Self-stabilizing DAG-constructing Protocols with Application to Geocast in MANET

Koichi Ito[†,‡], Yoshiaki Katayama[†] Koichi Wada[§], Naohisa Takahashi[†]

October 1, 2013

[†] Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya, Aichi, Japan
[‡] ANDEN, Co., LTD., Japan
[§] Hosei University, 3-7-2 Kajino-cho, Koganei, Tokyo, Japan

Abstract

Constructing Directed Acyclic Graph (DAG) on a graph is one of the fundamental problems of graph theory. We propose a self-stabilizing protocol for DAG construction on mobile ad-hoc network (MANET) and show that it can construct a DAG which has better property than known one. Furthermore, we evaluate our protocol by applying it to geocast protocol on MANET. Our computer simulation results show that our proposed geocast protocol is better than known geocast protocol GeoTORA which also uses a DAG.
keywords: *Self-stabilizing, DAG, Geocast, MANET*

1 Introduction

A distributed protocol is a protocol designed to run on a network (distributed system) that consists of processes communicating with each other through communication links. A *self-stabilizing protocol* is a distributed protocol that achieves its intended behavior regardless of the initial configuration of a network. Thus, a self-stabilizing protocol is resilient to any number and any type of transient faults: after the last transient fault occurs, the protocol starts to converge to its intended behavior. Furthermore, if topology of the network changes the protocol starts to converge to its intended behavior. Hence, a self-stabilizing protocol also tolerates to topology change of the network.

A *directed acyclic graph* (DAG) is a directed graph with no directed cycle. A node without any outgoing edges is called a *sink*. Therefore, there is at least one sink in any DAG. Maintaining a DAG is required to solve various problems, e.g. leader election [1], coloring [2], routing [3, 4] and so on. For example, a leader election protocol proposed in [1] constructs a DAG such that there exists just one sink in the connected network, and the sink elects itself a leader. TORA [3] implements a unicast routing in a mobile ad hoc network (MANET) by using a DAG which has one sink to deliver packets to a destination node (a sink node in the DAG). GeoTORA [4], which is one of the protocols for geocasting in MANET, also uses the DAG-based approach.

Geocasting has been proposed as a mechanism to deliver messages to interest nodes within a given geographical region [5]. In geocasting, a node automatically joins a set of nodes that can receive geocast messages (we call this set a *geocast group*) if the node is in the region specified for the geocasting (we call this region a *geocast region*) while a node becomes a member of the multicast group by explicitly joining the group in a multicast protocol. GeoTORA [4] implements the geocasting by the following steps. First, GeoTORA floods control packets in the network. If some node in a geocast group receives the control packet, it begins to construct a DAG whose sink is itself. After constructing the DAG in the network, geocast messages are transmitted along the DAG edges from a source node to members of the geocast group.

In the use of a DAG for geocasting in MANET, it is important to construct a DAG that has high robustness or high accuracy of geocast message delivery. To maintain a DAG adapting to the network topology change, we propose a self-stabilizing DAG constructing protocol. To construct paths from any node to a node in the geocast region, we need to construct a DAG such that each node in a given set of nodes becomes a sink. A self-stabilizing protocol for this DAG constructing problem has been proposed in [6]. This protocol takes a simple approach which uses identifiers of nodes and its distance from the nearest sink. So, we call this protocol $\mathcal{DAG}_{NO}$ (Nearest sink Oriented). To enhance the accuracy of the

geocast delivery, we propose a new self-stabilizing DAG constructing protocol, which uses distances from all sinks and is called $\mathcal{DAG}_{AO}$ (All sinks Oriented). Then, we show that our protocol $\mathcal{DAG}_{AO}$ has better property than $\mathcal{DAG}_{NO}$ by computer simulations.

Furthermore, We perform some computer simulations to evaluate the performance of the geocast protocols that use $\mathcal{DAG}_{NO}$ and $\mathcal{DAG}_{AO}$ for constructing a DAG, and GeoTORA. The simulation results show that our geocast protocols can deliver a geocast message to members of the geocast group with high accuracy although the communication overhead becomes slightly large.

The outline of this paper is as follows. The model and definitions are presented in Section 2. In Section 3 we describe protocol $\mathcal{DAG}_{NO}$ proposed in [6]. In Section 4 we present $\mathcal{DAG}_{AO}$. We compare properties of DAGs constructed by $\mathcal{DAG}_{NO}$ and $\mathcal{DAG}_{AO}$ in Section 5. In Section 6 we describe implementation of our geocast protocols on practical networks and outline of GeoTORA. In Section 7 we discuss the simulation results for performance evaluation of geocast protocols. Finally, we conclude in Section 8.

2 The System model and Assumption

2.1 Graph

An (undirected) graph G is denoted by $G = (V, E)$, where $V = \{v_0, v_1, \cdots, v_{n-1}\}$ is a set of nodes and E is a set of edges. When there is an edge between v_i and v_j, we denote $(v_i, v_j) \in E$. The sequence of nodes $v_0, v_1, \cdots, v_k$ is called a v_0-v_k path if $\forall i, 0 \leq i \leq k-1, (v_i, v_{i+1}) \in E$. The length of v_0-v_k path is k. The *distance* of v_i and v_j is the minimum length over p_i-p_j paths, and denoted by $dist(p_i, p_j)$. $e(v)$ is defined as $e(v) = \max\{dist(v, u) | u \in V\}$.

A directed graph $\overrightarrow{G}$ is denoted by $\overrightarrow{G} = (V, \overrightarrow{E})$, where $V = \{v_0, v_1, \cdots, v_{n-1}\}$ is a set of nodes and $\overrightarrow{E}$ is a set of directed edges. When there is a directed edge from v_i to v_j, we denote $\overrightarrow{(v_i, v_j)} \in \overrightarrow{E}$. If $\overrightarrow{(v_i, v_j)} \in \overrightarrow{E}$, we say v_i has an outgoing edge to v_j and v_j has an incoming edge from v_i. If a directed graph $\overrightarrow{G}$ has no cycle, $\overrightarrow{G}$ is called a *directed acyclic graph* (DAG). A node with only incoming edges is called a *sink*.

Definition 1 (DAG constructing problem)**.** For a given connected graph $G = (V, E)$ and a given set of nodes $T \subseteq V$, we give a direction to each edge in $E' = E - \{(u, v) | u, v \in T\}$ and construct a DAG $\overrightarrow{G} = (V, \overrightarrow{E}) = \{\overrightarrow{e} | \overrightarrow{e} = \overrightarrow{(u, v)} \text{ or } \overrightarrow{(v, u)} \text{ for each } (u, v) \in E'\})$ such that each node in T is a sink. □

We show an example of a DAG which satisfies this definition in Fig. 1.

2.2 Distributed System

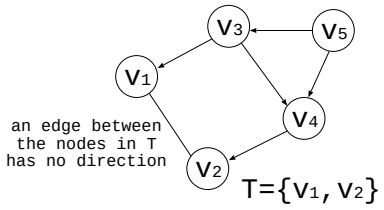


Figure 1: An example of the DAG defined in Definition 1

A distributed system consists of n communicating processes. We model its communication network by an undirected graph $N = (\mathcal{P}, \mathcal{L})$, where $\mathcal{P}$ and $\mathcal{L}$ respectively represent the sets of the processes and the bidirectional communication links. Note that we can consider a network as a graph, hence we may use "nodes" and "edges", and "processes" and "links" interchangeably. Each process has a *totally ordered identifier*. Simply, the identifier of process P_i is denoted by P_i. When a link connects processes P_i and P_j, this link is denoted by (P_i, P_j). If $(P_i, P_j) \in \mathcal{L}$, we say P_j is a *neighbor* of P_i and vice versa, and the set of neighbors of P_i is denoted by N_i. We assume each process knows its own and neighbors' identifiers. Communication is carried out by means of *shared variables*. Each process holds a set of shared variables and can access them to read and write. Each process can also access its neighbors' variables to read.

A *state* of a process is defined by values of its variables. A *configuration* of a network is specified by an n-tuple $c = (q_0, q_1, \cdots, q_{n-1})$ where q_i stands for the state of P_i. Let c_i and c_{i+1} be configurations and $\mathcal{A}$ be a distributed protocol. A *computation* of a protocol $\mathcal{A}$ is a maximal (possibly infinite) sequence of configuration $E = c_1, c_2. \cdots$, when c_i changes to c_{i+1} by executing protocol $\mathcal{A}$ of any set of processes.

We say that process P_i is *enabled* in a configuration c if P_i can change its state by executing its protocol $\mathcal{A}$ in c. We assume a *strongly fair distributed daemon* as a scheduler [7]. The distributed daemon is a scheduler such that if one or more processes are enabled during an execution of a protocol

then the daemon chooses at least one (possibly more) of these enabled processes to execute the protocol. A daemon is strongly fair if any process that is enabled infinitely often is eventually chosen for execution.

In order to compute the time complexity of a computation, we use a *round*. The first round in an computation E is the shortest prefix E' of E such that each process executes at least one step in E'. Let E'' be the suffix of E that follows E', $E = E'E''$. The second round of E is the first round of E'', and so on.

2.3 Self-stabilizing

Let $\mathcal{LE}$ be any set of configurations. Protocol $\mathcal{A}$ is said to be self-stabilizing for $\mathcal{LE}$, iff the following two conditions hold:

1. convergence: the protocol eventually reaches a configuration in $\mathcal{LE}$.
2. closure: every process never changes its variables at a configuration in $\mathcal{LE}$.

When protocol $\mathcal{A}$ is self-stabilizing for $\mathcal{LE}$, $\mathcal{LE}$ is called a set of legitimate configurations.

3 Protocol $\mathcal{DAG}_{NO}$ [6]

In this section, we describe a self-stabilizing DAG constructing protocol for a given graph $G = (V, E)$ and a given set of nodes $T(\subseteq V)$, which is proposed in [6]. Each node P_i has constant t_i such that if $P_i \in T$ then t_i is set to $true$ and otherwise t_i is set to $false$, automatically.

$\mathcal{DAG}_{NO}$ assumes that, on each node P_i, there exists an underlying topology maintenance scheme that stores identifiers of the neighbors in N_i. Each node P_i refers N_i and accesses the neighbors' variables, then P_i changes its own variables, if possible, according to the protocol.

Every node P_i maintains the following two variables:

- d_i: d_i denotes $\min\{dist(P_i, P_j)|P_j \in T\}$.
- Out_i: $Out_i \subseteq N_i$. A set of nodes connected by outgoing edges from P_i.

We can construct a DAG simply to use the totally ordered identifier, e.g. if $P_j < P_i$ then the edge between P_i and P_j is directed from P_i to P_j. However, this DAG may not satisfy Definition 1. Since every node in T should be a sink, each node P_i maintains d_i that denotes $\min\{dist(P_i, P_t)|P_t \in T\}$. Then, the direction of each edge is determined by a lexicographic order of the distance and the identifier, i.e. if $d_j < d_i$ or $(d_j = d_i) \wedge (P_j < P_i)$ then the edge is directed from P_i to P_j and $P_j \in Out_i$. And all edges between nodes in T have no direction. Notice that Out_i is not essential for construction of a DAG, which is used by another protocol that uses a DAG. To compute the minimum distance from T, each node P_i takes the following strategies:

- When $P_i \in T$ then d_i is set to 0.
- When $P_i \notin T$ then d_i is set to the minimum distance that the neighbors have plus 1.

We show protocol $\mathcal{DAG}_{NO}$ in Fig. 2.

Then, Miura et al. showed the following theorem.

Theorem 1. $\mathcal{DAG}_{NO}$ constructs a DAG $\overrightarrow{G} = (V, \overrightarrow{E})$ for $T \subseteq V$, satisfying the following conditions:

- $\forall P_i \in T, d_i = 0$
- $\forall P_i \notin T, d_i = \min\{dist(P_i, P_t)|P_t \in T\}$
- $\overrightarrow{(P_i, P_j)} \in \overrightarrow{E}$ iff $(P_i \notin T) \wedge (d_j < d_i \vee (d_j = d_i) \wedge (P_j < P_i)) \wedge ((P_i, P_j) \in E)$

□

1: **if** $t_i \wedge (d_i \neq 0)$ **then**
2: $\quad d_i := 0$
3: **else if** $\neg t_i \wedge (d_i \neq \min\{d_j | P_j \in N_i\} + 1)$ **then**
4: $\quad d_i := \min\{d_j | P_j \in N_i\} + 1$
5: **end if**
6: **if** t_i **then**
7: $\quad Out_i := \phi$
8: **else**
9: $\quad Out_i := \{P_j | \forall P_j \in N_i, (d_j < d_i) \vee ((d_j = d_i) \wedge (P_j < P_i))\}$
10: **end if**

Figure 2: Protocol $\mathcal{DAG}_{NO}$ on node P_i

4 Protocol $\mathcal{DAG}_{AO}$

In this section, we describe protocol $\mathcal{DAG}_{AO}$, which is a self-stabilizing DAG constructing protocol for a given graph $G = (V, E)$ and a given set of nodes $T(\subseteq V)$. Then, we prove the correctness of the protocol.

In $\mathcal{DAG}_{NO}$, the direction of an edge between nodes that are equidistance apart from the nearest sink is determined by identifier of them, regardless of the distance from other sinks. We consider that to determine the direction of such edges by using distances from the other sinks, there is the potential for an increase of the number of the reachable sinks from any node, which leads to an enhancement of the accuracy of a geocast message delivery.

4.1 Description of the Protocol

Each node P_i has constant t_i such that if $P_i \in T$ then t_i is set to *true* and otherwise t_i is set to $false$, automatically.

$\mathcal{DAG}_{AO}$ assumes that, on each node P_i, there exists an underlying topology maintenance scheme that stores identifiers of the neighbors in N_i. Each node P_i refers N_i and accesses the neighbors' variables, then P_i changes its own variables, if possible, according to the protocol.

Every node P_i maintains the following one constant and two variables:

- **N**: A constant which denotes an upper bound of the number of nodes.
- L_i: $L_i = \{(P_k, l_k) | P_k \in \mathcal{P}, l_k = dist(P_i, P_k)\}$. We assume that we can access to l_k by $L_i[P_k]$. Notice that if P_i does not receive any information about $dist(P_i, P_k)$, (P_k, l_k) is not included in L_i. At this time, for simplicity, let $L_i[P_k]$ be ∞.
- Out_i: $Out_i \subseteq N_i$. A set of nodes which are connected by outgoing edges from P_i.

In $\mathcal{DAG}_{NO}$, each node P_i maintains the distance from the nearest sink, i.e. $min\{dist(P_i, P_t) | P_t \in T\}$. On the other hand, in $\mathcal{DAG}_{AO}$ each node P_i maintains every distance from every sink. The distance from each sink to P_i is stored in L_i. If a node P_i is in T, P_i maintains $(P_i, 0)$, which denotes the distance from P_i to P_i, in L_i. Each node P_i records the minimum distance that the neighbors have plus 1. To delete distance information for the nodes that are not in T, we use an upper bound **N** of the number of nodes.

The direction of an edge connecting a pair of nodes P_i and P_j is defined as from P_i to P_j, i.e. $P_j \in Out_i$, iff $(L_j <_d L_i) \vee ((L_j =_d L_i) \wedge (P_j < P_i))$. The binary relation $L_i <_d L_j$ and $L_i =_d L_j$ is defined as follows:

Definition 2. Let P_i and P_j be any nodes. Let $L'_i = \{a_0, a_1, \cdots, a_m\}$ be an ordered set such that the set $\{a_t | a_t \in L_i\}$ is sorted in ascending order with lexicographical manner. Let $L'_j = \{b_0, b_1, \cdots, b_n\}$ be defined by the same way. Let $k = \min(m, n)$.

- $L_i <_d L_j$ iff $(a_0 < b_0) \vee (\exists s, 0 < s \leq k, \forall u, 0 \leq u < s, (a_u = b_u) \wedge (a_s < b_s))$
- $L_i =_d L_j$ iff $\forall s, 0 \leq s \leq k, a_s = b_s$

□

We show protocol $\mathcal{DAG}_{AO}$ in Fig. 3.

1: $L_{tmp} := \phi$
2: **for all** $P_j \in N_i$ **do**
3: **for all** $(P_k, l_k) \in L_j$ **do**
4: **if** $(P_k \neq P_i) \wedge (l_k < \mathbf{N} - 1)$ **then**
5: **if** $L_{tmp}[P_k] = \infty$ **then**
6: $L_{tmp} := L_{tmp} \cup \{(P_k, l_k + 1)\}$
7: **else if** $l_k + 1 < L_{tmp}[P_k]$ **then**
8: $L_{tmp}[P_k] := l_k + 1$
9: **end if**
10: **end if**
11: **end for**
12: **end for**
13: $L_i := L_{tmp}$
14: **if** t_i **then**
15: $L_i := L_i \cup \{(P_i, 0)\}$
16: $Out_i := \phi$
17: **else**
18: $Out_i := \{P_j | P_j \in N_i, (L_j <_d L_i) \vee ((L_j =_d L_i) \wedge (P_j < P_i))\}$
19: **end if**

Figure 3: Protocol $\mathcal{DAG}_{AO}$ on node P_i

4.2 Correctness Proof

First, we define the legitimate configuration.

Definition 3 (Legitimate Configuration)**.** A configuration $\mathcal{LE}_{AO}$ of protocol $\mathcal{DAG}_{AO}$ is legitimate if every node P_i satisfies the following two conditions.

1. $L_i = \{(P_j, dist(P_j, P_i)) | \forall P_j \in T)\}$
2. $(t_i \wedge Out_i = \phi) \vee (\neg t_i \wedge Out_i = \{P_k | P_k \in N_i, (L_k <_d L_i) \wedge ((L_k =_d L_i) \vee (P_k < P_i))\})$

□

$\mathcal{DAG}_{AO}$ updates L_i in lines 1-13 of Fig. 3. Then, focus on the distance from a node P_k ($P_k \neq P_i$): whether an information about a distance from P_k to P_i is in L_i and the value of the distance from P_k to P_i depends on the information about the distance from P_k to a neighbor of P_i, and the following property is satisfied.

Property 1. Let P_i and P_k ($\neq P_i$) be any nodes. If P_i executes $\mathcal{DAG}_{AO}$ at least once, the following condition holds.

$$L_i[P_k] = \begin{cases} \infty & (\forall P_j \in N_i, \\ & \quad L_j[P_k] = \mathbf{N} - 1 \vee L_j[P_k] = \infty) \\ \min\{L_j[P_k] | P_j \in N_i\} + 1 & (otherwise) \end{cases}$$

□

Then, we show that protocol $\mathcal{DAG}_{AO}$ is self-stabilizing for $\mathcal{LE}_{AO}$.

Lemma 1. Let P_t in T. After at most k-th round, the following two conditions hold.

- $\forall P_i, dist(P_i, P_t) \leq k \rightarrow L_i[P_t] = dist(P_i, P_t)$
- $\forall P_j, dist(P_j, P_t) > k \rightarrow L_j[P_t] > k$

Proof. We prove by induction on the number of rounds.

After 0th round, $L_t[P_t] = 0$ holds and remains the same afterward. Let P_i be any node such that $dist(P_i, P_t) > 0$ ($P_i \neq P_t$). Since $\min\{L_j[P_t] | P_j \in N_i\} \geq 0$ holds, then $L_i[P_t] > 0$ holds.

Next, we assume that the following conditions hold after m-th round.

- $\forall P_i, dist(P_i, P_t) = m \rightarrow L_i[P_t] = dist(P_i, P_t)$
- $L_i[P_t]$ remains unchanged afterward
- $\forall P_j, dist(P_j, P_t) > m \rightarrow L_j[P_t] > m$

After $(m+1)$-th round, let P_i be any node such that $dist(P_i, P_t) = m+1$. P_i has at least one neighbor P_j such that $dist(P_j, P_t) = m$. From the assumption of the induction, any node P_k, with $L_k[P_t] < m$, is not a neighbor of P_i. Then $L_i[P_t] = m + 1$ holds. In addition, since $L_j[P_t]$ remains unchanged, $L_i[P_t]$ also remains unchanged afterward. Let P_l be any node such that $dist(P_l, P_t) > m + 1$. Since $\min\{dist(P_q, P_t)|P_q \in N_l\} \geq m + 1$ and the assumption of induction, $L_l[P_t] > m + 1$ holds. □

Lemma 2. Let P_i be any node not in T and c be any configuration. If $\min\{L_j[P_i]|P_j \in V\} = l(l \neq \infty)$ holds in c, $\min\{L_j[P_i]|P_j \in V\} \geq l + 1$ or $\forall P_j \in V, L_j[P_i] = \infty$ holds after one round starting from c.

Proof. From Property 1, if $l < \mathbf{N} - 1$ holds then $min\{L_j[P_i]|P_j \in V\} = l + 1$ holds after one round starting from c. And if $l \geq \mathbf{N} - 1$ holds then $\forall P_j \in V, L_j[P_i] = \infty$ holds. □

Lemma 3. Let P_i be any node not in T. After at least $\mathbf{N}$-th round, $\forall P_j \in V, L_j[P_i] = \infty$ holds.

Proof. Follows from Property 1 and Lemma 2. □

Lemma 4 (Self-Stabilizing)**.** Protocol $\mathcal{DAG}_{AO}$ is self-stabilizing for $\mathcal{LE}_{AO}$.

Proof. Let P_t be any node in T. Let $e(T) = \max\{e(P_t)|P_t \in T\}$. By Lemma 1, after at least $e(T) + 1$ round, for each node P_i, $L_i[P_t] = dist(P_i, P_t)$ holds. By Lemma 3, after at least $\mathbf{N}$-th round, for each node P_i and each node $P_j \notin T$, $L_i[P_j] = \infty$ holds. Then, after at least $\mathbf{N}$-th round, each node satisfies condition 1 in Definition 3.

It is clear that if each node P_i executes the protocol once P_i satisfies condition 2 in Definition 3.

Thus, the proposed protocol $\mathcal{DAG}_{AO}$ satisfies the convergence condition, and it is obvious that $\mathcal{DAG}_{AO}$ also satisfies the closure condition. □

Second, we show that a DAG defined on Definition 1 is constructed in any legitimate configuration in $\mathcal{LE}_{AO}$. The following two lemmas are obvious.

Lemma 5. Let P_i and P_j be any nodes. Let $P_i < P_j$ be defined as $(L_i <_d L_j) \vee ((L_i =_d L_j) \wedge (P_i < P_j))$. $P_i < P_j$ is a totally ordered relation.

Lemma 6. Let $G = (V, E)$ be any graph such that V is a totally ordered set. The directed graph $\overrightarrow{G} = (V, \overrightarrow{E})$ given by setting direction to each edge in E by order of connected nodes, from the higher node to the lower node, becomes a DAG.

Lemma 7. For any legitimate configuration in $\mathcal{LE}_{AO}$, a DAG defined in Definition 1 is constructed on the graph.

Proof. By Lemmas 5 and 6, the directed graph constructed by $\mathcal{DAG}_{AO}$ becomes a DAG.

In legitimate configuration $\mathcal{LE}_{AO}$, it is obvious from the protocol that each node in T becomes a sink.

Let P_i be any node not in T. And let P_j be any node in T such that $dist(P_i, P_j) = \min\{dist(P_i, P_t)|P_t \in T\}$. Consider the shortest path from P_i to P_j. This path must contain a node P_k such that P_k is neighbor of P_i and $L_i[P_t] > L_k[P_t]$ holds. Then, P_i must have an outgoing edge to P_j. So, any node which is not belong to T does not become a sink. □

We obtain the following theorem from Lemma 4 and Lemma 7.

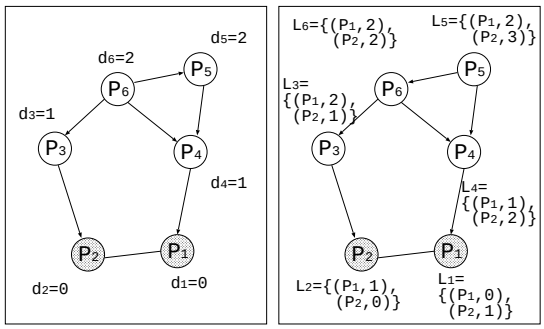
Theorem 2. Protocol $\mathcal{DAG}_{AO}$ is a self-stabilizing DAG constructing protocol. □

5 Comparison of $\mathcal{DAG}_{NO}$ and $\mathcal{DAG}_{AO}$

Protocols $\mathcal{DAG}_{NO}$ and $\mathcal{DAG}_{AO}$ are designed to maintain DAGs defined in Definition 1. If these two protocols work on the same topology of the network, there may be an edge that is determined different direction between two protocols. This difference may occur at an edge between two nodes that are equidistance apart from the nearest sink respectively.

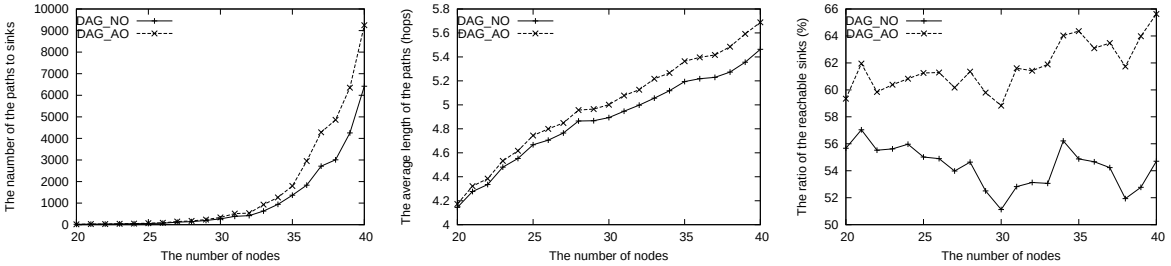
Fig. 4 shows examples of the DAGs constructed by $\mathcal{DAG}_{NO}$ and $\mathcal{DAG}_{AO}$. Nodes P_1 and node P_2 are sinks. In this example, the edge between node P_5 and P_6 is directed different direction. Whereas node P_5 has only a single path to a sink in $\mathcal{DAG}_{NO}$ (Fig. 4(a)), node P_5 has 3 paths to sinks in $\mathcal{DAG}_{AO}$ (Fig. 4(b)).

In this paper, we utilize the DAG construction for the geocast routing. We evaluate the property of paths from any node to any sink on a DAG. We use the following performance metrics: the number of paths to sinks, the average length of paths to sinks and the ratio of reachable sinks.



(a) $\mathcal{DAG}_{NO}$ (b) $\mathcal{DAG}_{AO}$

Figure 4: Example of DAGs constructed $\mathcal{DAG}_{NO}$ and $\mathcal{DAG}_{AO}$



(a) The number of paths to sinks (b) The average length of paths to sinks (c) The ratio of the reachable sinks

Figure 5: Comparison of property of DAGs

5.1 Description of Experiment

In our experiments, we produce a graph at random and compare two DAGs constructed by two protocols $\mathcal{DAG}_{NO}$ and $\mathcal{DAG}_{AO}$ in the same graph. We use a unit disk graph (UDG). A unit disk graph is an intersection graph of equal sized circles in the plane: they provide a graph-theoretic model for MANET.

The nodes are randomly allocated within the field which size is 1000×1000, and the edges are defined between two nodes whose distance is less than 250. We change the number of the nodes from 20 to 40. Every node within the circle of radius 150 around the coordinate $(250, 250)$ is included in T. When the produced graph is not connected or a node belonging to T does not exist, we destroy the graph and produce a graph again.

We use the following evaluation measures:

- The number of paths to sinks: This is defined as the total number of paths to all sinks from a node that is not a sink. We report the average value over all the nodes that are not sinks.
- The average length of the paths to sinks: This is defined as the average length of the paths to all sinks from a node that is not a sink. We report the average value over all the nodes that are not sinks.
- The ratio of the reachable sinks: This is defined as the ratio of the number of reachable sinks, and the number of sinks. We report the average value over all the nodes that are not sinks.

5.2 Experimental Result

Figs. 5(a) and 5(c) show that the number of paths to sinks and the ratio of the reachable sinks, respectively. The both results show that $\mathcal{DAG}_{AO}$ constructs a better DAG than $\mathcal{DAG}_{NO}$. A DAG can provide multiple paths to sinks from any node. When a DAG is used as a path to sinks, it may be better property that there are many paths and many reachable sinks.

6 Protocol Implementations

In this paper, we perform some computer simulations to evaluate the performance of the geocast protocols using self-stabilizing DAG constructing protocols $\mathcal{DAG}_{NO}$ and $\mathcal{DAG}_{AO}$, and known geocast protocol GeoTORA [4]. In this section, we make the necessary preparation for explaining our simulations. First, we describe how to adapt self-stabilizing protocols $\mathcal{DAG}_{NO}$ and $\mathcal{DAG}_{AO}$, in which nodes communicate by using shared variables, for practical wireless networks. Second, we describe about implementations of geocast protocols by using $\mathcal{DAG}_{NO}$ and $\mathcal{DAG}_{AO}$. At last, we introduce outline of GeoTORA [4].

6.1 Adaptation to practical networks

To adapt $\mathcal{DAG}_{NO}$ and $\mathcal{DAG}_{AO}$ for practical networks, we have to consider the following problems: how to implement communication by using shared variables, how to detect neighbors (N_i) and how to set t_i. To implement communication by using shared variables and detection of neighbors, we use *hello packet*. Every node puts all of its own values of shared variables into the hello packet and periodically sends it to all neighbors. If a node P_i receives a hello packet from some neighbor node P_j, P_i can know the existence of P_j as P_i's neighbor, hence P_i adds P_j to N_i, and stores values of variables in the hello packet as values of P_j's shared variables. If P_i does not receive a hello packet from the neighbor node P_j during certain time interval, P_i deletes P_j from N_i and resets stored values of P_j's shared variables. In other words, each process can detect appearance and disappearance of their links. We assume that if process P_i is in the geocast region, t_i is automatically set to $true$. Otherwise, t_i is automatically set to $false$.

In $\mathcal{DAG}_{NO}$, network partitioning is not considered. That is, if there is a partition of the network which is disconnected from all nodes in the geocast region, every node in the partition infinitely increases the variable d according to the protocol. So, we introduce the upper bound $\mathbf{N}$ of the distance. When a process P_i executes protocol $\mathcal{DAG}_{NO}$, if $d_i > \mathbf{N}$ then d_i is set to $\mathbf{N}$.

6.2 Implementations of Geocast Protocols with $\mathcal{DAG}_{NO}$ and $\mathcal{DAG}_{AO}$

We call the geocast protocols that use $\mathcal{DAG}_{NO}$ and $\mathcal{DAG}_{AO}$, Nearest sink Oriented Geocast Protocol (NOGP) and All sinks Oriented Geocast Protocol (AOGP), respectively. NOGP and AOGP construct and maintain a DAG by using $\mathcal{DAG}_{NO}$ and $\mathcal{DAG}_{AO}$ regardless of occurrence of a geocast transmission.

A sender selects an outgoing edge at random and sends messages. A node which is not in the geocast region relays the messages via one of its outgoing edges. When a packet reaches to a process in the geocast region, the packet is flooded in the geocast region.

6.3 GeoTORA

GeoTORA implements the geocasting by the following steps: in GeoTORA, a node which wants to send a message to members of the geocast group floods control packets in the network. If some process in the geocast region receives a control packet, it begins to construct a DAG whose sinks are node in the geocast region. After constructing a DAG in the network, geocast messages are transmitted along the DAG edges from source process to the nodes in the geocast region. When some process in the geocast region receives a geocast packet, the packet is flooded in the geocast region.

The DAG construction of GeoTORA is comprised of three steps: creating routes, maintaining routes and erasing routes. A control packet is used in every step. When the first transmission request occurs, GeoTORA starts to construct a DAG via control packets. If a process outside of the geocast region becomes a sink, maintaining routes is performed to rebuild the DAG. Erasing routes is performed in a partition of the network from the geocast region. Notice that the DAG is reconstructed after the DAG is not usable as a route to the geocast region.

7 Performance Evaluation

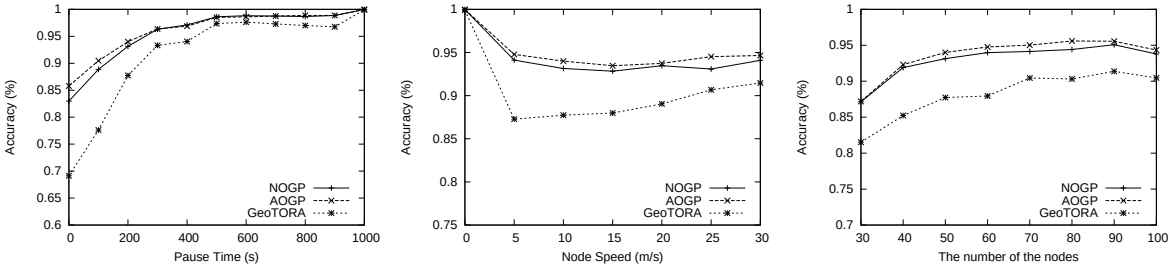
In this section, we perform some computer simulations to evaluate the performance of the geocast protocols NOGP, AOGP and GeoTORA [4]. We use a network simulator ns-3 [8] for computer simulation environment.

Table 1: Simulation parameters

Experiment	1	2	3
# of processes	50	50	30 - 100
Speed (m/s)	10	0 - 30	10
Pause time (s)	0 - 1000	200	200

Table 2: The size of each packet

Protocol	NOGP	AOGP	GeoTORA
Geocast (bytes)	512	512	512
Control (bytes)	-	-	32
Hello (bytes)	20	variable	8



(a) Experiment 1)# of processes:50, Speed:10, The pause time (s):0-1000

(b) Experiment 2)# of processes:50, Speed:0-30, The pause time (s):200

(c) Experiment 3)# of processes:30-100, Speed:10, The pause time (s):200

Figure 6: The Accuracy in Experiment 1-3

7.1 Simulation Model

The wireless model follows IEEE802.11b and link bandwidth is 11Mbps. We assume the range loss model, where the propagation loss depends only on the distance (range) between transmitter and receiver. The transmission range is set to 250 meters.

Our simulation environment is characterized by a 1000×1000 meters rectangle, with random initial processes' location. The geocast region is a circle of radius 150 meters around the coordinate $(250, 250)$. The processes move according to the random way point model. In this model, processes repeatedly select uniform random destinations, move to them, and pause there.

Simulation time is 1000 seconds. We simulate a variety of networks by varying the following parameters: the number of processes, the movement speed and the pause time. We perform three simulations : the pause time is varied from 0 seconds to 1000 seconds (Experiment 1), the movement speed is varied from 0 m/s to 30 m/s (Experiment 2) and the number of processes is varied from 30 to 100 (Experiment 3). Notice that a pause time of 0 seconds corresponds to continuous motion, whereas a pause time of 1000 seconds is equivalent to a static network. While one parameter is varied the other parameters are fixed. Table 1 summarizes our simulation parameters.

Geocast packets are sent by a particular process. The packet size is 512 bytes and the packet is sent every 10 seconds after 5.5 seconds from starting simulations. The hello packets are sent every 1 seconds. In NOGP, since a hello packet includes the variable d_i, the number of bytes of a hello packet is 20 bytes. On the other hand, in AOGP, since a hello packet includes the variable L_i, the number of bytes of a hello packet is variable. In GeoTORA, the number of bytes of a hello packet is 8 bytes because the hello packets are used to detect neighbors. The number of bytes of a control packet, which is used in GeoTORA, is 32 bytes. Table 2 summarizes the number of bytes of each packet.

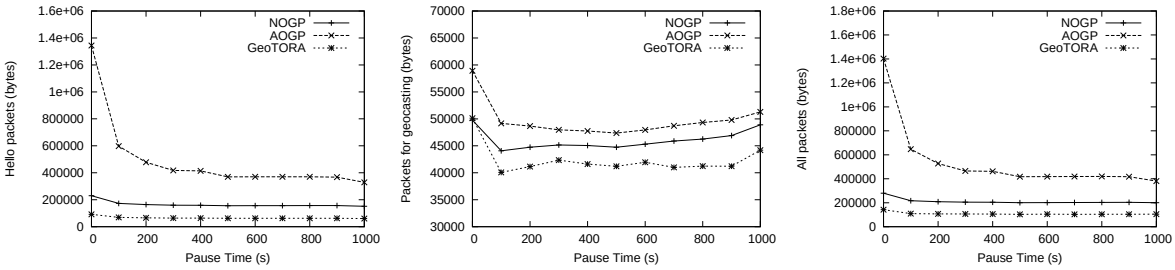
We use the following evaluation measures:

- *The Accuracy*: If a source process sends a geocast packet and at least one of the processes in the geocast region receives the geocast packet, we call the entire scenario *one success*. The accuracy is defined as ratio of the number of one successes divided by the number of geocastings.

- *The average number of packets and bytes*: They are defined as the average number of packets and bytes received by each process during the simulation, respectively. We classify packets in three as follows: hello packets, packets for geocasting and all packets. In GeoTORA, the control packets are contained in the packets for geocasting.

7.2 Simulation Result

Fig. 6 shows the accuracy in Experiment 1-3. The accuracy of NOGP and AOGP are higher than that of GeoTORA through three experiments. This can be explained as follows: when the network topology changes, GeoTORA replaces the local variables and delivers a control packet to share them with neighbors, and at this time, information before the network topology change may be left, which leads to a failure



(a) The average number of bytes of hello packets (b) The average number of bytes of packets for geocasting (c) The average number of bytes of all packets

Figure 8: The average number of bytes in Experiment 1)# of processes:50, Speed:10, The pause time (s):0-1000

of the DAG and a decrease of the accuracy. In AOGP and NOGP, since the values of the local variables are transmitted periodically via the hello packets the older information will never be left in the network.

From Figs. 6(b) and 6(c), varying the number of processes and the node speed seem not to have much effect on the accuracy. When there is no process in the geocast region or when source detects that it is partitioned from the geocast region, no packet reaches to the geocast region. In this case, this geocasting (or request to send) is ignored for calculating the accuracy hence the accuracy does not decrease. In low density of processes, a request to send a geocast is unlikely to accomplish, while the accuracy is high.

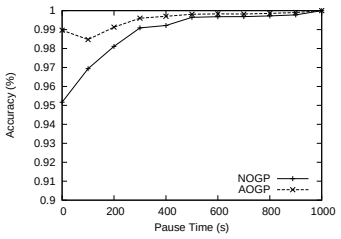


Figure 7: The Accuracy in additional experiment

From Fig. 6, the accuracy of AOGP is a little higher than that of NOGP. The results of experiments in Section 5 suggests that the accuracy of AOGP is higher than the accuracy of NOGP by using multiple paths on the DAG. Since each process relays through one of the outgoing edges, we perform an additional simulation, which we change protocols so that each process relays through all of the outgoing edge. The result of this simulation is shown in Fig. 7. The simulation parameters are similar to Experiment 1. This result shows that the accuracy of NOGP and AOGP become more higher. However, there is not so large difference between the accuracy of NOGP and AOGP.

Fig. 8 shows the average number of bytes in Experiment 1. From Fig. 8(a), the average number of bytes ofhello packets in AOGP increases with the decrease of the pause time. On the other hand, in NOGP and GeoTORA, they are nearly-constant. The reason for this is that in AOGP if a process goes out of geocast region it will take at most **N** rounds to reduce the information about the distance from this process. Increase of the number of the processes going the geocast region in and out leads to an increase of amount of hello packets.

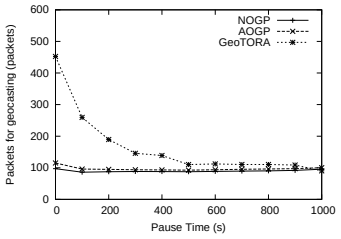


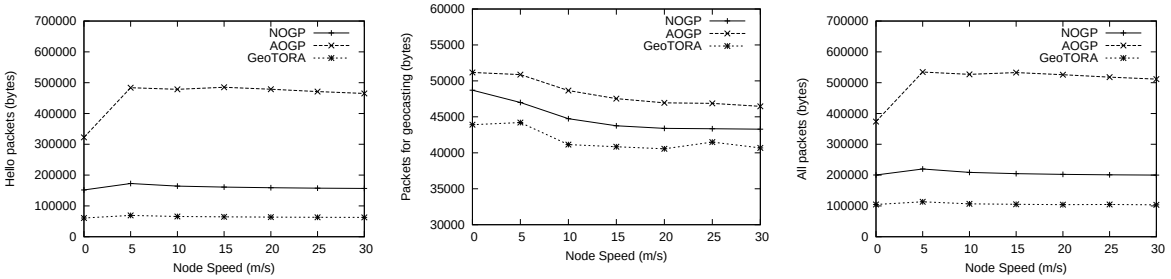
Figure 9: The average number of packets for geocasting in Experiment 1)# of processes:50, Speed:10, The pause time (s):0-1000

Fig. 9 shows the average number of packets for geocasting in Experiment 1. From Fig. 9, the average number of packets of packets for geocasting of GeoTORA is higher than those of the other protocols. One possible reason for this is that a reconstruction of a DAG occurs more often as the pause time decreases, which leads to an increase of the number of control packets. However, Fig. 8(b) shows that the increase of the number of control packets has much effect on the average number of the bytes of packets for geocasting because a control packet is much smaller than a geocast packet.

Figs. 10 shows that the increase of the movement speed has much effect on the average number of bytes.

In the nature of the MANET, the number of processes which receive a massage increases with the increase of the density of processes. From Fig. 11(a), in AOGP, a ratio of an increase of the number of bytes is higher than those of the other protocols. This can be explained as follows: the number of processes in the geocast region increases with the increase the density of processes, and hence the amount of the hello packet also increases.

From Figs. 8(b), 10(b) and 11(b), the number of bytes of packets for geocasting of AOGP is higher

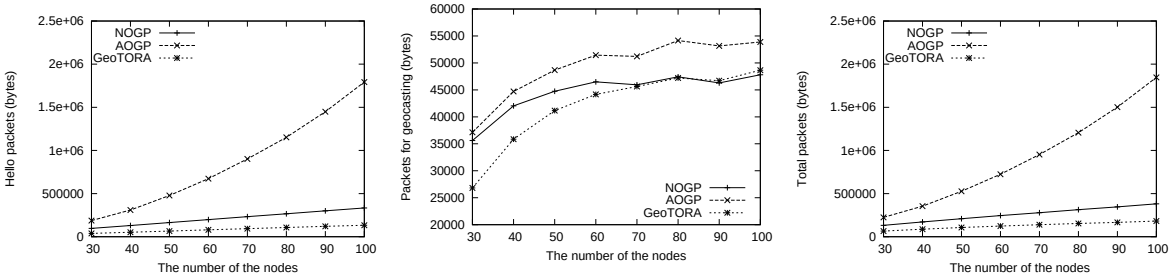


(a) The total size of hello packets

(b) The total size of packets for geocasting

(c) The total size of all packets

Figure 10: The total packet size in Experiment 2)# of processes:50, Speed:0-30, The pause time (s):200



(a) The total size of hello packets

(b) The total size of packets for geocasting

(c) The total size of all packets

Figure 11: The total packet size in Experiment 3)# of processes:30-100, Speed:10, The pause time (s):200

that that of NOGP. This is due to the fact that the average length of paths to sinks of $\mathcal{DAG}_{AO}$ is higher than that of $\mathcal{DAG}_{NO}$ from the results of experiments in Section 5, and hence the number of processes that relays a geocast packet, which leads to an increase of the number of transmission of geocast packets in the network.

From Figs. 8(c), 10(c) and 11(c), the amount of the hello packets is the largest among all packets in our simulations.

8 Conclusions

In this paper, we propose the DAG constructing self-stabilizing protocol $\mathcal{DAG}_{AO}$. Then we perform some computer simulations. First, we clarify the property of DAGs constructed $\mathcal{DAG}_{AO}$ and $\mathcal{DAG}_{NO}$. Simulation results show that $\mathcal{DAG}_{AO}$ has better property for implementing some routing protocols, like a geocast protocol, than $\mathcal{DAG}_{NO}$ [6]. Second, we describe the implementation of our geocast protocols that use $\mathcal{DAG}_{NO}$ and $\mathcal{DAG}_{AO}$. Simulation results show that the accuracy of our protocol is higher than GeoTORA in any network conditions. AOGP shows better performance for accuracy, but it needs more overhead than others with current implementation. Our future work is to reduce the overhead of AOGP and get more accuracy.

References

[1] Malpani, N., Welch, J.L., Vaidya, N.: Leader election algorithms for mobile ad hoc networks. In: Proceedings of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications. DIALM '00, New York, NY, USA, ACM (2000) 96–103

[2] Ghosh, S., Karaata, M.: A self-stabilizing algorithm for coloring planar graphs. Distributed Computing **7** (1993) 55–59 10.1007/BF02278856.

[3] Park, V., Corson, M.: A highly adaptive distributed routing algorithm for mobile wireless networks. In: INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE. Volume 3. (apr 1997) 1405 –1413 vol.3

[4] Ko, Y.B., Vaidya, N.: Geotora: a protocol for geocasting in mobile ad hoc networks. In: Network Protocols, 2000. Proceedings. 2000 International Conference on. (2000) 240 –250

[5] Navas, J.C., Imielinski, T.: Geocast - geographic addressing and routing. In: Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking. MobiCom '97, New York, NY, USA, ACM (1997) 66–76

[6] Miura, T., Katayama, Y., Wada, K., Takahashi, N.: A fault-containing self-stabilizing protocol for constructing a directed acyclic graph. IPSJ SIG Technical Report. AL **2011**(20) (2011-02-28) 1–8

[7] Karaata, M., Chaudhuri, P.: A self-stabilizing algorithm for strong fairness. Computing **60** (1998) 217–228 10.1007/BF02684333.

[8] : The ns-3 network simulator