社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2005－AL－103　(7)
2005／11／11

クラスタに基づく動的センサーネットワークアーキテクチャについて

内田次郎 [†]　イスラム A.K.M. ムジャヒドゥル [†]　片山喜章 [†]　陳慰 [††]　和田幸一 [†]

[†] 名古屋工業大学大学院情報工学専攻
[††] テネシー州立大学計算機学科

概要

本論文では，ノードの出現操作 (node-move-in) と消滅操作 (node-move-out) を許すような動的なセンサーネットワークに対してクラスタに基づくアーキテクチャの構成およびその維持について考える．提案するアーキテクチャのもとではブロードキャストが決定的に $O(p)$ ラウンドで完了する．ここで p はクラスタの数を表す．また，各ノードが 1 ホップの情報を維持して本アーキテクチャを構成している場合には node-move-in に対するランダマイズドアルゴリズムと node-move-out に対する決定的アルゴリズムを示す．実行時間はそれぞれ平均 $O(q)$ ラウンド，$O(|T|)$ ラウンドである．ここで q は出現するノードの隣接ノード数，T は消滅するノードを根とする本アーキテクチャの部分木を表す．また，各ノードが 1-ホップのすべての情報を維持しなくてもよい場合，node-move-in は平均 $O(\log q)$ ラウンドでできることを示す．このとき，node-move-out も 1-ホップの情報を維持する場合に比べてそれほど性能が悪くならないことも示される．

A Dynamic Cluster-based Architecture for Sensor Networks

Jiro Uchida,[†] Islam A.K.M. Muzahidul,[†] Yoshiaki Katayama,[†] Wei Chen,[††] and Koichi Wada[†]

[†]Nagoya Institute of Technology
Email: {jiro, islam}@phaser.elcom.nitech.ac.jp, {katayama, wada}@nitech.ac.jp,

[††]Tennessee State University
Email: WChen@tnstate.edu

Abstract

In this paper, we consider the construction and maintenance of a cluster-based architecture for a sensor network, with two atomic operations node-move-in and node-move-out which are performed by appearance and disappearance of a node. In our proposed architecture, a deterministic broadcasting can be done in $O(p)$ rounds, where p is the number of clusters. We present a randomized algorithm for a node-move-in, and a deterministic algorithm for a node-move-out operations, when nodes in the network are organized with total 1-hop data, which work in expected $O(q)$ rounds, and $O(|T|)$ rounds, respectively, where q is the number of neighbors in the network of the joining node, and T is a subtree of the architecture whose root is the leaving node. We also show that if nodes in the network are organized with partial 1-hop data, node-move-in can be done in expected $O(\log q)$ rounds and node-move-out can have a similar performance as the case of the total 1-hop data.

1 Introduction

A sensor network is a collection of sensor nodes, where each sensor node has a sensor array, a controling processor and a transimitter-receiver communication unit.

There is increasing interest in self-organizing multi-hop wireless sensor networks composed of a large number of autonomous nodes communicating via radio without any additional infrastructure. These nodes can be static or mobile, and they are usually constrained as for the critical resources, such as power and radio frequency band. A typical example is given by wireless sensor networks, where sensor nodes are usually irreplaceable, and become unusable after energy depletion or other failures. After the nodes (devices) of a sensor network are deployed physically, a *flat* network topology is formed in which a link exists between two nodes if they are in each others communication range. In such a flat network topology, there is no established structure on which the nodes could take efficient communication. *Clustering* is seen as the step to provide the flat sensor network topology with a *hierarchical* organization. The basic idea is that of breaking the network into physical proximity clusters which are smaller in scale and usually simpler to manage by the nodes called as *cluster head*. The subsequent *backbone* construction uses the clustering induced hierarchy to form a structure which mainly consists of cluster heads and provides the communication between the clusters. The structured network is functional in providing desirable properties such as minimizing communication overhead, choosing data aggregation points, increasing the probability of aggregating redundant data, and minimizing the overall power consumption [10]. Considering the mobility and scalability, we need the operations such as *nodes getting out* of and *nodes joining into* an existing network. Even for stationary nodes, when battery is low, it must get out and go to charge mode. Then, the charged nodes should join back to the network once again. Therefore, once a hierarchical clustering established, the maintenance of the cluster organization turns to be crucial in the presence of network topology changing.

Distributed clustering for a flat sensor network topology G has been investigated in many literatures. Most of the proposed protocols end up generating a clustering of G and a corresponding backbone in which the cluster heads forming a *dominating set (DS)* or an *independent set (IS)* of G [2–5, 7, 9]. A set is a *DS* of G if any node of G is either the node of *DS* or is the neighbor of a node of *DS*. A set is an *IS* of G if no two nodes of the set are the neighbors in G. In most of these distributed algorithms, the nodes need two hops knowledge, i.e., the knowledge of the neighbors and the neighbors' neighbors which need $O(n)$ time to get, to establish a clustering structure of G. It is known that finding a minimum *DS* (*MDS*) or a maximum *IS* of a graph is an *NP*-complete problem. Therefore, finding a clustering with the minimum number of clusters is also an *NP*-complete problem. Sometimes, a sensor network can be modeled by a *unit disk graph*, where an edge exists between two nodes iff the Euclidean distance of two nodes is at most 1. When G is a unit disk graph, a cluster structure can be formed by selecting a maximul *IS* (*MIS*) and then connect them to a backbone whose size is a constant times of *MDS* in $O(n)$ - $O(n^2)$ time [4, 7, 9]. In [3], a randomized algorithm is presented to compute an asymptotically optimal *MDS* (i.e., it finds a set of cluster heads but not connect them to form a cluster structure) in polylogarithmic time. This algorithm computes clustering but does not make a backbone.

The comparative performance evaluation of these above algorithms for clustering and backbone formation is shown in [10].

Although many efforts have been made for establishment of a hierarchical clustering on a sensor network, the research for the maintenance of the cluster organization under the similar scenario is seldom seen. This paper puts emphasis on the maintenance of the clustering structure of the sensor network. In this paper we consider a sensor network in which the network topology dynamically changes. We propose a novel cluster structure on which two operations *node-move-in* and *node-move-out* are defined for maintaining the cluster organization. Our work is based on the following radio network model [1]: each node has a distinct ID, nodes transmit or receive message in each synchronized round, and all nodes use a single radio channel without collision detection capability. In our distributed clustering, the nodes of a flat network G are grouped into disjoint clusters, and the backbone is a tree consisting of cluster heads and gateway nodes (gateway nodes are used to connect the heads). Furthermore, the backbone and the clusters are combined into a tree which we call as *cluster-based network* of G, denoted as $CNet(G)$. Let n be the number of the nodes in G and p be the number of the clusters in our clustering. A $CNet(G)$ has several novel properties: (1) the backbone consists of at most $2p-1$ nodes; (2) p is not larger than the smallest number of disjoint complete subgraphs of G, and so when G is a dense graph, $p \ll n$; (3) a broadcasting on G can be executed via the backbone in $O(p)$ time (it needs $\Omega(n)$ time on a flat network [8]); and (4) if G is a unit disk graph, $p \leq 5 \times |MDS|$.

In our clustering, when a $CNet(G)$ is established, each node has only one hop knowledge (i.e., each node knows their neighbors in the backbone, $CNet(G)$ and G, respectively). We will show that a $CNet(G)$ can be established either in a static way which means that all topological information are gathered somewhere and the problem is solved there, or in a dynamic way which means that each node solves the problem locally without gathering all information, in $O(n)$ time or in $O(|E|)$ time, respectively. The operations node-move-in and node-move-out maintain the cluster structure for G with the same properties when a node gets out of or joins into G. We will show algorithms for these operations. Note that proofs of our lemmas and theorems can be seen in [11].

Sometimes, nodes need only to know partial one hop knowledge (nodes don't need to know all the neighbors in G, but their neighbors in the backbone and $CNet(G)$). We will also show algorithms for maintaining such a structure. Our results are summarized in Table 1. The emphasis of our algorithms is to keep the clustering structure of the sensor network. The algorithms to make the structure follows on it. Though you may use a previous algorithm to make the clustering, you should check what information nodes have. In a word, our clustering is not worse than a previous results, and the main point of this paper is to keep the structure but to make it.

Table 1: Our results

maintained information	operation	completion time
1-hop, partial 1-hop	broadcast	$O(p)$
1-hop	node-move-in	expected $O(q)$
	node-move-out	$O(\|T\|)$
partial 1-hop	node-move-in	expected $O(\log q)$
	node-move-out	expected $O(\|T\| + r.\ log\Delta)$
		$O(\|T\| + p + t)$

p : the number of the clusters in $CNet(G)$.
q : the number of the move-in node's neighbors in G.
T : the subtree of $CNet(G)$ whose root is the leaving node.
H : the subtree obtained from $CNet(G)\backslash T$.
r : the number of border nodes in T that are connected with nodes in H.
Δ: the number of neighbors in H of a node in T.
t : the number of nodes that are 3-hop distance from the border nodes in T.

2 A Cluster-based Architecture

2.1 Definitions

We give definitions for some technical terms which will be used throughout the paper. Let $G = (V, E)$ be a directed graph.

Bi-directional graph: A graph G is a bi-directional graph if there is an edge from node u to node v, then there is an edge from v to u.
Strongly connected component: A directed graph, in which there exists at least one path from u to v for any two distinct nodes u and v, is said to be strongly connected. A strongly connected component C of a directed graph G is a subgraph $C = (V', E')(V' \subseteq V$ and $E' \subseteq E)$ which satisfies that no node of G can be added to C such that C is strongly connected.
Induced subgraph: The graph $H = (U, F)$, where $U \subseteq V$ and F is the set of all edges in G with both ends in U, is called the subgraph of G induced by U, denoted by $G[U]$.
In-neighbors: Node u is an in-neighbor of node v, if there is a directed edge from node u to node v.
Independent set: An independent set of G is a set $U \subseteq V$ in which no pair of nodes are adjacent in G.
Maximal independent set (MIS): An independent set I of nodes in a graph G such that no more nodes can be added and it still be an independent set.
Dominating set: A set $D(\subseteq V)$ of nodes is a dominating set of G if any node in G is either in D or the neighbor of a node in D.
Tree: A connected bi-directional graph such that it is acyclic when all bi-directional edges are replaced with undirected ones.
Unit disk graph: A graph G is a unit disk graph if for each edge $(u, v) \in E$, the Euclidean distance between u and v is at most 1.

Since a bi-directional graph can be treated as an undirected graph, in this paper, our networks are represented by undirected graphs, and the figure of a graph and technical terms also follow it (e.g. strongly connected component $\rightarrow$ connected component).

2.2 Model of Sensor Networks

The model of a sensor network G in this paper is as follows:

- Nodes repeat transmissions and receptions in synchronized fixed intervals, called *rounds*. In each round, each node acts as either a transmitter or a receiver.
- A node acting as a receiver in a given round gets a message iff exactly one of its neighbors transmits in this round. When more than one neighbor transmits simultaneously in a given round, collision occurs and none of the messages is received in this round. A node can not notice the occurrence of a collision, i.e., there is no collision detection in the network.
- Each node knows its ID, which is distinct for every node.

2.3 A Cluster-based Sensor Network and Its Properties

Let $G = (V, E)$ be a connected bi-directional graph. *A cluster* of G is a star subgraph of G, where one node, called *a cluster head*, has an edge to each other node called as *cluster member*, and no edge exists between any two cluster members in the cluster. *A clustering* of G is to partition G into node disjoint clusters. The union of the clusters produced by a clustering of G is denoted as $C(G) = (V, E_C)$, where the edges of E_C are those between the cluster heads and their members. In order to minimize the number of clusters, our clustering does not allow two cluster heads to be neighbors with each other. In other words, the set of the cluster heads in our clustering is a maximal independent set in G. We consider a *MIS* such that a graph induced by cluster heads ($\in$ *MIS*) and the nodes with two or more neighboring cluster heads is connected. Any two cluster heads are jointed through one special cluster member called *gateway node* which is in an intersection of neighbors in G of two cluster heads.

Clustering provides a flat graph G to a hierarchical organization. *A backbone* of G is a connected subgraph of G formed by only the cluster heads and the gateway nodes, where a gateway node must connect two or more cluster heads. Since heads can not be neighbors with each other, any edge in a backbone must be formed between a cluster head and a gateway node. As G is a connected bi-directional graph, a backbone must exist. *A backbone tree* of G, denoted as $BT(G) = (V_{BT}, E_{BT})$, is defined to be a spanning tree of a backbone of G (see Fig.1).

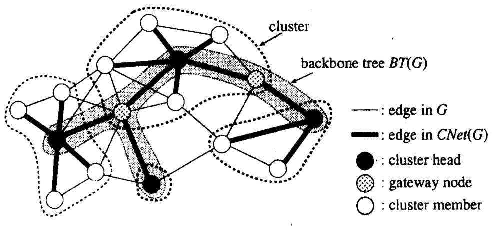

Figure 1: G, $BT(G)$, $CNet(G)$

Backbone tree $BT(G)$ can be considered as a communication highway on G. To see this, let u and v be two cluster members, and h_u and h_v be their cluster heads, respectively. If u wants to send message to v, u first sends the message to its cluster head h_u. And h_u sends the message to h_v via $BT(G)$, then h_v sends the message to its cluster member v. A transmission between a cluster head and its members is called *local transmission* and a transmission between cluster heads is called *backbone transmission*.

Now we use the backbone tree to connect the clusters for forming a structured network on G. *A cluster-based network* of $G = (V, E)$ is a rooted tree $CNet(G) = (V, E_{BT} \cup E_C)$ with one cluster head as a root, where the edges of E_{BT} come from the backbone tree and the edges of E_C come from all the clusters (see Fig 1).. Since gateway nodes are also cluster members,, we call the cluster members which are not gateway nodes as *pure cluster members*. In $CNet(G)$ a pure cluster member connects only with its cluster head.

Lemma 1. *Let G be a bidirectional graph and $BT(G)$ be a backbone tree of G. If $BT(G)$ has p cluster heads, then it has at most $2p - 1$ nodes.*

Lemma 2. *Let G be a bidirectional graph and $CNet(G)$ be a cluster-based network of G. The $CNet(G)$ is a spanning tree of G.*

Lemma 3. *Let G be a bidirectional graph and p_G be the smallest number of disjoint complete subgraphs of G. The number of the clusters in $CNet(G)$ is at most p_G.*

Lemma 4. *Let $G = (V, E)$ be a unit disk graph, and $MDS(G)$ be the minimum dominating set of G. The number of the clusters in a $CNet(G)$ is not larger than $5 \times |MDS(G)|$.*

In the following sections, we will show how a flat graph G can self-organize and self-maintain itself into a cluster-based network $CNet(G)$. Before we discuss the algorithms, we first declare the data structure for $CNet(G)$ clearly.

A $CNet(G)$ has two level structures: a set of clusters, and a backbone tree which is used to connect the clusters. Each node v in G maintains the information described below:

- For each node v in $CNet(G)$, it keeps a value $v.stat$ to denote v's status as a cluster head, a gateway node or a pure cluster member.
- For each node v in G, it keeps a node $v.prt$ and a node set $v.chd$ to denote who are its parent and children in $CNet(G)$, respectively. For root r, $r.prt = \perp$, and for each pure cluster member m, $m.chd = \emptyset$.
- For each node v in G, it keeps a node set $v.oneigh$ to denote the neighbors of v in G except $v.prt$ and $v.chd$.
- For each node v in the backbone tree $BT(G)$ (i.e., v is either a cluster head or a gateway node), it keeps a set $v.bneigh$ to denote who are its neighbors in the backbone tree.
- For each node v in G, it keeps status of its known neighbors in G.

We call above information as *total 1-hop data*. When the information are maintained for each node v in G, it is called that G *is organized* with total 1-hop data.

Hereafter we use $v.neigh$ as a node set which represents neighbors of v in G, (i.e., $v.neigh = \{v.prt\} \cup v.chd \cup v.oneigh$).

We define two operations *node-move-in* and *node-move-out* on a $CNet(G)$.

- node-move-in: a new node v announces itself by sending a message to join the existing $CNet(G)$ and the network re-organizes itself to a new $CNet(G')$, where G' is the graph obtained by adding v to G.
- node-move-out: a node v of $CNet(G)$ announces itself by sending a message to leave the existing $CNet(G)$, and the network re-organizes itself to a new $CNet(G')$, where G' is the graph obtained by removing v from G.

Theorem 1. *A $CNet(G)$ can be formed statically and dynamically in $O(n)$ and $O(\sum_{i=1}^{n} T(i))$ time, respectively, where $T(i)$ is the number of rounds which node-move-in operation requires for an i-node graph.*

Since our randomized node-move-in algorithm which runs in $O(q)$ expected time will be mentioned in 4.1, where q is the number of neighbors of added node in the existing network, a $CNet(G)$ can be formed dynamically in $O(|E|)$ expected time.

3 Broadcasting

In this section, we present our broadcasting algorithm using $CNet(G)$.

In our algorithm, we use a broadcasting technique shown in [1], and we call it as procedure *Eulerian*.

$Eulerian(H)$ performs a broadcast on a bi-directional graph H. A message called token starts from the source node, visits every node and turns to the source node. At the beginning, the token is in the source node. It then visits each node in H from the source node in depth-first order. When node v gets the token, it sends the token with the message and its ID to one of its neighbors which have not received the token yet. If v has no neighbor which has not been visited by the token yet, v returns the token to the node from which it got the token for the first time. The movement of the token forms an Eulerian cycle of H. It patrols every node in H and returns to the source node finally.

Lemma 5. [1] *Let $H = (V, E)$ be a bi-directional graph. If each node of H knows all its neighbors in H, procedure $Eulerian(H)$ completes broadcasting for H in $O(|V|)$ rounds.* □

Our broadcasting algorithm BroadcastALG in $CNet(G)$ can be found in [11].

In our work a broadcasting in $CNet(G)$ can be completed by performing *Eulerian* on $BT(G)$.

Theorem 2. *Let $CNet(G)$ be a cluster-based network of G, and p be the number of clusters in $CNet(G)$. A broadcasting on $CNet(G)$ can be done in $O(p)$ rounds.*

Theorem 3. *Let $CNet(G)$ be a cluster-based network of G, and p be the number of clusters in $CNet(G)$. The number of the nodes in G can be counted in $O(p)$ rounds.*

4 Constructing Cluster-based Sensor Network Dynamically

In this section we describe our two algorithms of node-move-in and node-move-out respectively on $CNet(G)$. The proposed cluster-based structure is always maintained after completing the operations of node-move-in or node-move-out. Though we assume that atomic operations cannot be performed simultaneously i.e., only one atomic operation can be performed at a time, it is possible to schedule for two or more simultaneous operations. When G is organized with total 1-hop data, our first randomized algorithm for a node-move-in operation and deterministic algorithm for a node-move-out operation work in $O(q)$ expected rounds and $O(|T|)$ rounds, respectively, where q is the number of neighbors in G of the move-in node and T is a subtree of $CNet(G)$ with the leaving node as root. When G is organized with partial 1-hop data, our randomized algorithms for a node-move-in and a node-move-out operations work in expected $O(\log q)$ rounds and $O(|T| + r \cdot \log \Delta)$ rounds, respectively, where r is the number of border nodes in T that has edges with nodes in subtree H, and Δ is the number of neighbors in H of a node in T, where H is the subtree of the architecture other than T. We also present a deterministic node-move-out algorithm which work in $O(|T| + p + t)$ rounds, where t is the number of nodes that are in 3-hop distance from the border nodes in T.

4.1 Node-move-in Algorithm

Let *new* be a node who wants to join a network $G = (V, E)$, and let $G' = (V \cup \{new\}, E \cup E_{new})$ in this subsection, where $E_{new} = \{(u, new) | u$ is in transmitting range of the node new, $u \in V\}$. Let $|E_{new}| = q$.

What should be performed by node-move-in is to decide the status of *new* and update the information which the neighbors of *new* in G' have.

To decide the status of *new*, *new* needs to know the status of its neighbors. If there exist cluster heads in the neighbors of *new* in G', *new* selects one to be it's head and itself becomes a pure cluster member. Else if there are gateway nodes in its neighbors, *new* selects one of the gateway nodes as its gateway and becomes the cluster head of a new cluster. If there are no cluster heads and no gateway nodes in its neighbors, *new* becomes a cluster head and sets one neighboring pure cluster member to be the gateway of itself. Based on the status of *new* has decided, the neighbors of *new* update their information. In no matter which cases, the process affects only 2-hop neighbors of new in G'.

In order to determine the status of *new* and update the information, it is sufficient that the neighbors of *new* transmit their own IDs and status one by one. It can be done by numbering the neighbors of *new* from 1 to q and transmitting their information in order of the numbers.

Lemma 6. *The neighbors of new can be numbered from 1 to q in $O(q)$ expected rounds, where q is the number of neighbors of new in G'.*

Our node-move-in algorithm move-in-1 can be seen in [11].

Lemma 7. *Let $CNet(G)$ be a cluster-based network of G. When G is organized with total 1-hop data, after an execution of move-in-1 for a node new, G' is organized with total 1-hop data.* □

Theorem 4. *Let $CNet(G)$ be a cluster-based network of G, q be the number of neighbors of new in G', and p be the number of clusters in $CNet(G)$. When G is organized with total 1-hop data, node-move-in of new can be done in $O(q)$ expected rounds, and G' is organized with total 1-hop data.*

In the execution of move-in-1 only the neighbors of *new*, and neighbors of the neighbors of *new* in *CNet*(G) receive a message caused by node-move-in. Hence, node-move-in can be performed locally in only 2-hop neighbors of *new* without changing status of other nodes.

4.2 Node-move-out Algorithm

Here, we show our first node-move-out algorithm called move-out-1. Let *del* be a node who wants to leave from G and G' be the graph after *del* leaves, that is, $G' = G[V \setminus \{del\}]$ in this subsection ("\" is substraction operator of sets). We assume that both G and G' are connected. It is possible to judge whether G' is connected in $O(|T|)$ rounds, where $T = (V(T), E(T))$ is a subtree of *CNet*(G) with the leaving node *del* as the root (see Fig.2).

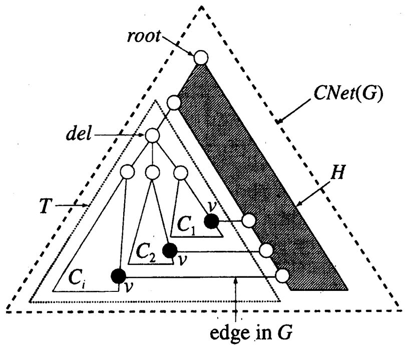

Figure 2: *CNet*(G), a subtree T and connected components in T

Our node-move-out algorithm is executed when *del* wishes to leave from the network. If *del* is a pure cluster member, it sends *I'mLeaving* message and simply leaves from the network. Otherwise, the node-move-out algorithm works as follows: First, we consider the case where *del* is not the root of *CNet*(G). The case where *del* is the root is described later. If *del* is a cluster head or a gateway node, *CNet*(G) is divided into two subtrees. One is the tree T with *del* as the root (not including the root in *CNet*(G)), and one is the tree H with the root of *CNet*(G) as the root. The algorithm move-out-1 removes *del* from T, and adds other nodes of T to H so that the resulted tree is *CNet*(G').

Let $C_i (i = 1, 2, \ldots)$ be the connected components of $G[V(T) \setminus \{del\}]$ (Fig.2). H always changes and grows larger each time when a node in T is added to H.

The edges in G between T and H are used in order to add the nodes of T to H. By using node-move-in operation described in the previous section, the nodes in T can be merged into H. Each node already knows its neighbors in G and their status in *CNet*(G), therefore, a node-move-in operation can be performed deterministically in $O(1)$ rounds. First, *del* calls *Eulerian*(T) to wake up each node of T. Whenever the waken node $v \in C_j$ has an edge connected with H, v moves to H by move-in-1, then v calls *Eulerian*(C_j) and each node in C_j moves to H following v by move-in-1 one by one. The above process will be repeated until all of the nodes in T other than *del* are moved to H.

Here we describe about an exception, when *del* is a root of *CNet*(G). If *del.bneigh* $\neq \emptyset$, electing a cluster head which is connected with *del* by a gateway node and setting it to a new root of *CNet*(G), our algorithm in the general case can be used. Otherwise *del* is the only cluster head in *CNet*(G), select one node in *del.neigh* becomes the new root of *CNet*(G'). The new root calls *Eulerian*$(G[V \setminus \{del\}])$ and a cluster-based network is constructed sequentially by repeating node-move-in for the node with token.

Our first node-move-out algorithm move-out-1 can be seen in [11].

Lemma 8. *Let CNet(G) be a cluster-based network of G. When G is organized with total 1-hop data, after an execution of move-out-1 for a node del, G′ is organized with total 1-hop data.* □

Theorem 5. *Let CNet(G) be a cluster-based network of G, p be the number of clusters in CNet(G) and T be the subtree of CNet(G) with the leaving node del as root. When G is organized with total 1-hop data, node-move-out of del can be done in O(|T|) rounds, and G′ is organized with total 1-hop data.*

Only nodes in T, their neighbors in G and neighbors of them in G receive messages during execution of move-out-1. Hence, node-move-out can be performed locally without changing other part of *CNet*(G).

4.3 Trade-off of Time and Information

In this subsection, we consider networks where node-move-in operations are performed more frequently than node-move-out operations. We reduce the amount of information each node maintains in order to reduce the running time of node-move-in algorithms. If node v does not maintain *oneigh*, the neighbor list of v in G, node-move-in operation will be faster. Given a graph G, when *v.stat*, *v.prt*, *v.chd*, *v.bneigh* (called partial 1-hop data) are maintained for each node v in G, we say that *G is organized* with partial 1-hop data.

4.3.1 Node-move-in Algorithm with Less Information

Let *new* be a node who wants to join a network, and let $G' = (V \cup \{new\}, E \cup E_{new})$ in this subsection, where $E_{new} = \{(u, new) | u$ is in transmitting range of the node new, $u \in V\}$. Let $|E_{new}| = q$. Operation node-move-in has been already described in subsection 4.1. First, a node *new* who wants to join a network confirms whether there is a cluster head in its neighbors in G' and if not any, whether there is a gateway node in them. Then *new* elects a leader of them and sets its status.

After status of *new* is decided, the neighbors of *new* update their information according to the status. In no matter which cases, the operation does not affect other than the neighbors of *new* and the neighbors of them (i.e., 2-hop neighbors) like move-in-1.

Here we show how to check the status of the neighbors of *new* in G'. One leader is first elected from the neighbors of *new* in G'. If the leader is not a cluster head, the leader and *new*'s neighbor cluster heads in G' send their IDs in one round. Assume that every node $v \in S$, $S = \{$the leader (not a cluster head)$\} \cup \{$neighbor cluster heads of new in $G'\}$, sends its ID at some round t. If *new* receives an ID at round t, it means $|S| = 1$, i.e., S contains only the leader, and there is no cluster head in neighbors of *new* in G'. Else if *new* receives no ID at round t, $|S| \geq 2$, i.e., there exists at least one cluster head. *new* can also check about gateway nodes similarly. Then, if there are neighbor cluster heads, *new* elects a leader from them, else if there are gateway nodes, *new* elects one leader from them, else *new* elects one leader from neighbor pure cluster member, and updates its status and then the information are maintained.

Our node-move-in algorithm move-in-2 and *SelectWinner* procedure which is used to elect a leader in move-in-2 are described in [11]

In *SelectWinner* procedure, all nodes receiving *AddMe* message toss a coin $\lceil t^j \rceil$ ($j = 1, 2, \ldots$) times to select a single *winner*, where t is going to be determined later. If "head" comes, the nodes are in the race, otherwise keep silent.

$O(\log q)$ times of tossing are expected until a winner is decided, where q is the number of the neighbors of *new* (see [11]).

Lemma 9. *Let CNet(G) be a cluster-based network of G. When G is organized with partial 1-hop data, after an execution of move-in-2 for a node new, G' is organized with partial 1-hop data.*

Theorem 6. *Let CNet(G) be a cluster-based network of G, q be the number of neighbors of new in G', and p be the number of clusters in CNet(G). When G is organized with partial 1-hop data, node-move-in of new can be done in $O(\log q)$ expected rounds, and G' is organized with total 1-hop data.*

A joining of a node can be locally performed like move-in-1.

4.3.2 Node-move-out Algorithm with Less Information

Here, we show two different algorithms for the node-move-out operation, when the nodes in the network have partial 1-hop information. First, we describe our randomized move-out-2R algorithm for the operation. The deterministic algorithm move-out-2D is described later in this subsection. Our move-out-2R is very similar to move-out-1 with few differences. Since the nodes know their neighbors in $CNet(G)$ only, even after $Eulerian(T)$ is performed a node in T do not know whether it has any edge (or edges) with H or not. Therefore, when $Eulerian(G[V] \setminus V(H) \setminus \{del\})$ is called, in order to find an edge with H, the waken node u in T sends $JoinMe$ message. Upon receiving the message, nodes in H that have received the message and the leader (the node that has sent the wake up call in previous round acts as a leader for next two rounds) sends their ids and status to u. If u receives message in current round, u knows that it does not have any neighbor in H, therefore continues $Eulerian(G[V] \setminus V(H) \setminus \{del\})$ to find an edge with H. Otherwise, u along with nodes in H that have received $JoinMe$ message from u invoke move-in-2, where u acts as *new*. Once a node $v \in C_j$ moves to H, it calls $Eulerian(C_j)$ and each node in C_j, moves to H by the last step of move-in-1 one by one, except the border nodes in T that follow u which is mentioned above.

Lemma 10. *Let CNet(G) be a cluster-based network of G. When G is organized partial 1-hop data, after an execution of move-out-2R for a node del, G' is organized with partial 1-hop data.* □

Theorem 7. *Let CNet(G) be a cluster-based network of G, p be the number of clusters in CNet(G) and T be the subtree of CNet(G) with the leaving node del as root. When G is organized with partial 1-hop data, node-move-out of del can be done in expected $O(|T| + r \cdot \log \Delta)$ rounds, and G' is organized with partial 1-hop data, where r is the number of border nodes in T that has edges with pure cluster member nodes in subtree H of CNet(G), and Δ is the number of neighbors in H of a node in T.*

Now, we describe our deterministic node-move-out algorithm move-out-2D. Our move-out-2D algorithm works as follows: first *del* calls $Eulerian(T)$, by which nodes in H that are connected with the border nodes in T know their position. Next, *del* calls $Eulerian(BT(G))$. During this process, if a cluster head and (or) a gateway node have received any message during $Eulerian(BT(G))$ they transmit their ids and status, else each cluster head node in H when receives the token during $Eulerian(BT(G))$ asks its members to inform their ids and status one by one, where a node transmits iff it has received a message(s) from at least one node in T during $Eulerian(T)$, otherwise keeps silent. Therefore, after finishing $Eulerian(BT(G))$ all nodes in T that have edges with H can learn about all of their neighbors in H and can move to H easily when it receives the wake up call during $Eulerian(G[V] \setminus V(H) \setminus \{del\})$ later. Once a node

$u \in C_j$ moves to H, it calls $Eulerian(C_j)$ and each node in C_j, moves to H by the last step of move-in-1 one by one.

Theorem 8. *Let CNet(G) be a cluster-based network of G, p be the number of clusters in CNet(G) and T be the subtree of CNet(G) with the leaving node del as root. When G is organized with partial 1-hop data, node-move-out of del can be done in* $O(|T| + p + t)$ *rounds, and G′ is organized with partial 1-hop data, where t is the number of nodes in H that are 3-hop distance from the border nodes in T.*

5 Conclusions

In this paper we have proposed a new architecture of sensor networks in which broadcasting can be done in $O(p)$ rounds. In order to support dynamic changes of the architecture we used two operations node-move-in and node-move-out. To add a new node, we have presented two different randomized algorithms working in expected rounds $O(q)$ and $O(\log q)$. To remove a node, our algorithms work in rounds $O(|T|)$, and in expected rounds $O(|T|+r \cdot \log \Delta)$ and in rounds $O(|T|+p+t)$, depending on the network information available to the nodes. The simulation will be done in our final version.

Acknowledgement

This work is supported in part by grant ARO W911NF-04-2-0049, U.S.A., the Japan Society for the Promotion of Science, Grant-in-Aid for Scientific Research(C), 17500036, and for Young Scientists (B), 16700010, 2004 and the Telecommunication Advancement Foundation in 2005.

References

[1] B.S. Chlebus, L. Gąsieniec, A.M. Gibbons, A. Pelc, and W. Rytter. Deterministic broadcasting in ad hoc radio networks. *Distributed Computing 15*, pages 27–38, 2002.

[2] D. Dubhashi, A .Mei, A. Panconesi, J. Radhakrishnan, A. Srinivasan, Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons, *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 717–724, 2003.

[3] F. Kuhn, T. Moscibroda, T. Wattenhofer, Initializing Newly Deployed Ad Hoc and Sensor Networks, *in Proceedings of 10 Annual International Conference on Mobile Computing and Networking* (MOBICOM), 2004.

[4] I. Chlamtac, A. Farago., A new approach to the design and analysis of peer-to-peer mobile networks, *Wireless Networks*, vol. 5, no. 3, pp. 149–156, 1999.

[5] J. Wu and H. Li, On calculating connected dominating set for efficient routing in ad hoc wireless networks, *Telecommunication Systems*, vol. 18, no. 1/3, pp. 13–36, 2001.

[6] K. Nakano and S. Olariu. Randomized initialization protocols for radio networks. *Handbook of wireless networks and mobile computing*, pp.195–218, 2002.

[7] P.-J. Wan, K. M. Alzoubi, and O. Frieder, Distributed construction of connected dominating sets in wireless ad networks, *ACM/Kluwer Mobile Networks and Applications*, MONET, bol. 9, no.2, pp. 141–149, 2004.

[8] R. Bar-Yehuda, O. Goldreich, and A. Itai, On the time-complexity of broadcast in radio networks: an exponential gap between determinism and randomization, *Journal of Computer and System Science*, no. 45, pp. 104–126, 1992.

[9] S. Basagni, Distributed clustering for ad hoc networks, *Proceedings of the 1999 International Symposium on Parallel Architectures, Algorithms, and Network*, pp. 310–315, 1999.

[10] S. Basagni, M. Mastrogiovanni, C. Petrioli, A performance comparison of protocols for clustering and backbone formation in large scale ad hoc networks, *The 1st Internatinal Conderence on Mobile Ad-hoc and Sensor Systems*, pp.70-79, 2005.

[11] Jiro Uchida, Islam A.K.M. Muzahidul, Yoshiaki Katayama, Wei Chen, and Koichi Wada, Construction and Maintenance of a Cluster-based Architecture for Sensor Networks, to appear in *Hawai Int'l Conference on System Sciences*, 2006.