^{i+1}} \tag{7}$$

computes the time estimate for a packet to reach the source node (n_1) from the destination (n_k). For any intermediate node, for example n_r, the time estimate, for a packet to reach the source node from that intermediate node is

$$\widehat{T}_P = \sum_{i=1}^{r-1} \widehat{T_i^{i+1}} \tag{8}$$

Now at each intermediate node i, the time estimate and hop count (which the FANT carries with itself) are used to calculate τ_s^i.

$$\tau_s^i = \left(\frac{\widehat{T_s^i} + h\, T_{hop}}{2} \right)^{-1} \tag{9}$$

where h is the number of hops from the current node to the source and T_{hop} is a constant that represents the time to take one hop in unloaded conditions. τ_s^i can then be used to update the value of T_{ns}^i.

$$T_{ns}^i = \gamma T_{ns}^i + (1-\gamma)\tau_s^i, \gamma \in [0,1] \tag{10}$$

where T_{ns}^i is the pheromone value in the routing table of node i for reaching s via the next hop node n. Once the FANT reaches the destination, it is

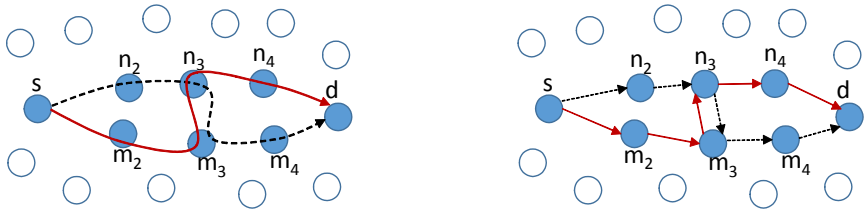


Fig. 2. Scenario 1 (a) reactive FANTS traveling (b) paths from s to d, which can result in looping of data packets between n_3 and m_3.

converted into a BANT and sent toward the source. Now, as described in Section 2.1, BANT moves from the destination toward the source via intermediate nodes, and a path is constructed from each intermediate node and source node to the destination. That is, as shown in Fig. 1(c), if BANT moves along nodes [n_{k-1}, n_{k-2}, n_{k-3},⋯, n_2] and reaches source node s=n_1 from destination d=n_k, each intermediate node [n_{k-1}, n_{k-2}, n_{k-3},⋯, n_2] and source node s will have a path towards the destination d, as shown in Fig. 1(d); that is, they will have an entry for d in their pheromone table T^i.

Ⅳ. Looping Issues and Solutions in AntHocNet

The use of multiple paths and stochastic data routing in AntHocNet helps in load balancing. However, these can also cause the looping of data packets. It is worth mentioning here that in this case, data packets do not always continue looping indefinitely. That is, a data packet might loop any number of times, and then exit of the loop. However, the possibility of indefinite looping also exists. These looping problems increase the end-to-end packet delay.

We describe the scenarios that can lead to the looping of data packets and explain how looping occurs therein. We also propose solutions to prevent such loops.

4.1 Scenario 1

Consider a network where a source node s launches a FANT (reactive or route repair FANT) toward a destination d. The ant is broadcast by the source node and it can also be broadcast by other nodes. Therefore, several replicas of the same ant, or in other words, several FANTs of the same generation, travel along various paths.

4.1.1 Single-hop loop

Fig. 2(a) shows the paths taken by two reactive FANTs (black and red ants) of the same generation. The red ant (solid line) traversed nodes [m_2, m_3, n_3, n_4], whereas the black ant (dashed line) traversed nodes [m_2, m_3, n_3, n_4] on their way from source s to destination d.

P_{red}: [m_2, m_3, n_3, n_4, d]
P_{black}: [n_2, n_3, m_3, m_4, d]

P_{red} and P_{black} denote the paths taken by the red and black ants, respectively. Nodes n_3 and m_3 are visited by both ants, but in the reverse order. The paths thus constructed are shown in Fig. 2(b).

Now we explain how looping of data packets can be caused in this situation. Source node s has two paths to destination d. Because, data packets are routed stochastically using (6), in AntHocNet, s can choose either n_2 or m_2 as the next hop. Assume that it chooses n_2 and forwards a data packet to it. n_2 only knows n_3 as the next hop to d, and thus it forwards the data packet to n_3. Now n_3 has two paths to d: through n_4 and m_3. Assume that n_3 chooses m_3 as the next hop and forwards the data packet to it. m_3 also has two paths to d: through m_4 and n_3. Assume that m_3 chooses n_3 as the next hop and thus the data packet arrives for the second time at n_3. This way, the data packet has looped once between n_3 and m_3. The same process can occur any number of times and thus the data packet can loop

any number of times between these two nodes. The data packet might eventually exit the loop if either n_3 chooses to forward it to n_4 or m_3 chooses to forward it to m_4. We call this a 'single-hop loop' because the data packet loops back and forth over one hop.

4.1.2 Multi-hop loop

Multi-hop loops of this type are also possible. Consider slightly different paths by introducing nodes *a* and *b* in the paths of the *red* and *black* ants, respectively.

P_{red}: [m_2, m_3, a, n_3, n_4, d]
P_{black}: [n_2, n_3, b, m_3, m_4, d]

Again, nodes n_3 and m_3 are visited by both ants and in reverse order, but here n_3 and m_3 are not neighbors, or more importantly, the visits to these two nodes by the two ants are not consecutive. In this case, data packets can continue looping through nodes [m_3, a, n_3, b]. Here, also, the packets can loop any number of times before exiting the loop. Because this involves more than one hop, we call this a 'multi-hop loop'.

4.1.3 Solution

Occurrence of the single-hop loop of scenario 1 can be eliminated as follows. Before forwarding a BANT to the next hop, each node checks its routing table to determine whether it has a path to the involved destination (BANT's source node) that passes through the BANT's next hop. If such a path exists, the node does not forward the BANT further. As a result, data packets do not encounter the single-hop looping of scenario 1.

For further elaboration, refer to Fig. 2. Suppose that the BANT that corresponds to the red ant has already returned and a path along P_{red} has already been setup. Now when the BANT that corresponds to the black ant is received by m_3, it finds that the next hop of the BANT is n_3. Because m_3 already has a path to d through n_3, m_3 will drop the BANT without forwarding it further. Fig. 3 shows the resultant paths.

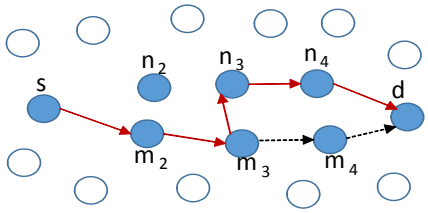


Fig. 3. Single-hop loop prevention of scenario 1.

In this case, unlike Fig. 2(b), single-hop looping of data packets is not possible. Source node *s* has a path to *d* through m_2 only. Node m_3 has two paths to *d*, but none can cause looping. As a result, data packets can reach the destination without encountering looping.

4.2 Scenario 2

This scenario involves some existing paths and new paths constructed by a proactive (or a route repair) ant.

4.2.1 Single-hop loop

Consider again a network where a source node *s* has already established some paths to a destination *d*. Assume, as shown in Fig. 4(a), that three such paths are:

P_1: [n_2, n_3, n_4, n_5, d]
P_2: [m_2, m_3, n_4, n_5, d]
P_3: [m_2, m_3, m_4, d]

Note that P_1 and P_2 merge at node n_4. Now, assume that a proactive FANT is travelling along P_1 and is broadcasted at n_4. Because m_3 is a neighbor of n_4, it picks up the broadcast and forwards this FANT to the next hop m_4 (along P_3). The proactive FANT travels along the rest of P_3 and reaches destination *d*. Now, if the corresponding BANT successfully arrives back at source *s*, we have the updated paths shown in Fig. 4(b).

In this scenario, looping of data packets between nodes m_3 and n_4 can occur. For example, if m_3 receives a packet destined for *d*, it finds that there are two paths to *d*: through m_4 and n_4. Because of stochastic data routing, m_3 might choose n_4 as the next hop. Now n_4 also has two paths to *d*: through n_5 and m_3. Again, because of stochastic data routing,

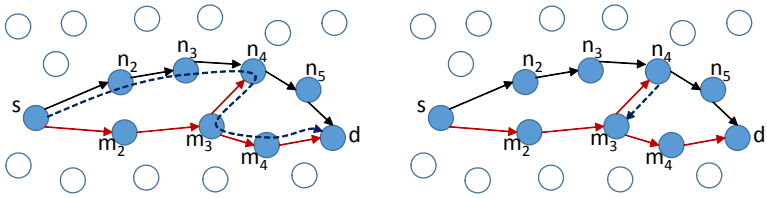


Fig. 4. Scenario 2 (a) proactive FANT traveling (b) updated paths from s to d, which can result in looping of data packets between m_3 and n_4.

n_4 might choose m_3 as the next hop. This way, the data packet traverses the loop once. Now the same process can continue unless either m_3 chooses to forward the data packet to m_4, or n_4 chooses to forward it to n_5.

4.2.2 Multi-hop loop

Consider the slightly different paths shown below:

P_1: [n_2, n_3, n_4, n_5, n_6, n_7, d]
P_2: [m_2, m_3, u, n_4, n_5, n_6, n_7, d]
P_3: [m_2, m_3, m_4, m_5, m_6, m_7, d]

Again, suppose that while data packets are in transit along P_1, the link between n_4 and n_5 breaks. Node n_4 attempts to locally repair the path, and therefore it broadcasts a route repair FANT.

Suppose that a node v receives this broadcast and rebroadcasts it further. This rebroadcast is received by m_3. Now, from this point forward, this FANT can travel along the rest of P_3 to the destination. As a result, P_1 has changed and appears as follows:

P_1:[n_2,n_3,n_4,v,m_3,m_4,m_5,m_6,m_7,d]

Now with P_3 and the changed P_1, it is possible for data packets to loop through nodes [m_3, u, n_4, v].

4.2.3 Solution

Occurrence of the single-hop loops of scenario 2 can be prevented. Before forwarding a route-repair (or proactive) FANT, each node checks whether it (the node) has a path to the involved destination that passes through the previous hop of the FANT; if so, the node drops the ant. As a result, data packets do not encounter the single-hop looping of scenario 2.

To elaborate this further, refer to Fig. 4(a). When node m_3 receives the proactive FANT broadcasted by n_4, it (m_3) finds that it already has path to d through n_4. Because n_4 is the previous hop of the FANT, m_3 drops the FANT. As result, the paths remain as before. Therefore, unlike Fig. 4(b), no data packet looping is possible between m_3 and n_4.

4.3 General solution for single-hop loop prevention

The solutions, presented in Sections 4.1.3 and 4.2.3 prevent single-hop looping by preventing multiple paths that can result in such looping. Each of these solutions only prevents single-hop loops of the particular scenario for which it was proposed. In this section, we propose a solution which can prevent single-hop loops of any type. Moreover, this solution does not impose any restriction on multiple paths; rather it prevents loops at the time of data packet forwarding.

When a node receives a data packet, the node records the previous hop, n_{ph}, of the packet. The node then stochastically determines the next hop, n_{nh}, for the packet from the routing table. If n_{nh} is the same as n_{ph}, the node can exclude n_{ph} as the next hop. The node then stochastically determines an alternate next hop. As a result, single-hop looping of data packets does not occur.

V. Performance Evaluation

To evaluate the effect of bidirectional path setup and loop prevention on performance, simulations were performed using the ns-2 simulator. For performance comparison, LP-AntHocNet refers to the implementation of AntHocNet where the

proposed single-hop loop prevention solution is incorporated. LP-BD-AntHocNet refers to the implementation of AntHocNet where the loop prevention solution and bidirectional path setup scheme are incorporated.

5.1 Simulation setup

A total of 50 nodes were randomly deployed in an area of 1500×300 m^2. For each scenario, ten simulations were run. The results were averaged and 95% confidence interval was computed. Each simulation was run for 600 *sec*. At the MAC layer, IEEE 802.11 Distributed Coordination Function (DCF) was used. The propagation model used was two-ray ground. The transmission range was assumed to be 250 *m*. Random waypoint mobility model was used.

In the random waypoint mobility model[17], initially, nodes are placed at random locations within the simulation area. Each node pauses at its current location for some time. When that time expires, the node chooses a random location and starts moving to that location with a speed uniformly distributed between *S_Min* and *S_Max*, where *S_Min* is the minimum speed limit and *S_Max* the maximum speed limit. Upon reaching the destination point, the node repeats the process; that is, pause at that point for some time and then choose another random destination point and start moving towards that point with a speed uniformly distributed between *S_Min* and *S_Max*.

Bidirectional communication sessions were used where five randomly chosen pairs of nodes communicated with each other. A communication session between a pair of nodes *A* and *B* is bidirectional when *A* sends data to *B* and *B* sends data to *A*. The traffic type used was Constant-Bit-Rate (CBR). We did not use TCP, although it seems appealing for bidirectional sessions, because its flow control and congestion control features can have indirect influence on the performance of the routing protocol.

5.2 Simulation results

In Section 4, we mentioned single-hop loops, which involve two nodes. We extend that definition to n-hop loops. Therefore, the loop length is *n*-hops if it involves *n+1* nodes.

Table 1 lists the percentage of packets that has experienced some type of loop. As indicated in Table 1, approximately 10% of the data packets encounter some type of looping. Approximately 7.5% encounter single-hop looping whereas approximately 2.1% encounter multi-hop looping. Approximately 0.4% of the packets encounter both single- and multi-hop loops. This means that among those packets that encounter looping, approximately 79% encounter single-hop looping. Therefore, our proposed single-hop loop prevention scheme is highly beneficial.

As shown in Fig. 5, packets encounter single-hop loops much more frequently. As the loop length increases, its ratio of occurrence decreases. The figure shows results for loops of up to six-hops long. Loops of even higher length (up to 15 hops) also occur, but at a much smaller ratio. Longer loops are more likely to cause the Time-to-live (TTL) field of a packet to expire, and as a result of

Table 1. Percentage of packets experiencing loops.

Type of loops	Percentage (%)
Packets without looping	90
Packets with only single-hop looping	7.5
Packets with only multi-hop looping	2.1
Packets with both single- and multi-hop looping	0.4

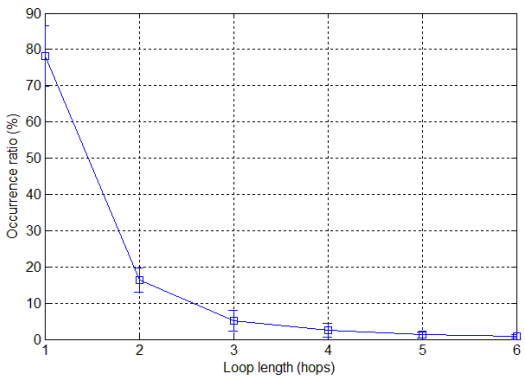


Fig. 5. Ratio of loop occurrence according to loop length.

which the packet is dropped.

Fig. 6 shows packet ratios according to loop count. Loop count represents the number of times a packet loops. As this figure shows, as the loop count increases, the packet ratio decreases. The largest ratio of packets loops only once. A packet that experiences a smaller loop count can still reach the destination, but with a slightly larger delay. As the loop count increases, it can either cause the packet to be dropped because of the of the TTL field expiring, or it can cause a long end-to-end delay if the packet ultimately reaches the destination.

Fig. 7 compares the bidirectional path setup delay of AntHocNet and BD-AntHocNet. The path setup delay for BD-AntHocNet is approximately half of that of AntHocNet. This is because BD-AntHocNet sets up forward and backward paths simultaneously, whereas AntHocNet has to setup forward and backward paths separately. Consequently, in order to set up bidirectional paths, AntHocNet requires twice as much time as BD-AntHocNet.

For both schemes, path availability delay tends to increase with increasing pause time. This can be explained by a characteristic of the random waypoint mobility model. As found in [19], in this model, nodes tend to concentrate in the center of the deployment area. This tendency decreases as the pause time increases[20]. Higher concentration in the middle means fewer hops between source and destination, and therefore shorter path setup time and smaller path availability delay.

Fig. 8 compares the end-to-end delay of AntHocNet and LP-AntHocNet. The end-to-end delay for both schemes decreases with increasing pause time. A common reason for this behavior is that at higher pause times, the network is less dynamic, therefore there are fewer path breakages. Fewer path breakages results in fewer packets waiting in queue for route repair. As a result the average end-to-end delay is reduced.

Data packet looping increases the end-to-end delay of data packets. From Fig. 8, it is clear that loop prevention can significantly reduce end-to-end delay, especially for smaller pause times. The performance difference between AntHocNet and LP-AntHocNet is higher, with 79% improvement for the smallest pause time, and it gradually diminishes to 56% as the pause time increases.

Fig. 9 shows a comparison of the normalized overhead. For all three schemes, the overhead drops with increasing pause time. At higher pause times,

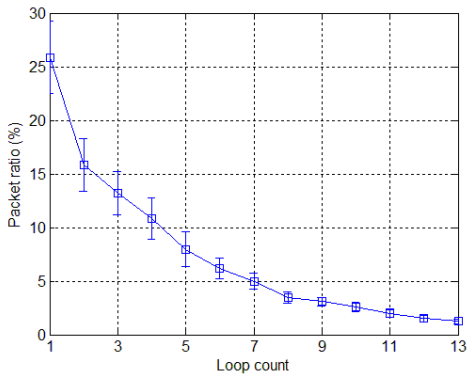


Fig. 6. Packet ratio according to loop count.

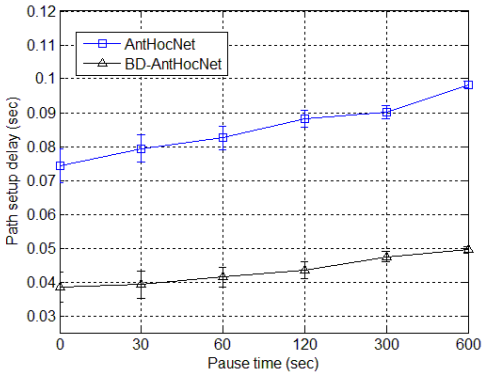


Fig. 7. Comparison of path setup delay.

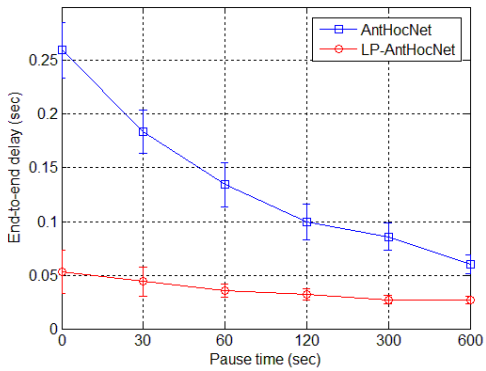


Fig. 8. Comparison of end-to-end delay.

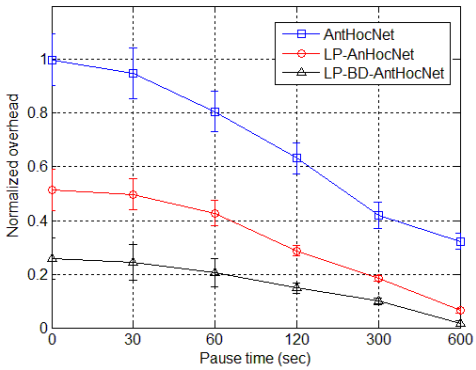


Fig. 9. Comparison of normalized overhead.

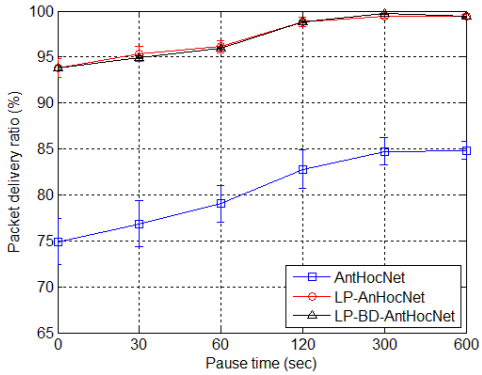


Fig. 10. Comparison of packet delivery ratio.

the network is less dynamic, and therefore, there are fewer path breakages, which means overhead caused by route repair ants is also reduced. Similarly to data packets, an ant can go into a loop and continue looping before exiting and moving toward the destination. This causes additional overhead. Therefore, loop prevention results in considerable overhead improvement. Use of the bidirectional path setup scheme-LP-BD-AntHocNet-results in further overhead reduction. This is as expected because in LP-BD-AntHocNet, the constructed backward paths often eliminate the need for reactive path setup. It should be noted that in LP-BD-AntHocNet, overhead reduction is mainly achieved for two reasons: 1) a reduction in the number of times path setup is required, and 2) a reduction in the number of times a route repair is required. The effect of the latter diminishes as the pause time increases, and consequently, the performance difference between LP-AntHocNet and LP-BD-AntHocNet narrows.

Fig. 10 shows a comparison of packet delivery ratios. Such ratio is the percentage of sent data packets successfully received at the destination. LP-AntHocNet and LP-BD-AntHocNet perform nearly identically and much better than AntHocNet. Again, it is evident that loop prevention results in significant improvement in terms of delivery ratio.

For all three schemes, the delivery ratio increases as the pause time increases. This is because at longer pause times, the network is less dynamic and there are fewer path breakages. It is interesting to note that the delivery ratio for AntHocNet, reported in [9], drops as the pause time increases. The authors explain that due to sparseness of their network, the pause time affects connectivity in such a way that at high pause times, some nodes remain disconnected from the rest of the network for a long time and thus no packets can be delivered to them.

Ⅵ. Conclusion

In this paper, two major shortcomings of AntHocNet, unidirectional paths and looping issues, were identified. A bidirectional path construction scheme was presented. The simulation results showed that for a similar delivery ratio, the proposed scheme provides considerable performance improvement in terms of end-to-end delay and overhead. Furthermore, solutions to the looping issues were proposed and evaluated. The simulation results indicated that looping was a serious problem in AntHocNet and degraded the performance significantly. The results also showed that single-hop looping was by far the most frequently occurring looping type, and approximately 7.9% of all data packets (or approximately 79% of looping packets) encounter them. The results demonstrated that single-hop loop prevention significantly reduces end-to-end delay and overhead, and improves the delivery ratio.

References

[1] A. Boukerche, B. Turgut, N. Aydin, M. Z.

Ahmad, L. Bőlőni, and D. Turgut, “Routing protocols in ad hoc networks: A survey,” *Comput. Netw.*, vol. 55, pp. 3032-3080, 2011.

[2] Z. Chenyu and D. C. Sicker, “A survey on biologically inspired algorithms for computer networking,” *IEEE Commun. Surv. & Tuts.*, vol. 15, pp. 1160-1191, 2013.

[3] S. Bitam, A. Mellouk, and S. Zeadally, “Bio-inspired routing algorithms survey for vehicular Ad Hoc networks,” *IEEE Commun. Surv. & Tuts.*, vol. 17, pp. 843-867, 2015.

[4] F. Dressler and O. B. Akan, “Bio-inspired networking: from theory to practice,” *IEEE Commun. Mag.*, vol. 48, pp. 176-183, 2010.

[5] F. Dressler and O. B. Akan, “A survey on bio-inspired networking,” *Comput. Netw.*, vol. 54, pp. 881-900, 2010.

[6] M. Roth and S. Wicker, “Termite: ad-hoc networking with stigmergy,” in *Proc. IEEE GLOBECOM*, Dec. 2003.

[7] H. F. Wedde, M. Farooq, T. Pannenbaecker, B. Vogel, C. Mueller, J. Meth, and R. Jeruschkat, “BeeAdHoc: an energy efficient routing algorithm for mobile ad hoc networks inspired by bee behavior,” in *Proc. 7th Annu. Conf. Genetic and Evolutionary Computation*, pp. 153-160, Washington DC, USA, Jun. 2005.

[8] M. Gunes, U. Sorges, and I. Bouazizi, “ARA-the ant-colony based routing algorithm for MANETs,” in *Int. Conf. Parall. Process.*, pp. 79-85, 2002.

[9] G. Di Caro, F. Ducatelle, and L. M. Gambardella, “AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks,” *Eur. Trans. Emerging Telecommun. Technol.*, vol. 16, no. 5, pp. 443-455, Oct. 2005.

[10] G. D. Caro, F. Ducatelle, and L. M. Gambardella, *AntHocNet: An adaptive nature-inspired algorithm for routing in mobile Ad Hoc networks*, in Technical Report, No. IDSIA-27-04-2004, 2004.

[11] M. Dorigo, G. D. Caro, and L. M. Gambardella, “Ant algorithms for discrete optimization,” *Artif. Life*, vol. 5, pp. 137-172, 1999.

[12] K. M. Sim and S. Weng Hong, “Ant colony optimization for routing and load-balancing: survey and new directions,” *IEEE Trans. Syst., Man, and Cybernetics - Part A: Syst. and Humans*, vol. 33, pp. 560-572, 2003.

[13] J. S. Baras and H. Mehta, “A probabilistic emergent routing algorithm for mobile Ad Hoc networks,” in *Proc. WiOpt03: Modeling and Optimization in Mob. Ad Hoc and Wirel. Netw.*, Sophia-Antipolis, France, Mar. 2003.

[14] K. M. Sim and W. H. Sun, “Multiple ant-colony optimization for network routing,” in *Proc. First Int. Symp. CyberWorld*, pp. 277-281, Nov. 2002.

[15] G. D. Caro and M. Dorigo, “AntNet: Distributed stigmergetic control for communications networks,” *JAIR*, vol. 9, pp. 317-365, 1998.

[16] C. E. Perkins and E. M. Royer, “Ad-hoc on-demand distance vector routing,” in *Proc. 2nd IEEE Workshop on Mob. Comput. Syst. and Appl.*, p. 90, Feb. 1999.

[17] D. B. Johnson and D. A. Maltz, “Dynamic source routing in Ad Hoc wireless networks,” in *Mob. Comput.*, vol. 353, pp. 153-181, 1996.

[18] R. R. Stewart, *Stream Control Transmission Protocol*, in RFC 4960, 2007.

[19] C. Bettstetter and O. Krause, “On border effects in modeling and simulation of wireless ad hoc networks,” in *Proc. IEEE MWCN*, 2001.

[20] D. M. Blough, G. Resta, and P. Santi, “A statistical analysis of the long-run node spatial distribution in mobile ad hoc networks,” in *Proc. MSWiM’02*, pp. 30-37, Atlanta, Georgia, USA, Sept. 2002.

라프만 샴스 우르 (Shams ur Rahman)

2007년 : 페샤와르 공학 기술 대학교 졸업
2012년 : GIK 과학기술연구원 석사
2012년~2014년 : GIK 과학기술연구원, 연구원
2014년~현재 : 경북대학교 전자공학부 박사과정
<관심분야> 집단생태특성 모방형 네트워크, 지연내성망, 무선 애드혹 네트워크

아즈말 칸 (Ajmal Khan)

2007년 : 페샤와르 공학 기술 대학교 졸업
2010년 : 페샤와르 공학 기술 대학교 석사
2014년 : 경북대학교 전자공학부 박사
2014년~2016년 : IQRA 국립대학교 조교수
2016년~현재 : COMSATS 정보기술연구원 조교수
<관심분야> 차량 애드혹 네트워크, 지연내성망, P2P 네트워크

남 재 충 (Jae-Choong Nam)

2010년 : 경북대학교 전자전기컴퓨터학부 졸업
2013년 : 경북대학교 전자 공학부 석사
2013년~현재 : 경북대학교 전자공학부 박사과정
<관심분야> 차세대 이동 네트워크, 무선 애드혹 네트워크, 드론 ICT 융합 기술

조 유 제 (You-Ze Cho)

1982년 : 서울대학교 전자공학과 졸업
1983년 : 한국과학기술원 전기전자공학 석사
1988년 : 한국과학기술원 전기전자공학 박사
1989년~현재 : 경북대학교 전자공학부 교수
1992년~1994년 : Univ. of Toronto, 방문교수
2002년~2003년 : 미국 국립표준연구소(NIST), 객원연구원
<관심분야> 드론 ICT 융합 기술, 차세대 이동 네트워크, 무선 애드혹 네트워크, 이동성 관리 기술, Future Internet