Move-efficient algorithms for group gossiping of mobile agents

Jun Ri, Masahiro Shibata, Fukuhito Ooshita, Hirotsugu Kakugawa and Toshimitsu Masuzawa
Graduate School of Information Science and Technology
Osaka University
Osaka, Japan
Email: {jun-ri, m-sibata, f-oosita, kakugawa, masuzawa}@ist.osaka-u.ac.jp

***Abstract*—We introduce a concept of agent groups and formulate the group gossiping problem. An (agent) group is a set of agents that work together to attain a single objective. For example, agents created by a single application form one group. Since multiple applications are executed in a single mobile agent system, there exist multiple groups in such a system. In this case, it is useful to support cooperation among agents in each group.**

From this motivation, we formulate the group gossiping problem. The group gossiping problem requires each agent to collect information of all agents in its group. Since information to be collected is private and precious, no agents want to expose the information to other groups. For this reason, agents can exchange only control information (e.g., counter values or IDs) with other groups. In this setting, we aim to minimize the total number of moves to solve the group gossiping problem. As a result, we show the asymptotically tight upper and lower bounds for several network topologies.

***Keywords*-Mobile agent, Group gossiping, Move complexity, Distributed algorithm**

I. Introduction

A. Background and Motivation

As a design paradigm for distributed systems, *mobile agents* have received a lot of attention [3], [14], [15]. A (mobile) agent is an autonomous piece of software that moves in a network carrying its state information. There are many reasons for using agents [15]; For example, if remote hosts store a large amount of data for processing, agents can reduce the network traffic by moving to the remote hosts and processing the data locally. Besides adaptability and flexibility of agents simplify design of distributed systems.

In most mobile agent systems, multiple agents work cooperatively to improve system performance. For example, in a network management system, multiple agents traverse the network to collect load information of nodes and links, share the collected information, and inform each node about the information. To support the cooperation of mobile agents, many algorithms for fundamental operations such as gossiping and gathering have been developed [20], [25]. Gossiping requires all agents to share information of all agents, and gathering requires all agents to meet at a single node.

In this paper, we introduce a concept of *agent groups*. Intuitively, agents in a single agent group work to attain the same objective. For example, if an application creates multiple agents to attain some objective, the agents form a single agent group. Previous algorithms for gossiping and gathering assume all agents belong to a single group in the sense that all agents aim to achieve the same goal (i.e., share the information of all agents or meet all agents).

On the other hand, mobile agents from *multiple* agent groups often share a single mobile agent system. For example, if multiple applications share a single mobile agent system, agents created by each application make up one agent group. Another example is a mobile agent system used by multiple organizations. In such a system, agents created by each organization make up one agent group. For both cases, agents are not interested in agents that belong to other agent groups. Consequently agents want to share the information or to gather among agents in their agent groups. This motivates us to consider new variants of operations, gossiping or gathering in each agent group.

B. Our contributions

From the above motivation, we formulate the group gossiping problem. In this problem, agents are divided into groups, and the goal is to make each agent collect information of all agents in its group. We aim to minimize the total number of agent moves (over all groups) to solve the group gossiping problem. This is because, since all agents share a single mobile agent system, it is important to optimize the performance of the whole system.

Since algorithms for gossiping among agents in a single agent group are proposed [25], there are two trivial solutions to solve the group gossiping problem. One is to execute the algorithm in [25] by assuming that all agents belong to a single group. However, to execute this algorithm, agents must expose its information to agents in other groups. Such a behavior is usually not allowed due to security reasons since the information may be private and precious. Another solution is to execute the algorithm in [25] independently for each group. This solution avoids the above security problem. However, since agents do not cooperate with agents in other groups, this requires a large total number of moves. For this reason, we aim to reduce the total number of moves by

Table I
THE TOTAL NUMBER OF MOVES TO SOLVE THE GROUP GOSSIPING PROBLEM

Graph	System model	Sense of direction	A trivial solution	Our results
Ring	synchronous	without	$O(gN)$	$\Theta(gN)$
	asynchronous	without	$O(gN \log k')$	$\Theta(N \log k' + gN)$
Tree	asynchronous	without	$O(gN)$	$\Theta(gN)$
Complete	asynchronous	without	$O(gN \log k')$	$\Theta(N \log k)$
		with	$O(gN)$	$\Theta(N)$
Arbitrary	asynchronous	without	$O(gN \log k' + gM)$	$\Theta(N \log k' + M + gN)$

N, M and k are the numbers of nodes, links and agents respectively.
g is the numbers of groups, and k' is the maximum number of agents in a group.

exchanging some control data (e.g., counter values or IDs) among agents in different groups.

Table I summarizes the contribution of this paper for the group gossiping problem. The column of a trivial solution shows the total number of moves required to execute the algorithm in [25] independently for each group. For synchronous rings and asynchronous trees, we show that the trivial algorithm is asymptotically optimal in terms of the total number of moves. For asynchronous ring networks, asynchronous complete networks, and asynchronous arbitrary networks, we propose algorithms that can reduce the total number of moves compared to the trivial solution. This means cooperation of agents in different groups reduces the total number of moves to solve the group gossiping problem. We also show that the proposed algorithms are asymptotically optimal in terms of the total number of moves.

C. Related Works

To improve the performance of mobile agent systems, many fundamental problems for cooperation of agents have been studied. The most investigated problem is the gathering problem [14], [20], which requires all agents to meet at a single node[1]. The gathering problem has been considered in several settings [2], [4], [6], [7], [8], [10], [18]. Dessmark et al. [6] give algorithms for synchronous agents in trees, rings, and arbitrary graphs. Dieudonné and Pelc [7] propose deterministic algorithms for multiple synchronous agents in arbitrary networks. Dieudonné et al. [8] show that two asynchronous agents can meet in arbitrary networks at polynomial number of moves.

The gossiping problem is considered in [17], [25]. Note that algorithms for the gathering problem are also solutions for the gossiping problem because all agents can exchange their information by meeting at a single node. However, the gossiping problem can be solved without gathering of all agents. Suzuki et al. [25] show that the gossiping problem requires a smaller total number of moves than the gathering problem. In [17], self-stabilizing algorithms for the gossiping problem are studied.

[1]The gathering problem for two agents is sometimes called the rendezvous problem.

Another fundamental problem is the exploration problem, which requires agents to visit every node. Exploration algorithms for a single agent [19], [22] and for multiple agents [5], [9] have been proposed. The important notion for the exploration problem is a universal exploration sequence (UXS) [13]. By moving based on a UXS, agents can visit every node. Reingold [22] gives a log-space constructible UXS and shows that an agent solves the exploration problem with $O(\log n)$ bits of memory.

While all the above researches do not consider agent groups, Shibata et al. [23], [24] consider some groups and introduce the partial gathering problem. The partial gathering problem requires, for given input g, that each agent makes a group composed of at least g agents and meet them at a single node. The concept looks similar to groups in this paper, however that is different because groups are initially given in the group gossiping problem.

II. PRELIMINARIES

In this section, we define the system model and the group gossiping problem. Note that we use the same system model as [25] except for a concept of groups.

A. Mobile Agent Systems

A network is modeled as an undirected graph $G = (V, E)$, where V and E are respectively the node set and the link set in G. The numbers of nodes and links are denoted by N (i.e., $N = |V|$) and M (i.e., $M = |E|$), respectively. A link in E connects two distinct nodes in V. The link between nodes u and v is denoted by e_{uv} or e_{vu}. On each node, each incident link is locally labeled by port numbers. Let $\lambda_u(e)$ be the port number of link e on a node u ($\lambda_u(e) \in \{1, 2, \ldots, deg_u\}$, where deg_u is the number of u's incident links). There exist k agents in the network. An agent a_i ($0 \leq i < k$) is an autonomous state machine that can migrate from one node to another in the network. Agents on a node $u \in V$ can migrate to a node $v \in V$ only when link e_{uv} is contained in E. We assume that each agent has a distinct identifier (ID) [2]. Let id_i be the ID of agent a_i (use id_i and a_i interchangeably in this paper). Each agent does not initially know IDs of

[2]Node IDs are not required in this paper, however this is not essential. This is because distinct agents can assign distinct IDs to nodes when they visit the nodes.

other agents. Agents are divided into g *groups*. Each agent a_i knows its group ID gid_i, and agents with the same group ID form a single group. For each agent a_i, we define $\mathcal{G}(a_i)$ as the set of agents that belong to the same group as a_i (i.e., have the same group ID as a_i). Note that $a_i \in \mathcal{G}(a_i)$ holds for any i. We assume each group consists of at least two agents. Let k' be the maximum number of agents that belong to a single group (i.e., $k' = \max_{0 \leq i < k}\{|\mathcal{G}(a_i)|\}$). We assume that each agent has prior knowledge of neither G, N, M, k, g nor k'. Each agent is initially located on any node in G, and more than one agent is not located on the same node. A node on which an agent is initially located is called the *home node* of the agent. When a node v is the home node of agent a_i, we also say a_i is the *home agent* of v.

Each node v is provided with a *whiteboard*, i.e., local storage where agents on v can write and read data. When multiple agents on a node execute their operations, the operations are sequentially executed in an arbitrary order with the following assumption: at the beginning of the algorithm, each agent executes operations at its home node before other agents access a whiteboard on the node.

An agent a_i on each node v performs a sequence of the following operations;

- *read*(v) : agent a_i reads data written on node v's whiteboard, and executes local computation.
- *write*(v, d) : agent a_i writes data d on node v's whiteboard.
- *move*($v, \lambda_v(e)$) : agent a_i moves to one of v's neighbor nodes through the link labeled by port number $\lambda_v(e)$. If $\lambda_v(e)$ is zero, a_i stays on v.

We assume that these three operations are executed atomically. Agents are said to be *asynchronous* if migration time and local processing time of agents are unpredictable but finite. In contrast, agents are *synchronous* if all agents execute every rounds synchronously; in each round, every agent arrives at a node, accesses the whiteboard and executes local computation on the node, and stays on the node or starts migration to one of the neighboring nodes. The agents arrive at the destination nodes at the beginning of the next round.

A state of an agent is represented by a set of variables the agent has and a set of information the agent has collected. A state of a node is represented by the state of its whiteboard. A system configuration C is represented by the states of all nodes, the states of all agents, and the locations of all agents. A system configuration is changed by events of agents (e.g., read and write on a whiteboard, and migration). Let C_0 be an initial configuration of a system and Ev_i be a set of events that occur simultaneously at the configuration C_i. An execution of a mobile agent system is an alternate sequence of configurations and sets of events $EX = C_0, Ev_0, C_1, Ev_1, C_2, \cdots$, such that occurrence of events Ev_{i-1} changes the configuration from C_{i-1} to C_i. We say that an agent a_j terminates in a configuration C_i iff a_j never executes any operation after C_i.

B. Group Gossiping Problem

In this paper, we define the *group gossiping problem*. In an initial configuration, each agent a_j has only its own information I_j. The goal of this problem is that every agent collects information of all agents in its group. Hereinafter information means information each agent has to collect. The group gossiping problem is defined as follows.

Let $S_j(C_i)$ be a set of the information an agent a_j has in configuration C_i. In initial configuration C_0, each agent a_j has only its own information I_j;

$$S_j(C_0) = \{I_j\}.$$

The group gossiping problem is solved in configuration C_i iff all k agents terminate and the following condition is satisfied;

$$\forall j (0 \leq j < k) : \ S_j(C_i) = \bigcup_{a_\ell \in \mathcal{G}(a_j)} \{I_\ell\}$$

Agents can write only the control data on a whiteboard, e.g., some number of identifiers and counter values. We disallow each agent a_j to leave any information I_j on a whiteboard due to security reason. It is insecure to write precious information on a whiteboard because every agent in different groups can access the whiteboard. Instead, we allow agents to exchange the set of information if agents in the same group stay on the same node. When a set P of agents is located on the same node in configuration C_i, then the subsequent configuration C_{i+1} satisfies;

$$\forall a_j \in P : \ S_j(C_{i+1}) = \bigcup_{a_\ell \in P \cap \mathcal{G}(a_j)} S_\ell(C_i).$$

We define a move as a migration of an agent from one node to its neighbor node. The complexity of the group gossiping problem is measured by the total number of moves until all agents terminate in the worst case.

III. A LEADER-ELECTION-BASED ALGORITHM

In this section, we show a leader-election-based algorithm to solve the group gossiping problem for several networks. To elect a leader among agents, we use an algorithm for the node leader election problem in message-passing systems. For this reason, we first give the definition of the node leader election problem.

Definition 1 In message-passing systems, an algorithm is said to solve the node leader election problem for network G iff it satisfies the following conditions: 1) exactly one of nodes is elected as a leader and all nodes in G know the ID of the leader node, and 2) once a node decides to be a leader, the node never changes its decision.

If we have a node leader election algorithm for asynchronous message-passing systems, we can use it to elect a leader agent in mobile agent systems from the following theorem in [25].

Theorem 1 [25] Suppose the total number of exchanged messages to execute a node leader election algorithm with k initiators in asynchronous message-passing systems is at most m_{mp}. Then, the (agent) leader election with k agents is solved with at most $2 \cdot m_{mp}$ moves.

In the rest of this section, we explain an algorithm to solve the group gossiping problem by using an algorithm for the agent leader election. First, we explain the idea of algorithms for the gossiping problem in [25] to extend it to the group gossiping problem. Remind that the gossiping problem requires agents to collect information of all agents. In the algorithm, agents first execute a leader election algorithm and elect a single leader. Then, the leader completes gossiping by traversing the network twice: The leader collects information of all agents in the first traversal, and distribute the information to every agent in the second traversal. Since the leader election usually requires a larger number of moves, the leader election dominates the total number of moves throughout the algorithm.

For the group gossiping problem, we can use the same idea. That is, each group elects a single group leader, and the group leader completes gossiping in the group by traversing the network twice. However, if each agent group executes a leader election independently, this requires a lot of moves. To avoid this, we elect a single leader among all agents at the beginning of the algorithm. Then, the leader elects a group leader for each group by traversing the network once. Concretely, when the leader meets an agent of group gid, it nominates the agent as a group leader if it has not yet met another agent of group gid.

From the above algorithm, we have the following lemma.

Lemma 1 Suppose that the leader election with k agents is solved with at most m_l moves and the number of moves required for an agent to traverse the whole network is at most m_t. Then, the group gossiping problem with k agents and g groups is solved with $m_l + (2g+1)m_t$ moves.

Proof: Agents elect a single leader and then the leader elects a group leader for each group by traversing the network once. This requires $m_l + m_t$ moves. After that, each group leader completes gossiping in its group by traversing the network twice. This requires $2gm_t$ moves. ■

IV. Upper and lower bounds

In this section, we present the upper and lower bounds on the total number of moves for the group gossiping problem in several networks.

A. Synchronous ring networks

In this subsection, we consider synchronous ring networks without sense of direction. For ring networks, the node leader election problem in synchronous message-passing systems can be solved with $O(N)$ messages [11]. Since this algorithm works in only synchronous systems, Theorem 1 cannot be applied. However, fortunately this result can be easily applied to agent systems, and it is proved that gossiping among k agents can be solved with $O(N)$ moves in any synchronous ring network [25]. By executing this algorithm independently for each group, we have the following theorem.

Theorem 2 The group gossiping problem is solved with $O(gN)$ moves in synchronous ring networks.

For the lower bound, we have the following theorem.

Theorem 3 Any algorithm for the group gossiping problem requires $\Omega(gN)$ moves in synchronous ring networks.

Proof: For any given N, k, g and k', we show that there exists an initial configuration that requires $\Omega(gN)$ moves to solve the group gossiping problem. Assume that nodes $v_0, v_1, \ldots, v_{N-1}$ are located in this order. Let $\mathcal{G}_i$ $(0 \leq i < g)$ be a set of agents in group i. Since $|\mathcal{G}_i| \geq 2$ holds, we can deploy agents as follows: First deploy two agents in $\mathcal{G}_i$ on v_i and $v_{i+N/2}$ for each i $(0 \leq i < g)$, and then deploy remaining agents on remaining nodes arbitrarily. Since the two agents in each group must exchange their information, this requires at least $N/2$ moves for each group. Therefore, the total number of moves is at least $gN/2$. ■

B. Asynchronous ring networks

In this subsection, we consider asynchronous ring networks without sense of direction. For ring networks, the node leader election problem in asynchronous message-passing systems can be solved with $O(N \log N)$ messages [21]. This is the message complexity for an arbitrary number of initiator nodes in the worst case. It can easily be shown that the message complexity for k initiator nodes is $O(N \log k)$ by using an algorithm similar to [21]. Therefore, from Theorem 1, leader election among k agents can be solved with $O(N \log k)$ moves.

An agent can travel the whole network with N moves. Thus, from Lemma 1 and $k \leq gk'$, the following theorem is obtained.

Theorem 4 The group gossiping problem is solved with $O(N \log k' + gN)$ moves in asynchronous ring networks.

In the following, we show the lower bound. Similarly to Theorem 3, we have the following lemma.

Lemma 2 Any algorithm for the group gossiping problem requires $\Omega(gN)$ moves in asynchronous ring networks.

In addition, we prove the following lemma from the lower bound of the node leader election problem in message-passing systems.

Lemma 3 Any algorithm for the group gossiping problem requires $\Omega(N \log k')$ moves in asynchronous ring networks.

Proof: We show the lemma holds even if the number of group g is given to each node. For contradiction, assume that the group gossiping problem is solved with $o(N \log k')$ moves in any asynchronous ring network. Then, we show that the node leader election problem with k' initiator nodes in message-passing systems is solved with $o(N \log k')$ messages.

Consider a N-node ring network $G = (V, E)$ in message-passing systems. Assume that nodes $v_0, v_1, \ldots, v_{N-1}$ are located in this order. Let V_I be a set of k' initiator nodes. To solve the node leader election in G with the initiator node set V_I, we let nodes in G simulate (agent) leader election in a virtual ring network $G' = (V', E')$ constructed as follows: For each $v_i \in V_I$, replace v_i in G by a path $P_i = (v_i^0, v_i^1, \ldots, v_i^{g-1})$, and add links $\{v_{i-1}, v_i^0\}$ and $\{v_i^{g-1}, v_{i+1}\}$ to G. Then, the length of ring G' is $N' = N + (g-1)k' \leq 2N$. To solve the node leader election in G, each initiator node v_i simulates all of nodes in P_i of the virtual ring network G'.

Each node can easily simulate the agent algorithm by sending a state of an agent in a message. Then, the number of total messages is equal to the number of total moves of agents. First, each initiator node v_i puts g agents $a_i^0, a_i^1, \ldots, a_i^{g-1}$ on every node in path P_i. At that time, each agent a_i^j has an ID (id_i, j) and a group ID j, where id_i is the ID of node v_i. In the whole virtual ring network, there exist $k = gk'$ agents and each group consists of k' agents. Each agent uses its ID as information to be collected. In this setting, nodes simulate an algorithm to solve the group gossiping problem. This is done with $o(N' \log k') = o(N \log k')$ messages. After completing the group gossiping, each initiator node knows IDs of all initiator nodes. Thus, the node with the minimum ID can become a leader. After that, the leader broadcasts its ID with $O(N)$ messages and then the node leader election problem is solved. Throughout the algorithm, nodes exchange $o(N \log k')$ messages to solve the node leader election problem.

However, it is proved in [**?**] that the lower bound on messages for the node leader election problem with k' initiator nodes is $\Omega(N \log k')$. This is a contradiction. ■

From Lemmas 2 and 3, we have the following theorem.

Theorem 5 Any algorithm for the group gossiping problem requires $\Omega(N \log k' + gN)$ moves in asynchronous ring networks.

C. Asynchronous non-rooted tree networks

In this subsection, we consider asynchronous non-rooted tree networks without sense of direction. For asynchronous tree networks, it is proved in [25] that gossiping among k agents can be solved with $O(N)$ moves. By executing this algorithm independently for each group, we have the following theorem.

Theorem 6 The group gossiping problem is solved with $O(gN)$ moves in asynchronous tree networks.

For the lower bound, we have the following theorem.

Theorem 7 Any algorithm for the group gossiping problem requires $\Omega(gN)$ moves in asynchronous tree networks.

Proof: We consider a N-node line network as a tree network. Then, we can prove the theorem similarly to Theorem 3. ■

D. Asynchronous complete networks without sense of direction

In this subsection, we consider asynchronous complete networks without sense of direction. It is proved in [1] that the message complexity of an algorithm for the node leader election problem in message-passing systems is $O(N \log N)$. This is the message complexity for an arbitrary number of initiator nodes in the worst case. It can easily be shown that the message complexity for k initiator nodes is $O(N \log k)$ by using an algorithm similar to [1]. Thus, from Theorem 1, k agents can elect a leader in $O(N \log k)$ moves. Since an agent can travel the whole network in $O(N)$ moves, agents can solve the group gossiping problem in $O(N \log k + gN)$ moves from Lemma 1.

In the following, we reduce the total number of moves from $O(N \log k + gN)$ to $O(N \log k)$. This is done by reducing the total number of moves after one leader is elected. First, we explain the overview of the whole algorithm. In the first phase, agents execute leader election and then move back to their home nodes. In the second phase, an elected leader a_ℓ traverses the whole network, and recognizes, for each group, the links incident at a_ℓ's home node that connect to home nodes of the agents in the group. Agent a_ℓ writes this information on the whiteboard of its home node to guide group leaders later. In the third phase, a_ℓ traverses the whole network to elect one group leader from each group. In the fourth phase, each group leader visits every agent in its group twice. To do this, each group leader first moves to a_ℓ's home node and then visits every agent in its group by using the information written in the second phase. Note that, since the sets of nodes visited by group leaders are disjoint except for a_ℓ's home node, the fourth phase requires $O(N)$ total moves.

We give the details of the algorithm. First, we explain some important variables on an agent or a whiteboard.

For each agent a_i, we denote the ID and the group ID by $a_i.id$ and $a_i.gid$, respectively. Agent a_i has a variable $a_i.role$ which indicates the role of a_i. The role of a_i can be a candidate ($a_i.role = candidate$), a leader ($a_i.role = leader$), a non-leader ($a_i.role = non\_leader$), or a group leader ($a_i.role = group\_leader$). At the beginning of the algorithm, every agent is a candidate. After the first phase, one agent becomes a leader and others become non-leaders. After the third phase, one agent in each group becomes a group leader. Each node v has the following variables in its whiteboard: $v.id$, $v.gid$, $v.port\_label[0, 1, \ldots, N-2]$, and $v.p_{leader}$. If v is the home node of some agent a_i, a_i first assigns its ID and group ID to $v.id$ and $v.gid$ respectively. Variables $v.port\_label[0, 1, \ldots, N-2]$ and $v.p_{leader}$ store the information to guide group leaders. We assume every variable on the whiteboard has $\perp$ as its initial value.

In the first phase, each agent assigns its ID and group ID to $v.id$ and $v.gid$, and then executes a leader election algorithm described in the first paragraph of this subsection. After completing the leader election, each agent goes back to its home node. This behavior is easily realized by memorizing its trajectory (i.e., a sequence of port numbers) and moving based on the reverse of the trajectory. At the end of the first phase, one agent a_ℓ is elected as a leader (i.e., $a_\ell.role = leader$) and others are non-leaders (i.e., $a_i.role = non\_leader$ for $i \neq \ell$).

In the second and third phases, only a leader agent a_ℓ moves. The pseudocode of the second and third phases is given in Algorithm 1. In the second phase, for each port p, a_ℓ leaves its home node v_ℓ through port p, visits a node v, records v's group ID (if v is a home node of an agent), returns to v_ℓ, and then writes v's group ID to $v_\ell.port\_label[p]$. During the second phase, after a_ℓ moves from v_ℓ to some node v, a_ℓ writes on $v.p_{leader}$ the port number connecting to v_ℓ. In the third phase, for each group except for a_ℓ's group, a_ℓ moves to a home node of one agent and nominates it as a group leader. At the end of the third phase, a_ℓ also becomes a group leader. In the pseudocode, a_ℓ uses some additional variables on agents. In line 17, a leader nominates an agent as a group leader by exchanging some messages with the agent. The message exchange is easily realized by using the whiteboard.

In the fourth phase, each group leader a_g visits home nodes of agents in its group twice. The pseudocode of the fourth phases is given in Algorithm 2. On the first visit a_g collects information of each agent, and on the second visit a_g delivers collected information. Each group leader visits only home nodes of agents in its group by moving based on $v_\ell.port\_label[0 \ldots N-2]$. Each group leader terminates after it completes delivery of information. Each non-leader agent terminates after it receives information of all agents in the same group.

From the above algorithms, the following theorem is obtained.

Algorithm 1 The second and third phases: the behavior of leader a_ℓ.

```
1:  // The second phase
2:  for p := 0 to N − 2 do
3:     leave through port p and visit v through port q
4:     v.p_leader := q
5:     v_gid := v.gid // v.gid =⊥ if v has no home agent
6:     leave through port q and go back to v_ℓ
7:     v_ℓ.port_label[p] := v_gid
8:  end for
9:
10: // The third phase
11: selected := {⊥, a_ℓ.gid}
12: for p := 0 to N − 2 do
13:    if v_ℓ.port_label[p] ∈ selected then
14:       continue
15:    else
16:       leave through port p and visit v
17:       wait for a home agent of v and nominate it as a
          group leader
18:       selected := selected ∪ {v.gid}
19:       leave through port v.p_leader and go back to v_ℓ
20:    end if
21:    become a group leader
22: end for
```

Algorithm 2 The fourth phase: the behavior of group leader a_g.

```
1:  // v_g is the current node of a_g.
2:  if v_g.p_leader ≠⊥ then
3:     // v_g = v_ℓ if v_g.p_leader =⊥ holds.
4:     leave through port v_g.p_leader and visit v_ℓ
5:  end if
6:  for p := 0 to N − 2 do
7:     if v_ℓ.port_label[p] = a_g.gid then
8:        leave through port p and visit v
9:        wait for the home agent of v and collect informa-
          tion
10:       leave through port v.p_leader and visit v_ℓ
11:    end if
12: end for
13: for p := 0 to N − 2 do
14:    if v_ℓ.port_label[p] = a_g.gid then
15:       leave through port p and visit v
16:       deliver information to the home agent of v
17:       leave through port v.p_leader and visit v_ℓ
18:    end if
19: end for
```

Theorem 8 The group gossiping problem is solved with $O(N \log k)$ moves in asynchronous complete networks without sense of direction.

Proof: Clearly the above algorithm solves the group gossiping problem. In the following, we consider the total number of moves. In the first phase, agents elect a leader with $O(N \log k)$ moves from the algorithm proposed in [1] and Theorem 1. They return to their home nodes with the same number of moves. Consequently, the first phase requires $O(N \log k)$ moves. In the second and third phases, a leader visits all nodes twice with $O(N)$ moves. In the fourth phase, each group leader visits home nodes of agents in its group twice. Since the sets of nodes visited by group leaders are disjoint except for v_ℓ, the fourth phase requires $O(N)$ total moves. Therefore, the toal number of moves is $O(N \log k)$ throughout the algorithm. ■

In the following, we show the lower bound. Similarly to [1], it can be proved that the message complexity of the node leader election problem with k initiator nodes is $\Omega(N \log k)$. By the proof similar to Lemma 3, we can show the lower bound is $\Omega(N \log k')$. However, since this is not tight, we prove the tight lower bound by another approach.

Theorem 9 Any algorithm for the group gossiping problem requires $\Omega(N \log k)$ moves in asynchronous complete networks without sense of direction.

Proof: For simplicity, we assume $k = 2^p$ and $N = qk$ hold for some positive integers p, q. Without loss of generality, we assume a_0 and a_{k-1} belong to the same group. To prove the lower bound on the total number of moves, we consider agents that have a communication capability stronger than that assumed in Section II. To be concrete, we assume that, when agent a_x visits some node v after agent a_y visits v, agents a_x and a_y can continue to obtain both of a_x's state and a_y's state after that. We say, in this case, a_x and a_y share the states, and denote the relation by $a_x \simeq a_y$. Clearly, the relation $\simeq$ satisfies reflexive law and symmetrical law. In addition, we assume $\simeq$ satisfies transitive law, that is, $\forall a_x, a_y, a_z : a_x \simeq a_y \wedge a_y \simeq a_z \Rightarrow a_x \simeq a_z$ holds. Then, $\simeq$ satisfies equivalence law and consequently a set of agents can be divided into equivalence classes. We define $[a_x]$ as the equivalence class of a_x under $\simeq$, that is, $[a_x] = \{a_y | a_y \simeq a_x\}$. We define the territory $T(a_x)$ of a_x as the set of nodes visited by an agent in $[a_x]$. From the definition, when agent a_y visits a node in $T(a_x)$, equivalence classes $[a_x]$ and $[a_y]$ are merged. To solve the group gossiping problem, all agents in the same group must belong to the same equivalence class.

To prove the lower bound, we consider an (adversary) scheduler. The scheduler decides the timing of agents' movements. In addition, we assume the scheduler decides the port number of each link during execution of an algorithm. That is, the scheduler assigns a port number to a link on a node when the port number on the node is first used. When an agent visits node v through link e, the port number of e on v is also assigned if it is first used.

Fix an algorithm A that solves the group gossiping problem. We construct the behavior of the scheduler that makes agents move $\Omega(N \log k)$ times to solve the group gossiping by A.

The behavior of the scheduler is divided into multiple rounds. The scheduler activates agents independently but synchronizes them at the end of each round. First, we consider the 0-th round. During the 0-th round, the scheduler makes every agent visit N/k nodes on the condition that no two agents visit the same node. That is, the scheduler assigns a port number to each link so that no two agents visit the same node. Then, for every agent a_x, $[a_x] = \{a_x\}$ and $|T(a_x)| = N/k$ hold. At the end of the 0-th round, the scheduler makes a_{2m} and a_{2m+1} ($m = 0, 1, \ldots, k/2 - 1$) visit a node in $T(a_{2m+1})$ and $T(a_{2m})$, respectively. Since equivalence classes $[a_{2m}]$ and $[a_{2m+1}]$ are merged, $[a_{2m}] = \{a_{2m}, a_{2m+1}\}$ and $|T(a_{2m})| = 2N/k$ hold. The total number of moves during the 0-th round is N because every agent moves N/k times.

After that, the first round starts. Let $[a_x]_1$ and $T_1(a_x)$ be the equivalence class and territory of a_x at the beginning of the first round, respectively. Then, $[a_{2m}]_1 = \{a_{2m}, a_{2m+1}\}$ and $|T_1(a_{2m})| = 2N/k$ hold for $m = 0, 1, \ldots, k/2 - 1$. During the first round, the scheduler makes agents in $[a_{2m}]_1$ move in their territory $T_1(a_{2m})$ as many times as possible. That is, the scheduler, if possible, chooses an agent $a_x \in [a_{2m}]_1$ and assigns a port number so that a_x moves to a node in $T_1(a_{2m})$. If such a behavior is impossible, the scheduler makes agents move so that an agent in $[a_{4m}]_1$ and $[a_{4m+2}]_1$ ($m = 0, 1, \ldots, k/4 - 1$) moves to a node in $T_1(a_{4m+2})$ and $T_1(a_{4m})$, respectively. Then, the first round ends. At that time, $[a_{4m}] = \{a_{4m}, a_{4m+1}, a_{4m+2}, a_{4m+3}\}$ and $|T(a_{4m})| = 4N/k$ hold for $m = 0, 1, \ldots, k/4 - 1$. We consider the total number of moves during the first round. Assume that, at the end of the first round, agent $a_x \in [a_{4m}]_1$ (resp., $a_x \in [a_{4m+2}]_1$) moves from node $u \in T_1(a_{4m})$ (resp., $u \in T_1(a_{4m+2})$) to node $v \in T_1(a_{4m+2})$ (resp., $v \in T_1(a_{4m})$). This behavior happens because every unused link around u connects to a node not in $T_1(a_x)$. In other words, every link connecting to a node in $T_1(a_x)$ is used. From $|T_1(a_x)| = 2N/k$, $2N/k - 1$ links around u is used. Note that, before the beginning of the first phase, the number of used links around u is at most N/k because $|T(a_y) = N/k|$ holds for each a_y at the end of the 0-th round. This implies at least $N/k - 1$ links around u are used during the first round. That is, during the first round, agents in $[a_x]_1$ move at least N/k times in total including the movement from u to v. Since there exist $k/2$ equivalence classes, the total number of moves during the first round is at least $N/2$.

We can repeat the same discussion for the subsequent rounds. Consider the i-th round. Let $[a_x]_i$ and $T_i(a_x)$ be the equivalence class and territory of a_x at the beginning of the i-th round, respectively. Then, $[a_{2^i m}]_i =$

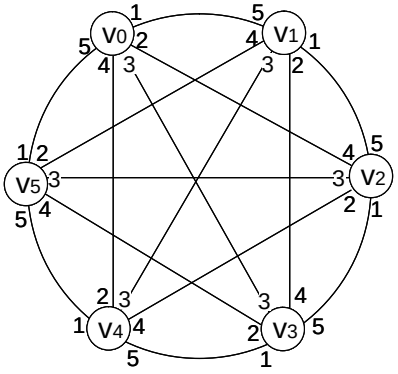

Figure 1. A complete network with sense of direction of six nodes.

$\{a_{2^i m}, a_{2^i m+1}, \ldots, a_{2^i(m+1)-1}\}$ and $|T_i(a_{2^i m})| = 2^i N/k$ hold for $m = 0, 1, \ldots, k/2^i - 1$. Similarly to the first round, the scheduler makes agents in $[a_{2^i m}]_i$ move in their territory $T_i(a_{2^i m})$ as many times as possible. At the end of the round, the scheduler makes agents move so that an agent in $[a_{2^{i+1} m}]_i$ and $[a_{2^{i+1} m+2^i}]_i$ $(m = 0, 1, \ldots, k/2^i - 1)$ moves to a node in $T_i(a_{2^{i+1} m+2^i})$ and $T_i(a_{2^{i+1} m})$, respectively. Then, the i-th round ends. We consider the total number of moves during the i-th round. Assume that, at the end of the i-th round, agent $a_x \in [a_{2^{i+1} m}]_i$ (resp., $a_x \in [a_{2^{i+1} m+2^i}]_i$) moves from node $u \in T_i(a_{2^{i+1} m})$ (resp., $u \in T_1(a_{2^{i+1} m+2^i})$) to node $v \in T_i(a_{2^{i+1} m+2^i})$ (resp., $v \in T_i(a_{2^{i+1} m})$). Since $2^i N/k - 1$ links around u are used at that time and at most $2^{i-1} N/k$ links are used before the i-th round, $2^{i-1} N/k - 1$ links are used during the i-th round. Consequently, agents in each equivalence class move at least $2^{i-1} N/k$ times in total during the i-th round. Since there exist $k/2^i$ equivalence classes, the total number of moves during the i-th round is at least $N/2$.

Lastly, we consider how many rounds are required to solve the group gossiping problem. Since a_0 and a_{k-1} belong to the same group, a_0 and a_{k-1} must belong to the same equivalence class. However, this occurs at the end of the $(\log k - 1)$-th round. This implies $\Omega(\log k)$ rounds are required. Since agents move $\Omega(N)$ times in total during each round, agents move $\Omega(N \log k)$ times in total to complete the group gossiping problem. ■

E. Asynchronous complete networks with sense of direction

In this subsection, we consider asynchronous complete networks with sense of direction. The sense of direction is given at each node as follows; nodes are denoted by $v_0, v_1, \ldots, v_{N-1}$, numbered clockwise in a ring, and for every i, j $(0 \leq i, j \leq N-1, i \neq j)$, the port number of link $e_{v_i v_j}$ on v_i is $(j - i) \bmod N$. Figure 1 shows a complete network with sense of direction of six nodes.

For this network, we can apply the algorithm proposed in the previous subsection. In addition, it is proved in [16] that the message complexity of an algorithm for the node leader election in message-passing system is $O(N)$. This implies the first phase of the algorithm requires only $O(N)$ total moves. Therefore we have the following theorem.

Theorem 10 The group gossiping problem is solved with $O(N)$ moves in asynchronous complete networks with sense of direction.

The following lower bound clearly holds since every node should be visited by at least one agent.

Theorem 11 Any algorithm for the group gossiping problem requires $\Omega(N)$ moves in asynchronous complete networks with sense of direction.

F. Asynchronous arbitrary networks

In this subsection, we consider asynchronous arbitrary networks without sense of direction. For a network G, a leader election in asynchronous message-passing systems can be solved by Gallager's algorithm for constructing a minimum spanning tree (MST) in G [12]. It is proved in [12] that the message complexity of the algorithm is $O(N \log N + M)$, where M is the number of links. In [25], it is proved that, in a network with k initiator nodes, the message complexity for constructing a MST is $O(N \log k + M)$ by using Gallager's algorithm. Therefore, from Theorem 1, leader election among k agents can be solved with $O(N \log k + M)$ moves.

An agent can travel the whole network with $O(N)$ moves by traversing the constructed MST. Thus, from Lemma 1 and $k \leq gk'$, the following theorem is obtained.

Theorem 12 The group gossiping problem is solved with $O(N \log k' + M + gN)$ moves in asynchronous arbitrary networks.

In the following, we consider the lower bound. Clearly, since every link should be traveled by at least one agent, any algorithm requires $\Omega(M)$ moves. In addition, since arbitrary networks include ring networks, any algorithm requires $\Omega(N \log k' + gN)$ moves from Theorem 5. Therefore, we have the following theorem.

Theorem 13 Any algorithm for the group gossiping problem requires $\Omega(N \log k' + M + gN)$ moves in asynchronous arbitrary networks.

V. Conclusion

In this paper, we introduced a concept of agent groups and formulated the group gossiping problem. We also showed the upper and lower bounds on the total number of moves to solve the group gossiping problem for various networks. As a future work, we would like to study some problems related to agent groups. For example, it is interesting to consider the

group gathering problem, which requires agents in the same group to meet at a single node.

REFERENCES

[1] Y. Afek and E. Gafni. Time and message bounds for election in synchronous and asynchronous complete networks. In *Proc. of the 4th Annual ACM Symposium on Principles of Distributed Computing*, pages 186–195, 1985.

[2] D. Baba, T. Izumi, F. Ooshita, H. Kakugawa, and T. Masuzawa. Linear time and space gathering of anonymous mobile agents in asynchronous trees. *Theoretical Computer Science*, 478:118–126, 2013.

[3] J. Cao and S. K. Das. *Mobile agents in networking and distributed computing*. Wiley-Interscience, 2012.

[4] J. Czyzowicz, D. Kowalski, and A. Pelc. Time versus space trade-offs for rendezvous in trees. *Distributed Computing*, 27:95–109, 2014.

[5] S. Das, P. Flocchini, A. Nayak, S. Kutten, and N. Santoro. Map construction of unknown graphs by multiple agents. *Theoretical Computer Science*, 385:34–48, 2007.

[6] A. Dessmark, P. Fraigniaud, D. Kowalski, and A. Pelc. Deterministic rendezvous in graphs. *Algorithmica*, 46:69–96, 2006.

[7] Y. Dieudonné and A. Pelc. Anonymous meeting in networks. In *Proc. of 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2013.

[8] Y. Dieudonné, A. Pelc, and V. Villain. How to meet asynchronously at polynomial cost. In *Proc. of 2013 ACM Symp. on Principles of Distributed Computing*, pages 92–99, 2013.

[9] P. Fraigniaud, L. Gasieniec, D. Kowalski, and A. Pelc. Collective tree exploration. *Networks*, 48:166–177, 2006.

[10] P. Fraigniaud and A. Pelc. Delays induce an exponential memory gap for rendezvous in trees. *ACM Transactions on Algorithms*, 9, 2013.

[11] G. N. Frederickson and N. A. Lynch. Electing a leader in a synchronous ring. *Journal of ACM*, 34(1):98–115, 1987.

[12] R. G. Gallager, P. A. Humblet, and P. M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Transactions on Programming Languages and Systems*, 5(1):66–77, 1983.

[13] M. Koucký. Universal traversal sequences with backtracking. *Journal of Computer and System Sciences*, 65:717–726, 2002.

[14] E. Kranakis, D. Krizanc, and E. Markou. *The mobile agent rendezvous problem in the ring*. Synthesis Lectures on Distributed Computing Theory, Lecture # 1. Morgan & Claypool Publishers, 2010.

[15] D. B. Lange and M. Oshima. Seven good reasons for mobile agents. *Communications of the ACM*, 42(3):88–89, 1999.

[16] M. C. Loui, T. A. Matsushita, and D. B. West. Election in a complete network with a sense of direction. *Information Processing Letters*, 22(4):185–187, 1986.

[17] T. Masuzawa and S. Tixeuil. Quiescence of self-stabilizing gossiping among mobile agents in graphs. *Theoretical Computer Science*, 411:1567–1582, 2010.

[18] F. Ooshita, S. Kawai, H. Kakugawa, and T. Masuzawa. Randomized gathering of mobile agents in anonymous unidirectional ring networks. *IEEE Transactions on Parallel and Distributed Systems*, 25:1289–1296, 2014.

[19] P. Panaite and A. Pelc. Exploring unknown undirected graphs. *Journal of Algorithms*, 33:281–295, 1999.

[20] A. Pelc. Deterministic rendezvous in networks: A comprehensive survey. *Networks*, 59:331–347, 2012.

[21] G. L. Peterson. An $o(n \log n)$ unidirectional algorithm for the circular extrema problem. *ACM Transactions on Programming Languages and Systems*, 4(4):758–762, 1982.

[22] O. Reingold. Undirected connectivity in log-space. *Journal of ACM*, 55, 2008.

[23] M. Shibata, S. Kawai, F. Ooshita, H. Kakugawa, and T. Masuzawa. Algorithms for partial gathering of mobile agents in asynchronous rings. In *Proc. of 16th Int'l Conf. on Principles of Distributed Systems*, pages 254–268, 2012.

[24] M. Shibata, F. Ooshita, H. Kakugawa, and T. Masuzawa. Move-optimal partial gathering of mobile agents in asynchronous trees. In *Proc. of 21st Int'l Colloquium on Structural Information and Communication Complexity*, 2014.

[25] T. Suzuki, T. Izumi, F. Ooshita, H. Kakugawa, and T. Masuzawa. Move-optimal gossiping among mobile agents. *Theoretical Computer Science*, 393:90–101, 2008.